Administrivia

• Reminder:
  – Homework 1 due on Thursday
  – in my inbox (midnight deadline)
  – need clarification, help? ask questions now

• There *may be* another lab class this Thursday (a short one: *room is in demand*)
  – check your email
  – and the course homepage
Last Time

- **Computer lab class:**
  - First time with SWI-Prolog
  - ... as will become apparent during this course, it’s a very convenient and powerful tool for expressing the rules of language
Last Time

• Important Concepts
  – English to logic
    • facts
      – e.g. Mary is a baseball fan \( \rightarrow \) baseball_fan(mary).
      – predicate: baseball_fan
      – argument: mary
      – e.g. Mary likes John \( \rightarrow \) likes(mary, john).
      – multiple arguments: express relations
    • inference rules
      – e.g. snoring presupposes sleeping \( \rightarrow \)
        sleeping(X) :- snoring(X).
      – logic variable: X
      – if: :-
      – note: basically expresses idea... snoring(X) \( \Rightarrow \) sleeping(X)
Last Time

• Important Concepts
  – **Prolog database**
    • the database represents a **scenario** or **possible world**
    • initially, the world is empty
    • we can **assert** and **retract** facts and rules
      – e.g. `- assert(baseball_fan(mary)).`
    • asserted facts and rules are true in that world
  – **Closed World Assumption**:
    – things that are not asserted or inferable are false (in this world)
  • can’t have negated facts (or head of rules) in the database
    – e.g. `- assert((\+ baseball_fan(mary))). is not allowed`
    – e.g. `- assert((\+ baseball_fan(X) :- hates(X,baseball))). is not allowed`
    – e.g. `- assert((baseball_fan(X) :- \+ hates(X,baseball))). is allowed`
Last Time

• Important Concepts
  – finally, we can evaluate logical queries with respect to this database
    • e.g. `?- baseball_fan(X).`
    • is true provided world contains one or more `baseball_fan/1` facts
    • is false otherwise
    • logic variable X is **bound** to the value produced by matching query to fact
    • **multiple matches** are possible: semicolon ; (disjunction)
    • query may also match against the **head** of a rule
    • e.g. `baseball_fan(X) :- loves(X,baseball).`
    • results in **subquery**: `?- loves(X,baseball).`
      – means to prove `baseball_fan(X) we have to in turn prove loves(X,baseball)`
Last Time

• Important Concepts
  – **negated queries** are ok
    • (though they return no answer other than Yes/No)
  – **query** ?- \+ baseball_fan(X). is *true if*
  – **subquery** ?- baseball_fan(X). *is not true*
  – ?- baseball_fan(X). would be not true for all possible worlds where there are no baseball fans
  – i.e. no **baseball_fan/1** facts
  – **and** we have no rules that could be used to conclude baseball_fan/1 is true from *logical inference*
  – e.g. no world like
    • baseball_fan(X) :- loves(X,baseball).
    • loves(john,baseball).
    • loves(john,football).
Computer Lab homework
  - asks you to write Prolog facts, rules and queries corresponding to a series of English sentences and questions

examples:
  - Mary is a student
  - Pete is a baseball fan
  - who is both a student and a baseball fan?
  - who is a baseball fan and not a student?
Larger Picture

- You’re translating English into logical meaning

Mary is a student who is a student?

\texttt{\textbf{student}(\textit{mary}).}

\texttt{?- student(\textbf{X}).}

to do this you have to be able to parse and assign meaning to the English input.
Larger Picture

• Goal: Formalize language so this can be done step by step

Mary is a student who is a student?

student(mary).
?- student(X).
Larger Picture

• In just a few more lectures, we’ll be able to do this...

• ... quick demo

to do this we have to be able to
(1) parse, and
(2) assign meaning to the English input

we’ll be developing the tools and techniques to do this
Today’s Topic

• We begin with...

• **Syntax** (or *grammar*)

• **motivation:**
  – to understand a sentence, we also have to be able to “*diagram it*”
  – i.e. know its constituents
  – subject
  – verb or predicate
  – object
A formal grammar enables us to logically break down a sentence into its constituent parts.

**X-bar phrase structure**
- **C2** = CP = S-bar (clause)
- **I2** = S (sentence)
- **VP** = Verb Phrase
- **V** = Verb
- **NP** = Noun Phrase
- **DET** = determiner
- **N1** = Noun (bar level 1)

Parsing: john is a student
LF (1):
A formal grammar enables us to logically break down a sentence into its constituent parts.

Parsing: John is a student

LF (1):

X-bar phrase structure

subject: [i2 [NP john] i1 ]

VP: is a student

copula: is

complement of VP: [NP [DET a][N1 student]]
(predicate NP)
A formal grammar enables us to logically break down a sentence into its constituent parts.

X-bar phrase structure
constituent labels
C2 = CP = S-bar (clause)
C = complementizer
I2 = S (sentence)
VP = Verb Phrase
V = Verb
NP = Noun Phrase
DET = determiner
NPt = NP trace

Syntactic Structure

- A formal grammar enables us to logically break down a sentence into its constituent parts.
A formal grammar enables us to logically break down a sentence into its constituent parts.

**X-bar phrase structure**
- Specifier of CP: \([_{\text{CP}} \quad _{\text{NP}} \text{ who}] \quad _{\text{C1}}\)
- Head of CP: C: auxiliary verb *is*
- Subject: \([_{\text{I2}} \quad _{\text{NP}} \text{ trace}] \quad _{\text{I1}}\)
- Subject is coindexed [1] with specifier of CP
- VP: \([_{\text{V trace}} \quad _{\text{a student}}]\)
- Verb (trace) is coindexed [2] with *is*
- Complement of VP: \([_{\text{NP}} \quad _{\text{DET a}}]\quad _{\text{N1 student}}]\)
Syntactic Structure

• We could but don’t have to specifically use X-bar phrase structure to diagram sentences
  – idea that all phrases have regular internal structure
  – \([_{\text{XP}} \text{specifier} \; [_{\text{X1}} \; [_{\text{X}} \text{head}] \text{complement}]]\)
  – \(X = \{C,I,V,N,A,P,..\}\)

  – so long as we’re able to identify (recover) configurations and (implied) grammatical positions
    • subject
    • object
    • verb (predicate)
Phrase Structure Rules

- **Simple rules:**
  - SBar → S
  - S → NP VP
  - VP → V NP
  - V → is
  - NP → DET N
  - NP → ProperNoun
  - ProperNoun → John
  - DET → a
  - N → student

Parsing: john is a student

LF (1):

```
C2
  C
    I2
      NP[1]
        I(AGR)[1]
          John
            I(AGR)[1]
              V[2]
                is
                  DET
                    N[1]
                      a
                        student
        V[2]
          Vt[2]
            NP[3]
```

subject

object
Phrase Structure Rules

- John is a \([_{\text{pred}} \text{student}]\)
- John \([_{\text{pred}} \text{likes}]\) Mary
- John is \([_{\text{pred}} \text{happy}]\)

- **which is the predicate?**
  - \(V\) (main verb: \(\text{likes}\))
  - \(V_{\text{aux}} \text{is}\) (copula carries little meaning)
  - complement of copula is the predicate

- **Note:**
  - gotta be careful
  - John is **the** student

- **Simple rules:**
  - SBar \(\rightarrow\) S
  - S \(\rightarrow\) NP VP
  - VP \(\rightarrow\) V(NP)
  - V \(\rightarrow\) is
  - NP \(\rightarrow\) DET N
  - NP \(\rightarrow\) ProperNoun
  - ProperNoun \(\rightarrow\) John
  - DET \(\rightarrow\) a
  - N \(\rightarrow\) student
Phrase Structure Rules

- **Rules:**
  - SBar → WhNoun Aux S
  - WhNoun → who
  - Aux → is
  - S → NPtrace VP
  - NPtrace → ε
  - VP → Vtrace NP
  - Vtrace → ε
  - NP → DET N
  - DET → a
  - N → student

plus associations by coindexation between traces and contentful items
Phrase Structure Rules

• To come...

– a very cool thing we’ll be using is that Prolog has a grammar rule system built into it
  • i.e. we can ask Prolog to do the diagramming for us
  • ... of course, we have to supply the phrase structure rules
Reading Assignment(s)

• for later this week
  – handout
  – Chapter 2: *Putting a Meaning Together from Pieces*
  – we will discuss it on Thursday

• are you comfortable diagramming sentences?
  – if not, grab any grammar/syntax book
  – or search the web:
    • http://en.wikipedia.org/wiki/Phrase_structure_rules

• Thursday
  – there will be a 15 minute quiz at the end of class