

Inference in rule-based systems. Knowledge Representation. Intro to Natural Language Processing.

Chapter 9, Sections 10.1-3, 22.2-3 & 22.5

Announcements

- Next Monday: midterm revision
- Next Wednesday: midterm 1
 - everything we've seen in lectures so far
 - open book, open notes, but no Google

Last time: FOL inference

- **Substitution** allows us to infer more specific cases from general cases
- Use **unification** identifies substitutions that make two expressions equal; the **most general unifier** makes the fewest commitments
- FOL **resolution** operates by unification on sentences in **clausal form**
- To convert to clausal form, remove implications and equivalences, **standardize apart** (rename vars to be different), **skolemize** (remove \exists), drop \forall , distribute *or* over *and*
- Entailment in FOL is **semidecidable**

Today

- KBs made of rules
- Knowledge representation and engineering
- Intro to Natural Language Processing

FOL resolution is powerful but hard

- But much knowledge can be expressed in ways that don't require the full power of resolution

- KBs made of **definite clauses** **Horn form**

– a definite clause has at most a single positive literal

$$\neg P(x) \vee \neg Q(y) \vee R(x,y)$$

$$\neg LoA(x) \vee \neg Animal(y) \vee \neg Kills(x,y)$$

– looks like an implication rule in clausal form

$$P(x) \wedge Q(y) \Rightarrow R(x,y)$$

$$LoA(x) \wedge Animal(y) \wedge Kills(x,y) \Rightarrow fail$$

Rule-based systems

- Facts:

$$P$$

- Rules:

$$P_1 \wedge P_2 \wedge \dots \wedge P_n \Rightarrow Q \text{ (Logic)}$$

If $P_1 P_2 \dots P_n$ Then Q (Rule-based system)

Q :- P_1, P_2, \dots, P_n (Prolog)

- P_i are called **antecedents** (or **body**)
- Q is called the **consequent** (or **head**)

Limitations

- Not all logical statements can be made into definite clauses
- No negation on the right side of implication:

$$P \Rightarrow \neg Q$$
- No disjunctions either:

$$P_1 \wedge P_2 \Rightarrow Q_1 \vee Q_2$$

$$Q_1 \vee Q_2$$
- These are not definite clauses!

Wed, Oct 1, 2008

COMP 131, Lecture 9

7

Forward chaining

- Start from the premises, and **apply** every implication rule whose premises are satisfied (i.e., unify with the ground facts in the KB), until conclusion is reached

Wed, Oct 1, 2008

COMP 131, Lecture 9

8

Forward chaining algorithm

```

function FOL-FC-ASK(KB, α) returns a substitution or false
  new ← {}
  repeat until new is empty
    for each sentence r in KB do
      (p1 ∧ ... ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
      for each θ such that (p1 ∧ ... ∧ pn)θ = (p'1 ∧ ... ∧ p'n)θ
        for some p'1, ..., p'n in KB
          q' ← SUBST(θ, q)
          if q' is not a renaming of a sentence already in KB or new then do
            add q' to new
            φ ← UNIFY(q', α)
            if φ is not fail then return φ
    add new to KB
  return false
    
```

if you can find a substitution that unifies premises with facts, add conclusion (with subst.) to KB

if conclusion q' unifies with alpha, return

writing $P\theta$ equivalent to $SUBST(\theta, P)$

Wed, Oct 1, 2008

COMP 131, Lecture 9

9

Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- Prove that Col. West is a criminal

Wed, Oct 1, 2008

COMP 131, Lecture 9

10

Example knowledge base

... it is a crime for an American to sell weapons to hostile nations:
 $American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 Nono ... has some missiles, i.e., $\exists x Owns(Nono,x) \wedge Missile(x)$
 $Owns(Nono,M_1)$ and $Missile(M_1)$
 ... all of its missiles were sold to it by Colonel West
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 Missiles are weapons:
 $Missile(x) \Rightarrow Weapon(x)$
 An enemy of America counts as "hostile":
 $Enemy(x,America) \Rightarrow Hostile(x)$
 West, who is American ...
 $American(West)$
 The country Nono, an enemy of America ...
 $Enemy(Nono,America)$

Wed, Oct 1, 2008

COMP 131, Lecture 9

11

Forward chaining example

American(West)
 Missile(M1)
 Owns(Nono,M1)
 Enemy(Nono,America)

$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 $Missile(x) \Rightarrow Weapon(x)$
 $Enemy(x,America) \Rightarrow Hostile(x)$

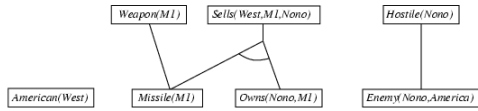
Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

12

Forward chaining example



$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 $Missile(x) \Rightarrow Weapon(x)$
 $Enemy(x,America) \Rightarrow Hostile(x)$

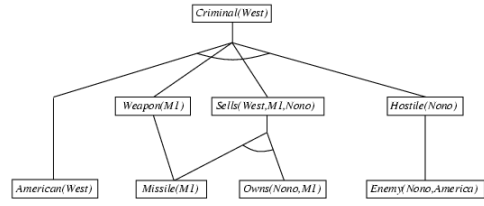
Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

13

Forward chaining example



$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 $Missile(x) \Rightarrow Weapon(x)$
 $Enemy(x,America) \Rightarrow Hostile(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

14

Properties of forward chaining

- Sound and complete for first-order definite clauses
- **Datalog** = first-order definite clauses + **no functions**
- FC terminates for Datalog in finite number of iterations
- May not terminate in general if conclusion is not entailed by KB
 - unavoidable: entailment with definite clauses is semidecidable

Wed, Oct 1, 2008

COMP 131, Lecture 9

15

Efficiency of forward chaining

Incremental forward chaining: no need to match a rule on iteration k if a premise wasn't added on iteration $k-1$
 \Rightarrow match each rule whose premise contains a newly added fact (positive literal)

Matching itself can be expensive:

Database indexing allows $O(1)$ retrieval of known facts

- e.g., query $Missile(x)$ retrieves $Missile(M_1)$

What about a rule like

- $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
- **most constrained variable** first (see CSP)

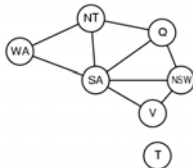
Forward chaining is widely used in **deductive databases** and **production rule** systems (expert systems)

Wed, Oct 1, 2008

COMP 131, Lecture 9

16

CSP and rule-based KB relationship



$Diff(wa,nt) \wedge Diff(wa,sa) \wedge Diff(nt,q) \wedge$
 $Diff(nt,sa) \wedge Diff(q,nsw) \wedge Diff(q,sa) \wedge$
 $Diff(nsw,v) \wedge Diff(nsw,sa) \wedge Diff(v,sa) \Rightarrow$
 $Colorable()$

$Diff(Red,Blue) \quad Diff(Red,Green)$
 $Diff(Green,Red) \quad Diff(Green,Blue)$
 $Diff(Blue,Red) \quad Diff(Blue,Green)$

- $Colorable()$ is inferred iff the CSP has a solution
- CSPs include 3SAT as a special case, hence matching is NP-hard

Wed, Oct 1, 2008

COMP 131, Lecture 9

17

Backward chaining

- Start from the conclusion C
- Push C onto stack; $Answer =$ empty substitution
- Repeat until stack empty or no actions available
 - Pop literal L from stack
 - Choose (with backup) a rule (or fact) whose consequent unifies with L
 - Push standardized apart antecedents (in order) onto stack
 - Apply unifier to entire stack
 - Add unifier to $Answer$
 - If no match, fail (backup to last choice)

Wed, Oct 1, 2008

COMP 131, Lecture 9

18

Backward chaining algorithm

```

function FOL-BC-ASK(KB, goals,  $\theta$ ) returns a set of substitutions
inputs: KB, a knowledge base
       goals, a list of conjuncts forming a query
        $\theta$ , the current substitution, initially the empty substitution {}
local variables: ans, a set of substitutions, initially empty
if goals is empty then return { $\theta$ }
 $q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(\text{goals}))$ 
for each r in KB where  $\text{STANDARDIZE-APART}(r) = (p_1 \wedge \dots \wedge p_n \Rightarrow q)$ 
  and  $\theta' \leftarrow \text{UNIFY}(q, q')$  succeeds
   $\text{ans} \leftarrow \text{FOL-BC-ASK}(\text{KB}, [p_1, \dots, p_n] \text{REST}(\text{goals}), \text{COMPOSE}(\theta, \theta')) \cup \text{ans}$ 
return ans
    
```

recursion with backups

subst. goal

unify goal with KB rule

push new premises onto stack
add new subst. into answer

$\text{SUBST}(\text{COMPOSE}(\theta_1, \theta_2), p) = \text{SUBST}(\theta_2, \text{SUBST}(\theta_1, p))$

Wed, Oct 1, 2008

COMP 131, Lecture 9

19

Backward chaining example

Criminal(West)

$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

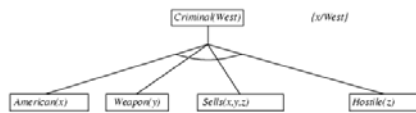
Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

20

Backward chaining example



$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

21

Backward chaining example



$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

22

Backward chaining example



$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

23

Backward chaining example



$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

$\text{Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

24

Backward chaining example



$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 $Missile(x) \Rightarrow Weapon(x)$
 $Enemy(x,America) \Rightarrow Hostile(x)$

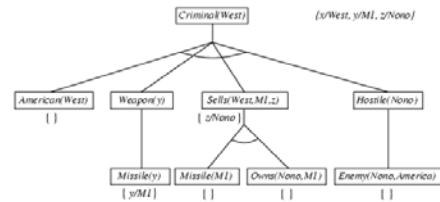
Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

25

Backward chaining example



$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 $Missile(x) \Rightarrow Weapon(x)$
 $Enemy(x,America) \Rightarrow Hostile(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

26

Backward chaining example



$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$
 $Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
 $Missile(x) \Rightarrow Weapon(x)$
 $Enemy(x,America) \Rightarrow Hostile(x)$

Is West a criminal?

Wed, Oct 1, 2008

COMP 131, Lecture 9

27

Backward chaining strategy

- Depth-first search for a proof
- Order matters
 - Rule order:
 - Try ground facts first
 - Then rules in given order
 - Antecedent order: left to right

Wed, Oct 1, 2008

COMP 131, Lecture 9

28

Properties of backward chaining

- Recursive DFS: space is linear in size of proof
- Incomplete due to possible infinite loops
 - fix by checking current goal against every goal on stack
- Widely used for **logic programming** (e.g., **Prolog**)

Wed, Oct 1, 2008

COMP 131, Lecture 9

29

Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

Wed, Oct 1, 2008

COMP 131, Lecture 9

30

Using logic to represent knowledge

- Categories and objects (see OOP)
 - $Basketball(x) \Rightarrow Ball(x)$
 - $Ball(x) \Rightarrow Round(x)$

taxonomies
inheritance
- Physical composition
 - $Biped(a) \Rightarrow \exists l_1, l_2, b . Leg(l_1) \wedge Leg(l_2) \wedge Body(b) \wedge PartOf(l_1, a) \wedge PartOf(l_2, a) \wedge PartOf(b, a) \dots$
- Measurements
 - $Diameter(Basketball12) = Inches(9.5)$
 - ordering measures without numerical values

$Exercise(e_1) \wedge Exercise(e_2) \wedge Wrote(Norvig, e_1) \wedge Wrote(Norvig, e_2) \Rightarrow GreaterThan(Difficulty(e_1), Difficulty(e_2))$

Wed, Oct 1, 2008

COMP 131, Lecture 9

31

FOL KB for Wumpus World

- Perception
 - $\forall t, b, g . Percept([Stench, b, g], t) \Rightarrow WumpusSmelt(t)$
 - $\forall t, s, b . Percept([s, b, Glitter], t) \Rightarrow AtGold(t)$
- Reflex
 - $\forall t . AtGold(t) \Rightarrow Action(Grab, t)$
- Reflex with state
 - $\forall t . AtGold(t) \wedge \neg HoldingGold(t) \Rightarrow Action(Grab, t)$
 - $HoldingGold(t)$ cannot be perceived directly; need to keep track of change

Wed, Oct 1, 2008

COMP 131, Lecture 9

32

Deducing hidden properties

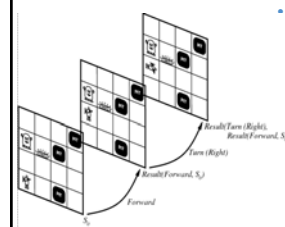
- Properties of squares:
 - $\forall x, t . At(Agent, x, t) \wedge Breezy(t) \Rightarrow Breezy(x)$
- Squares are breezy near a pit:
 - **Diagnostic rule** – infer cause from effect
 $\forall x . Breezy(x) \Rightarrow \exists y . Adjacent(y, x) \wedge Pit(y)$
 - **Causal rule** – infer effect from cause
 $\forall y . Pit(y) \Rightarrow [\forall x . Adjacent(y, x) \Rightarrow Breezy(x)]$

Wed, Oct 1, 2008

COMP 131, Lecture 9

33

Keeping track of change



- Some sentences are true in **situations** rather than eternally
 - $Holding(Gold, Now)$ not just $Holding(Gold)$

Situation calculus in FOL: add a situation argument to all predicates that are not eternal

- Situations are related by the function $Result(a, s)$ which returns a new situation, the result of doing a in s

$At(Agent, [1 2], Result(Forward, S_0))$

$Facing(Agent, Right, Result(Turn(Right), Result(Forward, S_0)))$

Wed, Oct 1, 2008

COMP 131, Lecture 9

34

Describing actions

- “Effect” axiom – changes due to action
 - $At(Agent, x, s) \wedge Action(Forward, s) \Rightarrow At(Agent, x+1, Result(Forward, s))$
 - not enough info in $Result(Forward, s)$ to reason about, say, gold
 - describes changes only
- “Frame” axiom – *non-changes* due to action
 - $HasArrow(Agent, s) \Rightarrow HasArrow(Agent, Result(Grab, s))$
- Frame Problem:
 - representation: avoid frame axioms
- Why?
 - quantification problem: true descriptions of actions require endless caveats (what if the gold is slippery, nailed down?..)
 - ramification problem: actions have many secondary consequences (wear and tear, ...)

Wed, Oct 1, 2008

COMP 131, Lecture 9

35

Describing actions: successor states

- **Successor-state axioms** solve the representational frame problem
- Each axiom is “about” a predicate, not an action
 - P true afterwards \Leftrightarrow [an action made P true \vee (P was already true \wedge no action made P false)]
- Gold-holding:
 - $\forall s, a . Holding(Gold, Result(a, s)) \Leftrightarrow [a = Grab \wedge AtGold(s) \vee Holding(Gold, s) \wedge \neg (a = Release)]$

Wed, Oct 1, 2008

COMP 131, Lecture 9

36

Making plans

- Initial condition in KB:
 - $At(Agent, [1, 1], S_0)$
 - $At(Gold, [1, 2], S_0)$
 - plus, all the axiomatic rules
- Query: in what situation will I be holding the gold?
 - $ASK(KB, \exists s. Holding(Gold, s))$
- Answer: substitution
 - $\{s\} Result(Grab, Result(Forward, S_0))$
 - go forward, then grab the gold

Wed, Oct 1, 2008

COMP 131, Lecture 9

37

Intro to Natural Language Processing (NLP)

- “Classical” view of natural language: it consists of sentences that are true or false (before 1953)
 - Ignore quirks and most of the beauty of language
 - Syntax/grammar
 - together with a **lexicon** specifies well-formed sentences of the language
 - “The cat jumped over the mat” is well-formed (grammatical)
 - “Cat the jumped mat the over” is not
 - Semantics
 - the meaning of sentences
 - Pragmatics
 - the intentions of the speaker
- logic useful for both
syntax and semantics

Wed, Oct 1, 2008

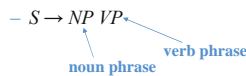
COMP 131, Lecture 9

38

Syntax: grammar

- Descriptive, not prescriptive
- Generative: produces sentences of a language

- Consists of rewrite rules



Wed, Oct 1, 2008

COMP 131, Lecture 9

39

A subset of English: ϵ_0

GRAMMAR	$S \rightarrow NP VP$	LEXICON
	$S Conjunction S$	
	$NP \rightarrow Pronoun$	
	$Name$	
	$Noun$	
	$Article Noun$	
	$Digit Digit$	
	$NP PP$	
	$NP RelClause$	
	$VP \rightarrow Verb$	
$VP NP$		
$VP Adjective$		
$VP PP$		
$VP Adverb$		
$PP \rightarrow Preposition NP$		
$RelClause \rightarrow that VP$		
	$Noun \rightarrow$	stench breeze glitter nothing agent wumpus pit pits gold ...
	$Verb \rightarrow$	is see smell shoot feel stinks go grab carry kill turn ...
	$Adjective \rightarrow$	right left dead smelly ...
	$Adverb \rightarrow$	here there ahead ...
	$Pronoun \rightarrow$	me you I it ...
	$Name \rightarrow$	John Mary Boston ...
	$Article \rightarrow$	the a an ...
	$Preposition \rightarrow$	to in on near ...
	$Conjunction \rightarrow$	and or but ...
	$Digit \rightarrow$	0 1 2 3 4 5 6 7 8 9

Wed, Oct 1, 2008

COMP 131, Lecture 9

40

Some sentences of ϵ_0

$S \rightarrow NP VP$	LEXICON
$S Conjunction S$	
$NP \rightarrow Pronoun$	
$Name$	
$Noun$	
$Article Noun$	
$Digit Digit$	
$NP PP$	
$NP RelClause$	
$VP \rightarrow Verb$	
$VP NP$	
$VP Adjective$	
$VP PP$	
$VP Adverb$	
$PP \rightarrow Preposition NP$	
$RelClause \rightarrow that VP$	
	• I shoot the wumpus
	• I feel breeze and I see glitter in 1 4
	• John is in the pit
	• You is wumpus here
	• I smell pit gold wumpus nothing
	• This grammar overgenerates
	• It also undergenerates
	– “I think the wumpus is smelly” is rejected

Wed, Oct 1, 2008

COMP 131, Lecture 9

41

Parsing: syntactic analysis

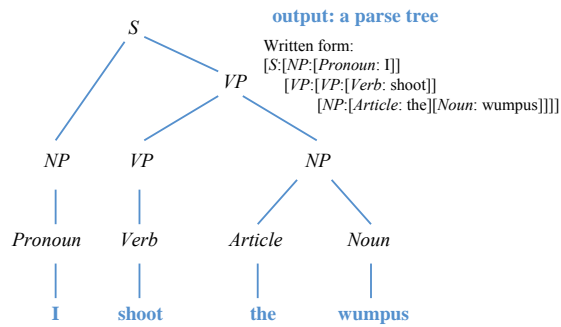
- Recovering the syntactic structure from a string (inverse problem)
- Represent sentence as an ordered list of words:
 - [“I” “shoot” “the” “wumpus”]

Wed, Oct 1, 2008

COMP 131, Lecture 9

42

Parsing example



Wed, Oct 1, 2008

COMP 131, Lecture 9

43

Logical grammars

- Magic connection between grammar and logic
 - grammar rules $S \rightarrow NP VP$ can be rewritten as

$$NP(s_1) \wedge VP(s_2) \Rightarrow S(Append(s_1, s_2))$$
 - lexical entries can be written as $Noun(["wumpus"])$
 - sentences can be rewritten as

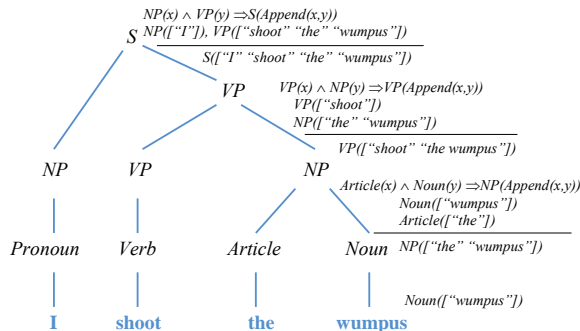
$$S(["I" "shoot" "the" "wumpus"])$$
- Parsing can be seen as inference:
 - ASK (KB, $S(["I" "shoot" "the" "wumpus"])$)
 - augment with labels to return the structure
 - ASK (KB, $S(x)$) generates new sentences

Wed, Oct 1, 2008

COMP 131, Lecture 9

44

Parsing: unify rules with ground facts



Wed, Oct 1, 2008

COMP 131, Lecture 9

45

Top-down vs. bottom-up

- **Forward chaining** results in **bottom-up** parsing: start with the leaves (words), unify with antecedents of parsing rules until predicate S (goal) is inferred
 - limitation: applies rules and generates subtrees that won't be part of the final structure
- **Backward chaining** results in **top-down** parsing: start with predicate S , find rule with consequent S , push its antecedents onto the stack, ..., proceed to leaves (words)
 - limitation: left recursion problem in $VP \rightarrow VP NP$ -type rules

Wed, Oct 1, 2008

COMP 131, Lecture 9

46

Summary

- Some important problems don't require the full power of resolution, if their KBs can be expressed as sets of **definite (Horn) clauses** (at most one positive literal)
- In that case, we can use **forward chaining** and **backward chaining** algorithms to prove conclusions (by finding the right substitution)
- We can represent **taxonomies** of **categories** and **objects** including properties and **inheritance** with FOL. You can also represent **measurements**, **situations** (using **situational calculus**), **actions** and **plans**. Solve the **frame problem** by using **successor-state axioms** to represent effects of actions
- Natural language is a system with **syntax**, **semantics** and **pragmatics**.
- **Parsing** is the problem of finding the syntactic structure of a string.
- Grammatical rules can be represented with FOL
- **Bottom-up parsing** is **forward chaining** and **top-down parsing** is **backward chaining**

Wed, Oct 1, 2008

COMP 131, Lecture 9

47

Final project ideas

- Write a theorem prover for a small but realistic domain
- Write an expert system (production rules) for a small but realistic domain
- Write a text processing system that does something useful
 - sorts your email
 - answers database queries posed in English

Wed, Oct 1, 2008

COMP 131, Lecture 9

48