

Natural Language Processing in Prolog Part II: Adding Semantics

CSCI 315: Artificial Intelligence

Simon D. Levy

Fall 2007

Beyond Acceptance

- We'd like to be able to *understand* sentences, not just accept them:

```
| ?- understand("the wumpus ate my shorts", Meaning).
```

```
Meaning = and(ate(wumpus, shorts), belong(shorts, me))
```

```
yes
```

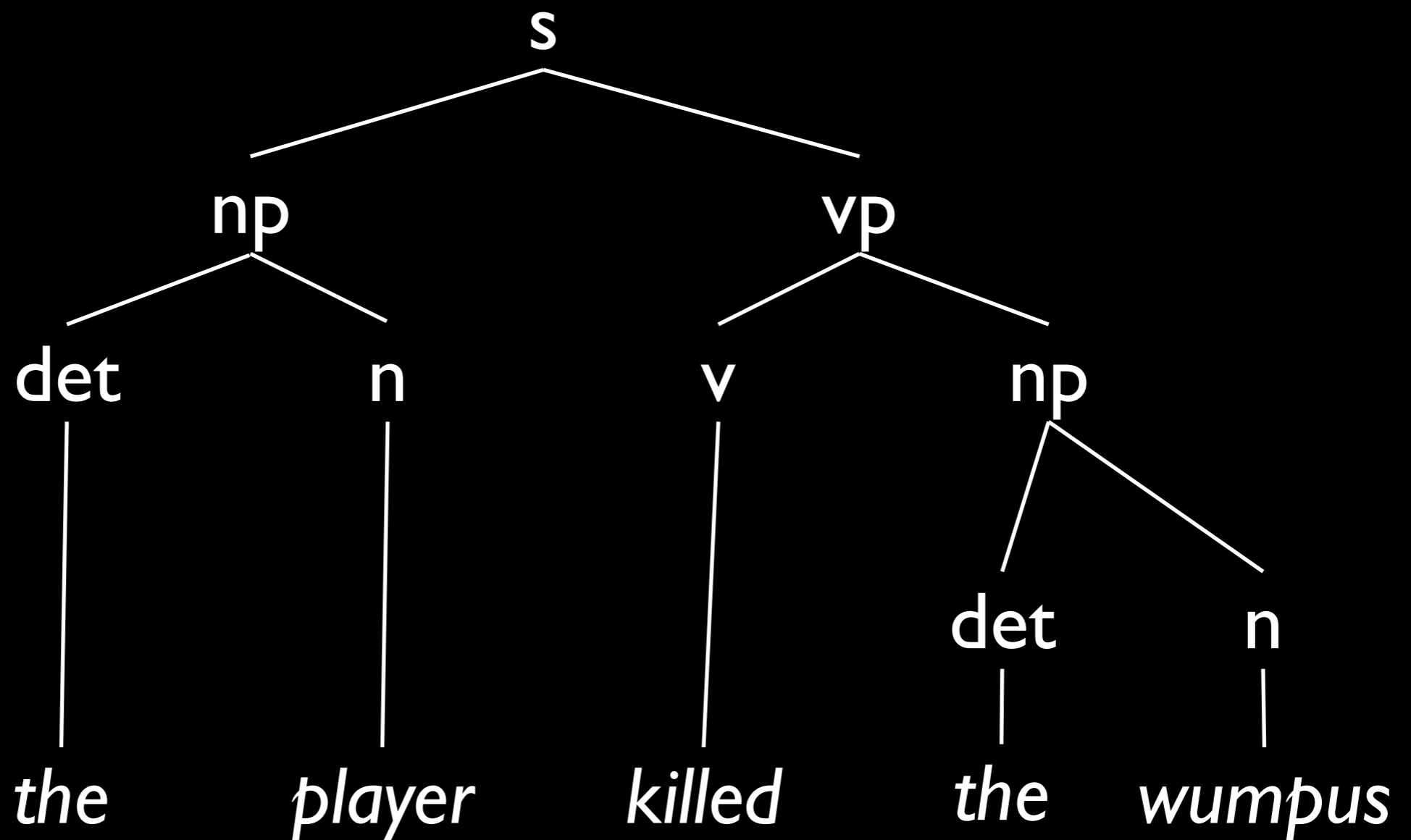
- Generation should also be semantically aware:

```
| ?- generate(smell(me, wumpus), S).
```

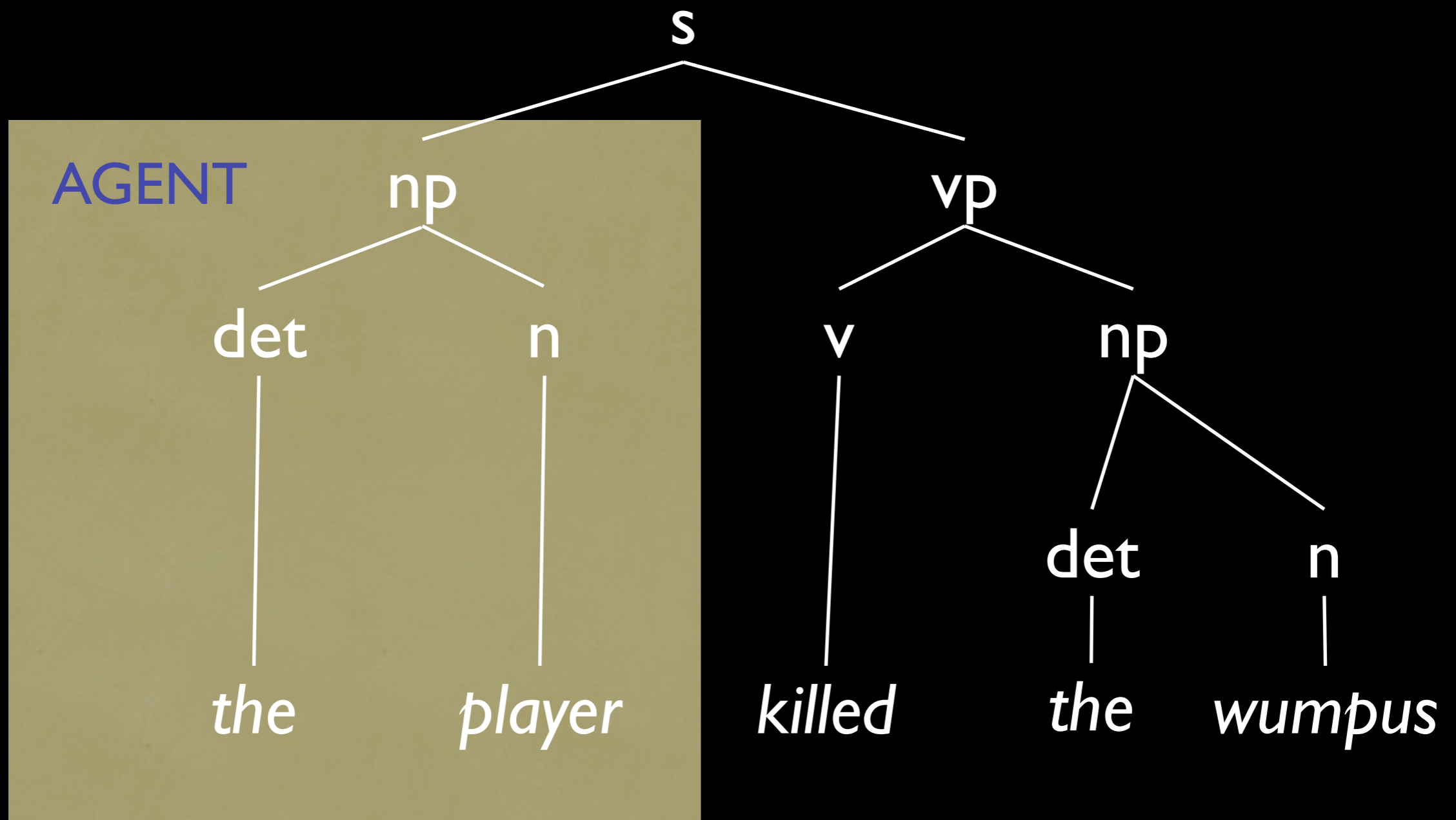
```
S = "I smell a wumpus"
```

```
yes
```

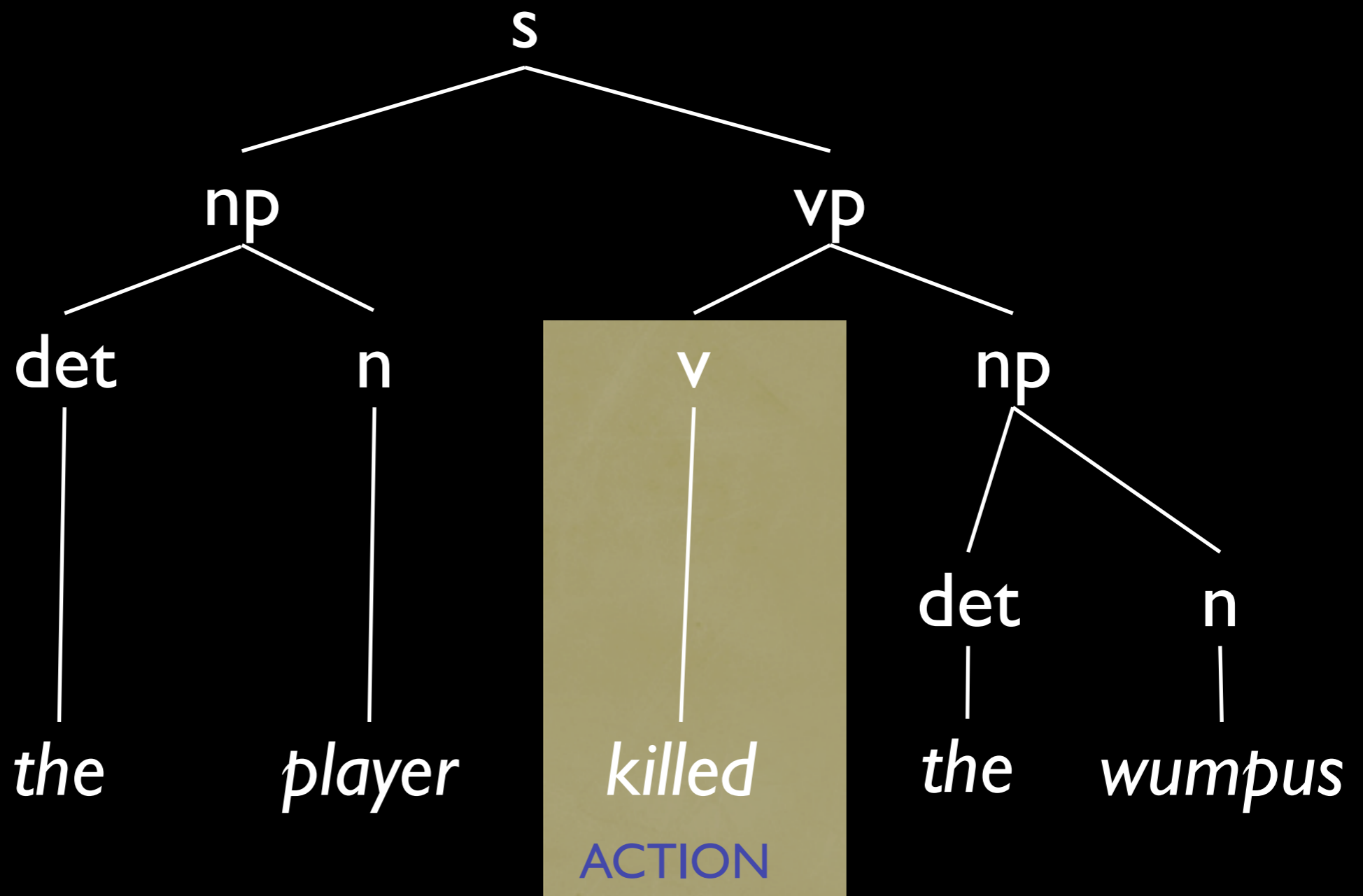
Parsing: Between Lists and Meanings



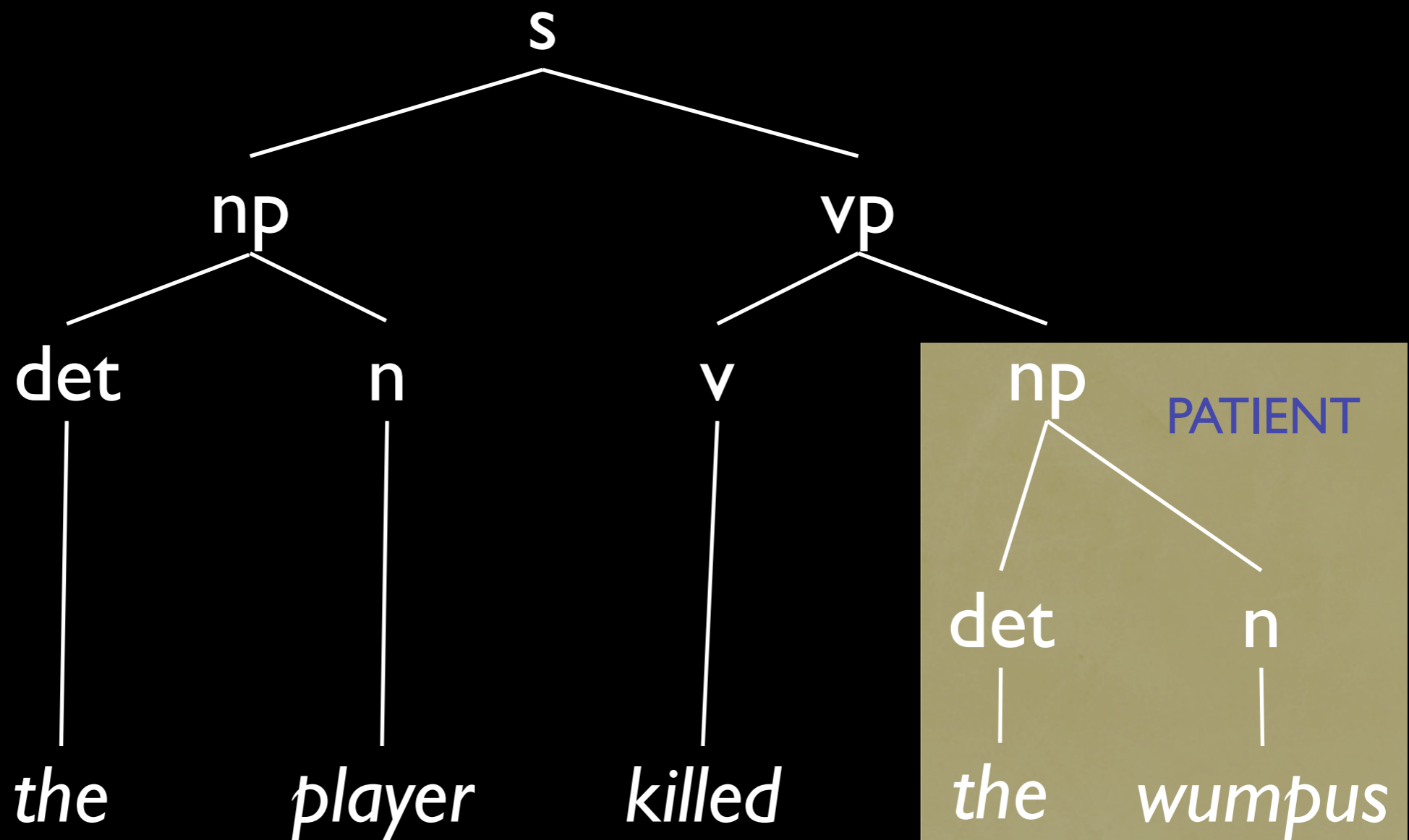
Parsing: Between Lists and Meanings



Parsing: Between Lists and Meanings



Parsing: Between Lists and Meanings



Parsing: Between Lists and Meanings

```
| ?- parse(P, [the, player, killed, the, wumpus]).
```

```
P = s(np(det(the), n(player)),
```

```
      vp(v(killed), np(det(the), n(wumpus))))
```

```
yes
```

Parsing: Basics

- Make a predicate for each grammar symbol,
on the left-hand side:

sentence(**s**(NP, VP)) -->

nounphrase(NP), verbphrase(VP) .

nounphrase(**np**(Det, N)) -->

determiner(Det), noun(N) .

verbphrase(**vp**(V, NP)) -->

verb(V), nounphrase(NP) .

Parsing: Basics

- Same for lexical rules:

determiner (det (the)) --> [the].

determiner (det (a)) --> [a].

noun (n (wumpus)) --> [wumpus].

noun (n (player)) --> [player].

verb (v (killed)) --> [killed].

Parsing: Basics

- Then `parse/2` is a helper for `sentence/3`:

```
parse(P, L) :- sentence(P, L, []).
```

```
| ?- parse(P,  
| [the,player,killed,the,wumpus]).
```

```
P = s(np(det(the),n(player)),  
      vp(v(killed),np(det(a),n(wumpus))))
```

yes

Okay to re-use symbols

$s(s(NP, VP)) \rightarrow np(NP), vp(VP)$.

$np(np(Det, N)) \rightarrow det(Det), n(N)$.

$vp(vp(V, NP)) \rightarrow v(V), np(NP)$.

$det(det(the)) \rightarrow [the]$.

$det(det(a)) \rightarrow [a]$.

$n(n(wumpus)) \rightarrow [wumpus]$.

$n(n(player)) \rightarrow [player]$.

$v(v(killed)) \rightarrow [killed]$.

From Parse to Meaning

```
understand(String, Meaning) :-  
    string2list(String, List),  
    parse(Parse, List),  
    parse2meaning(Parse, Meaning).
```

The Meaning of Life is [life]

```
parse2meaning(s(np(_, n(N1)), vp(v(V), np(_, n(N2))))),  
              event(action(V), agent(N1), patient(N2))).
```

The Meaning of Life is [life]

```
| ?- parse(P, [the,player,killed,the,wumpus]),  
      parse2meaning(P, M).
```

```
P = s(np(det(the),n(player)),  
      vp(v(killed),np(det(the),n(wumpus))))
```

```
M = event(action(killed),agent(player),patient(wumpus))
```

```
yes
```

Direct Parse from Words to Meaning

```
s(event(action(Action), agent(Agent), patient(Patient)))
```

```
--> np(Agent), v(Action), np(Patient).
```

```
np(Entity) --> det, n(Entity).
```

```
det --> [the].
```

```
det --> [a].
```

```
n(wumpus) --> [wumpus].
```

```
n(player) --> [player].
```

```
v(kill) --> [smells].
```

```
v(smell) --> [smells].
```

Direct Parse from Words to Meaning

```
| ?- parse(P, [the,player,kills,the,wumpus]).
```

```
P = event(action(kill),agent(player),patient(wumpus))
```

```
yes
```

Mixing Grammar and “Raw” Prolog

- Sometimes we want to add non-syntactic operations to our rules
- Curly-bracket notation does this:

```
noun(place, X) --> [X], {room(X)}.
```