

# Prolog programming: a do-it-yourself course for beginners

## Day 4

**Kristina Striegnitz**

Department of Computational Linguistics

Saarland University, Saarbrücken, Germany

`kris@coli.uni-sb.de`

`http://www.coli.uni-sb.de/~kris`

# Day 4: Definite Clause Grammars (1)

---

Today: Grammars as Prolog programs for recognizing natural language sentences.

Reader: Lecture 7 of *Learn Prolog Now!*

# Today's goal: a NL recognizer

---

A Prolog program for the following task:

Given

- a grammar specification,
- a list of words, and
- a syntactic category  $C$ ,

is the list of words a grammatical expression of category  $C$ ?

# Some examples

---

S → NP VP

category: S

NP → Det N

words: [the,nurse,whistles] → yes

NP → PN

category: NP

VP → Vi

words: [the,whiskey] → yes

VP → Vt NP

category: NP

Det → *the*

words: [drinks] → no

Det → *a*

category: VP

PN → *gogo*

words: [fights] → no

N → *nurse*

N → *whiskey*

Vi → *whistles*

Vt → *drinks*

# Strategy

---

For each syntactic category  $C$ , define a predicate  $c(\text{InList}, \text{OutList})$  which takes a list of words ( $\text{InList}$ ) as input, “bites off” a sequence of words corresponding to an expression of category  $C$  and returns the rest ( $\text{OutList}$ ).

# Examples

---

```
?- np([the,nurse,whistles],Out).  
Out = [whistles]
```

```
?- np([the,whiskey],Out).  
Out = []
```

```
?- vt([drinks,the,whiskey],Out).  
Out = [the,whiskey]
```

```
?- vp([nurse,whistles],Out).  
no
```

# Definite Clause Grammars (DCGs) — words

---

Let's start with single words:

```
n([bride|Out],Out).  
n([nurse|Out],Out).  
n([sword|Out],Out).  
det([the|Out],Out).  
pn([bill|Out],Out).  
vt([kills|Out],Out).  
:  
:
```

If the head of the input list is the word *bride*, then we have found a noun. Return the tail of the list.

# DCGs — complex categories

---

Now, we can build more complex categories:

```
np(In,Out) :- det(In,DetOut), n(DetOut,Out).
```

If we can bite a determiner off the list and then bite a noun of the list, then we have found an NP. Return what's left when biting off the determiner and the noun.

```
np(In,Out) :- pn(In,Out).
```

If we can bite a proper name off the list, then we have found an NP. Return what's left when biting off the proper name.

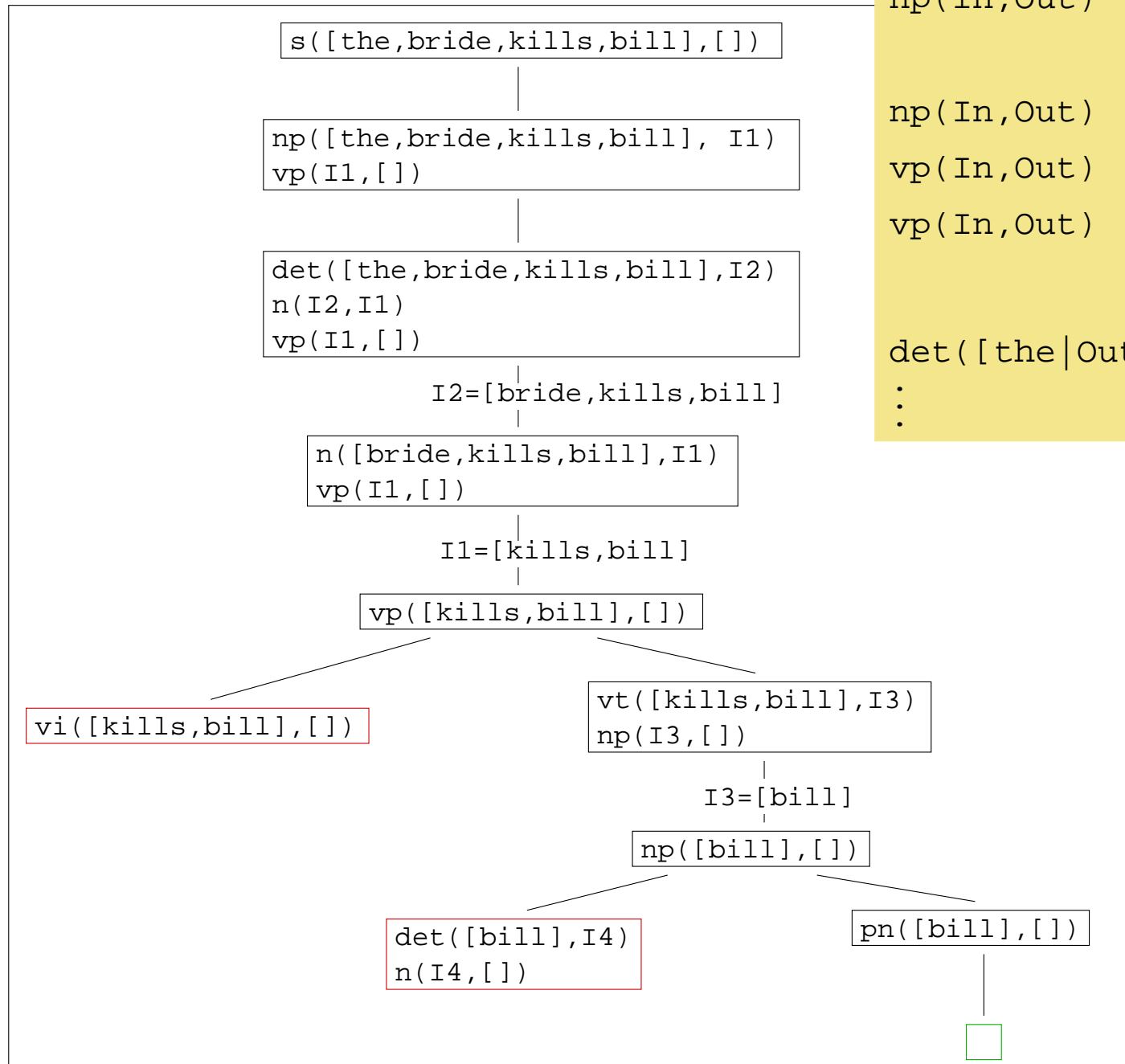
```
vp(In,Out) :- vi(In,Out).
```

```
vp(In,Out) :- vt(In,VtOut), np(VtOut,Out).
```

```
s(In,Out) :- np(In,NPOut), vp(NPOut,Out).
```



# How does this work?



```

s(In,Out) :- np(In,NPOut),
              vp(NPOut,Out).
np(In,Out) :- det(In,DetOut),
              n(DetOut,Out).
np(In,Out) :- pn(In,Out).
vp(In,Out) :- vi(In,Out).
vp(In,Out) :- vt(In,VtOut),
              np(VtOut,Out).

det([the|Out],Out).
:
:
:
  
```

# DCGs in Prolog

Prolog provides a simpler notation for specifying DCGs.

<code>s --&gt; np, vp.</code>	$\rightarrow$	<code>s(In,Out) :- np(In,NPOut), vp(NPOut,Out).</code>
<code>np --&gt; det, n.</code>	$\rightarrow$	<code>np(In,Out) :- det(In,DetOut), n(DetOut,Out).</code>
<code>np --&gt; pn.</code>	$\rightarrow$	<code>np(In,Out) :- pn(In,Out).</code>
<code>vp --&gt; vi.</code>	$\rightarrow$	<code>vp(In,Out) :- vi(In,Out).</code>
<code>vp --&gt; vt, np.</code>	$\rightarrow$	<code>vp(In,Out) :- vt(In,VtOut), np(VtOut,Out).</code>
<code>n --&gt; [bride].</code>	$\rightarrow$	<code>n([bride Out],Out).</code>
<code>det --&gt; [the].</code>	$\rightarrow$	<code>det([the Out],Out).</code>
<code>vi --&gt; [whistles].</code>	$\rightarrow$	<code>vi([whistles Out],Out).</code>
<code>vt --&gt; [kills].</code>	$\rightarrow$	<code>vt([kills Out],Out).</code>

Internally, Prolog uses this notation. Therefore: `?- s([the,nurse,whistles],[ ]).` to ask whether `[the,nurse,whistles]` is a sentence.

# Adding Pronouns

---

Here is our DCG:

```
s(In,Out) :- np(In,NPOut), vp(NPOut,Out).  
np(In,Out) :- pn(In,Out).  
np(In,Out) :- det(In,DetOut), n(DetOut,Out).  
vp(In,Out) :- vi(In,Out).  
vp(In,Out) :- vt(In,VtOut), np(VtOut,Out).  
  
n([bride|Out],Out).  
det([the|Out],Out).  
pn([bill|Out],Out).  
vt([kills|Out],Out).  
vi([whistles|Out],Out).
```

Imagine we want to add the pronouns  
*he, she, him, her...*

# Adding Pronouns — first try

```
s(In,Out) :- np(In,NPOut), vp(NPOut,Out).  
np(In,Out) :- pn(In,Out).  
np(In,Out) :- det(In,DetOut), n(DetOut,Out).  
vp(In,Out) :- vi(In,Out).  
vp(In,Out) :- vt(In,VtOut), np(VtOut,Out).
```

```
n([bride|Out],Out).  
det([the|Out],Out).  
pn([bill|Out],Out).  
vt([kills|Out],Out).  
vi([whistles|Out],Out).
```

```
?- s([she,kills,him],[ ]).
```

yes

```
?- s([him,kills,she],[ ]).
```

no

```
np(In,Out) :- pro(In,Out).
```

```
pro([he|Out],Out).  
pro([she|Out],Out).  
pro([him|Out],Out).  
pro([her|Out],Out).
```

⇒ Need to distinguish between subject and object pronouns.

# Adding pronouns — second try

---

We use an extra argument to mark whether an NP or a pronoun is in subject or in object position.

```
s(In, Out) :- np(subj, In, NPOut), vp(NPOut, Out).
```

```
vp(In, Out) :- vt(In, VtOut), np(obj, VtOut, Out).
```

```
np(CASE, In, Out) :- pro(CASE, In, Out).
```

```
pro(subj, [he|Out], Out).
```

```
pro(subj, [she|Out], Out).
```

```
pro(obj, [him|Out], Out).
```

```
pro(obj, [her|Out], Out).
```

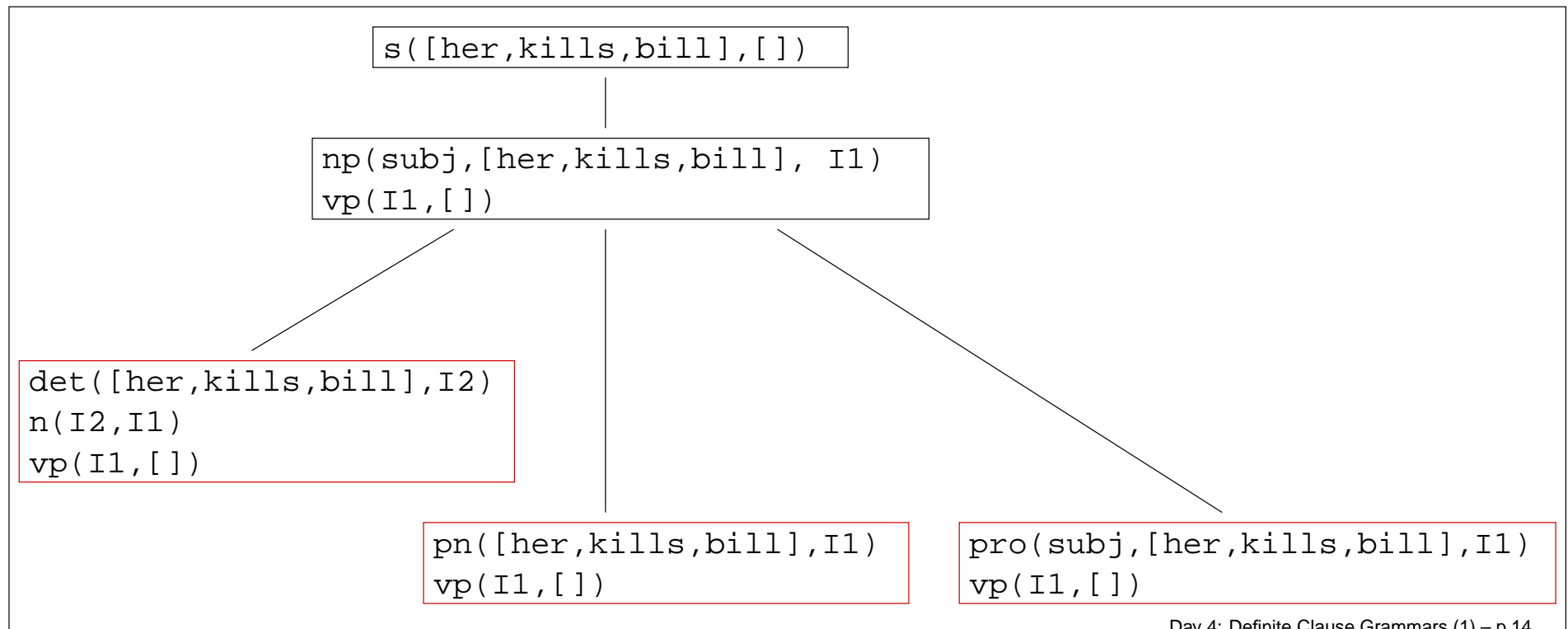
```
np(_, In, Out) :- pn(In, Out).
```

```
np(_, In, Out) :- det(In, DetOut), n(DetOut, Out).
```

# Example

```
s(In,Out) :- np(subj,In,NPOut),vp(NPOut,Out).
np(_,In,Out) :- det(In,DetOut),n(DetOut,Out).
np(_,In,Out) :- pn(In,Out).
np(CASE,In,Out) :- pro(CASE,In,Out).
vp(In,Out) :- vi(In,Out).
vp(In,Out) :- vt(In,VtOut),np(obj,VtOut,Out).

pro(subj,[he|Out],Out).
pro(obj,[her|Out],Out).
:
```



# Extra arguments in the --> notation

---

<code>s --&gt; np( subj ), vp.</code>	<code>→ s(In,Out) :- np( subj , In, NPOut ), vp(NPOut,Out).</code>
<code>np( _ ) --&gt; det, n.</code>	<code>→ np( _ , In, Out ) :- det( In, DetOut ), n( DetOut, Out ).</code>
<code>np( _ ) --&gt; pn.</code>	<code>→ np( _ , In, Out ) :- pn( In, Out ).</code>
<code>np( CASE ) --&gt; pro( CASE ).</code>	<code>→ np( CASE , In, Out ) :- pro( CASE , In, Out ).</code>
<code>vp --&gt; vi.</code>	<code>→ vp( In, Out ) :- vi( In, Out ).</code>
<code>vp --&gt; vt, np( obj ).</code>	<code>→ vp( In, Out ) :- vt( In, VtOut ), np( obj , VtOut, Out ).</code>

To query:

```
?- s([the,nurse,whistles],[ ]).
```

```
?- np( _, [the,bride], [ ] ).
```

# Practical Session

Write your own DCG.

`http://www.coli.uni-sb.de/~kris/esslli04prolog`