

Simply Logical – Chapter 3 © Peter Flach, 2000

Prolog Programming

We follow Peter Flach's Book Simply Logical, John Wiley.

Simply Logical – Chapter 3, p.44-5 © Peter Flach, 2000

```

student_of(X,T):-follows(X,C),teaches(T,C).
follows(paul,computer_science).
follows(paul,expert_systems).
follows(maria,ai_techniques).
teaches(adrian,expert_systems).
teaches(peter,ai_techniques).
teaches(peter,computer_science).

```

SLD-tree

Simply Logical – Chapter 3, p.45-6 © Peter Flach, 2000

```

brother_of(X,Y):-brother_of(Y,X).
brother_of(paul,peter).

```

Infinite SLD-trees

Simply Logical – Chapter 3, p.47 © Peter Flach, 2000

```

list([]).
list([_|_]):-list(_).

```

Exercise 3.2

Simply Logical - Chapter 3 p.52 © Peter Flach, 2000

```

likes(peter,Y):-friendly(Y).
likes(T,S):-student_of(S,T).
student_of(maria,peter).
student_of(paul,peter).
friendly(maria).

likes(peter,Y):-!,friendly(Y). likes(T,S):-student_of(S,T),!.

```

Exercise 3.3

Simply Logical - Chapter 3 p.54 © Peter Flach, 2000

```

p:-q,r.
p:-not(q),s.
s.

not(Goal):-Goal,!,fail.
not(Goal).

```

not vs. cut

Simply Logical - Chapter 3 p.55 © Peter Flach, 2000

```

p:-not(q),r.
q.
r.

not(Goal):-Goal,!,fail.
not(Goal).

```

:-not(q) fails

Simply Logical - Chapter 3 p.56-7 © Peter Flach, 2000

```

bachelor(X):-not(married(X)),man(X).
man(fred).
man(peter).
married(fred).

```

Prolog's not is unsound

Simply Logical - Chapter 3 p.65 © Peter Flach, 2000

`append_dl(XPlus-XMinus,YPlus-YMinus,XPlus-YMinus):-XMinus=YPlus.`

?-append_dl([a,b|X]-X,[c,d|Y]-Y,Z).
 x = [c,d|Y], z = [a,b,c,d|Y]-Y

Difference lists

Simply Logical - Chapter 3 © Peter Flach, 2000

- ☞ Very often employ
 - Grammar formalisms
 - Logic
 - Unification
- ☞ We : very brief introduction to
 - Definite clause grammars

An illustration : Natural Language Processing

Simply Logical - Chapter 3 © Peter Flach, 2000

Just to give an idea / to illustrate
 Computer science point of view !

- ☞ Parsing
 - Analyzing the syntactic structure of sentences
 - Construct the parse tree
- ☞ Interpretation
 - Determine the meaning (internal representation thereof) of the sentence
 - Internal representation can then be used to reason

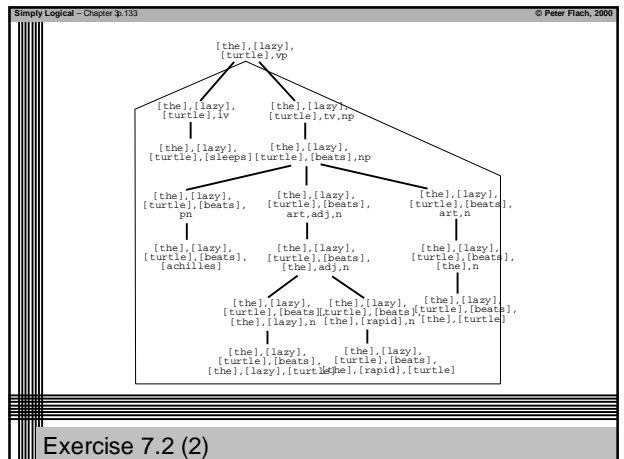
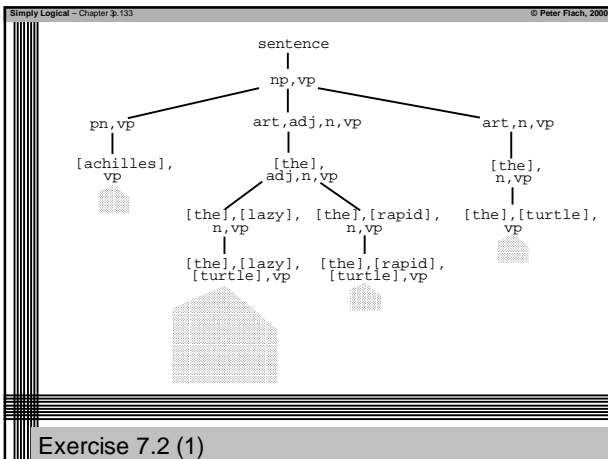
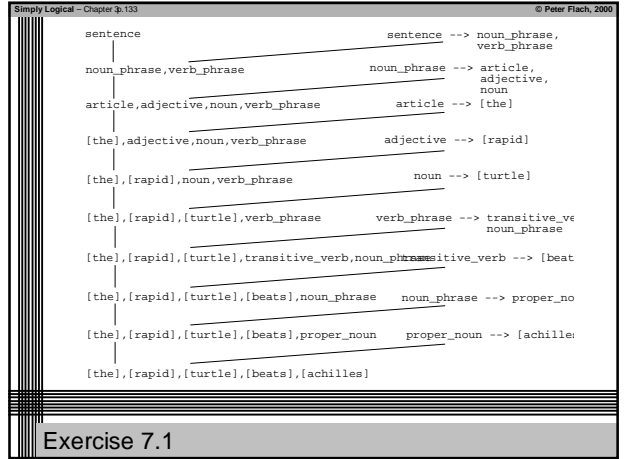
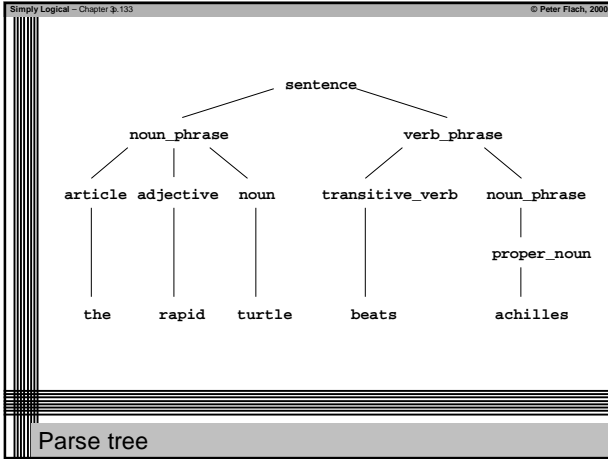
Example Application
 NLP interface to a database

Two tasks in NLP

Simply Logical - Chapter 3.132 © Peter Flach, 2000

sentence	--> noun_phrase,verb_phrase.
noun_phrase	--> proper_noun.
noun_phrase	--> article,adjective,noun.
noun_phrase	--> article,noun.
verb_phrase	--> intransitive_verb.
verb_phrase	--> transitive_verb,noun_phrase.
article	--> [the].
adjective	--> [lazy].
adjective	--> [rapid].
proper_noun	--> [achilles].
noun	--> [turtle].
intransitive_verb	--> [sleeps].
transitive_verb	--> [beats].

Context-free grammar



Simply Logical - Chapter 3 © Peter Flach, 2000

```

sentence --> noun_phrase, verb_phrase
translate as
sentence(S)-
  noun_phrase(Np),
  verb_phrase(Vp) ,
  append(Np,Vp,S).
verb--X[sleep].
Translate a s
verb([sleep]).
Parsing (answer following query)
:- sentence([the,rapid,turtle,beats,achilles]).

```

Translating CFG in Prolog

Simply Logical - Chapter 3:135 © Peter Flach, 2000

```

sentence(NP1-VP2):-
  noun_phrase(NP1-VP1),
  verb_phrase(VP1-VP2)

```

Difference lists in grammar rules

Simply Logical - Chapter 3 © Peter Flach, 2000

```

sentence --> noun_phrase, verb_phrase
translate a s
sentence(Np1,Vp2)-
  noun_phrase(Np1,Vp1),
  verb_phrase(Vp1,Vp2).
verb--X[sleep].
Translate a s
verb([sleep|X],X).
Parsing (answer following query)

```

Translating CFG in Prolog

Simply Logical - Chapter 3:136 © Peter Flach, 2000

	GRAMMAR	PARSING
META-LEVEL	$s \rightarrow np, vp$	$?-phrase(s, L)$
OBJECT-LEVEL	$s(L, L0) :- np(L, L1), vp(L1, L0)$	$?-s(L, [])$

Meta-level vs. object-level

```

Simply Logical - Chapter 3:137 © Peter Flach, 2000

sentence      --> noun_phrase(N),verb_phrase(N).
noun_phrase   --> article(N),noun(N).
verb_phrase   --> intransitive_verb(N).
article(singular) --> [a].
article(singular) --> [the].
article(plural) --> [the].
noun(singular) --> [turtle].
noun(plural)  --> [turtles].
intransitive_verb(singular) --> [sleeps].
intransitive_verb(plural)  --> [sleep].

```

Non-terminals with arguments

```

Simply Logical - Chapter 3:137-8 © Peter Flach, 2000

sentence(s(NP,VP)) --> noun_phrase(NP),verb_phrase(
noun_phrase(np(N)) --> proper_noun(N).
noun_phrase(np(Art,Adj,N)) --> article(Art),adjective(Ad
noun(N).
noun_phrase(np(Art,N)) --> article(Art),noun(N).
verb_phrase(vp(IV)) --> intransitive_verb(IV).
verb_phrase(vp(TV,NP)) --> transitive_verb(TV),
noun_phrase(NP).
article(art(the)) --> [the].
adjective(adj(lazy)) --> [lazy].
adjective(adj(rapid)) --> [rapid].
proper_noun(pn(achilles)) --> [achilles].
noun(n(turtle)) --> [turtle].
intransitive_verb(iv(sleeps)) --> [sleeps].
transitive_verb(tv(beans)) --> [beans].

?-phrase(sentence(T),[achilles,beans,the,lazy,turtl
T = s(np(pn(achilles)),
vp(tv(beans),
np(art(the),
adj(lazy),
n(turtle))))

```

Constructing parse trees

```

Simply Logical - Chapter 3:135-9 © Peter Flach, 2000

numeral(N) --> n1_999(N).
numeral(N) --> n1_9(N1),[thousand],n1_999(N2),
{N is N1*1000+N2}.
n1_999(N) --> n1_99(N).
n1_999(N) --> n1_9(N1),[hundred],n1_99(N2),
{N is N1*100+N2}.
n1_99(N) --> n0_9(N).
n1_99(N) --> n10_19(N).
n1_99(N) --> n20_90(N).
n1_99(N) --> n20_90(N1),n1_9(N2),{N is N1+N2}.
n0_9(0) --> [].
n0_9(N) --> n1_9(N).
n1_9(1) --> [one].
n1_9(2) --> [two].
...
?-phrase(numeral(2211),N).
N = [two,thousand,two,hundred,elevet
...
n10_19(10) --> [ten].
n10_19(11) --> [eleven].
...
n20_90(20) --> [twenty].
n20_90(30) --> [thirty].
...

```

Prolog goals in grammar rules

```

Simply Logical - Chapter 3:140 © Peter Flach, 2000

☞ The meaning of the proper noun 'Socrates' is the term socrates
proper_noun(socrates) --> [socrates].

☞ The meaning of the property 'mortal' is a mapping from terms to
literals containing the unary predicate mortal
property(X=>mortal(X)) --> [mortal].

☞ The meaning of a proper noun - verb phrase sentence is a
clause with empty body and head obtained by applying the
meaning of the verb phrase to the meaning of the proper noun
sentence((L:-true)) -->
proper_noun(X),verb_phrase(X=>L).
?-phrase(sentence(C),[socrates,is,mortal].
C = (mortal(socrates):-true)

```

Interpretation

Simply Logical - Chapter 3:140 © Peter Flach, 2000

☞ A transitive verb is a binary mapping from a pair of terms to literals

```
transitive_verb(Y=>X=>likes(X,Y)) --> [likes].
```

☞ A proper noun instantiates one of the arguments, returning a unary mapping

```
verb_phrase(M) -->
transitive_verb(Y=>M),proper_noun(Y).
```

Exercise 7.4

Simply Logical - Chapter 3:140-1 © Peter Flach, 2000

```
sentence((L:-true)) -->
proper_noun(X),verb_phrase(X=>L).
sentence((H:-B)) -->
[every],noun(X=>B),verb_phrase(X=>H).
% NB. separate 'determiner' rule removed, see later

verb_phrase(M) --> [is],property(M).
property(M) --> [a],noun(M).
property(X=>mortal(X)) --> [mortal].
proper_noun(socrates) --> [socrates].
noun(X=>human(X)) --> [human].
```

Interpretation (2)

Simply Logical - Chapter 3:141 © Peter Flach, 2000

```
?-phrase(sentence(C),S).

C = human(X):-human(X)
S = [every,human,is,a,human];

C = mortal(X):-human(X)
S = [every,human,is,mortal];

C = human(socrates):-true
S = [socrates,is,a,human];

C = mortal(socrates):-true
S = [socrates,is,mortal];
```

Interpretation (3)

Simply Logical - Chapter 3:141 © Peter Flach, 2000

☞ 'Determiner' sentences have the form 'every/some [noun] [verb-phrase]' (NB. meanings of 'some' sentences require 2 clauses)

```
sentence(Cs) --> determiner(M1,M2,Cs),noun(M1),verb_phrase(M2).
determiner(X=>B, X=>H,[(H:-B)]) --> [every].
determiner(sk=>H1,sk=>H2,[(H1:-true),(H1:-true)]) --> [some].

?-phrase(sentence(Cs),[D,human,is,mortal]).
D = every, Cs = [(mortal(X):-human(X))];
D = some, Cs = [(human(sk):-true),(mortal(sk):-true)]
```

Determiners

Simply Logical – Chapter 3:142 © Peter Flach, 2000

```

question(Q)      --> [who],[is],property(X=>Q).

question(Q)      -->
[is],proper_noun(X),property(X=>Q).

question((Q1,Q2)) --> [is],[some],noun(sk=>Q1),
                    property(sk=>Q2).

```

Questions

Simply Logical – Chapter 3:143 © Peter Flach, 2000

```

handle_input(Question,Rulebase):-
phrase(question(Query),Question), % question
prove_rb(Query,Rulebase),!,      % it can be
solved                           %
transform(Query,Clauses),        % transform to
phrase(sentence(Clauses),Answer), % answer
show_answer(Answer),
nl_shell(Rulebase).

```

Querying a rulebase

Simply Logical – Chapter 3 © Peter Flach, 2000

- ☞ **Real Grammars**
 - Are much more complicated
 - Account for detailed syntactic and semantic aspects of language
 - Take ages to develop
 - ...
- ☞ **Extension for NLP database interface of present grammar is possible**
 - Assert interpreted facts/clauses in Database
 - Logical reasoning possible
 - Extend with queries, e.g. What - Who - Where queries
 - Who is mortal ?
 - What is Socrates ?
 - ...

Real NLP

Simply Logical – Chapter 3 © Peter Flach, 2000

- ☞ **Use sicstus (installed in Pool) or download SWI-prolog**
<http://www.swi.psy.uva.nl/projects/SWI-Prolog/>
- ☞ **Beware of syntax !**
- ☞ **File append.pl**

```

append(nil,X,X).
append(cons(X,Y),Z,cons(X,W)):-append(Y,Z,W).

```
- ☞ **\$ sicstus**

```

?-consult(append).
Yes
?-append(cons(a,nil),cons(b,cons(c,nil)),X).
X= cons(a,cons(b,cons(c,nil)))
Yes

```
- ☞ **Good books on Prolog**
 - Peter Flach, Simply Logical, Wiley
 - Ivan Bratko, Prolog programming for AI, Addison Wesley.

Prolog

☞ Good books on Prolog

Peter Flach, Simply Logical, Wiley
Ivan Bratko, Prolog programming for AI, Addison Wesley.
Sterling and Shapiro, The art of Prolog, MIT Press