Exercises

(1) Extend the "sentence" grammar to allow the creation of "and" sentences

```
sentence  ::=  noun verb noun
noun      ::=  dogs
noun      ::=  teeth
verb      ::=  have
```

```
sentence  ::=  noun verb noun
sentence  ::=  sentence and sentence
…
```

(2) Write a grammar for binary numbers

```
digit  ::= 0              (R1)
digit  ::= 1              (R2)
bin    ::= digit          (R3)
bin    ::= digit bin      (R4)
```

```
bin  ::= 0 bin            (R1)
bin  ::= 1 bin            (R2)
bin  ::= ϵ                (R3)
```

"empty RHS"

Exercises

(3) Derive the sentence 101

digit	::= 0	(R1)
digit	::= 1	(R2)
bin	::= digit	(R3)
bin	::= digit bin	(R4)

bin	::= 0 bin	(R1)
bin	::= 1 bin	(R2)
bin	::= ϵ	(R3)

bin	
digit bin	(R4)
digit digit bin	(R4)
digit digit digit	(R3)
digit digit 1	(R2)
digit 0 1	(R1)
1 0 1	(R2)

bin	
1 bin	(R2)
1 0 bin	(R1)
1 0 1 bin	(R2)
1 0 1	(R3)

Exercises

(4) Write a grammar for boolean expression built from the constants T and F and the operation not

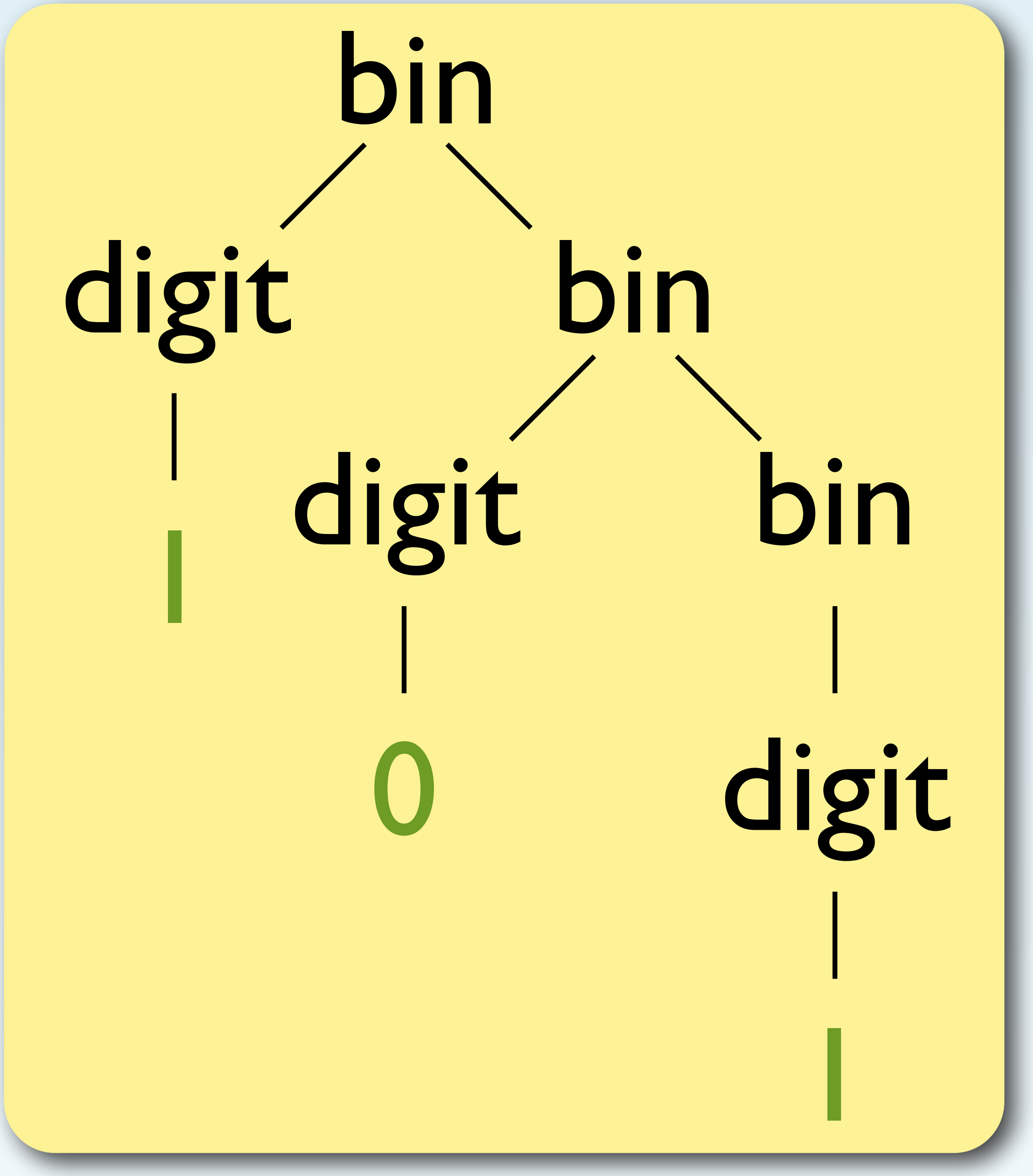
bool	::=	T	(R1)
bool	::=	F	(R2)
bool	::=	not bool	(R3)

(5) Derive the sentence not not F

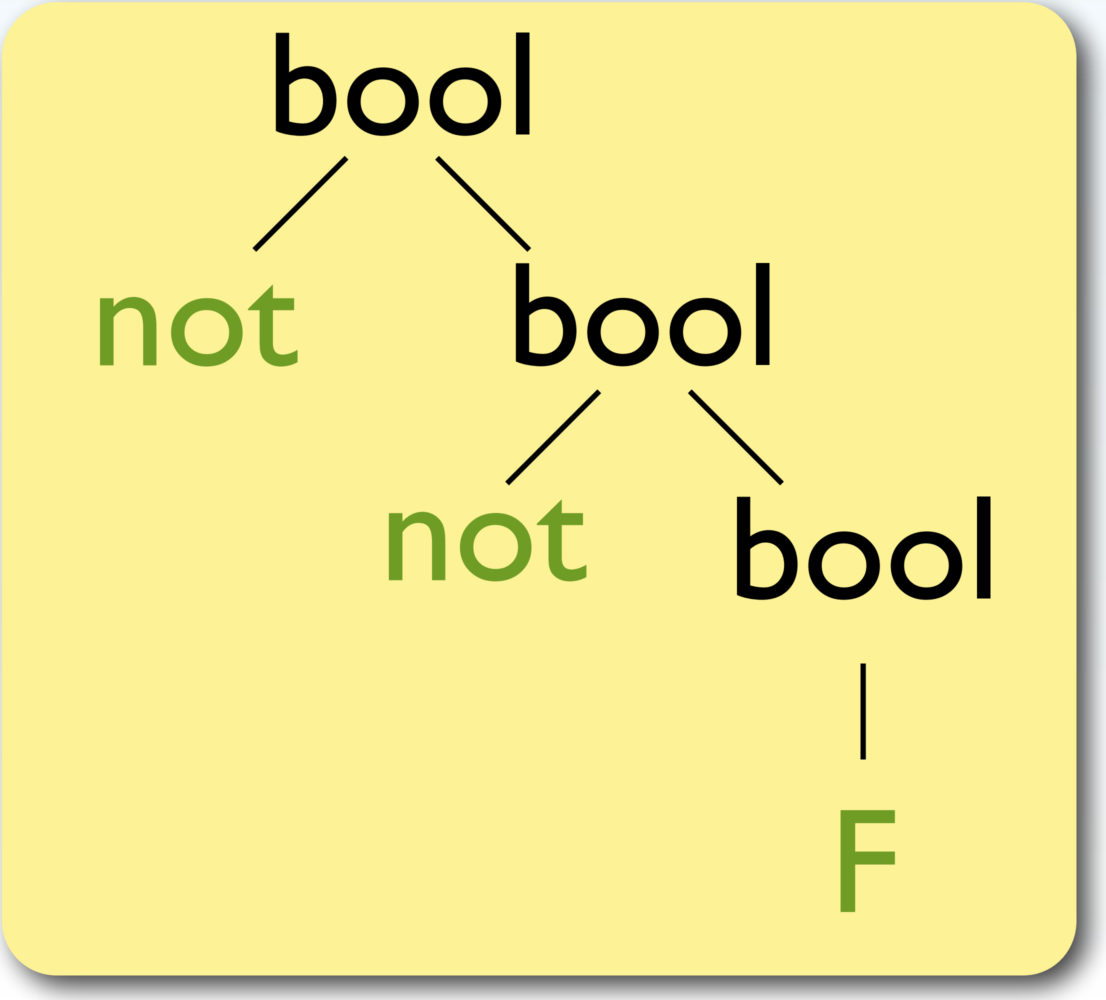
bool	
not bool	(R3)
not not bool	(R3)
not not F	(R2)

Exercises

(1) Draw the syntax tree for the sentence 101



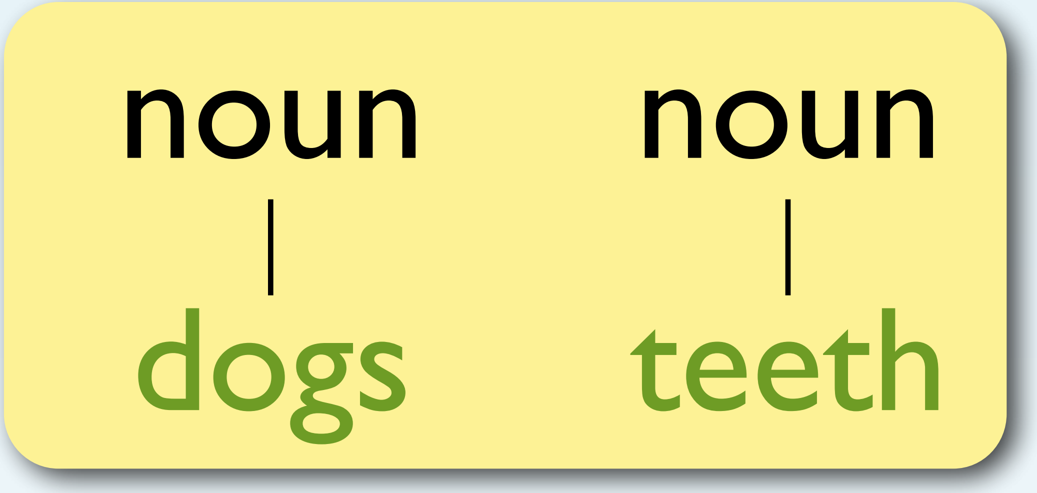
(2) Draw the syntax tree for the sentence not(not(F))



CS 381 • Winter 2015

Exercises

(3) Draw all syntax trees of type noun



noun
dogs

noun
teeth

(4) How many sentences/trees of type "stmt" can constructed with the following grammar?

```
cond  ::=  T
stmt  ::=  while cond do stmt
```

One, which is infinitely large. There is no way to avoid nesting stmt rules forever.

(5) How many with the following grammar?

```
cond  ::=  T
stmt  ::=  while cond do stmt
stmt  ::=  noop
```

Infinitely many. We can choose to nest any number of stmt rules, or stop with a noop