HW #5: Feature-based Parsing

Agreement with Heads and Features

- $\beta \rightarrow \beta_1 \ldots \beta_n$
  $\{set\ of\ constraints\}$ $\langle \beta_i\ feature\ path \rangle = Atomic\ value \mid \langle \beta_j\ feature\ path \rangle$

$$\boldsymbol{S \rightarrow NP\ \ VP}$$
$$\langle \boldsymbol{NP}\ \text{AGREEMENT} \rangle = \langle \boldsymbol{VP}\ \text{AGREEMENT} \rangle$$

$$\boldsymbol{S \rightarrow Aux\ NP\ \ VP}$$
$$\langle \boldsymbol{Aux}\ \text{AGREEMENT} \rangle = \langle \boldsymbol{NP}\ \text{AGREEMENT} \rangle$$

$$\boldsymbol{NP \rightarrow Det\ \ Nominal}$$
$$\langle \boldsymbol{Det}\ \text{AGREEMENT} \rangle = \langle \boldsymbol{Nominal}\ \text{AGREEMENT} \rangle$$
$$\langle \boldsymbol{NP}\ \text{AGREEMENT} \rangle = \langle \boldsymbol{Nominal}\ \text{AGREEMENT} \rangle$$

$$\boldsymbol{Aux} \rightarrow does$$
$$\langle \boldsymbol{A\text{UX}}\ \text{AGREEMENT NUMBER} \rangle = \boldsymbol{sg}$$
$$\langle \boldsymbol{A\text{UX}}\ \text{AGREEMENT PERSON} \rangle = \boldsymbol{3rd}$$

$$\boldsymbol{Det} \rightarrow this$$
$$\langle \boldsymbol{Det}\ \text{AGREEMENT NUMBER} \rangle = \boldsymbol{sg}$$

$$\boldsymbol{Det} \rightarrow these$$
$$\langle \boldsymbol{Det}\ \text{AGREEMENT NUMBER} \rangle = \boldsymbol{pl}$$

$$\boldsymbol{Verb} \rightarrow serve$$
$$\langle \boldsymbol{Verb}\ \text{AGREEMENT NUMBER} \rangle = \boldsymbol{pl}$$

$$\boldsymbol{Noun} \rightarrow flight$$
$$\langle \boldsymbol{Noun}\ \text{AGREEMENT NUMBER} \rangle = \boldsymbol{sg}$$

Goals

- Explore the role of features in implementing linguistic constraints.
- Identify some of the challenges in building compact constraints to define a precise grammar.
- Apply feature-based grammars to perform grammar checking.

Tasks

- Build a Feature-Based Grammar
  - We will focus on the building of the grammar itself — you may use NLTK's `nltk.parse.FeatureEarleyChartParser` or similar.
- Use the grammar to parse a small set of sentences we provide.

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
- Det[NUM=sg] -> 'this' | 'every’
- Det[NUM=pl] -> 'these' | 'all’
- N[NUM=sg] -> 'dog' | 'girl' | 'car' | 'child’
- N[NUM=pl] -> 'dogs' | 'girls' | 'cars' | 'children'

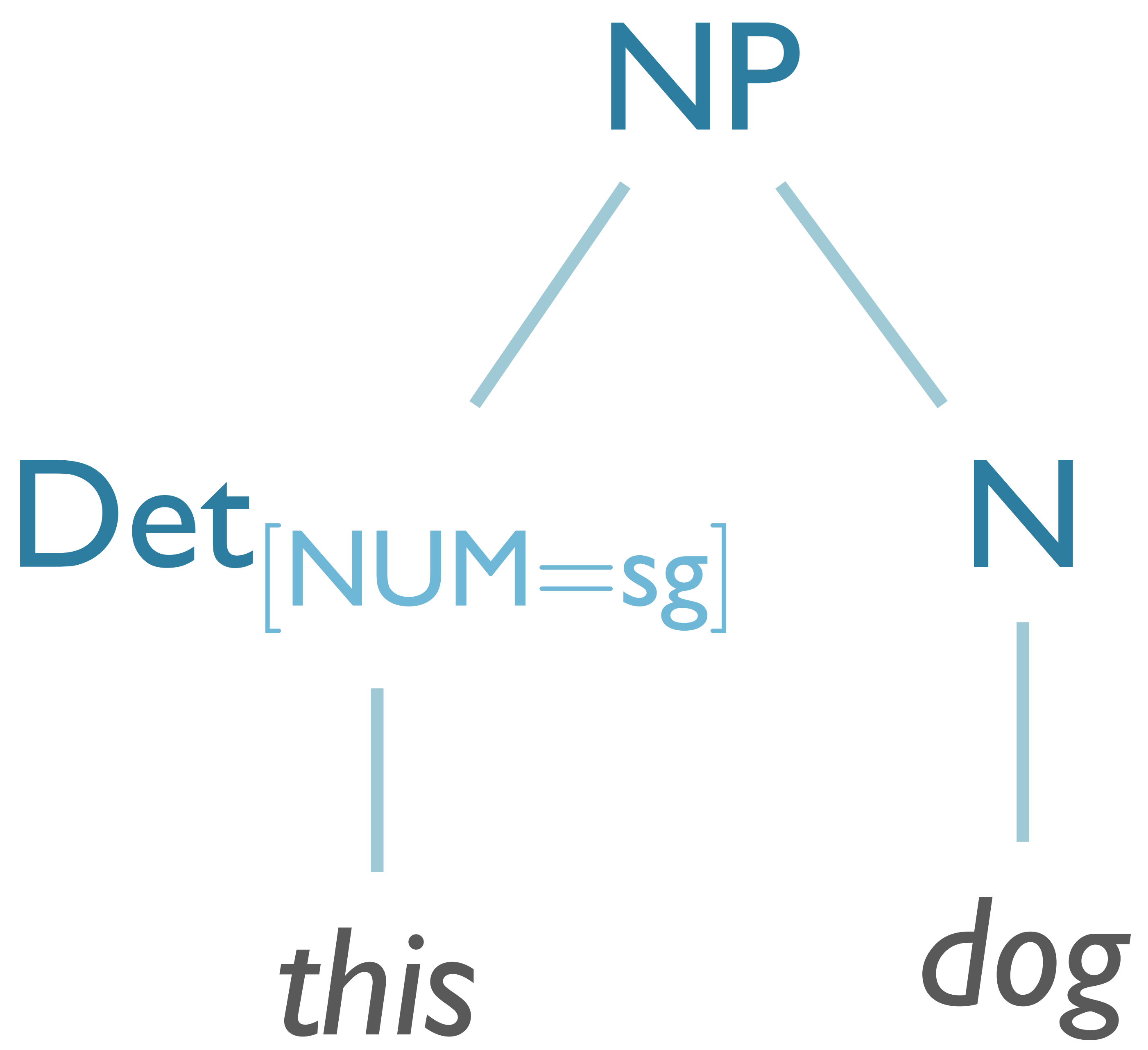
NLTK Feature Syntax

- Basics
  - `X[FEAT1=VALUE1, FEAT2=VALUE2]`
- Variables
  - `X[FEAT=?f]`
- Binary Values
  - `X[-FEAT], Y[+FEAT]`

HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

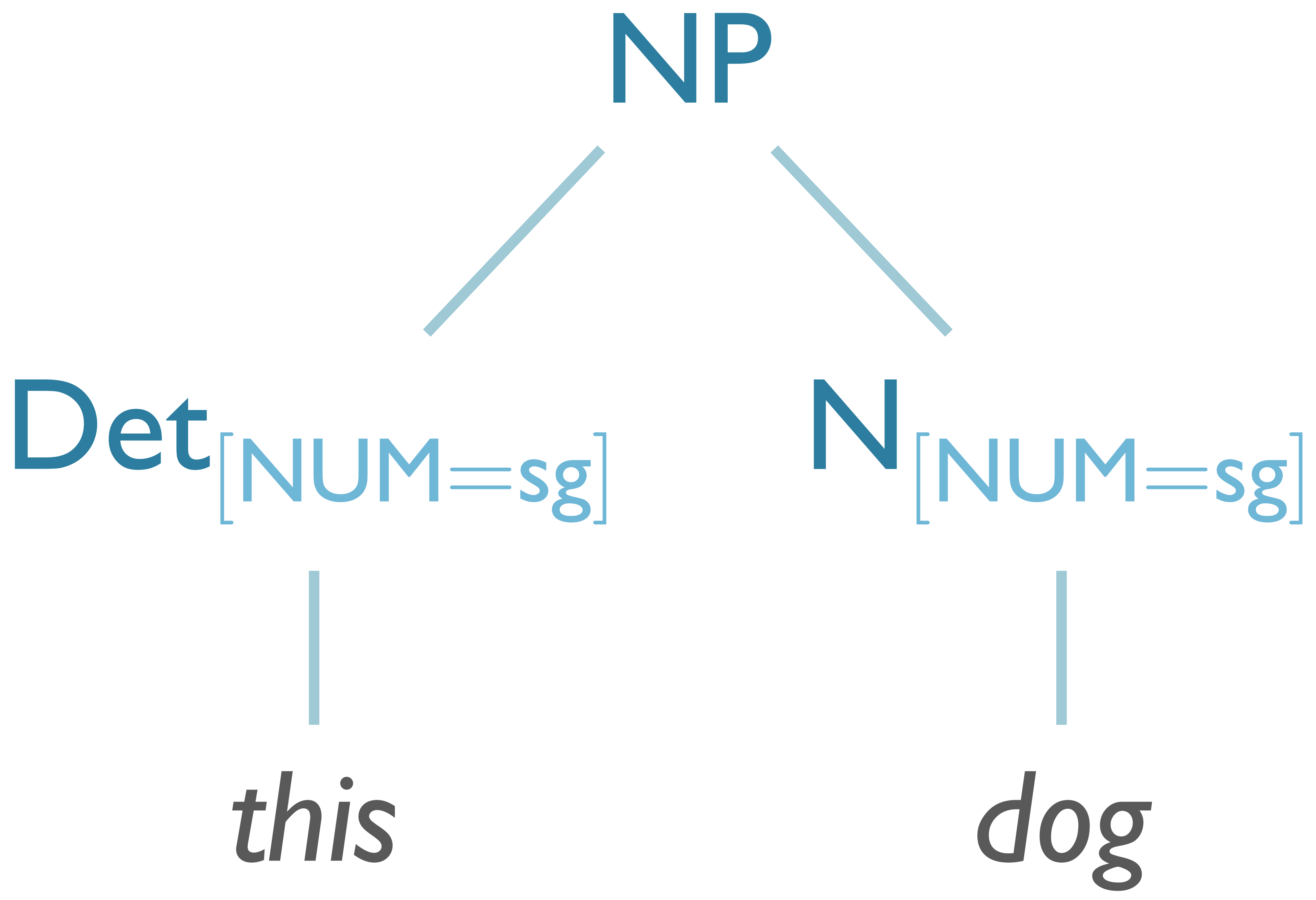
```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

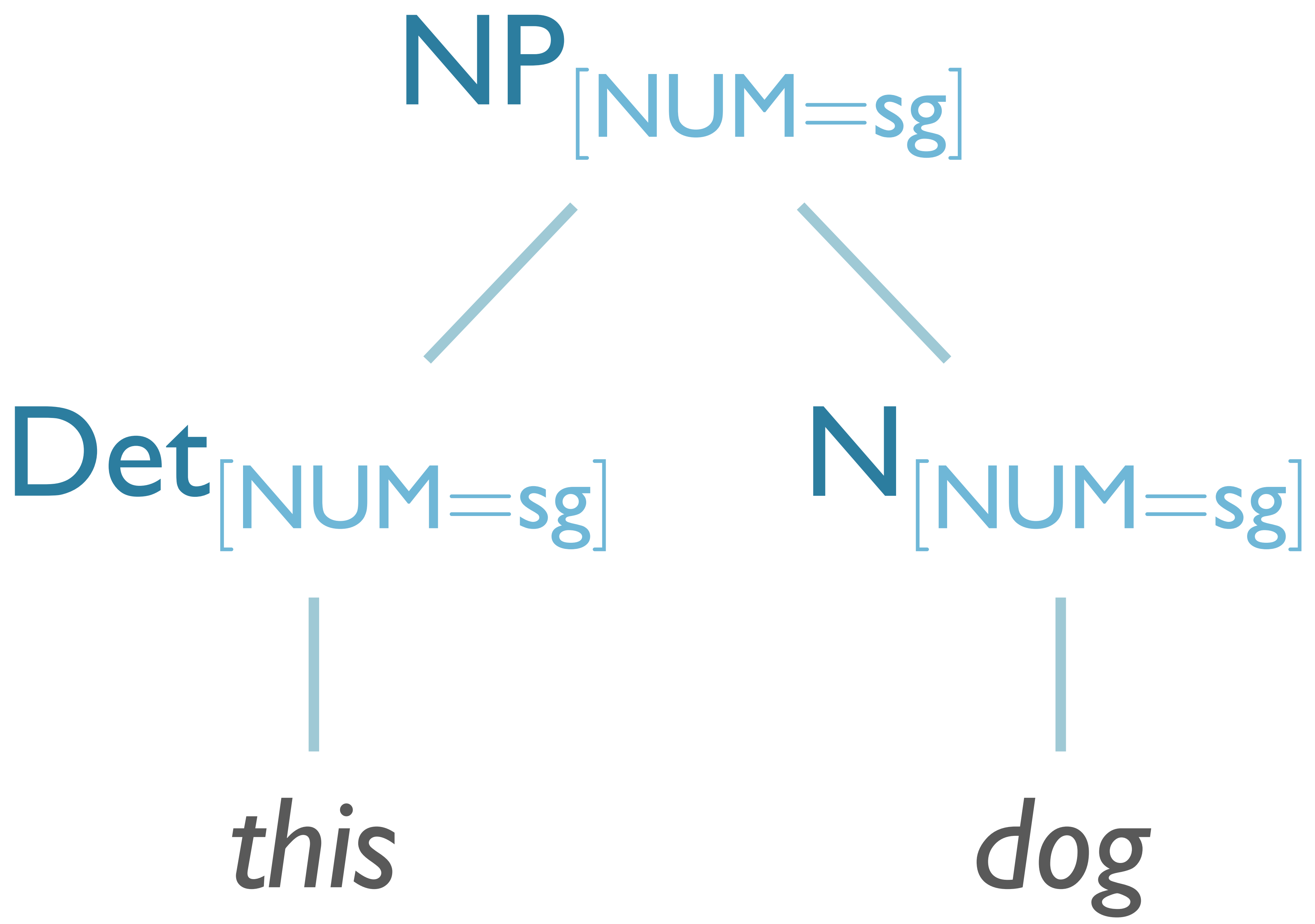
```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



$\text{NP}_{[\text{NUM}=\text{sg}]}$

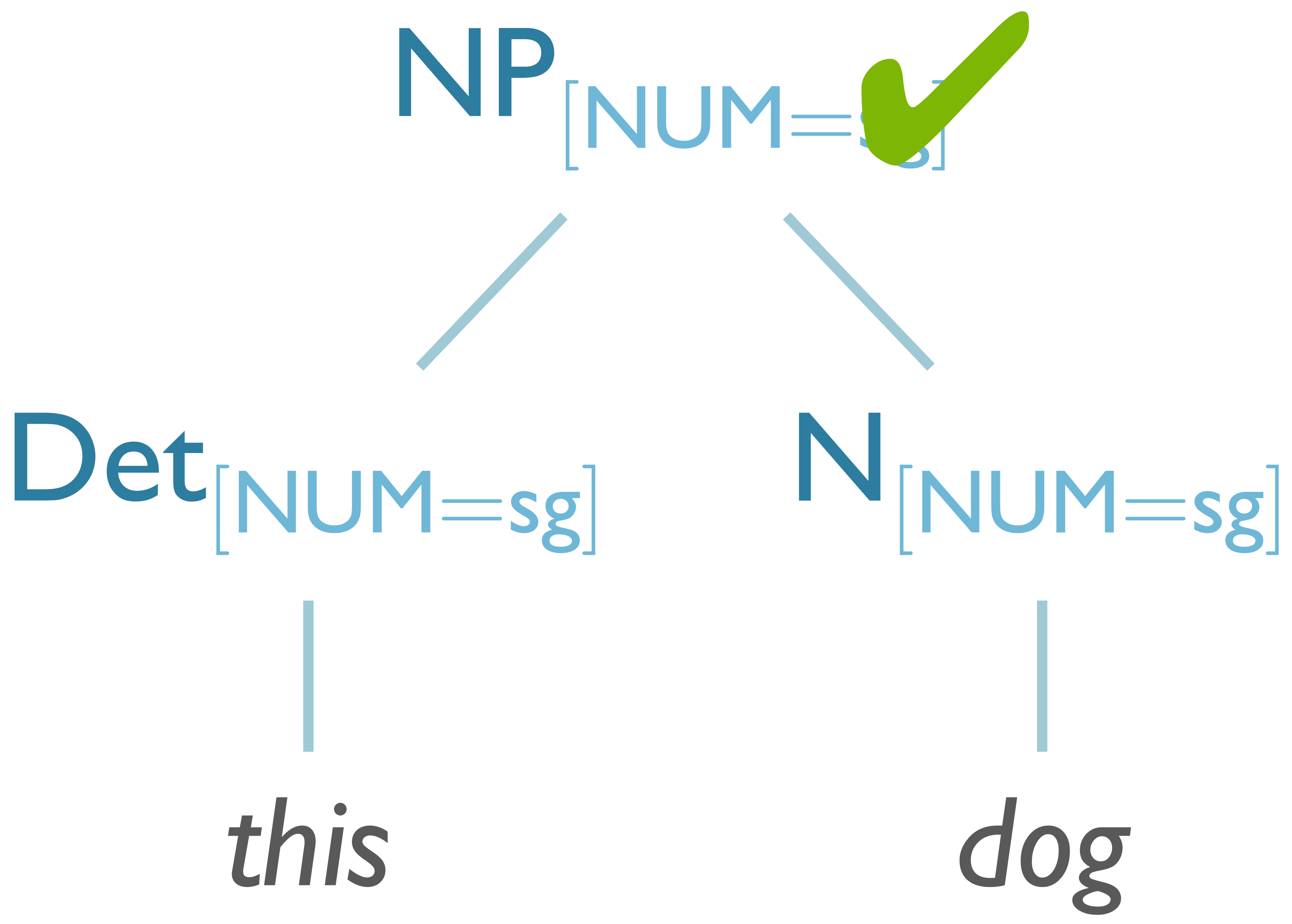
$\text{Det}_{[\text{NUM}=\text{sg}]}$ $\text{N}_{[\text{NUM}=\text{sg}]}$

this *dog*

HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

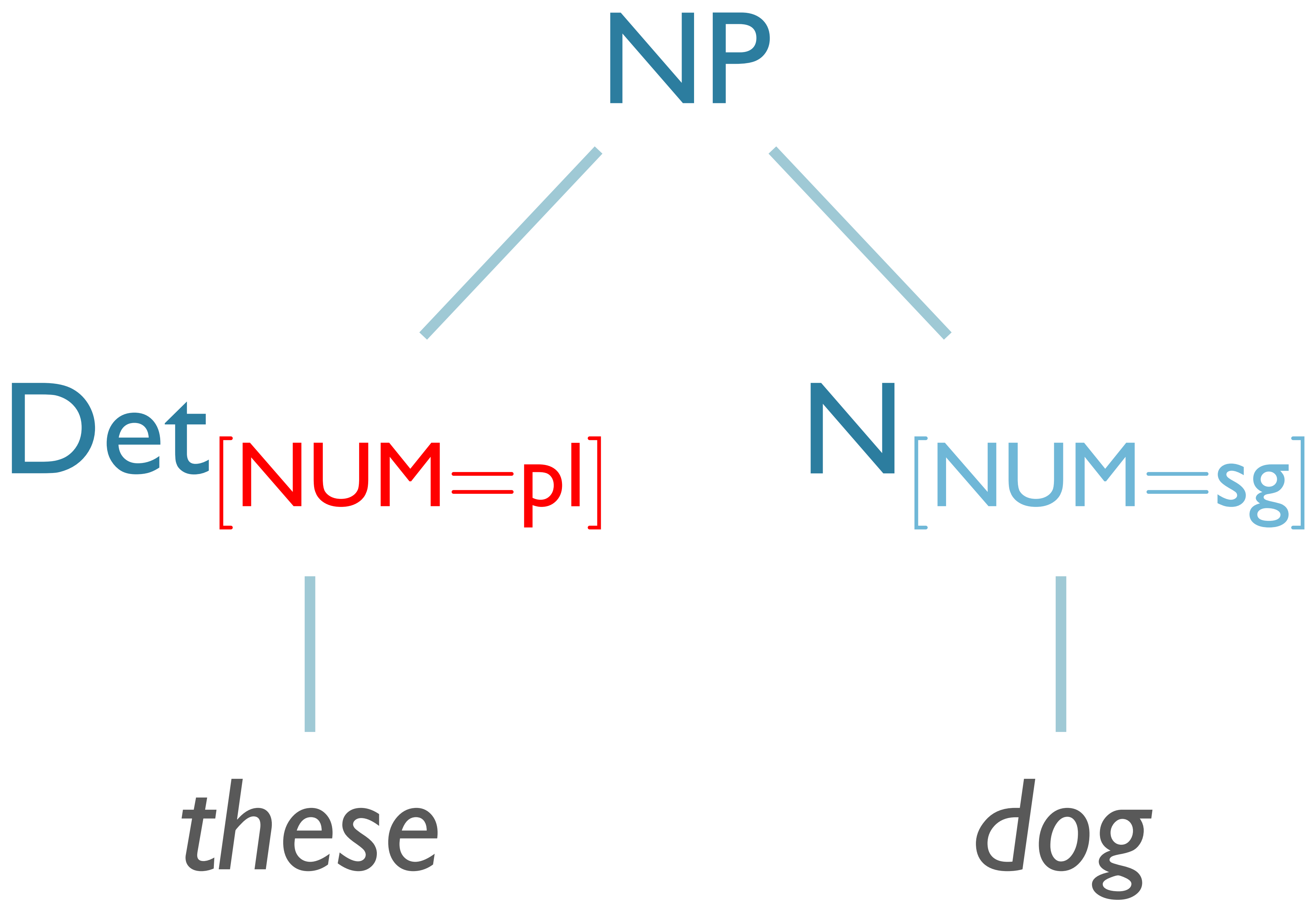
```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

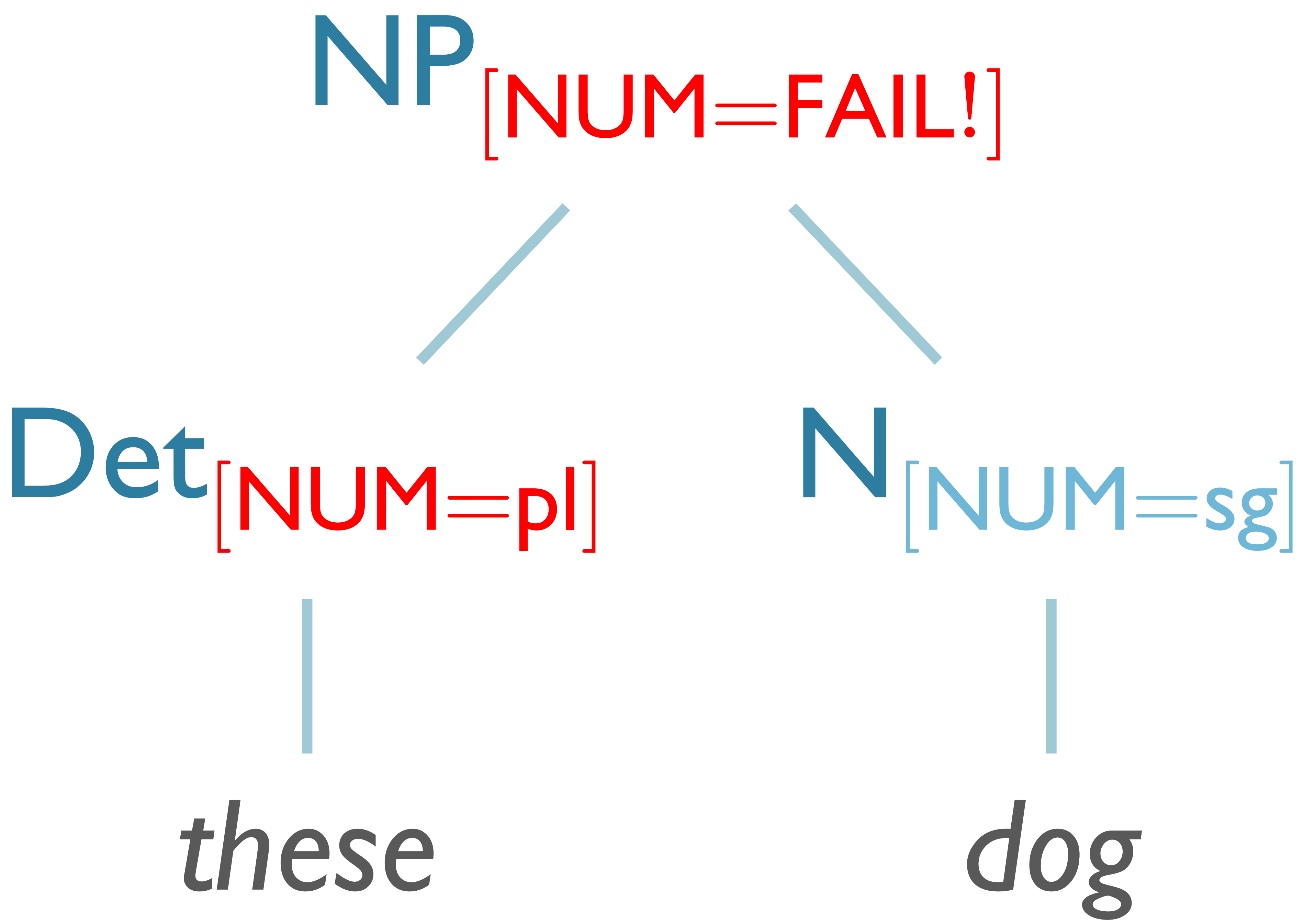
```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```


NP
$\text{Det}_{[\text{NUM}=\text{pl}]}$
$\text{N}_{[\text{NUM}=\text{sg}]}$
these
dog

HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



$\text{NP}_{[\text{NUM}=\text{FAIL!}]}$

$\text{Det}_{[\text{NUM}=\text{pl}]}$

$\text{N}_{[\text{NUM}=\text{sg}]}$

these

dog

HW #5: Grammars

- It's possible to get the grammar to work with completely arbitrary rules, BUT…
- We would prefer them to be linguistically motivated!
  - instead of `[IT_OK=yes]` or `[PRON_AGR=it]`
  - `[GENDER=neut, PERSON=3rd, NUMBER=sg]`

Parsing with Features

```
>>> cp = load_parser('grammars/book_grammars/
feat0.fcfg’)
>>> for tree in cp.parse(tokens):
...     print(tree)

(S[] (NP[NUM='sg']
  (PropN[NUM='sg'] Kim))
    (VP[NUM='sg', TENSE='pres']
      (TV[NUM='sg', TENSE='pres'] likes)
      (NP[NUM='pl'] (N[NUM='pl'] children))))
```

Feature Applications

- Subcategorization
  - Verb-Argument constraints
    - Number, type, characteristics of args
      - e.g. is the subject *animate*?
      - Also adjectives, nouns
- Long-distance dependencies
  - e.g. filler–gap relations in wh-questions

Morphosyntactic Features

- Grammtical feature that influences morphological or syntactic behavior
  - English:
    - Number:
      - Dog, dogs
    - Person:
      - am; are; is
    - Case (more prominent in other languages):
      - I / me; he / him; etc.

Semantic Features

- Grammatical features that influence semantic (meaning) behavior of associated units
- E.g.:
  - *?The rocks slept.*
- Many proposed:
  - `Animacy: +/-`
  - `Gender: masculine, feminine, neuter`
  - `Human: +/-`
  - `Adult: +/-`
  - `Liquid: +/-`

Aspect (J&M 17.4.2)

- *The climber [hiked] [for six hours].*

Aspect (J&M 17.4.2)

- *The climber [hiked] [for six hours].*
- *The climber [hiked] [on Saturday].*

Aspect (J&M 17.4.2)

- *The climber [hiked] [for six hours].*
- *The climber [hiked] [on Saturday].*
- *The climber [reached the summit] [on Saturday].*

Aspect (J&M 17.4.2)

- *The climber [hiked] [for six hours].*
- *The climber [hiked] [on Saturday].*
- *The climber [reached the summit] [on Saturday].*
- **The climber [reached the summit] [for six hours].*

Aspect (J&M 17.4.2)

- *The climber [**hiked**] [**for six hours**].*
- *The climber [**hiked**] [**on Saturday**].*
- *The climber [**reached the summit**] [**on Saturday**].*
- **The climber [**reached the summit**] [**for six hours**].*

- Contrast:
  - *Achievement* (in an instant) vs *activity* (for a time)

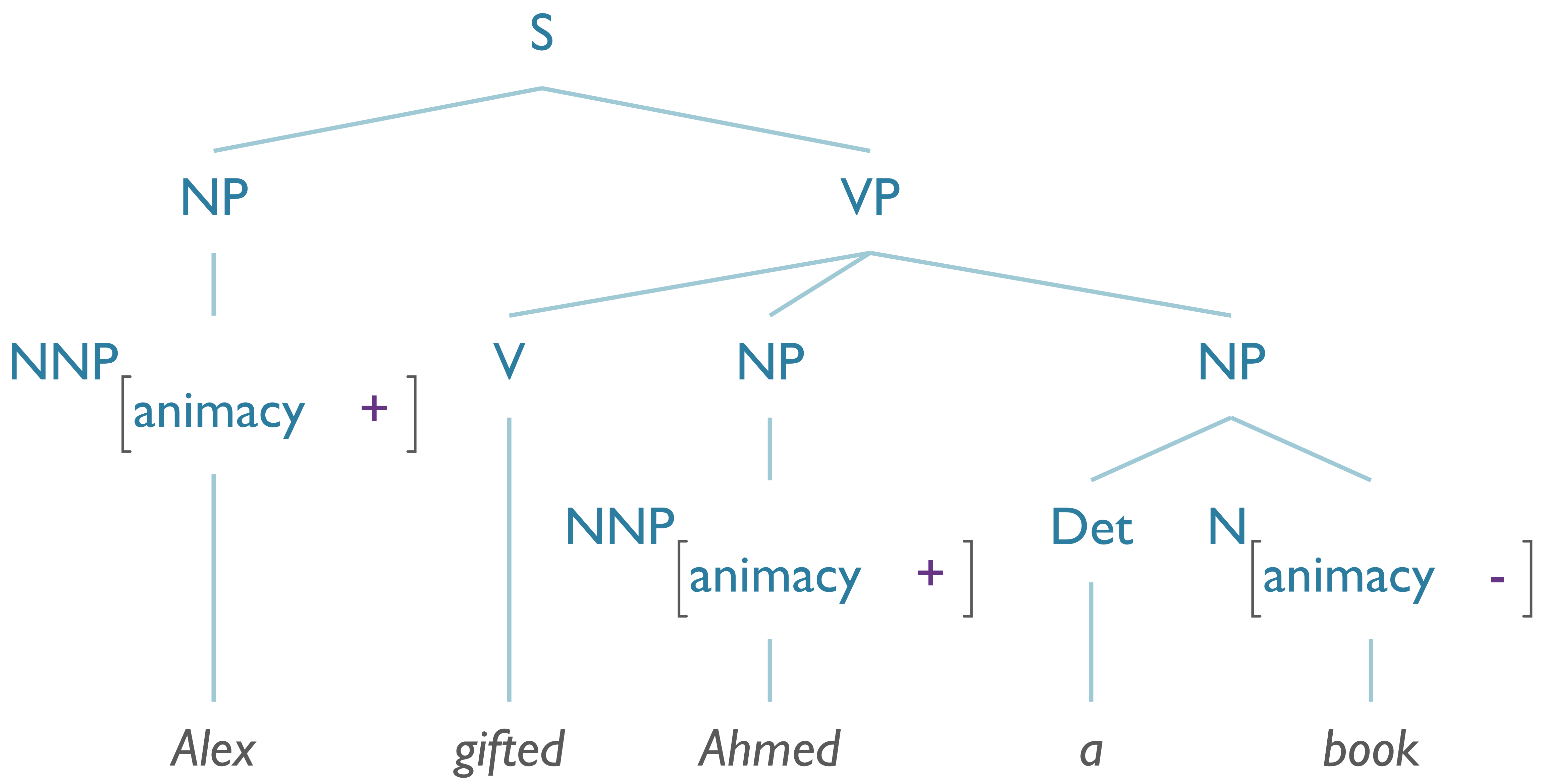
Feature Grammar Practice: Animacy

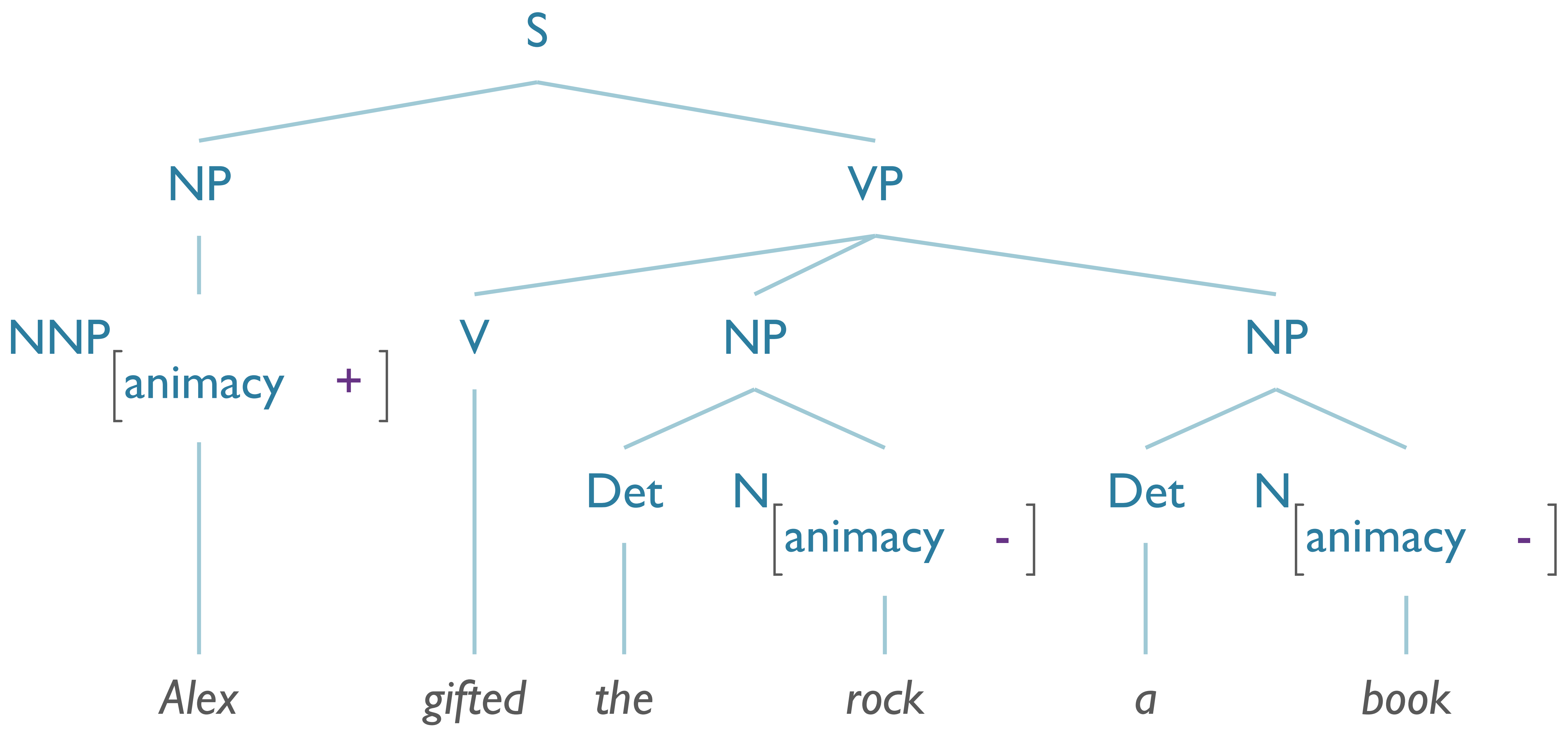
Feature Grammar Practice

- **Initial Grammar:**

```
S -> NP VP
VP[subcat=ditrans] -> V NP NP
NP -> NNP
NP -> Det N
NNP[animacy=True] -> 'Alex' | 'Ahmed'
V[subcat=ditrans] -> 'gifted'
Det -> 'a' | 'the'
N[animacy=False] -> 'book' | 'rock'
```

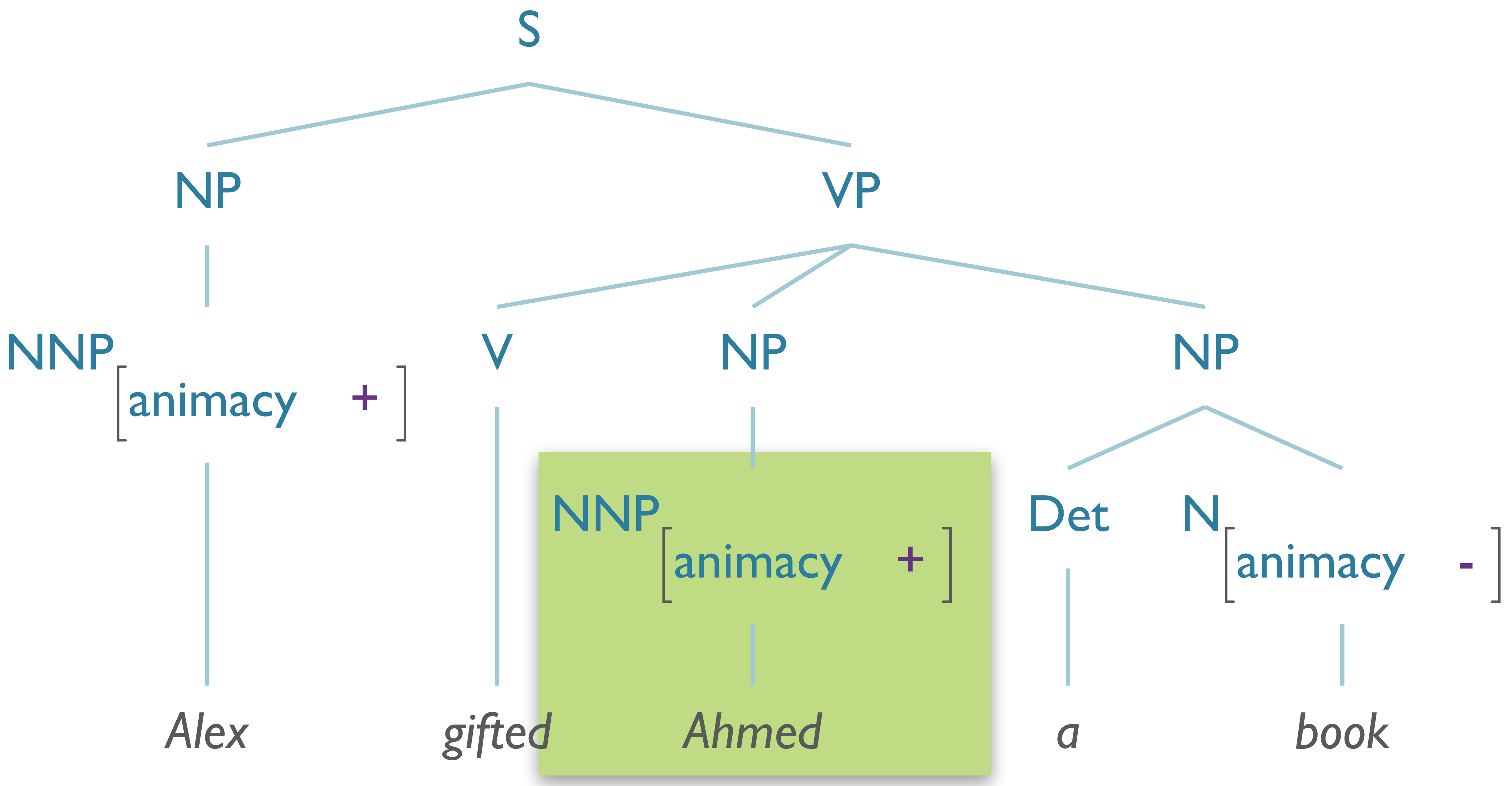
Feature Grammar Practice

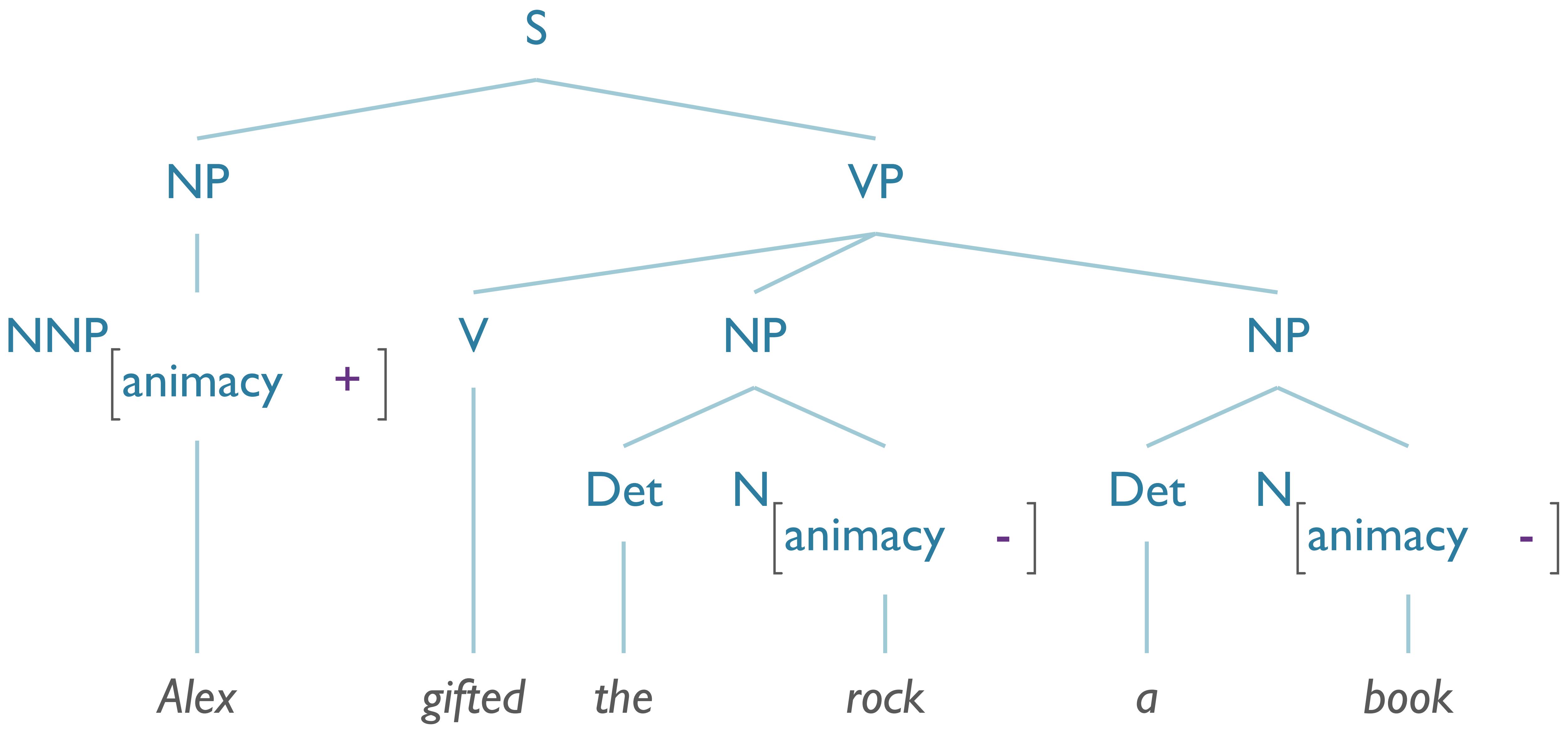




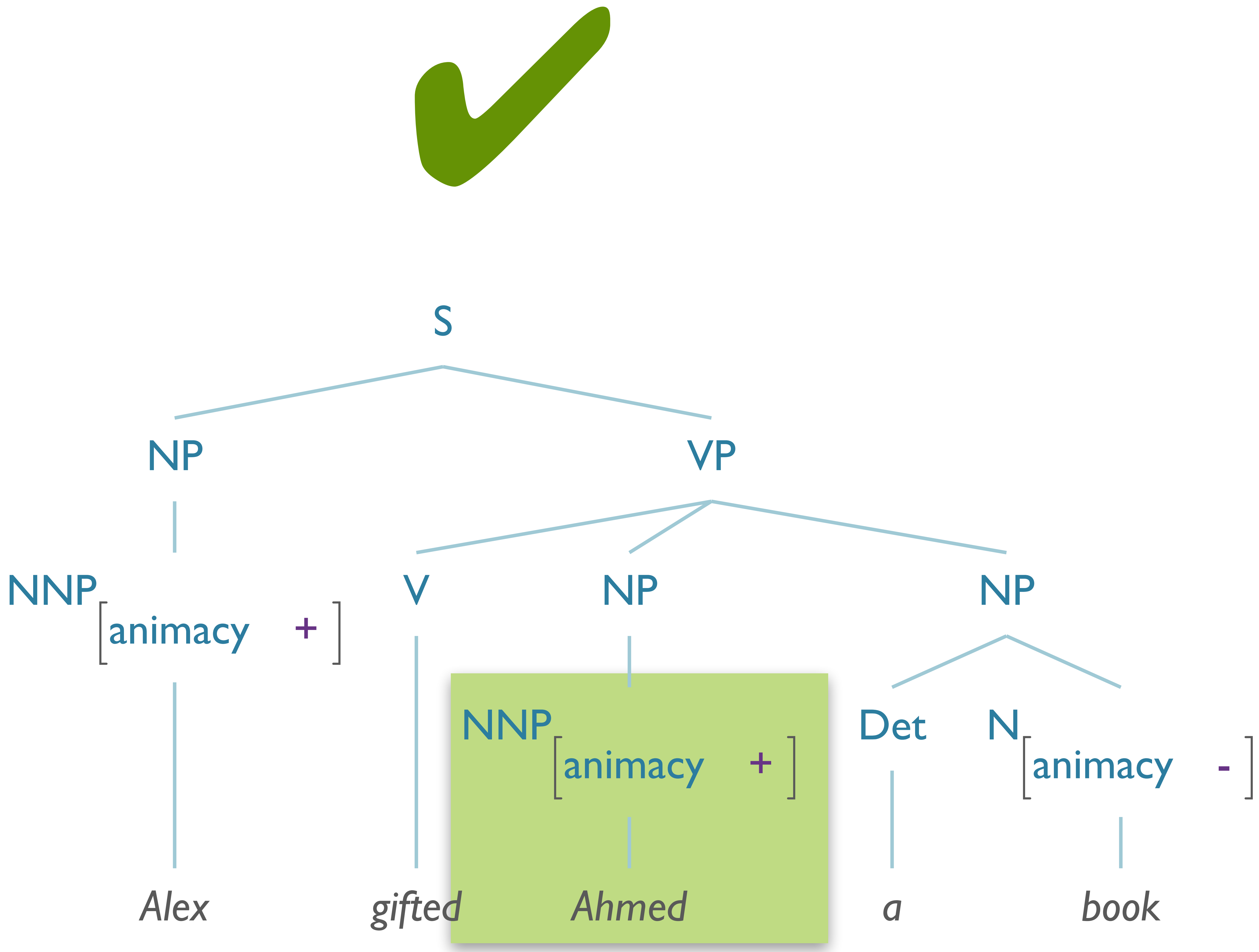
Feature Grammar Practice

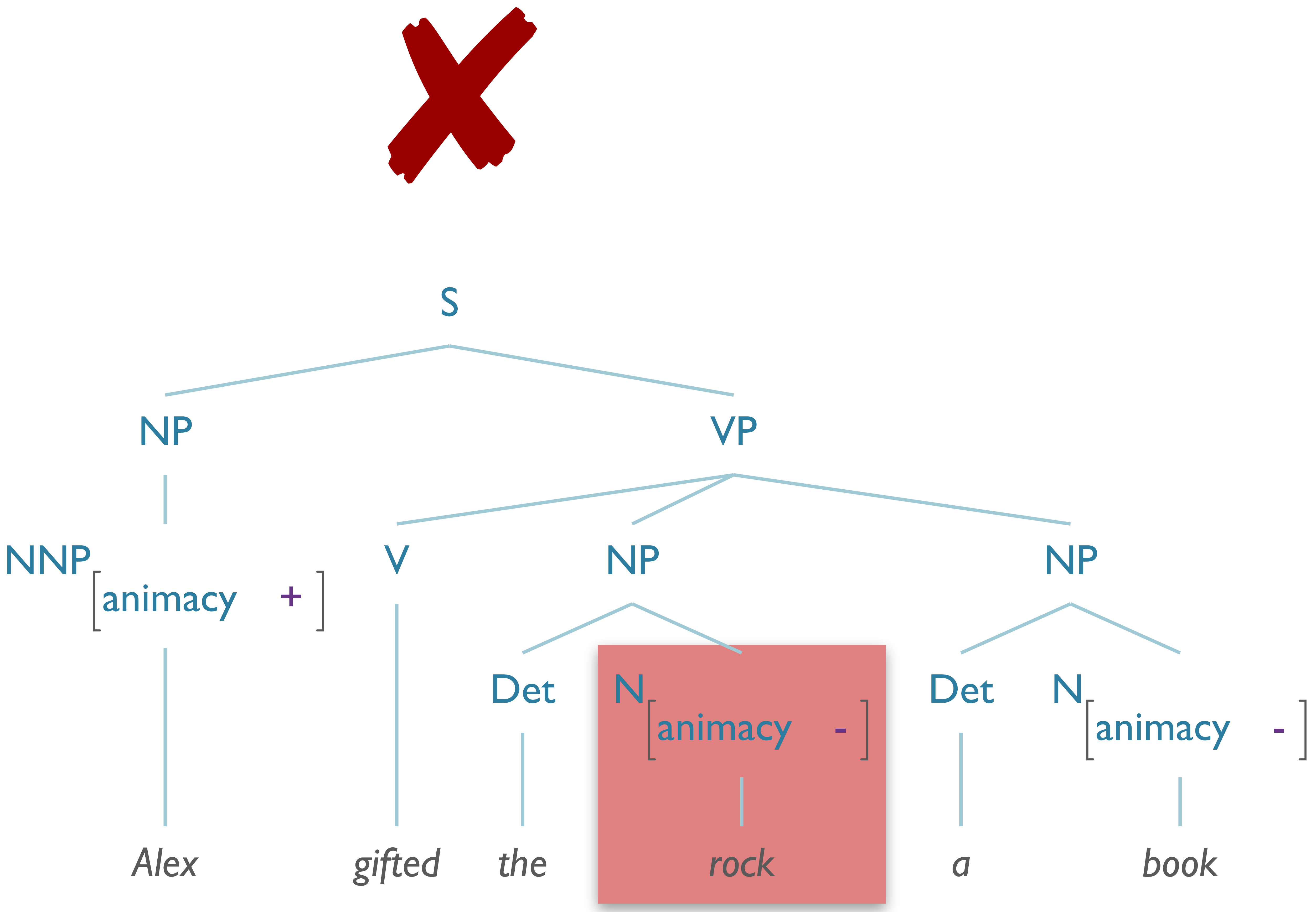
✓





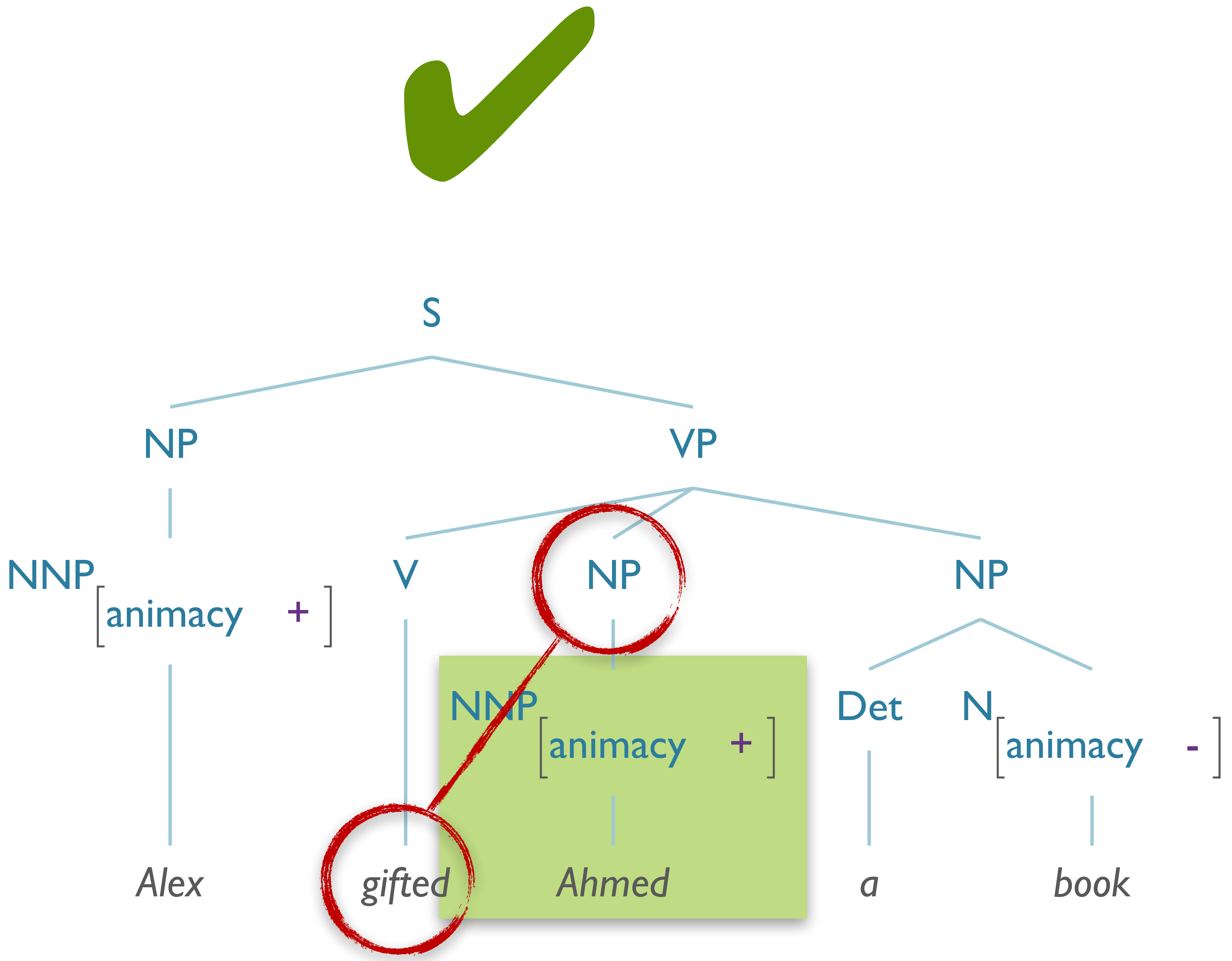
Feature Grammar Practice

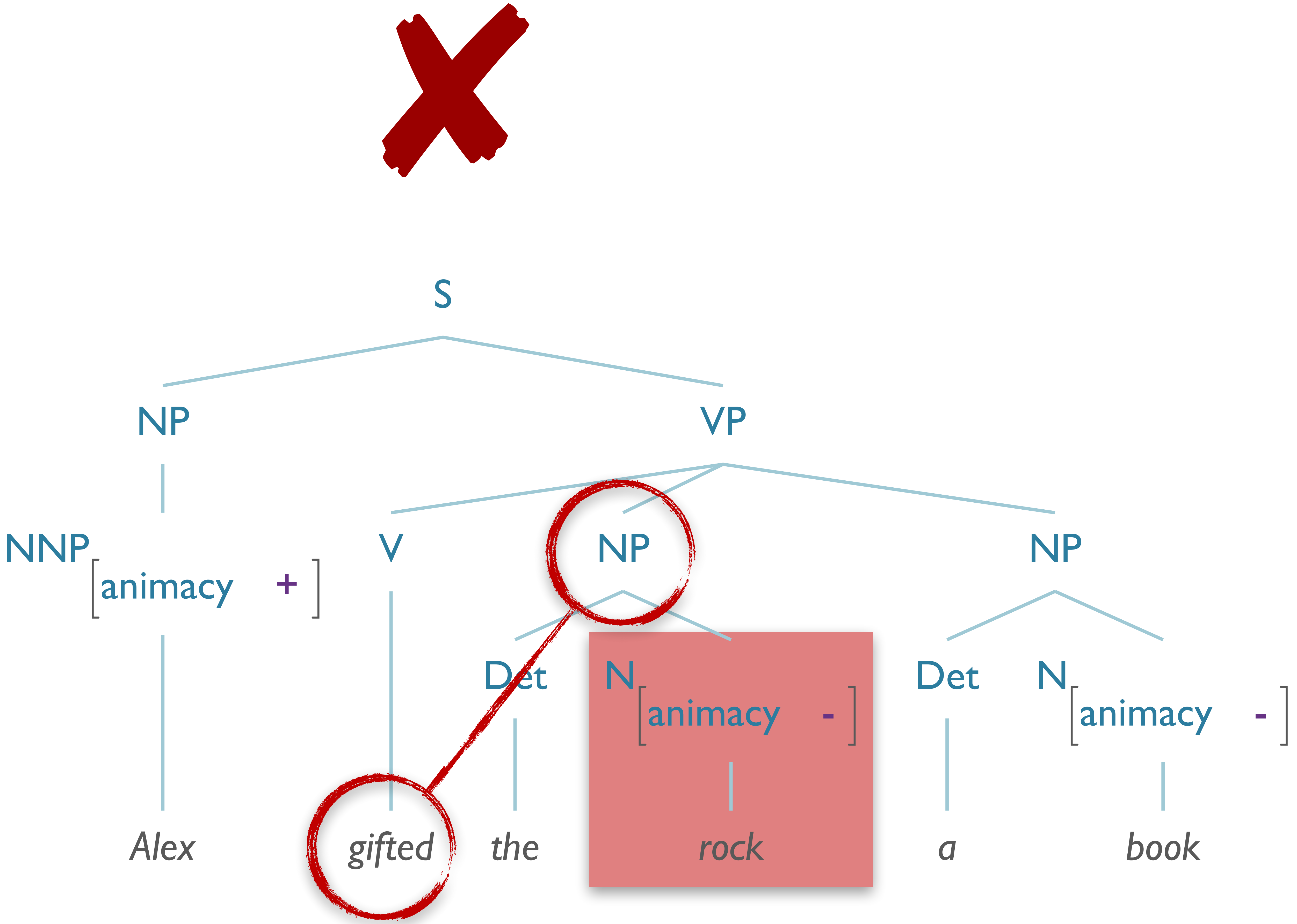




Feature Grammar Practice

✓ ✗





Practice Task

- Modify the initial grammar to incorporate animacy in such a way that you get the right results:
  - Alex gifted Ahmed a book
  - * Alex gifted the rock a book