Common Nouns

Lecture 09

February 8, 2024

Announcements:

This lecture is supplemented by the following readings:

- Heim & Kratzer: Ch.4 (61–63)

Your fourth homework assignment is available and is due on February 20th.

1 Introduction

We have up to this point built a semantic system for interpreting sentences that relies on the following set of rules:

(1) **Functional Application (FA)**
If X is a node that has two daughters, Y and Z, and if ⟦ Y ⟧ is a function whose domain contains ⟦ Z ⟧, then ⟦ X ⟧ = ⟦ Y ⟧(⟦ Z ⟧).

(2) **Non-Branching Nodes (NN) Rule**
If X is a non-branching node that has Y as its daughter, then ⟦ X ⟧ = ⟦ Y ⟧

(3) **Terminal Nodes (TN) Rule**
If X is a terminal node, then ⟦ X ⟧ is specified in the lexicon.

These rules are able to interpret structures built from lexical entries like the following:

(4) **Proper Nouns as entities**
⟦ NAME ⟧ = the thing referred to with NAME

(5) **Intransitive verbs as $\langle e, t \rangle$ functions**
⟦ $\text{VERB}_{intrans}$ ⟧ = $[\lambda x : x \in D_e . T \textit{ iff } x \text{ VERBs}]$

(6) **Transitive verbs as $\langle e, \langle e, t \rangle \rangle$ functions**
⟦ VERB_{trans} ⟧ = $[\lambda x : x \in D_e . [\lambda y : y \in D_e . T \textit{ iff } y \text{ VERBs } x]$

(7) **Adjectives as $\langle e, t \rangle$ functions**
⟦ ADJECTIVE ⟧ = $[\lambda x : x \in D_e . T \textit{ iff } x \text{ is ADJECTIVE}]$

We also introduced the idea of identity functions, which return their argument as their value. We saw the utility in treating the copula verb *is* as such an expression of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$.

(8) **The copula as a $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ identity function**
$[\![\text{ is }]\!] = [\ \lambda f\ :\ f \in D_{\langle e,t\rangle}\ .\ f\ ]$

During our last meeting we turned our attention to the semantics of adjectives, adjectival modification, and more complex nominal expressions. We have started with simple cases included those below:

(9) Felix is **gray**.

(10) Felix is **a cat**.

Having provide a way to interpret sentences with adjectival predicates, we turn to sentences with 'common nouns', such as *a cat*, serving as **predicate nominals**. We will find that such expressions can also be treated as $\langle e,t\rangle$ functions.

(11) **The semantics of common nouns**

a. **Function notation**
$[\![\text{ cat }]\!] = f\ :\ D_e \rightarrow D_t$
for every $x \in D_e$, $f(x) = T$ *iff* x is a cat

b. **Lambda Notation**
$[\![\text{ cat }]\!] = [\ \lambda x\ :\ x \in D_e\ .\ T$ *iff* x is a cat $]$

Interpreting sentences like those in (10) will also require us to have a lexical entry for the determiner *a*. Following familiar reasoning, we will see good reason for assuming that this element too is semantically vacuous, and can be treated as an **identity function**.

(12) **The determiner *a* as an identity function**

a. **Function Notation**
$[\![\text{ a }]\!] = g\ :\ D_{\langle e,t\rangle} \rightarrow D_{\langle e,t\rangle}$
for every $f \in D_{\langle e,t\rangle}$, $g(f) = f$

b. **Lambda Notation**
$[\![\text{ a }]\!] = [\ \lambda f\ :\ f \in D_{\langle e,t\rangle}\ .\ f\ ]$

With these pieces in place, we will soon be able to turn to more complex expressions, including those below.

(13) Felix is a gray cat.

(14) Travis likes the tall singer.

(15) Every professor dances.

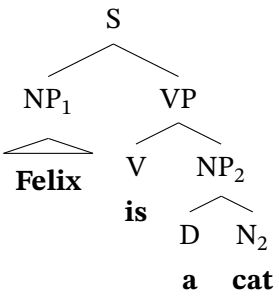
2 The Semantics of the Determiner

We are interested in developing our semantic system that it is able to compute the meaning of sentences with predicate nominals, like (16).

(16) Felix is a cat.

Let's assume that (16) has the syntactic representation in (17). The copula will be treated as a verb that combines with an NP that is headed by the noun *cat* and contains the determiner *a*.

(17)



In order for our system to interpret this sentence, we will need a lexical entry for the determiner *a* and the common noun *cat*. We will also need to ensure that our semantic system can use these lexical entries, alongside the others, to compute the meaning of the NP_2 and VP nodes.

We will start by determining a lexical entry for the determiner *a*.

2.1 The Semantic Type of the Determiner

1. Known Semantic Types. We start by listing out the semantic types we know and annotating the syntactic representation with this information.

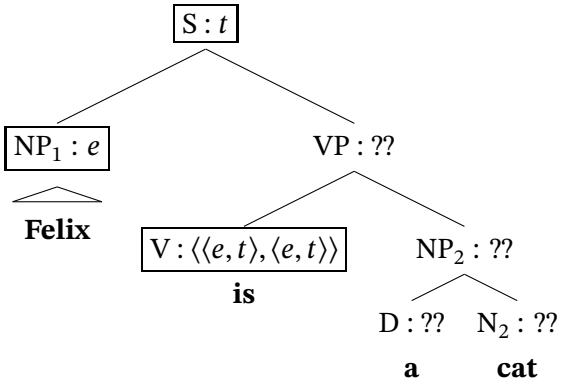
(18) **Known Semantic Types in (43)**

a. $[\![\text{S}]\!] \in D_t$

b. $[\![\text{Felix}]\!] \in D_e$

c. $[\![\text{is}]\!] \in D_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$

(19)



2. Reasoning out the Semantic Type of VP. We appeal to these known semantic types, as well as the rules of composition, to determine that the VP must be of type $\langle e, t\rangle$.

- The sentence must ultimately be an expression of type t.
- The subject NP_1 *Felix* is an expression of type e.

- The S node fits the description for Functional Application. So, the extension of the subject must combine with the extension of the VP to return the extension of the sentence.

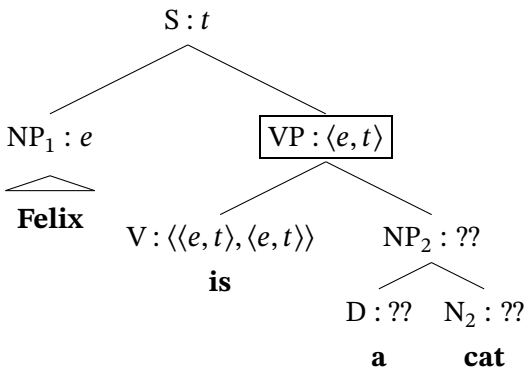
  (20) $[\![\text{Felix}]\!] + [\![\text{VP}]\!] = [\![\text{S}]\!]$

- Because the NP *Felix* is an expression of type e, the extension of the VP must be a function that takes *Felix* as an argument and gives back a truth value.

  (21) $f : D_e \rightarrow D_t$

- Thus, the VP must be an expression of type $\langle e, t \rangle$.

  (22)



3. Reasoning out the Semantic Type of the NP_2. We appeal to the known semantic types, as well as the rules of composition, to determine that the NP_2 must be of type $\langle e, t \rangle$.

- The VP node fits the structural description for Functional Application. So, the extension of the copula must combine with the extension of the NP_2 to return the extension of the VP.

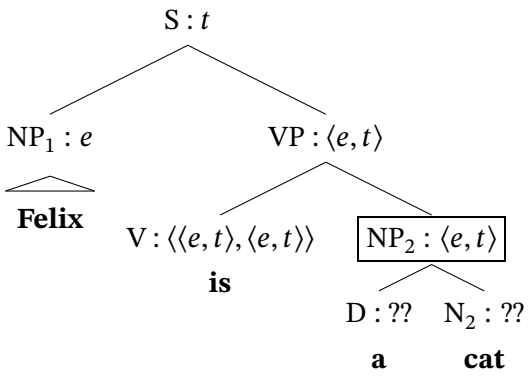
  (23) $[\![\text{is}]\!] + [\![\text{NP}_2]\!] = [\![\text{VP}]\!]$

- Because the copula is a function of type $\langle \langle e, t \rangle, \langle e, t \rangle \rangle$, the extension of the copula must take the NP_2 as its argument to return the type $\langle e, t \rangle$ extension of the VP as its value.

  (24) $[\![\text{is}]\!]([\![\text{NP}_2]\!]) = [\![\text{VP}]\!]$

- Thus, the NP_2 must be an expression of type $\langle e, t \rangle$.

  (25)



4. Reasoning out the Semantic Type of the D. We appeal to the known semantic types, as well as the rules of composition, to determine that the NP can be treated as an expression of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$.

- The NP node was just determined to be an expression of type $\langle e,t\rangle$.
- The NP node fits the structural description for Functional Application. So, the extension of the determiner must combine with the extension of *cat* to return the extension of the NP.

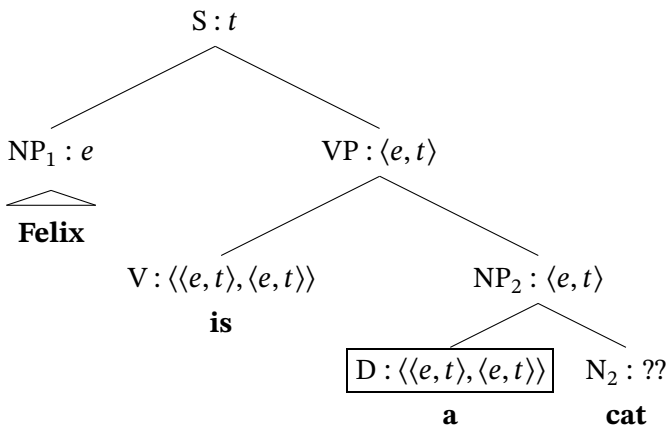
(26) $[\![\text{a}]\!] + [\![\text{cat}]\!] = [\![\text{NP}]\!]$

- Because the NP is an expression of $\langle e,t\rangle$, either *a* or *cat* is a type $\langle e,\langle e,t\rangle\rangle$ expression or a type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ expression, given our current inventory of semantic types.
- Neither *a* nor *cat* is plausibly treated as an expression of type *e*. So, neither is plausibly treated as an expression of type $\langle e,\langle e,t\rangle\rangle$.
- This means that the extension of either *a* or *cat* must be a function that takes an $\langle e,t\rangle$ function and returns a type $\langle e,t\rangle$ function as its argument, which is the type of the identity function.

(27) $f : D_{\langle e,t\rangle} \rightarrow D_{\langle e,t\rangle}$

- As a contentful part of the predicate, *cat* is not plausibly treated as an identity function.
- Thus, the extension of the determiner could be an identity function of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$.

(28)



2.2 The Determiner as an Identity Function

The idea under consideration is that the determiner *a*, much like we determined for the copula, is **semantically vacuous**. That is, it appears in the sentence for purely syntactic reasons and does not contribute to the meaning of the expression.

Much like we saw with the copula, there are many languages, including English, that do not require the determiner *a* or lack it altogether.

(29) **Russian Null-Copula Constructions**

a. Felix sʲirɨj
Felix gray.MS
'Felix is gray.'

b. Felix kot
Felix cat
'Felix is a cat.'

(30) **English Headline-ese**
BREAKING NEWS: Man bites dog!

Given some familiar reasoning about facts like those above, let us treat the determiner *a* in this construction also as an **identity function**.

(31) **Identity Function**
A function that takes an argument x and returns x as its value.

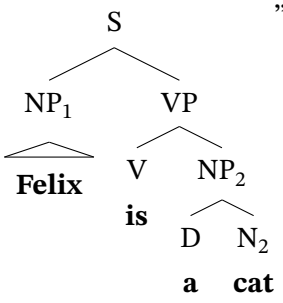
As an instance of the $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ identity function, the extension of *a* will be the same as the copula:

(32) **The determiner as an identity function**
$[\![\text{a}]\!] = [\, \lambda f \,:\, f \in D_{\langle e,t\rangle} \,.\, f\,]$

3 The Semantics of Nouns

We can now proceed with our goal of deriving the following truth-conditional statement:

(33) “

(34)


” is *T iff* Felix is a cat

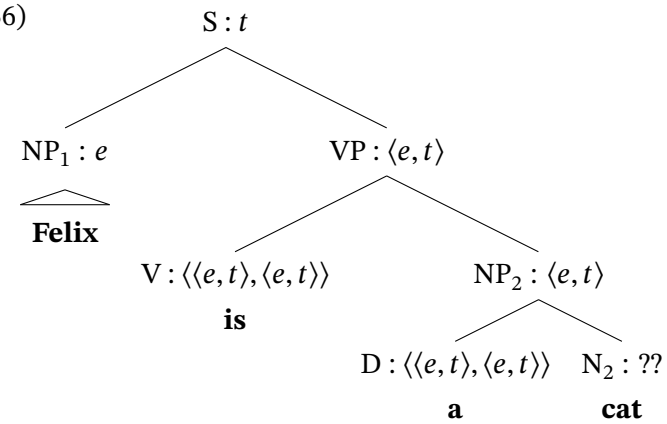
This will require that we determine the extension of the noun *cat* and demonstrate that our semantic system is capable of deriving these truth conditions.

3.1 The Semantic Type of Nouns

1. Known Semantic Types. We start by listing out the semantic types we know and annotating the syntactic representation with this information.

(35) **Known Semantic Types**

a. $[\![\text{S}]\!] \in D_t$
b. $[\![\text{Felix}]\!] \in D_e$
c. $[\![\text{is}]\!] \in D_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$
d. $[\![\text{a}]\!] \in D_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$

(36)


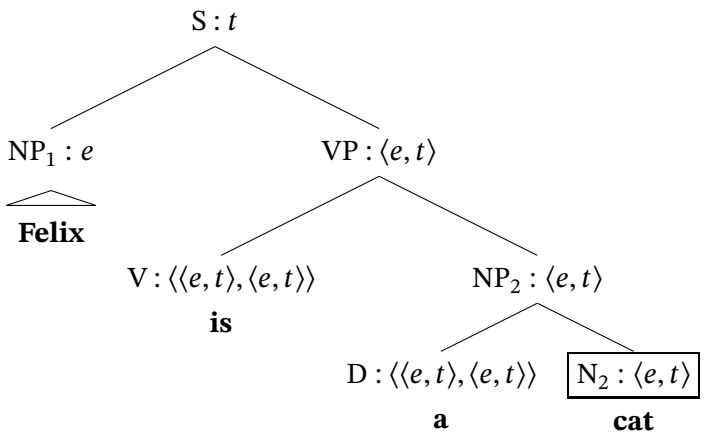
2. Reasoning out the Semantic Type of N. We appeal to our known semantic types, as well as the rules of composition, to determine that the N_2 must be of type $\langle e, t\rangle$.

- We just determined in section 2.1 that the NP_2 node must be an expression of type $\langle e, t\rangle$.
- The determiner *a* is to be treated as an identity function that returns a value with the same semantic type as its argument.
- The NP_2 node has the structural description specified by Functional Applications. So, the extension of the determiner must combine with the extension of the noun to return the extension of the NP_2.

(37) $[\![\text{a}]\!] + [\![\text{cat}]\!] = [\![\text{NP}_2]\!]$

- Because the NP_2 is an expression of type $\langle e, t\rangle$, the extension of the N_2 must be a type $\langle e, t\rangle$ function that can serve as an argument of the type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$ determiner.
- Thus, the N_2 must be an expression of type $\langle e, t\rangle$.

(38)



3.2 The Extension of Nouns

We now know what kind of expression the N_2 *cat* must be. It is a type $\langle e, t\rangle$ function, meaning it maps entities to truth values.

We therefore need to develop a lexical entry for *cat* that does the following:

- assigns as its extension a function of type $\langle e, t\rangle$ and
- allows our system to derive the following truth-conditional statement:

(39) "Felix is a cat" is *T iff* Felix is a cat

Some Preliminary Reasoning about the Extension of N. On the basis of what has preceded, we can appreciate that the extension of the VP will be equivalent to the extension of the N_2.

- We know from section 2.1 that the type $\langle e, t\rangle$ extension of the VP will be equivalent to the type $\langle e, t\rangle$ extension of the NP_2 on account of the copula's extension as an identity function.

- We know from section 3.1 that the type $\langle e, t\rangle$ extension of the NP will be equivalent to the type $\langle e, t\rangle$ extension of the N_2 on account of the determiner's extension as an identity function.
- From these two points it follows that the extension of the VP will be equivalent to the extension of the N_2.

(40) **The extension of the VP is identical to the extension of the N_2**

i.	$[\![\ [_{VP}$ is a cat $]\ ]\!]$	=	(by FA)
ii.	$[\![$ is $]\!]([\![\ NP_2\ ]\!])$	=	(by TN)
iii.	$[\ \lambda f : f \in D_{\langle e,t\rangle} . f]([\![\ NP_2\ ]\!])$	=	(by LC)
iv.	$[\![\ NP_2\ ]\!]$	=	(by FA)
v.	$[\![\ D\ ]\!]([\![$ cat $]\!])$	=	(by TN)
vi.	$[\ \lambda f : f \in D_{\langle e,t\rangle} . f]([\![$ cat $]\!])$	=	(by LC)
vii.	$[\![$ cat $]\!]$		

Reasoning out the Extension of the N. Given the results above, determining the meaning of the type $\langle e, t\rangle$ VP *is a cat* will deliver the meaning of the N_2 *cat*.

- Considering sentences like those in (41), the key generalization is that the VP *is a cat* systematically combines with entities to generate sentences with the meaning in (42).

(41) a. "Felix is a cat" is *T iff* Felix is a cat
b. "Mittens is a cat" is *T iff* Mittens is a cat
c. "Luna is a cat" is *T iff* Luna is a cat

(42) $[\![\ [_S$ NAME is a cat $]\ ]\!] = T$ *iff* NAME is a cat

- We know from section 2.1 that $[\![\ [_{VP}$ is a cat $]\ ]\!]$ is of type $\langle e, t\rangle$. Consequently, our rule of FA entails that the following equivalency holds:

(43) $[\![\ [_S$ NAME is a cat $]\ ]\!] = [\![\ [_{VP}$ is a cat $]\ ]\!]([\![$ NAME $]\!])$

- If follows from the previous two points that:

(44) $[\![\ [_{VP}$ is a cat $]\ ]\!]([\![$ NAME $]\!]) = T$ *iff* NAME is a cat

- Therefore, $[\![\ [_{VP}$ is a cat $]\ ]\!]$ is an $\langle e, t\rangle$ function that takes an argument x and yields *T iff* x is a cat.

(45) $[\![\ [_{VP}$ is a cat $]\ ]\!] = [\lambda x : x \in D_e . T$ *iff* x is a cat $]$

- Because $[\![\ [_{VP}$ is a cat $]\ ]\!] = [\![\ [_N$ cat $]\ ]\!]$, then:

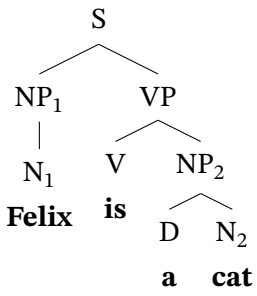
(46) $[\![$ cat $]\!] = [\lambda x : x \in D_e . T$ *iff* x is a cat $]$

4 Deriving the Meaning of Sentences with Predicate Nominals

Let us now check to be sure that our semantic system correctly makes use of these lexical entries to the derive the correct truth-conditional statement.

Our syntactic assumptions regarding the sentence at hand are as follows:

(47) a. Felix is a cat.

b. 

The lexical entries for the components parts of the sentence that are stored in the lexicon include:

(48) **Lexical entries**

a. $[\![\text{Felix}]\!] = \text{Felix}$

b. $[\![\text{is}]\!] = [\, \lambda f \,:\, f \in D_{\langle e,t \rangle} \,.\, f \,]$

c. $[\![\text{a}]\!] = [\, \lambda f \,:\, f \in D_{\langle e,t \rangle} \,.\, f \,]$

d. $[\![\text{cat}]\!] = [\lambda x \,:\, x \in D_e \,.\, T \textit{ iff } x \text{ is a cat} \,]$

Our system currently employs the set of semantic rules listed below:

(49) **Functional Application (FA)**
If X is a node that has two daughters, Y and Z, and if $[\![\text{Y}]\!]$ is a function whose domain contains $[\![\text{Z}]\!]$, then $[\![\text{X}]\!] = [\![\text{Y}]\!]([\![\text{Z}]\!])$.

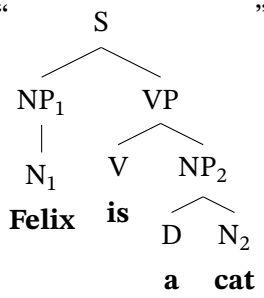
(50) **Non-Branching Nodes (NN) Rule**
If X is a non-branching node that has Y as its daughter, then $[\![\text{X}]\!] = [\![\text{Y}]\!]$

(51) **Terminal Nodes (TN) Rule**
If X is a terminal node, then $[\![\text{X}]\!]$ is specified in the lexicon.

We can now see how the proof of the truth conditions can proceed with subproofs of the major constituents of the sentence.

(52) **Truth-conditional Statement of *Felix is gray***
Felix is a cat is *T iff* Felix is a cat

(53) **Calculation of the Truth Conditions of *Felix is a cat***

i.	"Felix is a cat"	is *T iff*	(by syntax)
ii.	"  "	is *T iff*	(by notation)
iii.	$[\![\text{S}]\!]$	$= T$	
iv.	*Calculation of* $[\![\text{NP}_1]\!]$		
	a. $[\![\text{NP}_1]\!]$	=	(by NN)
	b. $[\![\text{Felix}]\!]$	=	(by TN)
	c. Felix		
v.	*Calculation of* $[\![\text{NP}_2]\!]$		
	a. $[\![\text{NP}_2]\!]$	=	(by FA, LE)
	b. $[\![\text{a}]\!]([\![\text{cat}]\!])$	=	(by TN)
	c. $[\, \lambda f : f \in D_{\langle e,t \rangle} \,.\, f]([\![\text{cat}]\!])$	=	(by TN)
	d. $[\, \lambda f : f \in D_{\langle e,t \rangle} \,.\, f]([\lambda x : x \in D_e \,.\, T \textit{ iff } x \text{ is a cat}])$	=	(by LC)
	e. $[\lambda x : x \in D_e \,.\, T \textit{ iff } x \text{ is a cat}]$		
vi.	*Calculation of* $[\![\text{VP}]\!]$		
	a. $[\![\text{VP}]\!]$	=	(by FA)
	b. $[\![\text{is}]\!]([\![\text{NP}_2]\!])$	=	(by TN)
	c. $[\, \lambda f : f \in D_{\langle e,t \rangle} \,.\, f]([\![\text{NP}_2]\!])$	=	(by iv.)
	d. $[\, \lambda f : f \in D_{\langle e,t \rangle} \,.\, f]([\lambda x : x \in D_e \,.\, T \textit{ iff } x \text{ is a cat}])$	=	(by LC)
	e. $[\lambda x : x \in D_e \,.\, T \textit{ iff } x \text{ is a cat}]$		
vii.	$[\![\text{S}]\!]$	$= T$ *iff*	(by FA, iv., vi.)
viii.	$[\![\text{VP}]\!]([\![\text{NP}_1]\!])$	$= T$ *iff*	(by vi.)
ix.	$[\lambda x : x \in D_e \,.\, T \textit{ iff } x \text{ is a cat}]([\![\text{NP}_1]\!])$	$= T$ *iff*	(by iv.)
x.	$[\lambda x : x \in D_e \,.\, T \textit{ iff } x \text{ is a cat}](\text{Felix})$	$= T$ *iff*	(by LC)
xi.	Felix is a cat		(nailed it)

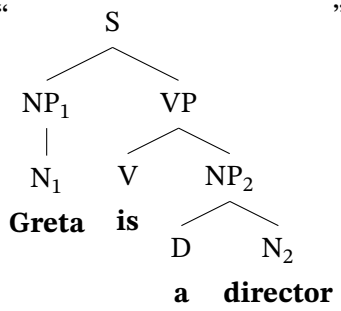
Thus, we conclude that (53i) *iff* (53xi), or **"Felix is a cat" is *T iff* Felix is a cat**.

By repeating the process made explicit in this handout, we would find that a similar lexical entry would work for a wide range of nouns in English and beyond. For example, this system will similarly derive the following truth-conditional statements for several predicate nominals, which seem to be correct:

(54) a. "Felix is a **dog**" is *T iff* Felix is a **dog**.
b. "Felix is a **boy**" is *T iff* Felix is a **boy**.
c. "Felix is a **bachelor**" is *T iff* Felix is a **bachelor**.

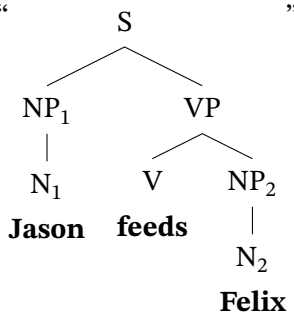
5 Practice

Exercise. Please provide a semantic proof of the truth-conditional statement below:

(55) "  " is *T iff* Greta is a director

In order to do this, we will need to provide the extensions of the lexical items and provide a step-by-step proof of these reported truth conditions.

Exercise. Please provide a semantic proof of the truth-conditional statement below:

(56) "  " is *T iff* Jason feeds Felix

In order to do this, we will need to provide the extensions of the lexical items and provide a step-by-step proof of these reported truth conditions.