Denotation of Predicates

- Assume a world w_5, where D = $\{$Ann, Betty, Connor$\}$, Betty and Connor are smokers, but Ann isn't.

- Set notation

  $[\![\text{smoke}]\!]^{w_5} = \{\text{Betty, Connor}\} = \{x \ : \ x \text{ smokes in } w_5\}$

- Functional notation

  $$[\![\text{smoke}]\!]^{w_5} = \begin{bmatrix} \text{Ann} \rightarrow 0 \\ \text{Betty} \rightarrow 1 \\ \text{Connor} \rightarrow 1 \end{bmatrix}$$

  "the function f from individuals to truth values such that for all $d \in D$, $f(d) = 1$ iff d smokes in w_5"

Function Application

- Definition

  A semantic rule for interpreting a syntactic structure with two branches: one branch is interpreted as a function, and the other branch is interpreted as an argument of the function.

  $$[\![\ [_{A}\ B\ \ C\ ]\ ]\!]^w = [\![B]\!]^w([\![C]\!]^w)$$

- Compositional interpretation of *Betty smokes* in w_5?

Syntax (LF)	Interpretation	Interpretation
[$_{TP}$ [$_{NP}$ [$_{N}$ Betty]] [$_{VP}$ [$_{V}$ smokes]]]	1 b $\begin{bmatrix} \text{Ann} \rightarrow 0 \\ \text{Betty} \rightarrow 1 \\ \text{Connor} \rightarrow 1 \end{bmatrix}$	1 iff b smokes in w_5 b the function f from individuals to truth values s.t. for all $d \in D$, $f(d)$ = 1 iff d smokes in w_5

Syntax of λ-operator

- If ϕ is a well-formed formula and x a variable, $\lambda x[\phi]$ is a one-place predicate.
- Expressions like $\lambda x[\phi]$ are called 'λ-abstracts' or 'λ-expressions.'
- How to read λ-expressions (informally)

  $\lambda x[\phi]$: "the property of being an x such that ϕ"

  $\lambda x[\text{smoke}(x)]$: "the property of being an x such that x smokes"

  $\lambda y[\text{snore}(y)]$: "the property of being a y such that y smokes"

  $\lambda x[\exists y[\text{love}(y, x)]]$: "the property of being an x s.t. for some y, y loves x"

  $\lambda y[\exists x[\text{love}(y, x)]]$: "the property of being a y s.t. for some x, y loves x"

- In $\lambda x[\phi]$, x is a variable bound by λ, and ϕ is the scope of that occurrence of the λ-operator.

$$\lambda x[\text{smoke}(x)]$$

$$\lambda x \qquad \text{smoke}(x)$$

λ-conversion

- We obtain a well-formed formula by applying a λ-expression to a term.

  λx[smoke(x)](j) = "J has the property of being x s.t. x smokes" = smoke(j)

  $\lambda y[\exists x$[love$(y, x)]](\text{m})$ = "M has the property of being y s.t. for some x, y loves x" = $\exists x$[love(m, x)]

- λ-conversion: $\lambda x[\phi](t) \leftrightarrow \phi[t/x]$

- Watch out! When applying λ-conversion, we must make sure that there is no variable clash.

  $\boxed{\lambda y[\exists x[\text{love}(y, x)]](x) \neq \exists x[\text{love}(x, x)]}$

  $\lambda y[\exists x$[love$(y, x)]](x)$ = "$g(x)$ has the property of being y s.t. for some x, y loves x" = "He loves someone"

  $\exists x$[love(x, x)] = "for some x, x loves x" = "Someone loves himself"

  So, choose your variables carefully.

  $\boxed{\lambda y[\exists x[\text{love}(y, x)]](z) = \exists x[\text{love}(z, x)]}$

Exercise in λ-conversion

Exercise 1 in p. 394 from *Meaning and Grammar*.

1. $\lambda x[\exists z[\lambda y[K(x,y)](z) \wedge R(z,x)]](j)$

2. $\lambda y[\lambda x[K(x,y)](j)](m)$

3. $\lambda z[\lambda x[[K(x,z) \wedge R(x,z)] \vee R(z,x)](j)](m)$

4. $\exists y[\lambda z[\lambda x[B(x) \rightarrow \exists w[R(x,w)]](j) \wedge \lambda x[B(x) \wedge Q(x)](z)](y)]$

Semantics of λ-operator

- $[\![\lambda x[\phi]]\!]^{w,g}$

  = a function f from individuals to truth values such that for all $d \in D$, $f(d) = 1$ iff $[\![\phi]\!]^{w,g[d/x]} = 1$

  = $\{d \in D : [\![\phi]\!]^{w,g[d/x]} = 1\}$

- $[\![\lambda x[\text{smoke}(x)]]\!]^{w,g}$

  = a function f from individuals to truth values such that for all $d \in D$, $f(d) = 1$ iff $[\![\text{smoke}(x)]\!]^{w,g[d/x]} = 1$

  = $\{d \in D : [\![\text{smoke}(x)]\!]^{w,g[d/x]} = 1\}$

Mapping Syntax to Logical Representation Compositionally

- With the introduction of λ-operator to IPC, we can map syntactic structures to IPC logical representations compositionally, which then can receive truth-conditional interpretation using model-theoretic/possible-worlds semantics.

Syntax (LF)	Logical Representation	Truth Conditional Interpretation w.r.t. an arbitrary w
TP NP VP N V Betty smokes	$\lambda x[\text{smoke}'(x)](\text{b})$ $= \text{smoke}'(\text{b})$ b $\quad \lambda x[\text{smoke}'(x)]$	1 iff B smokes in w B $\quad$ the function f from individuals to truth values s.t. for all $d \in D$, $f(d) = 1$ iff d smokes in w

Syntax of a Fragment of English (F3) Again

1. (a) TP $\longrightarrow$ NP T$'$
   (b) T$'$ $\longrightarrow$ T VP
   (c) TP $\longrightarrow$ TP conj TP
   (d) TP $\longrightarrow$ neg TP
   (e) T $\longrightarrow$ Past, Pres, Fut
   (f) VP $\longrightarrow$ V$_t$ NP
   (g) VP $\longrightarrow$ V$_i$
   (h) VP $\longrightarrow$ V$_{dt}$ NP PP[to]
   (i) NP $\longrightarrow$ Det N$_c$
   (j) NP $\longrightarrow$ N$_p$
   (k) PP[to] $\longrightarrow$ to NP
   (l) Det $\longrightarrow$ the, a, every
   (m) N$_p$ $\longrightarrow$ Frodo, Smeagol, Deagol, Sam, Aragorn, ... he$_1$, ..., he$_n$, ...
   (n) N$_c$ $\longrightarrow$ book, fish, man, hobbit, ...
   (o) V$_i$ $\longrightarrow$ be intelligent, be hungry, smoke, ...
   (p) V$_t$ $\longrightarrow$ destroy, kill, read, ...
   (q) V$_{dt}$ $\longrightarrow$ give, introduce, ...
   (r) Conj $\longrightarrow$ and, or
   (s) Neg $\longrightarrow$ not

2. Rule for Quantifier Raising (QR)
   [$_{TP}$ X NP Y] $\Rightarrow$ [$_{TP}$ NP$_i$ [$_{TP}$ X t$_i$ Y]]

3. Rule for Tense/Modal/Neg Raising (TR)
   [$_{TP}$ NP X VP] $\Rightarrow$ [$_{TP}$ X [$_{TP}$ NP VP]], where X = T or Neg

IPC Translation of F3

1. For any word or phrase α of F3, α' is its IPC translation.

2. Given a lexical item α,

F_3 category	IPC type	IPC translation	Examples
N_p	constants variables	α' x_n	Sam $\Rightarrow$ Sam$'$ he$_1$ $\Rightarrow$ x_1
V_i	1-place predicate	$\lambda x[\alpha'(x)]$	smoke $\Rightarrow$ λx[smoke$'(x)$]
N_c	1-place predicate	$\lambda x[\alpha'(x)]$	song $\Rightarrow$ λx[song$'(x)$]
V_t	2-place predicate	??	like $\Rightarrow$??

3. not$'$ = $\neg$

   and$'$ = $\wedge$

   or$'$ = $\vee$

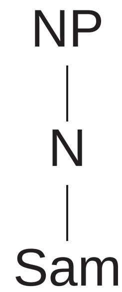
   Fut$'$ = **F**

   Past$'$ = **P**

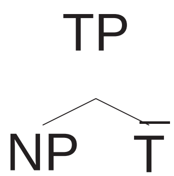
   t$_n$ = x_n, where t$_n$ is a trace or a pronoun

4. [

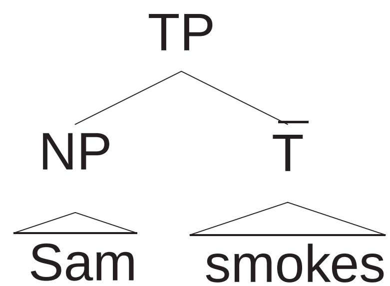
]′ = B′

[

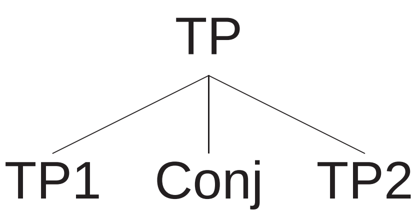
]′ = Sam′

5. [

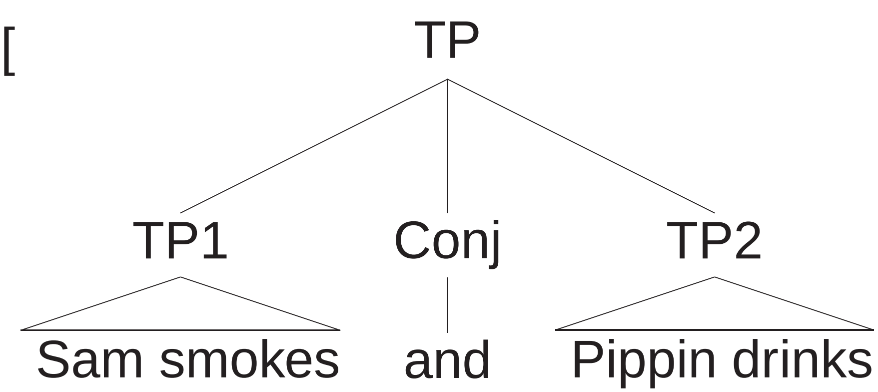
]′ = $\overline{\text{T}}$′(NP′)

[

] = smoke′(Sam′)

6. [

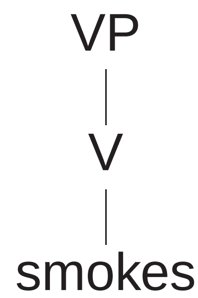
]′ = TP1′ Conj′ TP2′

[

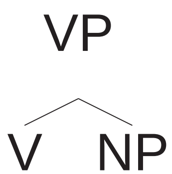
]′ = smoke′(Sam′) ∧ drink′(Pippin′)

7. [

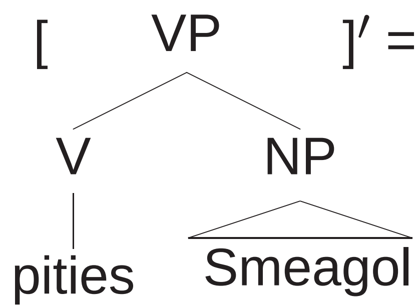
]$' = \lambda x[\text{V}'(x)]$

[

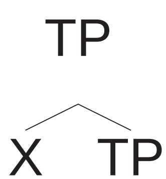
]$' = \lambda x[\text{smoke}'(x)]$

8. [

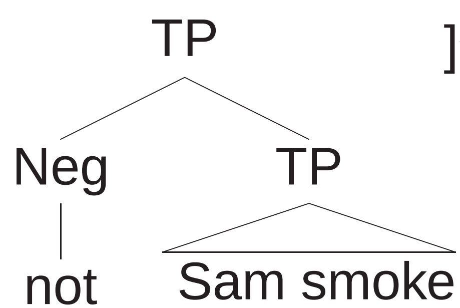
]$' = \lambda x[\text{V}'(x, \text{NP}')]$

[

]$' = \lambda x[\text{pity}'(x, \text{Smeagol}')]$

9. [

]$' = \text{X}'\ \text{TP}'$

[

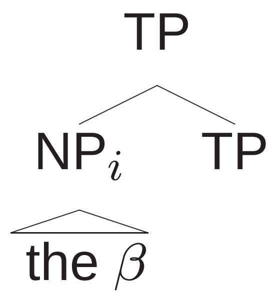
]$' = \neg\text{smoke}'(\text{Sam}')$

10. (i) $\left[\ [_{\text{TP}}\ [_{\text{NP}_i}\ \text{every}\ \beta]\ [_{\text{TP}}]\ ]\ \right]' = \forall x_i[\beta'(x_i) \rightarrow \text{TP}']$

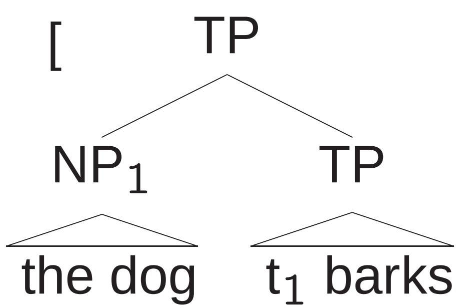
$\left[\ [_{\text{TP}}\ [_{\text{NP}_i}\ \text{every dog}]\ [_{\text{TP}}\ \text{t}_i\ \text{barks}]\ ]\ \right]' = \forall x_i[\text{dog}'(x_i) \rightarrow \text{bark}'(x_i)]$

(ii) $\left[\ [_{\text{TP}}\ [_{\text{NP}_i}\ \text{a}\ \beta]\ [_{\text{TP}}]\ ]\ \right]' = \exists x_i[\beta'(x_i) \wedge \text{TP}']$

$\left[\ [_{\text{TP}}\ [_{\text{NP}_i}\ \text{a dog}]\ [_{\text{TP}}\ \text{t}_i\ \text{barks}]\ ]\ \right]' = \exists x_i[\text{dog}'(x_i) \wedge \text{bark}'(x_i)]$

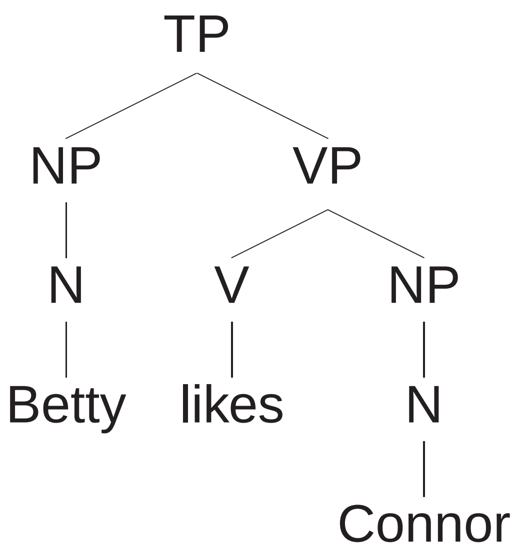
(iii) [

$$]' = \exists x_i[\beta'(x_i) \wedge \forall y[\beta'(y) \rightarrow x_i = y] \wedge \mathrm{TP}']$$

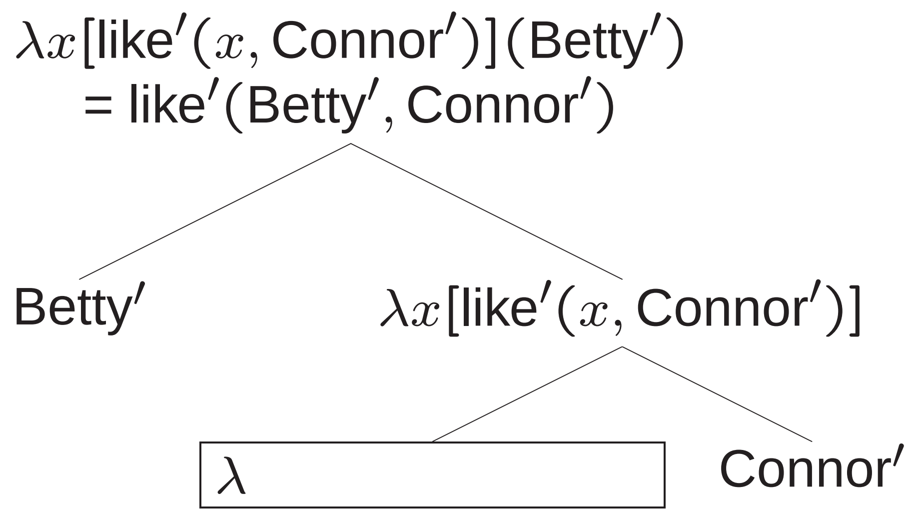
[

$$]' = \exists x_1[\mathrm{dog}'(x_1) \wedge \forall y[\mathrm{dog}'(y) \rightarrow x_1 = y] \wedge \mathrm{bark}'(x_1)]$$

Translating a Transitive Verb to a λ-expression



$\Rightarrow$



Exercise in Translating English to Logical Representation

1. Frodo respects Gandalf.

2. Bilbo must not kill Gollum.

3. Gandalf likes every hobbit.

4. Every hobbit knows a song.