Formal Semantics of Natural Language

Philippe de Groote

Lambda Calculus

Lambda-Calculus & Combinatory Logic

Haskell Curry (1900-1982)

Alonzo Church (1903-1995)

λ-Notation

$$\text{“}2x + y\text{”}$$

λ-Notation

$$\text{“}2x + y\text{”}$$

$f(x) = 2x + y$

λ-Notation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

λ-Notation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

λ-Notation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

$$\lambda y.\, 2x + y$$

λ-Notation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

$$\lambda y.\, 2x + y$$

$$f(x, y) = 2x + y$$

λ-Notation

$$\text{“}2x + y\text{”}$$

$$f(x) = 2x + y$$

$$\lambda x.\, 2x + y$$

$$f(y) = 2x + y$$

$$\lambda y.\, 2x + y$$

$$f(x, y) = 2x + y$$

$$\lambda xy.\, 2x + y$$

λ-Terms

Let $\mathscr{C}$ be a set of symbols whose elements are called *constants*, and let $\mathscr{X}$ be a countably infinite set of symbols, disjoint from $\mathscr{C}$, whose elements are called λ-*variables*. The set of λ-terms is inductively defined as follows:

- every $c \in \mathscr{C}$ is a λ-term;
- every $x \in \mathscr{X}$ is a λ-term;
- if t is a λ-term and x is a λ-variable then $(\lambda x.\, t)$ is a λ-term;
- if t and u are λ-terms then $(t\, u)$ is a λ-term.

λ-Terms

Abstraction

$$(\lambda x.\, t)$$

λ-Terms

Abstraction

$$(\lambda x.\, t)$$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

λ-Terms

Abstraction

$(\lambda x.\, t)$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

λ-Terms

Abstraction

$$(\lambda x.\, t)$$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

$$g(x, y) = x + y$$

λ-Terms

Abstraction

$$(\lambda x.\, t)$$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

$$g(x, y) = x + y$$
$$f_x(y) = x + y$$
$$g'(x) = f_x$$

λ-Terms

Abstraction

$(\lambda x.\, t)$

- The function that maps x to t.
- t is called the *body* of the abstraction.
- The free occurences of x in t are bound in $(\lambda x.\, t)$.

Curryfication

$g(x, y) = x + y$

$f_x(y) = x + y$

$g'(x) = f_x$

$$g'(x)(y) = f_x(y) = x + y = g(x, y)$$

λ-Terms

Application

$$(t\,u)$$

λ-Terms

Application

$(t\,u)$

- The function t applied to the argument u.
- t is called the operator, and u the operand.

λ-Terms

Application

$(t\, u)$

- The function t applied to the argument u.
- t is called the operator, and u the operand.

Usual notations:

$$f : x \mapsto x + 1$$
$$f(3)$$

λ-Terms

Application

$(t\,u)$

- The function t applied to the argument u.
- t is called the operator, and u the operand.

Usual notations:

$$f : x \mapsto x + 1$$
$$f(3)$$

λ-calculus notations:

$$\lambda x.\,\mathbf{add}\,x\,1$$
$$(\lambda x.\,\mathbf{add}\,x\,1)\,3$$

Notational conventions

- When writing a λ-term, we omit the outermost parentheses
- We write $\lambda xyz.\, t$ for $(\lambda x.\, (\lambda y.\, (\lambda z.\, t)))$
- We write $t\, u\, v$ for $((t\, u)\, v)$

Notational conventions

- When writing a λ-term, we omit the outermost parentheses
- We write $\lambda xyz.\, t$ for $(\lambda x.\, (\lambda y.\, (\lambda z.\, t)))$
- We write $t\, u\, v$ for $((t\, u)\, v)$

With these conventions

$$\lambda xy.\, \mathbf{add}\, x\, y$$

stands for

$$(\lambda x.\, (\lambda y.\, ((\mathbf{add}\, x)\, y)))$$

Examples

$\lambda x.\, x$

- The identiy function.

Examples

$\lambda x.\, x$

- The identiy function.

$\lambda f.\, f\, \mathbf{j}$

- A (higher-order) function that takes a function has an argument and applies it to the constant $\mathbf{j}$.

Examples

$\lambda x.\, x$

- The identiy function.

$\lambda f.\, f\, \mathbf{j}$

- A (higher-order) function that takes a function has an argument and applies it to the constant $\mathbf{j}$.

$\lambda f g x.\, f\, (g\, x)$

- Functional composition (usually written as $\circ$).

Examples

$\lambda x.\, x$

- The identiy function.

$\lambda f.\, f\, \mathbf{j}$

- A (higher-order) function that takes a function has an argument and applies it to the constant $\mathbf{j}$.

$\lambda f g x.\, f\, (g\, x)$

- Functional composition (usually written as $\circ$).

$\lambda x.\, x\, x$

- A function that takes a function as an argument and applies it to itself (?).

β-Reduction

$$(\lambda f x.\, f\; x\; x)\, (\lambda y z.\, \mathbf{add}\; y\; z)\; 3$$

β-Reduction

$$(\lambda f x.\, f\, x\, x)\, (\lambda y z.\, \mathbf{add}\, y\, z)\, 3$$

$$\rightarrow \quad (\lambda x.\, (\lambda y z.\, \mathbf{add}\, y\, z)\, x\, x)\, 3$$

β-Reduction

$$(\lambda f x.\, f\; x\; x)\; (\lambda y z.\, \mathbf{add}\; y\; z)\; 3$$

$$\begin{aligned} &\rightarrow \quad (\lambda x.\, (\lambda y z.\, \mathbf{add}\; y\; z)\; x\; x)\; 3 \\ &\rightarrow \quad (\lambda x y.\, \mathbf{add}\; x\; y)\; 3\; 3 \end{aligned}$$

β-Reduction

$$(\lambda f x.\, f\, x\, x)\, (\lambda y z.\, \mathbf{add}\, y\, z)\, 3$$

$$\begin{aligned}
&\rightarrow \quad (\lambda x.\, (\lambda y z.\, \mathbf{add}\, y\, z)\, x\, x)\, 3 \\
&\rightarrow \quad (\lambda x y.\, \mathbf{add}\, x\, y)\, 3\, 3 \\
&\rightarrow \quad (\lambda y.\, \mathbf{add}\, 3\, y)\, 3
\end{aligned}$$

β-Reduction

$$(\lambda f x.\, f\, x\, x)\, (\lambda y z.\, \mathbf{add}\, y\, z)\, 3$$

$$\begin{aligned}
&\rightarrow \quad (\lambda x.\, (\lambda y z.\, \mathbf{add}\, y\, z)\, x\, x)\, 3 \\
&\rightarrow \quad (\lambda x y.\, \mathbf{add}\, x\, y)\, 3\, 3 \\
&\rightarrow \quad (\lambda y.\, \mathbf{add}\, 3\, y)\, 3 \\
&\rightarrow \quad \mathbf{add}\, 3\, 3
\end{aligned}$$

β-Reduction

Substitution

Let t and u be λ-terms, and x be a λ-variable. $t[x := u]$ denotes the λ-term obtained by substituting u for the free occurrences of x in t. It is inductively defined as follows:

$c[x := u] = c$, for $c \in \mathscr{C}$.

$y[x := u] = y$, for $y \in \mathscr{X}$, and $y \neq x$.

$x[x := u] = u$

$(\lambda y.\, t_0)[x := u] = (\lambda y.\, t_0[x := u])$, where $y \neq x$ and y not free in u.

$(t_0\, t_1)[x := u] = (t_0[x := u]\, t_1[x := u])$

β-Reduction

Notion of β-reduction

$$(\lambda x.\, t)\, u \to_\beta t[x := u]$$

Relation of β-contraction

$$C[(\lambda x.\, t)\, u] \to_\beta C[t[x := u]]$$

Relation of β-reduction

The *reflexive, transitive* closure of the relation of β-contraction.

$$t \twoheadrightarrow_\beta u$$

Relation of β-equivalence

The *reflexive, transitive, symmetric* closure of the relation of β-contraction.

$$t =_\beta u$$

β-Reduction

Church-Rosser property

Let t_0, t_1, and t_2 be λ-terms such that

$$t_0 \twoheadrightarrow_\beta t_1$$
$$t_0 \twoheadrightarrow_\beta t_2$$

Then, there exists a λ-term t_3 such that

$$t_1 \twoheadrightarrow_\beta t_3$$
$$t_2 \twoheadrightarrow_\beta t_3$$

β-Reduction

Church-Rosser property

Let t_0, t_1, and t_2 be λ-terms such that

$$t_0 \twoheadrightarrow_\beta t_1$$
$$t_0 \twoheadrightarrow_\beta t_2$$

Then, there exists a λ-term t_3 such that

$$t_1 \twoheadrightarrow_\beta t_3$$
$$t_2 \twoheadrightarrow_\beta t_3$$

Corollary: unicity of the normal forms.

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$\Omega \quad =$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$\Omega \quad = \quad \delta\, \delta$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$\begin{array}{rcl} \Omega & = & \delta\,\delta \\ & = & (\lambda x.\, x\, x)\, \delta \end{array}$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\,\delta$.

Then, we have:

$$\begin{array}{rcl} \Omega & = & \delta\,\delta \\ & = & (\lambda x.\, x\, x)\,\delta \\ & \rightarrow_\beta & \delta\,\delta \end{array}$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$\begin{array}{rcl} \Omega & = & \delta\, \delta \\ & = & (\lambda x.\, x\, x)\, \delta \\ & \rightarrow_\beta & \delta\, \delta \\ & = & \Omega \end{array}$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$
\begin{array}{rcl}
\Omega & = & \delta\, \delta \\
 & = & (\lambda x.\, x\, x)\, \delta \\
 & \to_\beta & \delta\, \delta \\
 & = & \Omega \\
 & = & \delta\, \delta
\end{array}
$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$\begin{array}{lcl} \Omega & = & \delta\, \delta \\ & = & (\lambda x.\, x\, x)\, \delta \\ & \to_\beta & \delta\, \delta \\ & = & \Omega \\ & = & \delta\, \delta \\ & = & (\lambda x.\, x\, x)\, \delta \end{array}$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$
\begin{array}{rcl}
\Omega & = & \delta\, \delta \\
 & = & (\lambda x.\, x\, x)\, \delta \\
 & \rightarrow_\beta & \delta\, \delta \\
 & = & \Omega \\
 & = & \delta\, \delta \\
 & = & (\lambda x.\, x\, x)\, \delta \\
 & \rightarrow_\beta & \delta\, \delta
\end{array}
$$

β-Reduction

Let $\delta = \lambda x.\, x\, x$, and $\Omega = \delta\, \delta$.

Then, we have:

$$
\begin{array}{rcl}
\Omega & = & \delta\, \delta \\
& = & (\lambda x.\, x\, x)\, \delta \\
& \to_\beta & \delta\, \delta \\
& = & \Omega \\
& = & \delta\, \delta \\
& = & (\lambda x.\, x\, x)\, \delta \\
& \to_\beta & \delta\, \delta \\
& = & \Omega \\
& \vdots &
\end{array}
$$

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types*
The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \to \beta)$ is a simple type.

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types*
The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \to \beta)$ is a simple type.

The intended meaning is that $(\alpha \to \beta)$ is the type of the λ-terms that stand for functions whose domain is α, and range β.

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types*
The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \rightarrow \beta)$ is a simple type.

The intended meaning is that $(\alpha \rightarrow \beta)$ is the type of the λ-terms that stand for functions whose domain is α, and range β.

Given a set of atomic type $\mathscr{A}$, we write $\mathscr{T}(\mathscr{A})$ for the set of simple types built upon $\mathscr{A}$.

Simple types

Definition

Let $\mathscr{A}$ be a set of symbols whose elements are called *atomic types*
The set of simple types is inductively defined as follows:

- every $a \in \mathscr{A}$ is a simple type;
- if α and β are simple types then $(\alpha \rightarrow \beta)$ is a simple type.

The intended meaning is that $(\alpha \rightarrow \beta)$ is the type of the λ-terms that stand for functions whose domain is α, and range β.

Given a set of atomic type $\mathscr{A}$, we write $\mathscr{T}(\mathscr{A})$ for the set of simple types built upon $\mathscr{A}$.

Most often, we let $\mathscr{A} = \{\mathbf{e}, \mathbf{t}\}$

Simple types

Signature

A higher-order signature is a triple $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$, where:

- $\mathscr{A}$ is a set of atomic types;
- $\mathscr{C}$ is a set of constants;
- $\tau \in \mathscr{T}(\mathscr{A})^{\mathscr{C}}$ is a function that assigns each constant in $\mathscr{C}$ with a simple type built on $\mathscr{A}$.

Simply typed λ-terms

Definition

Let $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$ be a signature, and let $(\mathscr{X}_\alpha)_{\alpha \in \mathscr{T}(\mathscr{A})}$ be a family of countably infinite disjoint sets, disjoint from $\mathscr{C}$, whose elements are called *typed λ-variables*. The set of typed λ-terms is inductively defined as follows:

- every $c \in \mathscr{C}$ is a λ-term of type $\tau(c)$;
- every $x \in \mathscr{X}_\alpha$ is a λ-term of type α;
- if x is a λ-variable of type α and t is a λ-term of type β then $(\lambda x_\alpha.\, t)$ is a λ-term of type $(\alpha \to \beta)$;
- if t is a λ-term of type $(\alpha \to \beta)$ and u is a λ-term of type α then $(t\, u)$ is a λ-term is a λ-term of type β.

Normalization

Normalization

- Every simply-typed λ-term has a normal form.

Normalization

Normalization

- Every simply-typed λ-term has a normal form.

Strong normalization

- There is no infinite β-reduction path starting from a simply-typed λ-term.

Interpretation

Model

A model consists of a family of sets $(D_\alpha)_{\alpha \in \mathscr{T}(\mathscr{A})}$ and an interpretation function $\mathcal{I}$ defined on $\mathscr{C}$ such that:

- $D_{\alpha\to\beta} = {D_\beta}^{D_\alpha}$;
- for every $c \in \mathscr{C}$, $\mathcal{I}(c) \in D_{\tau(c)}$.

Valuation

A typed valuation ξ is a function from $\bigcup_{\alpha \in \mathscr{T}(\mathscr{A})} \mathscr{X}_\alpha$ into $\bigcup_{\alpha \in \mathscr{T}(\mathscr{A})} \mathscr{D}_\alpha$ such that:

- if $x \in \mathscr{X}_\alpha$ then $\xi(x) \in \mathscr{D}_\alpha$.

Interpretation

Interpretation

- $[\![c]\!]_\xi = \mathcal{I}(c)$
- $[\![x]\!]_\xi = \xi(x)$
- $[\![\lambda x.\, t]\!]_\xi = a \mapsto [\![t]\!]_{\xi[x:=a]}$
- $[\![t\, u]\!]_\xi = [\![t]\!]_\xi([\![u]\!]_\xi)$

Logical constants

Signature with logical constants

Let $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$ be such that:

- $\mathscr{A} = \{\mathbf{e}, \mathbf{t}\}$;
- $\mathbf{not}, \mathbf{and}, \mathbf{or}, \mathbf{implies}, \mathbf{all}, \mathbf{exists} \in \mathscr{C}$;
- $\tau(\mathbf{not}) = \mathbf{t} \to \mathbf{t}$;
- $\tau(\mathbf{and}) = \mathbf{t} \to (\mathbf{t} \to \mathbf{t})$;
- $\tau(\mathbf{or}) = \mathbf{t} \to (\mathbf{t} \to \mathbf{t})$;
- $\tau(\mathbf{implies}) = \mathbf{t} \to (\mathbf{t} \to \mathbf{t})$;
- $\tau(\mathbf{all}) = (\mathbf{e} \to \mathbf{t}) \to \mathbf{t}$;
- $\tau(\mathbf{exists}) = (\mathbf{e} \to \mathbf{t}) \to \mathbf{t}$.

Logical constants

Interpretation

Let $\mathscr{M} = ((D_\alpha)_{\alpha \in \mathscr{T}(\mathscr{A})}, \mathcal{I})$ be such that:

- $D_{\mathbf{t}} = \{0, 1\}$;
- $\mathcal{I}(\mathbf{not}) = \{(0,1),(1,0)\}$;
- $\mathcal{I}(\mathbf{and}) = \{(0,\{(0,0),(1,0)\}),(1,\{(0,0),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{or}) = \{(0,\{(0,0),(1,1)\}),(1,\{(0,1),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{implies}) = \{(0,\{(0,1),(1,1)\}),(1,\{(0,0),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{all})(f) = 1$ iff $f(a) = 1$ for every $a \in D_{\mathbf{e}}$;
- $\mathcal{I}(\mathbf{exists})(f) = 1$ iff $f(a) = 1$ for some $a \in D_{\mathbf{e}}$.

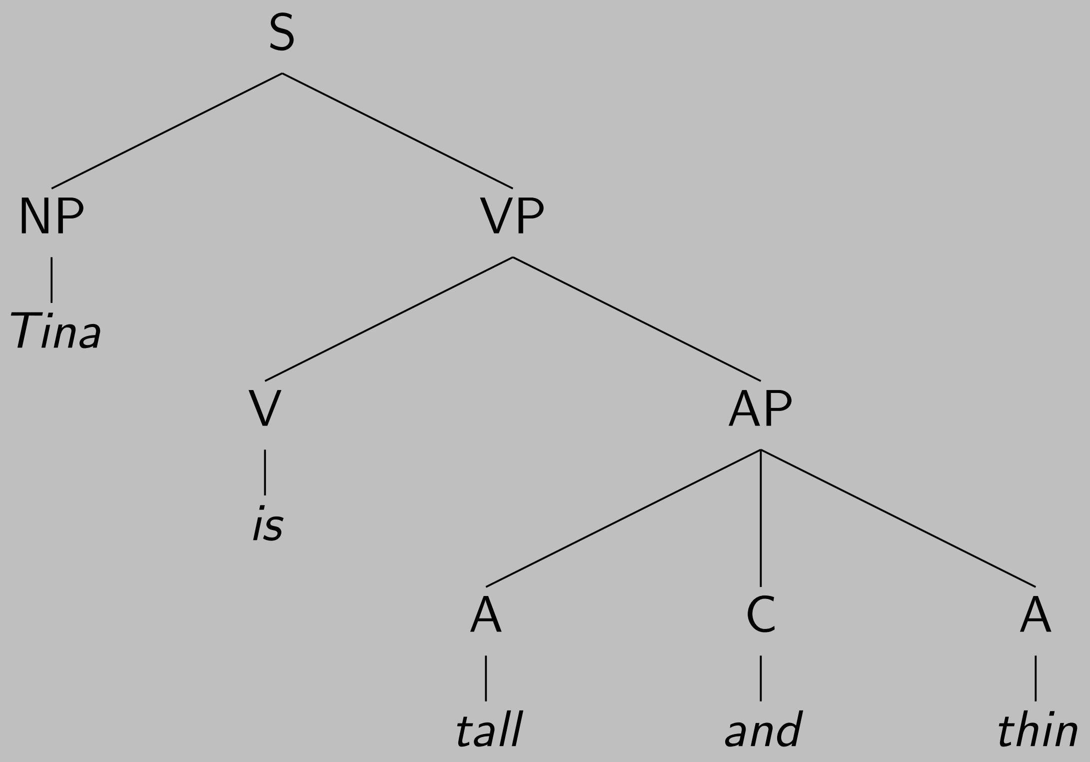
Logical constants

Notations

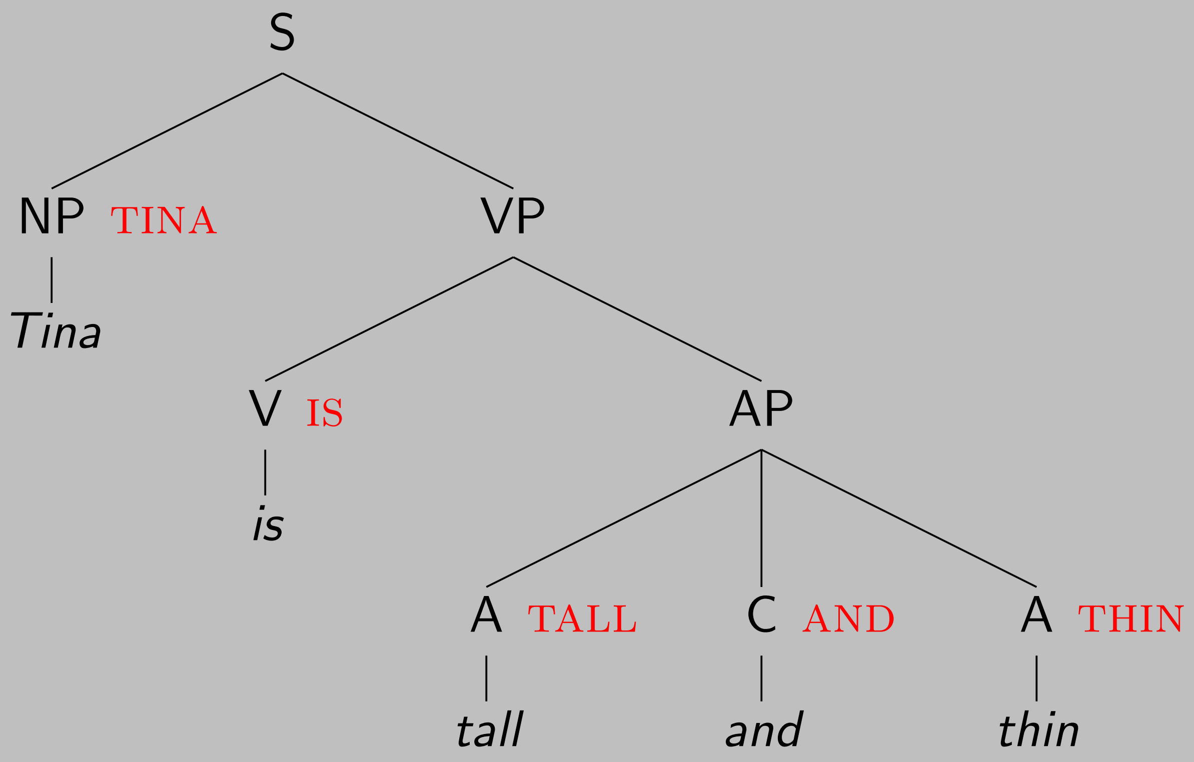
We write:

- $\neg a$ for $(\mathbf{not}\, a)$;
- $(a \wedge b)$ for $((\mathbf{and}\, a)\, b)$;
- $(a \vee b)$ for $((\mathbf{or}\, a)\, b)$;
- $(a \rightarrow b)$ for $((\mathbf{implies}\, a)\, b)$;
- $(\forall x.\, a)$ for $(\mathbf{all}\, (\lambda x.\, a))$;
- $(\exists x.\, a)$ for $(\mathbf{exits}\, (\lambda x.\, a))$.

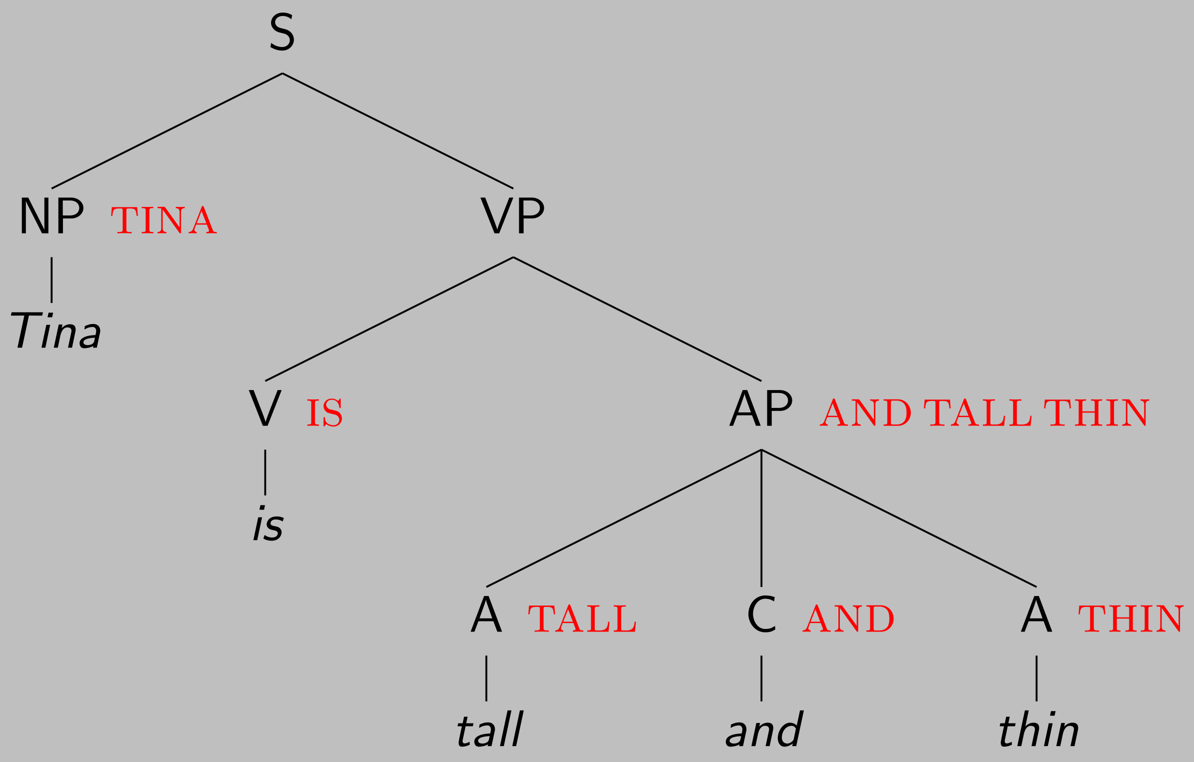
Example



Example

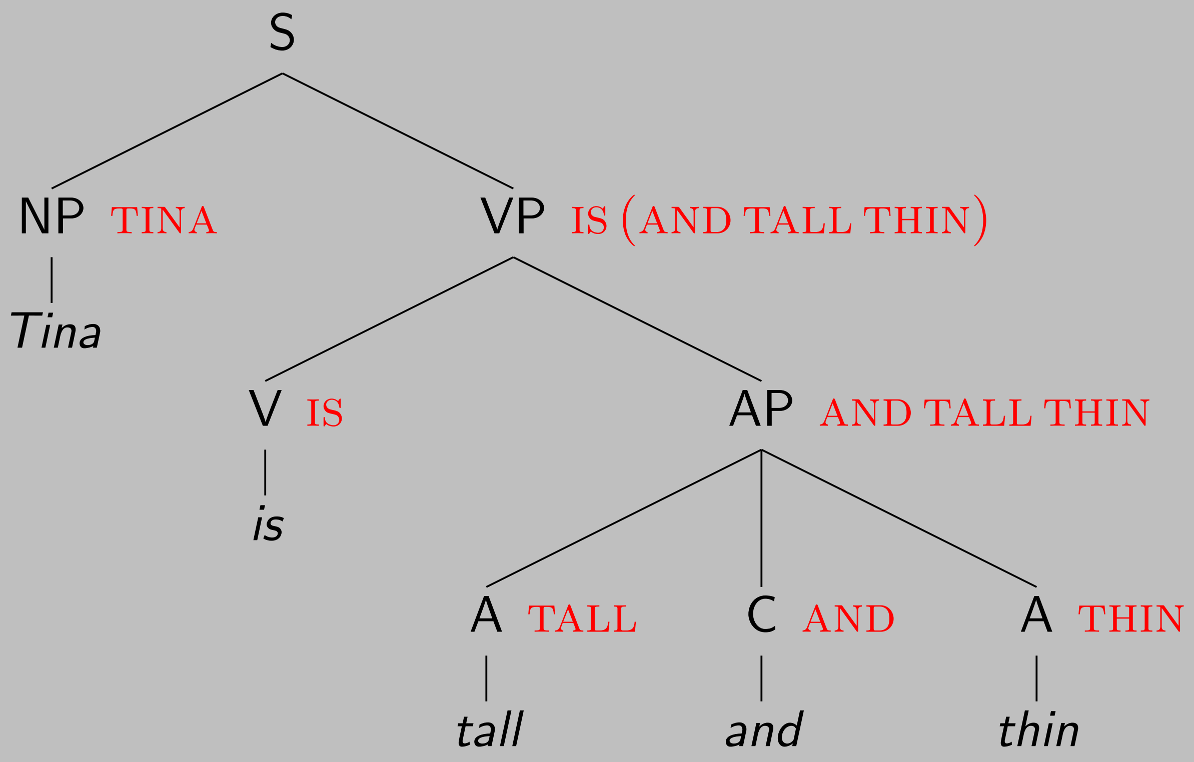


Example

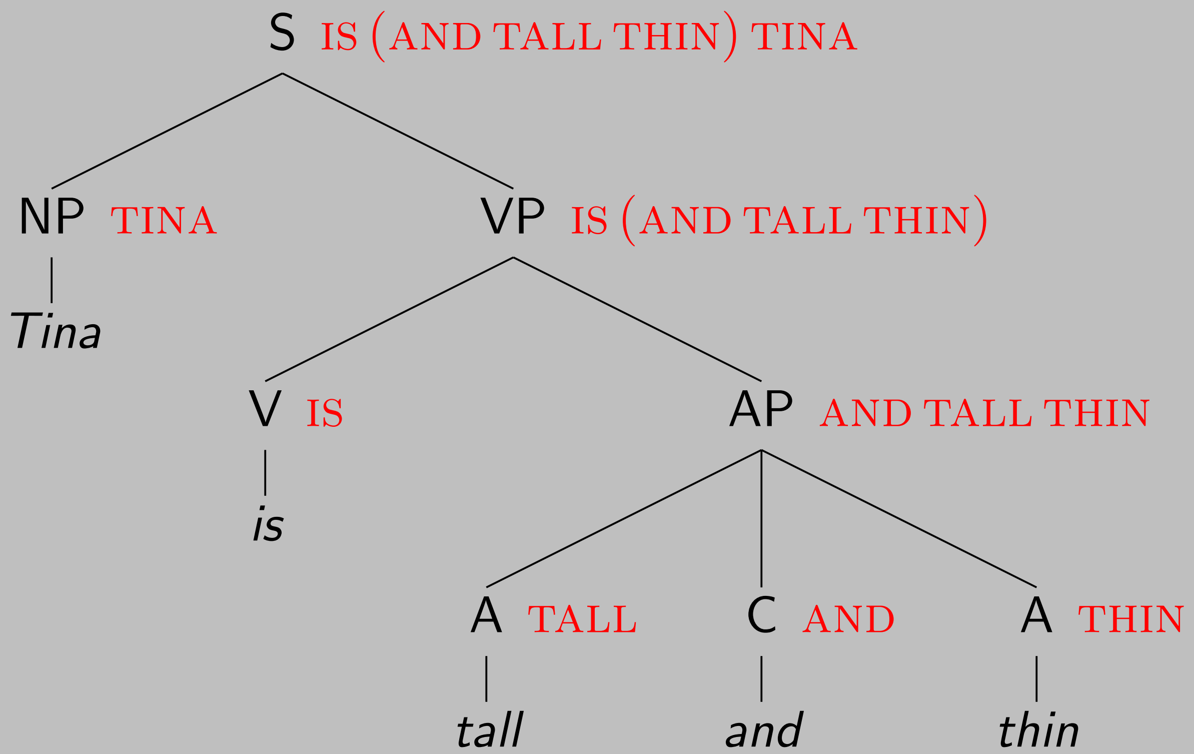


S
NP TINA
Tina
VP
V IS
is
AP AND TALL THIN
A TALL
tall
C AND
and
A THIN
thin

Example

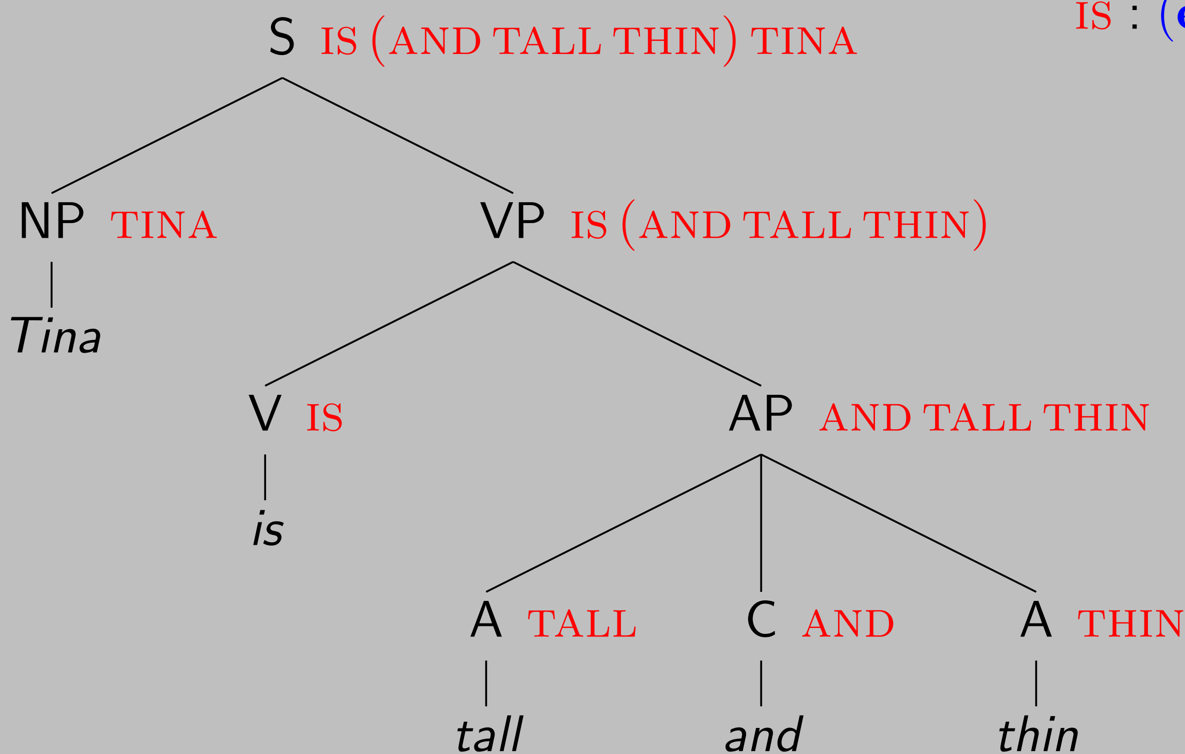

S
NP TINA
Tina
VP IS (AND TALL THIN)
V IS
is
AP AND TALL THIN
A TALL
tall
C AND
and
A THIN
thin

Example


S IS (AND TALL THIN) TINA
NP TINA
Tina
VP IS (AND TALL THIN)
V IS
is
AP AND TALL THIN
A TALL
tall
C AND
and
A THIN
thin

Example

$$
\begin{aligned}
\textsc{tina} &: \mathbf{e} \\
\textsc{tall} &: \mathbf{e} \to \mathbf{t} \\
\textsc{thin} &: \mathbf{e} \to \mathbf{t} \\
\textsc{and} &: (\mathbf{e} \to \mathbf{t}) \to (\mathbf{e} \to \mathbf{t}) \to \mathbf{e} \to \mathbf{t} \\
\textsc{is} &: (\mathbf{e} \to \mathbf{t}) \to \mathbf{e} \to \mathbf{t}
\end{aligned}
$$



Example

$$
\begin{aligned}
\text{TINA} &:= \mathbf{tina} && : \mathbf{e} \\
\text{TALL} &:= \lambda x.\, \mathbf{tall}\, x && : \mathbf{e} \to \mathbf{t} \\
\text{THIN} &:= \lambda x.\, \mathbf{thin}\, x && : \mathbf{e} \to \mathbf{t} \\
\text{AND} &:= \lambda pqx.\, (p\, x) \wedge (q\, x) && : (\mathbf{e} \to \mathbf{t}) \to (\mathbf{e} \to \mathbf{t}) \to \mathbf{e} \to \mathbf{t} \\
\text{IS} &:= \lambda px.\, p\, x && : (\mathbf{e} \to \mathbf{t}) \to \mathbf{e} \to \mathbf{t}
\end{aligned}
$$

Example

IS (AND TALL THIN) TINA

Example

$$\text{IS}\ (\text{AND TALL THIN})\ \text{TINA} \;=\; (\lambda p x.\, p\, x)\ (\text{AND TALL THIN})\ \text{TINA}$$

Example

$$
\begin{aligned}
\textsc{is}\,(\textsc{and tall thin})\,\textsc{tina} &= (\lambda p x.\, p\, x)\,(\textsc{and tall thin})\,\textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, \textsc{and tall thin}\; x)\,\textsc{tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\ (\textsc{and tall thin})\ \textsc{tina} &= (\lambda px.\, p\, x)\ (\textsc{and tall thin})\ \textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, \textsc{and tall thin}\ x)\ \textsc{tina} \\
&\rightarrow_\beta \textsc{and tall thin tina}
\end{aligned}
$$

Example

$$\begin{aligned}
\textsc{is}\ (\textsc{and tall thin})\ \textsc{tina} &= (\lambda px.\, p\, x)\ (\textsc{and tall thin})\ \textsc{tina} \\
&\to_\beta (\lambda x.\, \textsc{and tall thin}\ x)\ \textsc{tina} \\
&\to_\beta \textsc{and tall thin tina} \\
&= (\lambda pqx.\, (p\, x) \wedge (q\, x))\ \textsc{tall thin tina}
\end{aligned}$$

Example

$$
\begin{aligned}
\textsc{is}\ (\textsc{and tall thin})\ \textsc{tina} &= (\lambda px.\, p\, x)\, (\textsc{and tall thin})\ \textsc{tina} \\
&\to_\beta (\lambda x.\, \textsc{and tall thin}\ x)\ \textsc{tina} \\
&\to_\beta \textsc{and tall thin tina} \\
&= (\lambda pqx.\, (p\, x) \wedge (q\, x))\ \textsc{tall thin tina} \\
&\to_\beta (\lambda qx.\, (\textsc{tall}\ x) \wedge (q\, x))\ \textsc{thin tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\,(\textsc{and tall thin})\,\textsc{tina} &= (\lambda px.\, p\, x)\,(\textsc{and tall thin})\,\textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, \textsc{and tall thin}\, x)\,\textsc{tina} \\
&\rightarrow_\beta \textsc{and tall thin tina} \\
&= (\lambda pqx.\, (p\, x) \wedge (q\, x))\,\textsc{tall thin tina} \\
&\rightarrow_\beta (\lambda qx.\, (\textsc{tall}\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&= (\lambda qx.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\,\textsc{thin tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\,(\textsc{and tall thin})\,\textsc{tina} &= (\lambda p x.\, p\, x)\,(\textsc{and tall thin})\,\textsc{tina} \\
&\to_\beta (\lambda x.\, \textsc{and tall thin}\, x)\,\textsc{tina} \\
&\to_\beta \textsc{and tall thin tina} \\
&= (\lambda p q x.\, (p\, x) \wedge (q\, x))\,\textsc{tall thin tina} \\
&\to_\beta (\lambda q x.\, (\textsc{tall}\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&= (\lambda q x.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&\to_\beta (\lambda q x.\, (\mathbf{tall}\, x) \wedge (q\, x))\,\textsc{thin tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\ (\textsc{and tall thin})\ \textsc{tina} &= (\lambda p x.\, p\, x)\ (\textsc{and tall thin})\ \textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, \textsc{and tall thin}\ x)\ \textsc{tina} \\
&\rightarrow_\beta \textsc{and tall thin tina} \\
&= (\lambda p q x.\, (p\, x) \wedge (q\, x))\ \textsc{tall thin tina} \\
&\rightarrow_\beta (\lambda q x.\, (\textsc{tall}\ x) \wedge (q\, x))\ \textsc{thin tina} \\
&= (\lambda q x.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\ \textsc{thin tina} \\
&\rightarrow_\beta (\lambda q x.\, (\mathbf{tall}\, x) \wedge (q\, x))\ \textsc{thin tina} \\
&\rightarrow_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\textsc{thin}\ x))\ \textsc{tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\ (\textsc{and tall thin})\ \textsc{tina} &= (\lambda p x.\, p\, x)\ (\textsc{and tall thin})\ \textsc{tina} \\
&\to_\beta (\lambda x.\, \textsc{and tall thin}\ x)\ \textsc{tina} \\
&\to_\beta \textsc{and tall thin tina} \\
&= (\lambda p q x.\, (p\, x) \wedge (q\, x))\ \textsc{tall thin tina} \\
&\to_\beta (\lambda q x.\, (\textsc{tall}\ x) \wedge (q\, x))\ \textsc{thin tina} \\
&= (\lambda q x.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\ \textsc{thin tina} \\
&\to_\beta (\lambda q x.\, (\mathbf{tall}\, x) \wedge (q\, x))\ \textsc{thin tina} \\
&\to_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\textsc{thin}\ x))\ \textsc{tina} \\
&= (\lambda x.\, (\mathbf{tall}\, x) \wedge ((\lambda x.\, \mathbf{thin}\, x)\, x))\ \textsc{tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\,(\textsc{and tall thin})\,\textsc{tina} &= (\lambda p x.\, p\, x)\,(\textsc{and tall thin})\,\textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, \textsc{and tall thin}\, x)\,\textsc{tina} \\
&\rightarrow_\beta \textsc{and tall thin tina} \\
&= (\lambda p q x.\, (p\, x) \wedge (q\, x))\,\textsc{tall thin tina} \\
&\rightarrow_\beta (\lambda q x.\, (\textsc{tall}\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&= (\lambda q x.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&\rightarrow_\beta (\lambda q x.\, (\mathbf{tall}\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&\rightarrow_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\textsc{thin}\, x))\,\textsc{tina} \\
&= (\lambda x.\, (\mathbf{tall}\, x) \wedge ((\lambda x.\, \mathbf{thin}\, x)\, x))\,\textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\mathbf{thin}\, x))\,\textsc{tina}
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\,(\textsc{and tall thin})\,\textsc{tina} &= (\lambda p x.\, p\, x)\,(\textsc{and tall thin})\,\textsc{tina} \\
&\to_\beta (\lambda x.\, \textsc{and tall thin}\, x)\,\textsc{tina} \\
&\to_\beta \textsc{and tall thin tina} \\
&= (\lambda p q x.\, (p\, x) \wedge (q\, x))\,\textsc{tall thin tina} \\
&\to_\beta (\lambda q x.\, (\textsc{tall}\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&= (\lambda q x.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&\to_\beta (\lambda q x.\, (\mathbf{tall}\, x) \wedge (q\, x))\,\textsc{thin tina} \\
&\to_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\textsc{thin}\, x))\,\textsc{tina} \\
&= (\lambda x.\, (\mathbf{tall}\, x) \wedge ((\lambda x.\, \mathbf{thin}\, x)\, x))\,\textsc{tina} \\
&\to_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\mathbf{thin}\, x))\,\textsc{tina} \\
&\to_\beta (\mathbf{tall}\,\textsc{tina}) \wedge (\mathbf{thin}\,\textsc{tina})
\end{aligned}
$$

Example

$$
\begin{aligned}
\textsc{is}\ (\textsc{and tall thin})\ \textsc{tina} &= (\lambda px.\, p\, x)\, (\textsc{and tall thin})\ \textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, \textsc{and tall thin}\ x)\ \textsc{tina} \\
&\rightarrow_\beta \textsc{and tall thin tina} \\
&= (\lambda pqx.\, (p\, x) \wedge (q\, x))\ \textsc{tall thin tina} \\
&\rightarrow_\beta (\lambda qx.\, (\textsc{tall}\ x) \wedge (q\, x))\ \textsc{thin tina} \\
&= (\lambda qx.\, ((\lambda x.\, \mathbf{tall}\, x)\, x) \wedge (q\, x))\ \textsc{thin tina} \\
&\rightarrow_\beta (\lambda qx.\, (\mathbf{tall}\, x) \wedge (q\, x))\ \textsc{thin tina} \\
&\rightarrow_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\textsc{thin}\ x))\ \textsc{tina} \\
&= (\lambda x.\, (\mathbf{tall}\, x) \wedge ((\lambda x.\, \mathbf{thin}\, x)\, x))\ \textsc{tina} \\
&\rightarrow_\beta (\lambda x.\, (\mathbf{tall}\, x) \wedge (\mathbf{thin}\, x))\ \textsc{tina} \\
&\rightarrow_\beta (\mathbf{tall}\ \textsc{tina}) \wedge (\mathbf{thin}\ \textsc{tina}) \\
&= (\mathbf{tall}\ \mathbf{tina}) \wedge (\mathbf{thin}\ \mathbf{tina})
\end{aligned}
$$