Computational Semantics

Philippe de Groote

2018-2019

Outline I

Outline II

Outline III

Outline

Intoduction

- Semantics is the study of *meaning*.
- Entailment

  Eric is the husband of Rebecca and the father of John.
  $\rightarrow$ *Eric is the father of John.*
  $\rightarrow$ *Eric is the husband of Rebecca.*
- Formal semantics:
  - The meaning of an utterance depends upon its *form*, i.e., its linguistic structure.
  - The tools used to account for the meanings of utterances are *formal* mathematical tool.
- Truth conditional semantics.
- Model theoretic semantics.

Outline

Naive set theory

Membership

$x \in A$

- x is an element (or a member) of the set A.
- x belongs to A.

$x \notin A$

- x does not belong to A.

Inclusion

$A \subset B$

- the set A is a subset of the set B.
- every element x that belongs to A belongs to B.

Naive set theory

Definition by extension

$\{a, b, c\}$

- the set whose elements are a, b, and c.

Definition by comprehension

$\{x \in A : P\}$

- the set whose elements are the elements of A that obey the property P.
- example: $\{n \in \mathbb{N} : n > 0 \text{ and } n \leq 4\} = \{1, 2, 3, 4\}$

Naive set theory

Powerset

$\mathscr{P}(A)$

- the set whose elements are the subsets of A.
- example:

$$\mathscr{P}(\{a, b, c\}) = \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

Naive set theory

Intersection

$A \cap B$

- the set whose elements belongs to both A and B.
- example: $\{a, b, c\} \cap \{b, c, d\} = \{b, c\}$

Union

$A \cup B$

- the set whose elements belongs to A or B (or both).
- example: $\{a, b, c\} \cap \{b, c, d\} = \{a, b, c, d\}$

Difference

$A \setminus B$

- the set whose elements belongs to both A but does not belong to B.
- example: $\{a, b, c\} \setminus \{b, c, d\} = \{a\}$

Naive set theory

Ordered pair

(a, b)

- the ordered pair whose first component (coordinate) is a and second component is b.

Cartesian product

$A \times B$

- the set of all the ordered pairs (a, b) such that $a \in A$ and $b \in B$.

$A \times B \times C$

- the set of all the 3-tuples (a, b, c) such that $a \in A$, and $b \in B$, and $c \in C$.

$A \times B \times C \times D \times \ldots$

- ...

Naive set theory

Empty set

$\varnothing$

- the set that does not have any member.
- for every element x, $x \notin \varnothing$.

Natural numbers

$0 = \varnothing$
$1 = \{0\} = \{\varnothing\}$
$2 = \{0, 1\} = \{\varnothing, \{\varnothing\}\}$
$n = \{0, 1, \ldots, n-1\}$

Disjoint Union

$A + B$

- $A + B = (\{0\} \times A) \cup (\{1\} \times B)$

Relations

Binary relation

$R \in \mathscr{P}(A \times B)$

- A binary relation R between A and B is a subset of the cartesian product $A \times B$.
- In case $(a, b) \in R$, one says that a is related to b by R, and one writes $a\,R\,b$ or $R(a, b)$.

n-ary relation

$R \in \mathscr{P}(A_1 \times A_2 \ldots \times A_n)$

- In case $(a_1, a_2, \ldots, a_n) \in R$, one writes $R(a_1, a_2, \ldots, a_n)$.

Functions

Function

A function f from A to B is a binary between A and B such that:

- for every $a \in A$, there is a $b \in B$ such that $(a, b) \in f$ (totality);
- for every $a \in A$ and every $b, c \in B$, if $(a, b) \in f$ and $(a, c) \in f$ then $b = c$ (functionality).

- In case $(a, b) \in f$, one says that f maps a to b, and one writes $f(a) = b$.

Set of functions

B^A

- is the set of functions from A to B.

Functions

Characteristic function

- Let $B \subset A$. The characteristic function $\kappa_B \in 2^A$ is defined as follows:

$$\kappa_B(a) = 1 \text{ iff } a \in B$$

- Let $f \in 2^A$. The fiber $f^{-1}(1)$ is the set defined as follows::

$$f^{-1}(1) = \{a \in A : f(a) = 1\}$$

- Let $B \subset A$. $\kappa_B^{-1}(1) = B$ and κ_f.
- Let $f \in 2^A$. $\kappa_{f^{-1}(1)} = f$.
- $\mathscr{P}(A)$ and 2^A are isomorphic.

Identities

The set-theoretic notations (product, sum, exponential, natural numbers) are such that high-school algebraic identities hold between sets (up to isomorphism).

Examples:

$$A \times A \simeq A^2$$
$$A^0 \simeq 1$$
$$A^1 \simeq A$$
$$A^B \times A^C \simeq A^{B+C}$$

...

Outline

Model

Informal definition

A model is an abstract mathematical structure made of a set of entities (the interpretation domain), together with operations (i.e. functions) and relations on that set.

Example

- $D = \{e, j, r\}$.
- $F, H \in D^2$.
- $F = \{(e, j)\}$.
- $H = \{(e, r)\}$.

Object language

Definition

A first-order langage consists in two sets of symbols:

- A set $\mathscr{F}$, together with an arity function $\mathrm{ar}_{\mathscr{F}} \in \mathbb{N}^{\mathscr{F}}$, whose elements are called *function symbols*.
- A set $\mathscr{R}$, together with an arity function $\mathrm{ar}_{\mathscr{R}} \in \mathbb{N}^{\mathscr{R}}$, whose elements are called *relation symbols*.

Example

- $\mathscr{F} = \{\mathbf{e}, \mathbf{j}, \mathbf{r}, \mathbf{father}\}$;
- $\mathrm{ar}_{\mathscr{F}}(\mathbf{e}) = 0, \mathrm{ar}_{\mathscr{F}}(\mathbf{j}) = 0, \mathrm{ar}_{\mathscr{F}}(\mathbf{r}) = 0, \mathrm{ar}_{\mathscr{F}}(\mathbf{father}) = 1$;
- $\mathscr{R} = \{\mathbf{Is}, \mathbf{Husband}\}$;
- $\mathrm{ar}_{\mathscr{R}}(\mathbf{e}) = 2, \mathrm{ar}_{\mathscr{R}}(\mathbf{j}) = 2$.

Object language

Terms

Let $\mathscr{X}$ be a countably infinite set of symbols whose elements are called *variables*. The set of terms is inductively defined as follows:

- every $x \in \mathscr{X}$ is a term;
- every $a \in \mathscr{F}$ such that $\mathrm{ar}_{\mathscr{F}}(a) = 0$ is a term,
- if $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$, and if $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Proposition

- if $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$, and if $t_1, \ldots, t_n$ are terms, then $R(t_1, \ldots, t_n)$ is an atomic proposition.

Example

- Terms: $\mathbf{e}; \mathbf{father}(\mathbf{j}); \mathbf{father}(\mathbf{father}(\mathbf{r})); \mathbf{father}(\mathbf{x})$.
- Proposition: $\mathbf{Is}(\mathbf{e}, \mathbf{father}(\mathbf{j})); \mathbf{Husband}(\mathbf{e}, \mathbf{r})$.

Interpretation

Model

Given a first-order langage, a model consists of a set D and an interpretation function $\mathcal{I}$ defined on $\mathscr{F} \cup \mathscr{R}$ such that:

- for every $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$, $\mathcal{I}(f) \in D^{D^n}$;
- for every $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$, $\mathcal{I}(R) \in 2^{D^n}$.

Example

- $D = \mathbb{N}$
- $\mathcal{I}(\mathbf{e}) = 6$
- $\mathcal{I}(\mathbf{j}) = 3$
- $\mathcal{I}(\mathbf{r}) = 7$
- $\mathcal{I}(\mathbf{father}) = f \in \mathbb{N}^{\mathbb{N}}$ such that $f(n) = 2n$
- $\mathcal{I}(\mathbf{Is}) = \{(a, b) \in \mathbb{N}^2 : a = b\}$
- $\mathcal{I}(\mathbf{Husband}) = \{(a, b) \in \mathbb{N}^2 : a = 2n \text{ and } b = a+1 \text{ for some } n \in \mathbb{N}\}$

Interpretation

Interpretation of the ground terms

Given a first-order langage, and a model, the interpretation of the ground terms is inductively defined as follows:

- $[\![a]\!] = \mathcal{I}(a)$, for $a \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(a) = 0$;
- $[\![f(t_1, \ldots, t_n)]\!] = \mathcal{I}(f)([\![t_1]\!], \ldots, [\![t_n]\!])$, for $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$.

Example

$$
\begin{aligned}
[\![\mathbf{father}(\mathbf{father}(\mathbf{r}))]\!] &= \mathcal{I}(\mathbf{father})([\![\mathbf{father}(\mathbf{r})]\!]) \\
&= 2 \cdot ([\![\mathbf{father}(\mathbf{r})]\!]) \\
&= 2 \cdot (\mathcal{I}(\mathbf{father})([\![\mathbf{r}]\!])) \\
&= 2 \cdot (2 \cdot [\![\mathbf{r}]\!]) \\
&= 2 \cdot (2 \cdot 7) \\
&= 28
\end{aligned}
$$

Interpretation

Interpretation of the closed atomic propositions

Given a first-order langage, and a model, the interpretation of the closed atomic propositions is defined as follows:

- $[\![R(t_1, \ldots, t_n)]\!] = \mathcal{I}(R)([\![t_1]\!], \ldots, [\![t_n]\!])$, for $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$.

Example

$$\begin{aligned}
[\![\mathbf{Is}(\mathbf{e}, \mathbf{father}(\mathbf{j}))]\!] &= \mathcal{I}(\mathbf{Is})([\![\mathbf{e}]\!], [\![\mathbf{father}(\mathbf{j})]\!]) \\
&= \mathcal{I}(\mathbf{Is})([\![\mathbf{e}]\!], \mathcal{I}(\mathbf{father})([\![\mathbf{j}]\!])) \\
&= \mathcal{I}(\mathbf{Is})(6, 2 \cdot 3) \\
&= \mathcal{I}(\mathbf{Is})(6, 6) \\
&= 1
\end{aligned}$$

Interpretation

Valuation

Given a first-order langage, and a model, a valuation is a a function $\xi \in D^{\mathscr{X}}$.

Interpretation of the terms

Given a first-order langage, and a model, the interpretation of the terms is inductively defined as follows:

- $[\![x]\!]_\xi = \xi(x)$, for $x \in \mathscr{X}$;
- $[\![a]\!]_\xi = \mathcal{I}(a)$, for $a \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(a) = 0$;
- $[\![f(t_1, \ldots, t_n)]\!]_\xi = \mathcal{I}(f)([\![t_1]\!]_\xi, \ldots, [\![t_n]\!]_\xi)$, for $f \in \mathscr{F}$ with $\mathrm{ar}_{\mathscr{F}}(f) = n$ and $n > 0$.

Interpretation

Interpretation of the atomic propositions

Given a first-order langage, and a model, the interpretation of the closed atomic propositions is defined as follows:

- $[\![R(t_1, \ldots, t_n)]\!]_\xi = \mathcal{I}(R)([\![t_1]\!]_\xi, \ldots, [\![t_n]\!]_\xi)$, for $R \in \mathscr{R}$ with $\mathrm{ar}_{\mathscr{R}}(R) = n$.

Outline

4. First-order logic
   - Propositional logic
   - Quantification
   - Interpretation

Propositional logic

propositions

Given a first-order language, the set of proposition is inductively defined as follows:

- every atomic proposition is a proposition;
- if α is a proposition then $\neg\alpha$ is a proposition;
- if α and β are propositions then $(\alpha \wedge \beta)$ is a proposition;
- if α and β are propositions then $(\alpha \vee \beta)$ is a proposition;
- if α and β are propositions then $(\alpha \rightarrow \beta)$ is a proposition.

Example

$$\mathbf{Husband}(\mathbf{e}, \mathbf{r}) \wedge \mathbf{Is}(\mathbf{e}, \mathbf{father}(\mathbf{j}))$$

Propositional logic

Negation

$\neg\alpha$

- *not* α.
- $[\![\neg\alpha]\!]_\xi = 1$ iff $[\![\alpha]\!]_\xi = 0$.

α	$\neg\alpha$
0	1
1	0

Propositional logic

Conjunction

$\alpha \wedge \beta$

- α *and* β.
- $[\![\alpha \wedge \beta]\!]_\xi = 1$ iff $[\![\alpha]\!]_\xi = 1$ and $[\![\beta]\!]_\xi = 1$.

α	β	$\alpha \wedge \beta$
0	0	0
0	1	0
1	0	0
1	1	1

Propositional logic

Disjunction

$\alpha \vee \beta$

- α *or* β.
- $[\![\alpha \vee \beta]\!]_\xi = 1$ iff $[\![\alpha]\!]_\xi = 1$ or $[\![\beta]\!]_\xi = 1$ (or both).

α	β	$\alpha \vee \beta$
0	0	0
0	1	1
1	0	1
1	1	1

Propositional logic

Implication

$\alpha \rightarrow \beta$

- *If* α *then* β; α *implies* β.
- $[\![\alpha \rightarrow \beta]\!]_\xi = 1$ iff $[\![\beta]\!]_\xi = 1$ whenever $[\![\alpha]\!]_\xi = 1$.

α	β	$\alpha \rightarrow \beta$
0	0	1
0	1	1
1	0	0
1	1	1

Quantification

first-order formulas

Given a first-order language, the set of first-order formulas is inductively defined as follows:

- every atomic proposition is a first-order formula;
- if α is a first-order formula then $\neg\alpha$ is a first-order formula;
- if α and β are first-order formulas then $(\alpha \wedge \beta)$ is a first-order formula;
- if α and β are first-order formulas then $(\alpha \vee \beta)$ is a first-order formula;
- if α and β are first-order formulas then $(\alpha \rightarrow \beta)$ is a first-order formula;
- if α is a first-order formulas and x a variable then $(\forall x.\, \alpha)$ is a first-order formula;
- if α is a first-order formulas and x a variable then $(\exists x.\, \alpha)$ is a first-order formula.

Example

$$\forall x.\, \exists y.\, \mathbf{Is}(\mathbf{y}, \mathbf{father}(\mathbf{x}))$$

Quantification

Universal quantification

$\forall x.\, \alpha$

- *every entity* x is such that α.
- $[\![\forall x.\, \alpha]\!]_\xi = 1$ iff $[\![\alpha]\!]_{\xi[x:=d]} = 1$ for every $d \in D$.

Quantification

Existential quantification

$\exists x.\, \alpha$

- *There is some entity* x such that α.
- $[\![\exists x.\, \alpha]\!]_\xi = 1$ iff $[\![\alpha]\!]_{\xi[x:=d]} = 1$ for some $d \in D$.

Interpretation

Let a first-order language be given, and let ϕ, $\mathscr{M}$, and ξ be respectively a first-order formula, a model, and a valuation.

$\mathscr{M}, \xi \models \phi$

- $\mathscr{M}$ and ξ *satisfy* ϕ.
- ϕ *is valid in* $\mathscr{M}$ *according to* ξ.
- $[\![\phi]\!]_\xi = 1$.

$\mathscr{M} \models \phi$

- $\mathscr{M}$ *satisfies* ϕ.
- ϕ *is valid in* $\mathscr{M}$.
- $\mathscr{M}, \xi \models \phi$ for every possible valuation ξ.

$\models \phi$

- ϕ *is valid*.
- $\mathscr{M} \models \phi$ for every possible model $\mathscr{M}$.

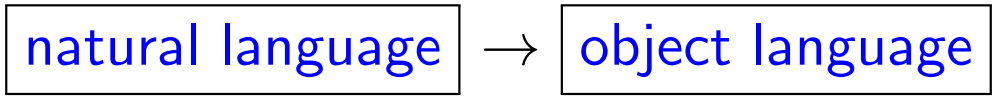
Outline

Using an object language

Using an object language

natural language

Using an object language



Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall}(\mathbf{e}) \wedge \mathbf{thin}(\mathbf{e})$

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall(e)} \wedge \mathbf{thin(e)}$

Rebecca has a husband

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall}(\mathbf{e}) \wedge \mathbf{thin}(\mathbf{e})$

Rebecca has a husband $\rightarrow$ $\exists x.\, \mathbf{husband}(x, \mathbf{r})$

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall}(\mathbf{e}) \wedge \mathbf{thin}(\mathbf{e})$

Rebecca has a husband $\rightarrow$ $\exists x.\, \mathbf{husband}(x, \mathbf{r})$

Eric has a wife

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall}(\mathbf{e}) \wedge \mathbf{thin}(\mathbf{e})$

Rebecca has a husband $\rightarrow$ $\exists x.\, \mathbf{husband}(x, \mathbf{r})$

Eric has a wife $\rightarrow$ $\exists x.\, \mathbf{husband}(\mathbf{e}, x)$

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall}(\mathbf{e}) \wedge \mathbf{thin}(\mathbf{e})$

Rebecca has a husband $\rightarrow$ $\exists x.\, \mathbf{husband}(x, \mathbf{r})$

Eric has a wife $\rightarrow$ $\exists x.\, \mathbf{husband}(\mathbf{e}, x)$

Everybody has a father

Using an object language

natural language $\rightarrow$ object language

Eric is tall and thin $\rightarrow$ $\mathbf{tall}(\mathbf{e}) \wedge \mathbf{thin}(\mathbf{e})$

Rebecca has a husband $\rightarrow$ $\exists x.\, \mathbf{husband}(x, \mathbf{r})$

Eric has a wife $\rightarrow$ $\exists x.\, \mathbf{husband}(\mathbf{e}, x)$

Everybody has a father $\rightarrow$ $\forall x.\, \exists y.\, \mathbf{Is}(\mathbf{y}, \mathbf{father}(\mathbf{x}))$

Using an object language

Using an object language

natural language

Using an object language

natural language $\rightarrow$ syntactic structure

Using an object language

Using an object language

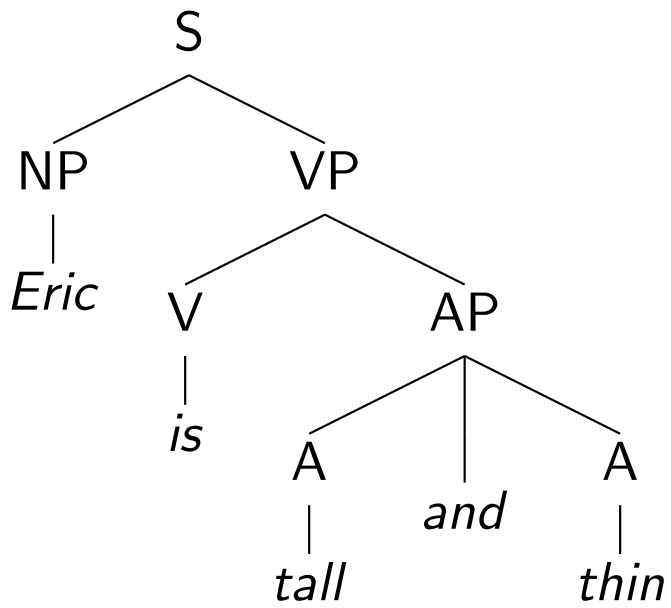
Using an object language

Eric is tall and thin

Using an object language

natural language → syntactic structure → object language → model

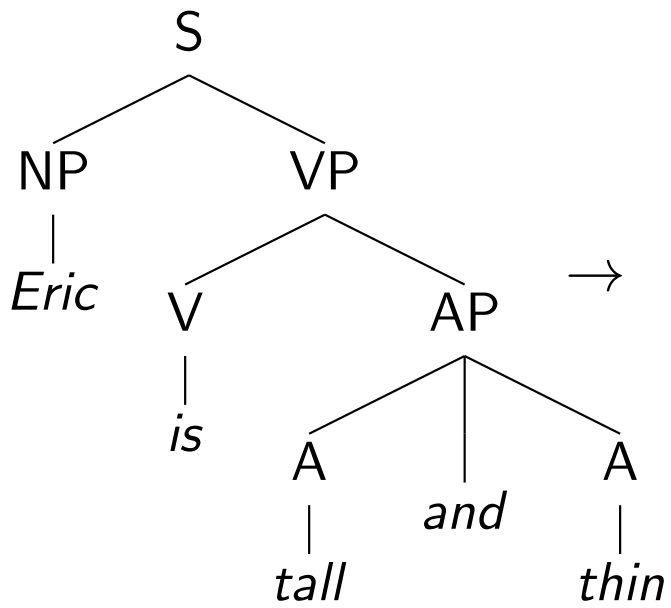
Eric is tall and thin →



Using an object language

natural language → syntactic structure → object language → model

Eric is tall and thin →



→ $\mathbf{tall(e) \wedge thin(e)}$

Compositionality

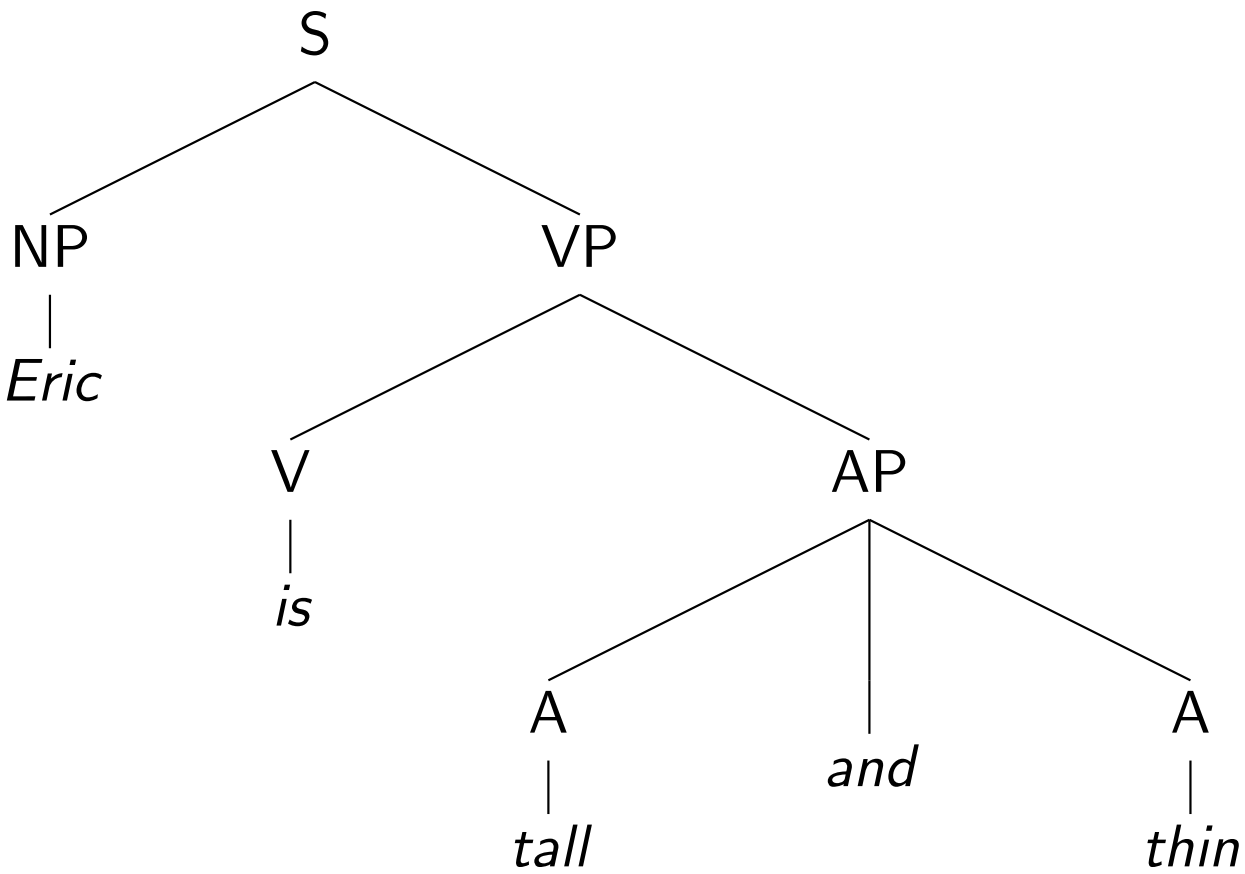
The principle of compositionality

Also known as Frege's principle:

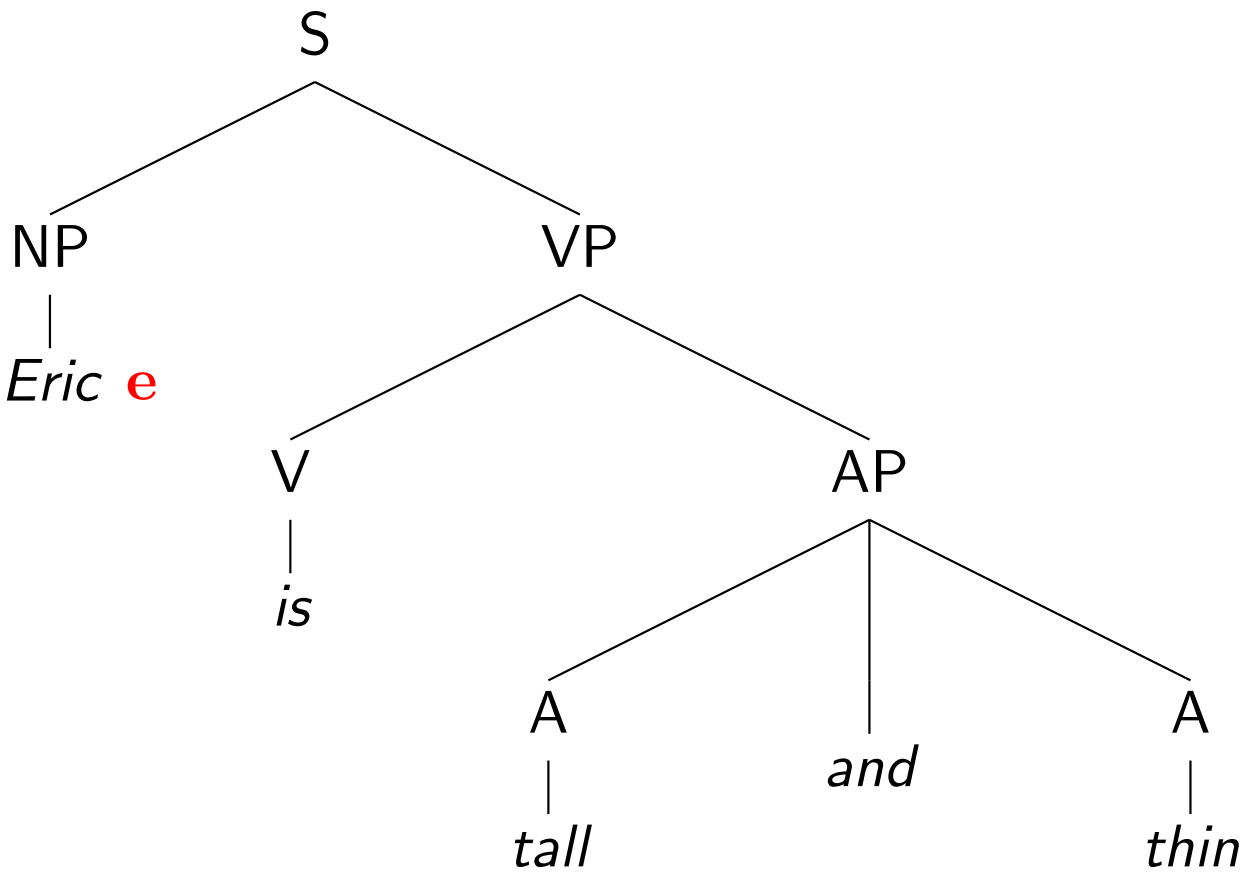
> *The meaning of a complex expression is determined by its structure and the meanings of its constituents.*

Compositionality

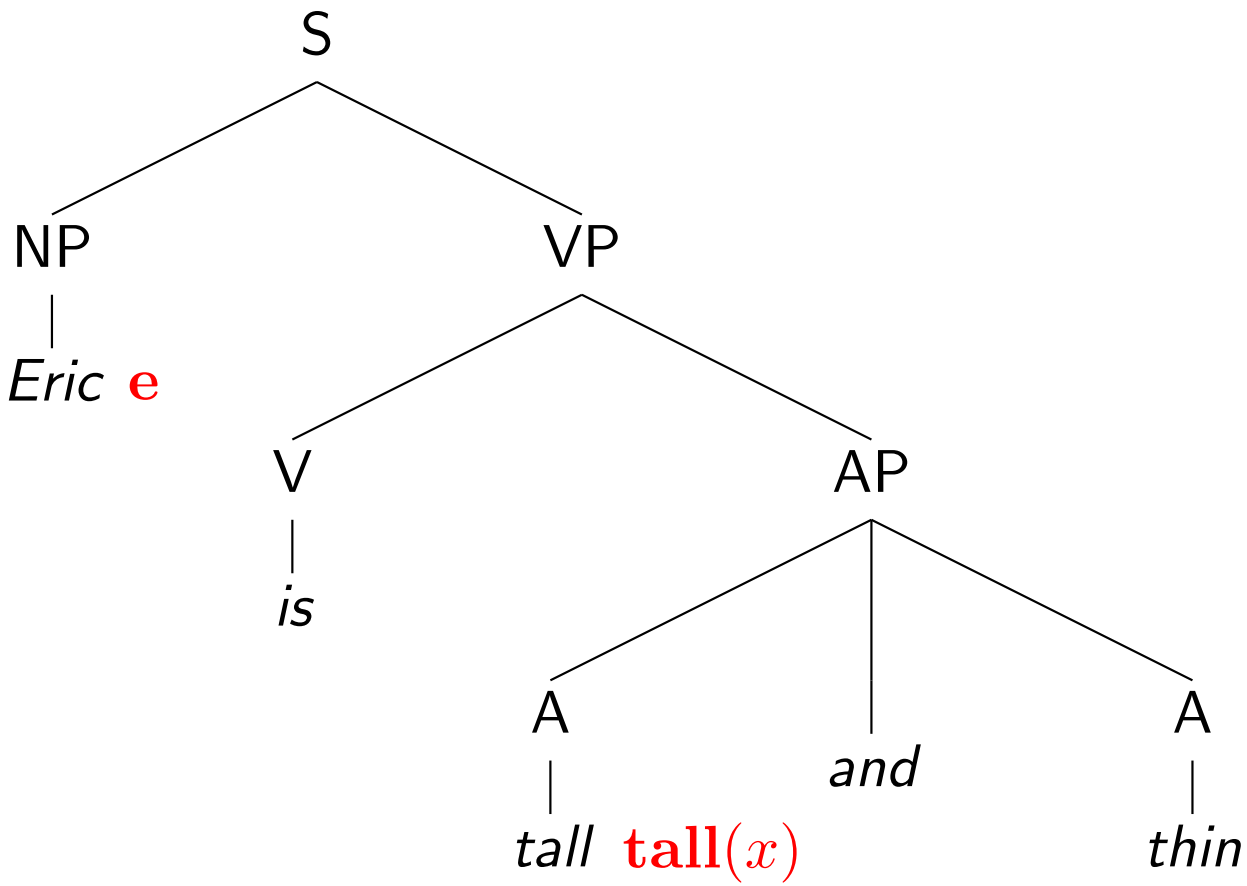
Compositionality



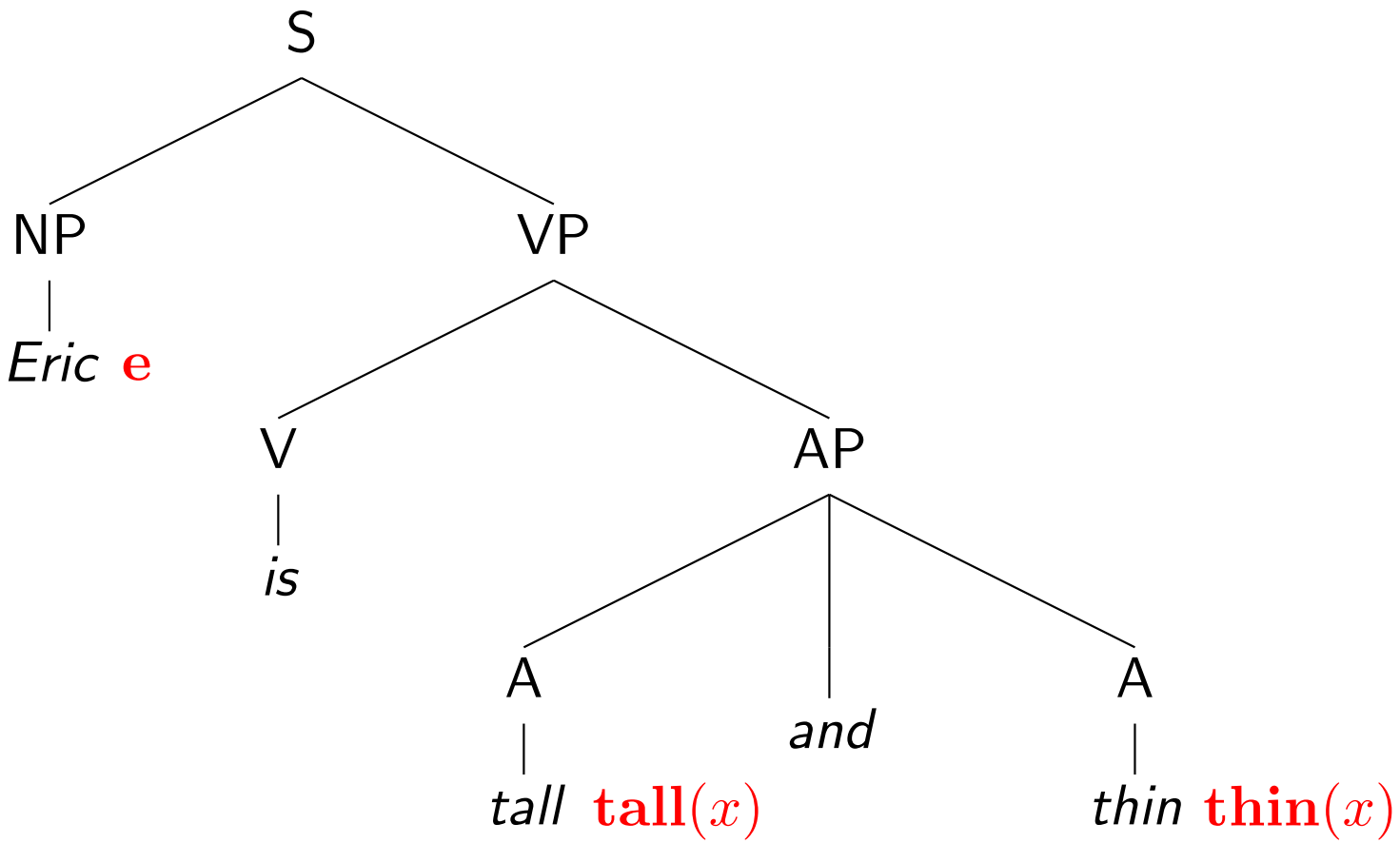
Compositionality



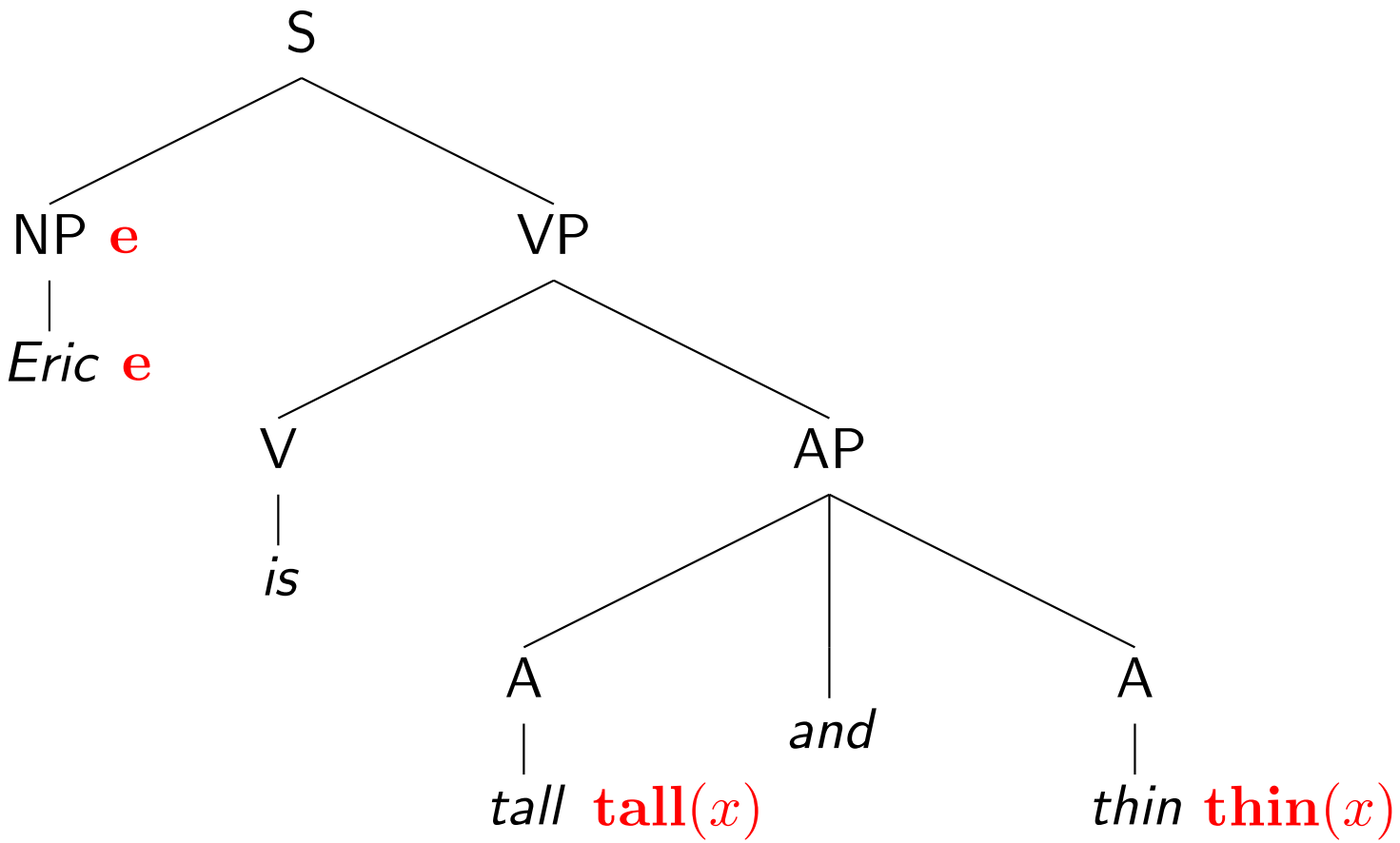
Compositionality



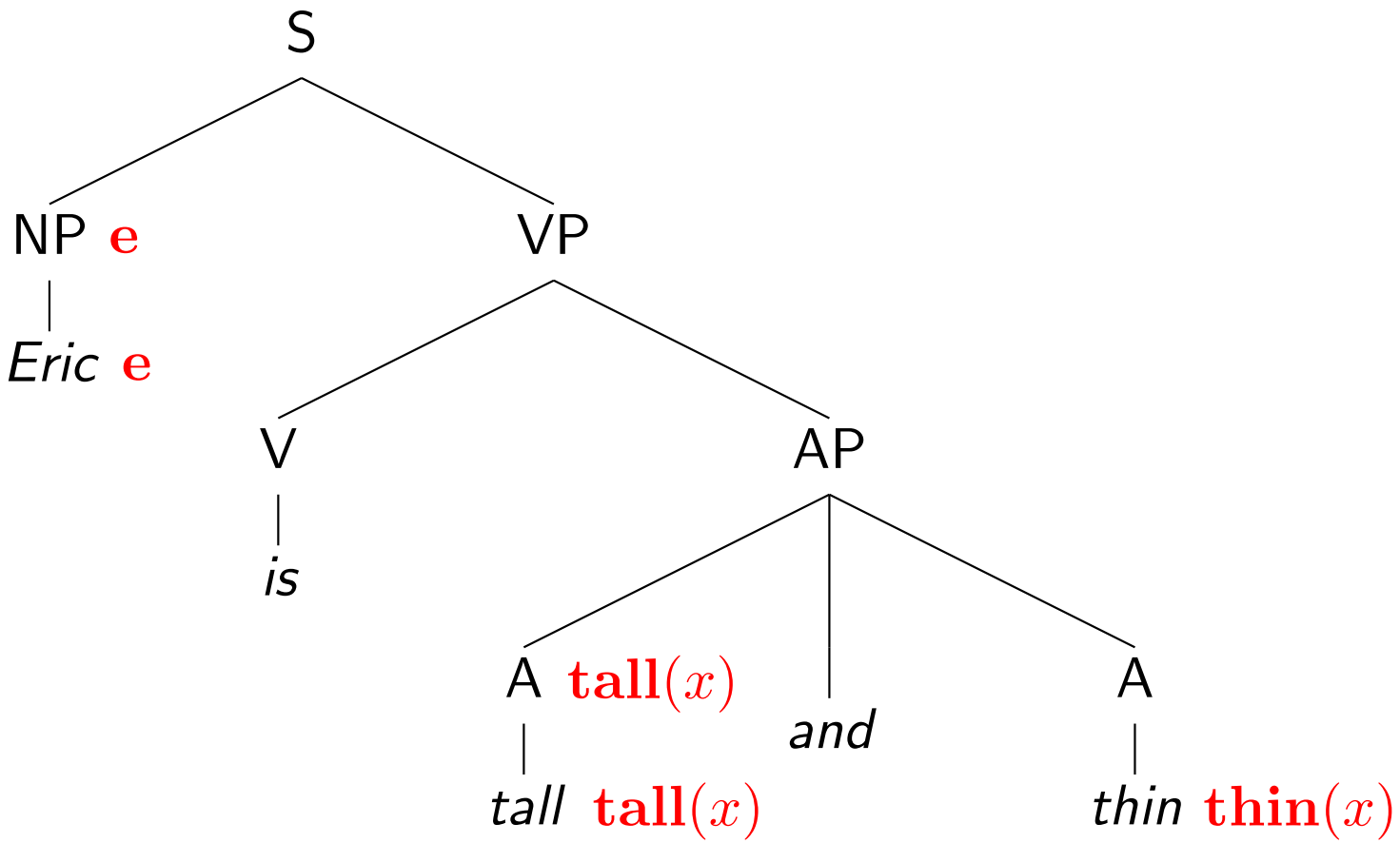
Compositionality



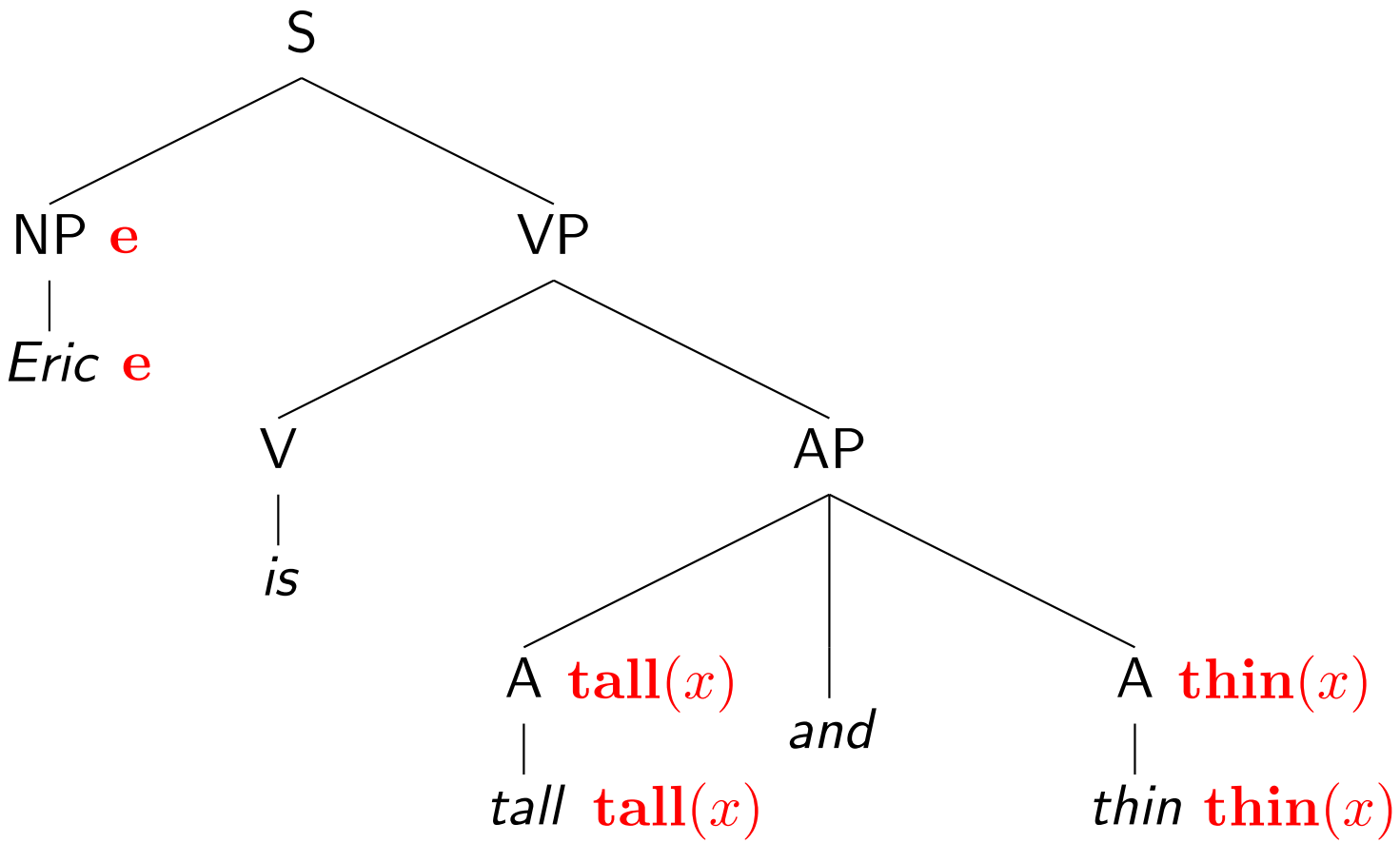
Compositionality



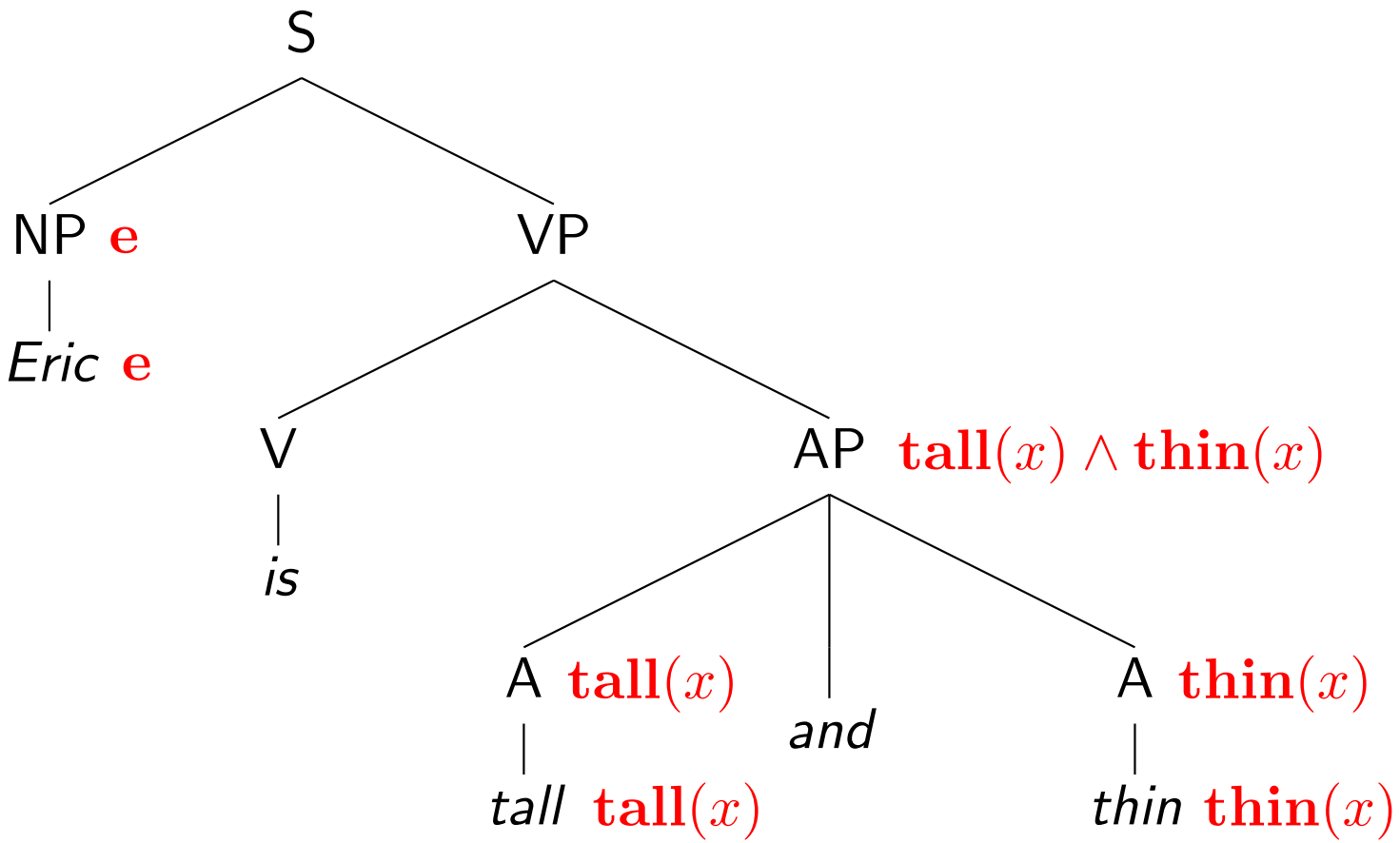
Compositionality



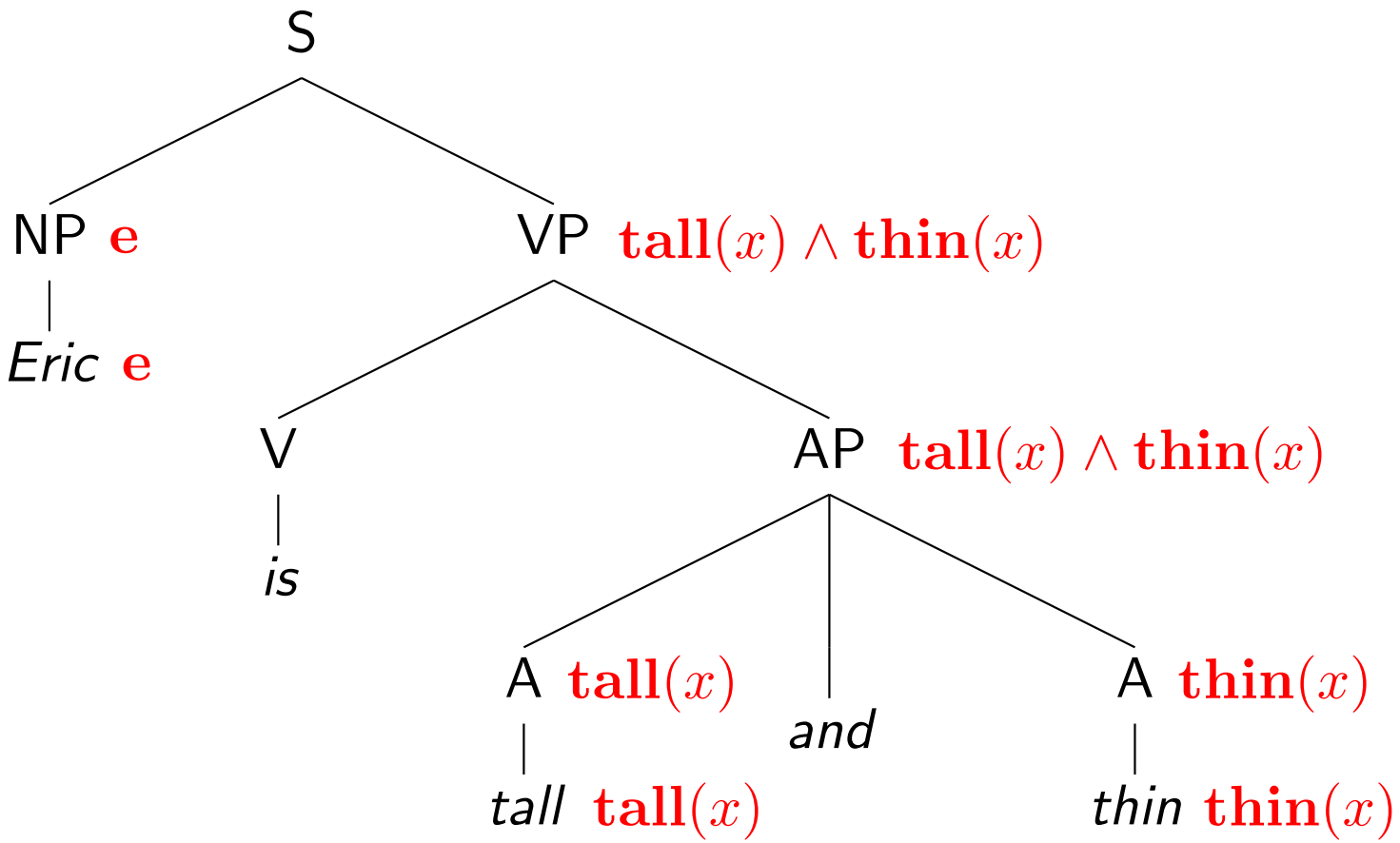
Compositionality



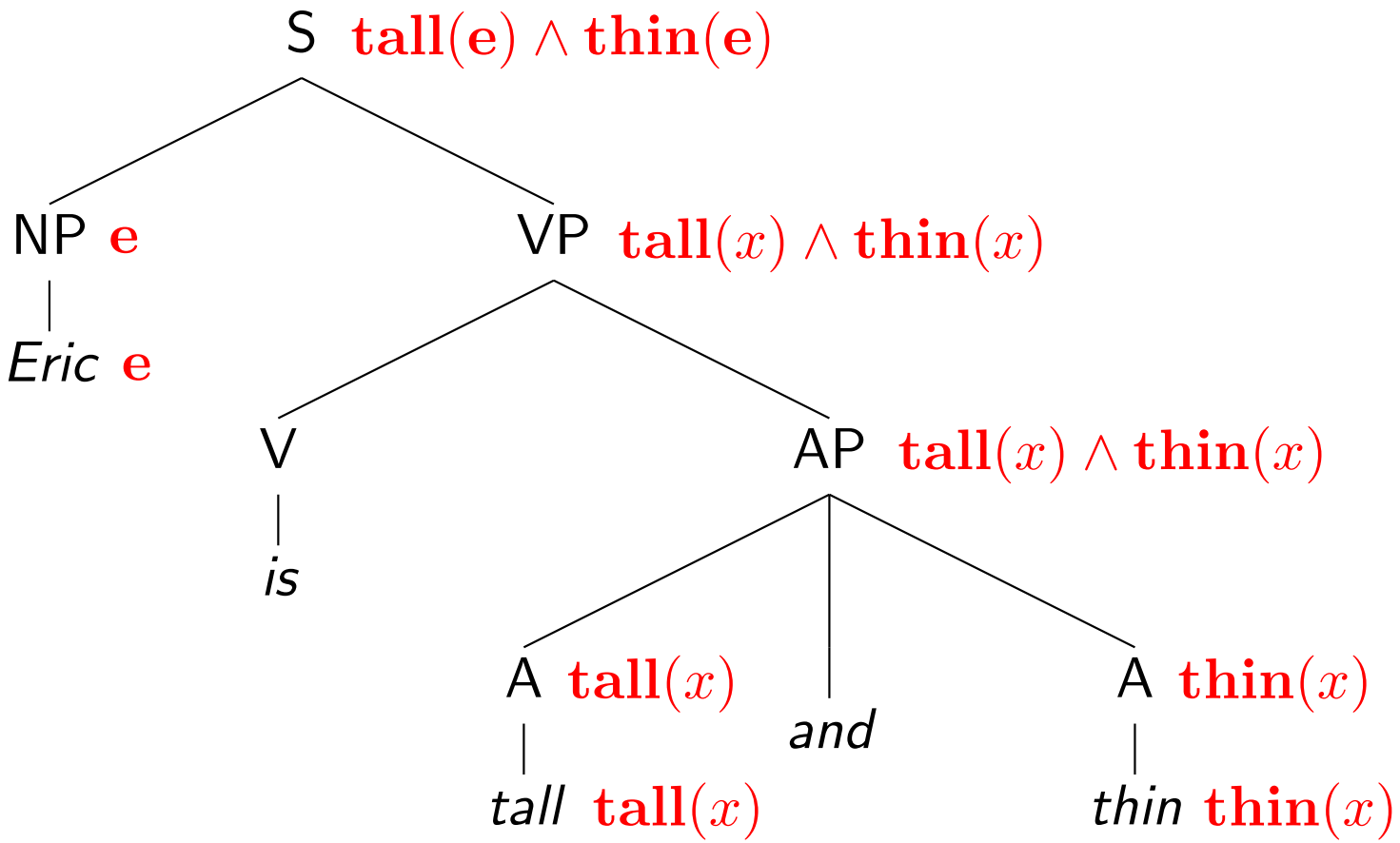
Compositionality



Compositionality



Compositionality



Compositionality

Let "tall" be a function from terms to formulas such that:

- $\mathrm{tall}(t) = \mathbf{tall}(t)$

Similarly, let:

- $\mathrm{thin}(t) = \mathbf{thin}(t)$

Let "and" be a function such that:

- $\mathrm{and}(f)(g)(t) = f(t) \wedge g(t)$

If we write $(A \to B)$ for the set of functions B^A then "and" is a function belonging to the following set:

$$(\mathscr{T} \to \mathscr{F}) \to ((\mathscr{T} \to \mathscr{F}) \to (\mathscr{T} \to \mathscr{F}))$$

where $\mathscr{T}$ and $\mathscr{F}$ are respectively the set of terms and the set of formulas of the object language.

Compositionality

Grammar:

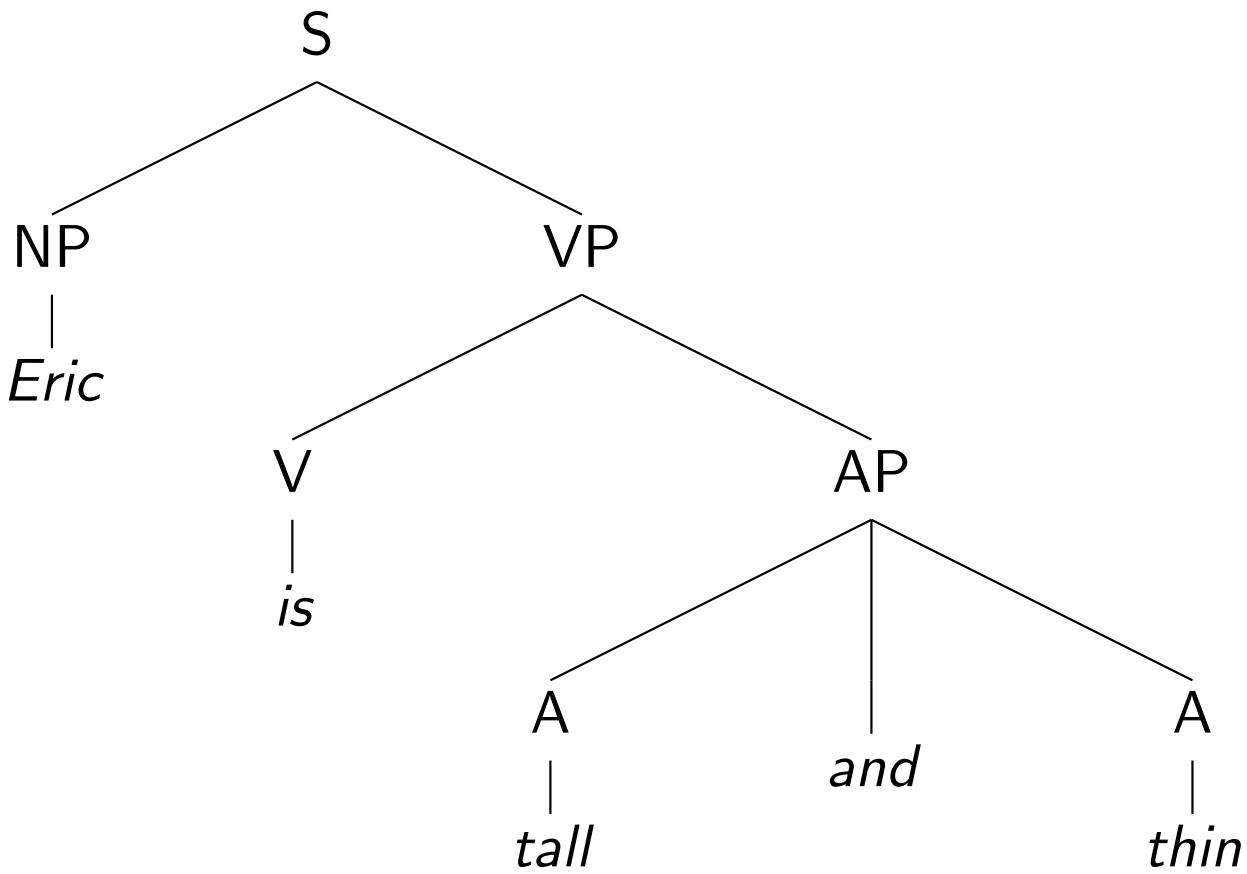
$$\mathrm{S} \rightarrow \mathrm{NP\ VP}$$
$$\mathrm{VP} \rightarrow \mathrm{V\ AP}$$
$$\mathrm{AP} \rightarrow \mathrm{A}\ \mathit{and}\ \mathrm{A}$$
$$\mathrm{A} \rightarrow \mathit{tall}$$
$$\mathrm{A} \rightarrow \mathit{thin}$$
$$\mathrm{NP} \rightarrow \mathit{Eric}$$

Semantic rules:

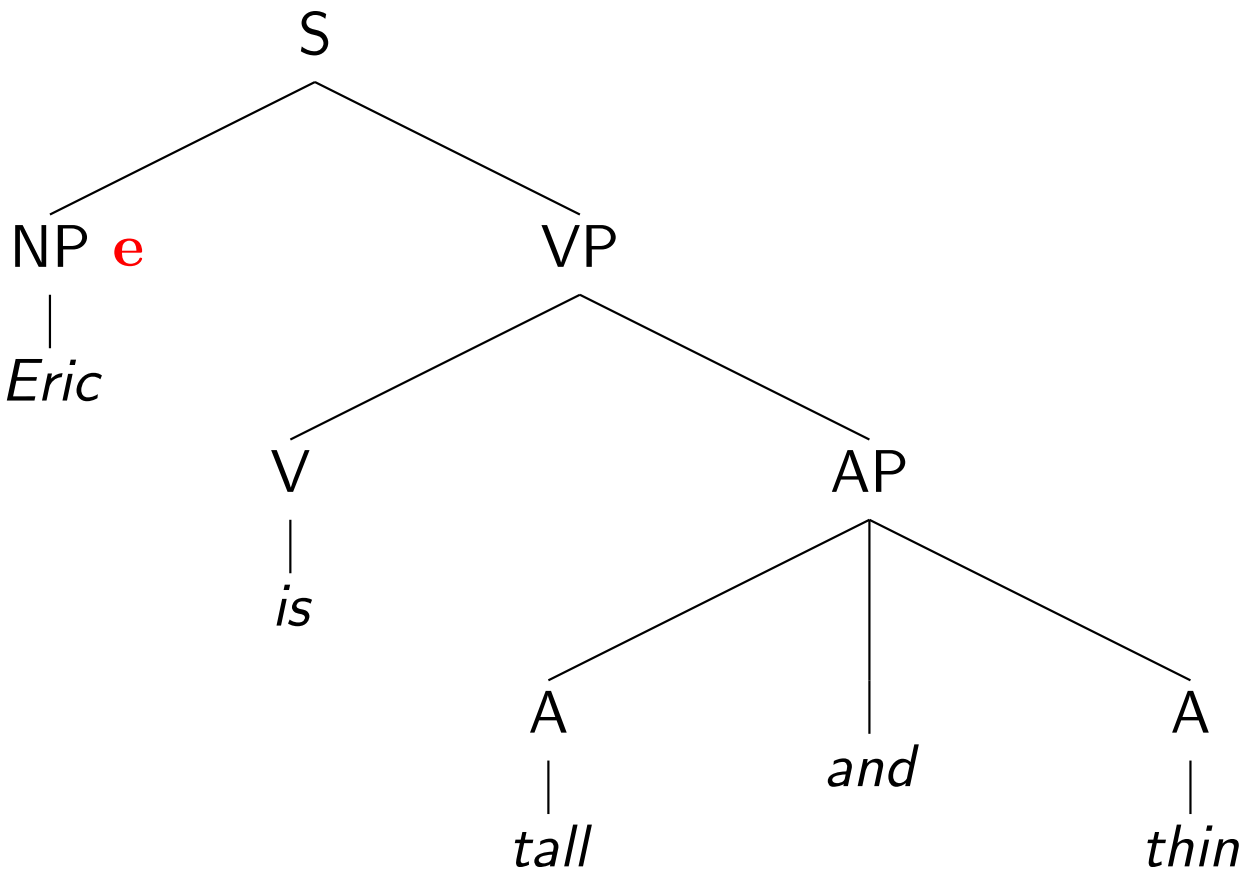
$$[\![\mathrm{S}]\!] = [\![\mathrm{VP}]\!]\,([\![\mathrm{NP}]\!])$$
$$[\![\mathrm{VP}]\!] = [\![\mathrm{AP}]\!]$$
$$[\![\mathrm{AP} = \mathrm{and}\,([\![\mathrm{A}_1]\!])\,([\![\mathrm{A}_2]\!])$$
$$[\![\mathrm{A}]\!] = \mathrm{tall}$$
$$[\![\mathrm{A}]\!] = \mathrm{thin}$$
$$[\![\mathrm{NP}]\!] = \mathbf{e}$$

Compositionality

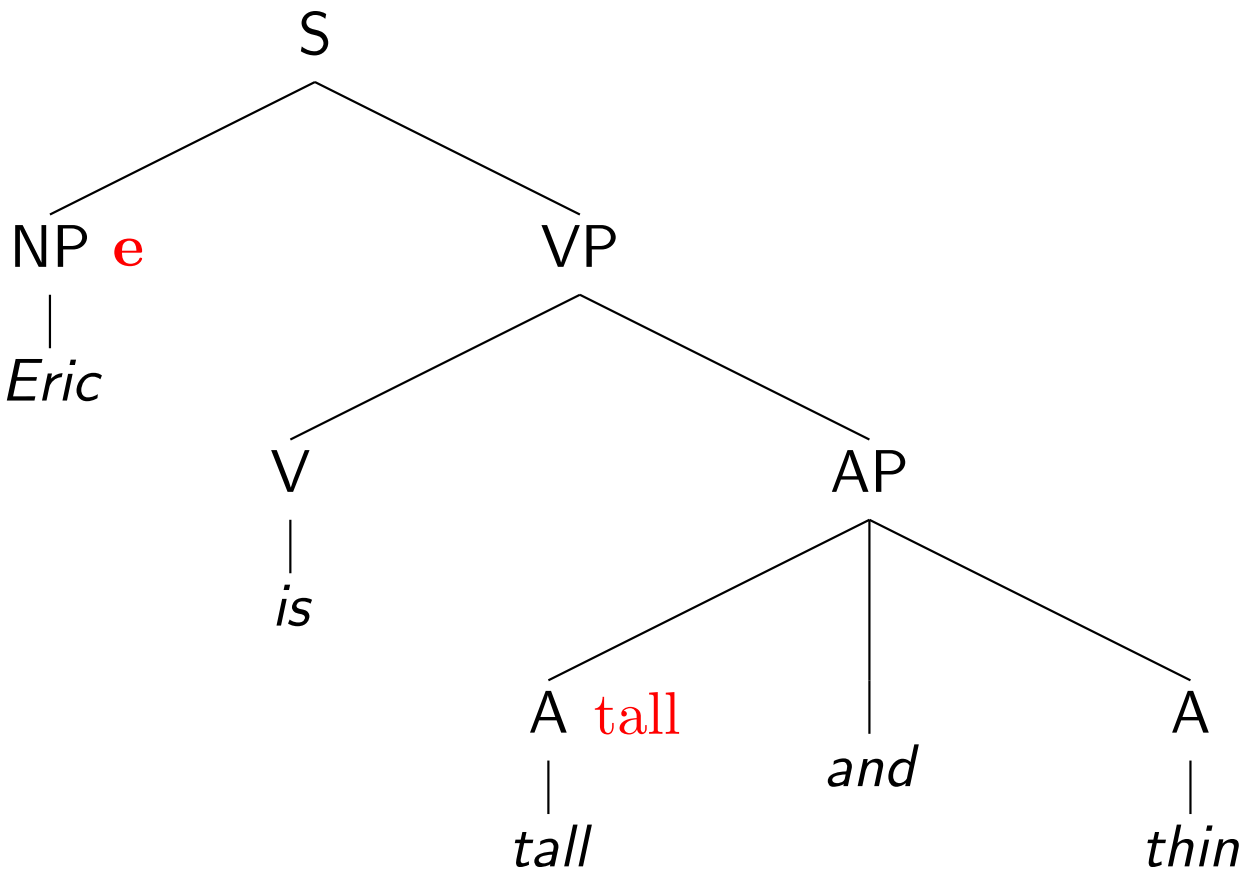
Compositionality



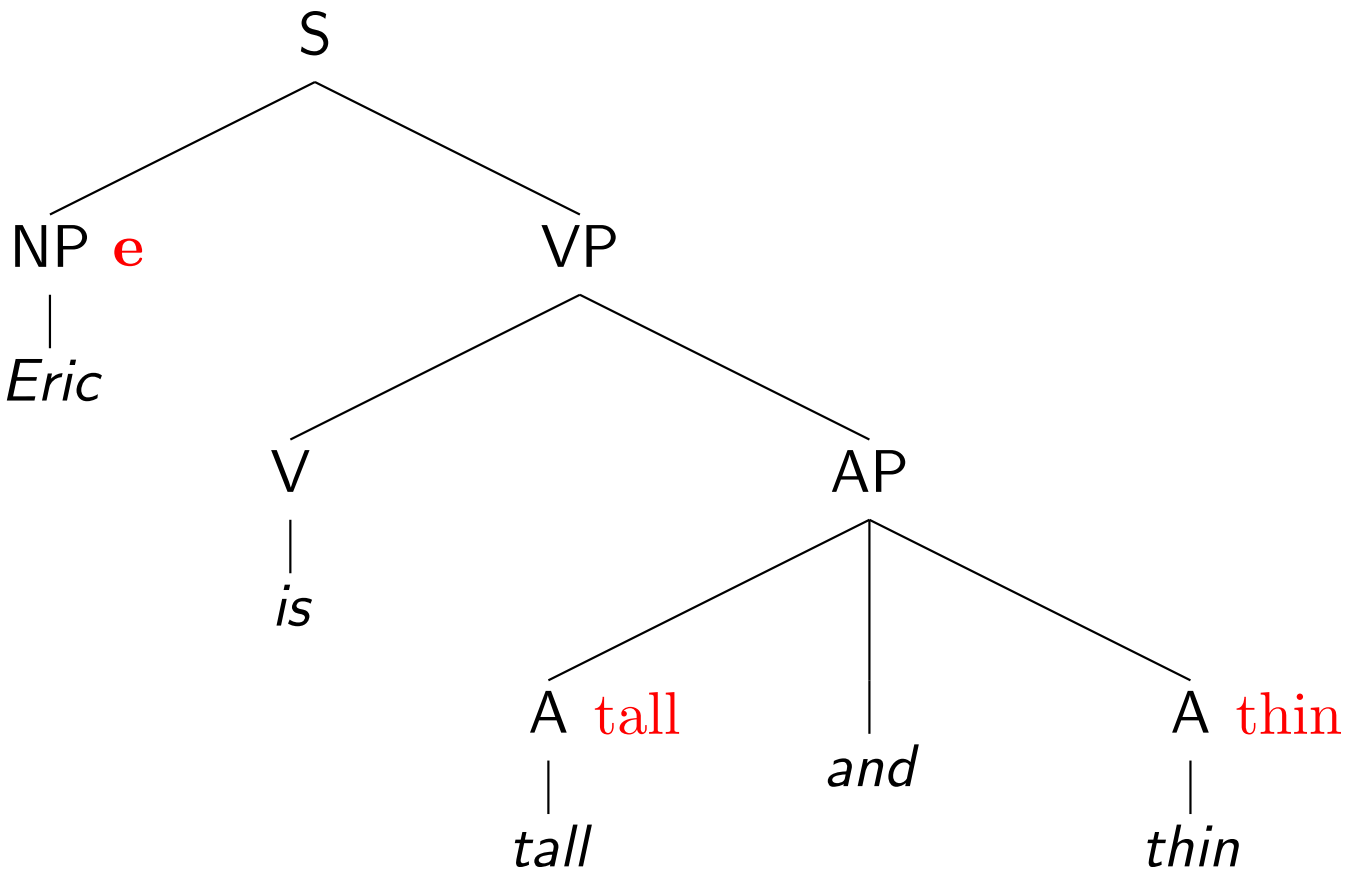
Compositionality



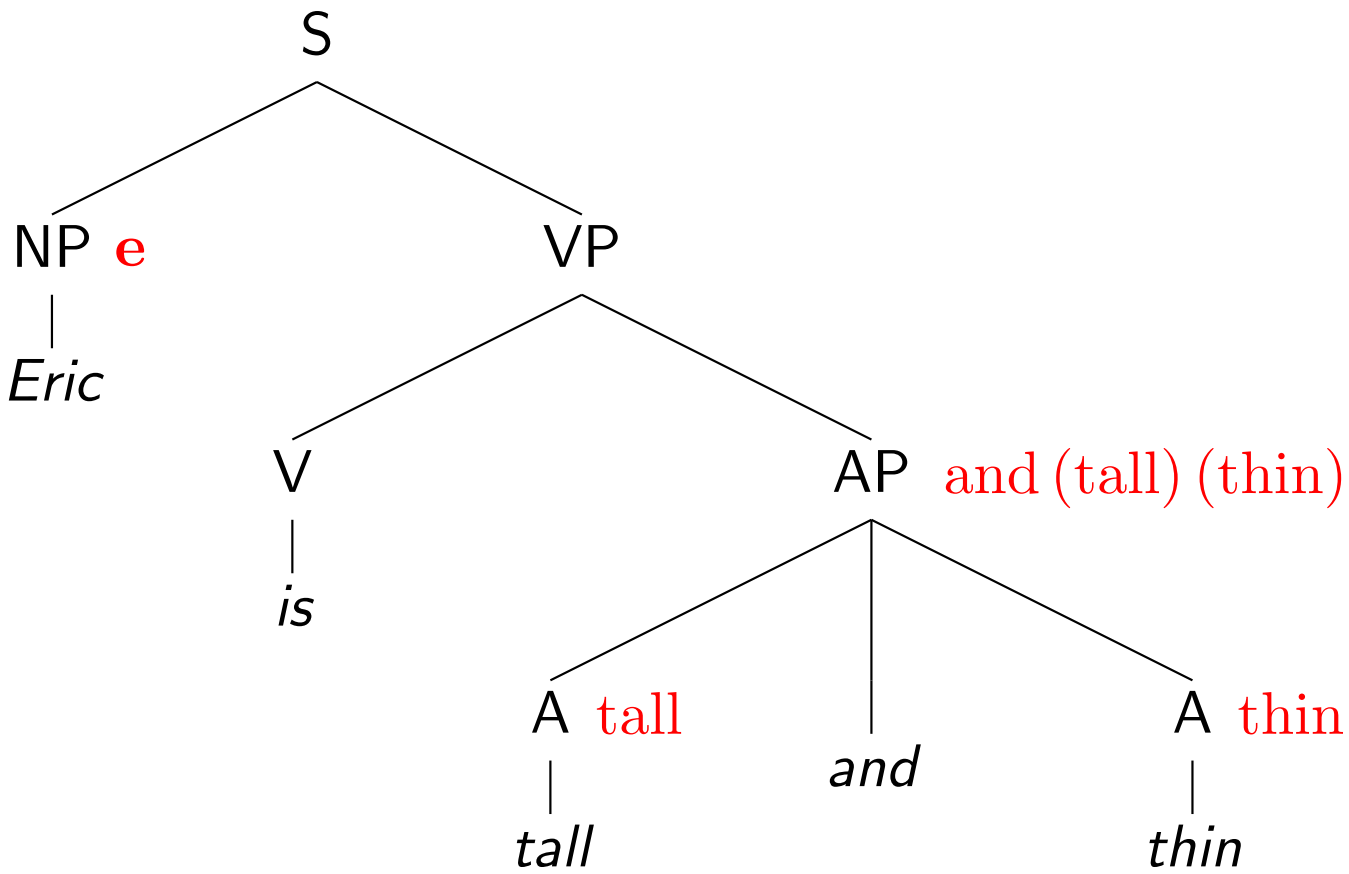
Compositionality



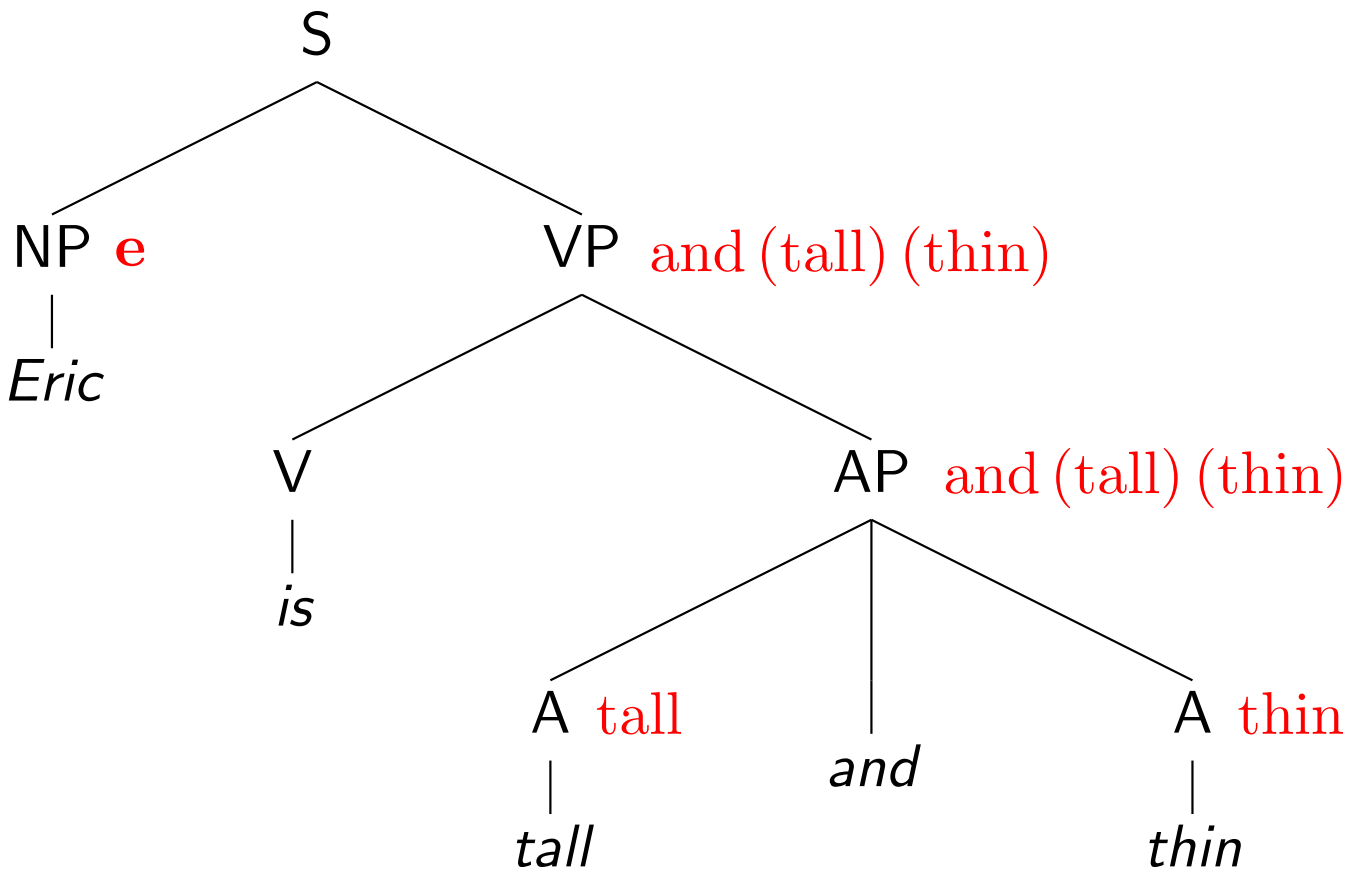
Compositionality



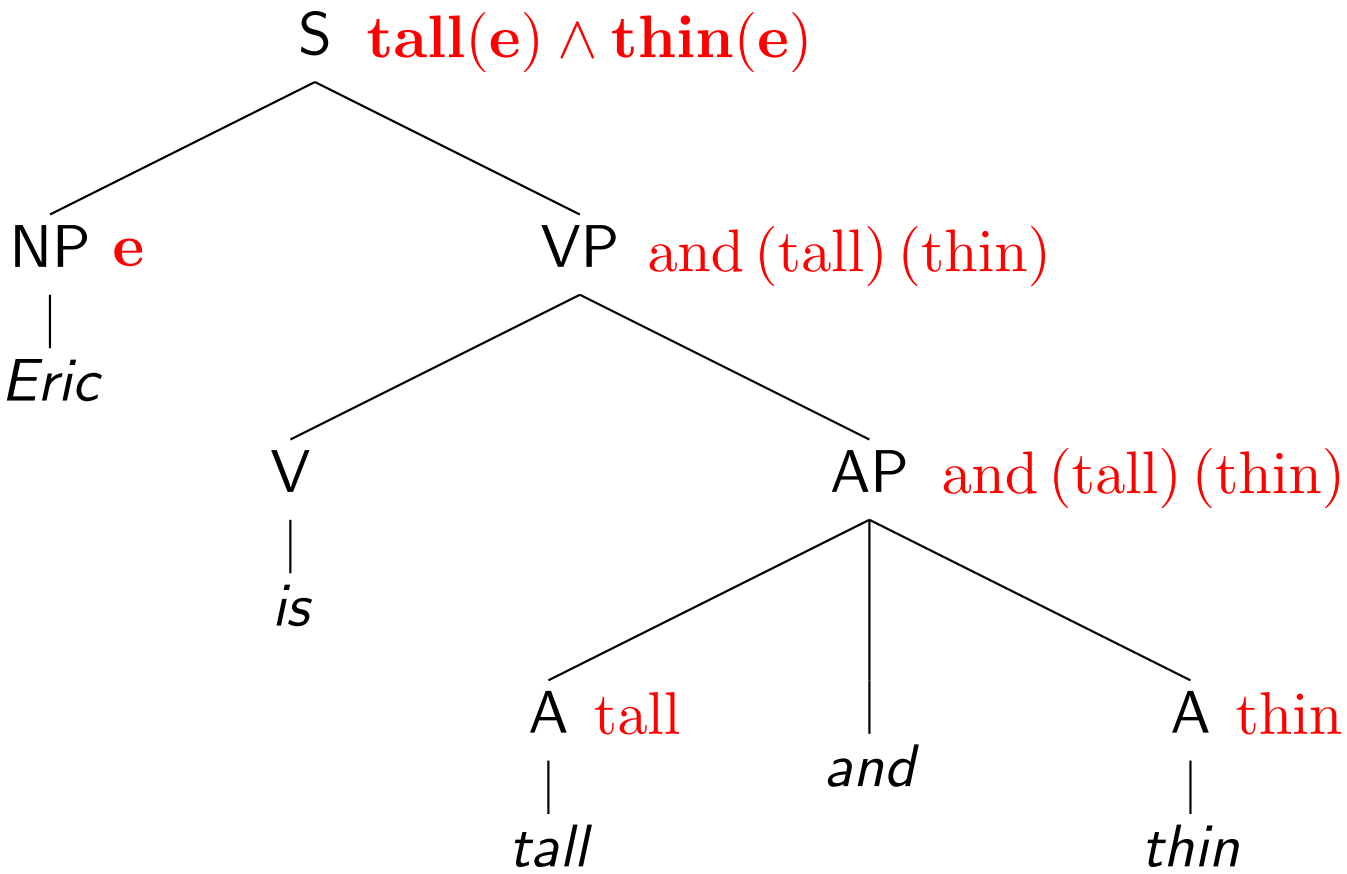
Compositionality



Compositionality



Compositionality



Outline

6. λ-Calculus
   - λ-Notatation
   - λ-Terms
   - β-Reduction
   - Simple types
   - interpretation
   - Logical constants

λ-Notatation

$$\text{“}2x + y\text{”}$$

λ-Notatation

$$\text{“}2x + y\text{”}$$

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

λ-Notatation

$$\text{“}2x + y\text{”}$$

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

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

λ-Notatation

$$\text{“}2x + y\text{”}$$

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

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

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

λ-Notatation

$$\text{“}2x + y\text{”}$$

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

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

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

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

λ-Notatation

$$\text{“}2x + y\text{”}$$

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

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

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

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

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

λ-Notatation

$$\text{“}2x + y\text{”}$$

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

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

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

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

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

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

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:

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

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:

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

$x[y := u] = y$, 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

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

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

Exercice:

Reduce the following terms:

1. $(\lambda x.\, y\, x\, x)\, z$
2. $(\lambda x y.\, y\, x\, x)\, y$
3. $(\lambda f x.\, f\, (f\, x)\, (\lambda y.\, y)\, x)\, (\lambda y z.\, z)$
4. $(\lambda x.\, x\, x)\, (\lambda y.\, y)$
5. $(\lambda x.\, x\, x)\, (\lambda x.\, x\, x)$

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.

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 $(\alpha \to \beta)$ 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.

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

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 $\mathscr{A}$.

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 $\mathscr{A}$.

Simple types

Typing environment

Given a signature a typing environment Γ is a finite set of ordered pairs $(x, \alpha) \in \mathscr{X} \times \mathscr{T}(\mathscr{A})$ such that $(x, \alpha), (x, \beta) \in \Gamma$ implies $\alpha = \beta$.

Given a typing environment Γ such that for every $(y, \beta) \in Gamma\ y \neq x$, we write "$\Gamma, x{:}\alpha$" for the typing environment "$\Gamma \cup \{(x, \alpha)\}$".

Simple types

Typing environment

Given a signature a typing environment Γ is a finite set of ordered pairs $(x, \alpha) \in \mathscr{X} \times \mathscr{T}(\mathscr{A})$ such that $(x, \alpha), (x, \beta) \in \Gamma$ implies $\alpha = \beta$.

Given a typing environment Γ such that for every $(y, \beta) \in Gamma\ y \neq x$, we write "$\Gamma, x{:}\alpha$" for the typing environment "$\Gamma \cup \{(x, \alpha)\}$".

Typing judgement

A typing judgement is an expression of the form

$$\Gamma \vdash t : \alpha$$

where Γ is a typing environment, t a λ-term, and α a simple type.

Simple types

Typing rules

$$\Gamma \vdash c : \tau(c)$$

$$\Gamma, x : \alpha \vdash x : \alpha$$

$$\frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash (\lambda x.\, t) : (\alpha \to \beta)}$$

$$\frac{\Gamma \vdash t : (\alpha \to \beta) \quad \Gamma \vdash u : \alpha}{\Gamma \vdash (t\, u) : \beta}$$

Simple types

Typable terms

A λ-term t is typable if and only if there exist a typing environment Γ and a simple type α such that

$$\Gamma \vdash t : \alpha$$

Simple types

Typable terms

A λ-term t is typable if and only if there exist a typing environment Γ and a simple type α such that

$$\Gamma \vdash t : \alpha$$

Normalization

Every typable term has a normal form.

Interpretation

Definition à la Church

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

Interpretation

Model

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

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

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:

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

Logical constants

Interpretation

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

- $D_{\mathrm{t}} = 2$;
- $\mathcal{I}(\mathbf{not}) = \{(0,1),(1,0)\}$;
- $\mathcal{I}(\mathbf{and}) = \{(0,\{(0,0),(1,0)\}),(1,\{(0,0),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{or}) = \{(0,\{(0,0),(1,1)\}),(1,\{(0,1),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{implies}) = \{(0,\{(0,1),(1,1)\}),(1,\{(0,0),(1,1)\})\}$;
- $\mathcal{I}(\mathbf{all})(f) = 1$ iff $f(a) = 1$ for every $a \in D_{\mathrm{e}}$;
- $\mathcal{I}(\mathbf{exists})(f) = 1$ iff $f(a) = 1$ for some $a \in D_{\mathrm{e}}$.

Logical constants

Notations

We write:

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

Logical constants

Consider the following non-logical constants:

$$\mathbf{e} : \mathrm{e}$$
$$\mathbf{tall} : \mathrm{e} \to \mathrm{t}$$
$$\mathbf{thin} : \mathrm{e} \to \mathrm{t}$$

Logical constants

Consider the following non-logical constants:

$$\mathbf{e} : \mathrm{e}$$
$$\mathbf{tall} : \mathrm{e} \to \mathrm{t}$$
$$\mathbf{thin} : \mathrm{e} \to \mathrm{t}$$

Grammar:

$$\mathrm{S} \to \mathrm{NP}\ \mathrm{VP}$$
$$\mathrm{VP} \to \mathrm{V}\ \mathrm{AP}$$
$$\mathrm{AP} \to \mathrm{A}\ \mathit{and}\ \mathrm{A}$$
$$\mathrm{A} \to \mathit{tall}$$
$$\mathrm{A} \to \mathit{thin}$$
$$\mathrm{NP} \to \mathit{Eric}$$

Logical constants

Consider the following non-logical constants:

$$\mathbf{e} : \mathrm{e}$$
$$\mathbf{tall} : \mathrm{e} \to \mathrm{t}$$
$$\mathbf{thin} : \mathrm{e} \to \mathrm{t}$$

Grammar:

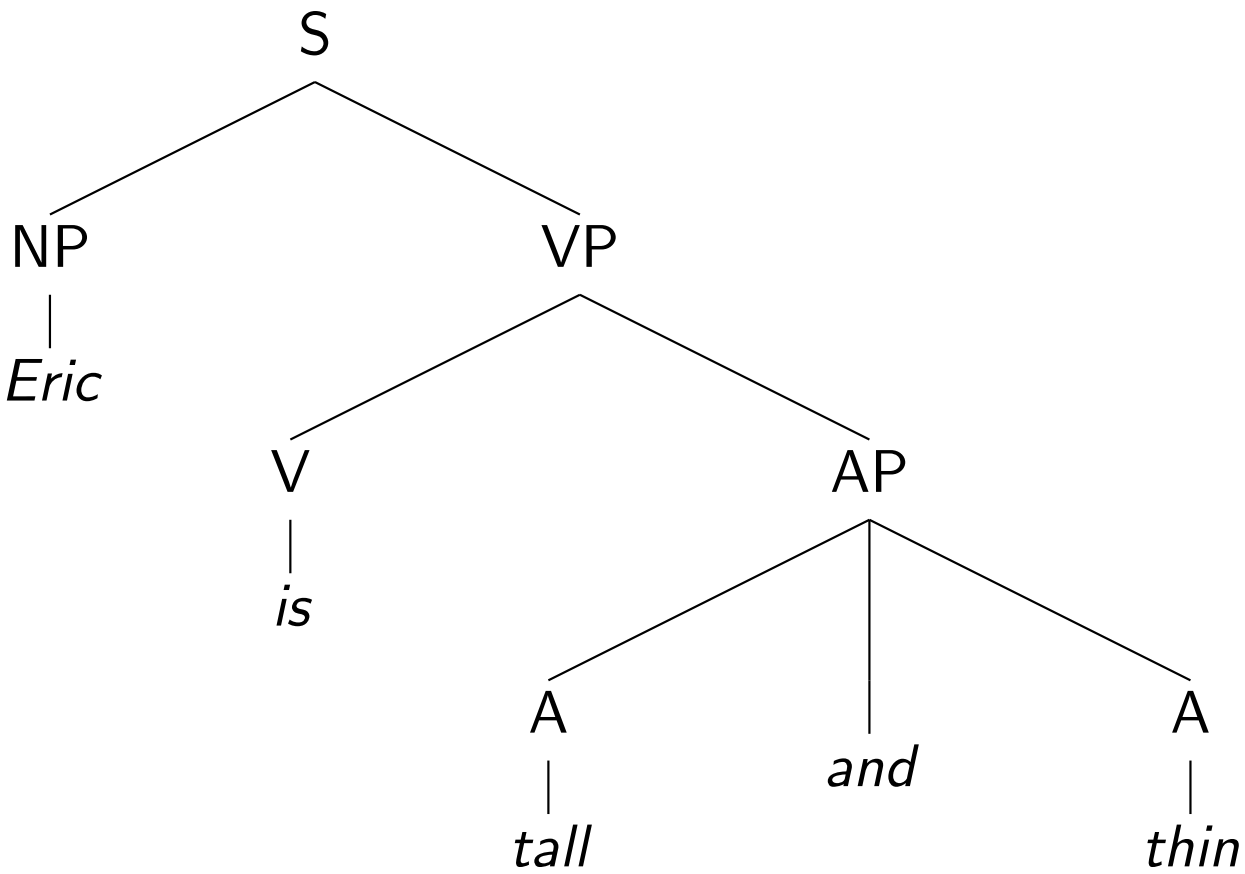
$$\begin{aligned}
\mathrm{S} &\to \mathrm{NP}\ \mathrm{VP} \\
\mathrm{VP} &\to \mathrm{V}\ \mathrm{AP} \\
\mathrm{AP} &\to \mathrm{A}\ \mathit{and}\ \mathrm{A} \\
\mathrm{A} &\to \mathit{tall} \\
\mathrm{A} &\to \mathit{thin} \\
\mathrm{NP} &\to \mathit{Eric}
\end{aligned}$$

Semantic rules:

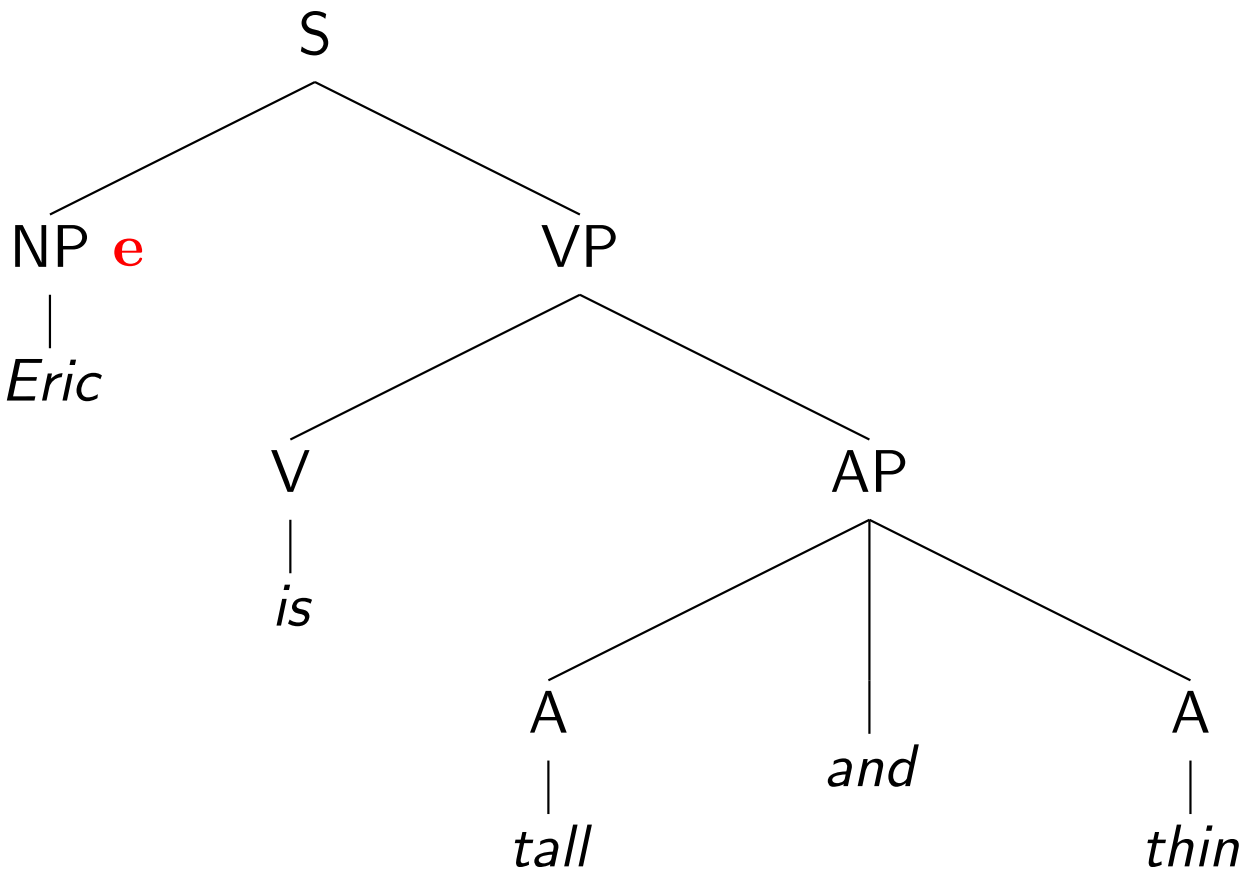
$$\begin{aligned}
[\![\mathrm{S}]\!] &= [\![\mathrm{VP}]\!]\,[\![\mathrm{NP}]\!] \\
[\![\mathrm{VP}]\!] &= [\![\mathrm{AP}]\!] \\
[\![\mathrm{AP}]\!] &= \lambda x.\,([\![\mathrm{A}_1]\!]\,x) \wedge ([\![\mathrm{A}_2]\!]\,x) \\
[\![\mathrm{A}]\!] &= \lambda x.\,\mathbf{tall}\,x \\
[\![\mathrm{A}]\!] &= \lambda x.\,\mathbf{thin}\,x \\
[\![\mathrm{NP}]\!] &= \mathbf{e}
\end{aligned}$$

Logical constants

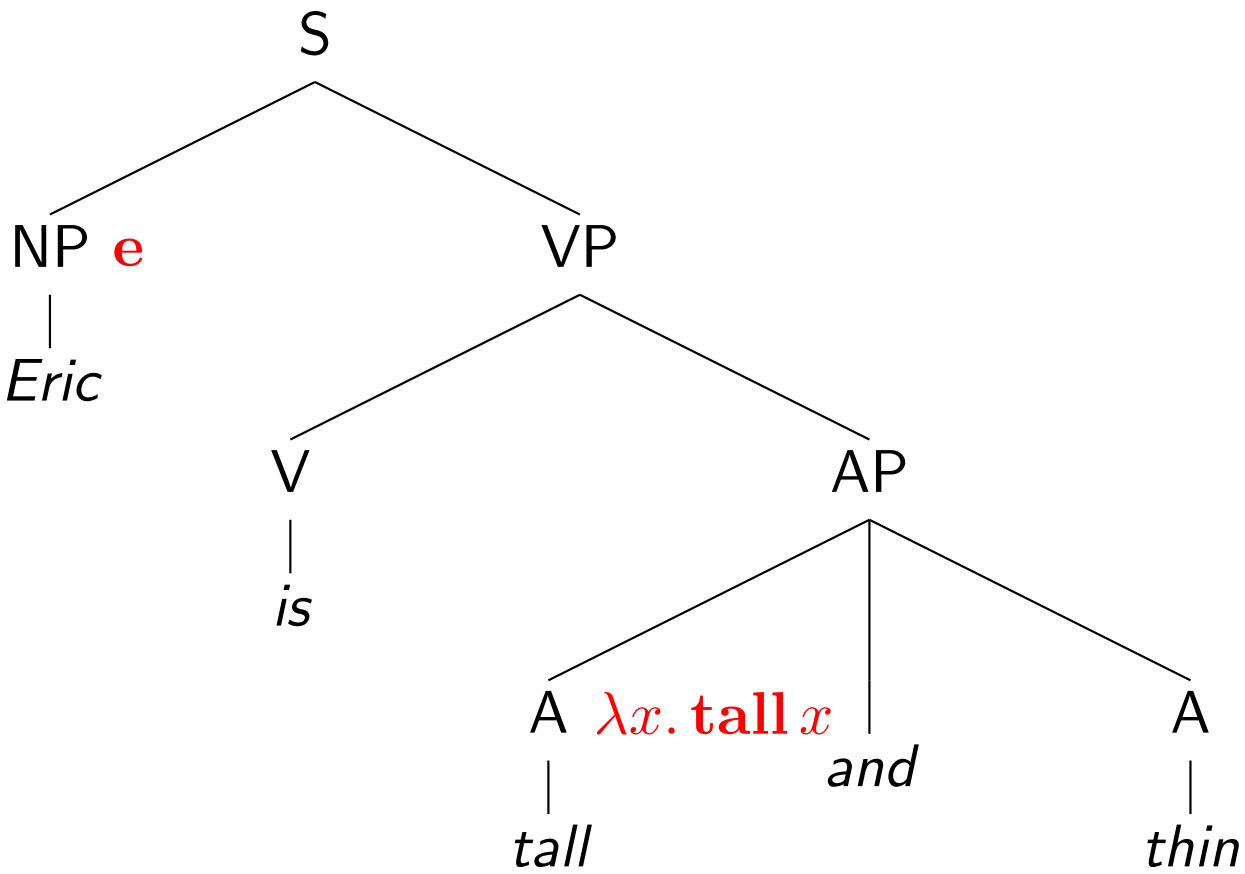
Logical constants



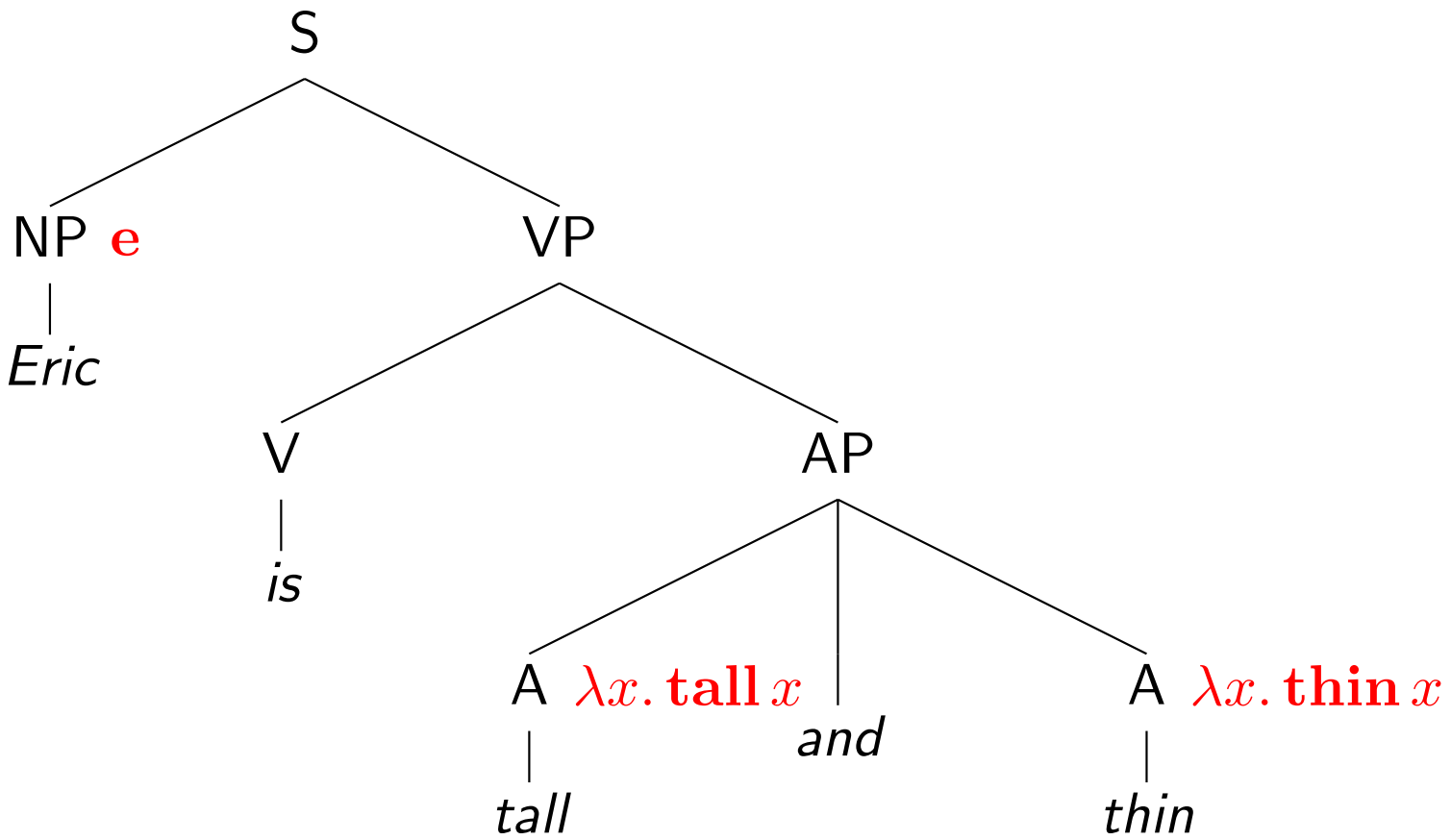
Logical constants



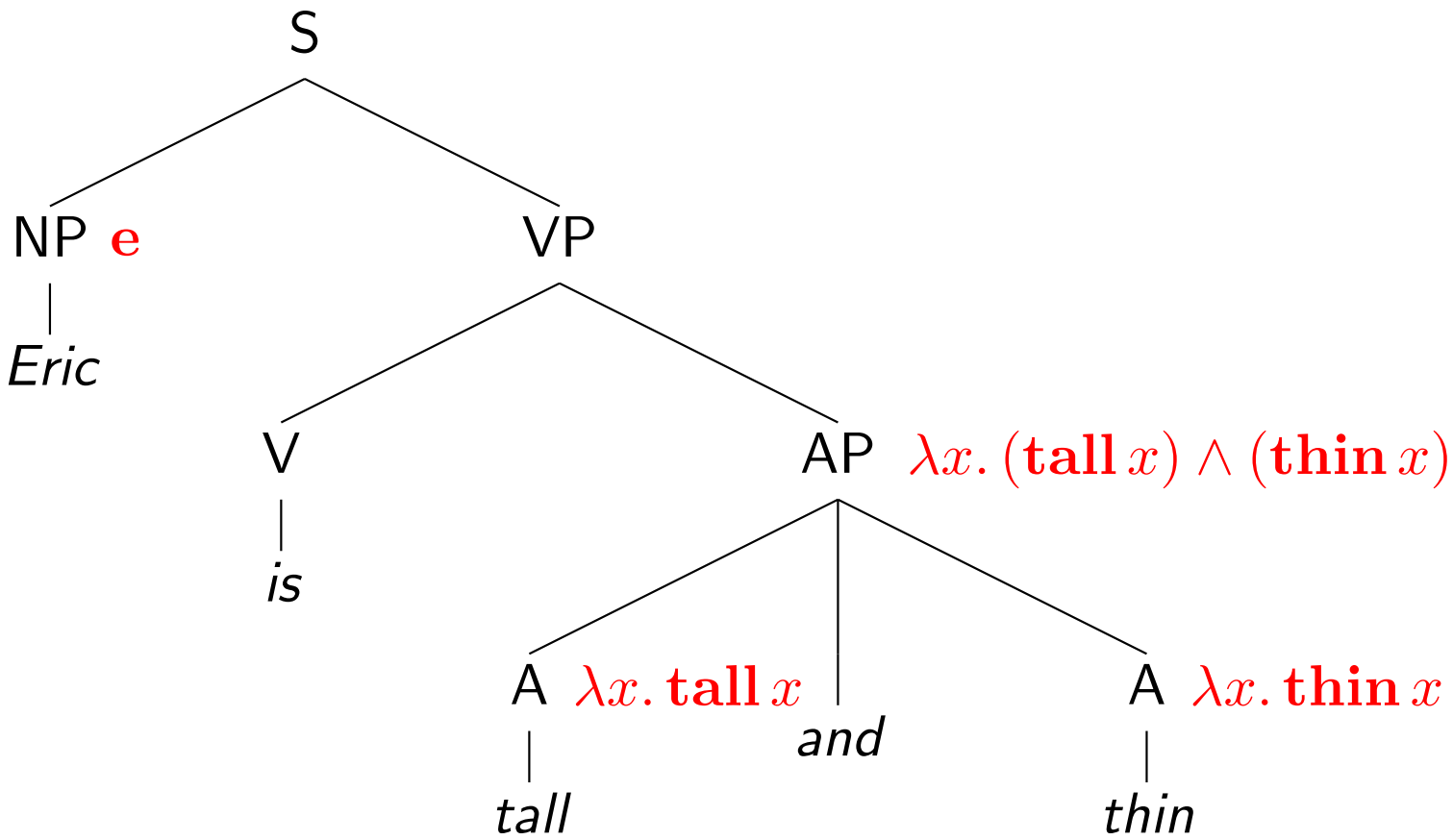
Logical constants



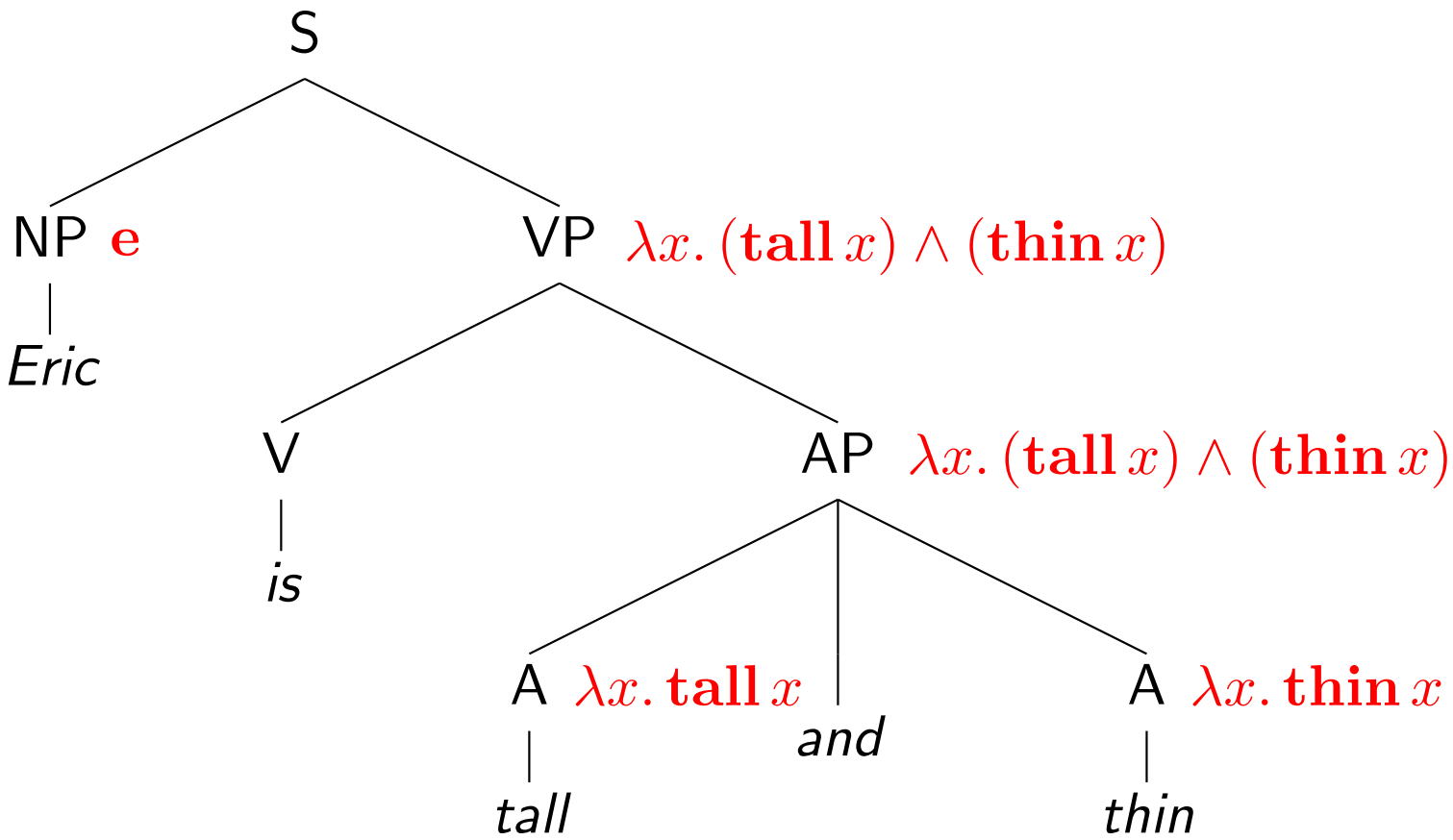
Logical constants



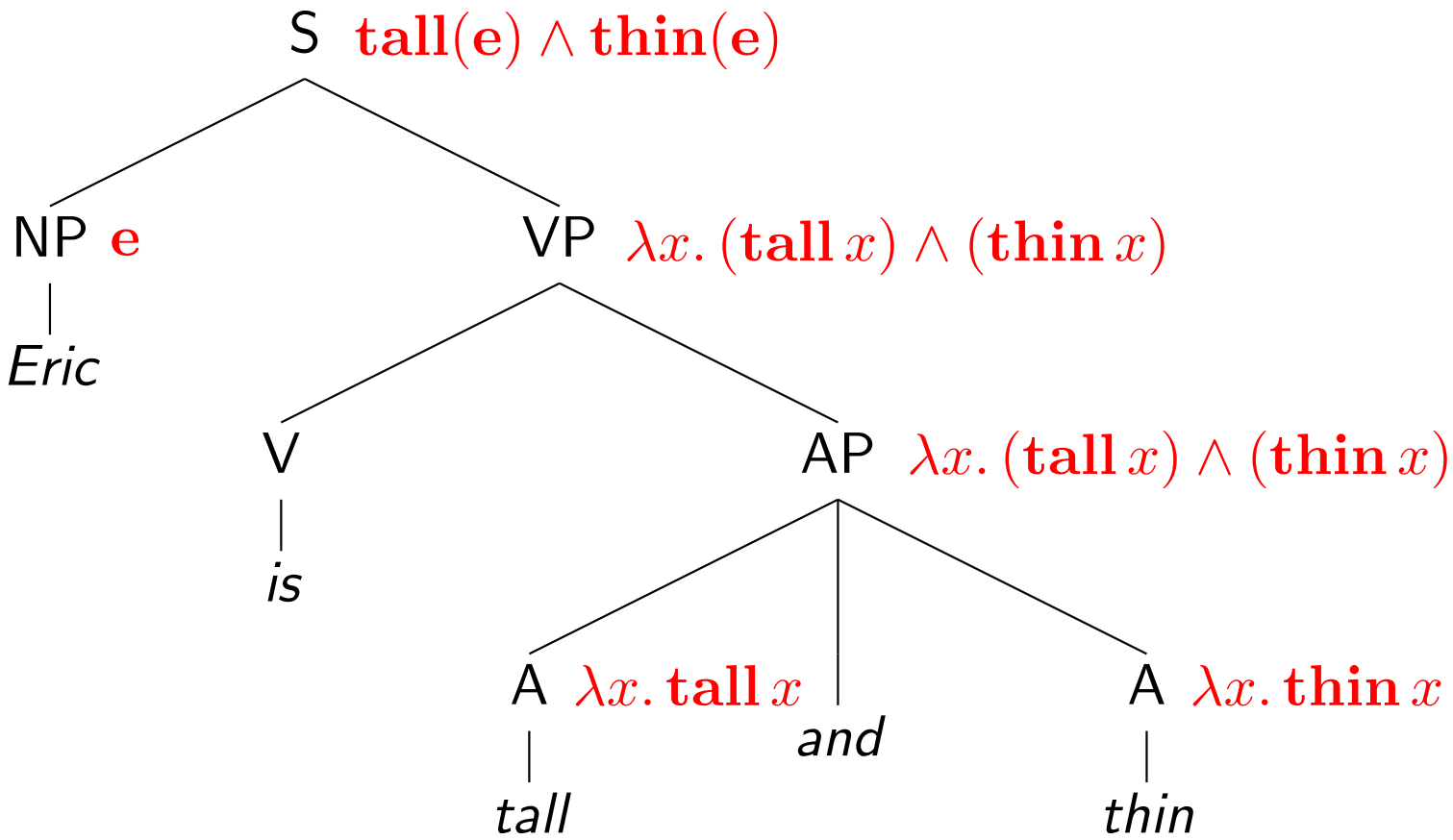
Logical constants



Logical constants



Logical constants



Outline

The homomorphism requirement

Richard Montague (1970)
Universal Grammar, *Theoria* 36:373–398.

- Semantics must be obtained as a homorphic image of syntax.

Abstract parse tree

Grammar:

$$
\begin{aligned}
P_0 &: \quad \mathrm{S} \rightarrow \mathrm{NP\ VP} \\
P_1 &: \quad \mathrm{VP} \rightarrow \mathrm{V\ AP} \\
P_2 &: \quad \mathrm{AP} \rightarrow \mathrm{A}\ \mathit{and}\ \mathrm{A} \\
P_3 &: \quad \mathrm{A} \rightarrow \mathit{tall} \\
P_4 &: \quad \mathrm{A} \rightarrow \mathit{thin} \\
P_5 &: \quad \mathrm{NP} \rightarrow \mathit{Eric} \\
P_6 &: \quad \mathrm{V} \rightarrow \mathit{is}
\end{aligned}
$$

Abstract parse tree

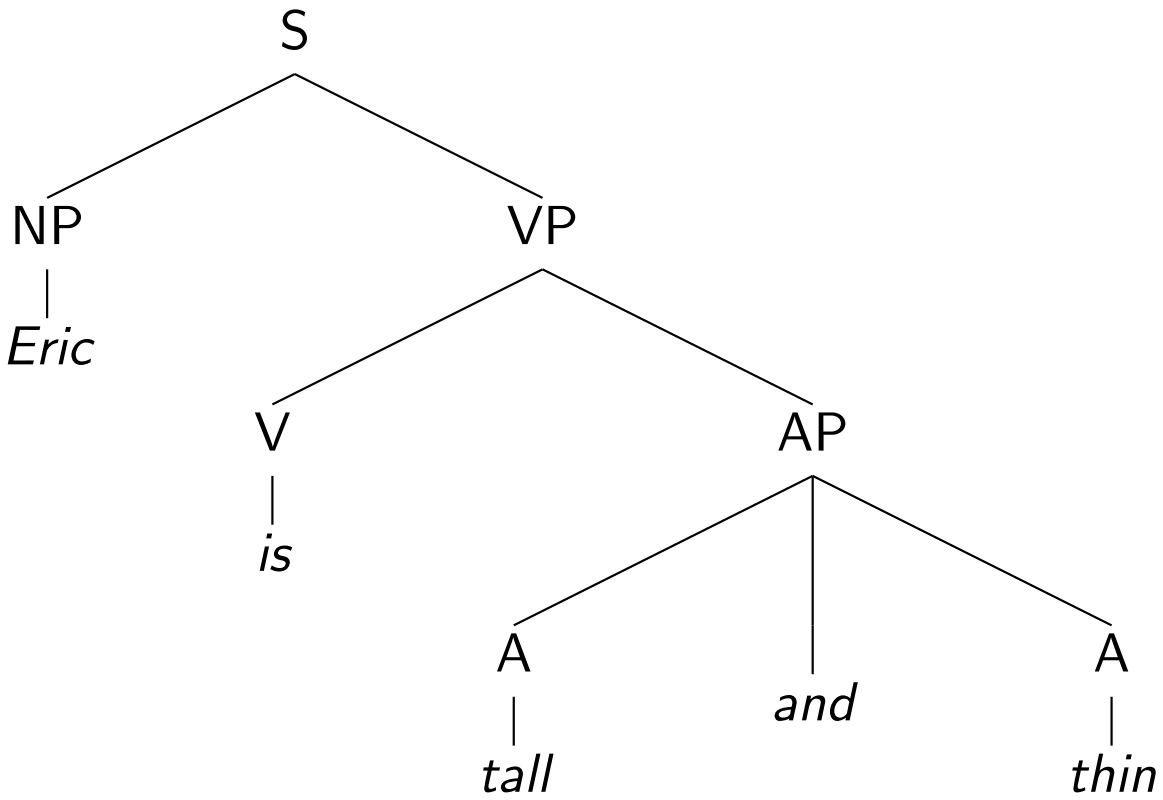
Grammar:

$$
\begin{array}{lll}
P_0: & \mathrm{S} & \rightarrow \mathrm{NP\ VP} \\
P_1: & \mathrm{VP} & \rightarrow \mathrm{V\ AP} \\
P_2: & \mathrm{AP} & \rightarrow \mathrm{A}\ \mathit{and}\ \mathrm{A} \\
P_3: & \mathrm{A} & \rightarrow \mathit{tall} \\
P_4: & \mathrm{A} & \rightarrow \mathit{thin} \\
P_5: & \mathrm{NP} & \rightarrow \mathit{Eric} \\
P_6: & \mathrm{V} & \rightarrow \mathit{is}
\end{array}
$$

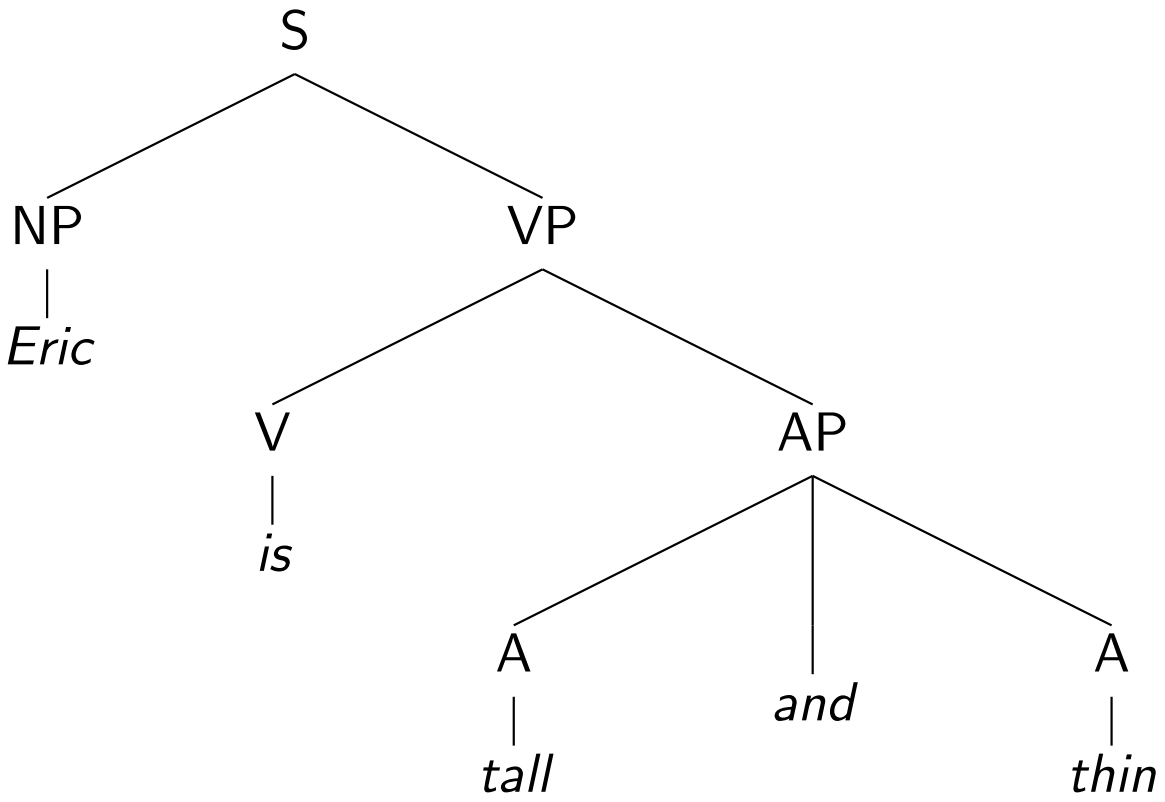
First-order multisorted signature:

$$
\begin{array}{lll}
\mathrm{P}_0: & \mathrm{NP} \times \mathrm{VP} & \rightarrow \mathrm{S} \\
\mathrm{P}_1: & \mathrm{V} \times \mathrm{AP} & \rightarrow \mathrm{VP} \\
\mathrm{P}_2: & \mathrm{A} \times \mathrm{A} & \rightarrow \mathrm{AP} \\
\mathrm{P}_3: & \mathrm{A} & \\
\mathrm{P}_4: & \mathrm{A} & \\
\mathrm{P}_5: & \mathrm{NP} & \\
\mathrm{P}_6: & \mathrm{V} &
\end{array}
$$

Abstract parse tree



Abstract parse tree



$$\mathrm{P}_0(\mathrm{P}_5, \mathrm{P}_1(\mathrm{P}_6, \mathrm{P}_2(\mathrm{P}_3, \mathrm{P}_4)))$$

Lexicalized case

Grammar:

$$P_0: \quad \text{S} \rightarrow \text{NP } is \text{ AP}$$
$$P_1: \quad \text{AP} \rightarrow \text{A } and \text{ A}$$
$$P_2: \quad \text{A} \rightarrow tall$$
$$P_3: \quad \text{A} \rightarrow thin$$
$$P_4: \quad \text{NP} \rightarrow Eric$$

Lexicalized case

Grammar:

$$
\begin{array}{lll}
P_0: & \mathrm{S} \rightarrow \mathrm{NP}\ \mathit{is}\ \mathrm{AP} \\
P_1: & \mathrm{AP} \rightarrow \mathrm{A}\ \mathit{and}\ \mathrm{A} \\
P_2: & \mathrm{A} \rightarrow \mathit{tall} \\
P_3: & \mathrm{A} \rightarrow \mathit{thin} \\
P_4: & \mathrm{NP} \rightarrow \mathit{Eric}
\end{array}
$$

First-order multisorted signature:

$$
\begin{array}{ll}
\textsc{is}: & \mathrm{AP} \times \mathrm{NP} \rightarrow \mathrm{S} \\
\textsc{and}: & \mathrm{AP} \times \mathrm{AP} \rightarrow \mathrm{AP} \\
\textsc{tall}: & \mathrm{AP} \\
\textsc{thin}: & \mathrm{AP} \\
\textsc{eric}: & \mathrm{NP}
\end{array}
$$

Lexicalized case

Grammar:

$$
\begin{array}{lll}
P_0: & \mathrm{S} \rightarrow \mathrm{NP}\ \mathit{is}\ \mathrm{AP} \\
P_1: & \mathrm{AP} \rightarrow \mathrm{A}\ \mathit{and}\ \mathrm{A} \\
P_2: & \mathrm{A} \rightarrow \mathit{tall} \\
P_3: & \mathrm{A} \rightarrow \mathit{thin} \\
P_4: & \mathrm{NP} \rightarrow \mathit{Eric}
\end{array}
$$

First-order multisorted signature:

$$
\begin{array}{ll}
\textsc{is}: & \mathrm{AP} \times \mathrm{NP} \rightarrow \mathrm{S} \\
\textsc{and}: & \mathrm{AP} \times \mathrm{AP} \rightarrow \mathrm{AP} \\
\textsc{tall}: & \mathrm{AP} \\
\textsc{thin}: & \mathrm{AP} \\
\textsc{eric}: & \mathrm{NP}
\end{array}
$$

$$\textsc{is}\big(\textsc{and}\big(\textsc{tall}, \textsc{thin}\big), \textsc{eric}\big)$$

Syntactic strucutre as λ-terms

First-order multisorted signature:

$$\begin{aligned}
\textsc{is} &: \quad \mathrm{AP} \times \mathrm{NP} \to \mathrm{S} \\
\textsc{and} &: \quad \mathrm{AP} \times \mathrm{AP} \to \mathrm{AP} \\
\textsc{tall} &: \quad \mathrm{AP} \\
\textsc{thin} &: \quad \mathrm{AP} \\
\textsc{eric} &: \quad \mathrm{NP}
\end{aligned}$$

Syntactic strucutre as λ-terms

First-order multisorted signature:

$$\begin{aligned}
\textsc{is} &: \quad \mathrm{AP} \times \mathrm{NP} \to \mathrm{S} \\
\textsc{and} &: \quad \mathrm{AP} \times \mathrm{AP} \to \mathrm{AP} \\
\textsc{tall} &: \quad \mathrm{AP} \\
\textsc{thin} &: \quad \mathrm{AP} \\
\textsc{eric} &: \quad \mathrm{NP}
\end{aligned}$$

Higher-order signature:

$$\begin{aligned}
\textsc{is} &: \quad \mathrm{AP} \to (\mathrm{NP} \to \mathrm{S}) \\
\textsc{and} &: \quad \mathrm{AP} \to (\mathrm{AP} \to \mathrm{AP}) \\
\textsc{tall} &: \quad \mathrm{AP} \\
\textsc{thin} &: \quad \mathrm{AP} \\
\textsc{eric} &: \quad \mathrm{NP}
\end{aligned}$$

Semantic rules:

$$\begin{aligned}
[\![\textsc{is}]\!] &= \lambda xy.\, x\, y \\
[\![\textsc{and}]\!] &= \lambda xyz.\, (x\, z) \wedge (y\, z) \\
[\![\textsc{tall}]\!] &= \lambda x.\, \mathbf{tall}\, x \\
[\![\textsc{thin}]\!] &= \lambda x.\, \mathbf{thin}\, x \\
[\![\textsc{eric}]\!] &= \mathbf{e}
\end{aligned}$$

Syntactic strucutre as λ-terms

$[\![\text{IS}\ (\text{AND TALL THIN})\ \text{ERIC}]\!]$

Syntactic strucutre as λ-terms

$$[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] = [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!]$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\rightarrow_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\rightarrow_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\rightarrow_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, (\mathbf{tall}\, z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, (\mathbf{tall}\, z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge ([\![\textsc{thin}]\!]\ z))\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, (\mathbf{tall}\, z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge ([\![\textsc{thin}]\!]\ z))\ [\![\textsc{eric}]\!] \\
&= (\lambda z.\, (\mathbf{tall}\, z) \wedge ((\lambda x.\, \mathbf{thin}\, x)\ z))\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, (\mathbf{tall}\, z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge ([\![\textsc{thin}]\!]\ z))\ [\![\textsc{eric}]\!] \\
&= (\lambda z.\, (\mathbf{tall}\, z) \wedge ((\lambda x.\, \mathbf{thin}\, x)\ z))\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge (\mathbf{thin}\, z))\ [\![\textsc{eric}]\!]
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\text{IS}\ (\text{AND}\ \text{TALL}\ \text{THIN})\ \text{ERIC}]\!] &= [\![\text{IS}]\!]\ ([\![\text{AND}]\!]\ [\![\text{TALL}]\!]\ [\![\text{THIN}]\!])\ [\![\text{ERIC}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\text{AND}]\!]\ [\![\text{TALL}]\!]\ [\![\text{THIN}]\!])\ [\![\text{ERIC}]\!] \\
&\to_\beta (\lambda y.\, [\![\text{AND}]\!]\ [\![\text{TALL}]\!]\ [\![\text{THIN}]\!]\ y)\ [\![\text{ERIC}]\!] \\
&\to_\beta [\![\text{AND}]\!]\ [\![\text{TALL}]\!]\ [\![\text{THIN}]\!]\ [\![\text{ERIC}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\text{TALL}]\!]\ [\![\text{THIN}]\!]\ [\![\text{ERIC}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\text{TALL}]\!]\ z) \wedge (y\, z))\ [\![\text{THIN}]\!]\ [\![\text{ERIC}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\text{THIN}]\!]\ [\![\text{ERIC}]\!] \\
&\to_\beta (\lambda yz.\, (\mathbf{tall}\, z) \wedge (y\, z))\ [\![\text{THIN}]\!]\ [\![\text{ERIC}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge ([\![\text{THIN}]\!]\ z))\ [\![\text{ERIC}]\!] \\
&= (\lambda z.\, (\mathbf{tall}\, z) \wedge ((\lambda x.\, \mathbf{thin}\, x)\ z))\ [\![\text{ERIC}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge (\mathbf{thin}\, z))\ [\![\text{ERIC}]\!] \\
&\to_\beta (\mathbf{tall}\ [\![\text{ERIC}]\!]) \wedge (\mathbf{thin}\ [\![\text{ERIC}]\!])
\end{aligned}
$$

Syntactic strucutre as λ-terms

$$
\begin{aligned}
[\![\textsc{is}\ (\textsc{and}\ \textsc{tall}\ \textsc{thin})\ \textsc{eric}]\!] &= [\![\textsc{is}]\!]\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&= (\lambda xy.\, x\, y)\ ([\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!])\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda y.\, [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ y)\ [\![\textsc{eric}]\!] \\
&\to_\beta [\![\textsc{and}]\!]\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda xyz.\, (x\, z) \wedge (y\, z))\ [\![\textsc{tall}]\!]\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, ([\![\textsc{tall}]\!]\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&= (\lambda yz.\, ((\lambda x.\, \mathbf{tall}\, x)\ z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda yz.\, (\mathbf{tall}\, z) \wedge (y\, z))\ [\![\textsc{thin}]\!]\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge ([\![\textsc{thin}]\!]\ z))\ [\![\textsc{eric}]\!] \\
&= (\lambda z.\, (\mathbf{tall}\, z) \wedge ((\lambda x.\, \mathbf{thin}\, x)\ z))\ [\![\textsc{eric}]\!] \\
&\to_\beta (\lambda z.\, (\mathbf{tall}\, z) \wedge (\mathbf{thin}\, z))\ [\![\textsc{eric}]\!] \\
&\to_\beta (\mathbf{tall}\ [\![\textsc{eric}]\!]) \wedge (\mathbf{thin}\ [\![\textsc{eric}]\!]) \\
&= (\mathbf{tall}\, \mathbf{e}) \wedge (\mathbf{thin}\, \mathbf{e})
\end{aligned}
$$

Outline

A direct naive interpretation

Syntax/semantics interface:

A direct naive interpretation

Syntax/semantics interface:

$$\begin{aligned} \textsc{eric} &: \mathrm{NP} \\ \textsc{rebecca} &: \mathrm{NP} \\ \textsc{loves} &: \mathrm{NP} \to \mathrm{NP} \to \mathrm{S} \end{aligned}$$

$$\begin{aligned} [\![\mathrm{NP}]\!] &= \mathrm{e} \\ [\![\mathrm{S}]\!] &= \mathrm{t} \end{aligned}$$

A direct naive interpretation

Syntax/semantics interface:

$$\begin{aligned} \textsc{eric} &: \mathrm{NP} \\ \textsc{rebecca} &: \mathrm{NP} \\ \textsc{loves} &: \mathrm{NP} \to \mathrm{NP} \to \mathrm{S} \end{aligned}$$

$$\begin{aligned} [\![\mathrm{NP}]\!] &= \mathrm{e} \\ [\![\mathrm{S}]\!] &= \mathrm{t} \end{aligned}$$

Semantic interpretation:

A direct naive interpretation

Syntax/semantics interface:

$$\begin{aligned} \textsc{eric} &: \mathrm{NP} \\ \textsc{rebecca} &: \mathrm{NP} \\ \textsc{loves} &: \mathrm{NP} \to \mathrm{NP} \to \mathrm{S} \end{aligned}$$

$$\begin{aligned} [\![\mathrm{NP}]\!] &= \mathrm{e} \\ [\![\mathrm{S}]\!] &= \mathrm{t} \end{aligned}$$

Semantic interpretation:

$$\begin{aligned} [\![\textsc{eric}]\!] &= \mathbf{e} \\ [\![\textsc{rebecca}]\!] &= \mathbf{r} \\ [\![\textsc{loves}]\!] &= \lambda y.\, \lambda x.\, \mathbf{love}\, x\, y \end{aligned}$$

where:

$$\begin{aligned} \mathbf{e}, \mathbf{r} &: \mathrm{e} \\ \mathbf{love} &: \mathrm{e} \to \mathrm{e} \to \mathrm{t} \end{aligned}$$

A direct naive interpretation

- *Rebecca loves Eric*
- *Somebody loves Eric*
- *Everybody loves Eric*
- *Nobody loves Eric*

Type raising

Syntax/semantics interface:

Type raising

Syntax/semantics interface:

$$
\begin{array}{ll}
\textsc{eric} & : \mathrm{NP} \\
\textsc{rebecca} & : \mathrm{NP} \\
\textsc{everybody} & : \mathrm{NP} \\
\textsc{somebody} & : \mathrm{NP} \\
\textsc{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S}
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathrm{NP}]\!] & = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t} \\
[\![\mathrm{S}]\!] & = \mathrm{t}
\end{array}
$$

Type raising

Syntax/semantics interface:

$$
\begin{array}{ll}
\textsc{eric} & : \mathrm{NP} \\
\textsc{rebecca} & : \mathrm{NP} \\
\textsc{everybody} & : \mathrm{NP} \\
\textsc{somebody} & : \mathrm{NP} \\
\textsc{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S}
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathrm{NP}]\!] & = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t} \\
[\![\mathrm{S}]\!] & = \mathrm{t}
\end{array}
$$

Semantic interpretation:

Type raising

Syntax/semantics interface:

$$
\begin{array}{ll}
\textsc{eric} & : \mathrm{NP} \\
\textsc{rebecca} & : \mathrm{NP} \\
\textsc{everybody} & : \mathrm{NP} \\
\textsc{somebody} & : \mathrm{NP} \\
\textsc{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S}
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathrm{NP}]\!] & = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t} \\
[\![\mathrm{S}]\!] & = \mathrm{t}
\end{array}
$$

Semantic interpretation:

$$
\begin{array}{ll}
[\![\textsc{eric}]\!] & = \lambda k.\, k\, \mathbf{e} \\
[\![\textsc{rebecca}]\!] & = \lambda k.\, k\, \mathbf{r} \\
[\![\textsc{everybody}]\!] & = \lambda k.\, \forall x.\, (\mathbf{human}\, x) \to (k\, x) \\
[\![\textsc{somebody}]\!] & = \lambda k.\, \exists x.\, (\mathbf{human}\, x) \wedge (k\, x) \\
[\![\textsc{loves}]\!] & = \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{love}\, x\, y))
\end{array}
$$

Type raising

Exercice:

Give an appropriate semantic interpretation to NOBODY.

Outline

Syntax/semantics interface:

Syntax/semantics interface:

$$
\begin{array}{ll}
\textsc{eric} & : \mathrm{NP} \\
\textsc{rebecca} & : \mathrm{NP} \\
\textsc{everybody} & : \mathrm{NP} \\
\textsc{somebody} & : \mathrm{NP} \\
\textsc{man} & : \mathrm{N} \\
\textsc{woman} & : \mathrm{N} \\
\textsc{every} & : \mathrm{N} \to \mathrm{NP} \\
\textsc{a} & : \mathrm{N} \to \mathrm{NP} \\
\textsc{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S}
\end{array}
$$

$$
\begin{aligned}
[\![\mathrm{N}]\!] &= \mathrm{e} \to \mathrm{t} \\
[\![\mathrm{NP}]\!] &= (\mathrm{e} \to \mathrm{t}) \to \mathrm{t} \\
[\![\mathrm{S}]\!] &= \mathrm{t}
\end{aligned}
$$

Semantic interpretation:

Semantic interpretation:

$$
\begin{aligned}
[\![\text{ERIC}]\!] &= \lambda k.\, k\,\mathbf{e} \\
[\![\text{REBECCA}]\!] &= \lambda k.\, k\,\mathbf{r} \\
[\![\text{EVERYBODY}]\!] &= \lambda k.\, \forall x.\, (\mathbf{human}\, x) \to (k\, x) \\
[\![\text{SOMEBODY}]\!] &= \lambda k.\, \exists x.\, (\mathbf{human}\, x) \wedge (k\, x) \\
[\![\text{MAN}]\!] &= \lambda x.\, \mathbf{man}\, x \\
[\![\text{WOMAN}]\!] &= \lambda x.\, \mathbf{woman}\, x \\
[\![\text{EVERY}]\!] &= \lambda n.\, \lambda m.\, \forall x.\, n\, x \to m\, x \\
[\![\text{A}]\!] &= \lambda n.\, \lambda m.\, \exists x.\, n\, x \wedge m\, x \\
[\![\text{LOVES}]\!] &= \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{love}\, x\, y))
\end{aligned}
$$

where:

$$\mathbf{woman}, \mathbf{man} : \mathrm{e} \to \mathrm{t}$$

Outline

Syntax/semantics interface:

Syntax/semantics interface:

$$
\begin{array}{ll}
\textsc{eric} & : \mathrm{NP} \\
\textsc{rebecca} & : \mathrm{NP} \\
\textsc{everybody} & : \mathrm{NP} \\
\textsc{somebody} & : \mathrm{NP} \\
\textsc{man} & : \mathrm{N} \\
\textsc{woman} & : \mathrm{N} \\
\textsc{every} & : \mathrm{N} \to \mathrm{NP} \\
\textsc{a} & : \mathrm{N} \to \mathrm{NP} \\
\textsc{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S} \\
\textsc{who} & : (\mathrm{NP} \to \mathrm{S}) \to \mathrm{N} \to \mathrm{N}
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathrm{N}]\!] & = \mathrm{e} \to \mathrm{t} \\
[\![\mathrm{NP}]\!] & = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t} \\
[\![\mathrm{S}]\!] & = \mathrm{t}
\end{array}
$$

Semantic interpretation:

Semantic interpretation:

$$
\begin{aligned}
[\![\text{ERIC}]\!] &= \lambda k.\, k\,\mathbf{e} \\
[\![\text{REBECCA}]\!] &= \lambda k.\, k\,\mathbf{r} \\
[\![\text{EVERYBODY}]\!] &= \lambda k.\, \forall x.\, k\,x \\
[\![\text{SOMEBODY}]\!] &= \lambda k.\, \exists x.\, k\,x \\
[\![\text{MAN}]\!] &= \lambda x.\, \mathbf{man}\,x \\
[\![\text{WOMAN}]\!] &= \lambda x.\, \mathbf{woman}\,x \\
[\![\text{EVERY}]\!] &= \lambda n.\, \lambda m.\, \forall x.\, n\,x \to m\,x \\
[\![\text{A}]\!] &= \lambda n.\, \lambda m.\, \exists x.\, n\,x \wedge m\,x \\
[\![\text{LOVES}]\!] &= \lambda o.\, \lambda s.\, s\,(\lambda x.\, o\,(\lambda y.\, \mathbf{love}\,x\,y)) \\
[\![\text{WHO}]\!] &= \lambda r.\, \lambda n.\, \lambda x.\, n\,x \wedge r\,(\lambda k.\, k\,x)
\end{aligned}
$$

Outline

Syntax/semantics interface:

Syntax/semantics interface:

$$
\begin{array}{ll}
\textsc{eric} & : \mathrm{NP} \\
\textsc{rebecca} & : \mathrm{NP} \\
\textsc{everybody} & : \mathrm{NP} \\
\textsc{somebody} & : \mathrm{NP} \\
\textsc{man} & : \mathrm{N} \\
\textsc{woman} & : \mathrm{N} \\
\textsc{every} & : \mathrm{N} \to \mathrm{NP} \\
\textsc{a} & : \mathrm{N} \to \mathrm{NP} \\
\textsc{french} & : \mathrm{N} \to \mathrm{N} \\
\textsc{loves} & : \mathrm{NP} \to \mathrm{NP} \to \mathrm{S} \\
\textsc{who} & : (\mathrm{NP} \to \mathrm{S}) \to \mathrm{N} \to \mathrm{N}
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathrm{N}]\!] & = \mathrm{e} \to \mathrm{t} \\
[\![\mathrm{NP}]\!] & = (\mathrm{e} \to \mathrm{t}) \to \mathrm{t} \\
[\![\mathrm{S}]\!] & = \mathrm{t}
\end{array}
$$

Semantic interpretation:

Semantic interpretation:

$$
\begin{aligned}
[\![\text{ERIC}]\!] &= \lambda k.\, k\,\mathbf{e} \\
[\![\text{REBECCA}]\!] &= \lambda k.\, k\,\mathbf{r} \\
[\![\text{EVERYBODY}]\!] &= \lambda k.\, \forall x.\, k\,x \\
[\![\text{SOMEBODY}]\!] &= \lambda k.\, \exists x.\, k\,x \\
[\![\text{MAN}]\!] &= \lambda x.\, \mathbf{man}\,x \\
[\![\text{WOMAN}]\!] &= \lambda x.\, \mathbf{woman}\,x \\
[\![\text{EVERY}]\!] &= \lambda n.\, \lambda m.\, \forall x.\, n\,x \rightarrow m\,x \\
[\![\text{A}]\!] &= \lambda n.\, \lambda m.\, \exists x.\, n\,x \wedge m\,x \\
[\![\text{FRENCH}]\!] &= \lambda n.\, \lambda x.\, n\,x \wedge \mathbf{french}\,x \\
[\![\text{LOVES}]\!] &= \lambda o.\, \lambda s.\, s\,(\lambda x.\, o\,(\lambda y.\, \mathbf{love}\,x\,y)) \\
[\![\text{WHO}]\!] &= \lambda r.\, \lambda n.\, \lambda x.\, n\,x \wedge r\,(\lambda k.\, k\,x)
\end{aligned}
$$

where:

$$\mathbf{french} : \mathrm{e} \rightarrow \mathrm{t}$$

Adjective classification:

Adjective classification:

- **Intersective**:

Adjective classification:

- **Intersective**:

  French, sick, carnivorous, red, ...

Adjective classification:

- **Intersective**:

  French, sick, carnivorous, red, ...

- **Subsective** but non intersective:

Adjective classification:

- **Intersective**:

  French, *sick*, *carnivorous*, *red*, ...

- **Subsective** but non intersective:

  typical, *recent*, *skillful*, ...

Adjective classification:

- **Intersective**:

  French, *sick*, *carnivorous*, *red*, ...

- **Subsective** but non intersective:

  typical, *recent*, *skillful*, ...

- **Privative**:

Adjective classification:

- **Intersective**:

  French, *sick*, *carnivorous*, *red*, ...

- **Subsective** but non intersective:

  typical, *recent*, *skillful*, ...

- **Privative**:

  fake, *former*, *spurious*, ...

Adjective classification:

- **Intersective**:

  French, *sick*, *carnivorous*, *red*, ...
- **Subsective** but non intersective:

  typical, *recent*, *skillful*, ...
- **Privative**:

  fake, *former*, *spurious*, ...
- **Plain nonsubsective**:

Adjective classification:

- **Intersective**:

  French, *sick*, *carnivorous*, *red*, ...
- **Subsective** but non intersective:

  typical, *recent*, *skillful*, ...
- **Privative**:

  fake, *former*, *spurious*, ...
- **Plain nonsubsective**:

  alleged, *arguable*, *putative*, ...

Adjective classification:

- **Intersective**:

  French, *sick*, *carnivorous*, *red*, ...
- **Subsective** but non intersective:

  typical, *recent*, *skillful*, ...
- **Privative**:

  fake, *former*, *spurious*, ...
- **Plain nonsubsective**:

  alleged, *arguable*, *putative*, ...

Meaning postulates:

Meaning postulates:

$$\mathrm{INT}(A) = \exists P.\, \forall Q\, x.\, A\, Q\, x \equiv (P\, x \wedge Q\, x)$$

Meaning postulates:

$$\mathrm{INT}(A) = \exists P.\, \forall Q\, x.\, A\, Q\, x \equiv (P\, x \wedge Q\, x)$$

$$\mathrm{SUB}(A) = \forall Q\, x.\, A\, Q\, x \rightarrow Q\, x$$

Meaning postulates:

$$\mathrm{INT}(A) = \exists P.\, \forall Q\, x.\, A\, Q\, x \equiv (P\, x \wedge Q\, x)$$

$$\mathrm{SUB}(A) = \forall Q\, x.\, A\, Q\, x \rightarrow Q\, x$$

$$\mathrm{PRIV}(A) = \forall Q\, x.\, A\, Q\, x \rightarrow \neg(Q\, x)$$

Meaning postulates:

$$\mathrm{INT}(A) = \exists P.\, \forall Q\, x.\, A\, Q\, x \equiv (P\, x \wedge Q\, x)$$

$$\mathrm{SUB}(A) = \forall Q\, x.\, A\, Q\, x \rightarrow Q\, x$$

$$\mathrm{PRIV}(A) = \forall Q\, x.\, A\, Q\, x \rightarrow \neg(Q\, x)$$

Beware!!!! Some intensionality involved!

Outline

Scope ambiguities

Scope ambiguities

Every man loves a woman

Scope ambiguities

Every man loves a woman

$$\forall x.\mathbf{man}\, x \rightarrow (\exists y.\mathbf{woman}\, y \wedge \mathbf{love}\, x\, y)$$
$$\exists y.\mathbf{woman}\, y \wedge (\forall x.\mathbf{man}\, x \wedge \mathbf{love}\, x\, y)$$

Scope ambiguities

Every man loves a woman

$$\forall x.\mathbf{man}\, x \rightarrow (\exists y.\mathbf{woman}\, y \wedge \mathbf{love}\, x\, y)$$
$$\exists y.\mathbf{woman}\, y \wedge (\forall x.\mathbf{man}\, x \wedge \mathbf{love}\, x\, y)$$

Subject wide scope:

$$[\![\text{LOVES}]\!] = \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{love}\, x\, y))$$

Scope ambiguities

Every man loves a woman

$$\forall x.\mathbf{man}\, x \to (\exists y.\mathbf{woman}\, y \wedge \mathbf{love}\, x\, y)$$
$$\exists y.\mathbf{woman}\, y \wedge (\forall x.\mathbf{man}\, x \wedge \mathbf{love}\, x\, y)$$

Subject wide scope:

$$[\![\text{LOVES}]\!] = \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{love}\, x\, y))$$

Object wide scope:

$$[\![\text{LOVES}_{\text{ows}}]\!] = \lambda o.\, \lambda s.\, o\, (\lambda y.\, s\, (\lambda x.\, \mathbf{love}\, x\, y))$$

Scope ambiguities

Another solution:

Scope ambiguities

Another solution:

$$\text{QR} \ : \ \text{NP} \to (\text{NP} \to \text{S}) \to \text{S}$$

Scope ambiguities

Another solution:

$$\mathrm{QR} \;:\; \mathrm{NP} \to (\mathrm{NP} \to \mathrm{S}) \to \mathrm{S}$$

$$[\![\mathrm{QR}]\!] \;=\; \lambda n.\, \lambda p.\, n\,(\lambda x.\, p\,(\lambda k.\, k\, x))$$

Scope ambiguities

Another solution:

$$\text{QR} \;:\; \text{NP} \to (\text{NP} \to \text{S}) \to \text{S}$$

$$[\![\text{QR}]\!] \;=\; \lambda n.\, \lambda p.\, n\,(\lambda x.\, p\,(\lambda k.\, k\, x))$$

Wide scope reading:

$$\text{QR}\,(\text{A WOMAN})\,(\lambda o.\, \text{LOVE}\, o\,(\text{EVERY MAN}))$$

Outline

De re and de dicto readings as scope ambiguities

De re and de dicto readings as scope ambiguities

John seeks a unicorn

De re and de dicto readings as scope ambiguities

John seeks a unicorn

$$\exists x.\mathbf{unicorn}\, x \wedge \mathbf{try}\, \mathbf{j}\, (\lambda z.\, \mathbf{find}\, z\, x)$$
$$\mathbf{try}\, \mathbf{j}\, (\lambda z.\, \exists x.\mathbf{unicorn}\, x \wedge \mathbf{find}\, z\, x)$$

De re and de dicto readings as scope ambiguities

John seeks a unicorn

$$\exists x.\mathbf{unicorn}\, x \wedge \mathbf{try}\, \mathbf{j}\, (\lambda z.\, \mathbf{find}\, z\, x)$$
$$\mathbf{try}\, \mathbf{j}\, (\lambda z.\, \exists x.\mathbf{unicorn}\, x \wedge \mathbf{find}\, z\, x)$$

De re reading:

$$[\![\mathrm{SEEK_{re}}]\!] \;=\; \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{try}\, x\, (\lambda z.\, \mathbf{find}\, z\, y)))$$

De re and de dicto readings as scope ambiguities

John seeks a unicorn

$$\exists x.\mathbf{unicorn}\, x \wedge \mathbf{try}\, \mathbf{j}\, (\lambda z.\, \mathbf{find}\, z\, x)$$
$$\mathbf{try}\, \mathbf{j}\, (\lambda z.\, \exists x.\mathbf{unicorn}\, x \wedge \mathbf{find}\, z\, x)$$

De re reading:

$$[\![\mathrm{SEEK_{re}}]\!] \;=\; \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{try}\, x\, (\lambda z.\, \mathbf{find}\, z\, y)))$$

De dicto reading:

De re and de dicto readings as scope ambiguities

John seeks a unicorn

$$\exists x.\mathbf{unicorn}\, x \wedge \mathbf{try}\, \mathbf{j}\, (\lambda z.\, \mathbf{find}\, z\, x)$$
$$\mathbf{try}\, \mathbf{j}\, (\lambda z.\, \exists x.\mathbf{unicorn}\, x \wedge \mathbf{find}\, z\, x)$$

De re reading:

$$[\![\mathrm{SEEK}_{\mathrm{re}}]\!] \;=\; \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{try}\, x\, (\lambda z.\, \mathbf{find}\, z\, y)))$$

De dicto reading:

$$[\![\mathrm{SEEK}_{\mathrm{dicto}}]\!] \;=\; \lambda o.\, \lambda s.\, s\, (\lambda x.\, \mathbf{try}\, x\, (\lambda z.\, o\, (\lambda y.\, \mathbf{find}\, z\, y)))$$

De re and de dicto readings as scope ambiguities

John seeks a unicorn

$$\exists x.\mathbf{unicorn}\, x \wedge \mathbf{try}\, \mathbf{j}\, (\lambda z.\, \mathbf{find}\, z\, x)$$
$$\mathbf{try}\, \mathbf{j}\, (\lambda z.\, \exists x.\mathbf{unicorn}\, x \wedge \mathbf{find}\, z\, x)$$

De re reading:

$$[\![\mathrm{SEEK}_{\mathrm{re}}]\!] \;=\; \lambda o.\, \lambda s.\, s\, (\lambda x.\, o\, (\lambda y.\, \mathbf{try}\, x\, (\lambda z.\, \mathbf{find}\, z\, y)))$$

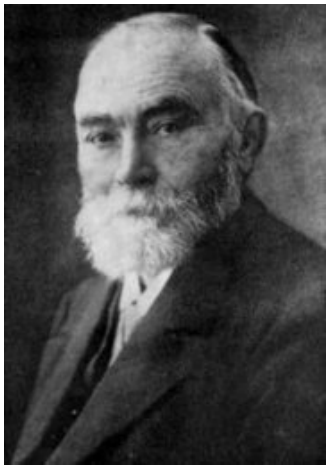
De dicto reading:

$$[\![\mathrm{SEEK}_{\mathrm{dicto}}]\!] \;=\; \lambda o.\, \lambda s.\, s\, (\lambda x.\, \mathbf{try}\, x\, (\lambda z.\, o\, (\lambda y.\, \mathbf{find}\, z\, y)))$$

Beware!!!! Some intensionality involved!

Outline

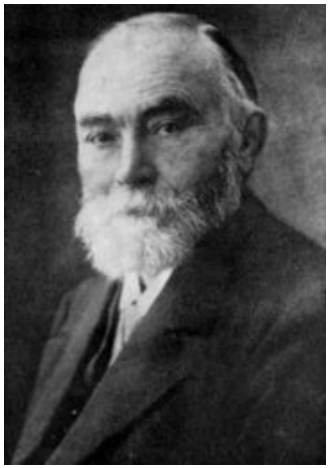
Sinn und bedeutung



F.L.G. Frege

(1848–1925)

Sinn und bedeutung



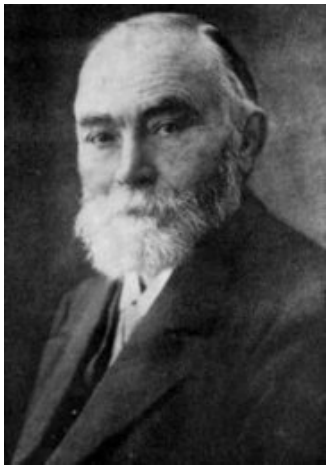
F.L.G. Frege

(1848–1925)

Sinn (sense)/Bedeutung (reference) — Frege
Intension/Extension — Carnap

According to Frege, the sense of an expression is its "mode of presentation", while the reference or denotation of an expression is the object it refers to.

Sinn und bedeutung



F.L.G. Frege

(1848–1925)

Sinn (sense)/Bedeutung (reference) — Frege
Intension/Extension — Carnap

According to Frege, the sense of an expression is its "mode of presentation", while the reference or denotation of an expression is the object it refers to.

For instance, both expressions "$1 + 1$" and "2" have the same denotation but not the same sense.

Intensional proposition

An intensional proposition is a proposition whose validity is not invariant under extensional substitution.

Intensional proposition

An intensional proposition is a proposition whose validity is not invariant under extensional substitution.

Frege gives the example of "the morning star" and "the evening star" which both refer to the planet Venus.

Intensional proposition

An intensional proposition is a proposition whose validity is not invariant under extensional substitution.

Frege gives the example of "the morning star" and "the evening star" which both refer to the planet Venus.

Compare "the morning star is the evening star" with "John does not know that the morning star is the evening star".

Intension and extension

Intension and extension

This red car is a Ferrari

Intension and extension

This red car is a Ferrari

This skillful surgeon is Dr Johnson

Intension and extension

This red car is a Ferrari

This skillful surgeon is Dr Johnson

$$(\forall x.\,(\textbf{surgeon}\,x) \leftrightarrow (\textbf{driver}\,x)) \rightarrow (\forall x.\,((\textbf{skillful surgeon})\,x) \rightarrow ((\textbf{skillful driver})\,x))$$

Intension and extension

This red car is a Ferrari

This skillful surgeon is Dr Johnson

$$(\forall x.\,(\textbf{surgeon}\,x) \leftrightarrow (\textbf{driver}\,x)) \\ \rightarrow (\forall x.\,((\textbf{skillful surgeon})\,x) \rightarrow ((\textbf{skillful driver})\,x))$$

Solution:

Intension and extension

This red car is a Ferrari

This skillful surgeon is Dr Johnson

$$(\forall x.\,(\textbf{surgeon}\,x) \leftrightarrow (\textbf{driver}\,x)) \rightarrow (\forall x.\,((\textbf{skillful}\,\textbf{surgeon})\,x) \rightarrow ((\textbf{skillful}\,\textbf{driver})\,x))$$

Solution:

$$\begin{aligned} \textbf{surgeon} &: \mathbf{e} \rightarrow (\mathbf{s} \rightarrow \mathbf{t}) \\ \textbf{driver} &: \mathbf{e} \rightarrow (\mathbf{s} \rightarrow \mathbf{t}) \end{aligned}$$

Intension and extension

This red car is a Ferrari

This skillful surgeon is Dr Johnson

$$(\forall x.\,(\textbf{surgeon}\,x) \leftrightarrow (\textbf{driver}\,x)) \rightarrow (\forall x.\,((\textbf{skillful surgeon})\,x) \rightarrow ((\textbf{skillful driver})\,x))$$

Solution:

$$\begin{aligned}\textbf{surgeon} &: \mathbf{e} \rightarrow (\mathbf{s} \rightarrow \mathbf{t})\\ \textbf{driver} &: \mathbf{e} \rightarrow (\mathbf{s} \rightarrow \mathbf{t})\\ \textbf{skillful} &: (\mathbf{e} \rightarrow (\mathbf{s} \rightarrow \mathbf{t})) \rightarrow \mathbf{e} \rightarrow (\mathbf{s} \rightarrow \mathbf{t})\end{aligned}$$