Introduction to Abstract Categorial grammars

Second course

Definition

Abstract Categorial Grammar

Abstract Categorial Grammar

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$$

where:

Abstract Categorial Grammar

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$$

where:

$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ (the abstract vocabulary)

Abstract Categorial Grammar

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$$

where:

$$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle \quad \text{(the abstract vocabulary)}$$

$$\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle \quad \text{(the object vocabulary)}$$

Abstract Categorial Grammar

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$$

where:

$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ (the abstract vocabulary)

$\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ (the object vocabulary)

$\mathscr{L} : \Sigma_1 \rightarrow \Sigma_2$ (the lexicon)

Abstract Categorial Grammar

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$$

where:

$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ (the abstract vocabulary)

$\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ (the object vocabulary)

$\mathscr{L} : \Sigma_1 \rightarrow \Sigma_2$ (the lexicon)

$s \in \mathscr{T}_{A_1}$ (the starting symbol)

Intuition

Intuition

- The abstract vocabulary specifies the abstract syntactic structure (Curry's tectogrammatics)

Intuition

- The abstract vocabulary specifies the abstract syntactic structure (Curry's tectogrammatics)
- The object vocabulary specifies the possible surface forms

Intuition

- The abstract vocabulary specifies the abstract syntactic structure (Curry's tectogrammatics)
- The object vocabulary specifies the possible surface forms
- The lexicon specifies how the abstract structures can be realized into surface forms (Curry's phenogrammatics)

Languages generated by an ACG

Languages generated by an ACG

Abstract language:

Languages generated by an ACG

Abstract language:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_1} \mid \vdash_{\Sigma_1} t : s \quad \text{is derivable}\}$$

Languages generated by an ACG

Abstract language:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_1} \mid \vdash_{\Sigma_1} t : s \quad \text{is derivable}\}$$

Object language:

Languages generated by an ACG

Abstract language:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_1} \mid \vdash_{\Sigma_1} t : s \quad \text{is derivable}\}$$

Object language:

$$\mathcal{O}(\mathcal{G}) = \{t \in \Lambda_{\Sigma_2} \mid \exists u \in \mathcal{A}(\mathcal{G}).\ t = \mathscr{L}(u)\}$$

A linguistic example

Abstract signature Σ_0

Abstract signature Σ_0

$$
\begin{array}{rcl}
N, NP, S & : & \text{type} \\
\text{J} & : & NP \\
\text{U} & : & N \\
\text{A} & : & N \to ((NP \to S) \to S) \\
\text{S} & : & ((NP \to S) \to S) \to (NP \to S)
\end{array}
$$

Abstract signature Σ_0

$$
\begin{array}{rcl}
N, NP, S & : & \text{type} \\
\text{J} & : & NP \\
\text{U} & : & N \\
\text{A} & : & N \to ((NP \to S) \to S) \\
\text{S} & : & ((NP \to S) \to S) \to (NP \to S)
\end{array}
$$

Object signature Σ_1

Abstract signature Σ_0

$$
\begin{array}{rcl}
N, NP, S & : & \text{type} \\
\text{J} & : & NP \\
\text{U} & : & N \\
\text{A} & : & N \to ((NP \to S) \to S) \\
\text{S} & : & ((NP \to S) \to S) \to (NP \to S)
\end{array}
$$

Object signature Σ_1

$$
/\text{a}/, /\text{John}/, /\text{seeks}/, /\text{unicorn}/ \quad : \quad \mathit{string}
$$

Abstract signature

$$
\begin{array}{rcl}
N, NP, S & : & \text{type} \\
\text{J} & : & NP \\
\text{U} & : & N \\
\text{A} & : & N \to ((NP \to S) \to S) \\
\text{S} & : & ((NP \to S) \to S) \to (NP \to S)
\end{array}
$$

Object signature

$$
/\text{a}/, /\text{John}/, /\text{seeks}/, /\text{unicorn}/ \quad : \quad \mathit{string}
$$

Abstract signature

$$
\begin{array}{rcl}
N, NP, S & : & \text{type} \\
\text{J} & : & NP \\
\text{U} & : & N \\
\text{A} & : & N \to ((NP \to S) \to S) \\
\text{S} & : & ((NP \to S) \to S) \to (NP \to S)
\end{array}
$$

Lexicon $\mathscr{L}_{\text{synt}} : \Sigma_0 \to \Sigma_1$

Object signature

/a/, /John/, /seeks/, /unicorn/ : *string*

Abstract signature

$$
\begin{array}{rcl}
N, NP, S & : & \text{type} \\
\text{J} & : & NP \\
\text{U} & : & N \\
\text{A} & : & N \to ((NP \to S) \to S) \\
\text{S} & : & ((NP \to S) \to S) \to (NP \to S)
\end{array}
$$

Lexicon $\mathscr{L}_{\text{synt}} : \Sigma_0 \to \Sigma_1$

$$
\begin{array}{rcl}
N, NP, S & := & \mathit{string} \\
\text{J} & := & /\text{John}/ \\
\text{U} & := & /\text{unicorn}/ \\
\text{A} & := & \lambda x.\, \lambda p.\, p\,(/\text{a}/ + x) \\
\text{S} & := & \lambda p.\, \lambda x.\, p\,(\lambda y.\, x + /\text{seeks}/ + y)
\end{array}
$$

Object signature

$$
/\text{a}/, /\text{John}/, /\text{seeks}/, /\text{unicorn}/ \quad : \quad \mathit{string}
$$

Another object signature Σ_2

Another object signature Σ_2

$$
\begin{array}{rcl}
\iota, o & : & \text{type} \\
\wedge & : & o \to (o \to o) \\
\exists & : & (\iota \Rightarrow o) \to o \\
\mathbf{j} & : & \iota \\
\mathbf{unicorn} & : & \iota \to o \\
\mathbf{find} & : & \iota \to (\iota \to o) \\
\mathbf{try} & : & \iota \to ((\iota \to o) \to o)
\end{array}
$$

Another object signature Σ_2

$$
\begin{array}{rcl}
\iota, o & : & \text{type} \\
\wedge & : & o \to (o \to o) \\
\exists & : & (\iota \Rightarrow o) \to o \\
\mathbf{j} & : & \iota \\
\mathbf{unicorn} & : & \iota \to o \\
\mathbf{find} & : & \iota \to (\iota \to o) \\
\mathbf{try} & : & \iota \to ((\iota \to o) \to o)
\end{array}
$$

We write: $\exists x.\mathbf{unicorn}\, x \wedge \mathbf{find}\, \mathbf{j}\, x$

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

Lexicon $\mathscr{L}_{\text{sem}} : \Sigma_0 \to \Sigma_2$

Lexicon $\mathscr{L}_{\text{sem}} : \Sigma_0 \to \Sigma_2$

$$
\begin{array}{rcl}
N & := & \iota \Rightarrow o \\
NP & := & \iota \\
S & := & o \\
\mathrm{J} & := & \mathbf{j} \\
\mathrm{U} & := & \lambda x.\, \mathbf{unicorn}\, x \\
\mathrm{A} & := & \lambda p.\, \lambda q.\, \exists x.\, p\, x \wedge q\, x \\
\mathrm{S} & := & \lambda p.\, \lambda x.\, \mathbf{try}\, x\, (\lambda y.\, p\, (\lambda z.\, \mathbf{find}\, y\, z))
\end{array}
$$

Parsing $/\text{John}/ + /\text{seeks}/ + /\text{a}/ + /\text{unicorn}/$ **yields two abstract terms** t_1 **and** t_2 **such that:**

$$\mathscr{L}_{\text{synt}}(t_1) = /\text{John}/ + /\text{seeks}/ + /\text{a}/ + /\text{unicorn}/$$

$$\mathscr{L}_{\text{synt}}(t_2) = /\text{John}/ + /\text{seeks}/ + /\text{a}/ + /\text{unicorn}/$$

Parsing $/\text{John}/ + /\text{seeks}/ + /\text{a}/ + /\text{unicorn}/$ yields two abstract terms t_1 and t_2 such that:

$$\mathscr{L}_{\text{synt}}(t_1) = /\text{John}/ + /\text{seeks}/ + /\text{a}/ + /\text{unicorn}/$$

$$\mathscr{L}_{\text{synt}}(t_2) = /\text{John}/ + /\text{seeks}/ + /\text{a}/ + /\text{unicorn}/$$

with

$$t_1 = \text{S}\,(\text{A}\,\text{U})\,\text{J}$$

$$t_2 = \text{A}\,\text{U}\,(\lambda x.\,\text{S}\,(\lambda k.\,k\,x)\,\text{J})$$

Then, applying the second lexicon, $\mathscr{L}_{\text{sem}}$, to t_1 and t_2 yields the following terms:

Then, applying the second lexicon, $\mathscr{L}_{\text{sem}}$, to t_1 and t_2 yields the following terms:

$$\mathscr{L}_{\text{sem}}(t_1) = \mathbf{try}\,\mathbf{j}\,(\lambda x.\,\exists y.\,\mathbf{unicorn}\,y \wedge \mathbf{find}\,x\,y)$$

Then, applying the second lexicon, $\mathscr{L}_{\text{sem}}$, to t_1 and t_2 yields the following terms:

$$\mathscr{L}_{\text{sem}}(t_1) = \mathbf{try}\,\mathbf{j}\,(\lambda x.\,\exists y.\,\mathbf{unicorn}\,y \wedge \mathbf{find}\,x\,y)$$

$$\mathscr{L}_{\text{sem}}(t_2) = \exists y.\,\mathbf{unicorn}\,y \wedge \mathbf{try}\,\mathbf{j}\,(\lambda x.\,\mathbf{find}\,x\,y)$$

ABSTRACT
SYNTACTIC
STRUCTURE

Σ_0

ABSTRACT
SYNTACTIC
STRUCTURE

Σ_0

Σ_1

SYNTACTIC
FORM

ABSTRACT
SYNTACTIC
STRUCTURE

Σ_0

Σ_1

SYNTACTIC
FORM

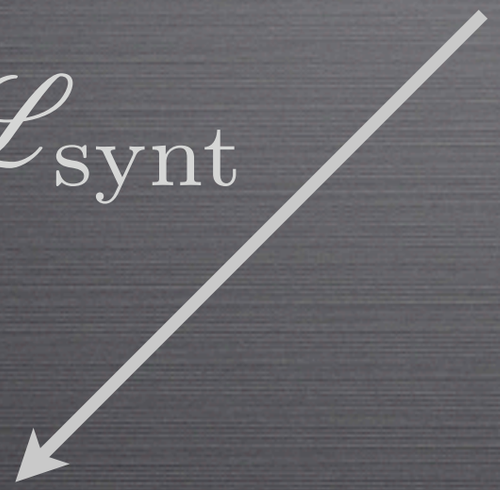
Σ_2

SEMANTIC
FORM

ABSTRACT
SYNTACTIC
STRUCTURE

Σ_0

$\mathscr{L}_{\text{synt}}$



Σ_1

SYNTACTIC
FORM

Σ_2

SEMANTIC
FORM

ABSTRACT
SYNTACTIC
STRUCTURE

Σ_0

$\mathscr{L}_{\text{synt}}$

$\mathscr{L}_{\text{sem}}$

Σ_1

SYNTACTIC
FORM

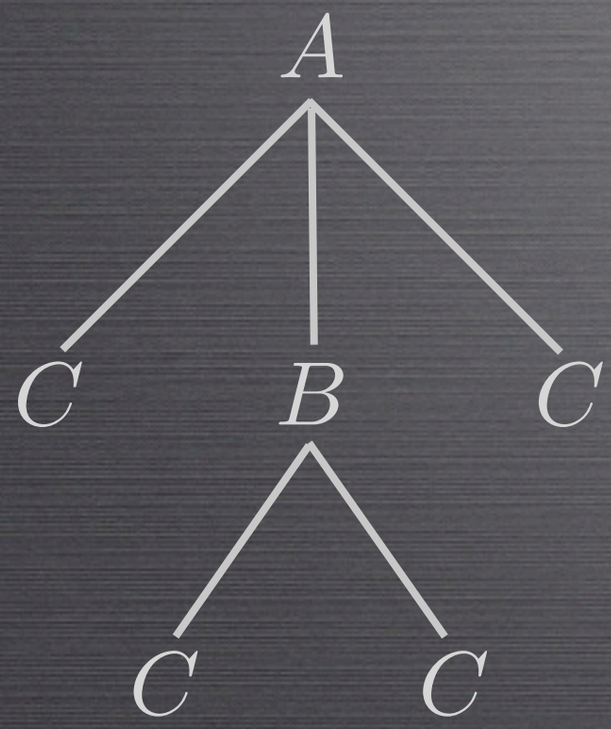
Σ_2

SEMANTIC
FORM

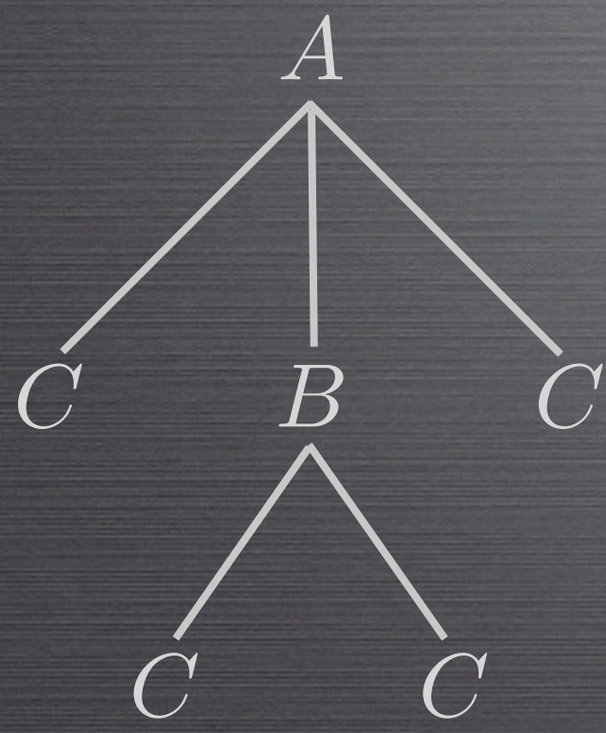
Language-theoretic examples

Strings and trees as linear lambda-terms

Strings and trees as linear lambda-terms

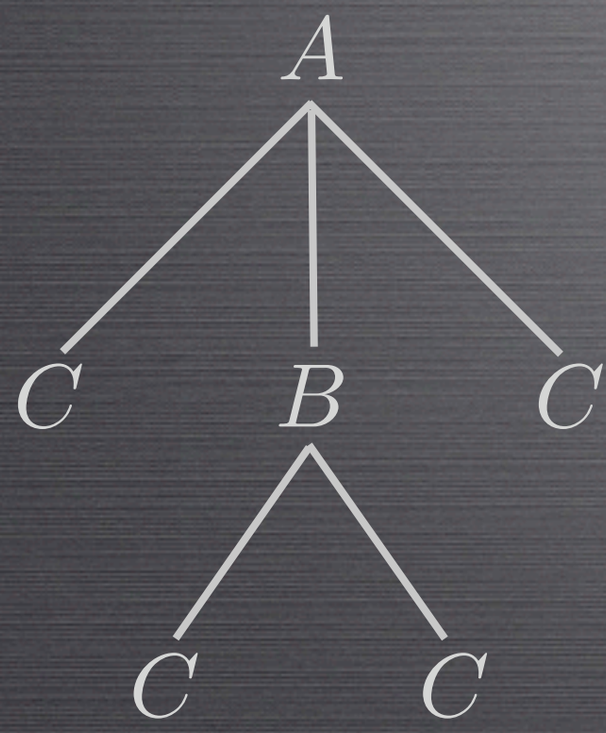


Strings and trees as linear lambda-terms



$$
\begin{array}{lcl}
\tau & : & \text{type} \\
A & : & \tau \to (\tau \to (\tau \to \tau)) \\
B & : & \tau \to (\tau \to \tau) \\
C & : & \tau
\end{array}
$$

Strings and trees as linear lambda-terms



$$\begin{array}{lcl} \tau & : & \text{type} \\ A & : & \tau \to (\tau \to (\tau \to \tau)) \\ B & : & \tau \to (\tau \to \tau) \\ C & : & \tau \end{array}$$

$$A\,C\,(B\,C\,C)\,C$$

Strings and trees as linear lambda-terms

Strings as monadic trees

Strings and trees as linear lambda-terms

Strings as monadic trees

$$\text{`}abbaac\text{'}$$

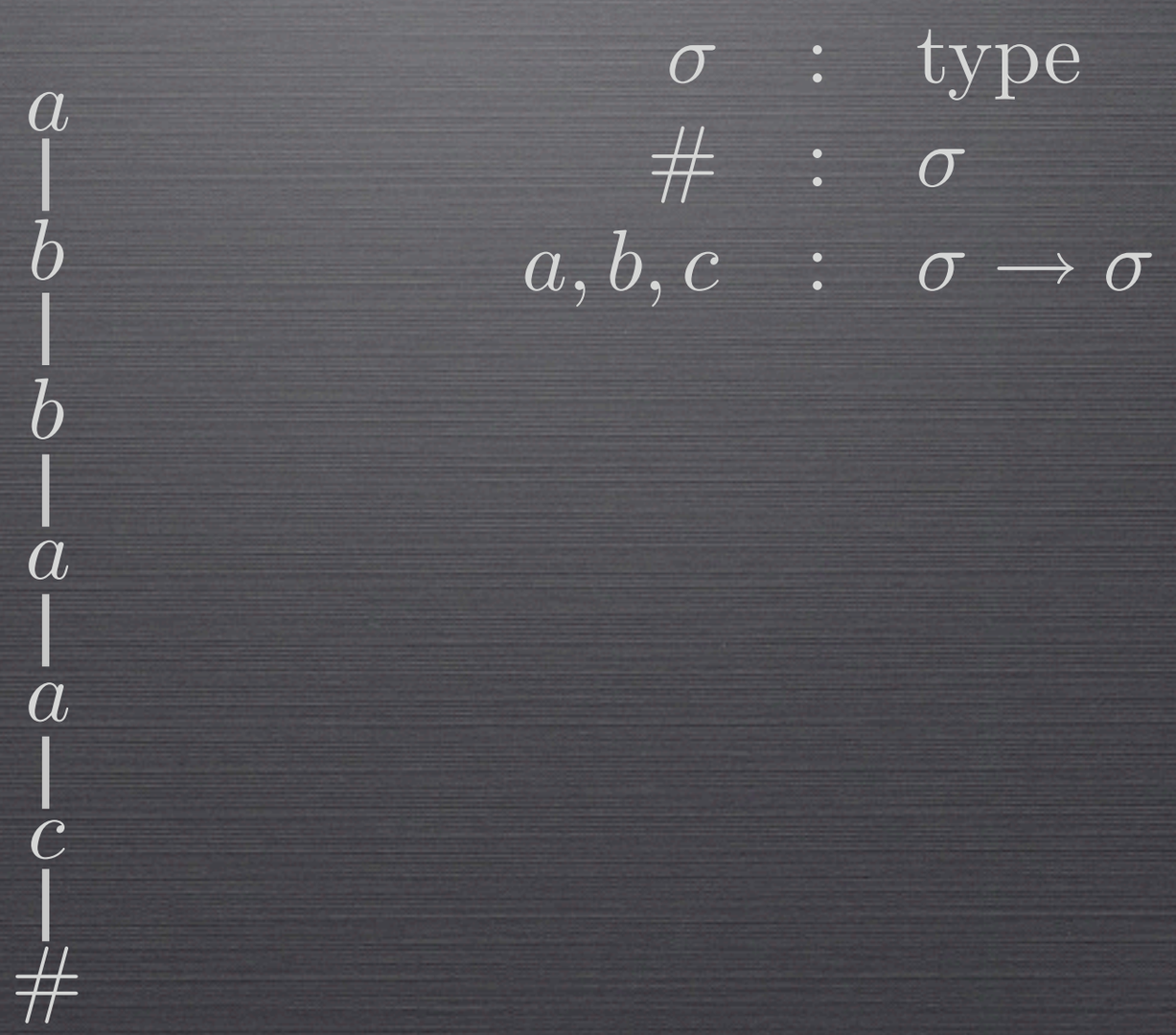
Strings and trees as linear lambda-terms

Strings as monadic trees

‘$abbaac$’

Strings and trees as linear lambda-terms

Strings as monadic trees



Strings and trees as linear lambda-terms

Strings as monadic trees

‘$abbaac$’

$$
\begin{array}{c}
a \\ | \\ b \\ | \\ b \\ | \\ a \\ | \\ a \\ | \\ c \\ | \\ \#
\end{array}
$$

$$
\begin{array}{rcl}
\sigma & : & \text{type} \\
\# & : & \sigma \\
a, b, c & : & \sigma \to \sigma
\end{array}
$$

$$a\,(b\,(b\,(a\,(a\,(c\,\#)))))$$

Strings and trees as linear lambda-terms

Strings as functional lambda-terms

$$\begin{array}{rcl} \sigma & : & \text{type} \\ a, b, c & : & \sigma \to \sigma \end{array}$$

Strings and trees as linear lambda-terms

Strings as functional lambda-terms

$$\begin{array}{rcl} \sigma & : & \text{type} \\ a, b, c & : & \sigma \to \sigma \end{array}$$

$$\text{`}abbaac\text{'}$$

as

$$\lambda x.\, a\,(b\,(b\,(a\,(a\,(c\,x)))))$$

Strings and trees as linear lambda-terms

Strings as functional lambda-terms

$$\begin{array}{rcl} \sigma & : & \text{type} \\ a, b, c & : & \sigma \to \sigma \end{array}$$

‘$abbaac$’

as

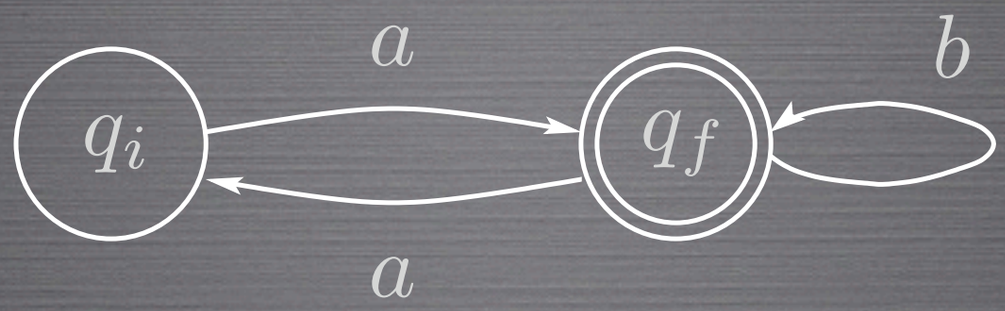
$$\lambda x.\, a\,(b\,(b\,(a\,(a\,(c\,x)))))$$

In this setting:

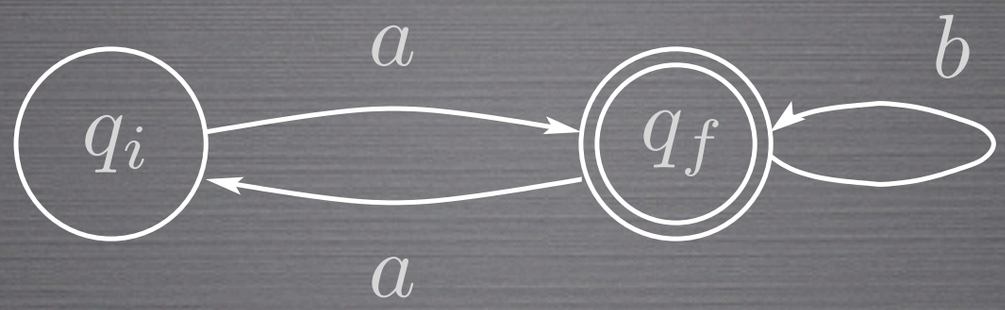
$$\begin{array}{rcl} \epsilon & \triangleq & \lambda x.\, x \\ \_ + \_ & \triangleq & \lambda \alpha.\, \lambda \beta.\, \lambda x.\, \alpha\,(\beta\, x) \end{array}$$

Regular languages

Regular languages



Regular languages



Abstract signature

q_{if}, q_{ff} : type

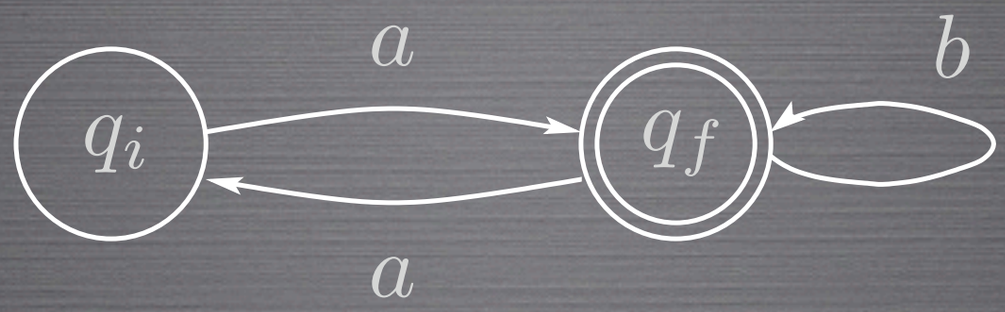
$t_0 : q_{ff}$

$t_1 : q_{ff} \to q_{if}$

$t_2 : q_{ff} \to q_{ff}$

$t_3 : q_{if} \to q_{ff}$

Regular languages



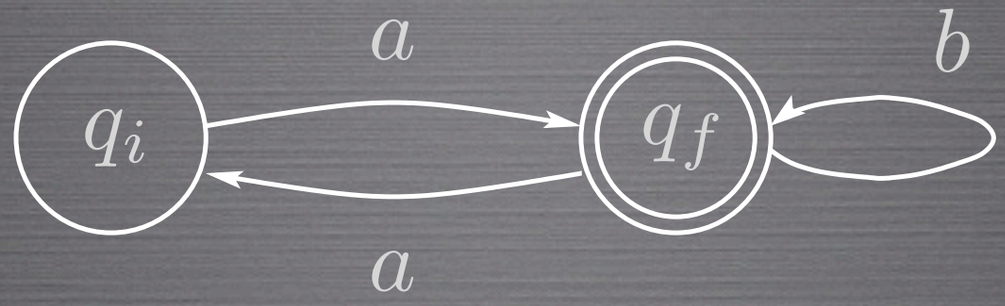
Abstract signature

$$q_{if}, q_{ff} : \text{type}$$
$$t_0 : q_{ff}$$
$$t_1 : q_{ff} \to q_{if}$$
$$t_2 : q_{ff} \to q_{ff}$$
$$t_3 : q_{if} \to q_{ff}$$

Object signature

$$\sigma : \text{type}$$
$$\# : \sigma$$
$$a, b : \sigma \to \sigma$$

Regular languages



Abstract signature

$$q_{if}, q_{ff} : \text{type}$$
$$t_0 : q_{ff}$$
$$t_1 : q_{ff} \to q_{if}$$
$$t_2 : q_{ff} \to q_{ff}$$
$$t_3 : q_{if} \to q_{ff}$$

Object signature

$$\sigma : \text{type}$$
$$\# : \sigma$$
$$a, b : \sigma \to \sigma$$

Lexicon

$$t_0 := \#$$
$$t_1 := \lambda x.\, a\, x$$
$$t_2 := \lambda x.\, b\, x$$
$$t_3 := \lambda x.\, a\, x$$

Context-free grammars

Context-free grammars

$$
\begin{aligned}
S &\to \mathrm{l}\,S\,\mathrm{r}\,S \\
S &\to \mathrm{l}\,\mathrm{r}\,S \\
S &\to \mathrm{l}\,S\,\mathrm{r} \\
S &\to \mathrm{l}\,\mathrm{r}
\end{aligned}
$$

Context-free grammars

$$
\begin{array}{rcl}
S & \to & \mathrm{l}\, S\, \mathrm{r}\, S \\
S & \to & \mathrm{l}\, \mathrm{r}\, S \\
S & \to & \mathrm{l}\, S\, \mathrm{r} \\
S & \to & \mathrm{l}\, \mathrm{r}
\end{array}
$$

$$
\begin{array}{rcl}
S & : & \mathrm{type} \\
\mathrm{p}_1 & : & S \to (S \to S) \\
\mathrm{p}_2 & : & S \to S \\
\mathrm{p}_3 & : & S \to S \\
\mathrm{p}_4 & : & S
\end{array}
$$

Context-free grammars

$$
\begin{array}{rcl}
S & \to & \mathrm{l}\, S\, \mathrm{r}\, S \\
S & \to & \mathrm{l}\, \mathrm{r}\, S \\
S & \to & \mathrm{l}\, S\, \mathrm{r} \\
S & \to & \mathrm{l}\, \mathrm{r}
\end{array}
$$

$$
\begin{array}{rcl}
S & : & \mathrm{type} \\
\mathrm{p}_1 & : & S \to (S \to S) \\
\mathrm{p}_2 & : & S \to S \\
\mathrm{p}_3 & : & S \to S \\
\mathrm{p}_4 & : & S
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{p}_1 & := & \lambda x.\, \lambda y.\, \mathrm{l} + x + \mathrm{r} + y \\
\mathrm{p}_2 & := & \lambda x.\, \mathrm{l} + \mathrm{r} + x \\
\mathrm{p}_3 & := & \lambda x.\, \mathrm{l} + x + \mathrm{r} \\
\mathrm{p}_4 & := & \mathrm{l} + \mathrm{r}
\end{array}
$$

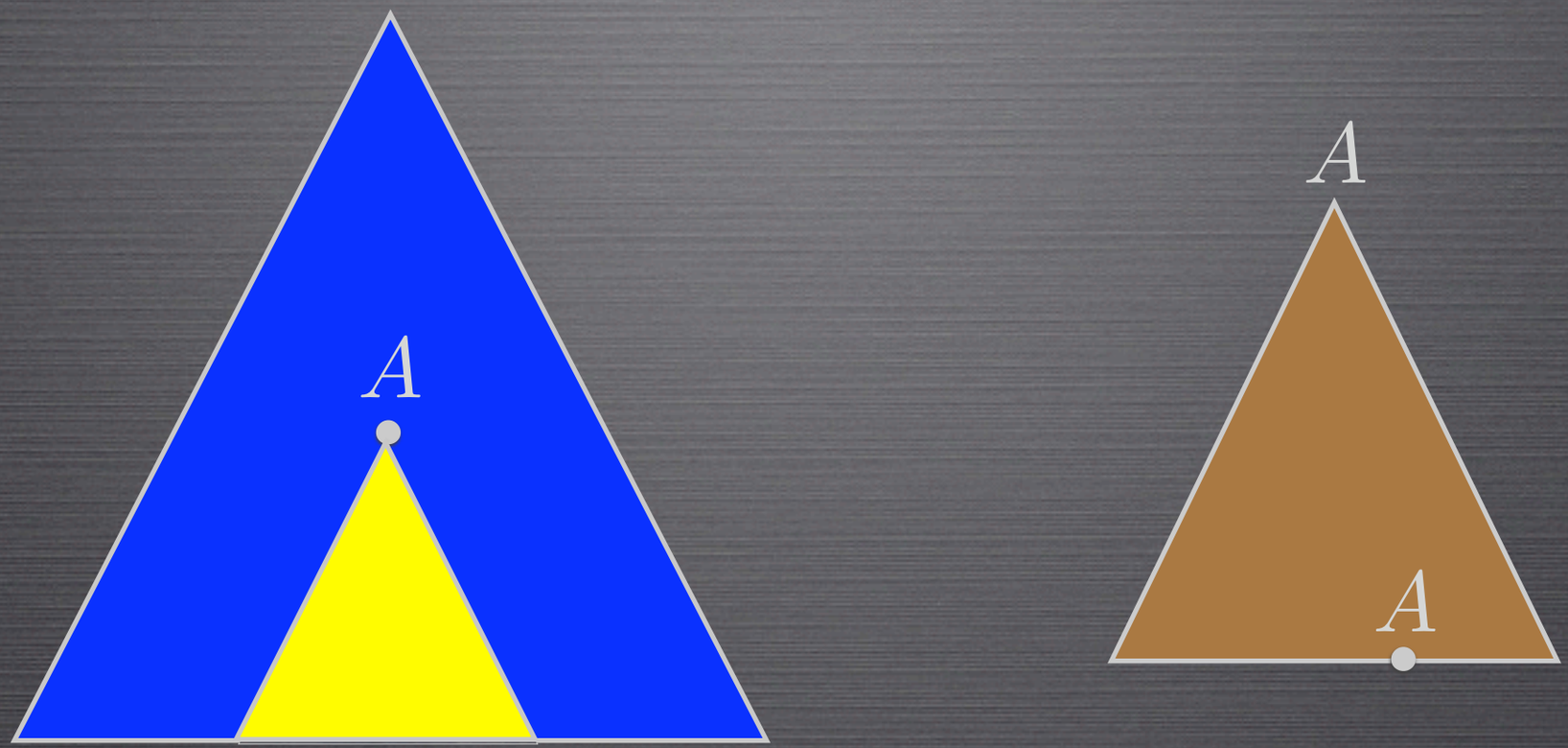
Tree Adjoining Grammars

Tree Adjoining Grammars

- Adjunction

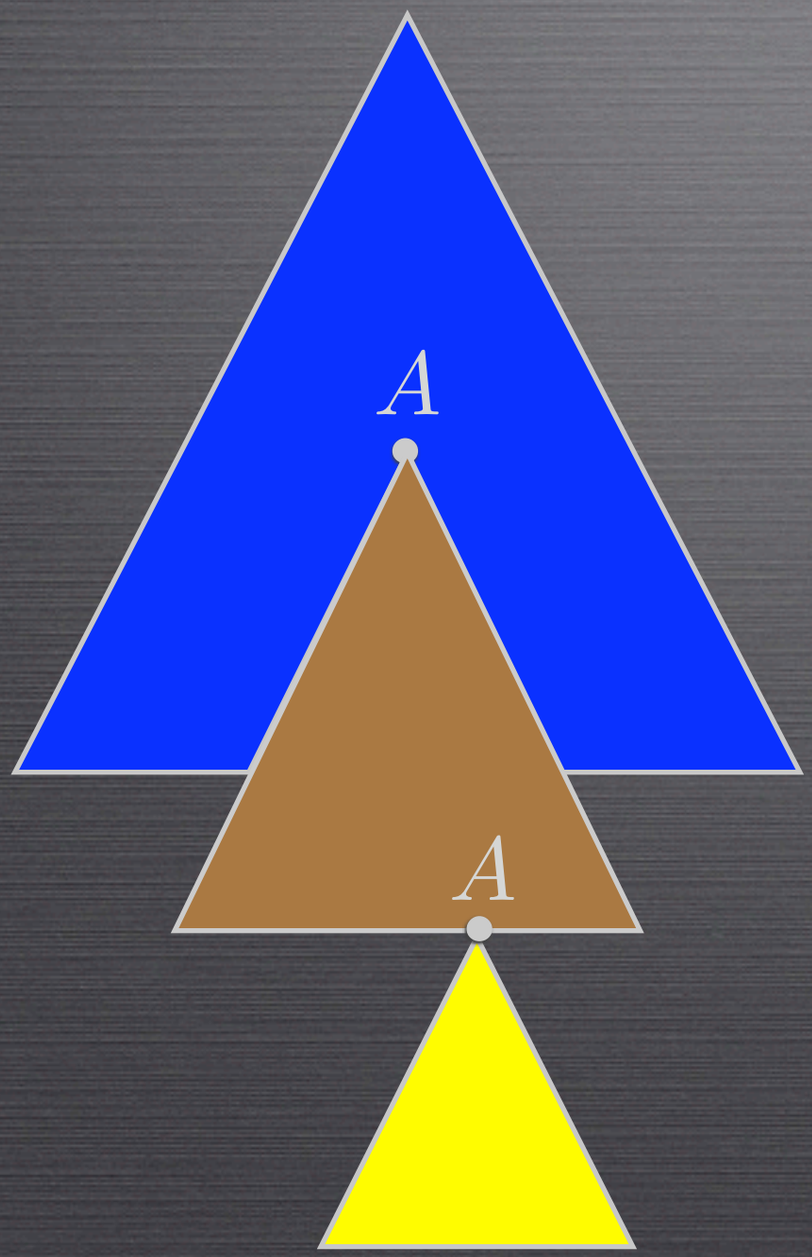
Tree Adjoining Grammars

Adjunction



Tree Adjoining Grammars

Adjunction



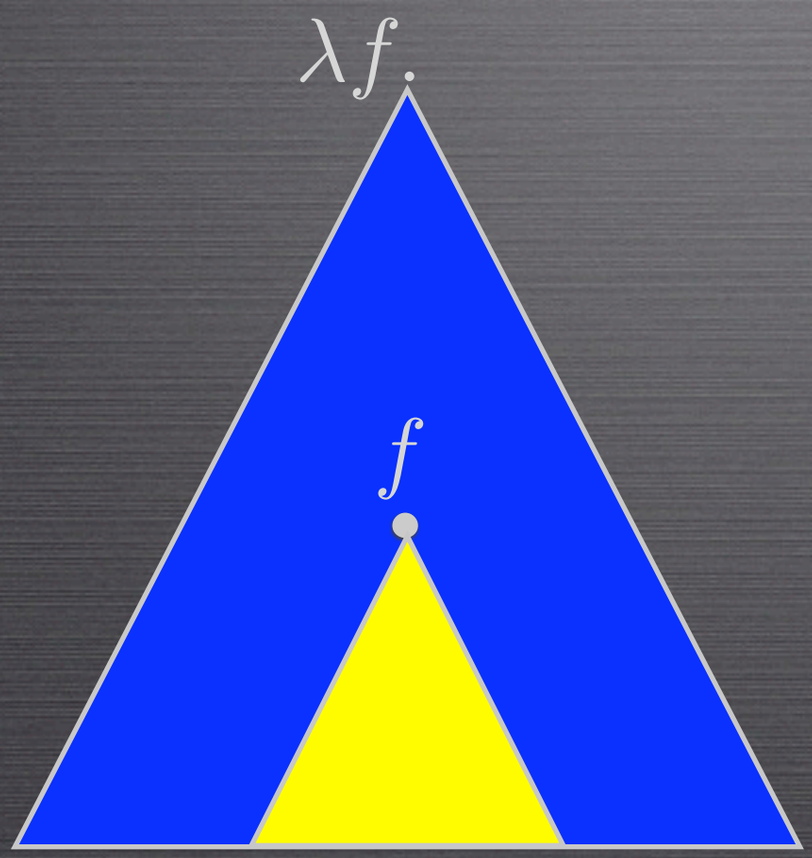
Tree Adjoining Grammars

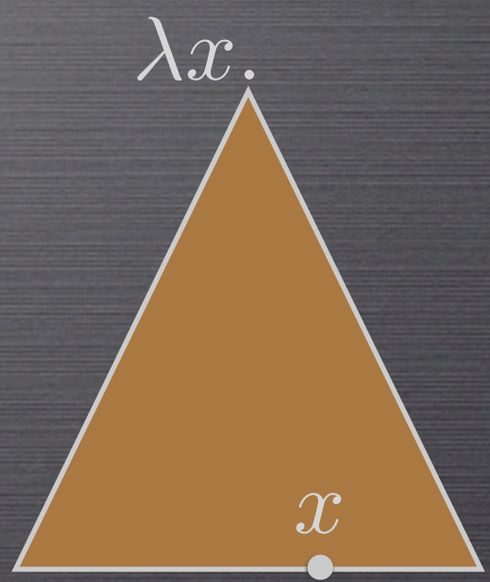
Tree Adjoining Grammars

Adjunction with lambda-terms

Tree Adjoining Grammars

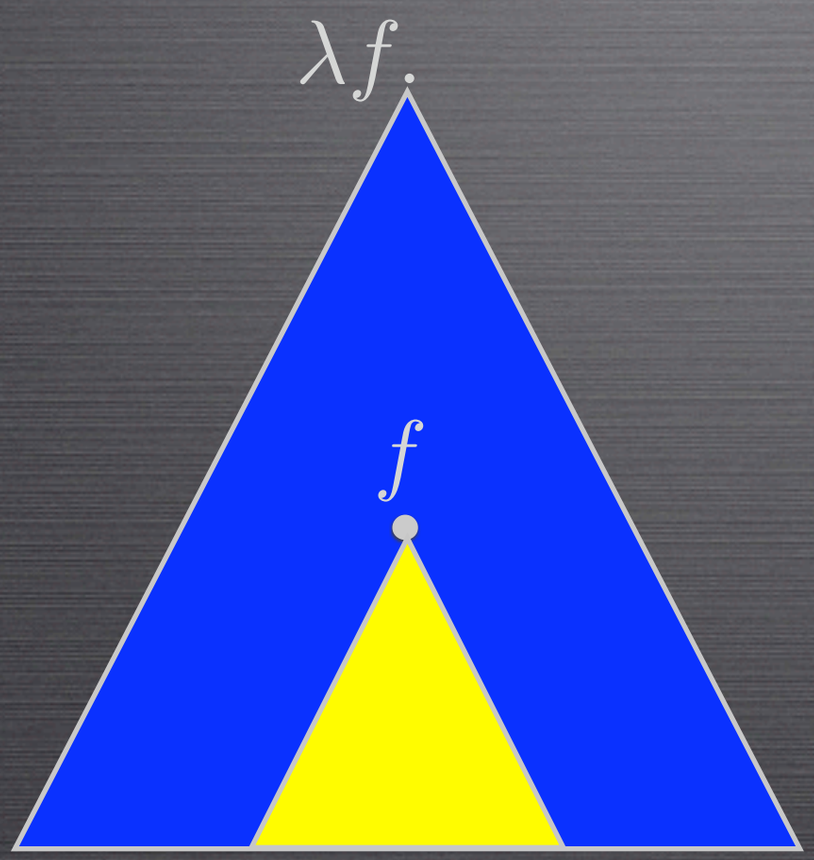
Adjunction with lambda-terms

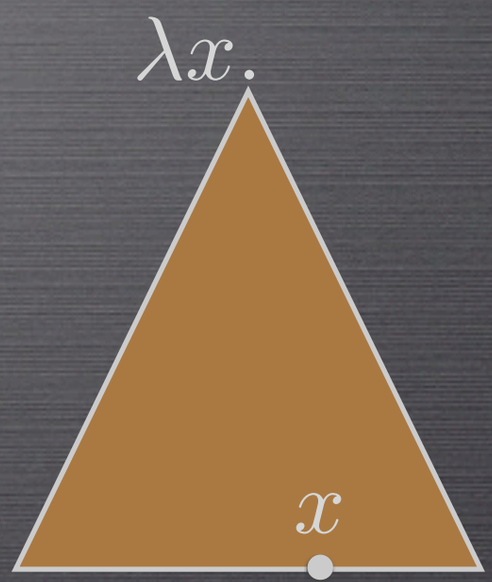




Tree Adjoining Grammars

Adjunction with lambda-terms

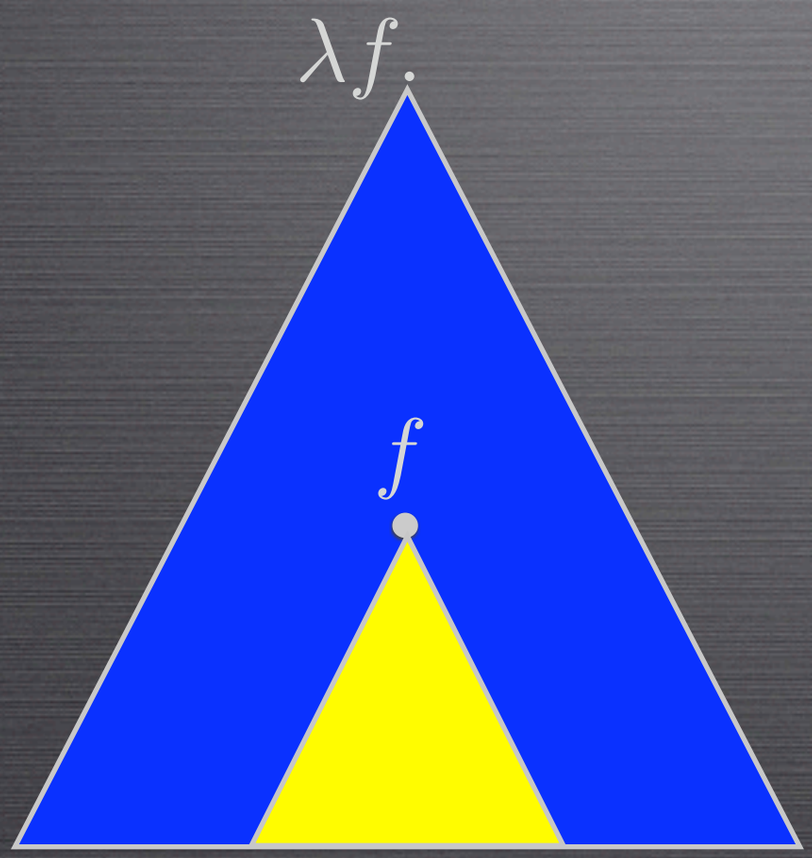




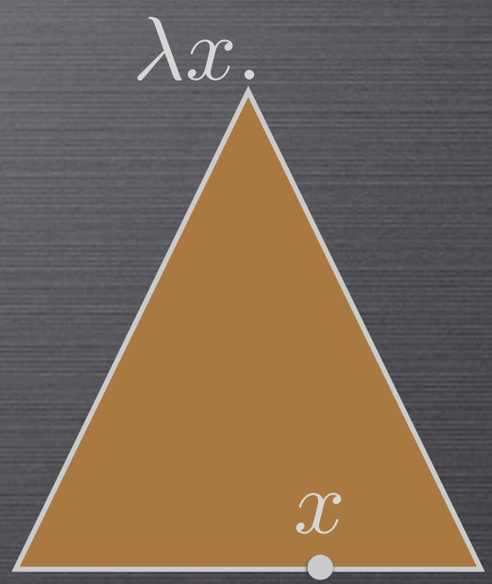
$string \rightarrow string$

Tree Adjoining Grammars

Adjunction with lambda-terms



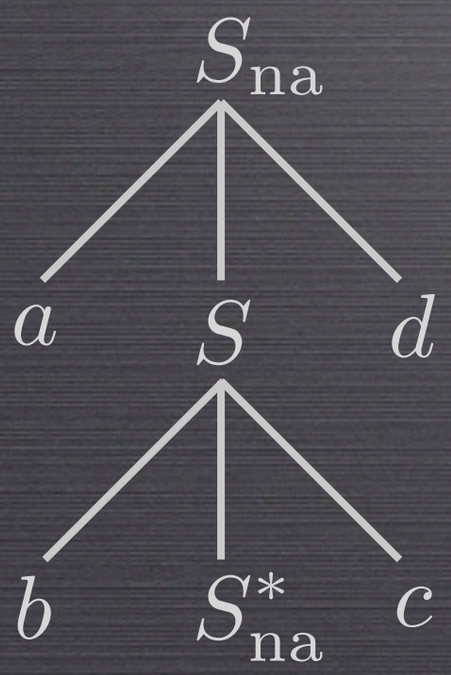
$(string \rightarrow string) \rightarrow string$



$string \rightarrow string$

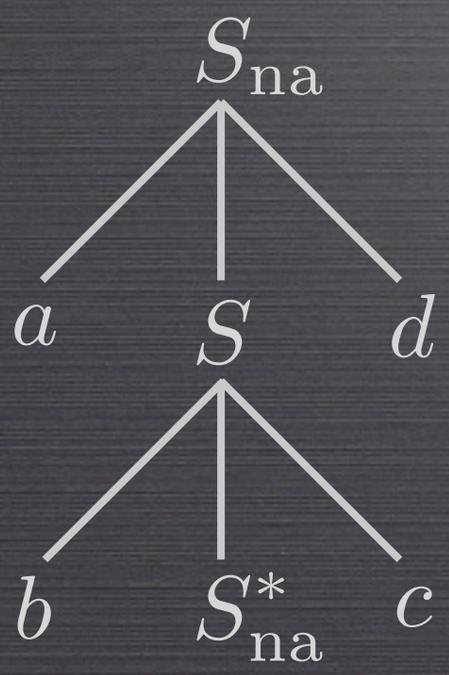
Example: $\{ a^n\, b^n\, c^n\, d^n \mid n \geq 0 \}$

Example: $\{ a^n\, b^n\, c^n\, d^n \mid n \geq 0 \}$



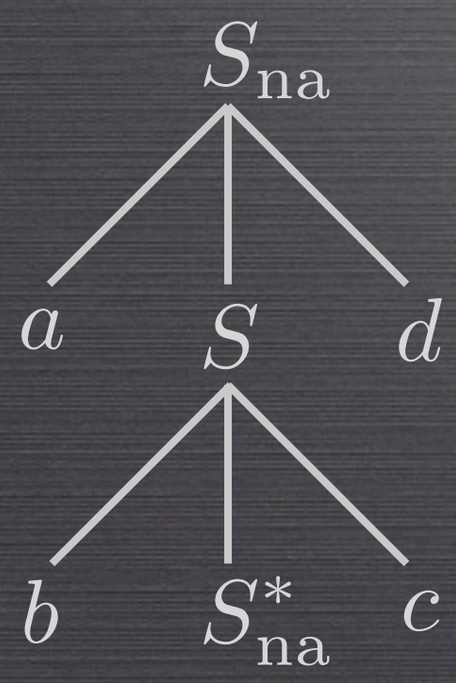
Example: $\{\, a^n\, b^n\, c^n\, d^n \mid n \geq 0 \,\}$

$$
\begin{array}{rcl}
S, S_{\mathrm{adj}} & : & \mathrm{type} \\
\mathrm{t}_i & : & S_{\mathrm{adj}} \to S \\
\mathrm{t}_a & : & S_{\mathrm{adj}} \to S_{\mathrm{adj}} \\
\mathrm{e} & : & S_{\mathrm{adj}}
\end{array}
$$



Example: $\{\, a^n\, b^n\, c^n\, d^n \mid n \geq 0 \,\}$

$$
\begin{array}{rcl}
S, S_{\mathrm{adj}} & : & \mathrm{type} \\
\mathrm{t}_i & : & S_{\mathrm{adj}} \to S \\
\mathrm{t}_a & : & S_{\mathrm{adj}} \to S_{\mathrm{adj}} \\
\mathrm{e} & : & S_{\mathrm{adj}}
\end{array}
$$



$$
\begin{array}{rcl}
S & := & string \\
S_{\mathrm{adj}} & := & string \to string \\
t_i & := & \lambda f.\, f\, \epsilon \\
t_a & := & \lambda g.\, \lambda x.\, a + g\,(b + x + c) + d \\
e & := & \lambda x.\, x
\end{array}
$$

Multiple context-free grammars

Multiple context-free grammars

Context-free grammars working on tuples, using linear, non-deleting operations

Multiple context-free grammars

Context-free grammars working on tuples, using linear, non-deleting operations

Tuples as linear lambda-terms:

$$\langle \alpha_1, \ldots, \alpha_n \rangle \triangleq \lambda f.\, f\, \alpha_1 \ldots \alpha_n$$

Multiple context-free grammars

Context-free grammars working on tuples, using linear, non-deleting operations

Tuples as linear lambda-terms:

$$\langle \alpha_1, \ldots, \alpha_n \rangle \triangleq \lambda f.\, f\, \alpha_1 \ldots \alpha_n$$

$$(string \cdots string \to string) \to string$$

Example: $\{\, a^m\, b^n\, c^m\, d^n \mid m, n \geq 0 \,\}$

Example: $\{\, a^m\, b^n\, c^m\, d^n \mid m, n \geq 0 \,\}$

$$S(x_1\, y_1\, x_2\, y_2) : -A(x_1, x_2), B(y_1, y_2)$$
$$A(a\, x_1, c\, x_2) : -A(x_1, x_2)$$
$$A(\epsilon, \epsilon)$$
$$B(b\, y_1, d\, y_2) : -B(y_1, y_2)$$
$$B(\epsilon, \epsilon)$$

Example: $\{\, a^m\, b^n\, c^m\, d^n \mid m, n \geq 0 \,\}$

$$
\begin{array}{l}
S(x_1\, y_1\, x_2\, y_2) : -A(x_1, x_2), B(y_1, y_2) \\
A(a\, x_1, c\, x_2) : -A(x_1, x_2) \\
A(\epsilon, \epsilon) \\
B(b\, y_1, d\, y_2) : -B(y_1, y_2) \\
B(\epsilon, \epsilon)
\end{array}
$$

$$
\begin{array}{rcl}
A, B, S & : & \text{type} \\
\mathrm{p}_1 & : & A \to (B \to S) \\
\mathrm{p}_2 & : & A \to A \\
\mathrm{p}_3 & : & A \\
\mathrm{p}_4 & : & B \to B \\
\mathrm{p}_5 & : & B
\end{array}
$$

Example: $\{\, a^m\, b^n\, c^m\, d^n \mid m, n \geq 0 \,\}$

$$
\begin{aligned}
&S(x_1\, y_1\, x_2\, y_2) :- A(x_1, x_2), B(y_1, y_2)\\
&A(a\, x_1, c\, x_2) :- A(x_1, x_2)\\
&A(\epsilon, \epsilon)\\
&B(b\, y_1, d\, y_2) :- B(y_1, y_2)\\
&B(\epsilon, \epsilon)
\end{aligned}
$$

$$
\begin{aligned}
S &:= string\\
A, B &:= (string \to string \to string) \to string\\
\mathrm{p}_1 &:= \lambda P\, Q.\, P\, (\lambda x_1\, x_2.\, Q\, (\lambda y_1\, y_2.\, x_1 + y_1 + x_2 + y_2))\\
\mathrm{p}_2 &:= \lambda P.\, \lambda f.\, P\, (\lambda x_1\, x_2.\, f\, (a + x_1)\, (c + x_2))\\
\mathrm{p}_3 &:= \lambda f.\, f\, \epsilon\, \epsilon\\
\mathrm{p}_4 &:= \lambda Q.\, \lambda f.\, Q\, (\lambda y_1\, y_2.\, f\, (b + y_1)\, (d + y_2))\\
\mathrm{p}_5 &:= \lambda f.\, f\, \epsilon\, \epsilon
\end{aligned}
$$