Grammar Abstract Interpretation

Patrick COUSOT
ENS

Radhia COUSOT
CNRS – X

Seminar in Honor of Reinhard Wilhelm's 60th Birthday

Dagstuhl, Saturday, June 10th, 2006

Reinhard's work on grammar analysis

- Grammar analysis is like program/data flow analysis that is solving fixpoint equations
- Bottom-up equations:
  - e.g. first
  - $\underbrace{X} \rightarrow \underbrace{X_1 \ldots X_n}$

    flow of information
- Top-down equations:
  - e.g. follow
  - $\underbrace{X} \rightarrow \underbrace{X_1 \ldots X_n}$

    flow of information

Bottom-up grammar flow analysis (from Wilhelm & Maurer's book on compilation, french translation)

Définition 8.2.18 (Analyse de flux ascendante)
Soit G une GNC ; un **problème d'analyse de flux ascendant** pour G et I comprend :

- un **domaine de valeurs** $D\uparrow$: ce domaine est l'ensemble des informations possibles pour les non-terminaux ;
- une **fonction de transfert** $F_p\uparrow: D\uparrow^{n_p} \to D\uparrow$ pour chaque production $p \in P$;
- une **fonction de combinaison** $\nabla\uparrow: 2^{D\uparrow} \to D\uparrow$.

Abstract domain

Ceci étant posé, on définit pour une grammaire donnée un système récursif d'équations :

$$\boxed{I(X) \quad = \nabla\uparrow \{F_p\uparrow (I(p[1]), \ldots, I(p[n_p])) \mid p[0] = X\} \quad \forall X \in V_N} \qquad (I\uparrow)$$

□

System of abstract fixpoint equations

Exemple 8.2.12 (Productivité des non-terminaux)

$D\uparrow$	{ *vrai*, *faux* }	*vrai* pour productif
$F_p\uparrow$	$\wedge$	(*vrai* pour $n_p = 0$, i.e. pour les productions terminales)
$\nabla\uparrow$	$\vee$	(*faux* pour les non-terminaux sans alternative)

Le système d'équations pour le problème de la productivité des non-terminaux est alors :

$$\boxed{Pr(X) \quad = \bigvee\{\bigwedge_{i=1}^{n_p} Pr(p[i]) \mid p[0] = X\} \quad \text{pour tous les } X \in V_N} \qquad (Pr)$$

□

instantiation on an example (non-terminal productivity)

Top-down grammar analysis :

Définition 8.2.19 (Analyse de flux descendante)
Soit G une GNC ; un problème d'analyse de flux descendant pour G et I comprend :

- un domaine de valeurs $D\downarrow$;
- n_p fonctions de transfert $F_{p,i}\downarrow : D\downarrow \to D\downarrow$, $1 \le i \le n_p$, pour chaque production $p \in P$;
- une fonction de combinaison $\nabla\downarrow : 2^{D\downarrow} \to D\downarrow$;
- une valeur I_0 pour S.

abstract domain

Etant donnée une grammaire, on définit comme précédemment un système récursif d'équations pour I ; pour des raisons de lisibilité, nous donnons la définition de I à la fois pour les non-terminaux et pour les occurrences de non-terminaux :

$$\boxed{\begin{array}{lcl} I(S) & = & I_0 \\ I(p,i) & = & F_{p,i}\downarrow\,(I(p[0]))\ \text{pour tous}\ p \in P,\ 1 \le i \le n_p \\ I(X) & = & \nabla\downarrow\,\{I(p,i) \mid p[i] = X\},\ \text{pour tous}\ X \in V_N - \{S\} \end{array}} \qquad (I\downarrow)$$

□

system of abstract equations

Exemple 8.2.13 (Non-terminaux accessibles)

$D\downarrow$	$\{vrai, faux\}$	*vrai* pour accessible
$F_{p,i}\downarrow$	id	identité
$\nabla\downarrow$	$\bigvee$	OU booléen (*faux*, s'il n'existe pas d'occurrence de non-terminal)
I_0	*vrai*	

On en déduit pour Ac le système récursif d'équations :

$$\boxed{\begin{array}{lcl} Ac(S) & = & vrai \\ Ac(X) & = & \bigvee\{Ac(p[0]) \mid p[i] = X,\ 1 \le i \le n_p\}\ \ \forall X \in V_N - \{S\} \end{array}} \qquad (Ac)$$

□

Instantiation on an example (accessible non-terminals)

Contribution of this talk (building upon Reinhard's pioneer work) :

- We define an operational semantics of grammars ($\simeq$ pushdown automata)
- We abstract this semantics
  - Bottom-up $\underline{X} \to \underline{X_1 \dots X_n}$, synthesizing information from sons to father
  - Top-down $\boxed{X} \to X_1 \dots X_n$, inheriting information from father to sons, by a replacement / rewriting process of variables $\boxed{A}$
- The bottom-up semantics can be abstracted in bottom-up grammar analysis algorithms

- The top-down semantics can be abstracted in top-down grammar analysis algorithms
- The top-down semantics can be abstracted into the bottom-up semantics (explaining why there are often two equivalent ways $\downarrow$ or $\uparrow$ to define the same notion for grammars e.g. protolangage : inherited from axiom
  synthesized equationally
- Not only

  all grammar flow analysis algorithms

  but also

  all parsing algorithms

  are abstract interpretations of the operational semantics $\xrightarrow{\alpha}$ top-dow. semantics $\xrightarrow{\alpha}$ bottom-up semantics

- This pave the way for
  - automatic / computer assisted design of grammar analysis / parsing algorithms
  - automated formal verification of these algorithm
  - formal verification of compiler front-ends.

- A unifying formalization viewing
  - compilation as a science (with formal justifications for the principles and algorithms)

as opposed to
  - compilation as a technology (a collection of techniques and tools).

OPERATIONAL - SEMANTICS OF GRAMMARS

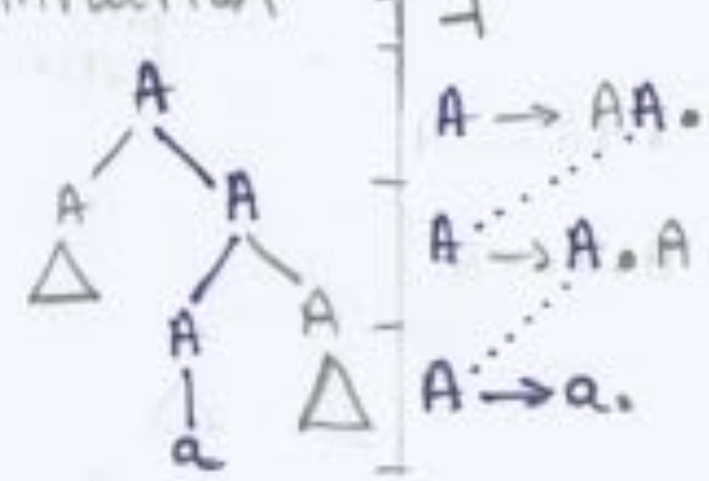
Transition system

Grammar $A \to AA \mid a$

- states : stacks

$\dashv [A \to AA.][A \to A.A][A \to a.]$

intuition



- transition : to traverse the syntax tree from top-down left-to right using a stack(*)

(*) the operational version of recursion !

Transition rules(*) (derivation from any nonterminal)

$$
\begin{array}{rl}
\vdash \xrightarrow{(\!|A} \dashv[A \to \bullet\sigma], & A \to \sigma \in \mathscr{R} \\
\varpi[A \to \sigma\bullet a\sigma'] \xrightarrow{a} \varpi[A \to \sigma a\bullet\sigma'], & A \to \sigma a\sigma' \in \mathscr{R} \\
\varpi[A \to \sigma\bullet B\sigma'] \xrightarrow{(\!|B} \varpi[A \to \sigma B\bullet\sigma'][B \to \bullet\varsigma], & A \to \sigma B\sigma' \in \mathscr{R} \wedge B \to \varsigma \in \mathscr{R} \\
\varpi[A \to \sigma\bullet] \xrightarrow{A|\!)} \varpi, & A \to \sigma \in \mathscr{R}.
\end{array}
$$

Initial state : $\vdash$

Intuition :

$\xrightarrow{(\!|A}$: start generating a terminal sentence from non-terminal A

$\xrightarrow{A|\!)}$: the generation of a terminal sentence for non-terminal A is finished

$\xrightarrow{a}$: generate a terminal a

(*)

Derivations

- maximal finite execution traces(*) of the transition system of the grammar
- Grammar $A \to AA \mid a$
- Ex. derivation for sentence a :

$$\vdash \xrightarrow{IA} \dashv [A \to \bullet a] \xrightarrow{a} \dashv [A \to a \bullet] \xrightarrow{AD} \dashv$$

- Ex : derivation for sentence aa :

$$\vdash \xrightarrow{IA} \dashv [A \to \bullet AA] \xrightarrow{IA} \dashv [A \to \bullet AA][A \to \bullet a] \xrightarrow{a} \dashv [A \to \bullet AA][A \to a \bullet] \xrightarrow{AD}$$
$$\dashv [A \to A \bullet A] \xrightarrow{IA} \dashv [A \to A \bullet A][A \to \bullet a] \xrightarrow{a} \dashv [A \to A \bullet A][A \to a \bullet] \xrightarrow{AD} \dashv$$
$$[A \to AA \bullet] \xrightarrow{AD} \dashv .$$

(*) immediate generalization to infinite languages

BOTTOM - UP SEMANTICS
OF GRAMMARS

Bottom-up derivation semantics of grammars

- Define the derivations for non-terminals
  - By a lfp of a system of equations
  - where derivations are built bottom-up

$$\underbrace{X}_{\text{derivation}} \longrightarrow \underbrace{X_1}_{\text{derivation}_1} \;\cdots\; \underbrace{X_n}_{\text{derivation}_n}$$

- Here is the bottom-up derivation semantics:

$$\mathsf{S}^d[\![G]\!] = \mathrm{lfp}^{\subseteq}\, \hat{\mathsf{F}}^d[\![G]\!]$$

the derivations defined by the operational semantics

denotational semantics

the fixpoint operator.

$$\hat{\mathsf{F}}^d[\![G]\!] \triangleq \lambda T \cdot \bigcup_{A \to \sigma \in \mathscr{R}} \vdash \xrightarrow{(A} \hat{\mathsf{F}}^d[A \to \bullet\sigma]T \xrightarrow{A)} \dashv$$

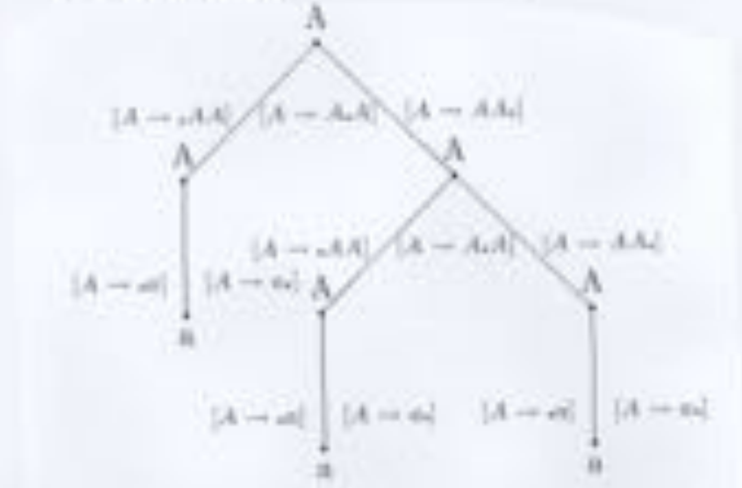
$$\hat{\mathsf{F}}^d[A \to \sigma\bullet a\sigma'] \triangleq \lambda T \cdot (\dashv[A \to \sigma\bullet a\sigma']) \xrightarrow{a} \hat{\mathsf{F}}^d[A \to \sigma a\bullet\sigma']T$$

$$\hat{\mathsf{F}}^d[A \to \sigma\bullet B\sigma'] \triangleq \lambda T \cdot (\langle \dashv[A \to \sigma\bullet B\sigma'], \dashv[A \to \sigma B\bullet\sigma'] \rangle \uparrow T.B) \,\mathring{\imath}\, \hat{\mathsf{F}}^d[A \to \sigma B\bullet\sigma']T$$

$$\hat{\mathsf{F}}^d[A \to \sigma\bullet] \triangleq \lambda T \cdot (\dashv[A \to \sigma\bullet]) .$$

Abstraction of derivations to <u>derivation trees</u>

– Derivation trees : $A \to AA, A \to a$



} abstract (derivation tree)

≡

$\alpha^{?(*)}$

$(A[A \to \bullet AA](A[A \to \bullet a]a[A \to a\bullet]A)[A \to A\bullet A](A[A \to \bullet AA](A[A \to \bullet a]a[A \to a\bullet]A)[A \to A\bullet A](A[A \to \bullet a]a[A \to a\bullet]A)[A \to AA\bullet]A)[A \to AA\bullet]A)$ } parenthesized representation

$$\vdash \xrightarrow{(A} \dashv [A \to \bullet AA] \xrightarrow{(A} \dashv [A \to A\bullet A]\,[A \to \bullet a] \xrightarrow{a} \dashv [A \to A\bullet A]\,[A \to a\bullet] \xrightarrow{A)} \dashv [A \to A\bullet A] \xrightarrow{(A} \dashv [A \to AA\bullet]\,[A \to \bullet AA] \xrightarrow{(A} \dashv [A \to AA\bullet]\,[A \to A\bullet A]\,[A \to \bullet a] \xrightarrow{a} \dashv [A \to AA\bullet]\,[A \to A\bullet A]\,[A \to a\bullet] \xrightarrow{A)} \dashv [A \to AA\bullet]\,[A \to A\bullet A] \xrightarrow{(A} \dashv [A \to AA\bullet]\,[A \to AA\bullet]\,[A \to \bullet a] \xrightarrow{a} \dashv [A \to AA\bullet]\,[A \to AA\bullet]\,[A \to a\bullet] \xrightarrow{A)} \dashv [A \to AA\bullet]\,[A \to AA\bullet] \xrightarrow{A)} \dashv [A \to AA\bullet] \xrightarrow{A)} \dashv.$$

} concrete derivation (for aaa)

(*) essentially get rid of → and abstract stacks by their top

— Fixpoint derivation tree semantics

- $\alpha \circ F^\# = F \circ \alpha \models\Rightarrow \alpha(\mathrm{lfp}\, F) = \mathrm{lfp}\, F^\#$
- $F^\# = \gamma \circ F \circ \alpha$

so their is only one possible $F^\#$ obtained by calculus :

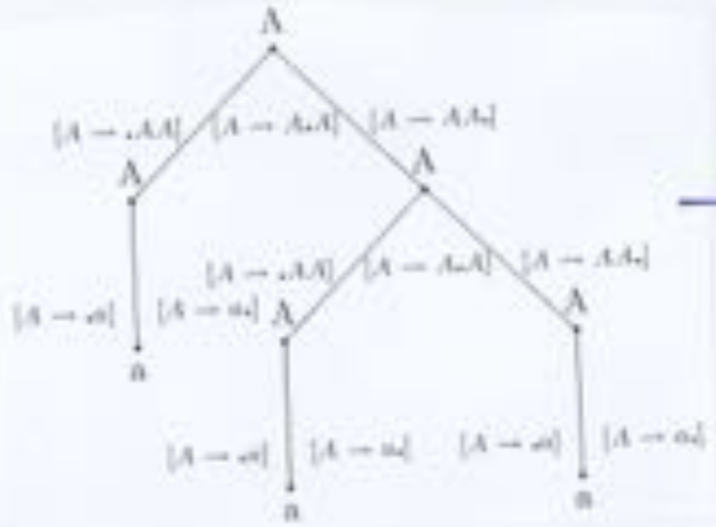
Definition : $\mathsf{S}^{\delta}[\![\mathcal{G}]\!] \triangleq \alpha^{\delta}(\mathsf{S}^{d}[\![\mathcal{G}]\!])$.

Abstraction theorem : $\mathsf{S}^{\delta}[\![\mathcal{G}]\!] = \mathrm{lfp}^{\subseteq}\, \mathsf{F}^{\delta}[\![\mathcal{G}]\!]$

$$
\begin{aligned}
\mathsf{F}^{\delta}[\![\mathcal{G}]\!] &\triangleq \lambda D \cdot \bigcup_{A \to \sigma \in \mathcal{G}} \{A\ \mathsf{F}^{\delta}[A \to \bullet\sigma] D\ A\} \\
\mathsf{F}^{\delta}[A \to \sigma \bullet a \sigma'] &\triangleq \lambda D \cdot [A \to \sigma \bullet a \sigma']\ a\ \mathsf{F}^{\delta}[A \to \sigma a \bullet \sigma'] D \\
\mathsf{F}^{\delta}[A \to \sigma \bullet B \sigma'] &\triangleq \lambda D \cdot [A \to \sigma \bullet B \sigma']\ D.B\ \mathsf{F}^{\delta}[A \to \sigma B \bullet \sigma'] D \\
\mathsf{F}^{\delta}[A \to \sigma \bullet] &\triangleq \lambda D \cdot [A \to \sigma \bullet] .
\end{aligned}
$$

Syntax tree abstraction and bottom-up semantics

– Abstraction



$\xrightarrow{\alpha^{\hat{s}}}$

representation :

$(A(AaA)(A(AaA)(AaA)A)A)$

– Fixpoint semantics :

- Definition : $\mathsf{S}^{\hat{s}}[\![\mathcal{G}]\!] \triangleq \alpha^{\hat{s}}(\mathsf{S}^{s}[\![\mathcal{G}]\!])$
- Abstraction theorem : $\mathsf{S}^{\hat{s}}[\![\mathcal{G}]\!] = \mathrm{lfp}^{\subseteq}\, \mathsf{F}^{\hat{s}}[\![\mathcal{G}]\!]$

$$\mathsf{F}^{\hat{s}}[\![\mathcal{G}]\!] \triangleq \lambda S \cdot \bigcup_{A \to \sigma \in \mathcal{R}} (A\, \hat{\mathsf{F}}^{\hat{s}}[A \to \bullet\sigma]S\; A)$$

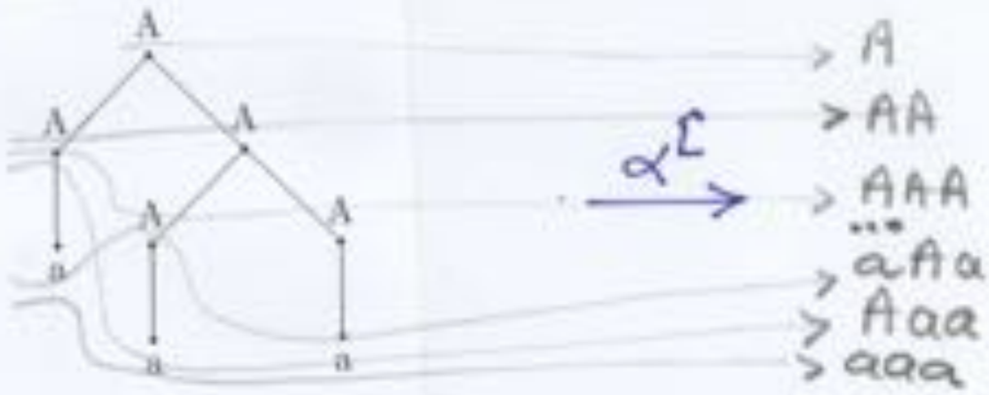
$$\hat{\mathsf{F}}^{\hat{s}}[A \to \sigma\bullet a\sigma'] \triangleq \lambda S \cdot a\, \hat{\mathsf{F}}^{\hat{s}}[A \to \sigma a\bullet\sigma']S$$

$$\hat{\mathsf{F}}^{\hat{s}}[A \to \sigma\bullet B\sigma'] \triangleq \lambda S \cdot S.B\, \hat{\mathsf{F}}^{\hat{s}}[A \to \sigma B\bullet\sigma']S$$

$$\hat{\mathsf{F}}^{\hat{s}}[A \to \sigma\bullet] \triangleq \lambda S \cdot \epsilon\,.$$

Protolanguage abstraction & bottom-up semantics

– Abstraction :



– Fixpoint semantics :

. Definition : $\mathsf{S}^{\mathcal{L}}[\![\mathcal{G}]\!] \triangleq \alpha^{\mathcal{L}}(\mathsf{S}^{t}[\![\mathcal{G}]\!])$

. Abstraction theorem : $\mathsf{S}^{\mathcal{L}}[\![\mathcal{G}]\!] = \mathit{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{\mathcal{L}}[\![\mathcal{G}]\!]$

$$
\begin{aligned}
\hat{\mathsf{F}}^{\mathcal{L}}[\![\mathcal{G}]\!] &\triangleq \boldsymbol{\lambda}\rho \cdot \boldsymbol{\lambda} A \cdot \bigcup_{A \to \sigma \in \mathcal{R}} \{A\} \cup \hat{\mathsf{F}}^{\mathcal{L}}[\![A \to \bullet\sigma]\!]\rho \\
\hat{\mathsf{F}}^{\mathcal{L}}[\![A \to \sigma \bullet a\sigma']\!] &\triangleq \boldsymbol{\lambda}\rho \cdot a\, \hat{\mathsf{F}}^{\mathcal{L}}[\![A \to \sigma a \bullet \sigma']\!]\rho \\
\hat{\mathsf{F}}^{\mathcal{L}}[\![A \to \sigma \bullet B\sigma']\!] &\triangleq \boldsymbol{\lambda}\rho \cdot (\{B\} \cup \rho(B))\, \hat{\mathsf{F}}^{\mathcal{L}}[\![A \to \sigma B \bullet \sigma']\!]\rho \\
\hat{\mathsf{F}}^{\mathcal{L}}[\![A \to \sigma \bullet]\!] &\triangleq \boldsymbol{\lambda}\rho \cdot \epsilon
\end{aligned}
$$

Terminal language abstraction & bottom-up semantics

- Abstraction :

$$A \quad AA \quad AaA \quad Aaa \ldots \quad aaa \xrightarrow{\alpha^{\hat{\ell}}} aaa$$

- Fixpoint semantics :
  - Definition : $\mathsf{S}^{\ell}[\![\mathcal{G}]\!] \triangleq \dot{\alpha}^{\ell}(\mathsf{S}^{\vec{L}}[\![\mathcal{G}]\!])$
  - Abstraction theorem (*) $\mathsf{S}^{\ell}[\![\mathcal{G}]\!] = \mathit{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{\ell}[\![\mathcal{G}]\!]$

$$\begin{aligned}
\hat{\mathsf{F}}^{\ell}[\![\mathcal{G}]\!] &\triangleq \boldsymbol{\lambda}\rho \cdot \boldsymbol{\lambda} A \cdot \bigcup_{A \to \sigma \in \mathscr{R}} \hat{\mathsf{F}}^{\ell}[A \to \bullet\sigma]\rho \\
\hat{\mathsf{F}}^{\ell}[A \to \sigma \bullet a\sigma'] &\triangleq \boldsymbol{\lambda}\rho \cdot a\, \hat{\mathsf{F}}^{\ell}[A \to \sigma a \bullet \sigma']\rho \\
\hat{\mathsf{F}}^{\ell}[A \to \sigma \bullet B\sigma'] &\triangleq \boldsymbol{\lambda}\rho \cdot \rho(B)\, \hat{\mathsf{F}}^{\ell}[A \to \sigma B \bullet \sigma']\rho \\
\hat{\mathsf{F}}^{\ell}[A \to \sigma \bullet] &\triangleq \boldsymbol{\lambda}\rho \cdot \epsilon
\end{aligned}$$

(*) Ginsburg, Rice, Schützenberger fixpoint characterization of the terminal language

The hierarchy of bottom-up grammar semantics

$S^{\ell}[\![G]\!] = \mathrm{lfp}^{\subseteq}\ \hat{F}^{\ell}[\![G]\!]$

language

$S^{\hat{L}}[\![G]\!] = \mathrm{lfp}^{\subseteq}\ \hat{F}^{\hat{L}}[\![G]\!]$

protolanguage

$S^{\hat{s}}[\![G]\!] = \mathrm{lfp}^{\subseteq}\ \hat{F}^{\hat{s}}[\![G]\!]$

syntax tree

$S^{\hat{\delta}}[\![G]\!] = \mathrm{lfp}^{\subseteq}\ \hat{F}^{\hat{\delta}}[\![G]\!]$

derivation trees

$S^{\hat{d}}[\![G]\!] = \mathrm{lfp}^{\subseteq}\ \hat{F}^{\hat{d}}[\![G]\!]$

derivations

TOP-DOWN SEMANTICS OF GRAMMARS

Generalize the protolanguage derivation $\Rightarrow$ and $\underbrace{post(\overset{*}{\Longrightarrow})}_{\text{all transitive derivations from axiom}}(\{\bar{S}\})$

initial state is the start symbol

Protoderivations

- A top-down definition of maximal derivations
- Example : $A \to AA \mid a$

variable

$\vdash \xrightarrow{\boxed{A}} \dashv$

rewritten using rule $A \to AA$.

$\boxed{D} \Rightarrow_o$

$$\vdash \xrightarrow{(A} \dashv[A \to \bullet AA] \xrightarrow{\boxed{A}} \dashv[A \to A \bullet A] \xrightarrow{\boxed{A}} \dashv[A \to AA\bullet] \xrightarrow{A)} \dashv$$

$\boxed{D} \Rightarrow_o$

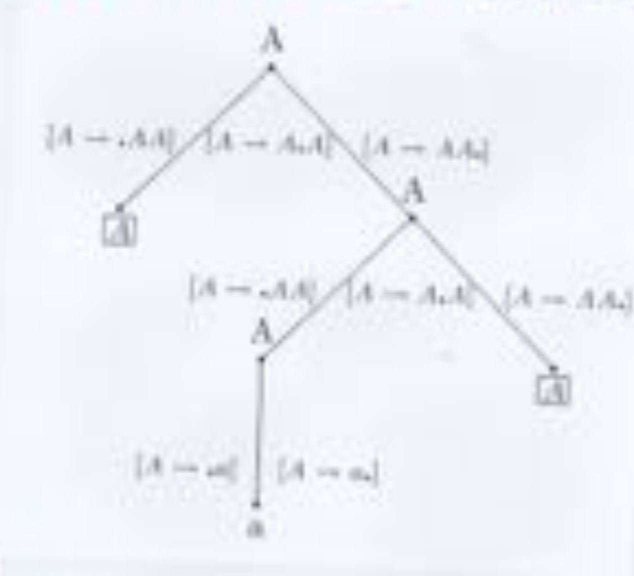
$$\vdash \xrightarrow{(A} \dashv[A \to \bullet AA] \xrightarrow{\boxed{A}} \dashv[A \to A \bullet A] \xrightarrow{(A} \dashv[A \to AA\bullet][A \to \bullet a] \xrightarrow{a} \dashv[A \to AA\bullet][A \to a\bullet] \xrightarrow{A)} \dashv[A \to AA\bullet] \xrightarrow{A)} \dashv$$

$\boxed{D} \Rightarrow_o$

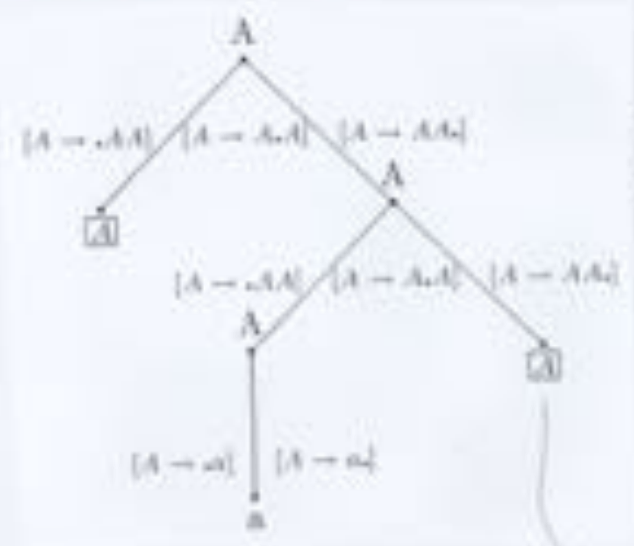
$$\vdash \xrightarrow{(A} \dashv[A \to \bullet AA] \xrightarrow{(A} \dashv[A \to A\bullet A][A \to \bullet a] \xrightarrow{a} \dashv[A \to A\bullet A][A \to a\bullet] \xrightarrow{A)} \dashv[A \to A\bullet A] \xrightarrow{(A} \dashv[A \to AA\bullet][A \to \bullet a] \xrightarrow{a} \dashv[A \to AA\bullet][A \to a\bullet] \xrightarrow{A)} \dashv[A \to AA\bullet] \xrightarrow{A)} \dashv$$

Abstraction of protoderivations into protoderivation trees

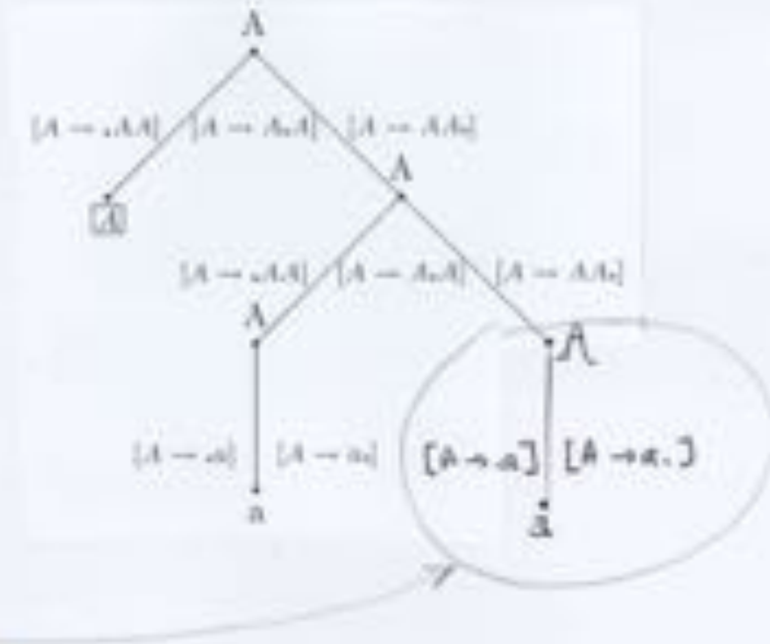
– Protoderivation tree :



– Example of derivation $\boxed{\delta}\Rightarrow$:



$\boxed{\delta}\Rightarrow$



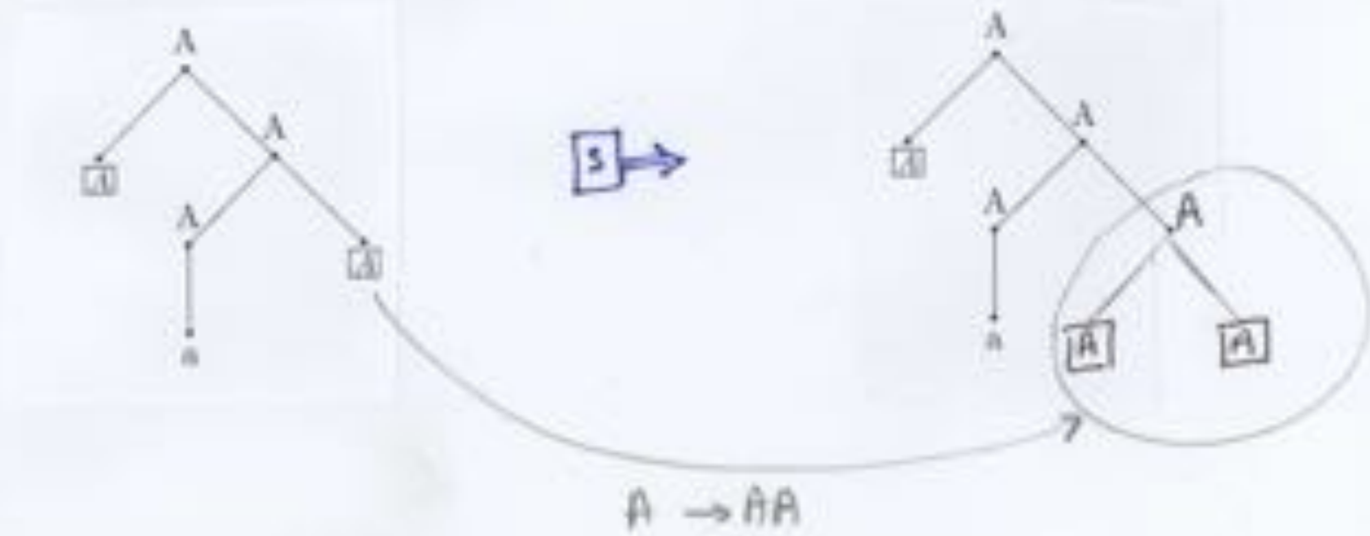
Abstraction of protoderivation trees into protosyntax tree (i.e. syntax trees with variables)

- Protosyntax tree :

Representation :

(A □)(A (A a A)(□ A) A)

- Example of derivation : $\boxed{S}\Rightarrow$:



Abstraction of protosyntax trees into protosentences

- Protosentences $(A \to AA \mid a)$

  $A \quad Aa \quad AaA \quad aaa \ \ldots$ $\qquad$ A ou $\boxed{A}$ variable

- Protosentence derivation (the classical notion)

  $$A \Rightarrow AA \Rightarrow Aa \Rightarrow AAa \Rightarrow aAa \Rightarrow aaa$$

- Example of abstraction :

$$\xrightarrow{\alpha^{\check{L}}} \quad AaA \qquad (\text{or } \boxed{A}\, a\, \boxed{A})$$

$$\alpha^{L}((A\boxed{A}(A(AaA)\boxed{A}A)A)) = AaA$$

Fixpoint top-down semantics

- All top-down semantics are based on a derivation relation $\Rrightarrow$ (for protoderivations, protoderivation trees, protosyntax trees, protosentences)
- The semantics is

$$S = \mathrm{post}(\Rrightarrow^*)(\underbrace{\mathcal{Y}(\bar{S})}_{\text{initial states for start symbol } \bar{S}})$$

$$= \mathrm{lfp}\ F$$

where $F(X) = \mathcal{Y}(\bar{S}) \cup \underbrace{\{x' \mid \exists x \in X : x \Rrightarrow x'\}}_{\mathrm{post}(\Rrightarrow) X}$

- fixpoint property preserved by abstraction (a result not specific to grammars).

The hierarchy of top-down semantics (*)

$$\begin{tikzcd}
\cdots \\
S^{\check{L}}[\![G]\!] & \text{protolanguage semantics} \\
S^{\check{\Sigma}}[\![G]\!] \arrow[u, "\alpha^{\check{L}}"] & \text{protosyntax tree semantics} \\
S^{\check{\delta}}[\![G]\!] \arrow[u, "\alpha^{\check{\Sigma}}"] & \text{protoderivation tree semantics} \\
S^{\check{D}}[\![G]\!] \arrow[u, "\alpha^{\check{\delta}}"] & \text{protoderivation semantics}
\end{tikzcd}$$

(*) Obviously no variables in terminal sentences !

ABSTRACTION OF TOP-DOWN
TO BOTTOM-UP SEMANTICS

Abstraction of the protoXXX top-down semantics into the XXX bottom-up semantics

$$\alpha(X) = \{ x \in X \mid x \text{ has no variables } \boxed{A} \text{ or } A \}$$

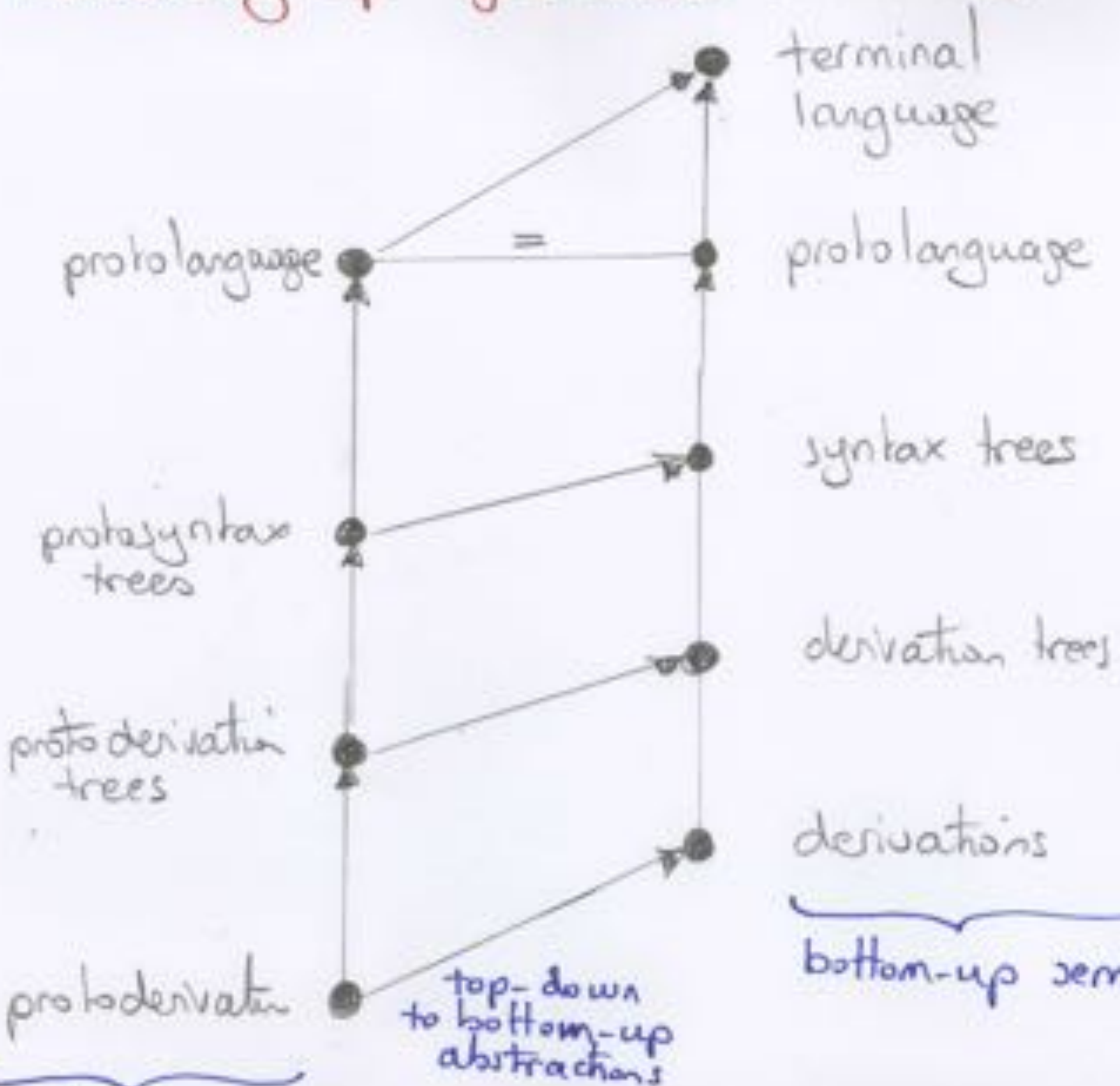
Example : protolanguage $\rightarrow$ terminal language

$$\alpha(X) = X \cap \text{terminals}^*$$

so we just record the finished derivations.

THE HIERARCHY OF GRAMMAR SEMANTICS

The hierarchy of grammar semantics



top-down semantics

bottom-up semantics

BOTTOM-UP GRAMMAR ANALYSIS

Bottom-up grammar analysis algorithms

- choose some bottom-up semantics $S = \mathrm{lfp}^{\subseteq} F$
- define an abstraction α into a finite domain
- design $F^\# = \alpha \circ F \circ \gamma$ such that $\alpha \circ F = F^\# \circ \alpha$
- it follows that $S^\# \stackrel{\Delta}{=} \alpha(S) = \mathrm{lfp}^{\subseteq} F^\#$
- the algorithm is just the iterative computation
  $X^0 = \bot, \ldots, X^{n+1} = F^\#(X^n)$ using chaotic iterations
  (as found in Reinhard's book!)
- To design $F^\#$, simplify $\alpha \circ F(X)$ into some expression $e(\alpha(X))$ and define $F^\#(X) \stackrel{\Delta}{=} e(X)$
  It follows that $F^\# = \alpha \circ F \circ \gamma$!

Example : nonterminal productivity

— Abstraction :

$$\dot{\alpha}^{\sharp} \triangleq \lambda L \cdot \lambda A \cdot \alpha^{\sharp}(L(A)),$$
$$\alpha^{\sharp} \triangleq \lambda \Sigma \cdot (\Sigma \neq \emptyset \; ? \; \mathrm{tt} : \mathrm{ff})$$

$$\langle \mathcal{N} \mapsto \wp(\mathcal{T}^*), \subseteq \rangle \xrightleftharpoons[\dot{\alpha}^{\sharp}]{\dot{\gamma}^{\sharp}} \langle \mathcal{N} \mapsto \mathbb{B}, \Longrightarrow \rangle .$$

— Non terminal productivity semantics :

- Definition $\quad \mathsf{S}^{\sharp}[\![\mathcal{G}]\!] \triangleq \dot{\alpha}^{\sharp}(\mathsf{S}^{\vec{\ell}}[\![\mathcal{G}]\!])$ $\quad$ abstraction of the bottom-up language semantics
- Abstraction theorem : $\quad \mathsf{S}^{\sharp}[\![\mathcal{G}]\!] = \mathsf{lfp}^{\Longrightarrow} \hat{\mathsf{F}}^{\sharp}[\![\mathcal{G}]\!]$

$$\hat{\mathsf{F}}^{\sharp}[\![\mathcal{G}]\!] \triangleq \lambda \rho \cdot \lambda A \cdot \bigvee_{A \to \sigma \in \mathcal{R}} \hat{\mathsf{F}}^{\sharp}[A \to \bullet\sigma]\rho$$
$$\hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet a\sigma'] \triangleq \lambda \rho \cdot \hat{\mathsf{F}}^{\sharp}[A \to \sigma a\bullet\sigma']$$
$$\hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet B\sigma'] \triangleq \lambda \rho \cdot \rho(B) \wedge \hat{\mathsf{F}}^{\sharp}[A \to \sigma B\bullet\sigma']\rho$$
$$\hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet] \triangleq \lambda \rho \cdot \mathrm{tt}$$

Calculational design

PROOF We calculate

$$
\begin{aligned}
& \dot{\alpha}^{\sharp} \circ \hat{\mathsf{F}}^{t}[\![\mathscr{G}]\!](\rho) && \text{⦃def. } \circ \text{⦄}\\
= \;& \dot{\alpha}^{\sharp}(\hat{\mathsf{F}}^{t}[\![\mathscr{G}]\!](\rho)) && \text{⦃def. } \hat{\mathsf{F}}^{t}[\![\mathscr{G}]\!]\text{⦄}\\
= \;& \dot{\alpha}^{\sharp}(\boldsymbol{\lambda} A \cdot \bigcup_{A \to \sigma \in \mathscr{G}} \hat{\mathsf{F}}^{t}[A \to \bullet\sigma]\rho) && \text{⦃def. } \dot{\alpha}^{\sharp}\text{⦄}\\
= \;& \boldsymbol{\lambda} A \cdot \alpha^{\sharp}(\bigcup_{A \to \sigma \in \mathscr{G}} \hat{\mathsf{F}}^{t}[A \to \bullet\sigma](\rho)) && \text{⦃}\alpha^{\sharp}\text{ preserves lubs⦄}\\
= \;& \boldsymbol{\lambda} A \cdot \bigvee_{A \to \sigma \in \mathscr{G}} \alpha^{\sharp}(\hat{\mathsf{F}}^{t}[A \to \bullet\sigma](\rho)) \\
= \;& \quad \text{⦃provided we can define } \hat{\mathsf{F}}^{\sharp} \text{ such that } \alpha^{\sharp} \circ \hat{\mathsf{F}}^{t}[A \to \bullet\sigma] = \hat{\mathsf{F}}^{\sharp}[A \to \bullet\sigma] \circ \dot{\alpha}^{\sharp}\text{⦄}\\
& \boldsymbol{\lambda} A \cdot \bigvee_{A \to \sigma \in \mathscr{G}} \hat{\mathsf{F}}^{\sharp}[A \to \bullet\sigma](\dot{\alpha}^{\sharp}(\rho)) && \text{⦃by defining } \hat{\mathsf{F}}^{\sharp}[\![\mathscr{G}]\!]\rho \triangleq \boldsymbol{\lambda} A \cdot \textstyle\bigvee_{A \to \sigma \in \mathscr{G}} \hat{\mathsf{F}}^{\sharp}[A \to \bullet\sigma]\rho\text{⦄}\\
= \;& \hat{\mathsf{F}}^{\sharp}[\![\mathscr{G}]\!](\dot{\alpha}^{\sharp}(\rho))
\end{aligned}
$$

It remains to define $\hat{\mathsf{F}}^{\sharp}$ such that $\alpha^{\sharp} \circ \hat{\mathsf{F}}^{t}[A \to \sigma\bullet\sigma'] = \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet\sigma'] \circ \dot{\alpha}^{\sharp}$. We proceed by structural induction on the length of σ' in $[A \to \sigma\bullet\sigma']$. We have the following cases

$$
\begin{aligned}
- \;& \alpha^{\sharp}(\hat{\mathsf{F}}^{t}[A \to \sigma\bullet a\sigma']\rho) && \text{⦃def. } \hat{\mathsf{F}}^{t}[A \to \sigma\bullet a\sigma']\text{⦄}\\
= \;& \alpha^{\sharp}(a\,\hat{\mathsf{F}}^{t}[A \to \sigma a\bullet\sigma']\rho) && \text{⦃def. } \alpha^{\sharp}\text{⦄}\\
= \;& \alpha^{\sharp}(\hat{\mathsf{F}}^{t}[A \to \sigma a\bullet\sigma'](\rho)) && \text{⦃by defining } \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet a\sigma'](\rho) \triangleq \boxplus\text{⦄}\\
= \;& \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet a\sigma'](\dot{\alpha}^{\sharp}(\rho))
\end{aligned}
$$

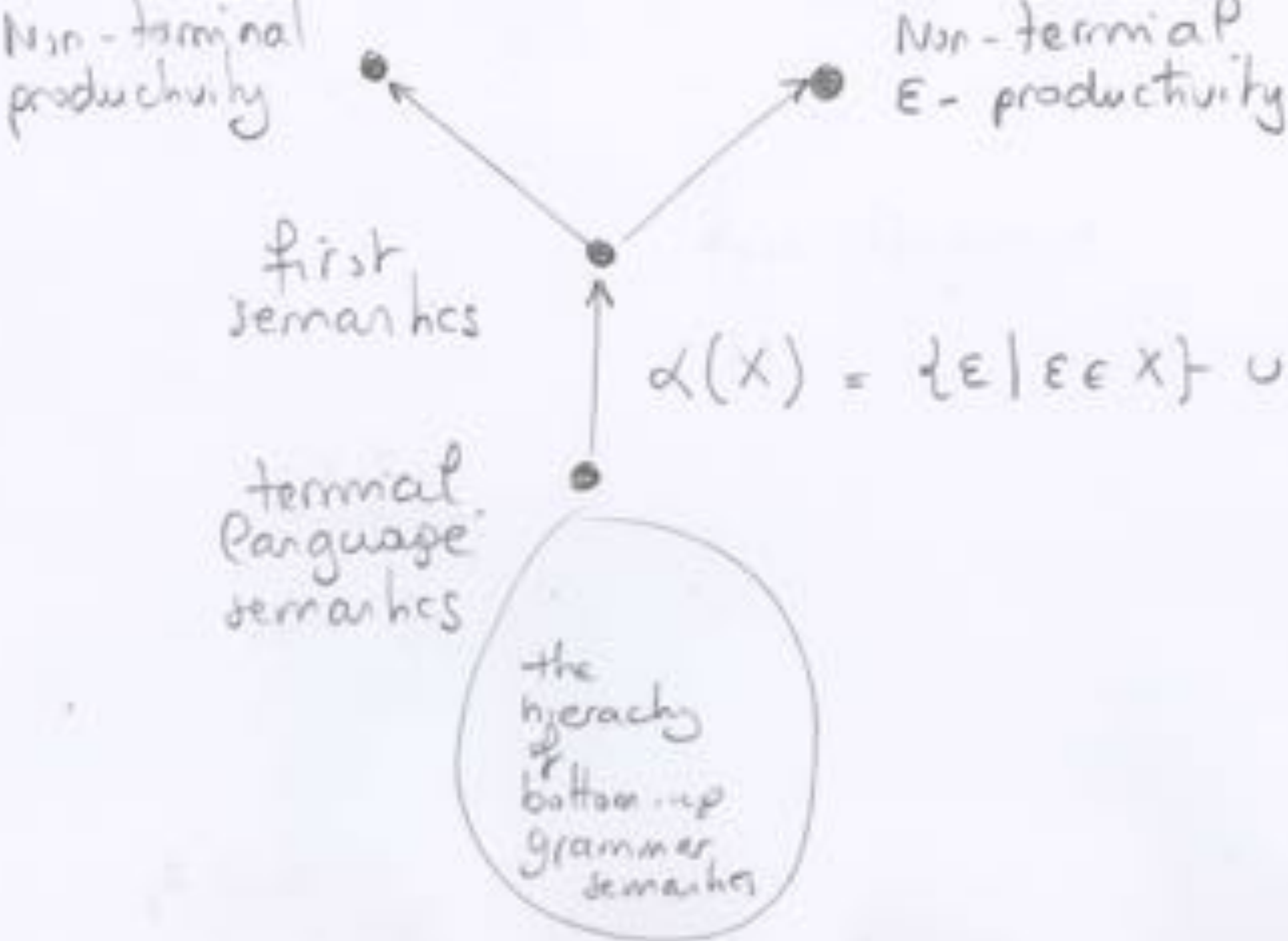
$$
\begin{aligned}
& \alpha^{\sharp}(\hat{\mathsf{F}}^{\ell}[A \to \sigma\bullet B\sigma']\rho) \\
= \;& \alpha^{\sharp}(\rho(B)\,\hat{\mathsf{F}}^{\ell}[A \to \sigma B\bullet\sigma']\rho) && \{\text{def. } \hat{\mathsf{F}}^{\ell}[A \to \sigma\bullet B\sigma']\} \\
= \;& \alpha^{\sharp}(\rho(B)) \wedge \alpha^{\sharp}(\hat{\mathsf{F}}^{\ell}[A \to \sigma B\bullet\sigma']\rho) && \{\text{def. concatenation and } \alpha^{\sharp}\} \\
= \;& \dot{\alpha}^{\sharp}(\rho)(B) \wedge \alpha^{\sharp}(\hat{\mathsf{F}}^{\ell}[A \to \sigma B\bullet\sigma']\rho) && \{\text{def. } \dot{\alpha}^{\sharp}\} \\
= \;& \dot{\alpha}^{\sharp}(\rho)(B) \wedge \hat{\mathsf{F}}^{\sharp}[A \to \sigma B\bullet\sigma'](\dot{\alpha}^{\sharp}(\rho)) && \{\text{ind. hyp.}\} \\
= \;& \quad \{\text{by defining } \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet B\sigma'](\rho) \triangleq \rho(B) \wedge \hat{\mathsf{F}}^{\sharp}[A \to \sigma B\bullet\sigma']\rho\} \\
& \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet B\sigma'](\dot{\alpha}^{\sharp}(\rho))
\end{aligned}
$$

$$
\begin{aligned}
& \alpha^{\sharp}(\hat{\mathsf{F}}^{\ell}[A \to \sigma\bullet]\rho) \\
= \;& \alpha^{\sharp}(\{\epsilon\}) && \{\text{def. } \hat{\mathsf{F}}^{\ell}[A \to \sigma\bullet]\} \\
= \;& \mathsf{tt} && \{\text{def. } \alpha^{\sharp}\} \\
= \;& \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet](\dot{\alpha}^{\sharp}(\rho)) && \{\text{by defining } \hat{\mathsf{F}}^{\sharp}[A \to \sigma\bullet]\rho \triangleq \mathsf{tt}\}
\end{aligned}
$$

We have shown the commutation property $\dot{\alpha}^{\sharp} \circ \hat{\mathsf{F}}^{\ell}[\mathcal{G}] = \hat{\mathsf{F}}^{\sharp}[\mathcal{G}] \circ \dot{\alpha}^{\sharp}$ and conclude by **Cor. 12**. ∎

- One can reasonably anticipate that this calculation is mechanizable
- otherwise use a proof assistant (e.g. COQ)

Hierarchy of bottom-up grammar analysis algorithms



$$\alpha(X) = \{\varepsilon \mid \varepsilon \in X\} \cup \{a \mid a\sigma \in X\}$$

Reinhard's bottom up abstract interpreter

$$\mathsf{S}^\ddagger[\![\mathcal{G}]\!] \in \mathscr{N} \mapsto L$$
$$\mathsf{S}^\ddagger[\![\mathcal{G}]\!] = \mathbf{lfp}^{\sqsubseteq}\, \vec{\mathsf{F}}^\ddagger[\![\mathcal{G}]\!]$$

where $\langle L, \sqsubseteq, \bot, \sqcup\rangle$ is a complete lattice and $\vec{\mathsf{F}}^\ddagger[\![\mathcal{G}]\!] \in (\mathscr{N} \mapsto L) \mapsto (\mathscr{N} \mapsto L)$ is a transformer defined in the form

$$\vec{\mathsf{F}}^\ddagger[\![\mathcal{G}]\!] \triangleq \boldsymbol{\lambda}\rho \cdot \boldsymbol{\lambda} A \cdot \bigsqcup_{A \to \sigma \in \mathscr{R}} A^\ddagger \sqcup \vec{\mathsf{F}}^\ddagger[A \to {\bullet}\sigma]\rho$$

$$\vec{\mathsf{F}}^\ddagger[A \to \sigma{\bullet}a\sigma'] \triangleq \boldsymbol{\lambda}\rho \cdot [A \to \sigma{\bullet}a\sigma']^\ddagger \cdot^\ddagger \vec{\mathsf{F}}^\ddagger[A \to \sigma a{\bullet}\sigma']\rho$$
$$\vec{\mathsf{F}}^\ddagger[A \to \sigma{\bullet}B\sigma'] \triangleq \boldsymbol{\lambda}\rho \cdot [A \to \sigma{\bullet}B\sigma']^\ddagger(\rho, B) \mathbin{\natural^\ddagger} \vec{\mathsf{F}}^\ddagger[A \to \sigma B{\bullet}\sigma']\rho$$
$$\vec{\mathsf{F}}^\ddagger[A \to \sigma{\bullet}] \triangleq \boldsymbol{\lambda}\rho \cdot [A \to \sigma{\bullet}]^\ddagger$$

Instances :

	Protolanguage	Language	First	ϵ–Productivity
L	$\wp(\mathscr{V}^*)$	$\wp(\mathscr{T}^*)$	$\wp(\mathscr{T} \cup \{\epsilon\})$	$\mathbb{B}$
$\sqsubseteq$	$\subseteq$	$\subseteq$	$\subseteq$	$\Longrightarrow$
$\bot$	$\emptyset$	$\emptyset$	$\emptyset$	ff
$\sqcup$	$\cup$	$\cup$	$\cup$	$\vee$
$A^\ddagger$	$\{A\}$	$\emptyset$	$\emptyset$	ff
$[A \to \sigma{\bullet}a\sigma']^\ddagger$	$\{a\}$	$\{a\}$	$\{a\}$	ff
$\cdot^\ddagger$	$\cdot^2$	$\cdot$	$\oplus^1$	$\wedge$
$[A \to \sigma{\bullet}B\sigma']^\ddagger(\rho, B)$	$\{B\} \cup \rho(B)$	$\rho(B)$	$\rho(B)$	$\rho(B)$
$\natural^\ddagger$	$\cdot$	$\cdot$	$\oplus^1$	$\wedge$
$[A \to \sigma{\bullet}]^\ddagger$	$\{\epsilon\}$	$\{\epsilon\}$	$\{\epsilon\}$	tt

TOP-DOWN GRAMMAR ANALYSIS

~~Bottom-up~~ Top-down grammar analysis algorithms

- Choose some ~~bottom-up~~ top-down semantics $S = \mathrm{lfp}^{\subseteq} F$
- define an abstraction α into a finite domain
- design $F^{\#} = \alpha \circ F \circ \gamma$ such that $\alpha \circ F = F^{\#}$
- it follows that $S^{\#} \triangleq \alpha(S) = \mathrm{lfp}^{\sqsubseteq^{\#}} F^{\#}$
- the algorithm is just the iterative computa $X^0 = \bot, \dots, X^{n+1} = F^{\#}(X^n)$ using chaotic iterat (as found in Reinhard's book!)
- To design $F^{\#}$, simplify $\alpha \circ F(X)$ into some expression $e(\alpha(X))$ and define $F^{\#}(X) = e$

It follows that $F^{\#} = \alpha \circ F \circ \gamma$!

Example : nonterminal accessibility

- Abstraction :
  - $\alpha^{\overline{S}}(f) = f(\overline{S})$
  - $\alpha^{a} \triangleq \lambda \Sigma \cdot \lambda A \cdot (\exists \sigma, \sigma' \in \mathcal{V}^{*} : \sigma A \sigma' \in \Sigma \; ? \; \mathrm{tt} : \mathrm{ff})$
- Nonterminal accessibility semantics
  - Definition
    $$\mathsf{S}^{a}[\![\mathcal{G}]\!] \triangleq \alpha^{a}(\mathsf{S}^{\mathcal{L}}[\![\mathcal{G}]\!](\overline{S})) = \alpha^{a} \circ \alpha^{\overline{S}}(\mathsf{S}^{\mathcal{L}}[\![\mathcal{G}]\!])$$
  - Abstraction theorems :
    $$\alpha^{\overline{S}}(\mathsf{S}^{\mathcal{L}}[\![\mathcal{G}]\!]) = \mathit{lfp}^{\subseteq} \lambda X \cdot \{\overline{S}\} \cup \mathrm{post}[\longmapsto_{\mathcal{G}}]X$$
    $$\begin{aligned} \mathsf{S}^{a}[\![\mathcal{G}]\!] &= \mathit{lfp}^{\subseteq} \mathsf{F}^{a}[\![\mathcal{G}]\!] \\ \text{where } \mathsf{F}^{a}[\![\mathcal{G}]\!] &\triangleq \lambda \phi \cdot \lambda A \cdot (A = \overline{S}) \vee \bigvee_{B \to \sigma A \sigma' \in \mathcal{R}} \phi(B) \end{aligned}$$

Hierarchy of top-down grammar analysis algorithms

Again, Reinhard's top-down grammar abstract interpreter.

TOP-DOWN PARSING

Nonrecursive predictive parser

Abstraction:

- Abstract maximal derivations into their prefixes

$$S^{\vec{\varpi}}[\![G]\!] = \mathrm{lfp}^{\subseteq} F^{\vec{\varpi}}[\![G]\!] \quad \text{where} \quad F^{\vec{\varpi}}[\![G]\!] \triangleq \lambda X \cdot \{\vdash\} \cup X_1 \longrightarrow$$

- Abstract these prefixes into items $\langle i, \varpi \rangle$

where the prefix is

$$\underbrace{\vdash \longrightarrow \cdots \longrightarrow}_{\sigma_1 \ldots \sigma_i} \varpi$$

as follows:

$$\alpha^{LL} \triangleq \lambda \overline{S} \cdot \lambda \sigma \cdot \lambda X \cdot \{\langle i, \varpi \rangle \mid \exists \theta = \varpi_0 \xrightarrow{t_0} \varpi_1 \ldots \varpi_{m-1} \xrightarrow{t_{m-1}} \varpi_m \in X.\overline{S} : i \in [0, |\sigma|] \wedge \alpha^{\tau}(\theta) = \sigma_1 \ldots \sigma_i \wedge \varpi = \varpi_m\}.$$

$$\begin{aligned}
\alpha^{\tau}(\theta_1 \xrightarrow{(A} \theta_2) &\triangleq \alpha^{\tau}(\theta_1)\alpha^{\tau}(\theta_2) & \\
\alpha^{\tau}(\theta_1 \xrightarrow{A)} \theta_2) &\triangleq \alpha^{\tau}(\theta_1)\alpha^{\tau}(\theta_2) & \\
\alpha^{\tau}(\theta_1 \xrightarrow{a} \theta_2) &\triangleq \alpha^{\tau}(\theta_1) a \alpha^{\tau}(\theta_2), & a \in \mathcal{T} \\
\alpha^{\tau}(\varpi) &\triangleq \epsilon, & \varpi \in \mathcal{S} \\
\alpha^{\tau}(\vdash) &\triangleq \epsilon & \\
\alpha^{\tau}(\dashv) &\triangleq \epsilon. &
\end{aligned}$$

- Correctness of the parser :

$$\sigma \in \mathsf{S}^t[\![\mathcal{G}]\!](\overline{S}) \iff \langle |\sigma|, \dashv \rangle \in \alpha^{LL}(\overline{S})(\sigma)(\mathsf{S}^{\vec{s}}[\![\mathcal{G}]\!]) \,.$$

- Nonrecursive predictive parsing semantics :

$$\alpha^{LL}(\overline{S})(\sigma)(\mathsf{S}^{\vec{s}}[\![\mathcal{G}]\!]) = \mathrm{lfp}^{\subseteq}\, \mathsf{F}^{LL}[\![\mathcal{G}]\!](\sigma)$$

where

$$\begin{aligned}
&\mathsf{F}^{LL}[\![\mathcal{G}]\!](\sigma) \in \wp([0, |\sigma|] \times \mathcal{S}) \mapsto \wp([0, |\sigma|] \times \mathcal{S})\\
&\mathsf{F}^{LL}[\![\mathcal{G}]\!](\sigma) = \lambda X \cdot \{\langle 0, \vdash \rangle\} \cup \{\langle 0, \dashv[\overline{S} \to \bullet\eta] \rangle \mid \langle 0, \vdash \rangle \in X \wedge \overline{S} \to \eta \in \mathscr{R}\} \cup\\
&\qquad \{\langle i+1, \varpi[A \to \eta a \bullet \eta'] \rangle \mid \langle i, \varpi[A \to \eta \bullet a \eta'] \rangle \in X \wedge a = \sigma_{i+1}\} \cup\\
&\qquad \{\langle i, \varpi[A \to \eta B \bullet \eta'][B \to \bullet\varsigma] \rangle \mid \langle i, \varpi[A \to \eta \bullet B \eta'] \rangle \in X \wedge B \to \varsigma \in \mathscr{R}\}\\
&\qquad \cup \{\langle i, \varpi \rangle \mid \langle i, \varpi[A \to \eta \bullet] \rangle \in X\} \,.
\end{aligned}$$

- Parsing algorithm : reachable states of :

the transition system $\langle [0, |\sigma|] \times \mathcal{S}, \xrightarrow{LL} \rangle$ where

$$\begin{array}{rcll}
\langle 0, \vdash \rangle & \xrightarrow{LL} & \langle 0, \dashv[\overline{S} \to \bullet\eta] \rangle & \overline{S} \to \eta \in \mathscr{R}\\
\langle i, \varpi[A \to \eta \bullet \sigma_{i+1} \eta'] \rangle & \xrightarrow{LL} & \langle i+1, \varpi[A \to \eta \sigma_{i+1} \bullet \eta'] \rangle & \\
\langle i, \varpi[A \to \eta \bullet B \eta'] \rangle & \xrightarrow{LL} & \langle i, \varpi[A \to \eta B \bullet \eta'][B \to \bullet\varsigma] \rangle & B \to \varsigma \in \mathscr{R}\\
\langle i, \varpi[A \to \eta \bullet] \rangle & \xrightarrow{LL} & \langle i, \varpi \rangle &
\end{array}$$

- Examples: $A \to A \mid a$
  - input $\sigma = a$

$$
\begin{array}{ll}
 & \langle 0, \vdash \rangle \\
\xrightarrow{LL} & \langle 0, \dashv [A \to \bullet a] \rangle \\
\xrightarrow{LL} & \langle 1, \dashv [A \to a \bullet] \rangle \\
\xrightarrow{LL} & \langle 1, \dashv \rangle
\end{array}
$$

  - input $\sigma = b$: loops!

$$
\begin{array}{ll}
 & \langle 0, \vdash \rangle \\
\xrightarrow{LL} & \langle 0, \dashv [A \to . A] \rangle \\
\xrightarrow{LL} & \langle 0, \dashv [A \to . A][A \to . A] \rangle \\
\xrightarrow{LL} & \langle 0, \dashv [A \to . A][A \to . A][A \to . A] \rangle \\
\xrightarrow{LL} & \cdots
\end{array}
$$

- Termination:

Theorem 107 *The nonrecursive predictive parsing algorithm for a grammar $\mathcal{G} = \langle \mathcal{T}, \mathcal{N}, \overline{S}, \mathcal{R} \rangle$ terminates (i.e. the transition relation $\xrightarrow{LL}$ has no infinite trace for all input sentences $\sigma \in \mathcal{T}^*$) if and only if the grammar $\mathcal{G}$ has no left recursion (that is $\exists A \in \mathcal{N} : \exists \eta \in \mathcal{V}^* : A \overset{+}{\Longrightarrow}_{\mathcal{G}} A\eta$).*

– Adding a lookahead :

The first symbol of the right context should be σ_{i+1} (or $\dashv$ if $i = n$) :

$$\begin{aligned}\alpha^{LL(1)} \triangleq \boldsymbol{\lambda} \overline{S} \cdot \boldsymbol{\lambda} \sigma \cdot \boldsymbol{\lambda} X \cdot \{\langle i, \varpi\rangle \mid\ & \exists \theta = \varpi_0 \xrightarrow{\ell_0} \varpi_1 \ldots \varpi_{m-1} \xrightarrow{\ell_{m-1}} \varpi_m \in X.\overline{S} : \\ & i \in [0, |\sigma|] \wedge \alpha^{\tau}(\theta) = \sigma_1 \ldots \sigma_i \wedge \varpi = \varpi_m \wedge \forall \varpi' \in \mathcal{S}, A \to \eta\eta' \in \mathscr{R} : \\ & (\varpi = \varpi'[A \to \eta \bullet \eta'] \wedge i \leqslant |\sigma|) \Longrightarrow (\sigma_{i+1} \in \mathsf{S}^{\top}[\![\mathcal{G}]\!][A \to \eta \bullet \eta'])\} .\end{aligned}$$

where $\mathsf{S}^{\mp}[\![G]\!]$ is the extension of the first semantics $\mathsf{S}^{\mp}[\![G]\!]$ to protosentences :

$$\mathsf{S}^{\top}[\![\mathcal{G}]\!] = \boldsymbol{\lambda} \eta \cdot \{a \in \mathscr{T} \mid \exists \sigma \in \mathscr{T}^* : \eta \overset{\bullet}{\Longrightarrow}_{\mathcal{G}} a\sigma\} \cup \{\epsilon \mid \eta \overset{\bullet}{\Longrightarrow}_{\mathcal{G}} \epsilon\}$$

(can be expressed in fixpoint form)

BOTTOM-UP PARSING

Approach

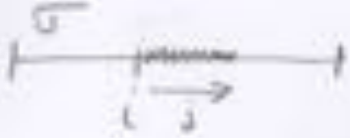
As was the case for top-down parsing (e.g. Earley, TCS 2003), the bottom-up parsers are complete abstract interpretations of the bottom-up semantics.

In general non deterministic, deterministic under specific conditions

e.g. non deterministic → Tomita algorithm
deterministic → Knuth LR(K) algorithm

The Cocke-Younger-Kasami (CYK) Algorithm

- Abstract domain for input σ:



$$\hat{\mathsf{D}}^{CYK} \triangleq \lambda\sigma \cdot \{\langle i, j\rangle \mid i \in [1, |\sigma|+1] \wedge j \in [0, |\sigma|] \wedge i + j \leq |\sigma| + 1\}$$

- Abstraction:

$$\alpha^{CYK} \triangleq \lambda\sigma \cdot \lambda X \cdot \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} \in X\}$$

$$\langle \wp(\mathscr{T}^*), \subseteq\rangle \underset{\alpha^{CYK}(\sigma)}{\overset{\gamma^{CYK}(\sigma)}{\leftrightarrows}} \langle \wp(\hat{\mathsf{D}}^{CYK}(\sigma)), \subseteq\rangle$$

$$\alpha^{CYK} \triangleq \lambda\sigma \cdot \lambda X \cdot \lambda A \cdot \alpha^{CYK}(X(A))$$

$$\langle \mathscr{N} \mapsto \wp(\mathscr{T}^*), \subseteq\rangle \underset{\alpha^{CYK}(\sigma)}{\overset{\gamma^{CYK}(\sigma)}{\leftrightarrows}} \langle \mathscr{N} \mapsto \wp(\hat{\mathsf{D}}^{CYK}(\sigma)), \subseteq\rangle$$

- Correctness of the parser:

$$\sigma \in \mathsf{S}^{\ell}[\![\mathcal{G}]\!](\overline{S}) \iff \langle 1, |\sigma|\rangle \in \alpha^{CYK}(\sigma)(\mathsf{S}^{\ell}[\![\mathcal{G}]\!])(\overline{S})$$

– The CYK parser :

$$\alpha^{CYK}(\sigma)(\mathsf{S}^t[\![\mathcal{G}]\!])(\overline{S}) \;=\; \mathit{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{CYK}[\![\mathcal{G}]\!](\sigma)$$

where

$$\begin{aligned}
\hat{\mathsf{F}}^{CYK}[\![\mathcal{G}]\!] &\in \wp(\hat{\mathsf{D}}^{CYK}) \mapsto \wp(\hat{\mathsf{D}}^{CYK}) \\
\hat{\mathsf{F}}^{CYK}[\![\mathcal{G}]\!] &\triangleq \boldsymbol{\lambda}\rho \cdot \boldsymbol{\lambda} A \cdot \bigcup_{A \to \sigma \in \mathcal{R}} \hat{\mathsf{F}}^{CYK}[A \to {\bullet}\sigma]\rho \\
\hat{\mathsf{F}}^{CYK}[A \to \sigma{\bullet}a\sigma'] &\triangleq \boldsymbol{\lambda}\rho \cdot \{\langle i,\, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i = a \,\wedge \\
&\qquad\qquad \langle i+1,\, j-1\rangle \in \hat{\mathsf{F}}^{CYK}[A \to \sigma a{\bullet}\sigma']\rho\} \\
\hat{\mathsf{F}}^{CYK}[A \to \sigma{\bullet}B\sigma'] &\triangleq \boldsymbol{\lambda}\rho \cdot \{\langle i,\, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \exists k : 0 \leqslant k \leqslant j : \langle i,\, k\rangle \in \rho(B) \\
&\qquad\qquad \wedge \langle i+k,\, j-k\rangle \in \hat{\mathsf{F}}^{CYK}[A \to \sigma B{\bullet}\sigma']\rho\} \\
\hat{\mathsf{F}}^{CYK}[A \to \sigma{\bullet}] &\triangleq \boldsymbol{\lambda}\rho \cdot \{\langle i,\, 0\rangle \mid 1 \leqslant i \leqslant |\sigma|\}
\end{aligned}$$

□

The calculational design of the CYK parser by abstract interpretation :

Proof We apply Cor. 12.

[illegible]

We proceed by induction on the length $|\sigma'|$ of σ', with three cases.

[illegible]

Because the abstract domain [illegible] is finite, the iterative computation of [illegible] terminates whence by Th. 201 and Th. 200 so does the CYK parsing algorithm. The CYK dynamic programming algorithm organizes the computation of the pairs [illegible] in order to avoid repetition of work already done.

- You can only see that it is not so long !
- Surely mechanizable or checkable by a proof assistant

CONCLUSION

THANKS TO REINHARD on abstract interpretation

- For being among the first to understand
- For extending (a.o. to grammars)
- For promoting (see the A.I. chapter in his compilation book)

...

and, most importantly, for a long friendship (including Margaret et les filles).

THE END, THANK YOU FOR YOUR ATTENTION !

Happy birth year for Reinhard !