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

Bottom-up grammar flow analysis (from Reinard'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** $F_p\uparrow : D\uparrow^{n_p} \rightarrow D\uparrow$ pour chaque production $p \in P$;
- une **fonction de combinaison** $\nabla\uparrow : 2^{D\uparrow} \rightarrow D\uparrow$.

Abstract domain

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

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

□

System of abstract fixpoint equations

Exemple 8.2.12 (Productivité des non-terminaux)

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

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

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

□

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** $F_{p,i}\downarrow : D\downarrow \rightarrow D\downarrow$, $1 \leq i \leq n_p$, pour chaque production $p \in P$;
- une **fonction de combinaison** $\nabla\downarrow : 2^{D\downarrow} \rightarrow D\downarrow$;
- 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 :

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

□

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 :

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

□

Instantiation on an example (accessible non-terminals)

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

- We can define an operational semantics of grammars ($\simeq$ pushdown automata)
- We can abstract this semantics
  - Bottom-up $X \to \underline{X_1 \dots X_n}$, 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-down 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

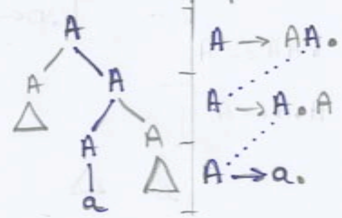
Transition system

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)

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

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 :

$$\vdash \xrightarrow{(A} \dashv [A \to .a] \xrightarrow{a} \dashv [A \to a.] \xrightarrow{AD} \dashv$$

- Ex : derivation for sentence aa :

$$
\begin{array}{c}
\vdash \xrightarrow{(A} \dashv [A \to .AA] \xrightarrow{(A} \dashv [A \to A.A][A \to .a] \xrightarrow{a} \dashv [A \to A.A][A \to a.] \xrightarrow{A)} \\
\dashv [A \to A.A] \xrightarrow{(A} \dashv [A \to AA.][A \to .a] \xrightarrow{a} \dashv [A \to AA.][A \to a.] \xrightarrow{A)} \dashv \\
[A \to AA.] \xrightarrow{A)} \dashv .
\end{array}
$$

(*) 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

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

- Here is the bottom-up derivation semantics :

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

the derivations defined by the operational semantics

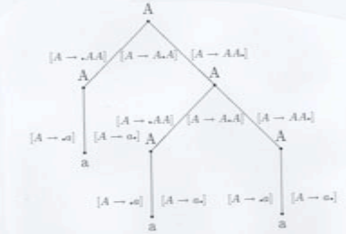
denotational semantics

the fixpoint operator.

$$\begin{aligned}
\hat{\mathsf{F}}^{d}[\![\mathcal{G}]\!] &\triangleq \lambda T \cdot \bigcup_{A\to\sigma\in\mathscr{R}} \vdash \xrightarrow{(\!|A} \hat{\mathsf{F}}^{d}[A \to \bullet\sigma]T \xrightarrow{A|\!)} \dashv \\
\hat{\mathsf{F}}^{d}[A \to \sigma\bullet a\sigma'] &\triangleq \lambda T \cdot (\dashv[A \to \sigma\bullet a\sigma']) \xrightarrow{a} \hat{\mathsf{F}}^{d}[A \to \sigma a\bullet\sigma']T \\
\hat{\mathsf{F}}^{d}[A \to \sigma\bullet B\sigma'] &\triangleq \lambda T \cdot (\langle \dashv[A \to \sigma\bullet B\sigma'], \dashv[A \to \sigma B\bullet\sigma']\rangle \uparrow T.B) \mathbin{;} \\
&\qquad \hat{\mathsf{F}}^{d}[A \to \sigma B\bullet\sigma']T \\
\hat{\mathsf{F}}^{d}[A \to \sigma\bullet] &\triangleq \lambda T \cdot (\dashv[A \to \sigma\bullet]) .
\end{aligned}$$

Abstraction of derivations to derivation trees

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



abstract (derivation tree)

$\equiv$

$(\!|A[A \to \bullet AA](\!|A[A \to \bullet a]a[A \to a\bullet]A|\!)[A \to A\bullet A](\!|A[A \to \bullet AA](\!|A[A \to \bullet a]a[A \to a\bullet]A|\!)[A \to A\bullet A](\!|A[A \to \bullet a]a[A \to a\bullet]A|\!)[A \to AA\bullet]A|\!)[A \to AA\bullet]A|\!)$

parenthesized representation

$\alpha^{\hat{\delta}}$ (*)

$$\begin{aligned}
&\vdash \xrightarrow{(\!|A} \dashv [A \to \bullet AA] \xrightarrow{(\!|A} \dashv [A \to A\bullet A]\, [A \to \bullet a] \xrightarrow{a} \dashv [A \to A\bullet A]\, [A \to a\bullet] \\
&\xrightarrow{A|\!)} \dashv [A \to A\bullet A] \xrightarrow{(\!|A} \dashv [A \to AA\bullet]\, [A \to \bullet AA] \xrightarrow{(\!|A} \dashv [A \to AA\bullet]\, [A \to A\bullet A] \\
&[A \to \bullet a] \xrightarrow{a} \dashv [A \to AA\bullet]\, [A \to A\bullet A]\, [A \to a\bullet] \xrightarrow{A|\!)} \dashv [A \to AA\bullet]\, [A \to A\bullet A] \\
&\xrightarrow{(\!|A} \dashv [A \to AA\bullet]\, [A \to AA\bullet]\, [A \to \bullet a] \xrightarrow{a} \dashv [A \to AA\bullet]\, [A \to AA\bullet]\, [A \to a\bullet] \\
&\xrightarrow{A|\!)} \dashv [A \to AA\bullet]\, [A \to AA\bullet] \xrightarrow{A|\!)} \dashv [A \to AA\bullet] \xrightarrow{A|\!)} \dashv .
\end{aligned}$$

concrete derivation (for aaa)

(*) essentially get rid of $\rightarrow$ and abstract stacks by their top

- Fixpoint derivation tree semantics

- $\alpha \circ F^{\#} = F \circ \alpha \implies \alpha(\mathit{lfp}\, F) = \mathit{lfp}\, F^{\#}$
- $F^{\#} = \gamma \circ F \circ \alpha$

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

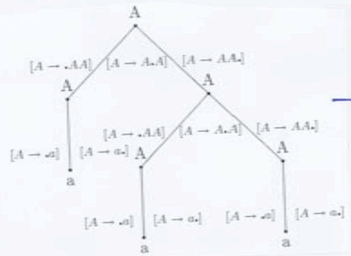
Definition : $\mathsf{S}^{\hat{\delta}}[\![\mathcal{G}]\!] \triangleq \alpha^{\hat{\delta}}(\mathsf{S}^{d}[\![\mathcal{G}]\!])$.

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

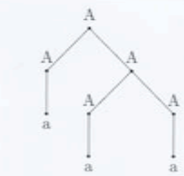
$$\begin{aligned}
\hat{\mathsf{F}}^{\hat{\delta}}[\![\mathcal{G}]\!] &\triangleq \lambda D \cdot \bigcup_{A\to\sigma\in\mathscr{R}} (\!|A\, \hat{\mathsf{F}}^{\hat{\delta}}[A \to \bullet\sigma]D\; A|\!) \\
\hat{\mathsf{F}}^{\hat{\delta}}[A \to \sigma\bullet a\sigma'] &\triangleq \lambda D \cdot [A \to \sigma\bullet a\sigma']\; a\; \hat{\mathsf{F}}^{\hat{\delta}}[A \to \sigma a\bullet\sigma']D \\
\hat{\mathsf{F}}^{\hat{\delta}}[A \to \sigma\bullet B\sigma'] &\triangleq \lambda D \cdot [A \to \sigma\bullet B\sigma']\; D.B\; \hat{\mathsf{F}}^{\hat{\delta}}[A \to \sigma B\bullet\sigma']D \\
\hat{\mathsf{F}}^{\hat{\delta}}[A \to \sigma\bullet] &\triangleq \lambda D \cdot [A \to \sigma\bullet] .
\end{aligned}$$

Syntax tree abstraction and bottom-up semantics

- Abstraction



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



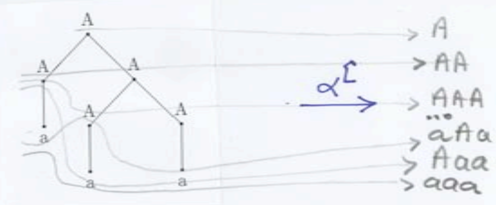
representation : $(\!|A(\!|AaA|\!)(\!|A(\!|AaA|\!)(\!|AaA|\!)A|\!)A|\!)$

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

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

Protolanguage abstraction & bottom-up semantics

– Abstraction :



– Fixpoint semantics :

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

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

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

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

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

The hierarchy of bottom-up grammar semantics

...

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

$\uparrow \alpha^{\ell}$

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

$\uparrow \alpha^{\hat{L}}$

$\mathsf{S}^{\hat{s}}[\![G]\!] = \mathsf{lfp}^{\subseteq}\ \hat{\mathsf{F}}^{\hat{s}}[\![G]\!]$ — syntax tree

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

$\mathsf{S}^{\hat{\delta}}[\![G]\!] = \mathsf{lfp}^{\subseteq}\ \hat{\mathsf{F}}^{\hat{\delta}}[\![G]\!]$ — derivation trees

$\uparrow \alpha^{\hat{\delta}}$

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

TOP-DOWN SEMANTICS OF GRAMMARS

Generalize the protolanguage derivation

$\Rightarrow$ and $\underbrace{\mathsf{post}(\overset{*}{\Longrightarrow})}_{\text{all transitive derivations from axiom}}(\{S\})$ ← 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 $A \to AA$.

$\boxed{D}\Rightarrow_G$

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

$\boxed{D}\Rightarrow_G$

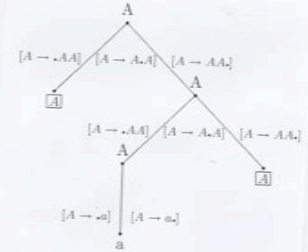
$\vdash \xrightarrow{(A} \dashv[A \to .AA] \xrightarrow{\boxed{A}} \dashv[A \to A.A] \xrightarrow{(A} \dashv[A \to AA.][A \to .a] \xrightarrow{a} \dashv[A \to AA.][A \to a.] \xrightarrow{A)} \dashv[A \to AA.] \xrightarrow{A)} \dashv$

$\boxed{D}\Rightarrow_G$

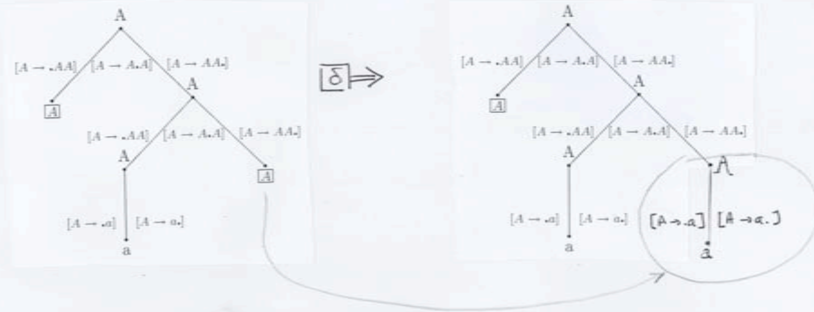
$\vdash \xrightarrow{(A} \dashv[A \to .AA] \xrightarrow{(A} \dashv[A \to A.A][A \to .a] \xrightarrow{a} \dashv[A \to A.A][A \to a.] \xrightarrow{A)} \dashv[A \to A.A] \xrightarrow{(A} \dashv[A \to AA.][A \to .a] \xrightarrow{a} \dashv[A \to AA.][A \to a.] \xrightarrow{A)} \dashv[A \to AA.] \xrightarrow{A)}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

Abstraction of protoderivations into protoderivation trees

- Protoderivation tree :

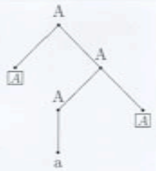


- Example of derivation $\boxed{\delta}\Rightarrow$:



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

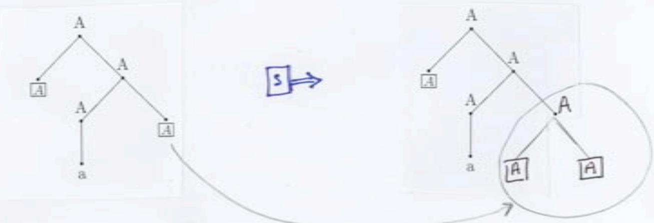
- Protosyntax tree :



Representation :

$(A\boxed{A}(A(AaA)\boxed{A}A)A)$

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



$A \to AA$

Abstraction of protosyntax trees into protosentences

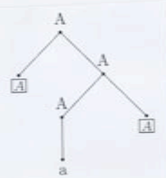
- Protosentences $(A \to AA \mid a)$

A   Aa   AaA   aaa ...   A ou $\boxed{A}$ variable

- Protosentence derivation (the classical notion)

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

- Example of abstraction :



$\xrightarrow{\alpha^{\check{L}}}$ A a A (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 $\Longmapsto$ (for protoderivations, protoderivation trees, proto syntax trees, protosentences)
- The semantics is

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

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

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

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

The hierarchy of top-down semantics (*)

$\check{S}^{L}[\![G]\!]$ protolanguage semantics

$\uparrow \alpha^{\check{L}}$

$\check{S}^{s}[\![G]\!]$ proto syntax tree semantics

$\uparrow \alpha^{\check{s}}$

$\check{S}^{\delta}[\![G]\!]$ protoderivation tree semantics

$\uparrow \alpha^{\check{\delta}}$

$\check{S}^{D}[\![G]\!]$ protoderivation semantics

(*) 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

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

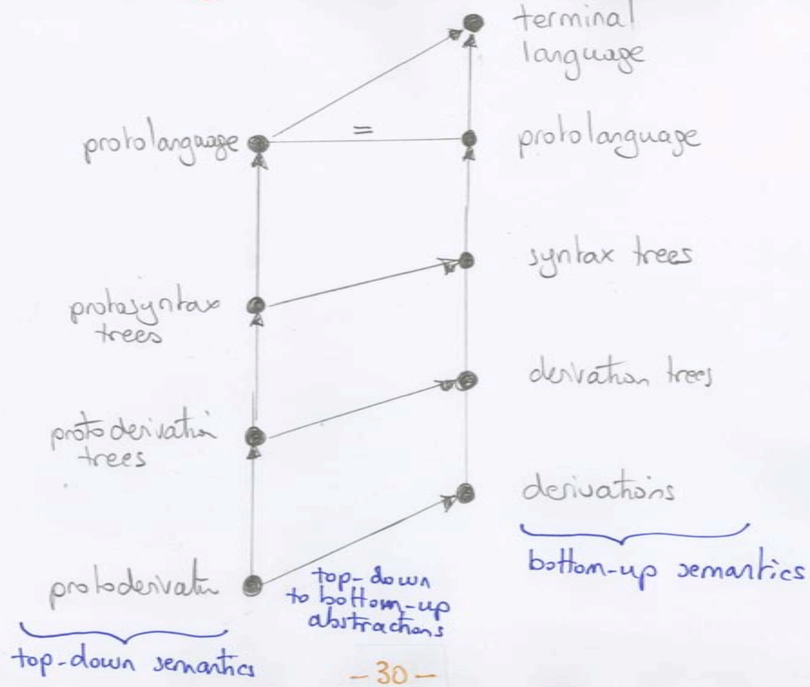
Example: protolanguage $\longrightarrow$ terminal language

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

so we just record the finished derivations.

THE HIERARCHY OF GRAMMAR SEMANTICS

The hierarchy of grammar semantics



BOTTOM-UP GRAMMAR ANALYSIS

Bottom-up grammar analysis algorithms

- Choose some bottom-up semantics $S = \mathrm{lfp}_{\perp}^{\subseteq} F$
- 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 $S^\# \triangleq \alpha(S) = \mathrm{lfp}_{\perp}^{\sqsubseteq} F^\#$
- the algorithm is just the iterative computation $X^0 = \perp, \ldots, X^{n+1} = F^\#(X^n)$ 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 $F^\#(X) \triangleq e(X)$. It follows that $F^\# = \alpha \circ F \circ \gamma$!

Example : nonterminal productivity

— Abstraction :

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

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

— Non terminal productivity semantics :

- Definition $\mathsf{S}^{\circledast}[\![\mathcal{G}]\!] \triangleq \dot{\alpha}^{\circledast}(\mathsf{S}^{\ell}[\![\mathcal{G}]\!])$ abstraction of the bottom-up language semantics
- Abstraction theorem ! $\mathsf{S}^{\circledast}[\![\mathcal{G}]\!] = \mathit{lfp}^{\dot{\Rightarrow}}\, \hat{\mathsf{F}}^{\circledast}[\![\mathcal{G}]\!]$

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

Calculational design

Proof We calculate

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

It remains to define $\hat{\mathsf{F}}^{\circledast}$ such that $\alpha^{\circledast} \circ \hat{\mathsf{F}}^{\ell}[A \to \sigma{\bullet}\sigma'] = \hat{\mathsf{F}}^{\circledast}[A \to \sigma{\bullet}\sigma'] \circ \dot{\alpha}^{\circledast}$. We proceed by structural induction on the length of σ' in $[A \to \sigma{\bullet}\sigma']$. We have the following cases

$$\begin{aligned}
— \;& \alpha^{\circledast}(\hat{\mathsf{F}}^{\ell}[A \to \sigma{\bullet}a\sigma']\rho) && \text{(def. } \hat{\mathsf{F}}^{\ell}[A \to \sigma{\bullet}a\sigma']\text{)} \\
= \;& \alpha^{\circledast}(a\, \hat{\mathsf{F}}^{\ell}[A \to \sigma a{\bullet}\sigma']\rho) && \text{(def. } \alpha^{\circledast}\text{)} \\
= \;& \alpha^{\circledast}(\hat{\mathsf{F}}^{\ell}[A \to \sigma a{\bullet}\sigma'](\rho)) && \text{(by defining } \hat{\mathsf{F}}^{\circledast}[A \to \sigma{\bullet}a\sigma'](\rho) \triangleq \mathrm{ff}\text{)} \\
= \;& \hat{\mathsf{F}}^{\circledast}[A \to \sigma{\bullet}a\sigma'](\dot{\alpha}^{\circledast}(\rho))
\end{aligned}$$

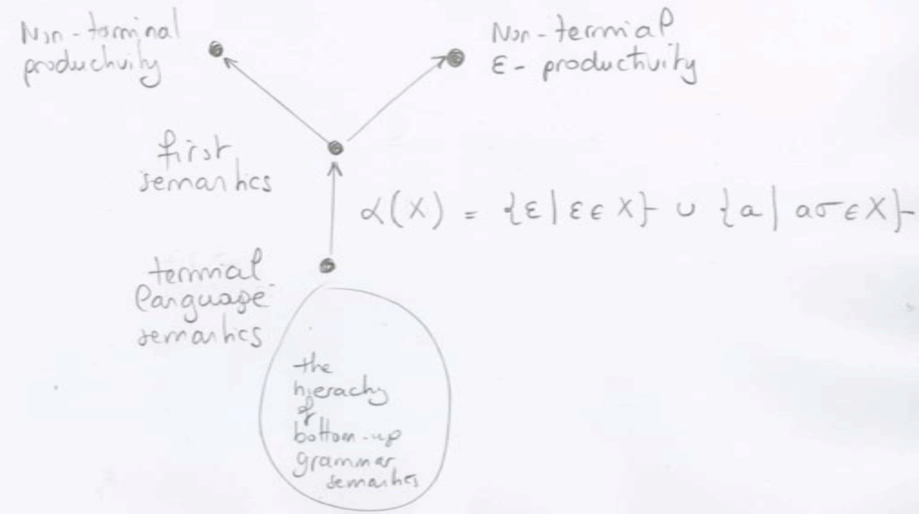
$$\begin{aligned}
— \;& \alpha^{\circledast}(\hat{\mathsf{F}}^{\ell}[A \to \sigma{\bullet}B\sigma']\rho) \\
= \;& \alpha^{\circledast}(\rho(B)\, \hat{\mathsf{F}}^{\ell}[A \to \sigma B{\bullet}\sigma']\rho) && \text{(def. } \hat{\mathsf{F}}^{\ell}[A \to \sigma{\bullet}B\sigma']\text{)} \\
= \;& \alpha^{\circledast}(\rho(B))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

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

We have shown the commutation property $\dot{\alpha}^{\circledast} \circ \hat{\mathsf{F}}^{\ell}[\![\mathcal{G}]\!] = \hat{\mathsf{F}}^{\circledast}[\![\mathcal{G}]\!] \circ \dot{\alpha}^{\circledast}$ 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



Reinhard's bottom up abstract interpreter

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

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

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

$$\hat{\mathsf{F}}^\sharp[A \to \sigma\bullet a\sigma'] \triangleq \boldsymbol{\lambda}\rho\cdot[A \to \sigma\bullet a\sigma']^\sharp \,\dot{\,}^\sharp\, \hat{\mathsf{F}}^\sharp[A \to \sigma a\bullet\sigma']\rho$$
$$\hat{\mathsf{F}}^\sharp[A \to \sigma\bullet B\sigma'] \triangleq \boldsymbol{\lambda}\rho\cdot[A \to \sigma\bullet B\sigma']^\sharp(\rho, B) \,\mathring{,}^\sharp\, \hat{\mathsf{F}}^\sharp[A \to \sigma B\bullet\sigma']\rho$$
$$\hat{\mathsf{F}}^\sharp[A \to \sigma\bullet] \triangleq \boldsymbol{\lambda}\rho\cdot[A \to \sigma\bullet]^\sharp$$

Instances :

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

TOP-DOWN GRAMMAR ANALYSIS

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

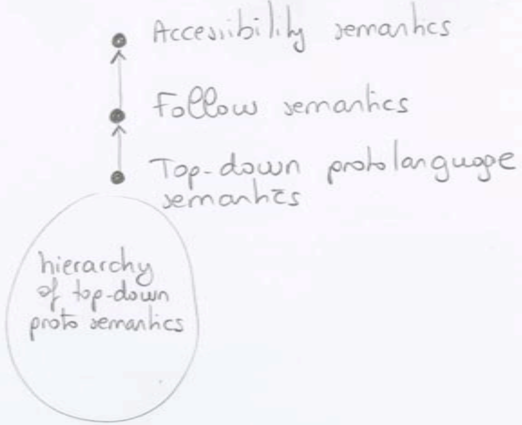
- Choose some ~~bottom-up~~ top-down semantics $S = \mathrm{lfp}^{\subseteq} F$
- define an abstraction α into a finite domain
- design $F^\# = \alpha \circ F \circ \gamma$ such that $\alpha \circ F = F^\#$
- it follows that $S^\# \triangleq \alpha(S) = \mathrm{lfp}^{\subseteq} F^\#$
- the algorithm is just the iterative computa
  $X^0 = \bot, \ldots, X^{n+1} = F^\#(X^n)$ 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 :
  - $\alpha^{\overline{S}}(f) = f(\overline{S})$
  - $\alpha^a \triangleq \boldsymbol{\lambda}\Sigma\cdot\boldsymbol{\lambda}A\cdot(\exists\sigma,\sigma' \in \mathscr{V}^* : \sigma A\sigma' \in \Sigma \mathrel{?} \mathrm{tt} \mathrel{\text{:}} \mathrm{ff})$
- Nonterminal accessibility semantics
  - Definition $\mathsf{S}^a[\![\mathcal{G}]\!] \triangleq \alpha^a(\mathsf{S}^L[\![\mathcal{G}]\!](\overline{S})) = \alpha^a \circ \alpha^{\overline{S}}(\mathsf{S}^L[\![\mathcal{G}]\!])$
  - Abstraction theorems : $\alpha^{\overline{S}}(\mathsf{S}^L[\![\mathcal{G}]\!]) = \mathbf{lfp}^{\subseteq}\, \boldsymbol{\lambda}X\cdot\{\overline{S}\}\cup \mathrm{post}[\Longrightarrow_{\mathcal{G}}]X$

$$\mathsf{S}^a[\![\mathcal{G}]\!] = \mathbf{lfp}^{\subseteq}\, \hat{\mathsf{F}}^a[\![\mathcal{G}]\!]$$
$$\text{where } \hat{\mathsf{F}}^a[\![\mathcal{G}]\!] \triangleq \boldsymbol{\lambda}\phi\cdot\boldsymbol{\lambda}A\cdot(A = \overline{S}) \vee \bigvee_{B\to\sigma A\sigma'\in\mathscr{R}} \phi(B)$$

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

$$\mathsf{S}^{\vec{\partial}}[\![\mathcal{G}]\!] = \mathsf{lfp}^{\subseteq}\, \mathsf{F}^{\vec{\partial}}[\![\mathcal{G}]\!] \quad \text{where} \quad \mathsf{F}^{\vec{\partial}}[\![\mathcal{G}]\!] \triangleq \boldsymbol{\lambda} X \cdot \{\vdash\} \cup X ; \longrightarrow$$

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

where the prefix is $\underbrace{\vdash \rightarrow \cdots \rightarrow}_{\sigma_1 \cdots \sigma_i} \varpi$

as follows :

$$\alpha^{LL} \triangleq \boldsymbol{\lambda}\overline{S} \cdot \boldsymbol{\lambda}\sigma \cdot \boldsymbol{\lambda} X \cdot \{\langle i, \varpi \rangle \mid \begin{array}{l} \exists \theta = \varpi_0 \xrightarrow{\ell_0} \varpi_1 \ldots \varpi_{m-1} \xrightarrow{\ell_{m-1}} \varpi_m \in X.\overline{S} : \\ i \in [0, |\sigma|] \wedge \alpha^{\tau}(\theta) = \sigma_1 \ldots \sigma_i \wedge \varpi = \varpi_m\} . \end{array}$$

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

- Correctness of the parser :

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

- Nonrecursive predictive parsing semantics :

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

where

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

- Parsing algorithm : reachable states of :

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

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

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

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

  - input $\sigma = b$: loops!

$$
\begin{aligned}
& \langle 0, \vdash \rangle \\
\xrightarrow{LL}\ & \langle 0, \dashv [A \to \bullet A] \rangle \\
\xrightarrow{LL}\ & \langle 0, \dashv [A \to \bullet A][A \to \bullet A] \rangle \\
\xrightarrow{LL}\ & \langle 0, \dashv [A \to \bullet A][A \to \bullet A][A \to \bullet A] \rangle \\
\xrightarrow{LL}\ & \ldots
\end{aligned}
$$

- Termination:

Theorem 107 *The nonrecursive predictive parsing algorithm for a grammar $\mathcal{G} = (\mathcal{T}, \mathcal{N}, \overline{S}, \mathcal{R})$ 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 $\exists A \in \mathcal{N} : \exists \eta \in \mathcal{V}^* : A \overset{+}{\Longmapsto}_{\mathcal{G}} A\eta$).*

- Adding a lookahead:

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

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

where $\mathsf{S}^{\bar{1}}[\![G]\!]$ is the extension of the first semantics $\mathsf{S}^{1}[\![G]\!]$ to protosentences:

$$
\mathsf{S}^{\bar{1}}[\![\mathcal{G}]\!] = \boldsymbol{\lambda} \eta \cdot \{ a \in \mathcal{T} \mid \exists \sigma \in \mathcal{T}^* : \eta \overset{\bullet}{\Longmapsto}_{\mathcal{G}} a\sigma \} \cup \{ \epsilon \mid \eta \overset{\bullet}{\Longmapsto}_{\mathcal{G}} \epsilon \}
$$

(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.

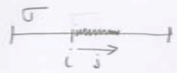
In general non deterministic, deterministic under specific conditions

e.g. nondeterministic $\to$ Tomita algorithm
deterministic $\to$ Knuth LR(k) algorithm

The Cocke-Younger-Kasami (CYK) Algorithm

- Abstract domain for input σ :

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



- Abstraction :

$$\alpha^{CYK} \triangleq \boldsymbol{\lambda}\sigma \cdot \boldsymbol{\lambda}X \cdot \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} \in X\} \qquad \langle \wp(\mathscr{T}^\star), \subseteq\rangle \underset{\alpha^{CYK}(\sigma)}{\overset{\gamma^{CYK}(\sigma)}{\leftrightarrows}} \langle \wp(\hat{\mathsf{D}}^{CYK}(\sigma)), \subseteq\rangle$$

$$\alpha^{CYK} \triangleq \boldsymbol{\lambda}\sigma \cdot \boldsymbol{\lambda}X \cdot \boldsymbol{\lambda}A \cdot \alpha^{CYK}(X(A)) \qquad \langle \mathscr{N} \mapsto \wp(\mathscr{T}^\star), \subseteq\rangle \underset{\alpha^{CYK}(\sigma)}{\overset{\gamma^{CYK}(\sigma)}{\leftrightarrows}} \langle \mathscr{N} \mapsto \wp(\hat{\mathsf{D}}^{CYK}(\sigma)), \subseteq\rangle$$

- 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 :

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

where

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

The calculational design of the CYK parser by abstract interpretation :

PROOF We apply **Cor. 12**.

$$\begin{aligned}
&- \ \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[\![\mathcal{G}]\!](\rho))\\
&= \alpha^{CYK}(\sigma)(\boldsymbol{\lambda}A \cdot \bigcup_{A \to \sigma \in \mathscr{R}} \mathsf{F}^\ell[A \to .\sigma]\rho) && \text{(def. (72) of } \mathsf{F}^\ell[\![\mathcal{G}]\!])\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} \in \bigcup_{A \to \sigma \in \mathscr{R}} \mathsf{F}^\ell[A \to .\sigma]\rho\} && \text{(def. (199) of } \alpha^{CYK})\\
&= \bigcup_{A \to \sigma \in \mathscr{R}} \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} \in \mathsf{F}^\ell[A \to .\sigma]\rho\} && \text{(def. } \in)\\
&= \bigcup_{A \to \sigma \in \mathscr{R}} \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[A \to .\sigma]\rho) && \text{(def. (198) of } \alpha^{CYK})\\
&= \bigcup_{A \to \sigma \in \mathscr{R}} \hat{\mathsf{F}}^{CYK}[A \to .\sigma](\alpha^{CYK}(\sigma)(\rho)) && \text{(provided we can define } \hat{\mathsf{F}}^{CYK} \text{ such that } \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[A \to \sigma.\sigma']\rho) = \hat{\mathsf{F}}^{CYK}[A \to \sigma.\sigma'](\alpha^{CYK}(\sigma)(\rho)))
\end{aligned}$$

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

$$\begin{aligned}
&- \ \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[A \to \sigma.a\sigma']\rho)\\
&= \alpha^{CYK}(\sigma)(a\, \mathsf{F}^\ell[A \to \sigma a.\sigma']\rho) && \text{(def. } \mathsf{F}^\ell[\![\mathcal{G}]\!])\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} \in (a\, \mathsf{F}^\ell[A \to \sigma a.\sigma']\rho)\} && \text{(def. (198) of } \alpha^{CYK})\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i = a \wedge \sigma_{i+1} \ldots \sigma_{i+j-1} \in \mathsf{F}^\ell[A \to \sigma a.\sigma']\rho\} && \text{(def. concatenation)}\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i = a \wedge \langle i+1, j-1\rangle \in \{\langle i', j'\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_{i'} \ldots \sigma_{i'+j'-1} \in \mathsf{F}^\ell[A \to \sigma a.\sigma']\rho\}\} && \text{(def. } \in)\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i = a \wedge \langle i+1, j-1\rangle \in \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[A \to \sigma a.\sigma']\rho)\} && \text{(def. (198) of } \alpha^{CYK})\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i = a \wedge \langle i+1, j-1\rangle \in \hat{\mathsf{F}}^{CYK}[A \to \sigma a.\sigma'](\alpha^{CYK}(\sigma)(\rho))\} && \text{(ind. hyp.)}\\
&= \hat{\mathsf{F}}^{CYK}[A \to \sigma.a\sigma'](\alpha^{CYK}(\sigma)(\rho)) && \text{(by defining } \hat{\mathsf{F}}^{CYK}[A \to \sigma.a\sigma']\rho \triangleq \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i = a \wedge \langle i+1, j-1\rangle \in \hat{\mathsf{F}}^{CYK}[A \to \sigma a.\sigma']\rho\})
\end{aligned}$$

$$\begin{aligned}
&- \ \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[A \to \sigma.B\sigma']\rho)\\
&= \alpha^{CYK}(\sigma)(\rho(B)\, \mathsf{F}^\ell[A \to \sigma B.\sigma']\rho) && \text{(def. } \mathsf{F}^\ell[\![\mathcal{G}]\!])\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} \in (\rho(B)\, \mathsf{F}^\ell[A \to \sigma B.\sigma']\rho)\} && \text{(def. (198) of } \alpha^{CYK})\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \exists k : 0 \leqslant k \leqslant j : \sigma_i \ldots \sigma_{i+k-1} \in \rho(B) \wedge \sigma_{i+k} \ldots \sigma_{i+j-1} \in \mathsf{F}^\ell[A \to \sigma B.\sigma']\rho\} && \text{(def. concatenation)}\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \exists k : 0 \leqslant k \leqslant j : \langle i, k\rangle \in \{\langle i', j'\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_{i'} \ldots \sigma_{i'+j'-1} \in \rho(B)\} \wedge \langle i+k, j-k\rangle \in \{\langle i', j'\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_{i'} \ldots \sigma_{i'+j'-1} \in \mathsf{F}^\ell[A \to \sigma B.\sigma']\rho\}\} && \text{(def. } \in)\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \exists k : 0 \leqslant k \leqslant j : \langle i, k\rangle \in \alpha^{CYK}(\rho(B)) \wedge \langle i+k, j-k\rangle \in \alpha^{CYK}(\mathsf{F}^\ell[A \to \sigma B.\sigma']\rho)\} && \text{(def. (198) of } \alpha^{CYK})\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \exists k : 0 \leqslant k \leqslant j : \langle i, k\rangle \in \alpha^{CYK}(\rho)(B) \wedge \langle i+k, j-k\rangle \in \alpha^{CYK}(\mathsf{F}^\ell[A \to \sigma B.\sigma']\rho)\} && \text{(def. (199) of } \alpha^{CYK})\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \exists k : 0 \leqslant k \leqslant j : \langle i, k\rangle \in \alpha^{CYK}(\rho)(B) \wedge \langle i+k, j-k\rangle \in \hat{\mathsf{F}}^{CYK}[A \to \sigma B.\sigma'](\alpha^{CYK}(\rho))\} && \text{(ind. hyp.)}\\
&= \hat{\mathsf{F}}^{CYK}[A \to \sigma.B\sigma'](\alpha^{CYK}(\sigma)(\rho)) && \text{(by defining } \hat{\mathsf{F}}^{CYK}[A \to \sigma.B\sigma']\rho \triangleq \{\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) \wedge \langle i+k, j-k\rangle \in \hat{\mathsf{F}}^{CYK}[A \to \sigma B.\sigma']\rho\})
\end{aligned}$$

$$\begin{aligned}
&- \ \alpha^{CYK}(\sigma)(\mathsf{F}^\ell[A \to \sigma.]\rho)\\
&= \alpha^{CYK}(\sigma)(\epsilon) && \text{(def. } \mathsf{F}^\ell[\![\mathcal{G}]\!])\\
&= \{\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma) \mid \sigma_i \ldots \sigma_{i+j-1} = \epsilon\} && \text{(def. (198) of } \alpha^{CYK})\\
&= \{\langle i, 0\rangle \mid 1 \leqslant i \leqslant |\sigma|\} && \text{(def. equality of sentences)}\\
&= \hat{\mathsf{F}}^{CYK}[A \to \sigma.](\alpha^{CYK}(\sigma)(\rho)) && \text{(by defining } \hat{\mathsf{F}}^{CYK}[A \to \sigma.]\rho \triangleq \{\langle i, 0\rangle \mid 1 \leqslant i \leqslant |\sigma|\}) \ \blacksquare
\end{aligned}$$

Because the abstract domain $\langle \mathscr{N} \mapsto \wp(\hat{\mathsf{D}}^{CYK}(\sigma)), \subseteq\rangle$ is finite, the iterative computation of $\mathbf{lfp}^{\subseteq}\, \hat{\mathsf{F}}^{CYK}[\![\mathcal{G}]\!](\sigma)$ 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 $\langle i, j\rangle \in \hat{\mathsf{D}}^{CYK}(\sigma)$ 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 !