Extensible syntax analyzers

Pavel Egorov
pe@skbkontur.ru

Problem statements

L – some formal language;

$T : L \rightarrow R$ – it's translator;

$L_{ext} \supset L$ – some extended language;

$T_{ext} : L_{ext} \rightarrow R_{ext}$ – it's extended translator.

$\forall \omega \in L \quad T_{ext}(\omega) = T(\omega)$ **We want compatibility!**

Disclaimer

- Consider **syntax analyzers** only.
- **Fixed algorithm** of syntax analysis.
- Extension due to **modification of the grammar** only.

What is syntax analyzer?

- SA converts a word to a syntax tree.
- What is syntax tree?
  - Root is labeled by the axiom of the grammar
  - Inner nodes are labeled by non-terminals
  - Leaf nodes are labeled by terminals or non-terminals or ε

Classic parse tree

Grammar:

$A \rightarrow aBd$

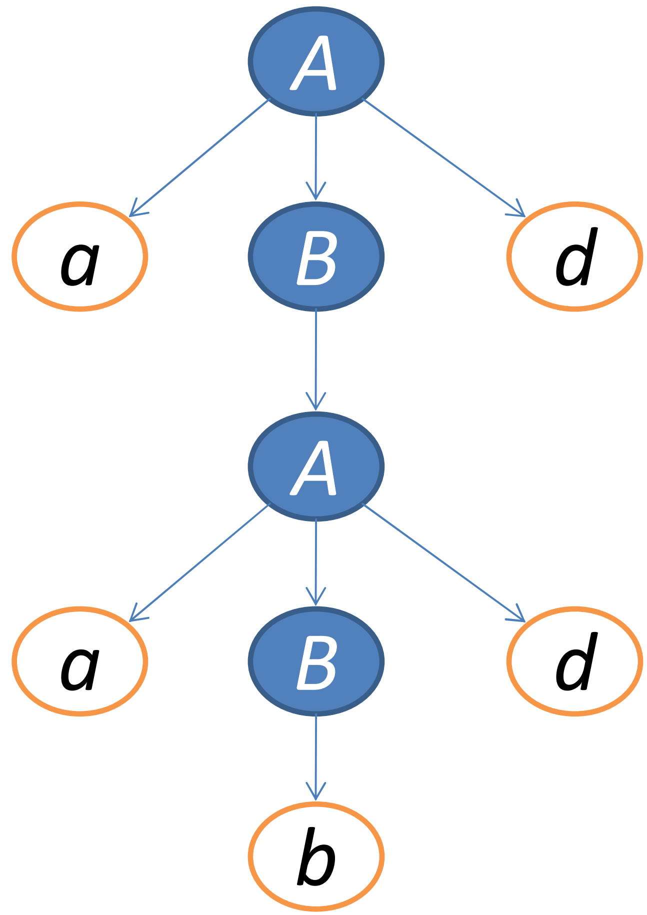
$B \rightarrow b$

$B \rightarrow \mathrm{A}$

Word: “aabdd”

Derivation:

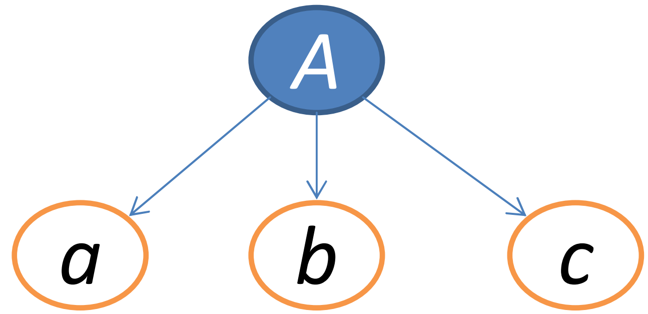
$$A \rightarrow aBd \rightarrow aAd \rightarrow$$
$$\rightarrow aaBdd \rightarrow aabdd$$



Classic parse tree
Is it good enough?

G_1:

$$A \rightarrow abc$$

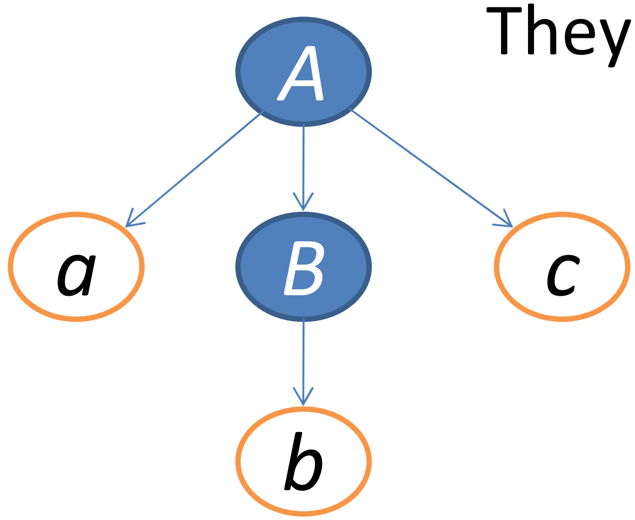


Extended G_2:

$$A \rightarrow aBc$$

$$B \rightarrow b$$

They are different!

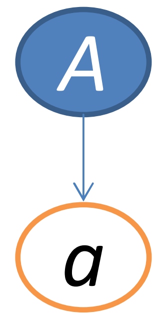


Word: "*abc*"

Classic parse tree
Is it good enough?

G_1:

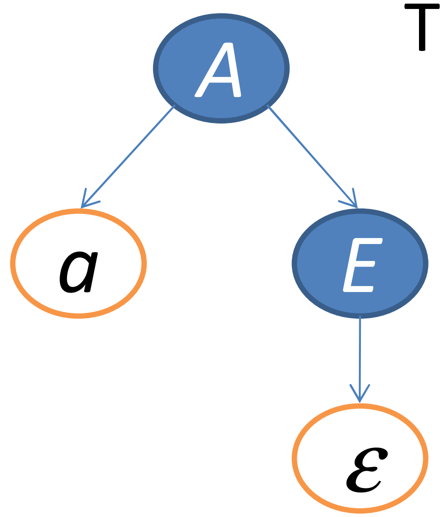
$A \rightarrow a$



Extended G_2:

$A \rightarrow aE$

$E \rightarrow \varepsilon$



They are different!

Word: "a"

Cancelled tree

- Driven by **labeled** grammar
- Cut off some inner nodes from parse tree
- Cut off ε-leafs from parse tree

Is much better for parser extension!

Cancelled tree example

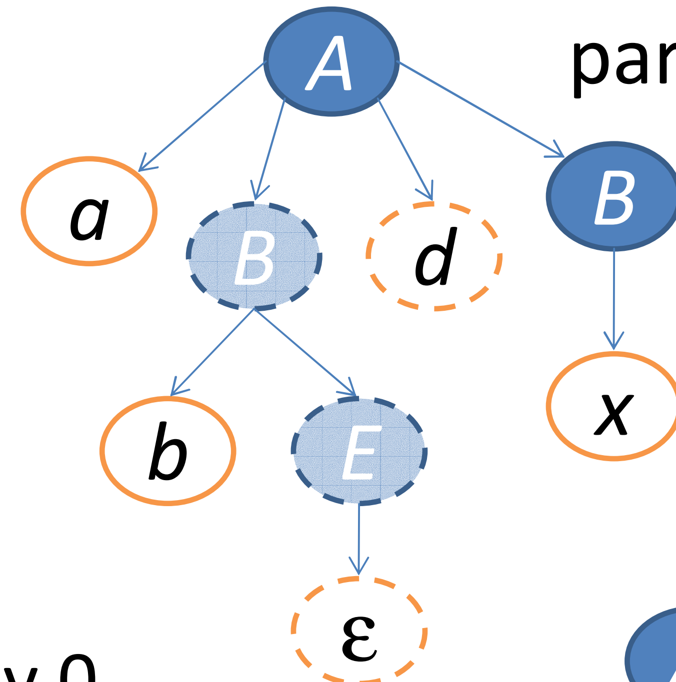
Labeled grammar:

$A \rightarrow a^1 B^0 d^0 B^1$

$B \rightarrow b^1 E^0$

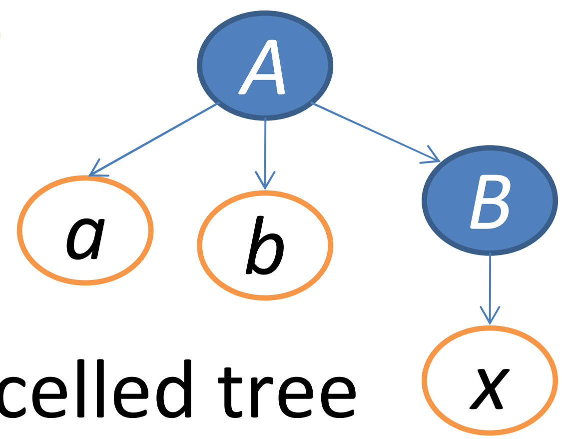
$B \rightarrow x^1$

$E \rightarrow \varepsilon^0$



parse tree

- Symbols labeled by 0 are not present in tree
- ε-leafs are always labeled by 0



cancelled tree

Intermediate results

- Labeled **grammar**
- **Syntax analyzer** driven by labeled grammar
- **Canceled tree** as a result of syntax analysis

So how can we extend grammars without breaking compatibility?

Compatibility: $\forall \omega \in L \quad SA_{ext}(\omega) = SA(\omega)$

Extending transformations of grammars

f – extending grammar transformation

G - grammar

$$L(f(G)) \supseteq L(G)$$

Extending transformations

- ADD-transform adds some fixed new production to a grammar.

- EXTRACT-transform:

  before: $A \longrightarrow \alpha^{x}\beta^{y}\gamma^{z}$

  after: $A \longrightarrow \alpha^{x}B^{0}\gamma^{z}$

  $B \longrightarrow \beta^{y}$

Extending transformations

AE * – all transforms, composed from arbitrary number of ADD- and EXTRACT-transforms.

Results

Compatibility

1. Any transform from AE* doesn’t break compatibility of corresponding syntax analyzers.
2. A grammar extended by AE* transform has the same production labeling as initial grammar has on the common productions. (AE* doesn’t change labeling of unchanged productions)

Independent transforms

$$\begin{tikzcd} & G \arrow[dl, "\psi_1"'] \arrow[dr, "\psi_2"] & \\ G_1 & & G_2 \end{tikzcd}$$

$\psi_1, \psi_2 \in AE^*$

$G_2 \in Domain(\psi_1)$?
$G_1 \in Domain(\psi_2)$?

G_{12} **???**

When it is possible?
How can we do it?

Independent transforms

What if $G_1 \notin$ Domain(ψ_2)?

Sometimes we can solve this problem!

$A \rightarrow \alpha\beta\gamma\delta$

ψ_1: $A \rightarrow \alpha\beta C\delta$; $C \rightarrow \gamma$

Conflict!

ψ_2: $A \rightarrow \alpha B\delta$; $B \rightarrow \beta\gamma$

$\boldsymbol{\psi_2/\psi_1}$: $A \rightarrow \alpha B\delta$; $B \rightarrow \beta C$ **Decision!**

ψ/ϕ – a **fixed** transform

- ψ/ϕ – "**fixed**" ψ, which can be applied after ϕ
- Sometimes we can't fix ψ... ☹

  $A \rightarrow \alpha\beta\gamma$

  ψ: $A \rightarrow X\gamma$; $X \rightarrow \alpha\beta$

  ϕ: $A \rightarrow \alpha Y$; $Y \rightarrow \beta\gamma$
- Recursive definition of the function ψ/ϕ for ψ and ϕ from $\boldsymbol{AE}^*$.
- Sometimes ψ/ϕ is not defined.

Independent transforms

$$
\begin{tikzcd}
 & G \arrow[dl, "\psi_1"'] \arrow[dr, "\psi_2"] & \\
G_1 \arrow[d, "\psi_2/\psi_1"'] & & G_2 \arrow[d, "\psi_1/\psi_2"] \\
G_{12} & \textbf{?} & G_{21}
\end{tikzcd}
$$

$$\psi_2/\psi_1\cdot\psi_1 \quad ? \quad \psi_1/\psi_2\cdot\psi_2$$

Independent transforms

$$
\begin{tikzcd}
 & G \arrow[dl, "\psi_1"'] \arrow[dr, "\psi_2"] & \\
G_1 \arrow[dr, "\psi_2/\psi_1"'] & & G_2 \arrow[dl, "\psi_1/\psi_2"] \\
 & G_{12} &
\end{tikzcd}
$$

$$\psi_2/\,\psi_1 \cdot \psi_1 = \psi_1/\,\psi_2 \cdot \psi_2$$

Conclusion

- Syntax analyzers
  – driven by labeled grammars
  – cancelled trees as a result
- ***AE**** - a way to extend grammars safelly
- ψ/ϕ – a way to compose independent grammar transforms
- Order of application of the extending transforms doesn’t matter!

Thank you!

Questions?