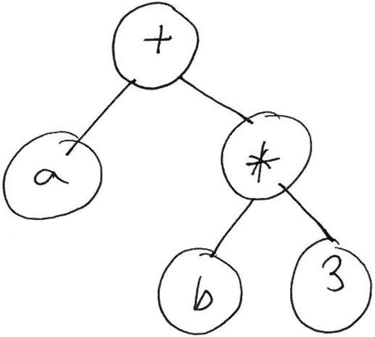
First Steps in ANTLR
Programming Languages Lab

Samuele Buro
Programming Languages (AY 2020-21)
University of Verona
November 6, 2020



The Big Picture

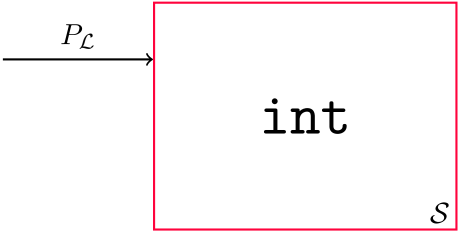
`int`

The Big Picture

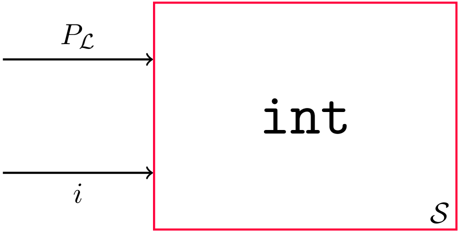
`int`

$\mathcal{S}$

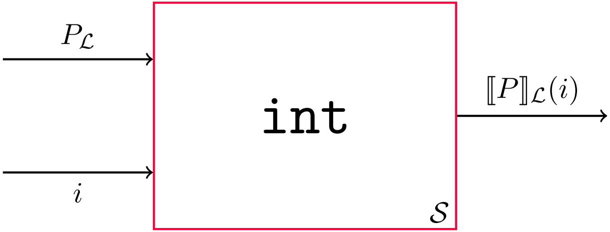
The Big Picture



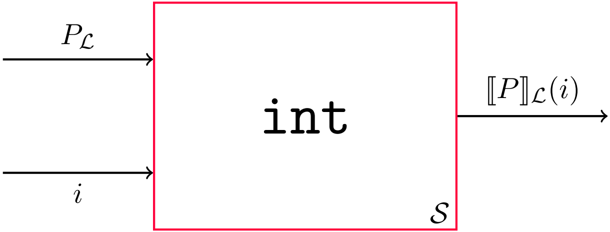
The Big Picture



The Big Picture

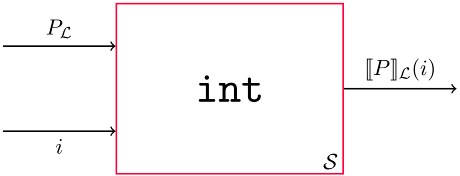


The Big Picture



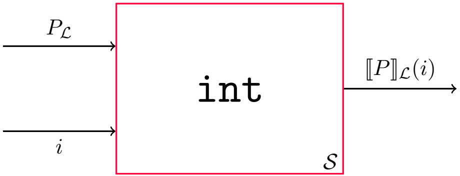
$$[\![\texttt{int}]\!]_{\mathcal{S}}(P, i) = [\![P]\!]_{\mathcal{L}}(i)$$

Handling Ill-Formed Input



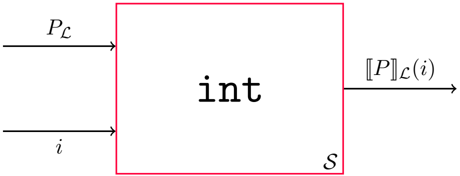
- What happens if $P \notin \mathcal{L}$?

Handling Ill-Formed Input



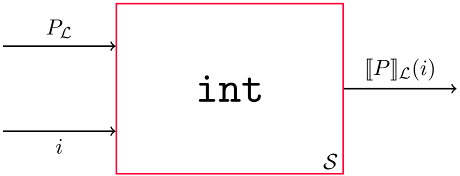
- What happens if $P \notin \mathcal{L}$?
  - We would like `int` to be able to decide if $P \in \mathcal{L}$

Handling Ill-Formed Input



- What happens if $P \notin \mathcal{L}$?
  - We would like `int` to be able to decide if $P \in \mathcal{L}$
    - if $P \in \mathcal{L}$, then compute $[\![P]\!]_{\mathcal{L}}(i)$

Handling Ill-Formed Input



- What happens if $P \notin \mathcal{L}$?
  - We would like `int` to be able to decide if $P \in \mathcal{L}$
    - if $P \in \mathcal{L}$, then compute $[\![P]\!]_{\mathcal{L}}(i)$
    - if $P \notin \mathcal{L}$, then return an error

Handling Ill-Formed Input

Case 1: No restriction on $\mathcal{L}$

Handling Ill-Formed Input

Case 1: No restriction on $\mathcal{L}$

- $P \in \mathcal{L}$ is undecidable

Handling Ill-Formed Input

Case 1: No restriction on $\mathcal{L}$

- $P \in \mathcal{L}$ is undecidable
  - (Spoiler Fondamenti dell'Informatica)

Handling Ill-Formed Input

Case 1: No restriction on $\mathcal{L}$

- $P \in \mathcal{L}$ is undecidable
  - (Spoiler Fondamenti dell'Informatica)

Case 2: Suppose $\mathcal{L} = \mathsf{Lang}(G)$, where G is a CFG

Handling Ill-Formed Input

Case 1: No restriction on $\mathcal{L}$

- $P \in \mathcal{L}$ is undecidable
    - (Spoiler Fondamenti dell'Informatica)

Case 2: Suppose $\mathcal{L} = \mathsf{Lang}(G)$, where G is a CFG

- $P \in \mathcal{L}$ is decidable

Handling Ill-Formed Input

Case 1: No restriction on $\mathcal{L}$

- $P \in \mathcal{L}$ is undecidable
  - (Spoiler Fondamenti dell'Informatica)

Case 2: Suppose $\mathcal{L} = \mathsf{Lang}(G)$, where G is a CFG

- $P \in \mathcal{L}$ is decidable
  - (Spoiler Compilatori)

ANTLR

ANTLR = ANother Tool for Language Recognition

$\boxed{\text{ANTLR}}$

ANTLR

ANTLR = ANother Tool for Language Recognition

$$\xrightarrow{\quad G \quad} \boxed{\text{ANTLR}}$$

ANTLR

ANTLR = ANother Tool for Language Recognition

$$\xrightarrow{G} \boxed{\text{ANTLR}} \xrightarrow{\text{Parser}_G}$$

ANTLR

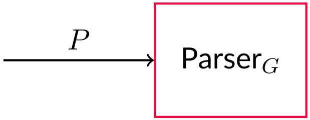
ANTLR $=$ ANother Tool for Language Recognition

- Parser generator

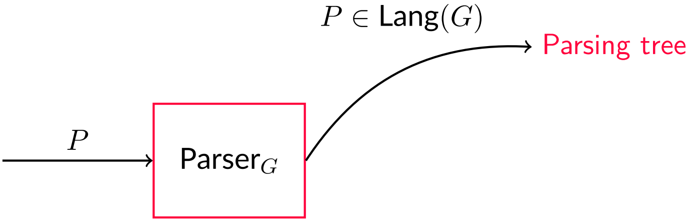
Parser

$\boxed{\text{Parser}_G}$

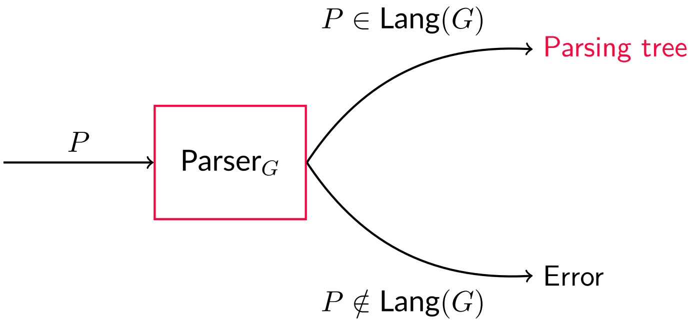
Parser



Parser



Parser



Grammar Golden Rule

No indirect left recursion!

Grammar Golden Rule

No indirect left recursion!

$S \to \mathtt{a}S$

Grammar Golden Rule

No indirect left recursion!

$S \to \mathtt{a}S$	✓

Grammar Golden Rule

No indirect left recursion!

$S \rightarrow \mathtt{a}S$	✓
$S \rightarrow S\mathtt{a}$	

Grammar Golden Rule

No indirect left recursion!

$S \rightarrow \texttt{a}S$	✓
$S \rightarrow S\texttt{a}$	✓

Grammar Golden Rule

No indirect left recursion!

$S \to \mathtt{a}S$	✓
$S \to S\mathtt{a}$	✓
$S \to S$	

Grammar Golden Rule

No indirect left recursion!

$S \to \mathtt{a}S$	✓
$S \to S\mathtt{a}$	✓
$S \to S$	✗

Grammar Golden Rule

No indirect left recursion!

$S \rightarrow \mathtt{a}S$	✓
$S \rightarrow S\mathtt{a}$	✓
$S \rightarrow S$	✗
$S \rightarrow T$ $T \rightarrow S$	

Grammar Golden Rule

No indirect left recursion!

$S \rightarrow \texttt{a}S$	✓
$S \rightarrow S\texttt{a}$	✓
$S \rightarrow S$	✗
$S \rightarrow T$ $T \rightarrow S$	✗

Grammar Golden Rule

No indirect left recursion!

$S \to \mathtt{a}S$	✓
$S \to S\mathtt{a}$	✓
$S \to S$	✗
$S \to T$ $T \to S$	✗
$S \to T$ $T \to S \mid \mathtt{a}S$	

Grammar Golden Rule

No indirect left recursion!

$S \to \mathtt{a}S$	✓
$S \to S\mathtt{a}$	✓
$S \to S$	✗
$S \to T$ $T \to S$	✗
$S \to T$ $T \to S \mid \mathtt{a}S$	✗

Exercise

Define

$$\lambda(A) = \{\, (a_1, \ldots, a_n) \mid n \in \mathbb{N} \wedge a_1 \ldots a_n \in A \,\}$$

Write a context-free grammar able to generate the language $\mathcal{L}$ of all the lists of decimal digits:

$$\mathcal{L} = \lambda(\texttt{Dig})$$

where $\texttt{Dig} = \{\, \texttt{0}, \texttt{1}, \ldots, \texttt{9} \,\}$. Note that, if $n = 0$, then $() \in \mathcal{L}$.

Exercise

Define, for $i \in \mathbb{N}$,

$$\begin{cases} \lambda^0(A) = A \\ \lambda^{i+1}(A) = \lambda\left(\bigcup_{0 \leq j \leq i} \lambda^j(A)\right) \end{cases}$$

Change the context-free grammar of the previous exercise to generate the language $\widehat{\mathcal{L}}$ of all the recursive lists of digits, i.e., lists of digits whose elements can also be recursive lists of digits:

$$\widehat{\mathcal{L}} = \bigcup_{i>0} \lambda^i(\mathtt{Dig})$$