

Programming Paradigms

Syntax (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Motivation

- **Goal:**

- Specify a programming language**

- What code is part of the language?
 - What is the meaning of a piece of code?

- **Important for both developers and tools**

- **In contrast: Natural languages not formally specified**

Syntax vs. Semantics

Structure of
code

Meaning of
code

Example:

Grammar to define a language: **Could mean**

digit \rightarrow 0 | 1 | ... | 9

non_zero_digit \rightarrow 1 | ... | 9

number \rightarrow non_zero_digit digit*

- Natural numbers
- Days of a 10-day week
- Colors
- ...

Syntax vs. Semantics

Structure of
code

Meaning of
code

Example:

Grammar to define a language: **Could mean**

digit → 0 | 1 | ... | 9

non_zero_digit → 1 | ... | 9

number → non_zero_digit digit*

- Natural numbers
- Days of a 10-day week
- Colors
- ...

Focus of this and next lecture

Syntax of Different PLs

Common: **Different syntax, same semantics**

Java:

```
if (foo > 100) {  
    ...  
}
```

Bash:

```
if [ $foo -gt 100 ]  
then  
    ...  
fi
```

Syntax of Different PLs

Common: Different syntax, same semantics

Java:

```
if (foo > 100) {  
    ...  
}
```

Bash:

```
if [ $foo -gt 100 ]  
then  
    ...  
fi
```

Sometimes: Same syntax, different semantics

Java:

```
if ("abc" != 5) {  
    ...  
}
```

JavaScript:

Syntax of Different PLs

Common: Different syntax, same semantics

Java:

```
if (foo > 100) {  
    ...  
}
```

Bash:

```
if [ $foo -gt 100 ]  
then  
    ...  
fi
```

Sometimes: Same syntax, different semantics

Java:

Type error at
compile time

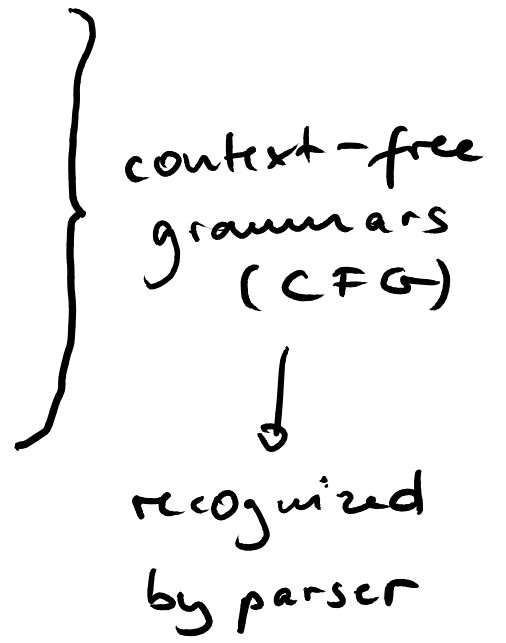
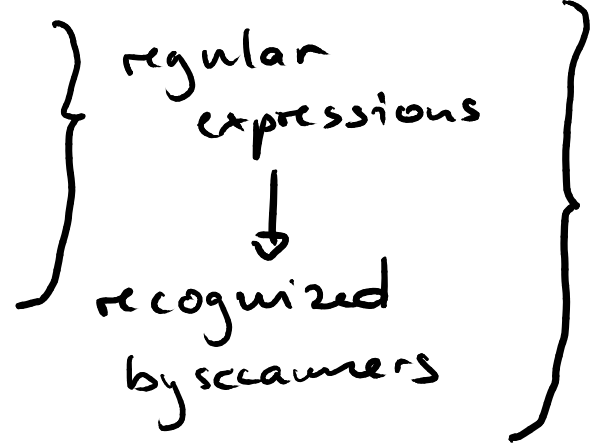
```
if ("abc" != 5) {  
    ...  
}
```

JavaScript:

Branch is
executed

4 concepts to specify syntax

- concatenation
- alternation / choice
- repetition ("Kleene closure")
- recursion



Overview

- **Specifying syntax**
 - Regular expressions
 - Context-free grammars
- **Scanning**
- **Parsing**

Tokens

Basic **building blocks** of every PL

- Keywords, identifiers, constants, operators
- Think: "Words" of the language

Example: C has more than 100 tokens

- Keywords, e.g., `double`, `if`, `return`, `struct`
- Identifiers, e.g., `my_var`, `printf`
- Literals, e.g., `6.022e23`, `'x'`
- Punctuators, e.g., `(`, `}`, `&&`

Regular Expressions

- Used to **specify tokens**
- A regular expression is one of:
 - A **character**
 - The **empty string** ϵ
 - The **concatentation** of two regular expressions
 - Two regular expressions separated by **|**
 - Means a string generated by one or the other
 - A reg. expression followed by the **Kleene star** $*$
 - Means zero or more repetitions

Regular Expressions

- Used to **specify tokens**
- A regular expression is one of:
 - A **character**
 - The **empty string** ϵ
 - The **concatentation** of two regular expressions
 - Two regular expressions separated by **|**
 - Means a string generated by one or the other
 - A reg. expression followed by the **Kleene star** $*$
 - Means zero or more repetitions

No recursion!

Example

Numeric constants accepted by a calculator

number \rightarrow integer | real

integer \rightarrow digit digit*

real \rightarrow integer exponent | decimal (exponent | E)

decimal \rightarrow digit* (. digit | digit.) digit*

exponent \rightarrow (e | E) (+ | - | E) integer

digit \rightarrow 0 | 1 | ... | 9

Quiz

Which of the following strings is accepted by the regular expression *number*?

- 23
- 72.611.24
- 12E-.43
- e75
- 0.0E+0E+0
- 003

Quiz

Which of the following strings is accepted by the regular expression *number*?

- 23 ✓
- 72.611.24 ✗
- 12E-.43 ✗
- e75 ✗
- 0.0E+0E+0 ✗
- 003 ✓

More compact syntax for regular expressions

Language of tokens: $\{ c, cac, cbc, cacac, cbcbe, cacbc, cbcac, \text{etc.} \}$

Token $\rightarrow c \text{ More}^*$

More $\rightarrow A \text{ or } B \ c$

A or B $\rightarrow a \mid b$

Shorter notation: Nest all into one

$$c \underbrace{\underbrace{\underbrace{(a \mid b) c}^*}_{A \text{ or } B}}_{\text{More}}}_{\text{Token}}$$

Identifiers in Popular PLs

Different PLs allow different identifiers

- Case-sensitive vs. case-insensitive
 - E.g., `f00`, `F00`, and `F00` are the same in Ada and Common Lisp, but not in Perl and C
- Letters and digits: Almost always allowed
- Underscore: Allowed in most languages

In addition to syntax rules: **Conventions**

- E.g., Java: `ClassName`, `variableName`

Identifiers in Popular PLs

Different PLs allow different identifiers

- Case-sensitive vs. case-insensitive
 - E.g., `f00`, `F00`, and `F00` are the same in Ada and Common Lisp, but not in Perl and C
- Letters and digits: Almost always allowed
- Underscore: Allowed in most languages

In addition to syntax rules: **Conventions**

- E.g., Java: `ClassName`, `variableName`

Know the rules of the language you use!

White space in Popular PLs

Free format **vs.** formatting as syntax

- Spaces and tabs sometimes matter
 - E.g., in Python
- Line breaks sometimes matter
 - E.g., to separate statements in JavaScript or Python