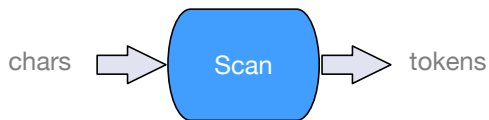


Compiler Construction

~ Lexical Analysis ~

Lexical Analysis



Break the input into individual words
or *tokens*

Remove noise

Lexical Token

Definition

A lexical token is a sequence of character that can be treated as a unit in the grammar of the programming language

Input

```
let var a := 2 in a + 1 end
```

Output

```
..... , TOK_NUMBER(2) , TOK_ASSIGN, TOK_ID(a), TOK_VAR, TOK_LET
```

How to express tokens?

Remark

A programming language classifies lexical tokens into a finite set of token types

Regular expressions

Each token is represented by a regular expression. In other words, each token defines a language.

Example: Regular Expression for some tokens

Each regular expression produce a token
(with or without data)

<code>let</code>	<code>(TOK_LET)</code>
<code>var</code>	<code>(TOK_VAR)</code>
<code>[a-z][a-z0-9]*</code>	<code>(TOK_ID)</code>
<code>[0-9]+</code>	<code>(TOK_NUM)</code>

Order is important!

Usually, the first regular expression that matches determines the token type!

Example: Regular Expression for some tokens

Each regular expression produce a token
(with or without data)

<code>let</code>	<code>(TOK_LET)</code>
<code>var</code>	<code>(TOK_VAR)</code>
<code>[a-z][a-z0-9]*</code>	<code>(TOK_ID)</code>
<code>[0-9]+</code>	<code>(TOK_NUM)</code>

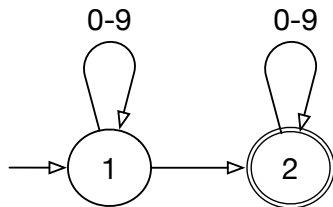
Order is important!

Usually, the first regular expression that matches determines the token type!

Finite automata

It is easy to translate regular expressions into Finite Automata (FA)

Finite automaton for $[0-9][0-9]^*$



Finite automata

Definition

A deterministic finite automaton is an automaton where no two edges leaving the same state have the same symbol

Putting altogether

- One can then combine all automata and determinize the resulting automaton
- This DFA can be encoded as a transition matrix and used programmatically

Existing lexical analyzer generator

- Flex
- Ragel
- re2c
- JFlex
- Annoflex
- PLY
- ...

Summary

NFA

regexp

DFA

tokens

transition
matrix