CS4450/7450: What is a Language?

Professor William L. Harrison

September 10, 2018

The ideas and issues which we will consider are:

- What is a language?
- Syntax: How do we define *precisely* what are the well-formed sentences of a language?
- Semantics: Given a well-formed sentence, what does it mean?
- The separation between syntax and semantics.

Formal Linguistics Point of View

Definition (Language)

Given an alphabet A, a **language** L is a set of strings—i.e., finite sequences—of elements of A. Each element of L is a **sentence**.

Formal Linguistics Point of View

Definition (Language)

Given an alphabet A, a **language** L is a set of strings—i.e., finite sequences—of elements of A. Each element of L is a **sentence**.

Remark

From this point of view, the following are all languages for the ASCII alphabet:

- $\{\texttt{abc}, \texttt{beavis}, \texttt{hrmph}\}$,
- $\{$*"my dog has fleas"*, *"irregardless opp glppp"*$\}$, *and*
- $\{\}$

Languages-as-Sets Incomplete

This language-as-set definition is unsatisfactory for our needs.

Languages-as-Sets Incomplete

This language-as-set definition is unsatisfactory for our needs.

- No way of talking about internal sentence structure
  e.g., can't express English's subject-verb-object pattern

Languages-as-Sets Incomplete

This language-as-set definition is unsatisfactory for our needs.

- No way of talking about internal sentence structure
  e.g., can't express English's subject-verb-object pattern
- "Non-constructive": not amenable to programming
  i.e., how do we write a program that recognizes whether string s is, indeed, in language L?

Languages-as-Sets Incomplete

This language-as-set definition is unsatisfactory for our needs.

- No way of talking about internal sentence structure
  e.g., can't express English's subject-verb-object pattern
- "Non-constructive": not amenable to programming
  i.e., how do we write a program that recognizes whether string s is, indeed, in language L?
- ... especially important since programming languages are infinite—i.e., have infinite number of programs

An example

Q: Is the following a legal C program?

```
$ cat helloworld.c
#include <stdio.h>
int main() {
  printf("hello world\n")
}
```

An example

Q: Is the following a legal C program?

```
$ cat helloworld.c
#include <stdio.h>
int main() {
  printf("hello world\n")
}
```

Nope.

```
$ gcc helloworld.c
helloworld.c: In function ‘main’:
helloworld.c:3: error: parse error
                       before ’}’ token
```

An example

Adding a semi-colon produces a legitimate C program:

```
$ cat helloworld.c
#include <stdio.h>
int main() {
  printf("hello world\n") ;
}
```

An example

Adding a semi-colon produces a legitimate C program:

```
$ cat helloworld.c
#include <stdio.h>
int main() {
  printf("hello world\n") ;
}
```

To whit, no errors are produced when we re-compile:

```
$ gcc helloworld.c
$
```

Discussion

- C has some means of expressing and checking structure of an input file that a programmer claims is a program.

Discussion

- C has some means of expressing and checking structure of an input file that a programmer claims is a program.
- "Context-free Grammar" (CFG): structural rules that determine whether a sequence of symbols is, in fact, a sentence (program) in a language.

Discussion

- C has some means of expressing and checking structure of an input file that a programmer claims is a program.
- "Context-free Grammar" (CFG): structural rules that determine whether a sequence of symbols is, in fact, a sentence (program) in a language.
- CFGs are expressive enough to describe PL syntax and can be readily adapted to programming (parsing).

Discussion

- C has some means of expressing and checking structure of an input file that a programmer claims is a program.
- "Context-free Grammar" (CFG): structural rules that determine whether a sequence of symbols is, in fact, a sentence (program) in a language.
- CFGs are expressive enough to describe PL syntax and can be readily adapted to programming (parsing).
- Kernighan & Ritchie (2nd edition, App. 9.2, page 222):

$$\textit{expression-statement} : \ \textit{expression}_{opt} \ ;$$

Discussion

- C has some means of expressing and checking structure of an input file that a programmer claims is a program.
- "Context-free Grammar" (CFG): structural rules that determine whether a sequence of symbols is, in fact, a sentence (program) in a language.
- CFGs are expressive enough to describe PL syntax and can be readily adapted to programming (parsing).
- Kernighan & Ritchie (2nd edition, App. 9.2, page 222):

$$\textit{expression-statement} : \ \textit{expression}_{opt} \ ;$$

Says "an expression-statement is an expression (in this case the call to printf) followed by a semicolon."