CSCI 334:
Principles of Programming Languages

Lecture 3: What is a language anyway?

Instructor: Dan Barowy

Williams

Topics

What is a language?

Turing equivalence

WCMA activity

Your to-dos

1. Reading check, **due Wednesday 9/14**.
2. Lab 1, **due Sunday 9/18** (partner lab)

What is a language?

In this class, we concern ourselves with a specific formulation of "language," called a **formal language**.

A **formal language** is the set of words whose letters are taken from some **alphabet** and whose construction follows some **rules**.

Example:

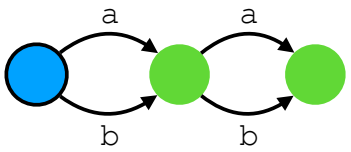
```
L = {a, aa, b, bb, ab, ba}
```

```
Σ = {a, b}
```

```
<expr> ::= <letter> | <letter><letter>
<letter> ::= a | b
```

Interesting fact: language = machine

```
<expr> ::= <letter> | <letter><letter>
<letter> ::= a | b
```



The above is a machine called a deterministic finite state automaton (DFA) that "accepts" only words ∈ L.

The two definitions above describe the same language, but precisely.

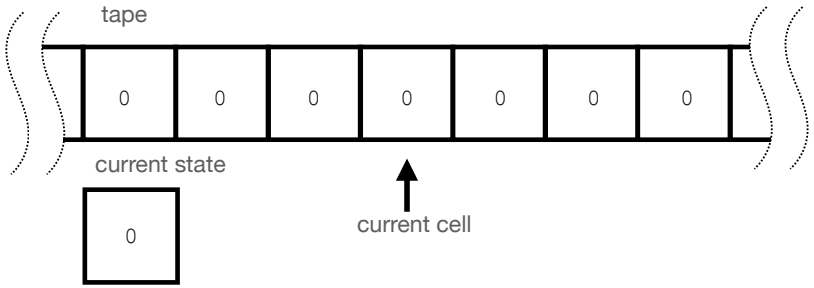
What is a programming language?

A **programming language** is defined by two machines:

1. A **syntax machine** that determines the set of strings that are in the language.
2. A **semantics machine** that determines what gets done (i.e., what computational work) with an accepted string.

We spend a lot of time in PL thinking about these machines, which we call **language models**.

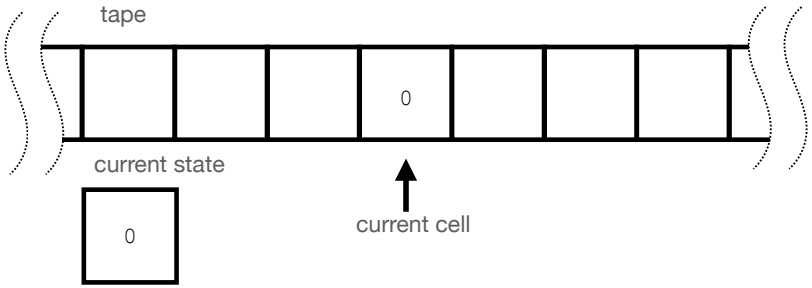
Turing machine



configuration behavior

Current state	Scanned symbol	Print symbol	Move tape	Final (i.e. next) state
A	0	1	R	B
A	1	1	L	C
B	0	1	L	A
B	1	1	R	B
C	0	1	L	B
C	1	1	N	H

Turing machine



Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
b	None	$P0, R$	c
c	None	R	c
e	None	$P1, R$	f
f	None	R	b

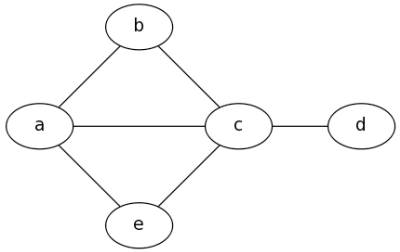
Surprising fact!

General purpose languages are usually **semantically equivalent** in computational power to a **Turing machine**.

Domain specific languages

A **domain-specific language** (DSL) is a computer language specialized to a particular application domain. DSLs are **intentionally** not Turing equivalent, **for simplicity**.

```
graph {
    a -- b;
    b -- c;
    a -- c;
    d -- c;
    e -- c;
    e -- a;
}
```



Keep in mind: two machines

Activity

Recap & Next Class

Today we covered:

WCMA

Next class:

F#