CSCI 334:
Principles of Programming Languages

Lecture 2: What is a language anyway?

Instructor: Dan Barowy

Williams

1

SureVeyor User Study

Come to experiment with our domain-specific programming language designed for social sciences! All backgrounds are welcomed!

For details, contact us at dwb1@williams.edu or ys5@williams.edu

Date: Monday, Sep 11

Time: 4pm–6pm

Location: Ward Lab

There will be pizzas!

Scan the QR code =>

2

Topics

What is a language?

Turing equivalence

WCMA activity

3

Your to-dos

4

1. Read *A Slightly Longer Introduction to F#*
2. Lab 1, **due Sunday 9/17** (partner lab)
   Be sure to tell me who your partner is in `collaborators.txt` file.

What is a language?

5

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**.

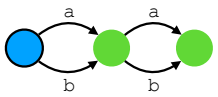
Example:

```
L = {a, aa, b, bb, ab, ba}
Σ = {a, b}
<expr> ::= <letter> | <letter><letter>
<letter> ::= a | b
```

Interesting fact: language = machine

6

```
<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?

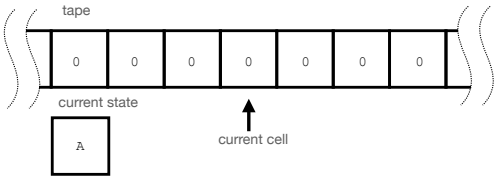
A **programming language** is defined by two machines:

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**.

7

Turing machine



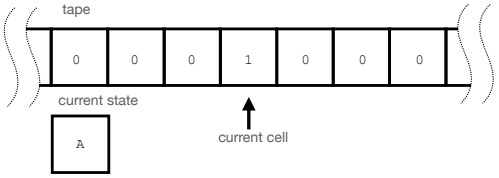
configuration behavior

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

8

Here is one standard language model: the Turing machine. Its operation is very simple. See if you can determine its next steps using the table below.

Turing machine

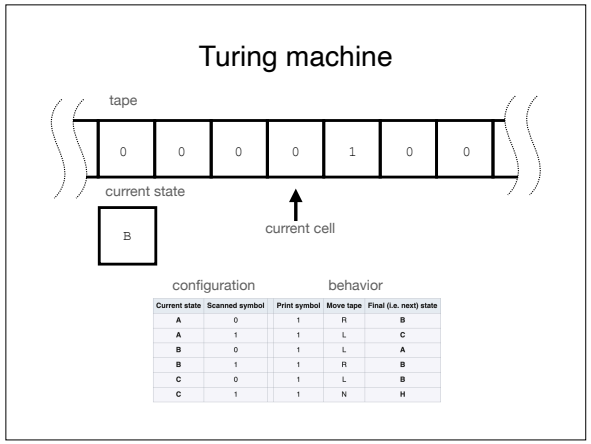


configuration behavior

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

9

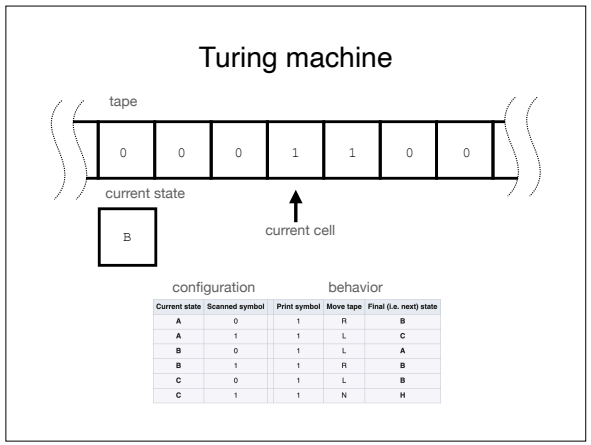
When in state A and current symbol 0, we write a 1 to the tape, then…



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

10

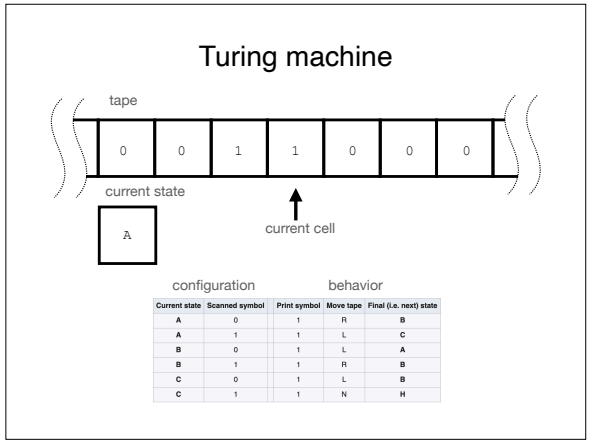
… move the tape to the right and then update the current state to B.



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

11

Next, with B in the current state with 0 in the current cell, we write a 1 and …

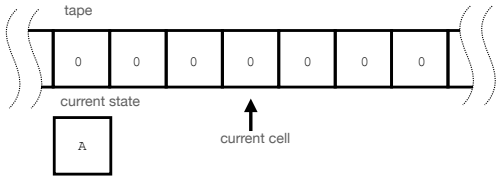


Current state	Scanned symbol	Print symbol	Move tape	Final (i.e. next) state
A	0	1	R	B
A	1	1	L	C
B	0	1	L	A
B	1	1	R	B
C	0	1	L	B
C	1	1	N	H

12

… move the tape to the left and update the current state to A. And so on until the machine halts (“H”).

Turing machine



Configuration *Behaviour*

m-config.	*symbol*	*operations*	*final m-config.*
b	None	*P*0, *R*	c
c	None	*R*	c
c	None	*P*1, *R*	f
f	None	*R*	b

13

Here is a snippet from Turing's paper where these machines were introduced.

Surprising fact!

Almost all general purpose programming languages are **equivalent** in computational power to a **Turing machine**.

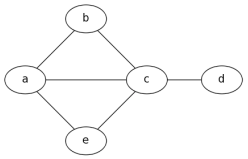
14

Believe it or not, this very simple machine is universal in the sense that all known computations can be performed on it (albeit inconveniently). That makes it good for studying lots of questions about computation. For example, in the Turing machine model, determining the cost of an algorithm is simple: assume each instruction is unit cost and count the number of instructions executed.

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;
}
```



15

But there are other models, and we don't need to use something as powerful as a Turing machine. For example, if all you want to do is to draw a graph, you might use graphViz, which is a declarative language for drawing a graph. There is a 1-to-1 correspondence between the program on the left and the image on the right. Easy to understand, and is purpose-built to draws graphs easily, but if you want to do something more sophisticated, you'll need to use a different language.

Keep in mind: two machines

16

Activity

17

Do the activity in the handout. The idea is to *describe* what you see, not to *interpret* it. Describe what you're looking at in terms a computer can understand. We like to start this activity with the Sol LeWitt wall drawing in the entrance of WCMA.

Recap & Next Class

Today we covered:

WCMA

Next class:

F#

18