

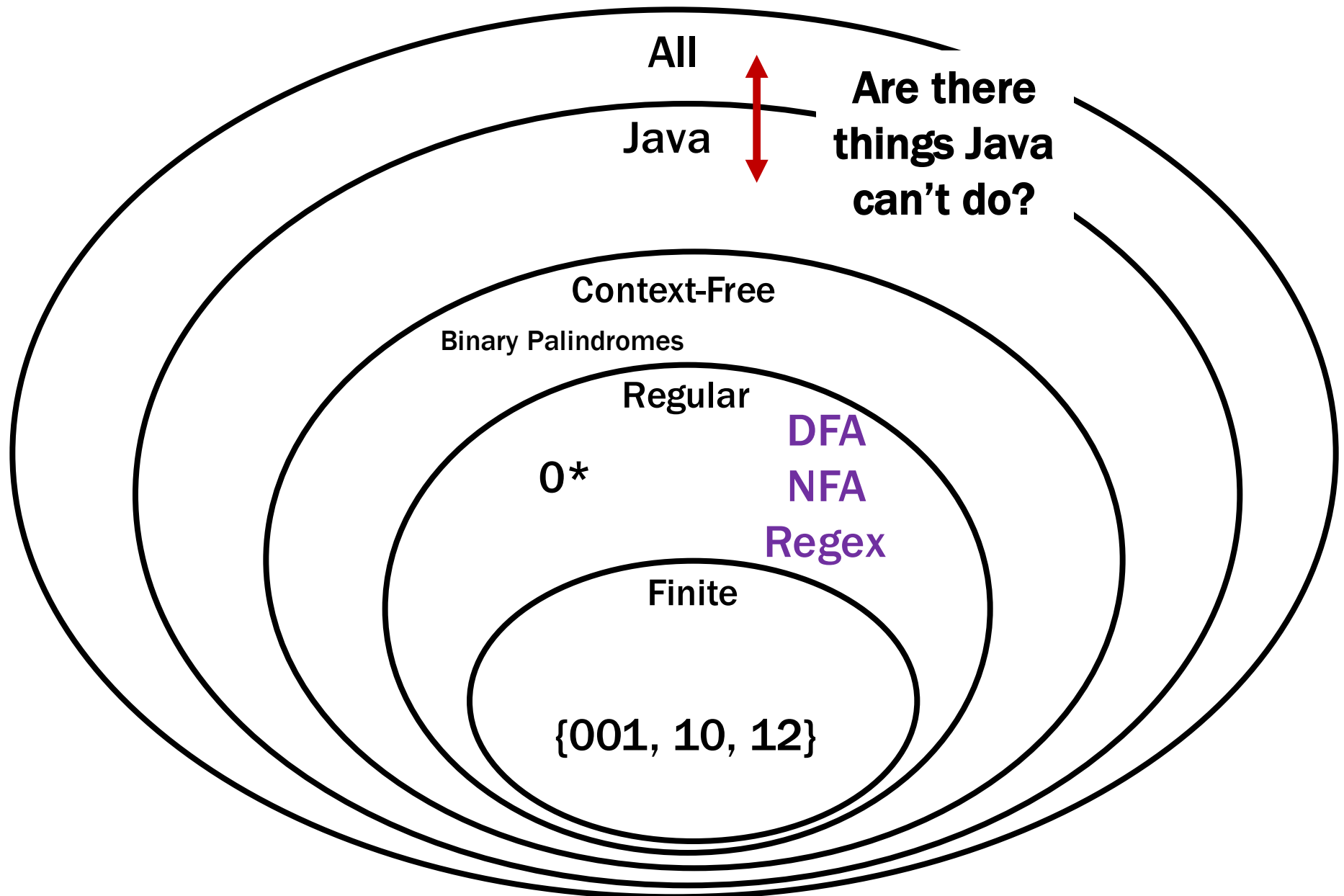
CSE 311: Foundations of Computing

Lecture 25: Limits of Computation

```
DEFINE DOESITHALT(PROGRAM):  
{  
    RETURN TRUE;  
}
```

THE BIG PICTURE SOLUTION
TO THE HALTING PROBLEM

Languages and Machines!



What We're About To Do....

Today, we will dispel the notion that Java is a magical language that allows us to solve any problem we want if we're smart enough.

An Assignment Too Simple for 142!

Students should write a Java program that...

- Prints “Hello” to the console**
- Eventually exits**

Gradelt, Practicelt, etc. need to grade the students.

How do we write that grading program?

Follow Up Question

What does this program do?

```
_( __, __, __ ) { __ / __ <= 1 ? _ ( __, __ + 1, __
_ ) : ! ( __ % __ ) ? _ ( __, __ + 1, 0 ) : __ % __ == __
/ __ && ! __ ? ( printf ( "%d\t", __ / __ ), _ ( __, _
__ + 1, 0 ) ) : __ % __ > 1 && __ % __ < __ / __ ? _ ( __, 1 +
__, __ + ! ( __ / __ % ( __ % __ ) ) ) : __ < __ * __
? _ ( __, __ + 1, __ ) : 0 ; } main () { _ ( 100, 0, 0 ) ; }
```

Follow Up Question

```
public static int collatz(n) {  
    if (n == 1) {  
        return 1;  
    }  
    if (n % 2 == 0) {  
        return collatz(n/2)  
    }  
    else {  
        return collatz(3n + 1)  
    }  
}
```

What is in the set $\{ x : \text{collatz}(x) = 1 \}$?

Some Notation and Starting Ideas

We're going to be talking about *Java code* a lot.

CODE(P) will mean “the code of the program P”

So, consider the following function:

```
public String P(String x) {  
    return new String(Arrays.sort(x.toCharArray()));  
}
```

What is $P(\text{CODE}(P))$?

“((()))..;AACPSSaaabceeggghiiiiInnnnnnooprrrrrrrrrrrrssstttttuuwxyy{ }”

The Halting Problem

Given:

- $\text{CODE}(\mathbf{P})$ for a program \mathbf{P}

wdA h'()E

while (true)?

Output:

- true if \mathbf{P} halts
- false if \mathbf{P} does not halt

}

$\text{HALT}(\text{CODE}(\mathbf{H})) =$

The “standard” version of the halting problem takes some number as input. We consider this one, because it’s easier to think about.

Proof Strategy

Remember, this means X is a program and HALT(X) is true when X halts and false otherwise.

Imagine we had a **HALT(X)** function which solved the halting problem...

Our goal is to write a program that **CONFUSES** the function HALT so that it does the wrong thing.

```
public static void PROGRAM() {  
    if (! halt) {  
        don't finish  
    }  
    else {  
        do finish  
    }  
}
```

Proof Strategy

Remember, this means X is a program and $\text{HALT}(X)$ is true when X halts and false otherwise.

Imagine we had a $\text{HALT}(X)$ function which solved the halting problem...

Our goal is to write a program that **CONFUSES** the function HALT so that it does the wrong thing.

```
public static void PROGRAM() {  
    if (/* I should halt */) {  
        /* don't halt */ while (true);  
    }  
    else {  
        /* halt */ return  
    }  
}
```

Proof Strategy

Remember, this means X is a program and $\text{HALT}(X)$ is true when X halts and false otherwise.

Imagine we had a $\text{HALT}(X)$ function which solved the halting problem...

Our goal is to write a program that **CONFUSES the function **HALT** so that it does the wrong thing.**

```
public static void PROGRAM() {  
    if (/* I should halt */) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

Proof Strategy

Remember, this means X is a program and $\text{HALT}(X)$ is true when X halts and false otherwise.

Imagine we had a $\text{HALT}(X)$ function which solved the halting problem...

Our goal is to write a program that **CONFUSES the function **HALT** so that it does the wrong thing.**

```
public static void PROGRAM() {  
    if (HALT(MY_SOURCE_CODE)) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

Proof Strategy

Remember, this means X is a program and HALT(X) is true when X halts and false otherwise.

Suppose for contradiction we had a HALT(X) function which solved the halting problem...

```
public static void P(String input) {  
    if (HALT(input)) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

P(MySourceCode)
↳ *P(CODE(P(MySourceCode)))*

Quick Question. What does this do?

OnlySourceCodeGENERATOR(P)

Proof Strategy

Remember, this means X is a program and HALT(X) is true when X halts and false otherwise.

Suppose for contradiction we had a HALT(X) function which solved the halting problem...

```
public static void P(String input) {  
    if (HALT(input)) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

↳ **Does POnMySourceCode halt?**
↳ *Does P(CODE(POMSC)) halt?*

```
public static void HALT(String input) {  
    // We don't know how this works,  
    // but we assume that it does.  
  
    // So, if input is a program that  
    // halts, then this returns true.  
    // Otherwise, it returns false.  
}
```

```
void P(String input) {  
    if (HALT(input)) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

Does **P**OnMySourceCode halt?

Recall that **P**OnMySourceCode does the same thing as **P**(**CODE**(**P**OnMySourceCode)).

```
void POnMySourceCode() {  
    if (HALT(CODE(POnMySourceCode))) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

```
public static void HALT(String input) {  
    // We don't know how this works,  
    // but we assume that it does.  
  
    // So, if input is a program that  
    // halts, then this returns true  
    // Otherwise, it returns false.  
}
```

Suppose **POnMySourceCode** halts.

HALT(CODE(**POnMySourceCode**)) is true.

```
void POnMySourceCode() {  
    if (HALT(CODE(POnMySourceCode))) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

So, this if statement is true!

So, the code loops forever!

This is a contradiction, so **POnMySourceCode** does not halt.


```
public static void HALT(String input) {  
    // We don't know how this works,  
    // but we assume that it does.  
  
    // So, if input is a program that  
    // halts, then this returns true.  
    // Otherwise, it returns false.  
}
```

Suppose **POnMySourceCode** does not halt.

HALT(CODE(**POnMySourceCode**)) is false.

```
void POnMySourceCode() {  
    if (HALT(CODE(POnMySourceCode))) {  
        while (true);  
    }  
    else {  
        return;  
    }  
}
```

So, this if statement is false!

So, the code halts!

This is a contradiction, so **POnMySourceCode** can't not halt.

Suppose for contradiction we had a **HALT(X)** function which solved the halting problem...

Suppose **POnMySourceCode** halts.

Then, **HALT(CODE(POnMySourceCode))** is true.

So, the if statement in **POnMySourceCode** is true!

So, the code loops forever!

This is a contradiction, so **POnMySourceCode** does not halt.

Suppose **POnMySourceCode** does not halt.

Then, **HALT(CODE(POnMySourceCode))** is false.

So, the if statement in **POnMySourceCode** is true!

So, the code loops forever!

This is a contradiction, so **POnMySourceCode** can't not halt.

So, **POnMySourceCode**. So, **P** does not exist. So, **HALT** does not exist.

That's it!

- **We proved that there is no Java program that can solve the Halting Problem.**
- **This tells us that there is no compiler that can check our programs and guarantee to find any infinite loops they might have.**

~~That's it!~~

BUT WAIT...

THERE'S MORE!

```
public static void D(String input) {  
    // Returns true if, when run, input  
    // does X prints hello, then hello  
    // Otherwise, it returns false.  
}
```

```
void P(String input) {  
    if (D(input)) {  
        D_IS_FALSE(); while (true)  
    }  
    else {  
        D_IS_TRUE(); // prints 'hello'  
        return  
    }  
}
```

Is **D**(**P**OnMySourceCode) true?

Recall that **P**OnMySourceCode does the same thing as **P**(**CODE**(**P**OnMySourceCode)).

```
void POnMySourceCode() {  
    if (D(CODE(POnMySourceCode))) {  
        D_IS_FALSE();  
    }  
    else {  
        D_IS_TRUE();  
    }  
}
```

Rice's Theorem

- We've now proven that for any property about the “behavior” of programs, D , if...
 - There is some program $D_IS_FALSE()$ for which D is false.
 - There is some program $D_IS_TRUE()$ for which D is true.
- Then, D does not exist.

Rice's Theorem

- Does P have a `NullPointerException`?
- Do P and Q do the same thing?
- Does P output 0 on any input?
- Does P have a buffer overflow?
- Does P have a virus?
- Does P have “dead code”?
- ...

~~That's it!~~

BUT WAIT...

THERE'S MORE!

Church-Turing Thesis

“All physically computable functions are Java-decidable”

That is, there is no programming language more powerful than Java.