Week 10 Tutorial

Beyond R and RE

Please evaluate this course on Axess.

Your feedback really makes a difference.

Part 1: ***Self-Reference***

An Undecidable Problem

- A ***nontrivial*** language is a language that isn't Ø and isn't Σ*.
- Consider the following language:

$$L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$

- This language is undecidable. Our goal is to prove this is the case.

$$L = \{\ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset \text{, and } \mathscr{L}(M) \neq \Sigma^*\ \}$$

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$

(Incorrect!) Theorem: L is decidable.

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$

(Incorrect!) Theorem: L is decidable.

(Incorrect!) Proof: Let M be a Turing machine whose behavior is the same as the program given here:

```
int main() {
    string input = getInput();
    if (input.length() % 2 == 0) {
        accept();
    } else {
        reject();
    }
}
```

Notice that $\mathscr{L}(M) \neq \emptyset$, since M accepts the string ε, and that $\mathscr{L}(M) \neq \Sigma^*$, since M rejects the string `aaa`. Moreover, M is a decider, since given any input the machine M will either accept or reject.

This means that M is a decider, $\mathscr{L}(M) \neq \emptyset$, and $\mathscr{L}(M) \neq \Sigma^*$. Therefore, L is decidable. ■

$L = \{ \langle M \rangle \mid M$ is a TM, $\mathscr{L}(M) \neq \varnothing$, and $\mathscr{L}(M) \neq \Sigma^* \}$

(Incorrect!) Theorem: L is decidable.

(Incorrect!) Proof: Let M be a Turing machine whose behavior is the same as the program given here:

```
int main() {
    string input = getInput();
    if (input.length() % 2 == 0) {
        accept();
    } else {
        reject();
    }
}
```

1. What's wrong with this proof?

Submit your answer on Gradescope.

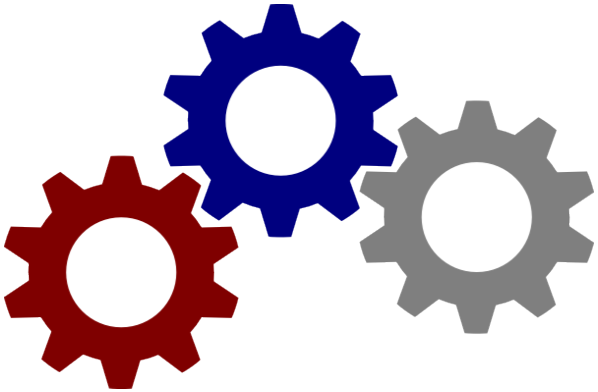
Notice that $\mathscr{L}(M) \neq \varnothing$, since M accepts the string ε, and that $\mathscr{L}(M) \neq \Sigma^*$, since M rejects the string `aaa`. Moreover, M is a decider, since given any input the machine M will either accept or reject.

This means that M is a decider, $\mathscr{L}(M) \neq \varnothing$, and $\mathscr{L}(M) \neq \Sigma^*$. Therefore, L is decidable. ■

Analogy Time!

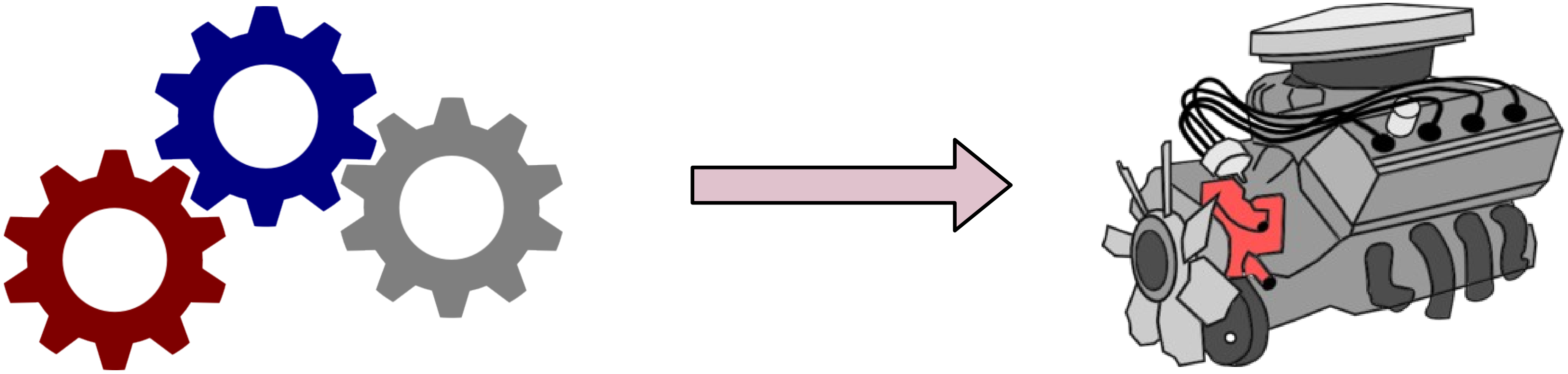
Engineering Problem: Design a diesel engine that doesn’t emit lots of NO_x pollutants.

Engineering Problem: Design a diesel engine that doesn’t emit lots of NO_x pollutants.



Engineering Prowess!

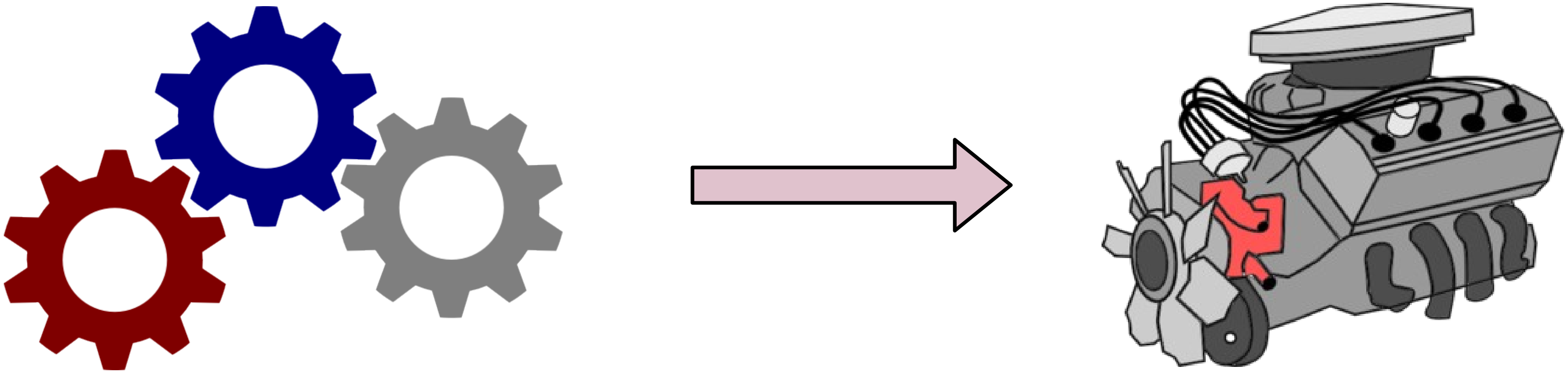
Engineering Problem: Design a diesel engine that doesn’t emit lots of NO_x pollutants.



Engineering Prowess!

Awesome Engine!

Engineering Problem: Design a diesel engine that doesn’t emit lots of NO_x pollutants.

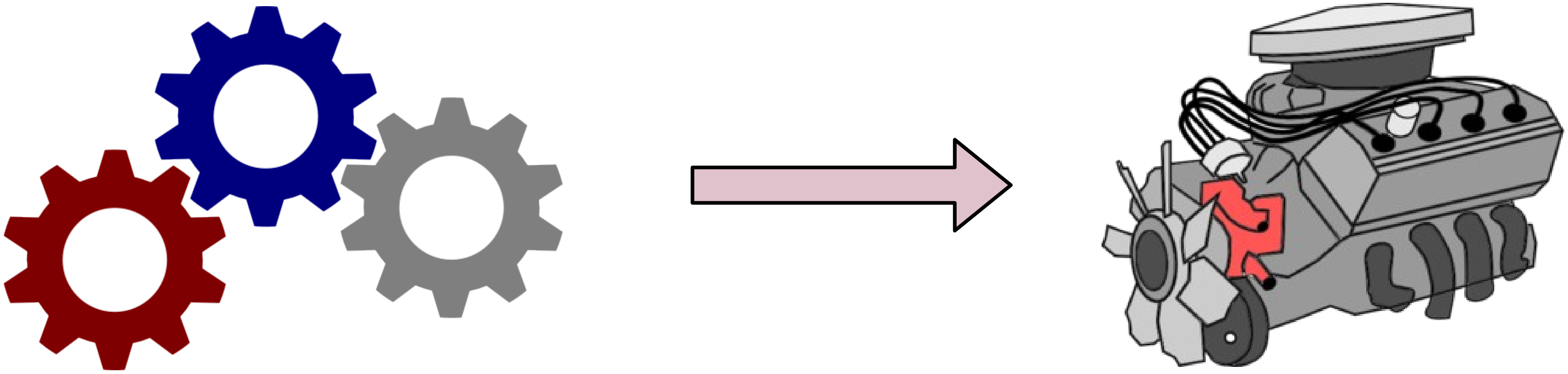


Engineering Prowess!

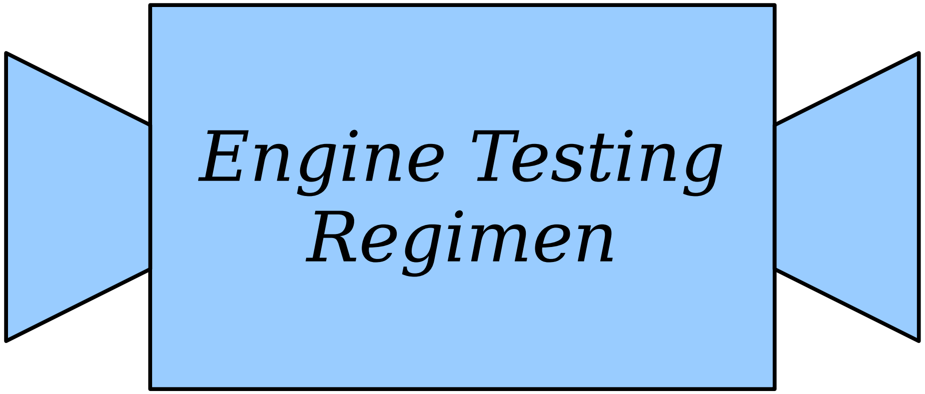
Awesome Engine!

Regulatory Problem: Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO_x pollutants.

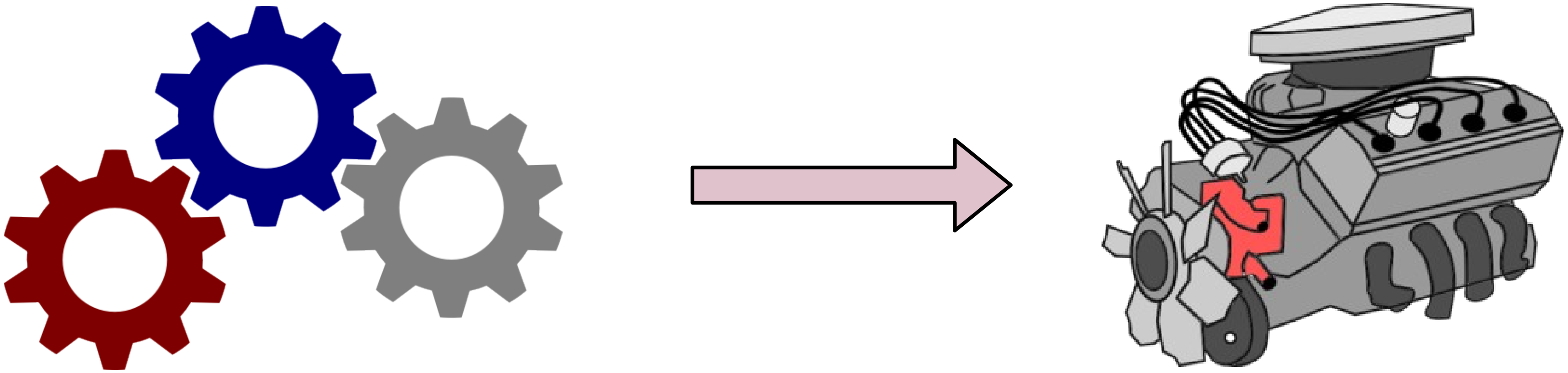
Engineering Problem: Design a diesel engine that doesn't emit lots of NO_x pollutants.



Regulatory Problem: Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO_x pollutants.


Engine Testing Regimen

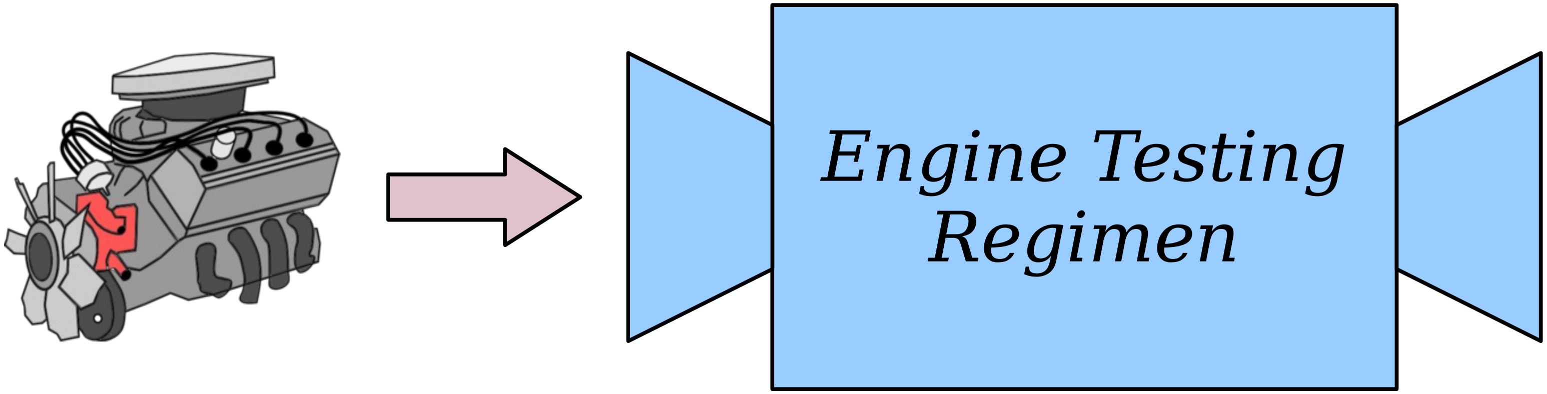
Engineering Problem: Design a diesel engine that doesn't emit lots of NO_x pollutants.



Engineering Prowess!

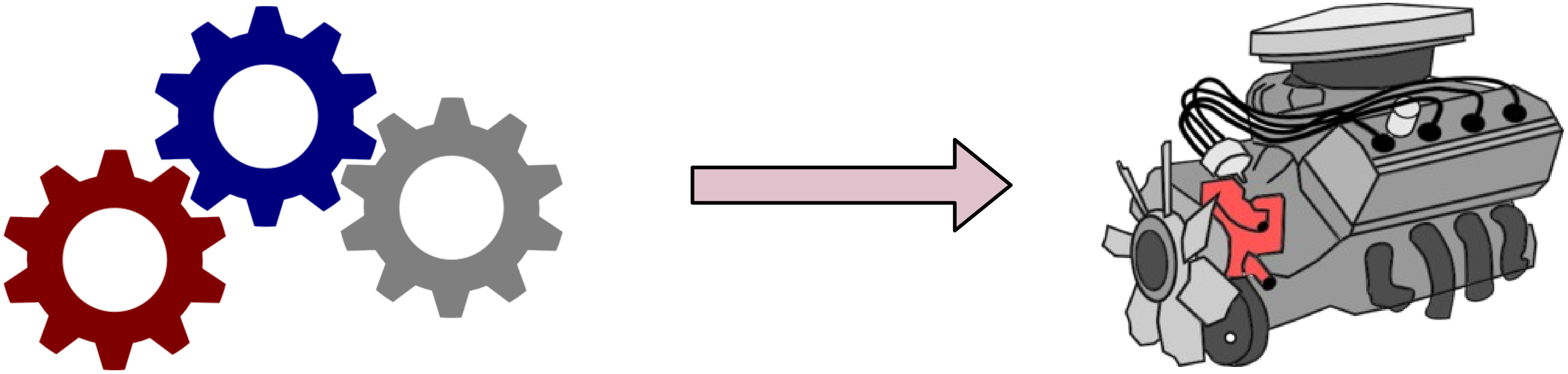
Awesome Engine!

Regulatory Problem: Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO_x pollutants.



Engine Testing Regimen

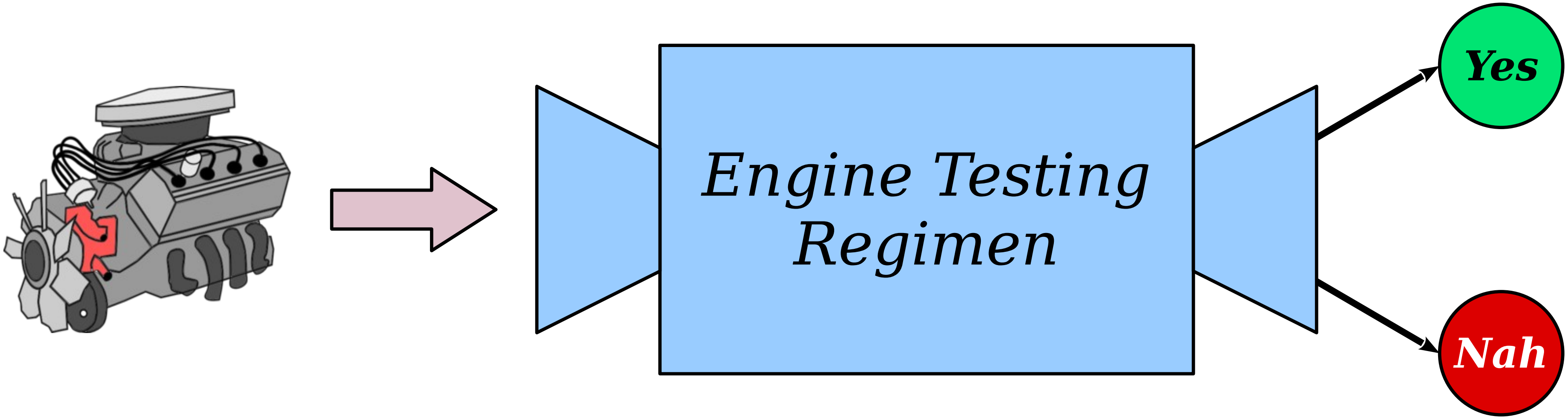
Engineering Problem: Design a diesel engine that doesn't emit lots of NO_x pollutants.



Engineering Prowess!

Awesome Engine!

Regulatory Problem: Design a testing procedure that, given a diesel engine, determines whether it emits lots of NO_x pollutants.



Engine Testing Regimen

Yes

Nah

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$

(Incorrect!) Theorem: L is decidable.

(Incorrect!) Proof: Let M be a Turing machine whose behavior is the same as the program given here:

```
int main() {
    string input = getInput();
    if (input.length() % 2 == 0) {
        accept();
    } else {
        reject();
    }
}
```

Notice that $\mathscr{L}(M) \neq \emptyset$, since M accepts the string ε, and that $\mathscr{L}(M) \neq \Sigma^*$, since M rejects the string `aaa`. Moreover, M is a decider, since given any input the machine M will either accept or reject.

This means that M is a decider, $\mathscr{L}(M) \neq \emptyset$, and $\mathscr{L}(M) \neq \Sigma^*$. Therefore, L is decidable. ■

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$

(Incorrect!) Theorem: L is decidable.

(Incorrect!) Proof: Let M be a Turing machine whose behavior is the same as the program given here:

```
int main() {
    string input = getInput();
    if (input.length() % 2 == 0) {
        accept();
    } else {
        reject();
    }
}
```

Notice that $\mathscr{L}(M) \neq \emptyset$, since M accepts the string ε, and that $\mathscr{L}(M) \neq \Sigma^*$, since M rejects the string aaa. Moreover, M is a decider, since given any input the machine M will either accept or reject.

This means that M is a decider, $\mathscr{L}(M) \neq \emptyset$, and $\mathscr{L}(M) \neq \Sigma^*$. Therefore, L is decidable. ■

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$

(Incorrect!) Theorem: L is decidable.

(Incorrect!) Proof: Let M be a Turing machine whose behavior is the same as the program given here:

```
int main() {
    string input = getInput();
    if (input.length() % 2 == 0) {
        accept();
    } else {
        reject();
    }
}
```

Engineering Problem: Build a TM whose language isn't Ø or Σ*.

Notice that $\mathscr{L}(M) \neq \emptyset$, since M accepts tl
$\mathscr{L}(M) \neq \Sigma^*$, since M rejects the string aa
decider, since given any input the machi
or reject.

This means that M is a decider, $\mathscr{L}(M) \neq \emptyset$, and $\mathscr{L}(M) \neq \Sigma^*$. Therefore, L is decidable. ■

$$L = \underbrace{\{\ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^*\ \}}$$

(Incorrect!) Theorem: L is decidable.

(Incorrect!) Proof: Let M be a Turing mac
the same as the program given here:

```
int main() {
    string input = getInput();
    if (input.length() % 2 == 0) {
        accept();
    } else {
        reject();
    }
}
```

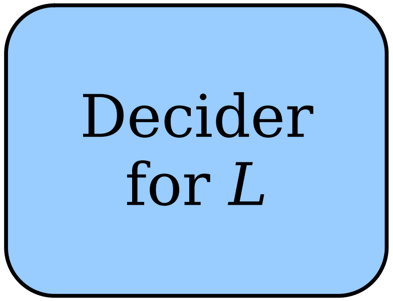
Notice that $\mathscr{L}(M) \neq \emptyset$, since M accepts t
$\mathscr{L}(M) \neq \Sigma^*$, since M rejects the string aaa
decider, since given any input the machi
or reject.

This means that M is a decider, $\mathscr{L}(M) \neq \emptyset$, and $\mathscr{L}(M) \neq \Sigma^*$. Therefore, L is decidable. ■

Regulatory Problem:
Design a procedure to test whether a TM indeed has a language that isn't Ø or Σ*.

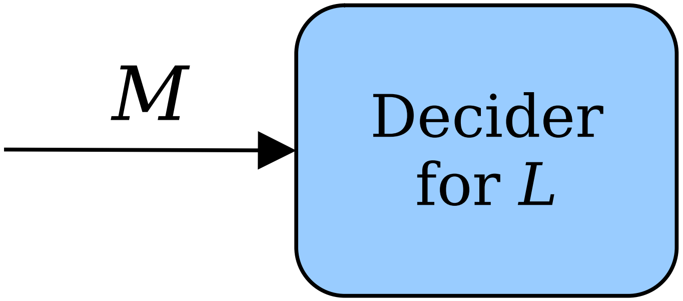
Engineering Problem:
Build a TM whose language isn't Ø or Σ*.

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$

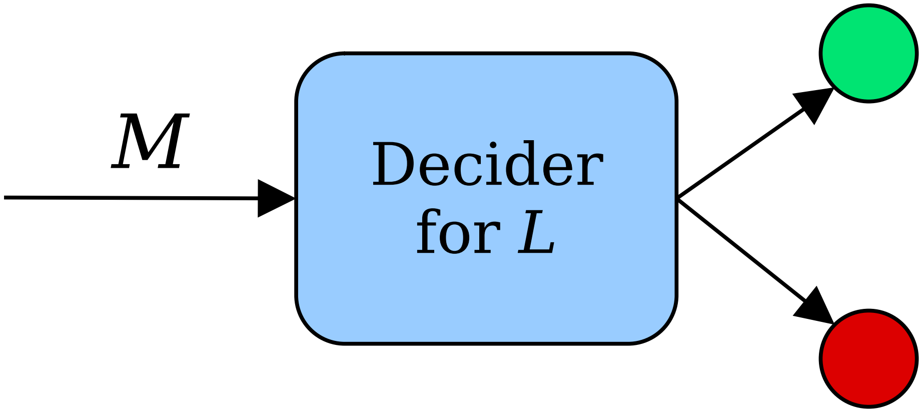


Decider for L

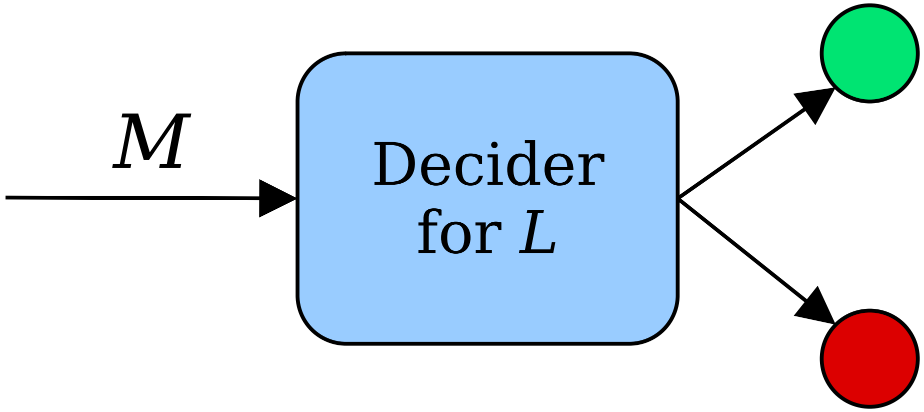
$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$



$L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset \text{, and } \mathscr{L}(M) \neq \Sigma^* \}$

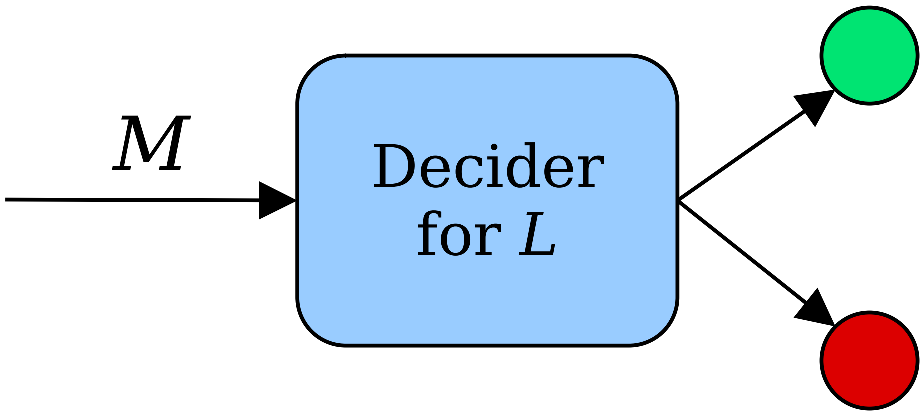


$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$



Yes, M accepts at least one string and does not accept at least one string.

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$



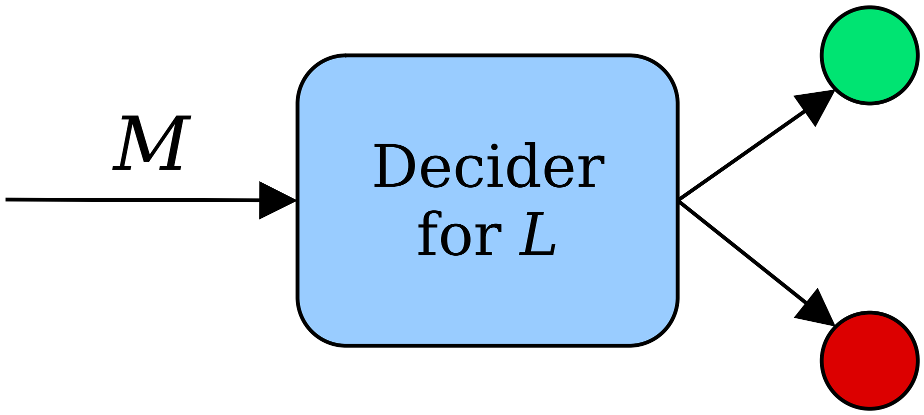
M

Decider for L

Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$



M

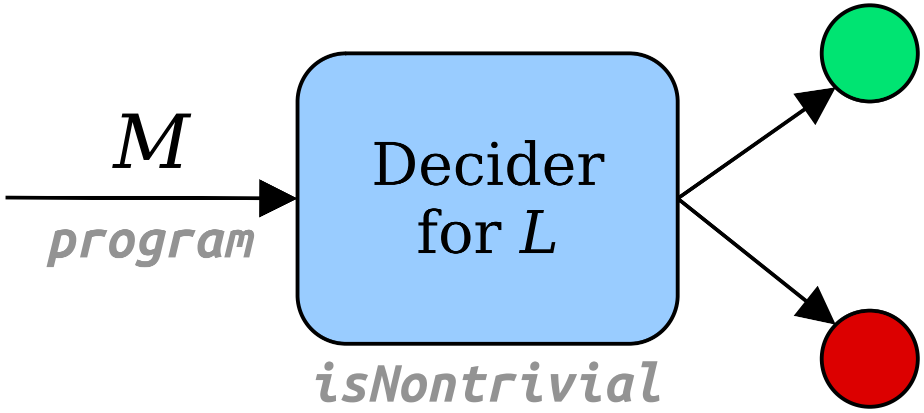
Decider for L

Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$

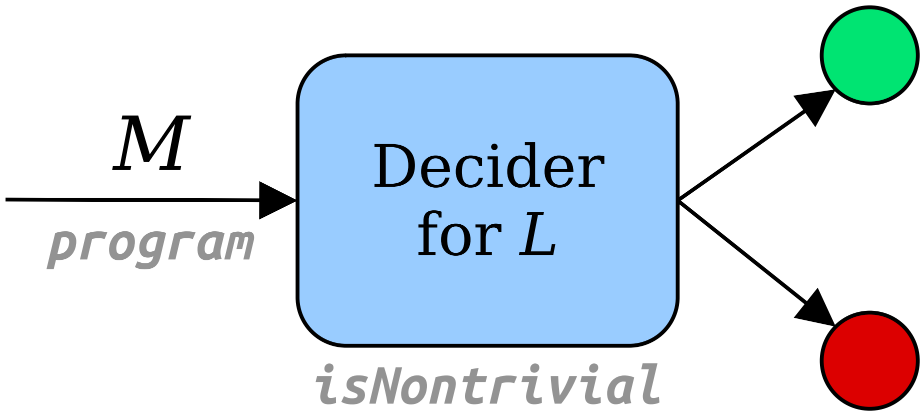


Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

$L = \{ \langle M \rangle \mid M$ is a TM, $\mathscr{L}(M) \neq \emptyset$, and $\mathscr{L}(M) \neq \Sigma^* \}$



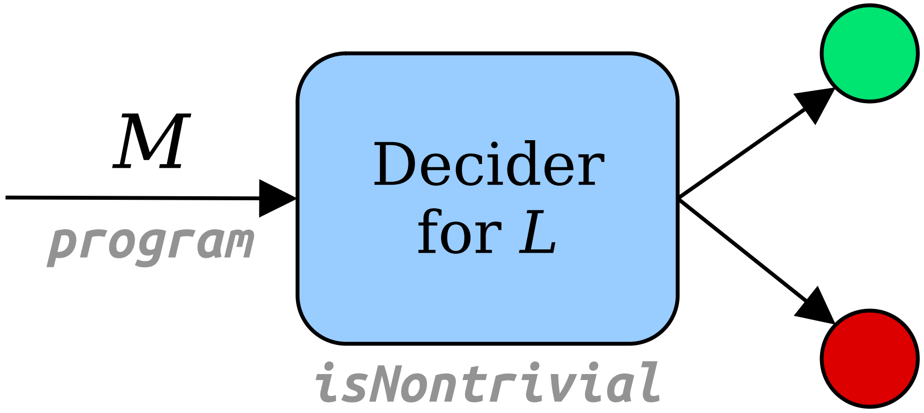
Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

Goal: Use self-reference to show that this decider cannot exist.

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$



M

program

Decider for *L*

isNontrivial

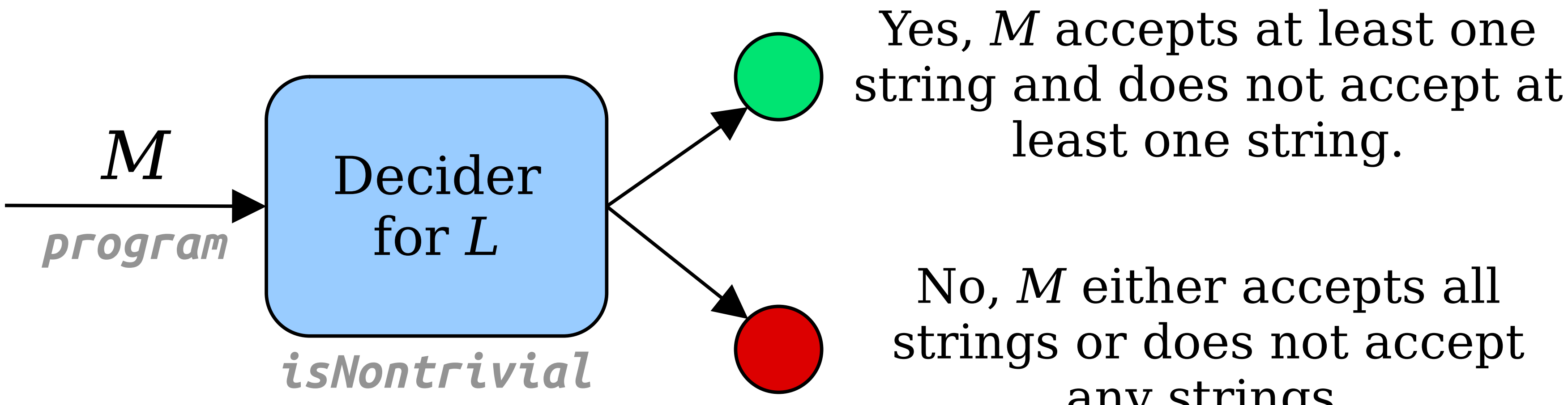
Yes, *M* accepts at least one string and does not accept at least one string.

No, *M* either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

```
// Program P
int main() {










}
```

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{and } \mathscr{L}(M) \neq \Sigma^* \}$$



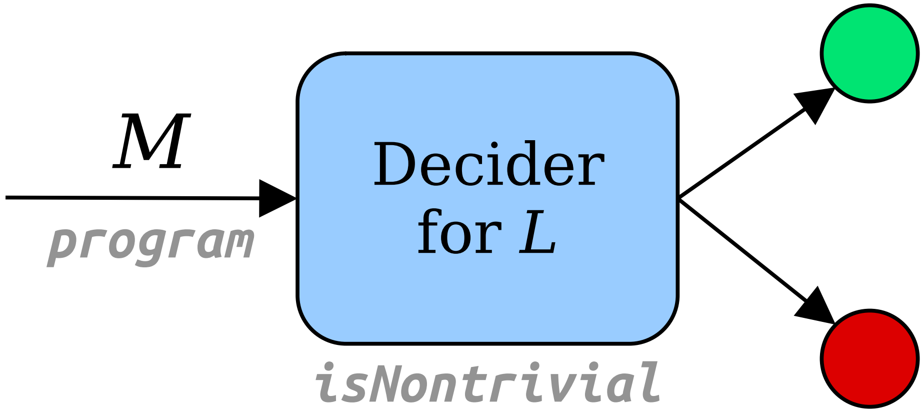
Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

```
// Program P
int main() {
  string input = getInput();










}
```

$$L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset \text{, and } \mathscr{L}(M) \neq \Sigma^* \}$$



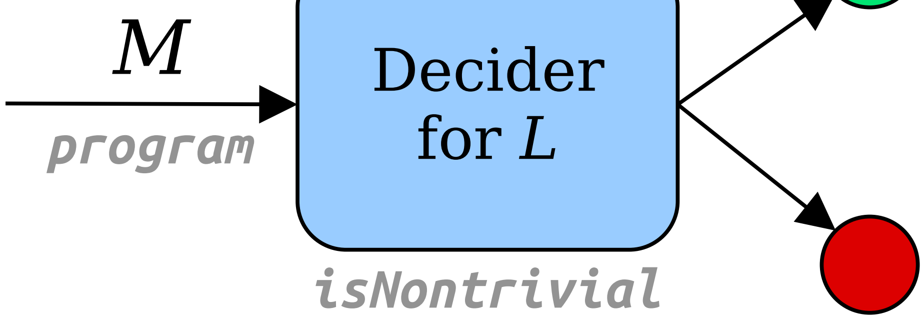
Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

```
// Program P
int main() {
  string input = getInput();
  string me = mySource();



}
```

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \varnothing, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$



M

program

Decider for L

isNontrivial

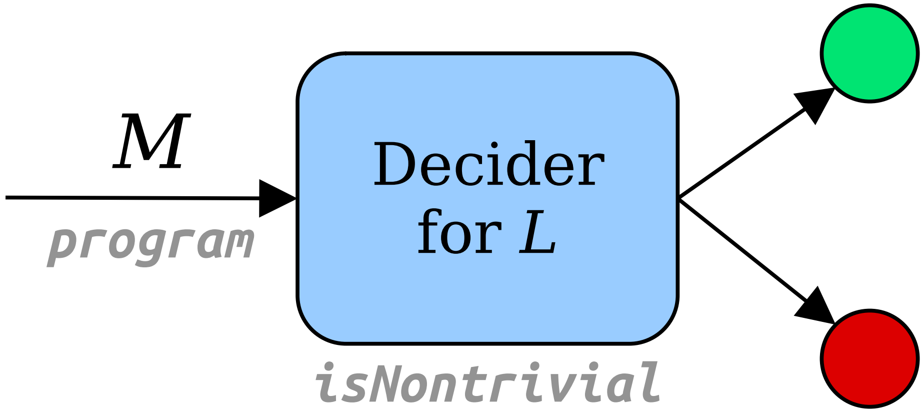
Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

```
// Program P
int main() {
  string input = getInput();
  string me = mySource();
  if (isNontrivial(me)) {


  } else {


  }
}
```

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$



Yes, M accepts at least one string and does not accept at least one string.

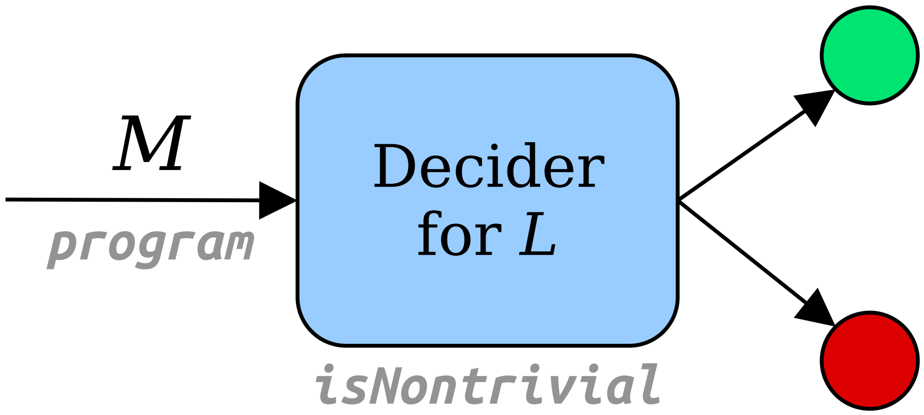
No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

```
// Program P
int main() {
  string input = getInput();
  string me = mySource();
  if (isNontrivial(me)) {


  } else {


  }
}
```

Program P design specification:

$L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset \text{, and } \mathscr{L}(M) \neq \Sigma^* \}$



Yes, *M* accepts at least one string and does not accept at least one string.

No, *M* either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

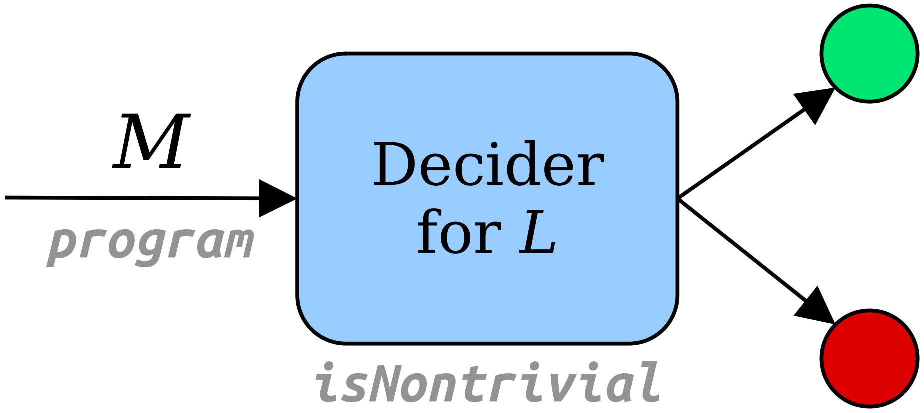
```
// Program P
int main() {
  string input = getInput();
  string me = mySource();
  if (isNontrivial(me)) {


  } else {


  }
}
```

Program P design specification:

If *P* accepts at least one string and doesn’t accept at least one string:

If *P* accepts all strings or does not accept any strings:

$$L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset \text{, and } \mathscr{L}(M) \neq \Sigma^* \}$$



M

program

Decider for *L*

isNontrivial

Yes, *M* accepts at least one string and does not accept at least one string.

No, *M* either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

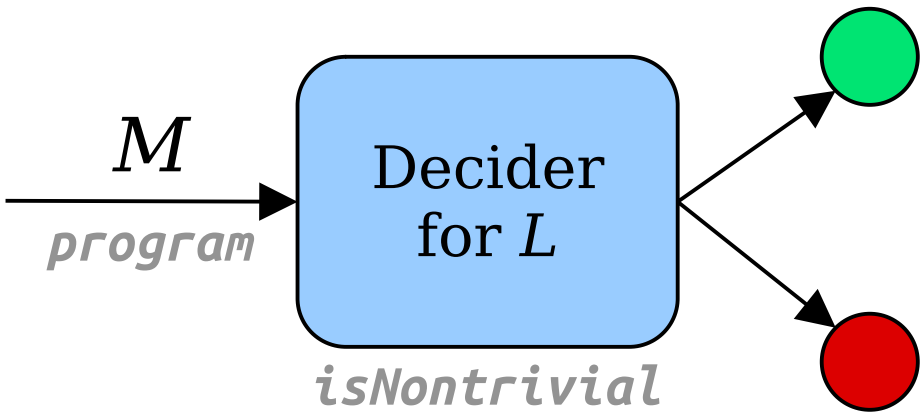
```
// Program P
int main() {
  string input = getInput();
  string me = mySource();
  if (isNontrivial(me)) {


  } else {


  }
}
```

Program P design specification:

If *P* accepts at least one string and doesn't accept at least one string:
    P must accept all strings or accept no strings at all.

If *P* accepts all strings or does not accept any strings:
    P must accept at least one string and not accept at least one string.

$$L = \{ \langle M \rangle \mid M \text{ is a TM, } \mathscr{L}(M) \neq \emptyset \text{, and } \mathscr{L}(M) \neq \Sigma^* \}$$



M

program

Decider for *L*

isNontrivial

Yes, *M* accepts at least one string and does not accept at least one string.

No, *M* either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

```
// Program P
int main() {
  string input = getInput();
  string me = mySource();
  if (isNontrivial(me)) {



  } else {



  }
}
```

Program P design specification:

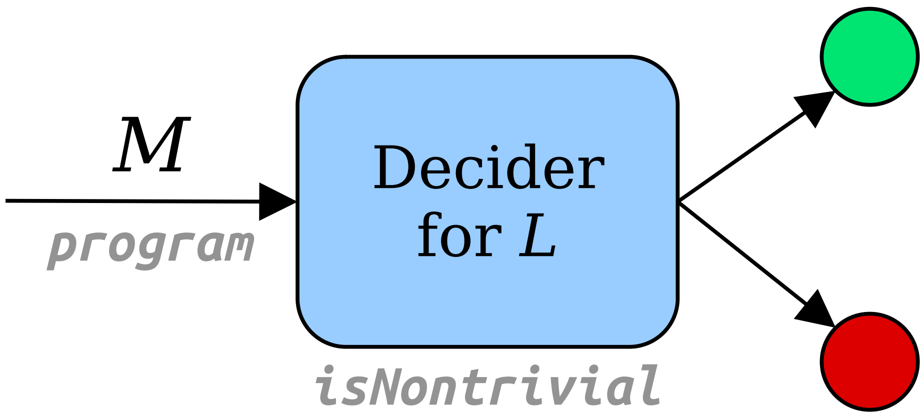
If *P* accepts at least one string and doesn't accept at least one string:
   P must accept all strings or accept no strings at all.

If *P* accepts all strings or does not accept any strings:
   P must accept at least one string and not accept at least one string.

2. Complete program *P*.

Submit your answer on Gradescope.

$$L = \{ \langle M \rangle \mid M \text{ is a TM}, \mathscr{L}(M) \neq \emptyset, \text{ and } \mathscr{L}(M) \neq \Sigma^* \}$$



Yes, M accepts at least one string and does not accept at least one string.

No, M either accepts all strings or does not accept any strings.

```
bool isNontrivial(string program);
```

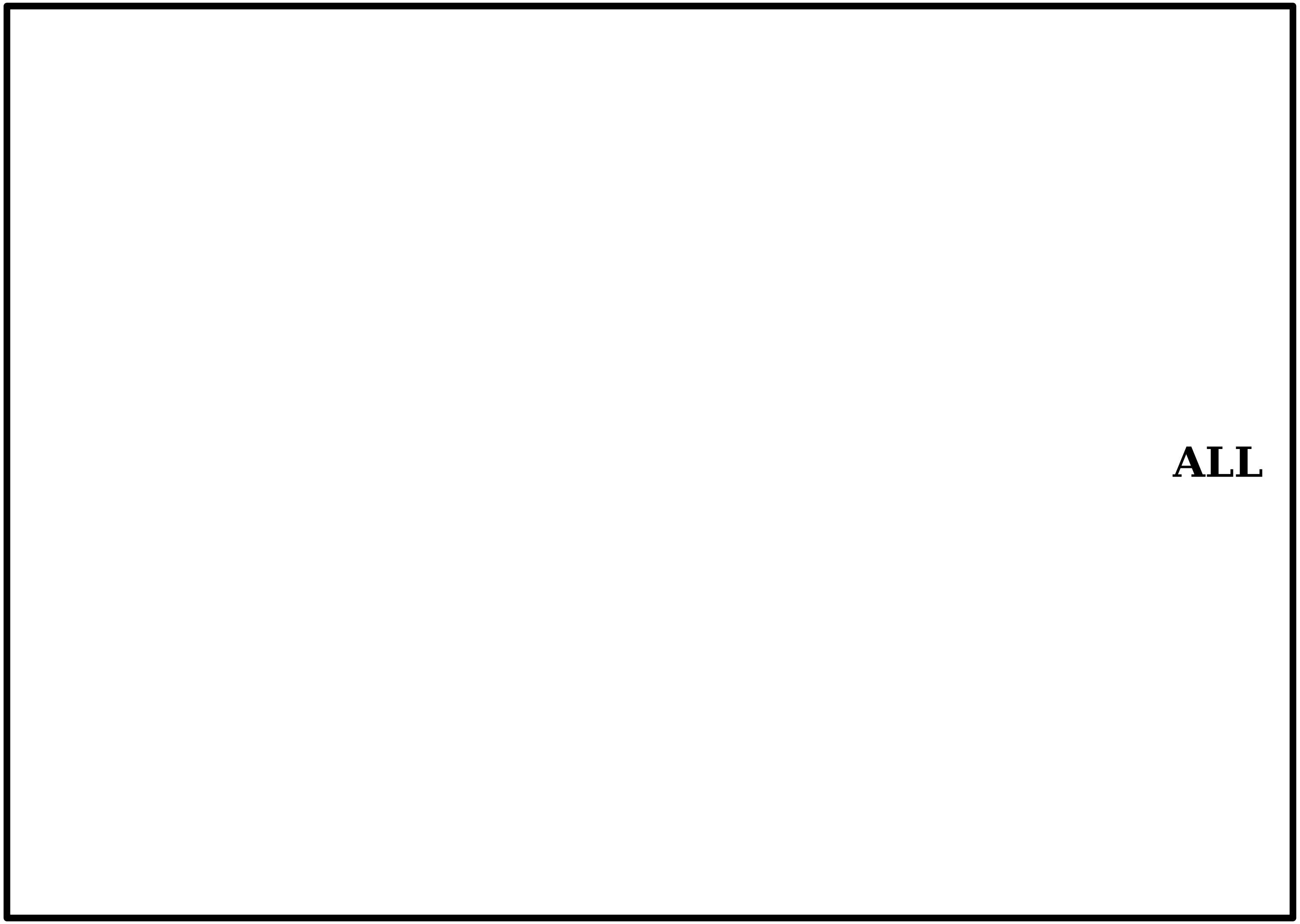
```
// Program P
int main() {
  string input = getInput();
  string me = mySource();
  if (isNontrivial(me)) {



  } else {



  }
}
```

Program P design specification:

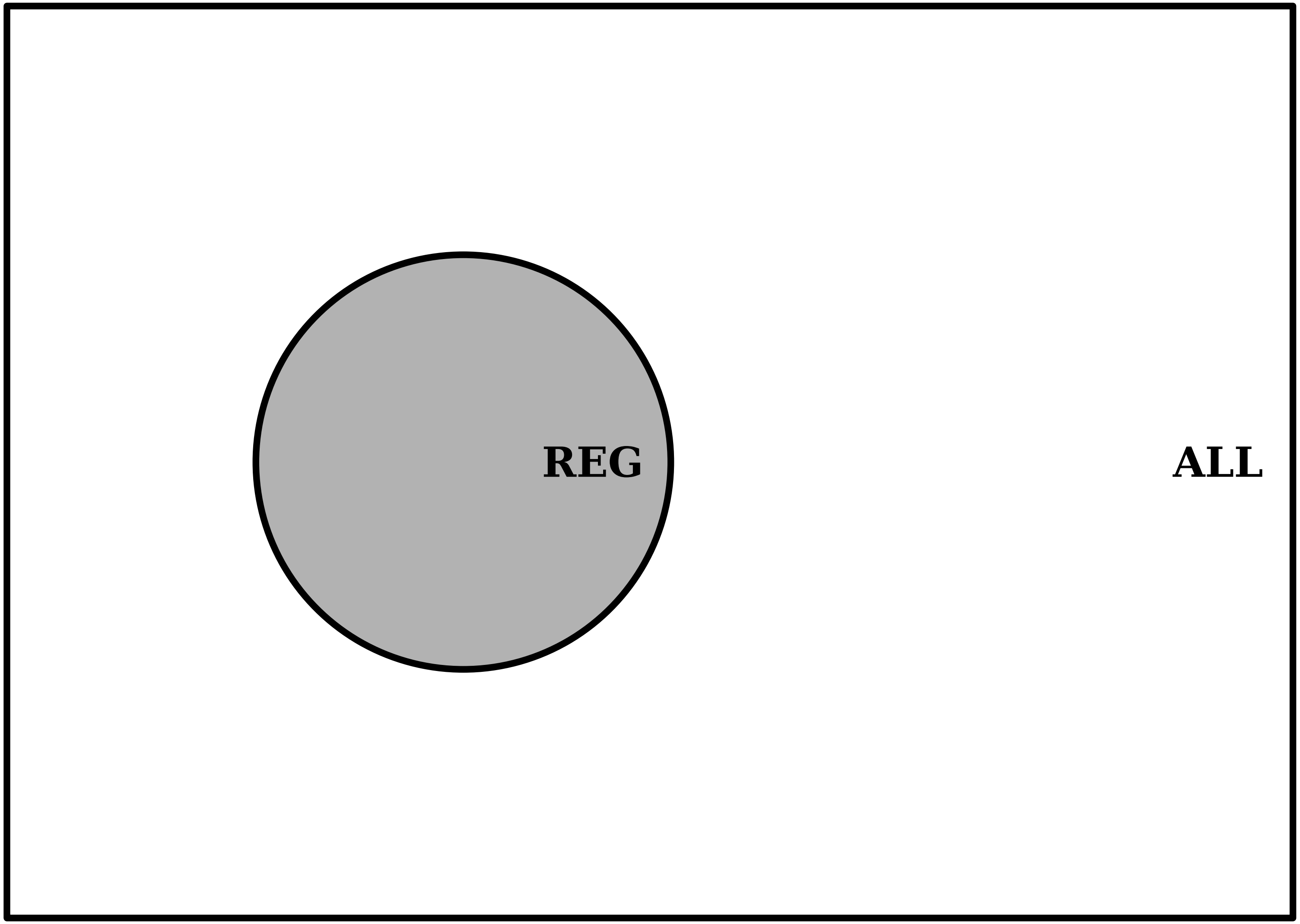
If P accepts at least one string and doesn't accept at least one string:
   P must accept all strings or accept no strings at all.

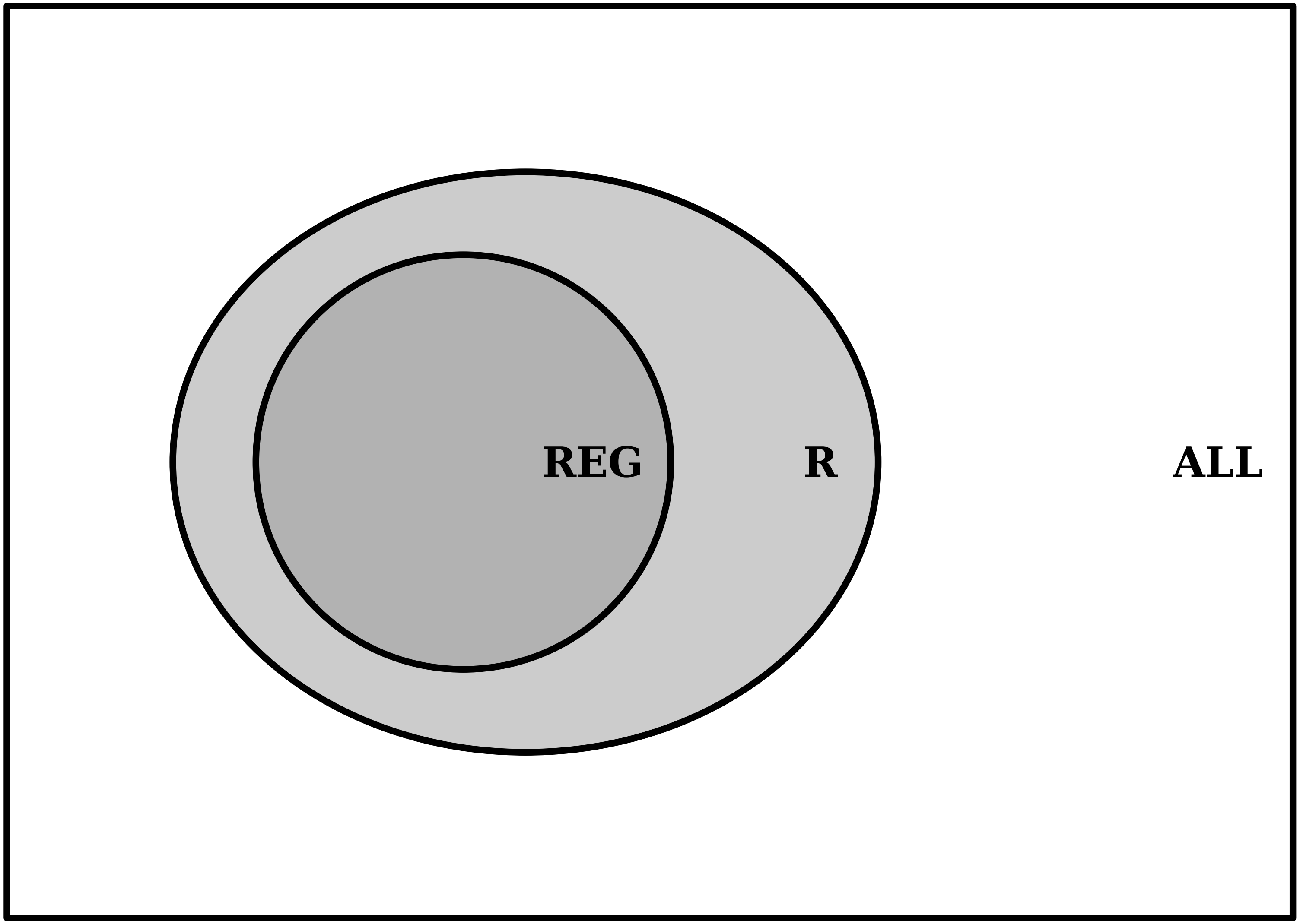
If P accepts all strings or does not accept any strings:
   P must accept at least one string and not accept at least one string.

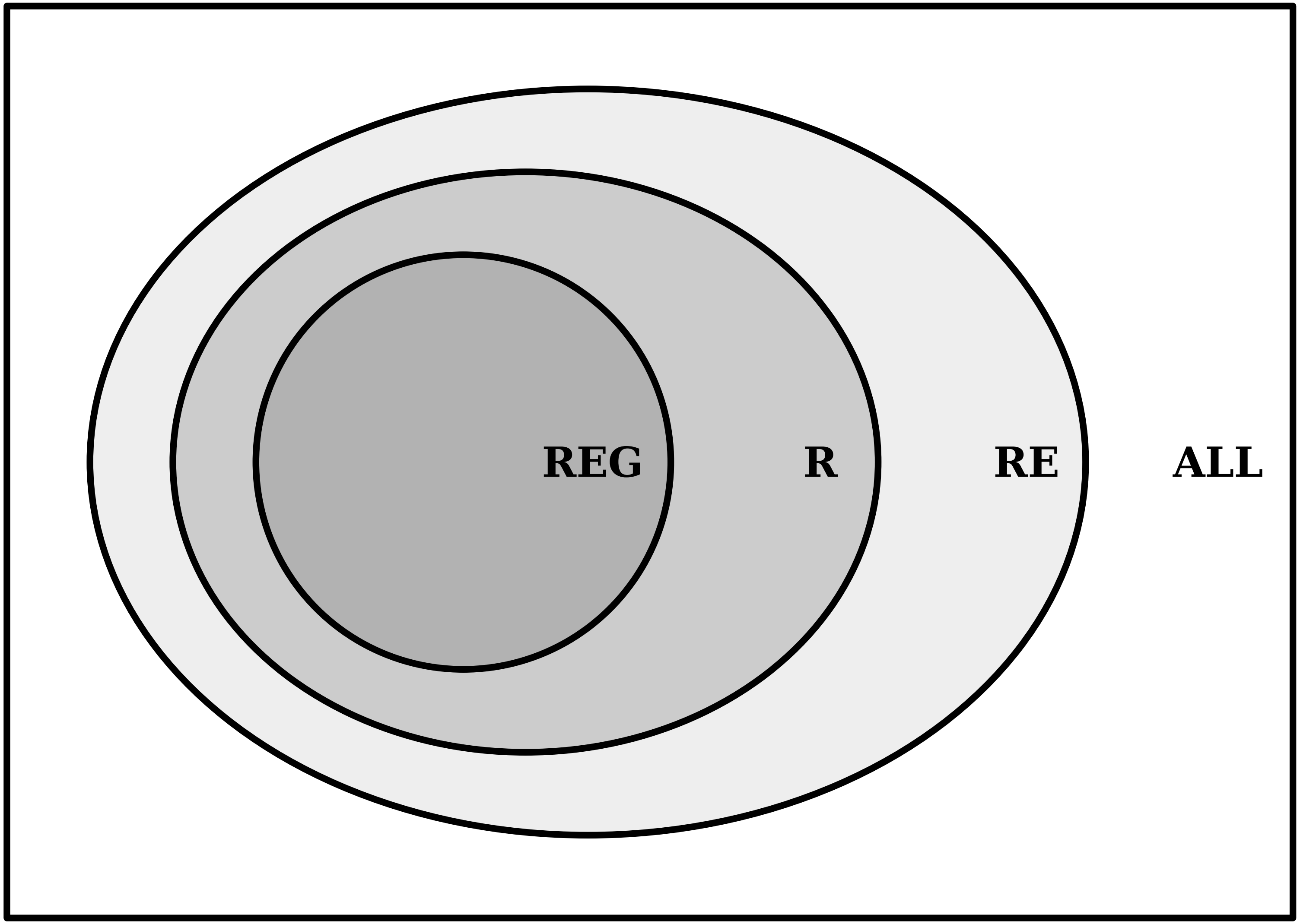
Part 2: ***The Lava Diagram***

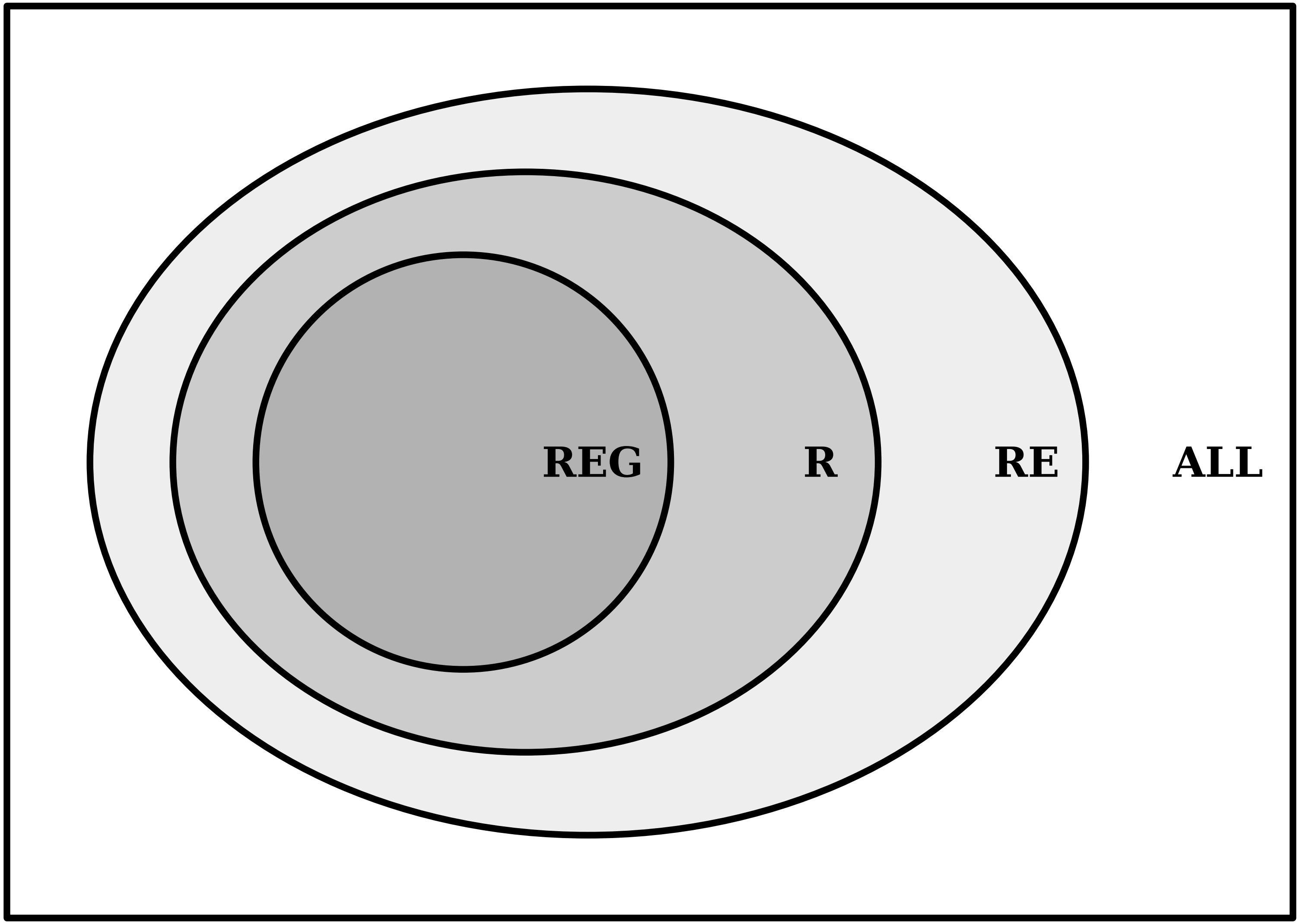


ALL


REG
ALL


REG
R
ALL

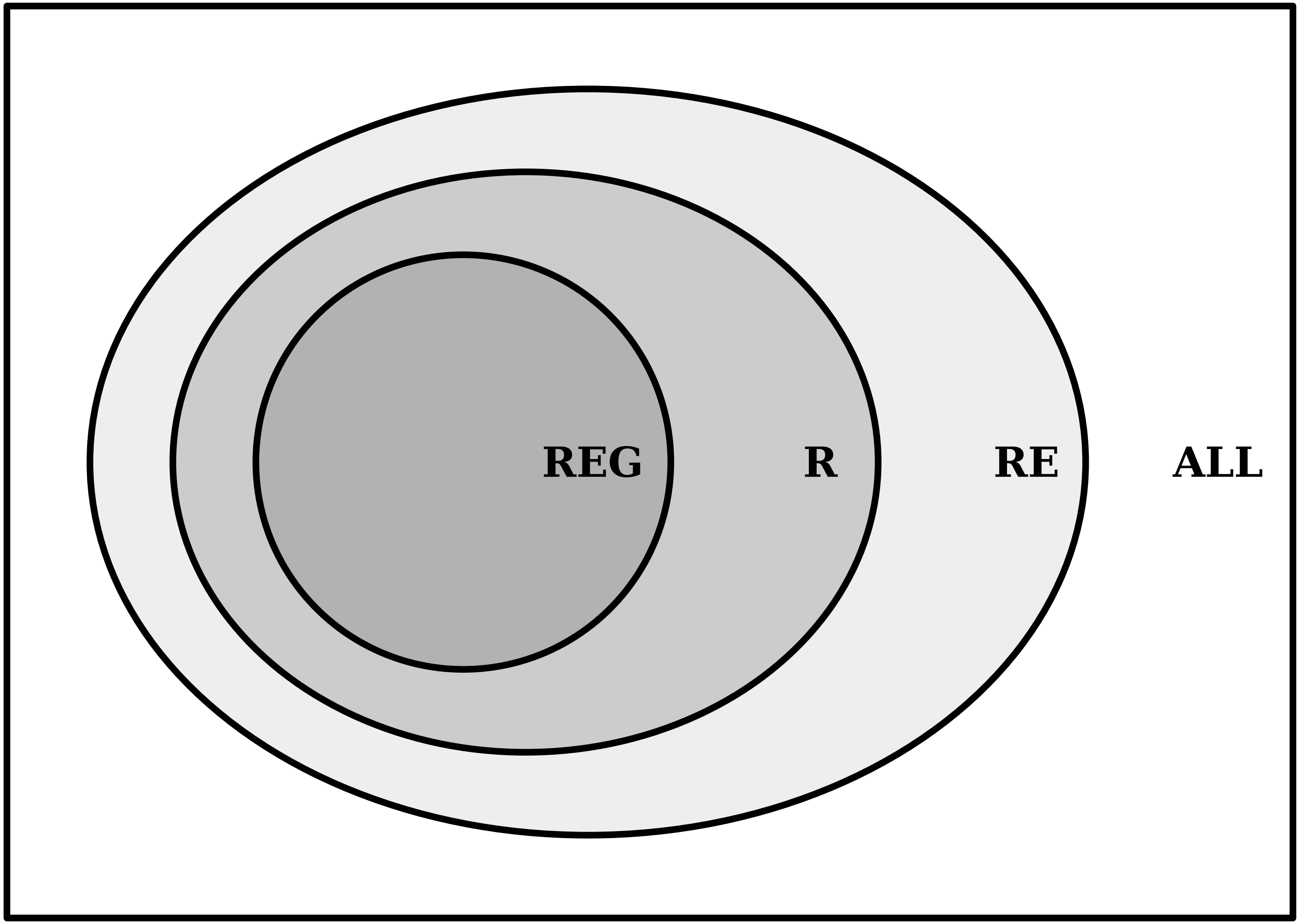

REG
R
RE
ALL



$L_1 = \{ \langle M \rangle \mid M$ is a TM and M accepts cocoa $\}$

$L_2 = \{ \langle M \rangle \mid M$ is a TM and M rejects cocoa $\}$

$L_3 = \{ \langle M \rangle \mid M$ is a TM and M loops on cocoa $\}$



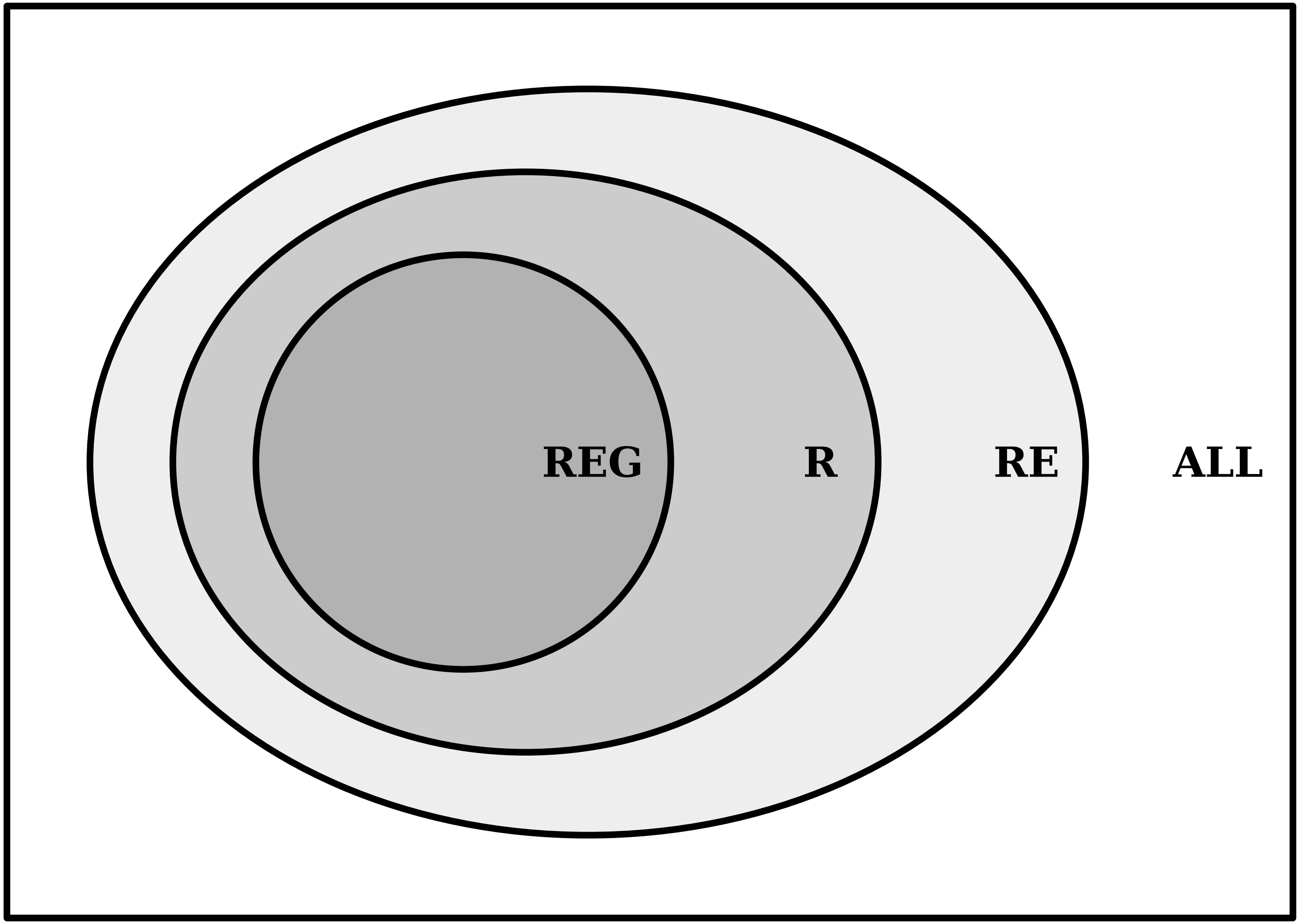
$L_1 = \{ \langle M \rangle \mid M$ is a TM and M accepts cocoa $\}$

$L_2 = \{ \langle M \rangle \mid M$ is a TM and M rejects cocoa $\}$

$L_3 = \{ \langle M \rangle \mid M$ is a TM and M loops on cocoa $\}$

3. Place these languages in the Lava Diagram.

Submit your answer on Gradescope.



$L_1 = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \texttt{cocoa} \}$

$L_2 = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ rejects } \texttt{cocoa} \}$

$L_3 = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ loops on } \texttt{cocoa} \}$

Thanks for Calling In!

It’s been great meeting you this quarter.
Stay safe, stay healthy, and stay in touch!

Enjoy the break!