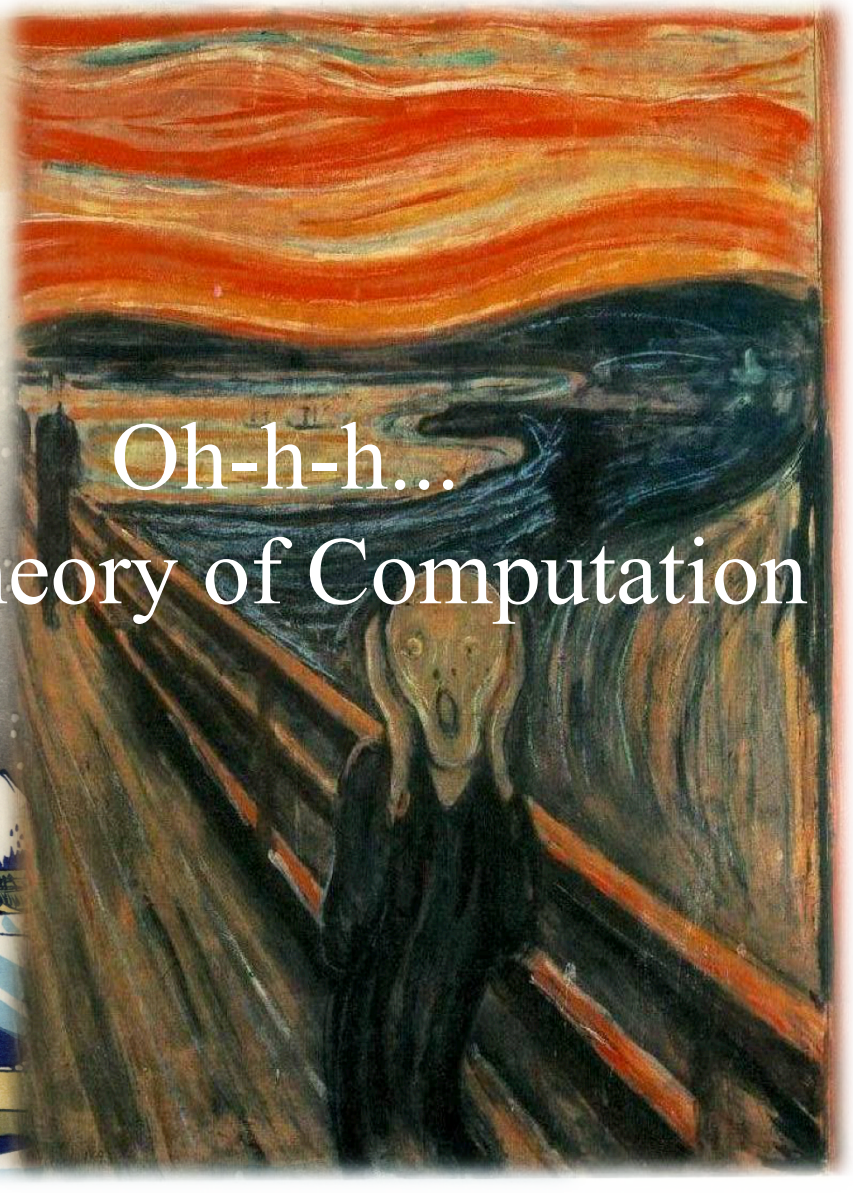
Università
della
Svizzera
italiana

**Facoltà
di scienze
informatiche**



Oh-h-h...
it's Theory of Computation

Today's lecture

- Decidable languages (continued)

Important points from last lecture

- All problems must be presented as language problems
  - What types do we know from Automata theory?
    - acceptance testing
    - emptiness testing
    - equivalence test

Important points from last lecture

- How languages are defined?

- String encoding
- General encoding representing each possible instantiation of the problem

Important points from last lecture

- How is the language decidability established?

- By designing a TM which is a decider

Important points from last lecture

- What does it imply that a language is decidable?

- That the problem it describes is solvable

Important Observation

For decidability purposes, presenting Turing Machines with DFAs, NFAs, or regular expressions are all equivalent because the Turing Machine is able to convert one form of encoding to another

More Automata problems

Now let's turn to a different kind of problem: Equivalence Testing, i.e.,

Checking if two DFAs recognize the same language?

What is the language to deal with?

$$EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$$

Equivalence test

Theorem: EQ_{DFA} is a decidable language

Proof Ideas?

Equivalence test

Theorem: EQ_{DFA} is a decidable language

Proof Ideas?

We need to make a conclusion about strings that are accepted by either automaton but *not by both*

Let's construct a new automaton with this feature

Equivalence test

Theorem: EQ_{DFA} is a decidable language

We need to make a conclusion about strings that are accepted by either automaton but *not by both*

Let's construct a new automaton with this feature

⇒ If two automata recognize the same language then the newly constructed automaton accepts nothing when the languages are the same

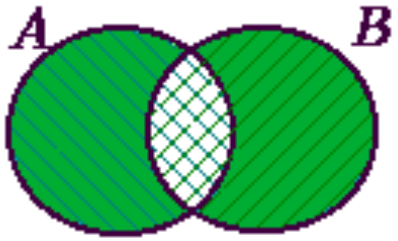
⇒ We can use the emptiness test result (e.g., E_{DFA} is decidable)

Equivalence test

What automaton do we need to construct?

The one with the Symmetric Difference feature

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

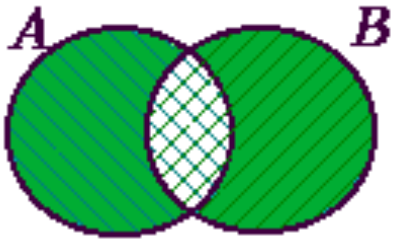


Symmetric difference is green

Symmetric Difference

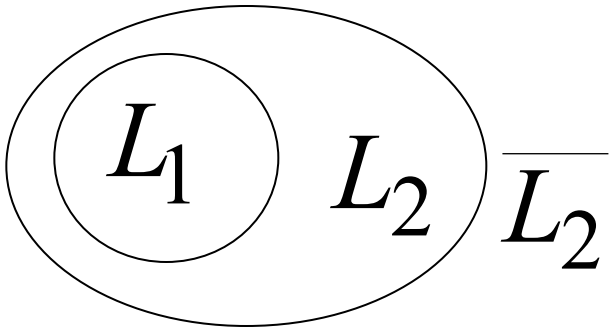
$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

When is $L(C) = \varnothing$?

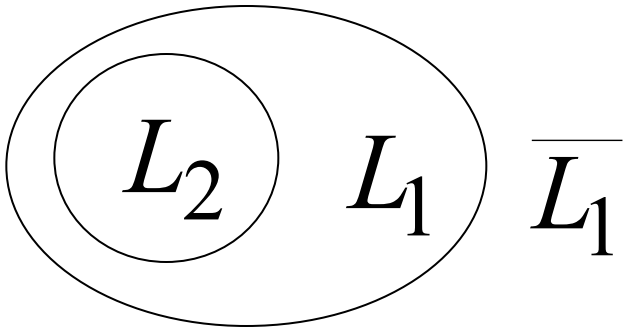


$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

$$L_1 \cap \overline{L_2} = \emptyset \quad \text{and} \quad \overline{L_1} \cap L_2 = \emptyset$$



$$L_1 \subseteq L_2$$

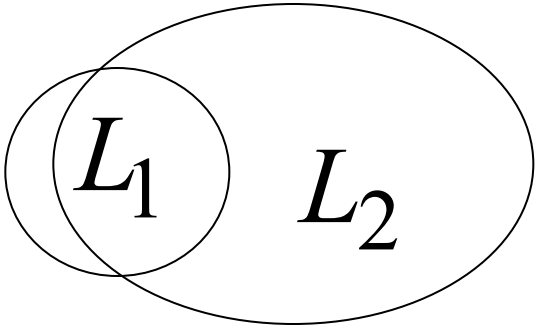


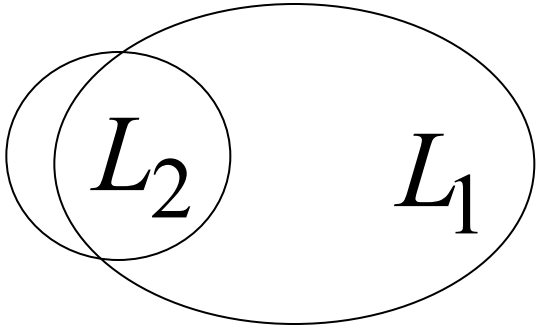
$$L_2 \subseteq L_1$$

$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

$$L_1 \cap \overline{L_2} \neq \emptyset \quad \text{or} \quad \overline{L_1} \cap L_2 \neq \emptyset$$





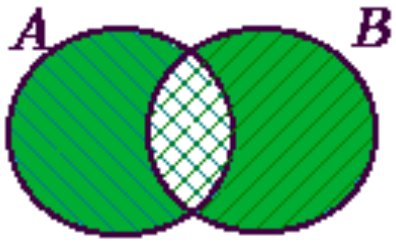
$$L_1 \not\subset L_2 \qquad L_2 \not\subset L_1$$

$$L_1 \neq L_2$$

Symmetric Difference

⇒ If two automata recognize the same language then the newly constructed automaton accepts nothing when the languages are the same

⇒ We can use the emptiness test result (e.g., E_{DFA} is decidable)



Equivalence test

Proof: On input <A,B>, where A and B are DFAs

1. Construct DFA C with symmetric difference feature
   Important: we need to take care of the class of regular languages closed under complementation, union and intersection. It works for regular languages since they are closed under complementation, union, and intersection
2. TM EQ = On input <C>:
   2.1 Mark the start state of C
   2.2 Repeat until no new states get marked:
   - mark any state that has a transition coming into it from any state that is already marked
   2.3 If no accept state is marked, ***accept***; otherwise, ***reject***

Decidability of CFLs

Theorem: A_{CFG} is a decidable language

Proof ideas?

Shall we try all possible derivations from grammar?

NO – bad idea. Why?

Because infinitely many derivations may have to be tried and the corresponding TM would not be a decider

Decidability of CFLs

Theorem: A_{CFG} is a decidable language

Proof: optional extra credit (2% of final grade) due to the next class

Hint: use problem 2.26 from Sipser's book (you need to prove it)

Decidability of CFLs

Are decidability results for CFG the same for PDAs?

YES

Why?

Because push down automata are the devices that recognize context-free languages

Decidability of CFLs

Theorem: E_{CFG} is a decidable language

Proof: ideas?

Decidability of CFLs

Theorem: E_{CFG} is a decidable language

Proof:

On input <G>, where G is a CFG:

1. Mark all terminal symbols in G
2. Repeat until no new variables get marked
   Mark any variable A where G has a rule A → U1U2…Uk and each symbol has already been marked
3. If the start variable is not marked, *accept*; otherwise, *reject*.

Decidability of CFLs

Decidability of EQ_{CFG}

Proof ideas?

Can we reuse the symmetric difference idea as we used to prove EQ_{DFA} ?

NO

CFLs are not closed under complementation or intersection

Decidability of CFLs

Every context-free language is decidable

What are the implications of it? (in terms of practical applications)

- Programming-related operations are under our control as most are modeled as CFLs.

Relationship among classes of languages