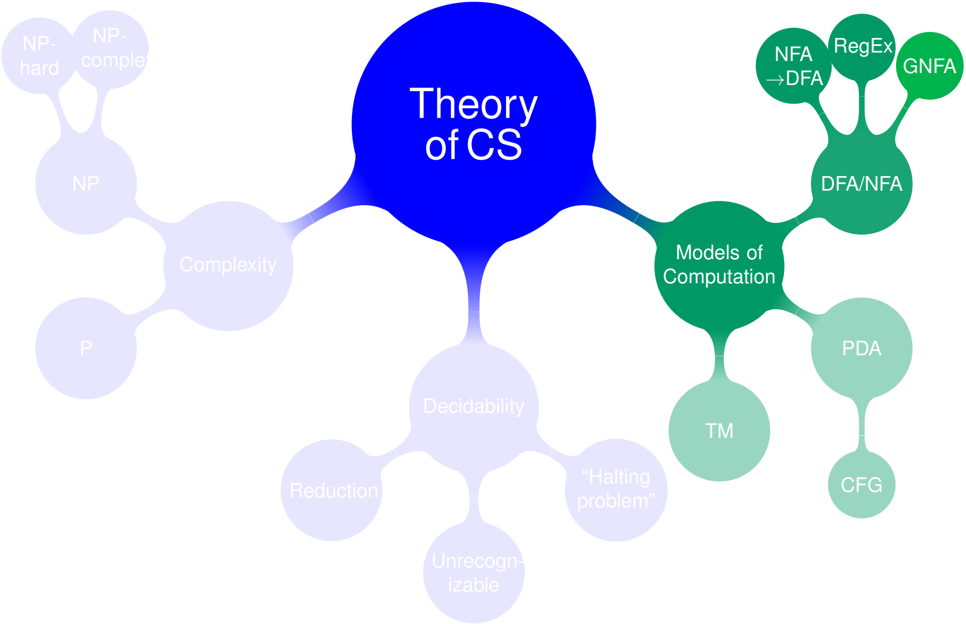
Decidability

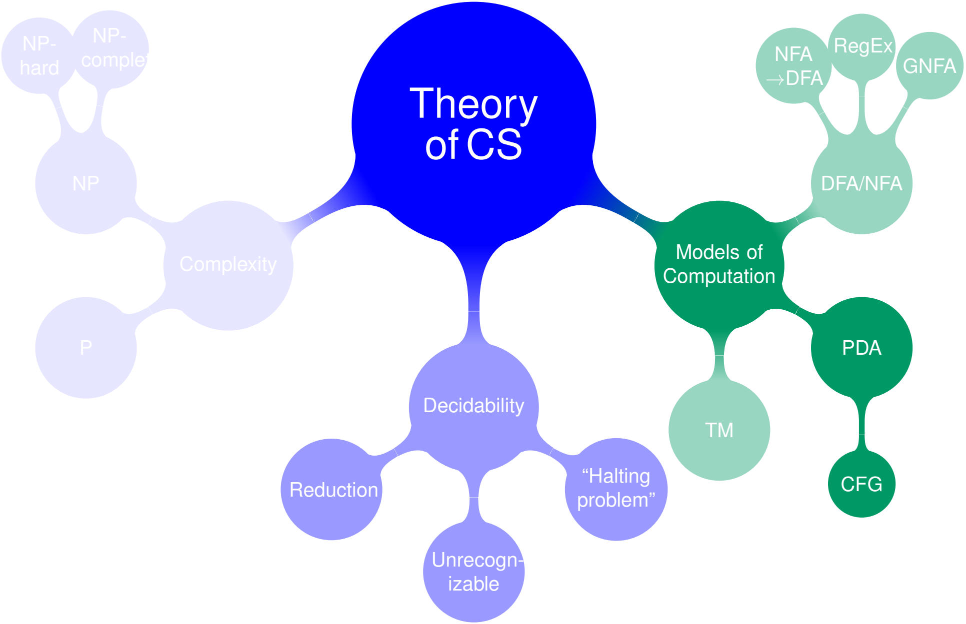
& Undecidability

Dr Kamal Bentahar

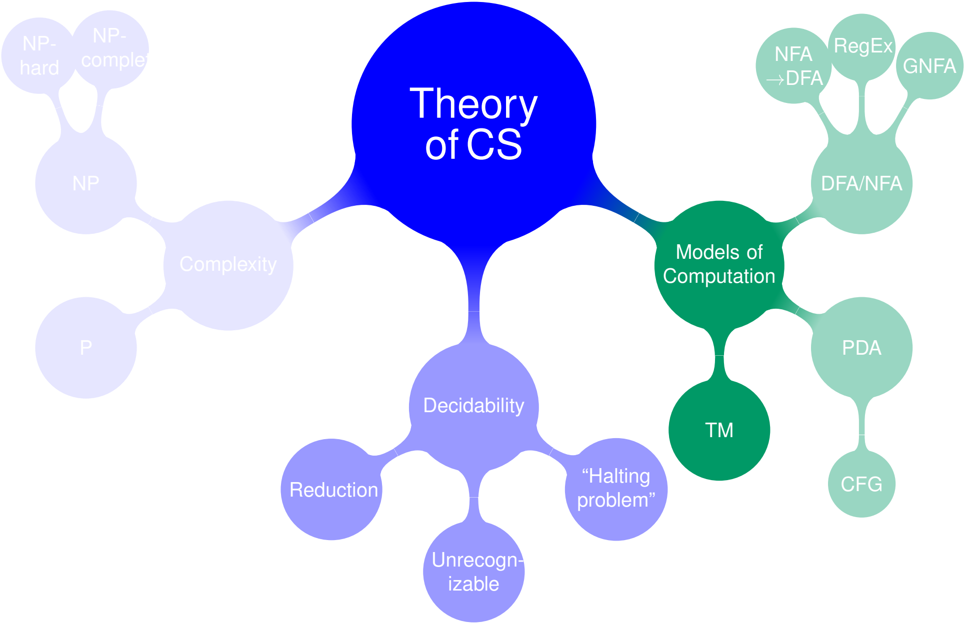
School of Computing, Electronics and Mathematics
Coventry University

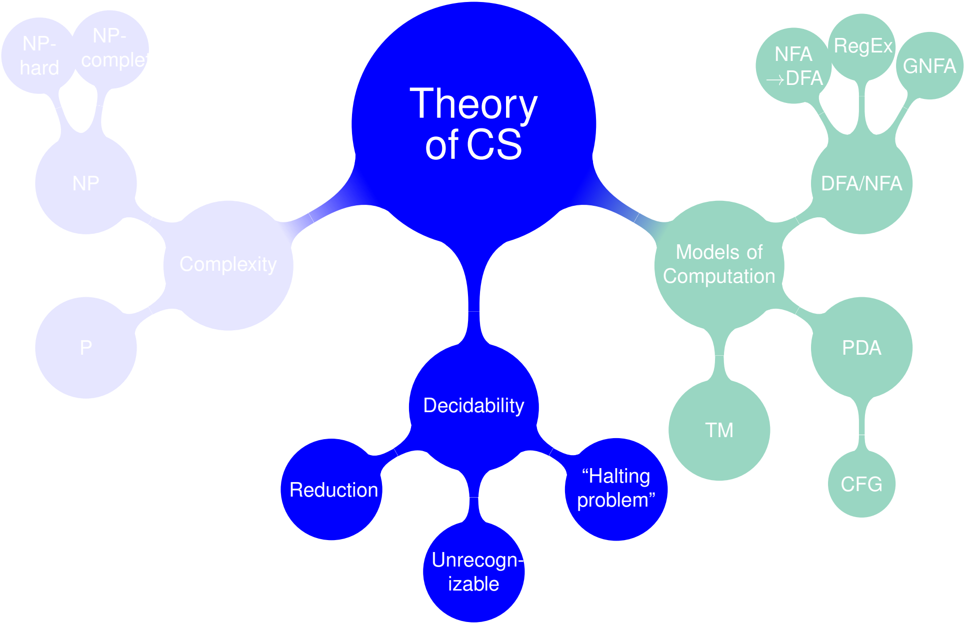
Lecture 7


Theory of CS
Complexity
NP
NP-hard
NP-complet
P
Decidability
Reduction
Unrecogn-izable
"Halting problem"
Models of Computation
DFA/NFA
NFA →DFA
RegEx
GNFA
PDA
CFG
TM



Decidability


Theory of CS
Models of Computation
DFA/NFA
NFA →DFA
RegEx
GNFA
PDA
CFG
TM
Decidability
Reduction
Unrecogn-izable
"Halting problem"
Complexity
NP
NP-hard
NP-complet
P


NP-hard
NP-complete
NP
Complexity
P
Theory of CS
Decidability
Reduction
Unrecogn-izable
"Halting problem"
Models of Computation
NFA →DFA
RegEx
GNFA
DFA/NFA
PDA
TM
CFG

Last time. . .

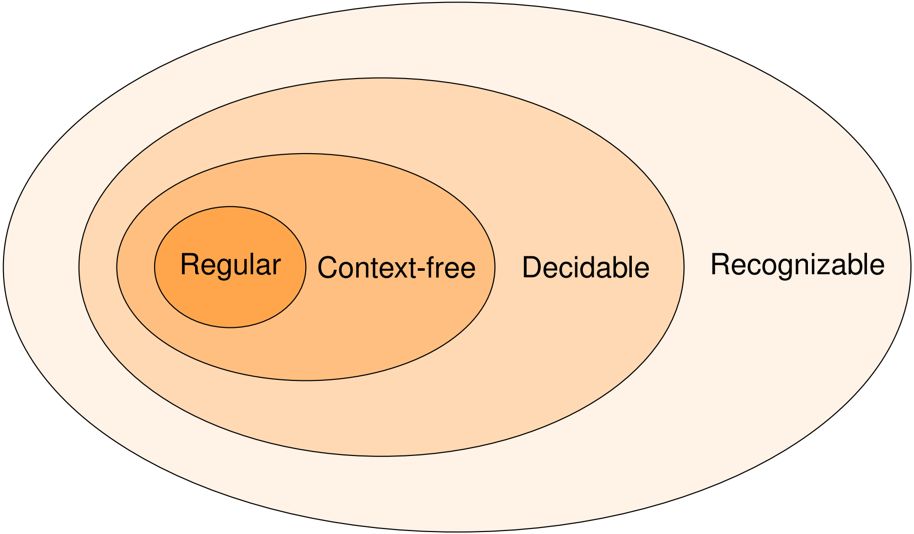
Turing Machines (TM) languages

- A language is **recognizable** if some TM **recognizes** it.
- A language is **decidable** if some TM **decides** it.
  (All branches of a NTM need to reject for it to reject a string.)

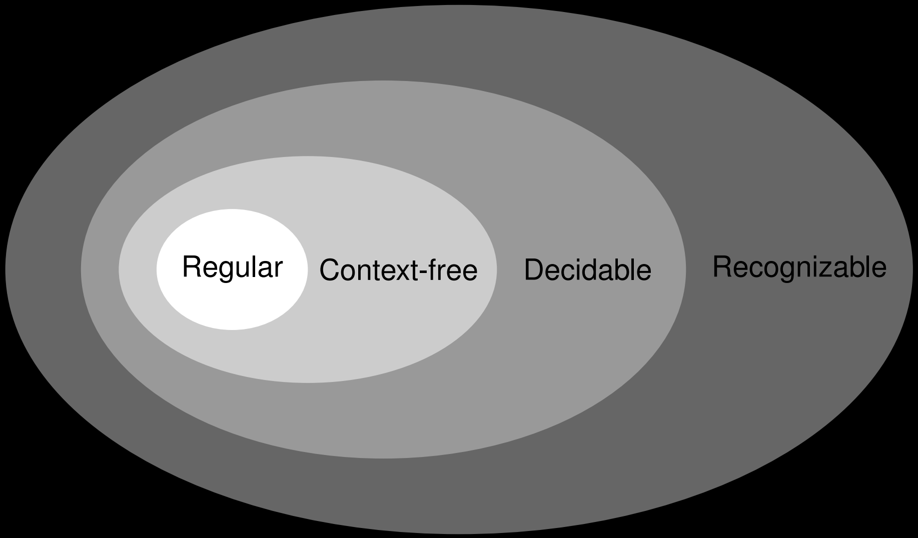
The Church-Turing Thesis – Algorithms

Intuitive concept of algorithms $=$ Turing machine algorithms

Venn diagram



The "computation universe" discovered so far. . .

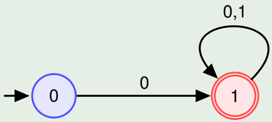


Notation: encoding of an object

We need to **encode** objects so that TMs can operate on them.
We use **angled brackets** to denote the encoding of a given *object*: $\langle \textit{object} \rangle$

Example

Let N be an NFA that accepts strings starting with 0.



We can encode N by listing the alphabet, the states, the start state, the accept states, and then the transition function.
Here is an example with $\Gamma = \{0, 1, ., ;, \#, \square\}$:

$$\langle N \rangle = \underbrace{0.1}_{\Sigma} \# \underbrace{0.1}_{Q} \# \underbrace{0}_{q_{\text{start}}} \# \underbrace{1}_{F} \# \underbrace{0.0.1; 1.0.1; 1.1.1}_{\delta}$$

Universal Turing Machines (UTMs)

There are TMs that can simulate any other TM!

Example

Universal TM U simulates TM T as follows:

- $\langle T \rangle$ is placed on the tape of U.
- U is designed to read $\langle T \rangle$ from the tape and do what T would have done in its tape. (This is a systematic process, so it has to be possible.)
- The part of the tape of U after $\langle T \rangle$ serves as T's tape.

In modern terminology: Both the **program** and the **data** are stored in the memory of the machine.
UTMs are what we now call **stored-program computers.**

Decidable problems – examples

- Problems about regular languages
  - **Acceptance**
    - $A_{\text{DFA}} = \{\langle D, w\rangle \mid D \text{ is a DFA that accepts the input string } w\}$
    - $A_{\text{NFA}} = \{\langle N, w\rangle \mid N \text{ is an NFA that accepts the input string } w\}$
    - $A_{\text{REX}} = \{\langle R, w\rangle \mid R \text{ is a RegEx that generates the string } w\}$
  - **Emptiness**
    - $E_{\text{DFA}} = \{\langle D\rangle \mid D \text{ is a DFA and } L(D) = \emptyset\}$
    - $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$
- Problems about context-free languages
  - **Acceptance**
    - $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates the string } w\}$
  - **Emptiness**
    - $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$

Undecidability

Computers seem so powerful – can they solve all (computational) problems?

Is $EQ_{\mathrm{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$ decidable?

No!

In the next few slides, we will see that:

- **Theorem**: Computers are limited in a **fundamental** way.
- One type of unsolvable problems:

  Given a computer program and a precise specification of what that program is supposed to do, verify that the program performs as specified.

$\rightarrow$ **Software verification** is, in general, not solvable by computers!

1/5: Liar paradox

$$S = \text{“I am lying.”}$$

If the liar **lied** then S is **false**. . .
but if S is **false** then the liar did **not lie**!
If the liar did **not lie** then S is **true**. . .
but if S is **true** then the liar **lied**!

$$S = \text{“}S\text{ is false.”}$$

If S is **true** then S is **false**.
But if S is **false** then S must be **true**.
But . . .
But . . .

2/5: Russel paradox

Does "the list of all lists that do not contain themselves" contain itself?

If it does then it does not belong to itself and should be removed.
But, if it does not list itself, then it should be added to itself.
But, ...
But, ...

3/5: Undecidability – the **Acceptance Problem**

Consider

$$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

Suppose that a decider D exists such that

$$D(\langle M\rangle) = \begin{cases} \textbf{reject} & \text{if } M \textbf{ accepts } \langle M\rangle, \quad \text{i.e. } \langle M, \langle M\rangle\rangle \in A_{\text{TM}} \\ \textbf{accept} & \text{if } M \textbf{ does not accept } \langle M\rangle \end{cases}$$

Now run it on itself:

$$D(\langle D\rangle) = \begin{cases} \textbf{reject} & \text{if } D \textbf{ accepts } \langle D\rangle \\ \textbf{accept} & \text{if } D \textbf{ does not accept } \langle D\rangle \end{cases}$$

Does it accept or reject?

It rejects if it accepts, and it accepts if it doesn't accept!!

There is a problem with the assumption that such a D exists.

The acceptance problem A_{TM} therefore cannot be a decidable problem.

4/5: Undecidability – the **Halting Problem**

The Halting Problem

The Halting Problem is

$$HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \textit{ halts} \text{ on input } w\}$$

$HALT_{\text{TM}}$ is also undecidable.
Proof uses "**reduction**."
We have: $HALT_{\text{TM}}$ is decidable $\implies$ A_{TM} is decidable.

5/5: Reductions

How do we show that $\mathit{HALT}_{\mathrm{TM}}$ is undecidable?

Idea: use $\mathit{HALT}_{\mathrm{TM}}$'s decider to decide A_{TM} $\rightarrow$ **reduce** A_{TM} to $\mathit{HALT}_{\mathrm{TM}}$.

Proof

- Suppose there exists a TM H that decides $\mathit{HALT}_{\mathrm{TM}}$.
- Construct TM D to decide A_{TM} as follows:

  $D =$ "On input $\langle M, w \rangle$:
  1. Run H on input $\langle M, w \rangle$.
  2. If H rejects, *reject*.
  3. If H accepts, simulate M on w until it halts.
  4. If M has accepted, *accept*; if M has rejected, *reject*."
- If H decides $\mathit{HALT}_{\mathrm{TM}}$ then D decides A_{TM}.
- Since A_{TM} is undecidable then $\mathit{HALT}_{\mathrm{TM}}$ must also be undecidable.

More undecidable problems – there are lots of them!

Using reducibility we can show that the following problems are all undecidable

1. $E_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$ — Reduce A_{TM} to it.
2. $REGULAR_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$ — Reduce A_{TM} to it.
3. $EQ_{\text{TM}} = \{\langle M, M' \rangle \mid M, M' \text{ are TMs and } L(M) = L(M')\}$ — Reduce E_{TM} to it.
4. Post Correspondence Problem (PCP). — Reduce A_{TM} to it – see lab.

Summary

(red: undecidable, blue: decidable)

1. Acceptance problems
   1. $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$
   2. $A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$
   3. $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
   4. $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
2. Language emptiness problems
   1. $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$
   2. $E_{\text{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$
   3. $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$
3. Language equality problems
   1. $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$
   2. $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
   3. $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$
4. Miscellenious
   1. $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \textbf{ halts} \text{ on input } w\}$
   2. $REGULAR_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$
   3. Post Correspondence Problem (PCP).

Unrecognizable languages

Theorem

L is **decidable** $\iff$ **both** L and $\overline{L}$ are **recognizable**

Corollary

$\overline{A}_{\text{TM}}$ is **not recognizable.**

Proof

- Take $L = A_{\text{TM}}$ $= \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
- We know L is recognizable.
- If $\overline{L} = \overline{A}_{\text{TM}}$ were also recognizable then A_{TM} would be decidable.
- But we know A_{TM} is not decidable!
- So $\overline{A}_{\text{TM}}$ cannot be recognizable.

Next week: Time Complexity

- Being decidable means that an algorithm exists to decide the problem.
- However, the algorithm may still be *practically* ineffective because of its **time** and/or **space** cost.