Logic.
Computational Complexity

Pedro Cabalar

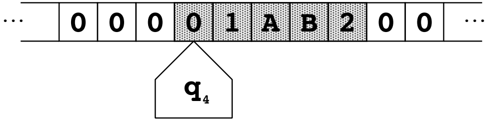
Dept. Computer Science
University of Corunna, SPAIN

March 15, 2023

1 Computational Complexity

Turing Machine

Alan Turing
(1912-1952)

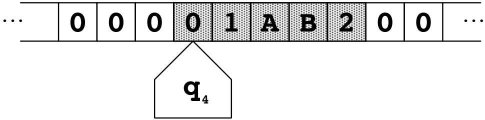


Turing machine (TM)

- TM = (theoretical) device that operates on an infinite tape with cells containing symbols in a finite alphabet (including blank ‘0’)

Turing Machine

Alan Turing
(1912-1952)

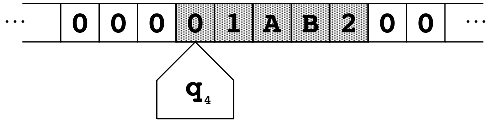


Turing machine (TM)

- TM = (theoretical) device that operates on an infinite tape with cells containing symbols in a finite alphabet (including blank ‘0’)
- The TM has a current state S_i among a finite set of states (including ‘*Halt*’), and a head pointing to “current” cell in the tape.

Turing Machine

Alan Turing
(1912-1952)

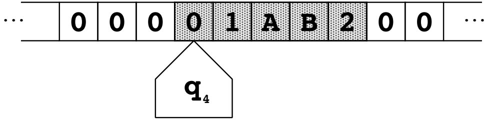


Turing machine (TM)

- TM = (theoretical) device that operates on an infinite tape with cells containing symbols in a finite alphabet (including blank '0')
- The TM has a current state S_i among a finite set of states (including '*Halt*'), and a head pointing to "current" cell in the tape.
- Its transition function describes jumps from state to next state.

Transition function

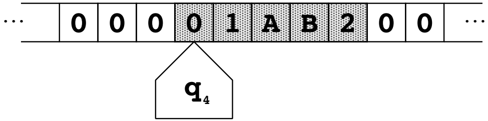
- Example: with scanned symbol 0 and state q_4, write 1, move *Left* and go to state q_2. That is:



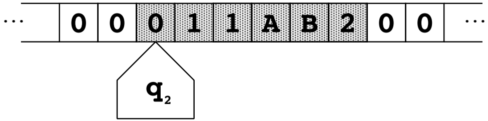
$$t(0, q_4) = (1, \mathit{Left}, q_2)$$

Transition function

- Example: with scanned symbol 0 and state q_4, write 1, move *Left* and go to state q_2. That is:



$$t(0, q_4) = (1, Left, q_2)$$



Decision problems

Definition (Decision problem)

A decision problem consists in providing a given tape input and asking the TM for a final output symbol answering *Yes* or *No*.

- Example: *SAT* = given (an encoding of) a propositional formula, answer *yes* if the formula has at least one model

Decision problems

Definition (Decision problem)

A decision problem consists in providing a given tape input and asking the TM for a final output symbol answering *Yes* or *No*.

- Example: *SAT* = given (an encoding of) a propositional formula, answer *yes* if the formula has at least one model
- Example: *HALTING* = given another TM M plus its input tape T, answer *yes* if M, T stops

Decision problems

Definition (Decision problem)

A decision problem consists in providing a given tape input and asking the TM for a final output symbol answering *Yes* or *No*.

- Example: *SAT* = given (an encoding of) a propositional formula, answer *yes* if the formula has at least one model
- Example: *HALTING* = given another TM M plus its input tape T, answer *yes* if M, T stops
- If X is a decision problem, then its complement $\overline{X}$ is the one where the Turing Machine answers the opposite.

Decision problems

Definition (Decision problem)

A decision problem consists in providing a given tape input and asking the TM for a final output symbol answering *Yes* or *No*.

- Example: SAT = given (an encoding of) a propositional formula, answer *yes* if the formula has at least one model
- Example: $HALTING$ = given another TM M plus its input tape T, answer *yes* if M, T stops
- If X is a decision problem, then its complement $\overline{X}$ is the one where the Turing Machine answers the opposite.
- Example: $\overline{SAT} = UNSAT$ answers *no* if the formula has a model.

Decision problems

- A decision problem is decidable if the TM stops (answering *Yes* or *No*) in a finite number of steps.

Decision problems

- A decision problem is decidable if the TM stops (answering *Yes* or *No*) in a finite number of steps.
- Examples: *SAT* is decidable. *HALTING* is undecidable.

Decision problems

- A decision problem is decidable if the TM stops (answering *Yes* or *No*) in a finite number of steps.
- Examples: *SAT* is decidable. *HALTING* is undecidable.
- A decision problem is in complexity class **P** iff the number of steps carried out by the TM is polynomial on the size n of the input.

Non-deterministic TM

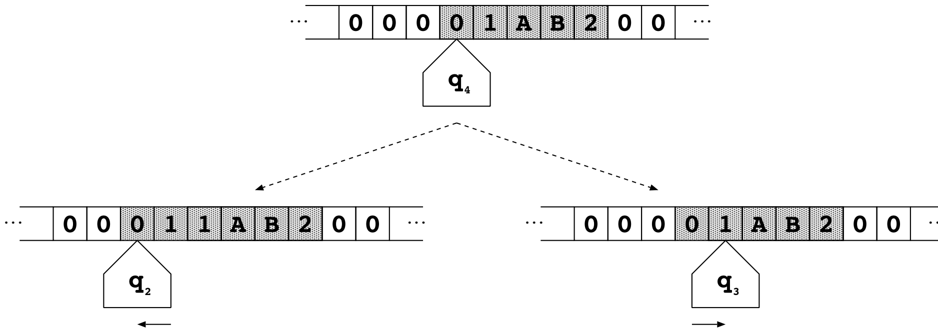
- Now, a non-deterministic Turing Machine (NDTM) is such that the transition function is replaced by a transition relation.

Non-deterministic TM

- Now, a non-deterministic Turing Machine (NDTM) is such that the transition function is replaced by a transition relation.
- We may have different possibilities for the next step.

Non-deterministic TM

- Now, a non-deterministic Turing Machine (NDTM) is such that the transition function is replaced by a transition relation.
- We may have different possibilities for the next step.
- Example: $t(0, q_4, 1, Left, q_2)$, $t(0, q_4, 0, Right, q_3)$



Non-deterministic TM

- Keypoint: an NDTM provides an affirmative answer to a decision problem when at least one of the executions for the same input answers *Yes*.

Non-deterministic TM

- Keypoint: an NDTM provides an affirmative answer to a decision problem when at least one of the executions for the same input answers *Yes*.
- A decision problem is in class **NP** iff the number of steps carried out by the NDTM is polynomial on the size n of the input.

Non-deterministic TM

- Keypoint: an NDTM provides an affirmative answer to a decision problem when at least one of the executions for the same input answers *Yes*.
- A decision problem is in class **NP** iff the number of steps carried out by the NDTM is polynomial on the size n of the input.
- For *SAT*, we can build an NDTM that performs two steps:
  1. For each atom, generate 1 or 0 nondeterministically. This provides an arbitrary interpretation in linear time.

Non-deterministic TM

- Keypoint: an NDTM provides an affirmative answer to a decision problem when at least one of the executions for the same input answers *Yes*.
- A decision problem is in class **NP** iff the number of steps carried out by the NDTM is polynomial on the size n of the input.
- For *SAT*, we can build an NDTM that performs two steps:
  1. For each atom, generate 1 or 0 nondeterministically. This provides an arbitrary interpretation in linear time.
  2. Test whether the current interpretation is a model or not. Complexity: **ALOGTIME** $\subseteq$ **P**

Non-deterministic TM

- Keypoint: an NDTM provides an affirmative answer to a decision problem when at least one of the executions for the same input answers *Yes*.
- A decision problem is in class **NP** iff the number of steps carried out by the NDTM is polynomial on the size n of the input.
- For *SAT*, we can build an NDTM that performs two steps:
  1. For each atom, generate 1 or 0 nondeterministically. This provides an arbitrary interpretation in linear time.
  2. Test whether the current interpretation is a model or not. Complexity: **ALOGTIME** $\subseteq$ **P**

  The sequence of these two steps takes polynomial time.

P vs NP

- Any TM is a particular type of NDTM, so **P** $\subseteq$ **NP** trivially,

P vs NP

- Any TM is a particular type of NDTM, so **P** $\subseteq$ **NP** trivially, but ...

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

P vs **NP**

- Any TM is a particular type of NDTM, so **P** $\subseteq$ **NP** trivially, but . . .

$$\boxed{\mathbf{P} \stackrel{?}{=} \mathbf{NP}}$$

- Unsolved problem: most accepted conjecture **P** $\subset$ **NP**, but remains unproved.

P vs NP

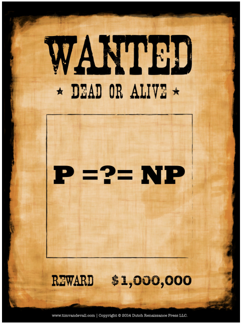
- Any TM is a particular type of NDTM, so **P** $\subseteq$ **NP** trivially, but ...

$$\boxed{\mathbf{P} \stackrel{?}{=} \mathbf{NP}}$$

- Unsolved problem: most accepted conjecture **P** $\subset$ **NP**, but remains unproved.

It is one of the 7 Millenium Prize Problems

http://www.claymath.org/millennium-problems



The Clay Mathematics Institute designated $1 million prize for its solution!

Completeness

- A problem X is **C**-complete, for some complexity class **C**, iff any problem Y in **C** is reducible to X in polynomial-time.

Completeness

- A problem X is **C**-complete, for some complexity class **C**, iff any problem Y in **C** is reducible to X in polynomial-time.
- A complete problem is a representative of the class. Example: if an **NP**-complete problem happened to be in **P** then **P** $=$ **NP**.

Completeness

- A problem X is **C**-complete, for some complexity class **C**, iff any problem Y in **C** is reducible to X in polynomial-time.
- A complete problem is a representative of the class. Example: if an **NP**-complete problem happened to be in **P** then **P** $=$ **NP**.
- *SAT* was the first problem to be identified as **NP**-complete (Cook's theorem, 1971).

Completeness

- A problem X is **C**-complete, for some complexity class **C**, iff any problem Y in **C** is reducible to X in polynomial-time.
- A complete problem is a representative of the class. Example: if an **NP**-complete problem happened to be in **P** then **P** $=$ **NP**.
- *SAT* was the first problem to be identified as **NP**-complete (Cook's theorem, 1971).
- *SAT* is commonly used nowadays for showing that a problem X is at least as complex as **NP**. To this aim, just encode *SAT* into X.

Completeness

- A problem X is $\mathbf{C}$-complete, for some complexity class $\mathbf{C}$, iff any problem Y in $\mathbf{C}$ is reducible to X in polynomial-time.
- A complete problem is a representative of the class. Example: if an **NP**-complete problem happened to be in **P** then $\mathbf{P} = \mathbf{NP}$.
- *SAT* was the first problem to be identified as **NP**-complete (Cook's theorem, 1971).
- *SAT* is commonly used nowadays for showing that a problem X is at least as complex as **NP**. To this aim, just encode *SAT* into X.
- The Complexity Zoo
  https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Complementary class

- If C is a complexity class, then $co - C$ is the complementary class. That is $X \in co - C$ iff $\overline{X} \in C$.

Complementary class

- If C is a complexity class, then $co - C$ is the complementary class. That is $X \in co - C$ iff $\overline{X} \in C$.
- $\mathbf{co-NP}$ = problems in which a NDTM answers *no* in a polynomial time

Complementary class

- If C is a complexity class, then $co - C$ is the complementary class. That is $X \in co - C$ iff $\overline{X} \in C$.
- **co−NP** = problems in which a NDTM answers *no* in a polynomial time
- In general, **co−NP** $\neq$ **NP** (the intersection is non-empty)

Complementary class

- If C is a complexity class, then $co - C$ is the complementary class. That is $X \in co - C$ iff $\overline{X} \in C$.
- **co−NP** = problems in which a NDTM answers *no* in a polynomial time
- In general, **co−NP** $\neq$ **NP** (the intersection is non-empty)
- *UNSAT* is in **co−NP**.

Complementary class

- If C is a complexity class, then $co - C$ is the complementary class. That is $X \in co - C$ iff $\overline{X} \in C$.
- **co−NP** = problems in which a NDTM answers *no* in a polynomial time
- In general, **co−NP** $\neq$ **NP** (the intersection is non-empty)
- *UNSAT* is in **co−NP**. This implies that *VAL* (deciding whether α is valid) is also **co−NP**.

Exercise: Turing machine in Prolog

- We use `tape(Ls,S,Rs)` to represent the current symbol `S`, the left fragment of the tape `Ls` (reversed) and the right one `Rs`.

```
compute(Q, T, T) :- final(Q), !.

compute(Q0, tape(Ls0,S,Rs0), T):-
  showmachine(Q0,Ls0,S,Rs0),
  t(Q0,S,  Q1,S1,Action),
  move(Action,tape(Ls0,S1,Rs0),T1),
  compute(Q1,T1,T).

move(l,tape([],    S,Rs), tape([],0,[S|Rs])).
move(l,tape([L|Ls],S,Rs), tape(Ls,L,[S|Rs])).

move(r,tape(Ls,S,[]),     tape([S|Ls],0,[])).
move(r,tape(Ls,S,[R|Rs]), tape([S|Ls],R,Rs)).
```