Week 12 - Friday

COMP 4500

Last time

- What did we talk about last time?
- Showing lots of problems are NP-Complete
- Started Co-NP

Questions?

Assignment 6

Exam 3 Post Mortem

Logical warmup

- You are paddling a canoe around a perfectly circular pond
- Enjoying yourself immensely, you fail to notice that a goblin has crept up to the shore
- You remember four things from your old lessons on goblin lore
  - Goblins can't swim
  - Goblins are always hungry for human flesh
  - Goblins can run four times as fast as people can paddle canoes
  - People can run faster than goblins
- The goblin always assumes that you are making for the closest point on the shore and is always trying to cut you off
- If you can get to the shore, you can escape, provided that the goblin isn't waiting for you (you need a little margin)
- What's your escape strategy?

Co-NP

Asymmetric certification

- Efficient certification is **asymmetric**
- A problem is in NP if and only if there is a certificate that can be checked in polynomial time for a "yes" answer
- If the answer is "no," there's no requirement for a certificate
- How would you give a short certificate that there's **no** satisfying assignment for 3-SAT?

Co-NP

- But there is an alternative definition
- Co-NP is the set of all problems for which there is a polynomial-size certificate if the answer is "no"
- Both Co-NP and NP problems can certainly be solved with exponential work
- Is Co-NP = NP?
- No one knows!

What if NP ≠ Co-NP?

- Then we would know that **P ≠ NP**
- **Proof:**
  - Consider the contrapositive: (**P** = **NP**) → (**NP** = **Co-NP**).
  - **P** is closed under complementation. That means that the negated problem (all the strings that would give a "no") is also in **P**.
  - $X \in \mathbf{NP} \rightarrow X \in \mathbf{P} \rightarrow \bar{X} \in \mathbf{P} \rightarrow \bar{X} \in \mathbf{NP} \rightarrow X \in \mathbf{Co\text{—}NP}$
  - $X \in \mathbf{co\text{—}NP} \rightarrow \bar{X} \in \mathbf{NP} \rightarrow \bar{X} \in \mathbf{P} \rightarrow X \in \mathbf{P} \rightarrow X \in \mathbf{NP}$
  - Thus, **NP** = **co-NP**.
  - But, because the contrapositive is logically equivalent, that means that (**NP ≠ Co-NP**) → (**P ≠ NP**).

■

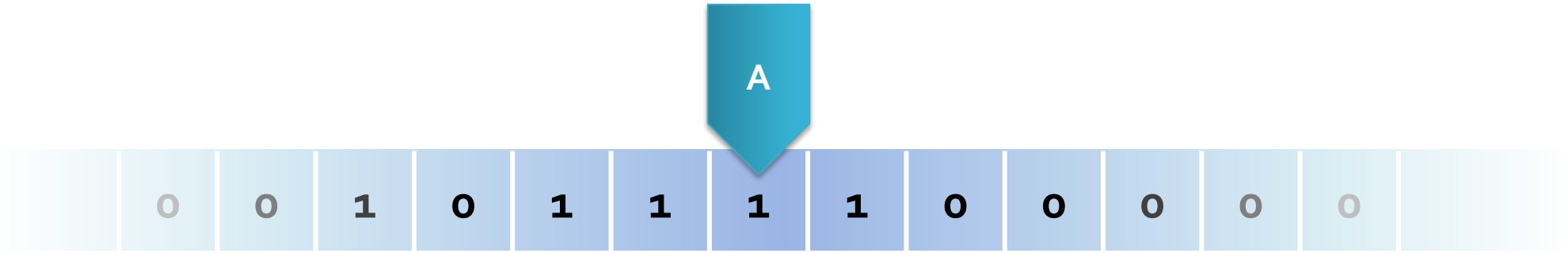
Is P = (NP ∩ Co-NP)?

- It's true that **P** ⊆ (**NP** ∩ **Co-NP**)
- But opinions are mixed as to whether there might be problems in **NP** ∩ **Co-NP** that cannot be solved in polynomial time
- Maybe you can find the answer!

Computability

Turing machine

- A Turing machine is a mathematical model for computation
- It consists of a head, an infinitely long tape, a set of possible states, and an alphabet of characters that can be written on the tape
- A list of rules saying what it should write and should it move left or right given the current symbol and state



More formally

- A Turing machine is an idealized machine with seven objects:
  - $\boldsymbol{Q}$ a finite set of states
  - Σ the finite input alphabet
  - Γ the finite tape alphabet
  - δ is a finite subset of $\boldsymbol{Q} \times (\Sigma \cup \varepsilon) \times \Gamma \times \boldsymbol{Q} \times \Gamma^*$ which gives a set of transition rules
  - $\boldsymbol{q}_0$ is the start state
  - $\square \in \Gamma$ is a special symbol called the blank
  - $\boldsymbol{F} \subseteq \boldsymbol{Q}$ is the set of accepting states
- Again, this is really just adding an infinite tape to a finite state automaton

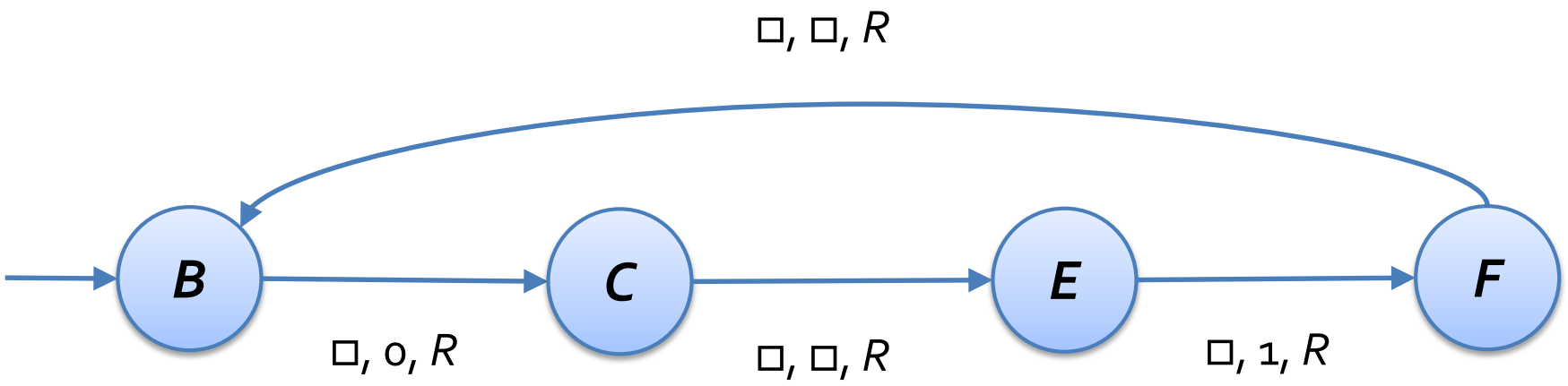
Turing machine example

- You can specify a Turing machine with a table giving its behavior for a specific configuration
- Turing's first example machine printed an infinite sequence of alternating 1s and 0s, separated by spaces:

Configuration		Behavior	
State	**Symbol**	**Operation**	**Result State**
B	Blank	Write 0, Move Right	C
C	Blank	Write Blank, Move Right	E
E	Blank	Write 1, Move Right	F
F	Blank	Write Blank, Move Right	B

A Turing machine as a transition diagram

- The transition table from the previous slide can be drawn as a transition diagram too:



Church-Turing thesis

- If an algorithm exists, a Turing machine can perform that algorithm
- In essence, a Turing machine is the most powerful model we have of computation
- Power, in this sense, means the ***ability*** to compute some function, **not** the ***speed*** associated with its computation
- Do you own a Turing machine?

Quiz

Upcoming

Next time…

- Finish theory of computing
- Load balancing
- Center selection
- Keep reading sections 11.1 and 11.2

Reminders

- Assignment 6 due tonight by midnight!