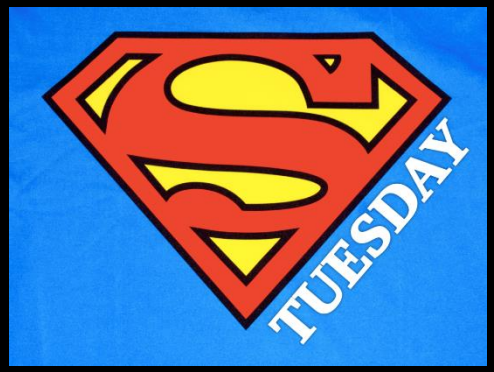


https://www.innovairre.com/super-tuesday/

6.045

Lecture 8: Communication Complexity, Start up Turing Machines

6.045

Announcements:

- Pest 3 is due tomorrow
- Midterm: March 19
- Reminder: these slides on the web!

L has a streaming alg using $\leq s(n)$ bits of space

means:

Give an algorithm A and prove that on all inputs x, A determines $x \in$ L correctly and uses $\leq s(|x|)$ bits of memory

Give an upper bound!

Every streaming alg for L needs $\geq s(n)$ bits of space

means:

For any n, give a streaming distinguisher S for L (a set of strings such that all pairs can be distinguished in L) where |S| $\geq 2^{s(n)}$

Give a lower bound!

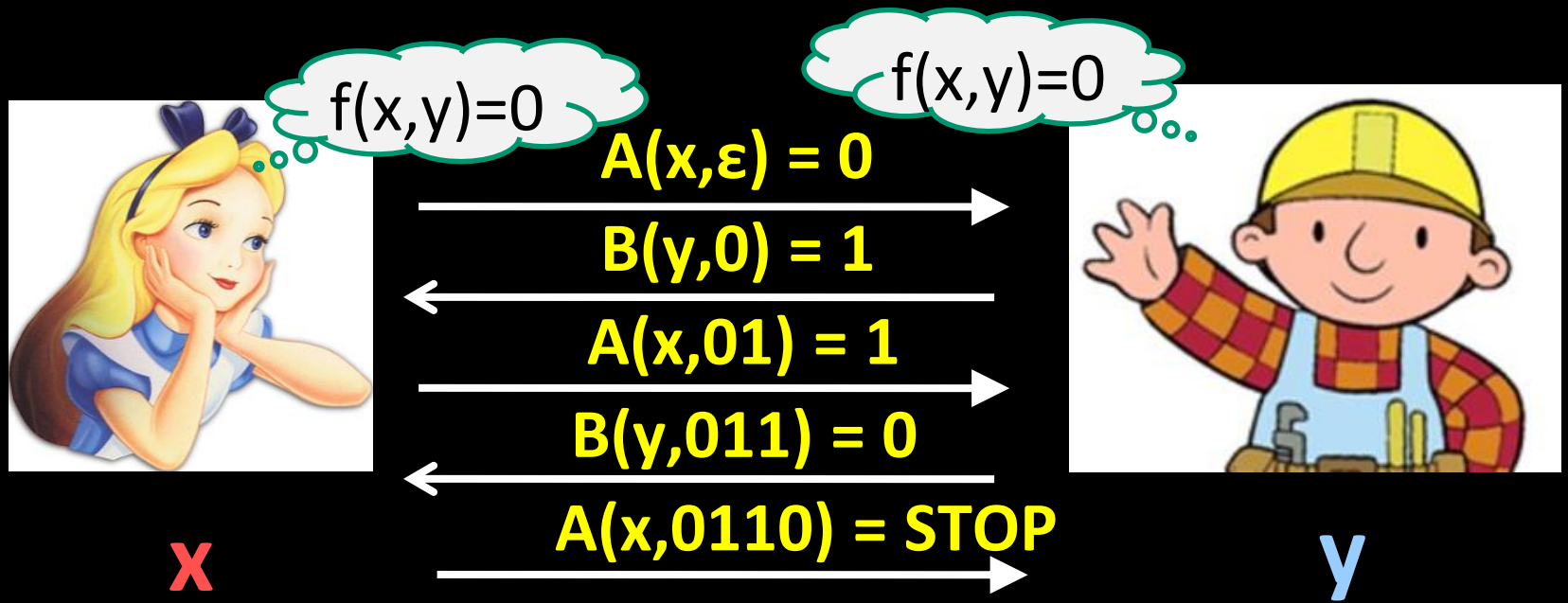
Communication Complexity

A theoretical model of distributed computing

- **Function $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$**
  - Two inputs, $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$
  - **We assume $|x|=|y|=n$. Think of n as HUGE**
- **Two computers: Alice** and **Bob**
  - **Alice** *only* knows x, **Bob** *only* knows y
- **Goal: Compute $f(x, y)$ by communicating as few bits as possible between Alice and Bob**

We do not count computation cost. We *only* care about the number of bits communicated.

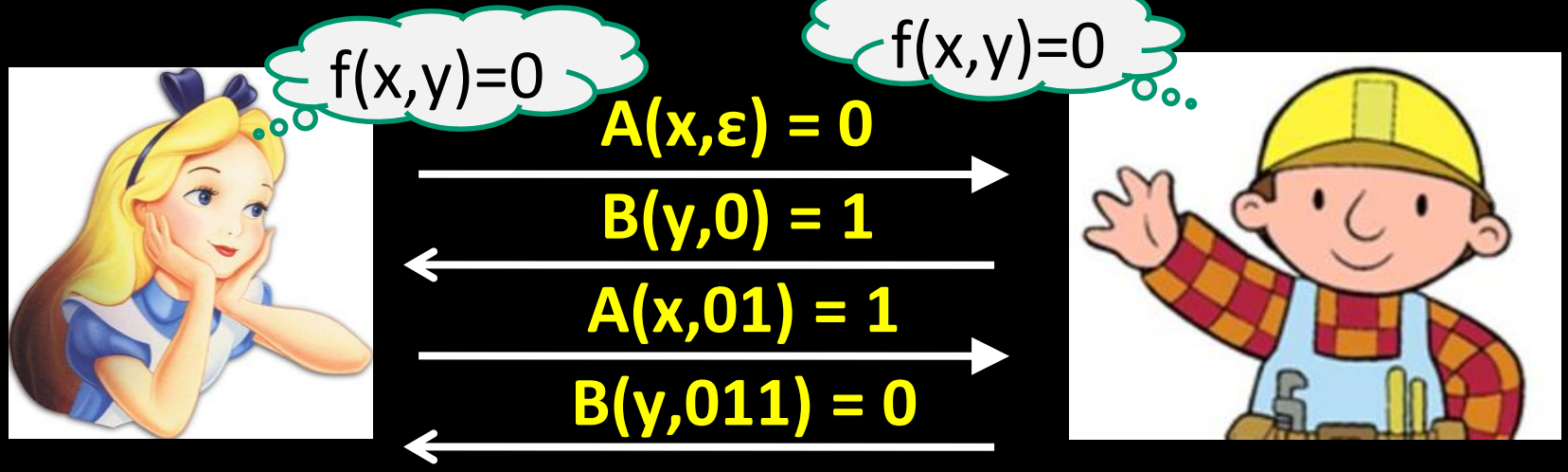
Alice and Bob Have a Conversation



In every step: A **bit** or **STOP** is sent, which is a function of the party's input and all the bits communicated so far.

Communication cost = number of bits communicated
= 4 (in the example)

We assume Alice and Bob alternate in communicating, and the last BIT sent is the value of $f(x,y)$



Def. A *protocol* computing f is a pair of functions $A, B : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0, 1, \text{STOP}\}$ with the semantics:

On input (x, y), let $r := 0$, $b_0 := \varepsilon$.

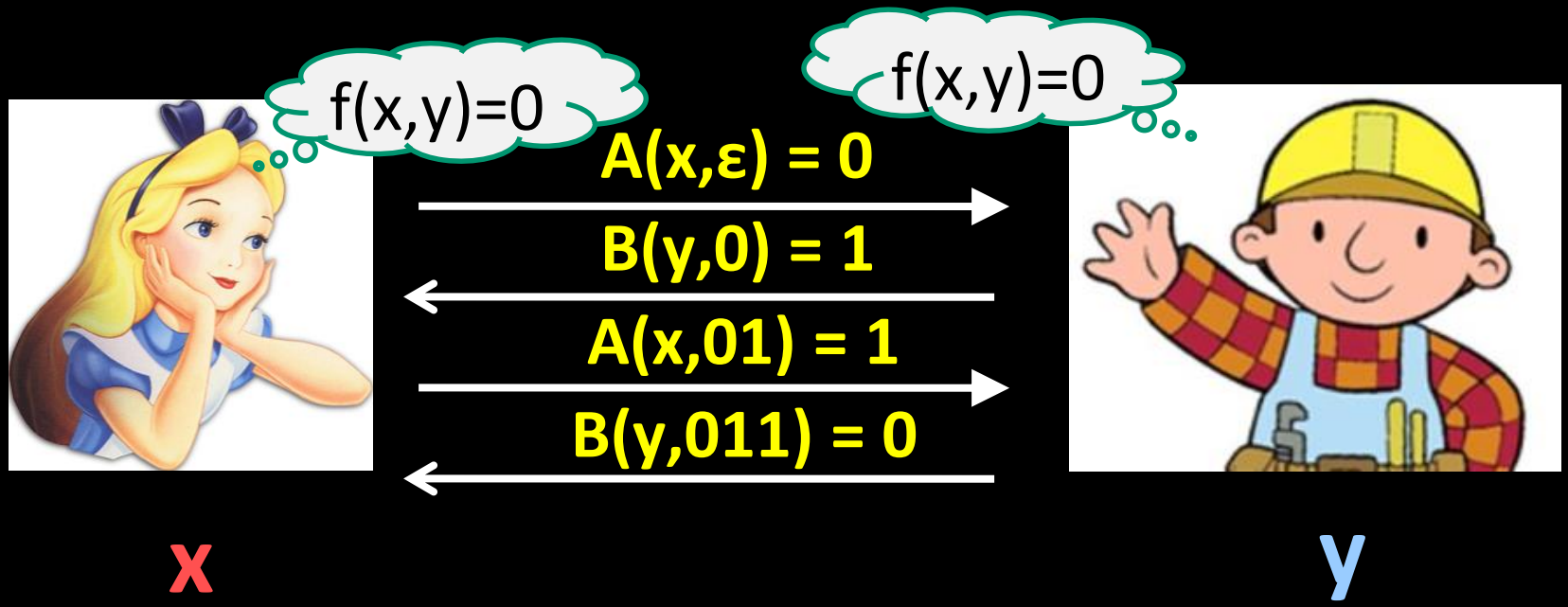
While ($b_r \neq$ STOP),

$r++$

If r is odd, Alice sends $b_r = A(x, b_1 \cdots b_{r-1})$

else Bob sends $b_r = B(y, b_1 \cdots b_{r-1})$

Output $b_{r-1} = f(x, y)$. Number of *rounds* $= r - 1$

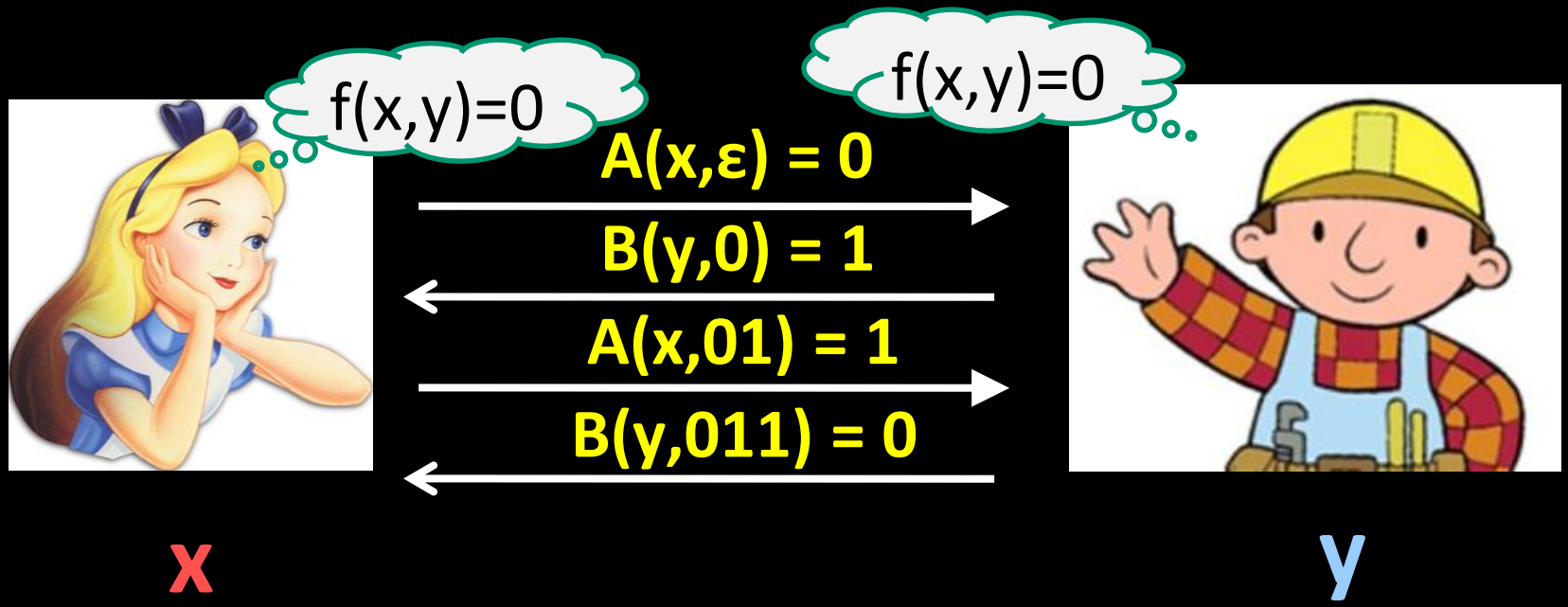


Def. The *cost of a protocol* (A,B) *on* n*-bit strings is*

$$\max_{x,y \in \{0,1\}^n} [\text{number of rounds taken by (A,B) on } (x, y)]$$

The *communication complexity* of *f* on n-bit strings, *cc(f)*, is *min* cost over *all protocols* computing *f* on n-bit strings

= the minimum number of rounds used by any protocol computing $f(x, y)$, over all n-bit x, y



Example. Let $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ be arbitrary

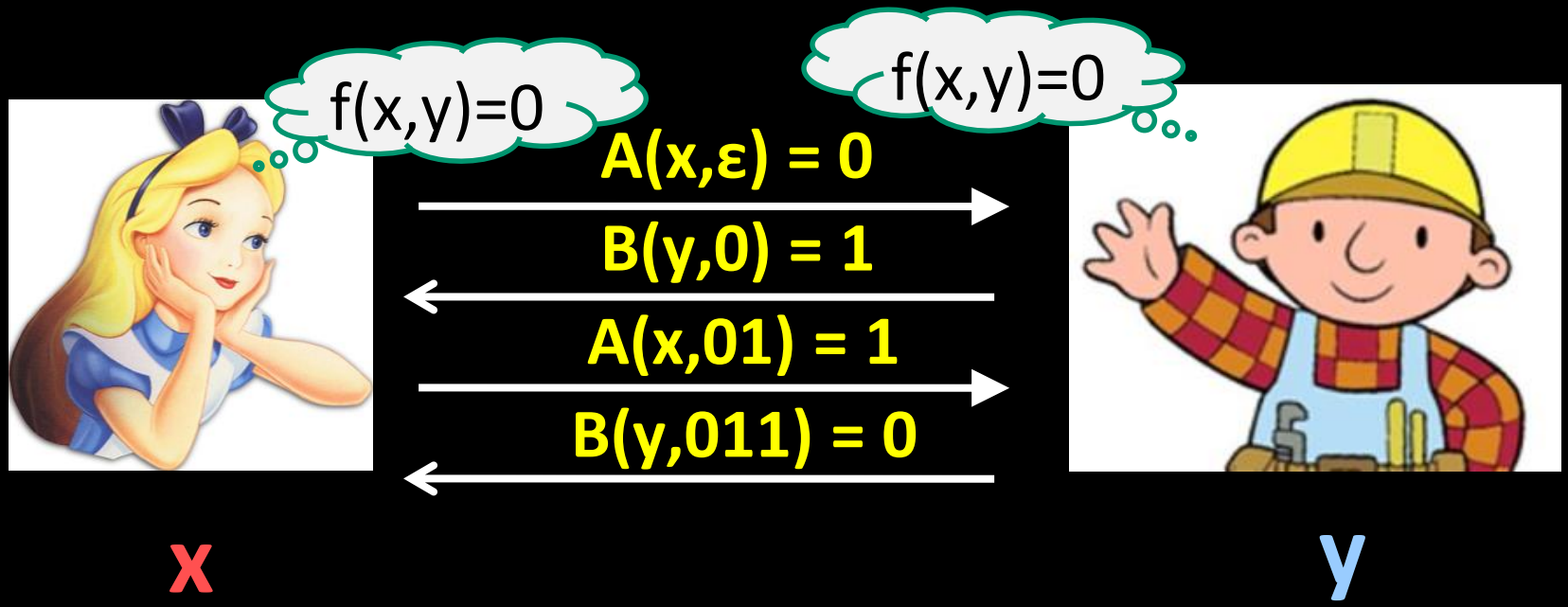
There is always a "trivial" protocol for f:

Alice sends the bits of her x in odd-numbered rounds

Bob sends whatever bit in even rounds

After $2n - 1$ rounds, Bob knows x and can send $f(x, y)$

Proposition: For every f, $cc(f) \leq 2n$



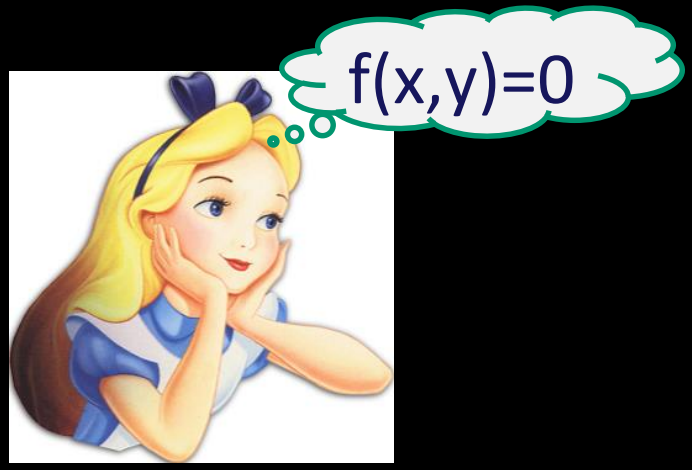
Example. PARITY$(x, y) = \sum_i x_i + \sum_i y_i$ mod 2.

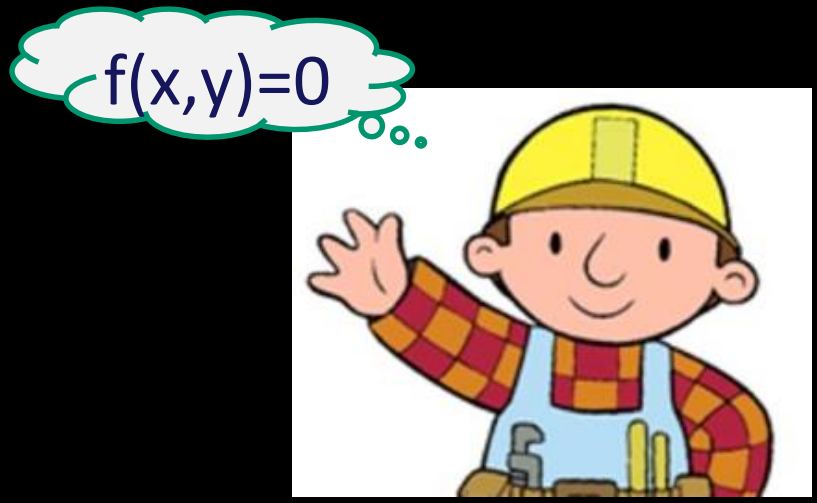
What's a good protocol for computing PARITY?

Alice sends $b_1 = (\sum_i x_i \bmod 2)$

Bob sends $b_2 = (b_1 + \sum_i y_i \bmod 2)$. Alice stops.

Proposition: cc(PARITY) = 2





x **y**

Example. MAJORITY(x, y) = most frequent bit in xy

Models voting in two "remote" locations; they want to determine a winner

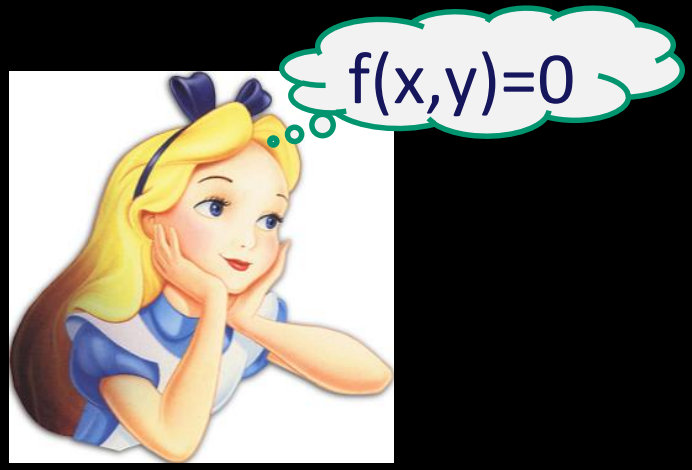
What's a good protocol for computing MAJORITY?

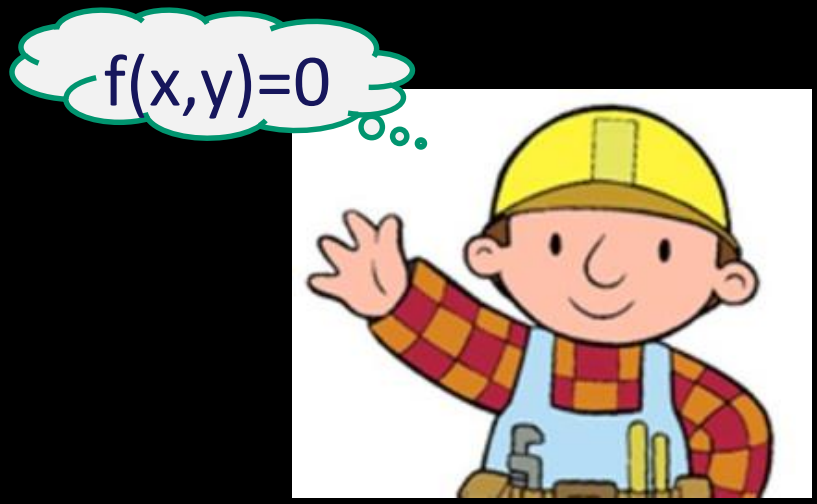
Alice sends N_x = number of 1s in x

Bob computes N_y = number of 1s in y,

sends 1 iff $N_x + N_y$ is greater than (|x|+|y|)/2 = n

Proposition: cc(MAJORITY) $\leq$ O(log n)





x **y**

Example. EQUALS(x, y) = 1 $\Leftrightarrow$ $x = y$

Useful for checking consistency of two far-apart databases!

What's a good protocol for computing EQUALS?

????

***Communication complexity* of EQUALS is at most $n + 2$**

Connection to Streaming Algs and DFAs

x y

Let $L \subseteq \{0,1\}^*$

Def. f_L: $\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$

for x, y with $|x|=|y|$ as:

$$f_L(x,y) = 1 \Leftrightarrow xy \in L$$

Examples:

L = { x | x has an odd number of 1s}

$\Rightarrow f_L(x,y)$ = PARITY(x,y) = $\sum_i x_i + \sum_i y_i$ mod 2

L = { x | x has at least as many 1s as 0s}

$\Rightarrow f_L(x,y)$ = MAJORITY(x,y)

L = { xx | x $\in \{0,1\}^*$}

$\Rightarrow f_L(x,y)$ = EQUALS(x,y)

Connection to Streaming Algs and DFAs

x y

Let $L \subseteq \{0,1\}^*$

Def. $f_L: \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ for x, y with $|x|=|y|$ as:

$$f_L(x,y) = 1 \Leftrightarrow xy \in L$$

Theorem: If L has a streaming alg using $\leq s(m)$ space on inputs of length $\leq 2m$, then $cc(f_L) \leq O(s(n))$.

Proof Idea: Alice runs streaming algorithm A on x, reaches a memory state m. She sends m to Bob in $O(s(n))$ rounds. Then Bob starts up A from state m, runs A on y. Gets an output bit, sends bit to Alice.

Connection to Streaming Algs and DFAs

Let $L \subseteq \{0,1\}^*$ Def. $f_L(x,y) = 1 \Leftrightarrow xy \in L$

Theorem: If L has a streaming alg using $\leq s(m)$ space on inputs of length $\leq 2m$, then $cc(f_L) \leq O(s(n))$.

Corollary: For every regular L, $cc(f_L) \leq O(1)$.

Example: cc(PARITY) = 2

Corollary: cc(MAJORITY) $\leq$ O(log n),
because there's a streaming algorithm for
{x : x has more 1's than 0's} with O(log n) space

What about the Comm. Complexity of EQUALS?

Communication Complexity of EQUALS

Theorem: cc(EQUALS) = $\Theta(n)$.
In particular, ***every*** communication protocol for EQUALS must send $\geq n$ bits between Alice and Bob.

No communication protocol can do much better than "send your whole input"!

Corollary: L = {xx | x in {0,1}*} is not regular.

Corollary: *Every streaming algorithm for* L *needs* $c\,n$ *bits of memory, for some constant* $c > 0$*!*

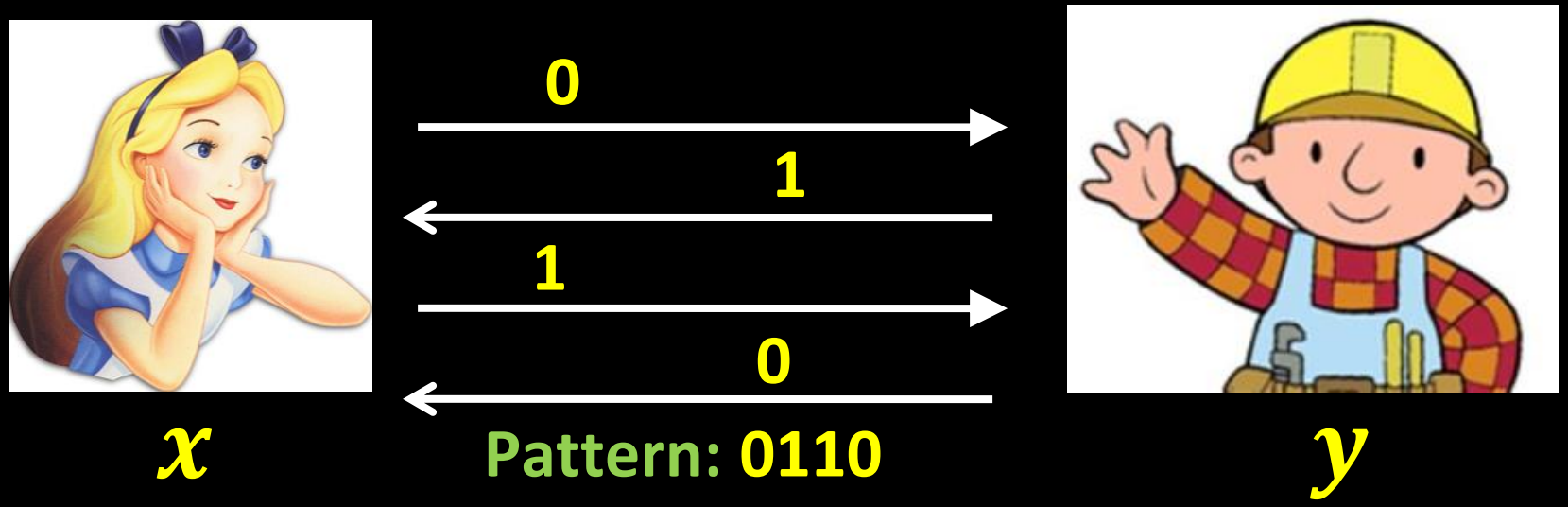
$\Omega(n)$

Communication Complexity of EQUALS

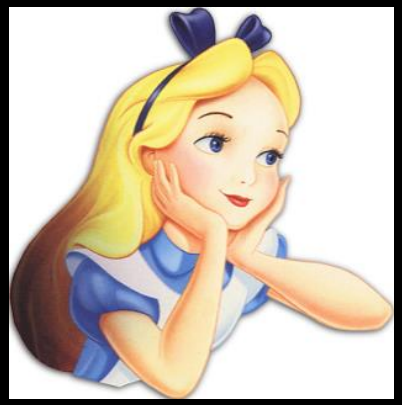
Theorem: cc(EQUALS) = $\Theta(n)$. In particular, *every* protocol for EQUALS needs $\geq n$ bits of communication!

Idea: Consider all possible ways A & B can communicate.

Definition: The *communication pattern* of a protocol on inputs (x, y) is the sequence of bits Alice & Bob send.



Key Lemma: If (x, y) and (x', y') have the same pattern P in a protocol, then (x, y') and (x', y) also have pattern P



x

A(x,ε) = 0

B(y,0) = 1

A(x,01) = 1

B(y,011) = 0

y

x'

A(x',ε) = 0

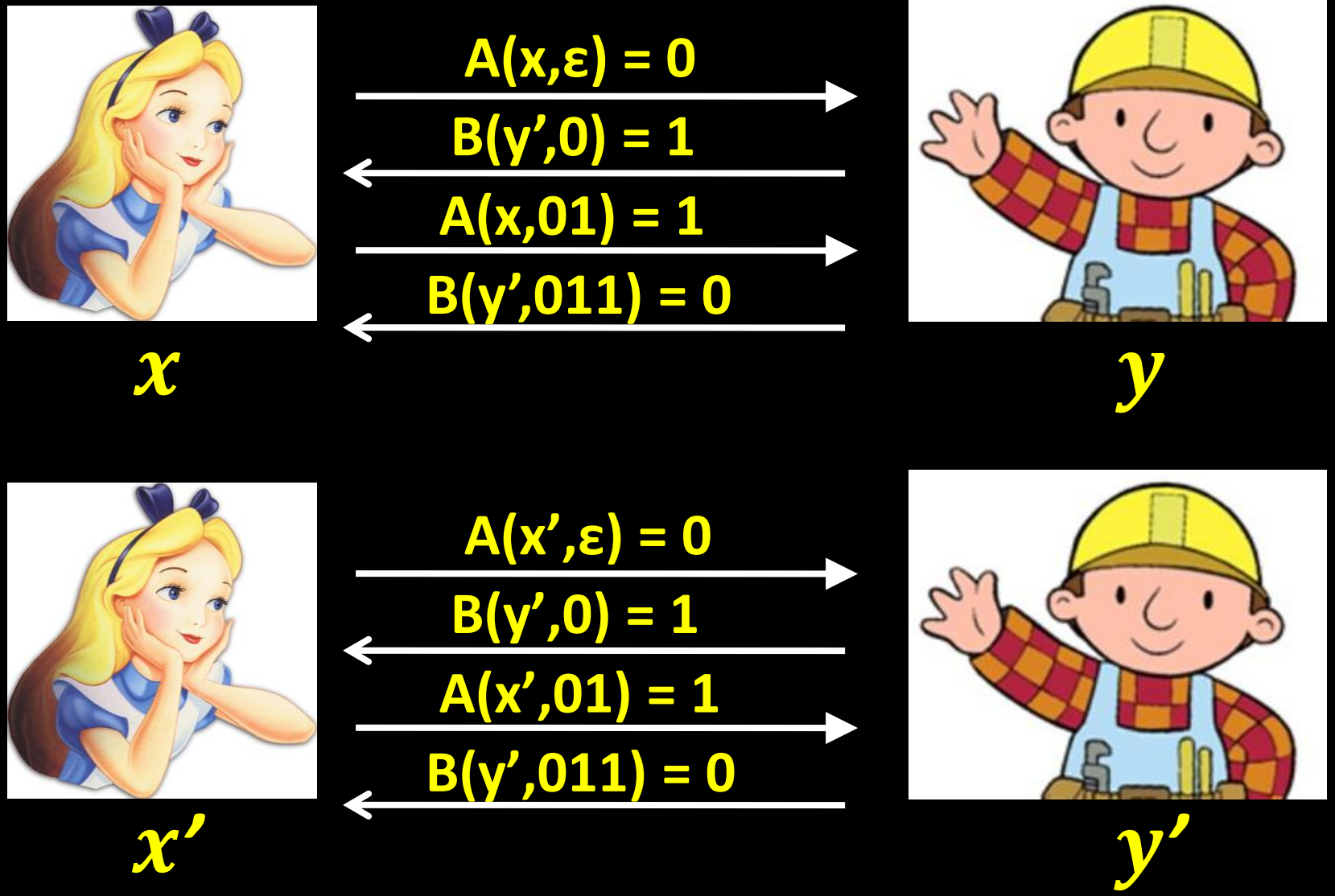
B(y',0) = 1

A(x',01) = 1

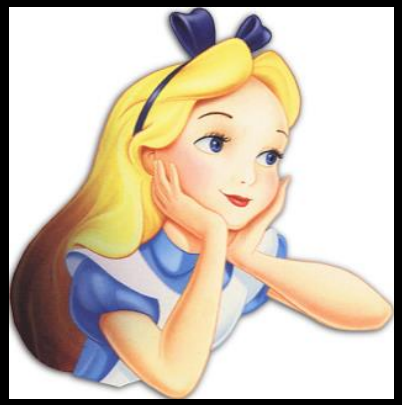
B(y',011) = 0

y'

Key Lemma: If (x, y) and (x', y') have the same pattern P in a protocol, then (x, y') and (x', y) also have pattern P



Key Lemma: If (x, y) and (x', y') have the same pattern P in a protocol, then (x, y') and (x', y) also have pattern P



x

A(x,ε) = 0

B(y,0) = 1

A(x,01) = 1

B(y,011) = 0

y

x'

A(x,ε) = 0

B(y',0) = 1

A(x,01) = 1

B(y',011) = 0

y'

Communication Complexity of EQUALS

Theorem: The comm. complexity of EQUALS is $\Theta(n)$.
In particular, *every* protocol for EQUALS needs $\geq n$ bits of communication.

Proof: By contradiction. Suppose cc(EQUALS) $\leq n - 1$.
Then there are $\leq 2^n - 1$ possible communication *patterns* of that protocol, over all pairs of inputs (x, y) with n bits each.

Claim: There are $x \neq y$ such that on (x, x) and on (y, y), the protocol uses the *same* pattern P.

By the Key Lemma, (x, y) and (y, x) also use pattern P

So Alice & Bob *output the same bit* on (x, y) and (x, x).
But EQUALS(x, y) = 0 and EQUALS(x, x) = 1. *Contradiction!*

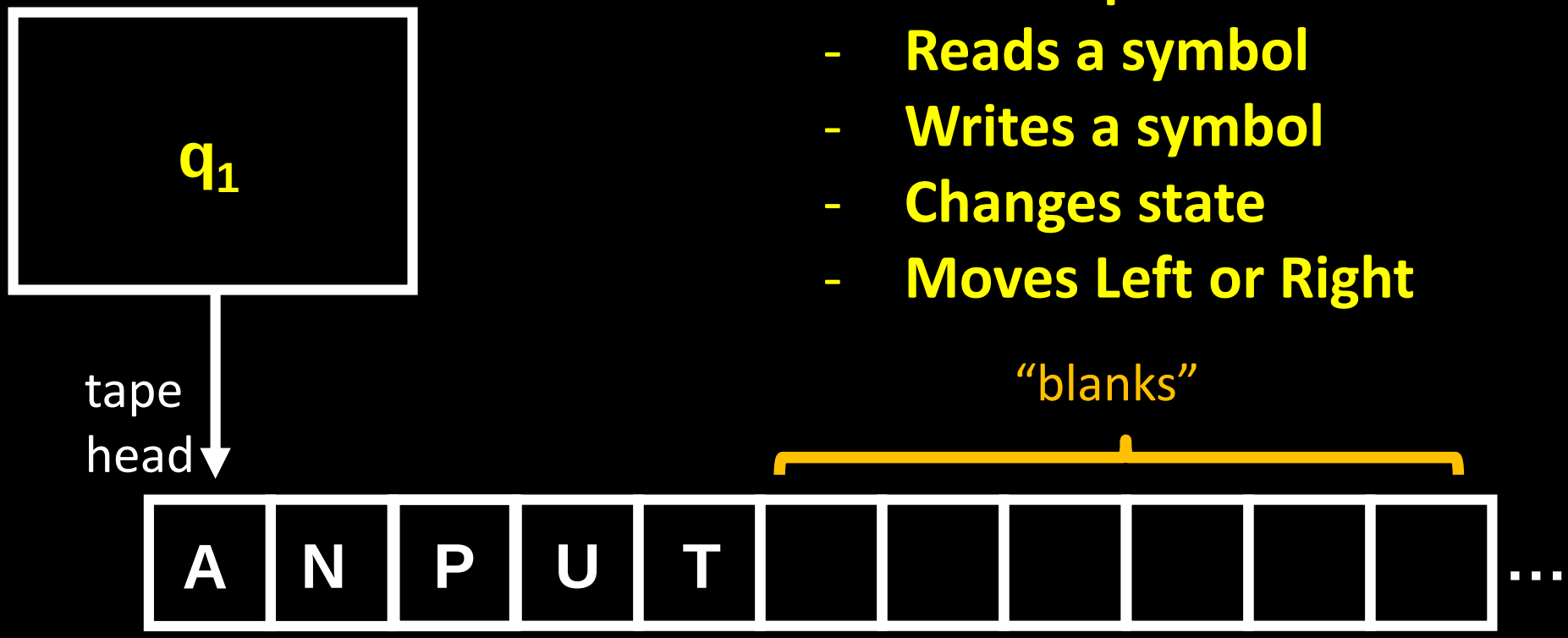
Randomized Protocols Help!

EQUALS needs $\geq n$ bits of communication, but…

Theorem: There is a ***randomized*** protocol for computing **EQUALS**(x, y) using only **O(log n)** bits of communication,
which is correct with probability **99.9%!**

Turing Machines

Turing Machine (1936)



In each step:

- **Reads a symbol**
- **Writes a symbol**
- **Changes state**
- **Moves Left or Right**

INFINITE REWRITABLE TAPE

Turing Machine (1936)

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers,