Introduction to Logic in Computer Science: Autumn 2007

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Communication Complexity

This will be a brief introduction to communication complexity.

Rather than analysing the *computational* difficulty of computing some function, communication complexity provides a formal framework for analysing the *amount of information* that needs to be exchanged when a function is computed in a distributed manner.

We will introduce the basics of the so-called *two-party model* put forward by Yao (1979). This lecture is based on the first chapter of the book by Kushilevitz and Nisan (1997).

A.C.-C. Yao. *Some Complexity Questions Related to Distributive Computing.* Proc. STOC-1979, ACM Press, 1979.

E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

The Two-Party Model

Let X, Y, Z be finite sets and let $f : X \times Y \rightarrow Z$ be some function.

Alice and Bob want to compute $f(x, y)$ for some $x \in X, y \in Y$.
Alice only knows x; Bob only knows y.

How many bits of information do they need to exchange before both of them know the answer?

Protocols

A *protocol* $\mathcal{P}$ over domain $X \times Y$ with range Z is a *binary tree*, where

- each *internal node* v is labelled with either a function $a_v : X \to \{0,1\}$ or a function $b_v : Y \to \{0,1\}$; and
- each *leaf node* is labelled with an element of Z.

Executing $\mathcal{P}$, when Alice holds x and Bob holds y, works as follows:

- Start with the root node.
- Whenever we reach an internal node v labelled with a function a_v, go to the left child if $a_v(x) = 0$ and to the right child otherwise. That is, depending on the history of communication so far (branch leading to v) and x, Alice decides what bit to send next.
- Similarly for internal nodes labelled with some b_v (for Bob).
- The value of $\mathcal{P}$ for (x, y) is the label z of the leaf node we end up in.

$\mathcal{P}$ computes f iff the value of $\mathcal{P}$ for input (x, y) is always $f(x, y)$.

Example

Let $X = \{x_1, x_2, x_3, x_4\}$ and $Y = \{y_1, y_2, y_3, y_4\}$.

Suppose $f : X \times Y \to \{0, 1\}$ is defined as follows:

	y_1	y_2	y_3	y_4
x_1	0	1	1	1
x_2	0	0	1	1
x_3	0	0	0	1
x_4	0	0	0	0

Give a protocol $\mathcal{P}$ that computes f.

Communication Complexity

The cost of a protocol $\mathcal{P}$ on input (x, y) is the length of the branch taken. The cost of a protocol $\mathcal{P}$ is the height of the tree defining $\mathcal{P}$.

The *communication complexity* $D(f)$ of the function $f : X \times Y \rightarrow Z$ is the cost of the least costly protocol $\mathcal{P}$ computing f.

There's a simple upper bound for arbitrary functions:

Proposition 1 *$D(f) \leq \log_2 |X| + \log_2 |Z|$ for any $f : X \times Y \rightarrow Z$.*

Alternative definitions of communication complexity are possible:

- We could drop the requirement that *both* players need to know $f(x, y)$ in the end. Changes complexity by at most $\log_2 |Z|$.
- We could require the protocol to be strictly *alternating*. Changes complexity by at most a factor of 2.

Example

Suppose $X = Y = \{1..n\}$ and $f = \max$.

That is, Alice and Bob each hold a positive integer $\leq n$ and they want to compute the value of the larger one of their two numbers.

A possible protocol:

- Alice sends her x to Bob: $\log_2 n$ bits.
- Bob computes the maximum and sends it back: $\log_2 n$ bits.

Hence, $D(\max) \leq 2 \cdot \log_2 n$. This exactly matches the upper bound of Proposition 1, so is not too exciting ...

Example

Suppose $X = Y = 2^{\{1..n\}}$ and let $f(x, y)$ be defined as the *median* of the multiset $x \cup y$ in case $|x \cup y|$ is odd, and 0 otherwise.

Proposition 1 predicts $D(f) \leq n + \log_2(n + 1)$.

But there is a better protocol:

- Check we are not in the situation where $|x \cup y|$ is even: $O(1)$.
- For the main protocol, Alice and Bob maintain an interval $[i, j]$ containing the median, initially $[1, n]$.
- In each round, both compute $k = \frac{1}{2} \cdot (i + j)$. Alice tells Bob how many of her numbers are below and above k: $O(\log n)$. Bob can then check whether the median is below or above k and tell Alice (1 bit).
- So in each round the players can halve the interval. Hence, after $O(\log n)$ rounds they must have narrowed it down to one number.

Hence, $D(f) \in O(\log^2 n)$.

Btw, there's an even better protocol (see Kushilevitz and Nisan).

Boolean Functions

From now on we only consider Boolean functions f: $Z = \{0, 1\}$.

This is not a serious restriction. We could always decompose the function f into several Boolean functions, one each for computing each of the bits in the binary representation of the value of f.

Lower Bounds

So far we have only discussed *upper bounds* for $D(f)$. We have seen one general (but fairly trivial) upper bound for arbitrary f, and we have seen how clever protocols can provide better upper bounds.

Next we are going to see two results that will allow us to establish *lower bounds* for $D(f)$.

Think of f as being represented by a matrix ($\leadsto$ earlier example). Whenever we go left (right) from an a-node, we are excluding some rows; and similarly for b-nodes and columns. That is, we are partitioning the matrix into (not necessarily connected) *rectangles*.

Rectangles

A *rectangle* in $X \times Y$ is a subset $R \subseteq X \times Y$ such that there exist some $A \subseteq X$ and $B \subset Y$ with $R = A \times B$.

An alternative characterisation: $R \subseteq X \times Y$ is a rectangle iff $(x, y') \in R$ whenever $(x, y) \in R$ and $(x', y') \in R$.

Lemma 1 *Let $\mathcal{P}$ be a protocol and let R_ℓ be the set of inputs (x, y) for which $\mathcal{P}$ reaches the leaf ℓ. Then R_ℓ is a rectangle.*

<u>Proof:</u> Suppose $(x, y), (x', y') \in R_\ell$. Need to show that $(x, y') \in R_\ell$. Follow the branch taken for input (x, y'). Whenever we are in an a-node, Alice will only consider her part of the input and behave as for (x, y). Whenever we are in a b-node, Bob will only consider his part of the input and behave as for (x', y'). hence, we take the same branch in all three cases. ✓

Monochromatic Rectangles

A subset $R \subseteq X \times Y$ is called *f-monochromatic* iff f gives the same value for all $(x, y) \in R$.

Observe that for any protocol computing f, R_ℓ (the set of inputs reaching leaf ℓ) must be f-monochromatic.

Proposition 2 *If partitioning $X \times Y$ into f-monochromatic rectangles requires at least t rectangles, then $D(f) \geq \log_2 t$.*

Proof: By Lemma 1 and above observation, any protocol $\mathcal{P}$ computing f induces a partition (given by the R_ℓ's) of $X \times Y$ into f-monochromatic rectangles. If t is the number of rectangles (leafs), then the tree has a height $\geq \log_2 t$. ✓

Of course, this condition is not easy to check . . .

Fooling Sets

The *fooling set technique* is a technique for proving lower bounds: Find a (large) set of input pairs such that no two of them can belong to the same monochromatic rectangle. Then the previous result becomes applicable for a large value of t.

A set $S \subset X \times Y$ is called a *fooling set* for $f : X \times Y \to \{0, 1\}$ iff there exists a $z \in \{0, 1\}$ such that

- $f(x, y) = z$ for all $(x, y) \in S$; and
- $f(x, y') \neq z$ or $f(x', y) \neq z$ for all distinct $(x, y), (x', y') \in S$.

Proposition 3 *If f has a fooling set of size t, then $D(f) \geq \log_2 t$.*

Proof: We show that no f-monochromatic rectangle R can contain more than one pair from S. Suppose otherwise: $(x, y), (x', y') \in R$. Because R is a rectangle, we have $(x, y'), (x', y) \in R$. By the second condition $f(x, y') \neq z$ or $f(x', y) \neq z$, which contradicts the first condition.
$\rightsquigarrow$ At least t monochromatic rectangles, and Proposition 2 applies. ✓

Fooling Sets: Refinement

For any fooling set S we need to choose a value $z \in \{0, 1\}$.

To be precise, the size t of S is a lower bound on the number of rectangles of colour z. If we can find one fooling set for $z = 0$ and one for $z = 1$ then the sum of their sizes is a lower bound for the number of rectangles.

Hence, we can use this refined fooling set technique:

- Find a fooling set S_0 of size t_0 using $z = 0$.
- Find a fooling set S_1 of size t_1 using $z = 1$.
- Then $D(f) \geq \log_2(t_0 + t_1)$.

Example

Let $X = Y = \{0,1\}^n$ be the set of n-bit strings and let f be the *equality* function returning $f(x,y) = 1$ iff $x = y$.

Upper bound: $D(f) \leq n+1 \quad (= \log_2 |X| + \log_2 |Z|)$

Fooling set for $z = 1$: $S_1 = \{(\alpha, \alpha) \mid \alpha \in \{0,1\}^n\}$
We check the two conditions:

- $f(x,y) = 1$ for all $(x,y) \in S_1$ ✓
- $f(x,y') \neq z$ or $f(x',y) \neq z$ for all distinct $(x,y), (x',y') \in S_1$. ✓

The size of S_1 is $t_1 = 2^n$.

Fooling set for $z = 0$ of size $t_0 = 2^n$: similar (corresponding to righthand neighbours of cells on diagonal)

Hence, we get as a lower bound $D(f) \geq \log_2(2^n + 2^n) = n + 1$.

Conclusion

- Communication complexity studies the amount of information that a number of players need to exchange to jointly compute the value of a function, if each of them only know part of the input. Computational restrictions are not considered.
- We have presented some basic results and examples for the *two-part model* introduced by Yao in 1979. This is a big research area with many extensions to the basic model.
- Fooling sets provide a powerful technique for proving lower bounds for the communication complexity of a given function.