CoCoNuT - Complexity - Lecture 6

N.P. Smart

Dept of Computer Science
University of Bristol,
Merchant Venturers Building

April 25, 2016

Outline

Communication Complexity

(Merlin and Arthur) and (Arthur and Merlin)

IP=PSPACE

Zero-Knowledge

Communication Complexity

The world consists of communicating computers

So we need to model this.

Image two PTMs with two special "communication tapes".

- PTM T_1 writes to tape t_1 and reads from tape t_2.
- PTM T_2 writes to tape t_2 and reads from tape t_1.
- PTM T_1 goes first, when it stops it hands control to PTM T_2.
- PTM T_2 then reads what T_1 has just written on tape t_1
- PTM T_2 then executes, when it stops it hands control to PTM T_2.
- PTM T_1 then reads what T_2 has just written on tape t_2
- ... and so on...

Communication Complexity

Can measure communication complexity in terms of

- Number of tape cells written (size of communication)
- Number of times we hand over control (rounds of communication).

In theory, and often in practice, rounds is more important.

Verifying Proofs

When we discussed NP we thought of proofs of statements which could be verified in poly-time by a *deterministic* TM.

The verifier is a DTM V which takes as input

- The element x
- A witness w for $x \in \mathcal{L}$.

The witness is magically produced by an infinitely powerful prover

- Lets call this prover Merlin

Now replace our DTM verifier by a PTM verifier which has an input randomness tape r.

- Lets call this verifier Arthur

Merlin-Arthur Games

Since Arthur is randomized two things can go wrong:

Completeness:
If $x \in \mathcal{L}$ then for the proof w and randomness r we have

$$\Pr[V(x, w, r) = 1] \geq 2/3.$$

i.e. Valid proofs are likely to verify.

Soundness:
If $x \notin \mathcal{L}$ then for all "proofs" w and randomness r we have

$$\Pr[V(x, w, r) = 1] \leq 1/3.$$

i.e. Proofs of invalid statements are likely to reject

- i.e. Merlin may try to trick us to accept something which is false

It's Good To Talk!!

The complexity class MA is defined by the above conditions

- Basically it is NP but with a BPP verifier
- We have NP $\subset$ MA and BPP $\subset$ MA.
- Conjecture: MA $=$ NP.

The class MA is like a lecture in which students do not ask questions.

- They get the proof from the lecturer
- They then verify it using some random process.

We tell students to ask questions

- So what happens if we allow Arthur to ask questions of Merlin?
- Does Arthur learn more stuff?

It's Good To Talk!!

The complexity class MA is defined by the above conditions

- Basically it is NP but with a BPP verifier
- We have NP $\subset$ MA and BPP $\subset$ MA.
- Conjecture: MA $=$ NP.

The class MA is like a lecture in which students do not ask questions.

- They get the proof from the lecturer
- They then verify it using some random process.

We tell students to ask questions

- So what happens if we allow Arthur to ask questions of Merlin?
- Does Arthur learn more stuff?

It's Good To Talk!!

The complexity class MA is defined by the above conditions

- Basically it is NP but with a BPP verifier
- We have NP $\subset$ MA and BPP $\subset$ MA.
- Conjecture: MA $=$ NP.

The class MA is like a lecture in which students do not ask questions.

- They get the proof from the lecturer
- They then verify it using some random process.

We tell students to ask questions

- So what happens if we allow Arthur to ask questions of Merlin?
- Does Arthur learn more stuff?

Arthur-Merlin Games

Define the class AM[k] as a k round protocol of the following form:

- If k odd, the first message comes from Merlin.
- If k even, the first message comes from Arthur.
- Merlin and Arthur take it in turn to send each other messages.
- Let message in round i be m_i.
- There are k messages.
- Last message is from Merlin to Arthur.

In the end Arthur runs a DTM algorithm

- With input $m_1, \ldots, m_k$ and x
- Decides on whether $x \in \mathcal{L}$.

Note in AM Arthur's final algorithm is deterministic, in MA Arthur is probabilistic.

- In AM Arthur can always send a random string as his first message!

Arthur-Merlin Games

Let Arthur's final DTM algorithm be called M

Completeness:

$$\text{If } x \in \mathcal{L} \text{ then } \Pr[M(x, m_1, \ldots, m_k) = 1) \geq 2/3.$$

Soundness:

$$\text{If } x \notin \mathcal{L} \text{ then } \Pr[M(x, m_1, \ldots, m_k) = 1) \leq 1/3.$$

Arthur-Merlin Games

For fixed k we have AM[k] $=$ AM[2]

- This means asking too many (but a fixed number of) questions does not gain you anything more!
- Proof is too advanced for this course but see `http://courses.cs.washington.edu/courses/cse532/04sp/lect10.pdf`
- So we call AM[k] $=$ AM.

MA $\subset$ AM $=$ AM[3].

- Merlin sends witness w
- Arthur sends randomness r
- Merlin sends something else, which Arthur ignores.

So interaction does not seem to buy us much, when the number of rounds is fixed.

But.... AM[poly(n)] is much bigger than MA

Arthur-Merlin Games

For fixed k we have AM[k] $=$ AM[2]

- This means asking too many (but a fixed number of) questions does not gain you anything more!
- Proof is too advanced for this course but see `http://courses.cs.washington.edu/courses/cse532/04sp/lect10.pdf`
- So we call AM[k] $=$ AM.

MA $\subset$ AM $=$ AM[3].

- Merlin sends witness w
- Arthur sends randomness r
- Merlin sends something else, which Arthur ignores.

So interaction does not seem to buy us much, when the number of rounds is fixed.

But.... AM[poly(n)] is much bigger than MA

Arthur-Merlin Games

For fixed k we have AM[k] $=$ AM[2]

- This means asking too many (but a fixed number of) questions does not gain you anything more!
- Proof is too advanced for this course but see `http://courses.cs.washington.edu/courses/cse532/04sp/lect10.pdf`
- So we call AM[k] $=$ AM.

MA $\subset$ AM $=$ AM[3].

- Merlin sends witness w
- Arthur sends randomness r
- Merlin sends something else, which Arthur ignores.

So interaction does not seem to buy us much, when the number of rounds is fixed.

But.... AM[poly(n)] is much bigger than MA

IP and IP[k]

We call AM[poly(n)] $=$ IP.

- IP is the class of interactive proofs

We define IP[k] in the same way as we defined AM[k] except Arthur runs a PTM at the end as opposed to a DTM.

We have

- BPP $=$ IP[0].
- NP $\subset$ IP[1] $=$ MA.
- IP[2] $\subset$ AM[4].

IP[poly(n)] $=$ AM[poly(n)] $=$ IP.

IP and IP[k]

We call AM[poly(n)] $=$ IP.

- IP is the class of interactive proofs

We define IP[k] in the same way as we defined AM[k] except Arthur runs a PTM at the end as opposed to a DTM.

We have

- BPP $=$ IP[0].
- NP $\subset$ IP[1] $=$ MA.
- IP[2] $\subset$ AM[4].

IP[poly(n)] $=$ AM[poly(n)] $=$ IP.

IP = PSPACE

How much bigger is IP than NP?

Answer: A huge amount as we can show IP = PSPACE!!!!

This is a truly amazing result.

- Poly time bounded machines if they use randomness and communication can compute more than if they had a huge amount of space to play with, and more time!

IP $\subset$ PSPACE

Let (P, V) be an IP protocol for $\mathcal{L} \in$ IP.

We are given a *fixed* verifier V for $\mathcal{L}$.

We want to show we can come up with a PSPACE algorithm A to decide $x \in \mathcal{L}$

- Without knowing P

Define $p(x, m_1, \ldots, m_i)$ to be the maximal probability over all provers that x is accepted when the first i messages are $m_1, \ldots, m_i$.

IP $\subset$ PSPACE

If m_i is from P to V then

$$p(x, m_1, \ldots, m_{i-1}) = \max_{m_i} p(x, m_1, \ldots, m_i).$$

If m_i is from V to P then

$$p(x, m_1, \ldots, m_{i-1}) = \mathbb{E}_{r_i} p(x, m_1, \ldots, m_i),$$

where r_i are the random bits used by V to determine m_i

Note $p(x)$ is the maximal probability that V accepts x over all provers.

So A computes $p(x)$ if this is close to one, then we accept $x \in \mathcal{L}$

- Otherwise reject

Algorithm A can be run in PSPACE using the same trick as in the proof that $TQBF \in$ PSPACE

PSPACE $\subset$ IP

This direction is much harder.
In notes (Advanced bit) we show $TQBF \in$ IP.
In the main bit (and this lecture) we show $3 - SAT_K \in$ IP.

$3{-}SAT_K = \{\phi : \phi$ is a 3-CNF with exactly K satisfying assingments.$\}$.

We can write such a formula ϕ as a *polynomial* of degree at most $3m$

$$P_\phi(X_1, \ldots, X_n)$$

which evaluates to one on satisfying assignments and zero otherwise.

- Over *any* finite field $\mathbb{F}$.

The $3 - SAT$ bit is needed to bound the degree by $3m$.

- There is nothing magic about 3 it is just a constant!

PSPACE $\subset$ IP

We form

$$N_\phi := \sum_{X_1 \in \{0,1\}} \sum_{X_2 \in \{0,1\}} \cdots \sum_{X_n \in \{0,1\}} P_\phi(X_1, \ldots, X_n).$$

To prove $3 - SAT_K \in$ IP we need to produce a prover which can convince a verifier that $N_\phi = K$.

In general prover has show that if $g(X_1, \ldots, X_n)$ has degree d then

$$\sum_{X_1 \in \{0,1\}} \sum_{X_2 \in \{0,1\}} \cdots \sum_{X_n \in \{0,1\}} g(X_1, \ldots, X_n) = K.$$

We do this recursively using the protocol on the next slide.

Assume verifier can evaluate $g(X_1, \ldots, X_n)$ over $\mathbb{F}_p$ in poly time for any fixed assignment of the variables.

- So pick a prime $p \in [2^n, \ldots, 2^{n+1}]$. Note $K \lll p$.

PSPACE $\subset$ IP

Assume $n \geq 1$

- If $n = 1$ just check $g(0) + g(1) = K$.

Prover sends a description of the polynomial

$$h(X_1) = \sum_{X_2 \in \{0,1\}} \cdots \sum_{X_n \in \{0,1\}} g(X_1, \ldots, X_n).$$

Note this is a degree d polynomial in one variable.

Verifier rejects if $h(0) + h(1) \neq K$.

Otherwise pick $r \in \mathbb{F}$ and repeat on the smaller problem with input $g_2(r, X_2, \ldots, X_n)$ and $K_2 = h(r)$.

- i.e. Prove

$$\sum_{X_2 \in \{0,1\}} \cdots \sum_{X_n \in \{0,1\}} g_2(X_2, \ldots, X_n) = K_2.$$

PSPACE $\subset$ IP

Completeness: Trivial

Soundness: Suppose the prover tries to cheat and sends an incorrect h in some round.

- Call it h'
- We have $h'(0) + h'(1) = K$ (Otherwise prover would be found out).

The poly $h'(X_1) - h(X_1)$ has degree d and so d roots mod p.

The verifier therefore has a d/p probability of picking an r for which $h'(r) = h(r)$

- If he does not pick such an r the prover has something incorrect to prove (so hopeless for the prover)

Therefore prob that verifier successfully rejects is $(1 - d/p)^n$.

Example

Consider the boolean formula in CNF

$$\phi : (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee \not{x}_2)$$

I have picked a 2-CNF to make things easier, but 2 is just a constant.

Merlin claims this has $K = 3$ satisfying solutions.

- For an n variable formulae the brute force method will require 2^n work.
- Using interaction we pay $O(n)$ rounds plus polynomial computing

Example

We need to turn

$$\phi : (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee \not{x}_2)$$

into a polynomial.

To turn ϕ into a polynomial we arithmetize as follows, over any finite field:

- $x \wedge y$ we replace with $x \cdot y$
- $x \vee y$ we replace with $x + y - x \cdot y$
- $\not{x}$ we replace with $(1 - x)$.

We obtain

$$P_\phi(X_1, X_2, X_3) = (X_1 + X_2 - X_1 \cdot X_2) \cdot (X_2 + X_3 - X_2 \cdot X_3) \cdot (X_1 + (1 - X_2) - X_1 \cdot (1 - X_2)).$$

Example Round One

Merlin picks a large prime p and sends it to Arthur

- In our example let us pick $p = 101$.

Merlin computes (he is infinitely powerful)

$$h_1(X_1) = \sum_{X_2 \in \{0,1\}} \sum_{X_3 \in \{0,1\}} P(X_1, X_2, X_3) = 3 \cdot X_1$$

The polynomial $h_1(X_1)$ is sent to Arthur

Arthur checks that

$$h_1(0) + h_1(1) = K = 3.$$

Arthur picks a random integer r_1 mod p, say $r_1 = 13$, and computes

$$K_1 = h_1(13) \pmod{101} = 39.$$

The values (r_1, K_1) are sent to Merlin.

Example Round Two

Both parties can now compute

$$\begin{aligned} P_1 &= P_\phi(r_1) \pmod{101} \\ &= (43 \cdot X_3 + 58) \cdot X_2^3 + (15 \cdot X_3 + 43) \cdot X_2^2 \\ &\quad + (30 \cdot X_3 + 13) \cdot X_2 + 13 \cdot X_3. \end{aligned}$$

And we need Merlin to prove that the sum of this equation over all binary variables is $K_1 = 39$.

Merlin computes

$$h_2(X_2) = \sum_{X_3 \in \{0,1\}} P_1(X_2, X_3) = 159 \cdot X_2^3 + 56 \cdot X_2 + 13.$$

The polynomial $h_2(X_2)$ is sent to Arthur

Arthur checks that

$$h_2(0) + h_2(1) = K_1 = 39.$$

Example Round Three

Arthur picks a random integer r_2 mod p, say $r_2 = 7$, and computes

$$K_2 = h_2(7) \pmod{101} = 99.$$

The values (r_2, K_2) are sent to Merlin.

Both parties can now compute

$$P_2 = P_1(r_2) \pmod{101} = 52 \cdot X_3 + 74.$$

But this has one variable, and so it is trivial for Arthur to verify that

$$P_2(0) + P_2(1) = K_2 = 99 \pmod{101}.$$

Zero-Knowledge

Finally a variant of the class IP has huge applications in Cryptography.

The question is whether Arthur can take the "proof" from Merlin and use it to convince Lancelot that $x \in \mathcal{L}$.

- i.e. Can Lancelot skip the lecture and just copy the notes from Arthur?

We want Arthur's interaction to "buy" him something.

NB: In crypto Merlin's name is Peggy and Arthur's name is Victor, for Prover and Verifier respectively.

Zero-Knowledge

We say a protocol in IP is zero-knowledge if we can simulate it.

The transcript of a protocol is the set of messages $\{m_1, \ldots, m_k\}$

Simulation:
There is a PTM S which on input of x will output $m'_1, \ldots, m'_k$ such

- One cannot tell the difference between a valid transcript output by the real prover and verifier and the simulators output.

But what power does one have?

Perfect Zero-Knowledge: "One" is an infinitely powerful computer.

Computational Zero-Knowledge: "One" is an PTM

Zero-Knowledge

The class ZK is the set of languages which are accepted by zero-knowledge protocols.

- ZK $\subset$ IP.

The simulator pretends to be a valid prover against the specific verifier.

Note if the simulator can be run

- No point for Lancelot to copy, as Arthur could come up with convincing notes without turning up to the lecture.

It is known NP $\subset$ ZK

If one-way functions exist then IP $=$ ZK.

- Basically: If crypto is possible, then IP $=$ ZK.