INF561: Du calcul probabiliste au calcul quantique	**Hiver 2012**

Lecture 7 — 15 février

Lecturer: Frédéric Magniez *Scribe: Anna-Isavella Rerra , Danai Boutara*

The purpose of this course is to define the concept of interactive proof and apply it on simple examples.

7.1 Definition of the model

Definition 7.1. *(recall)*
$NP = \{L : \exists V$ *a deterministic and polynomial time algorithm (called verifier) s.t.*
$\forall x \in L, \quad \exists y \in \{0,1\}^{poly(|x|)}$ *s.t.* $V(x,y) = 1$ *and* $\Pr\{V(x,y,r) = 1\} = 1,$
$\forall x \notin L, \quad \exists y \in \{0,1\}^{poly(|x|)}$ *s.t.* $V(x,y) = 0$ *and* $\Pr\{V(x,y,r) = 0\} \geq \frac{1}{2}\}.$

Extensions

- Interaction $\Rightarrow IP_{det}$
- Randomness $\Rightarrow$ Merlin Arthur (MA)
- Interaction + Randomness $\Rightarrow$ IP

Definition 7.2. *IP*
Let V be a verifier and let P be a prover
$q_1 = V(x,1), \quad r_1 = P(x,1,a_1)$
...
$q_i = V(x,i,r_1,...,r_{i-1}), \quad r_i = P(x,i,a_1,...,a_i)$
The protocol of questions-responses ends after m messages and we note the result:
$< P,V > (x)$ *accepts or rejects*

Definition 7.3.
$IP_{det} = \{L : \exists V$ *a deterministic and polynomial time verifier V with a polynomial number of messages s.t.*
$\forall x \in L, \quad \exists P$ *prover s.t. out* $< P,V > (x) = 1$
$\forall x \notin L, \quad \forall P$ *prover s.t. out* $< V,P > (x) = 0\}.$

Question: IP_{det} vs. NP ?

- $NP \subseteq IP_{det}$
  Let $L \in NP \rightarrow V_o$
  $V = \{V(x) = 0$,no query $\quad$ or $\quad (V,P)(x) = V_o(x, P(x))\}$
  if $x \in L$ then $P(x) = y$ s.t. $V_o(x,y) = 1$
  if $x \notin L$ $\forall i, j$ $V_o(x,y) = 0$ $\quad$ and $\quad \forall P V_o(x, P(x)) = 0$

- $IP_{det} \subseteq NP$
  Let $L \in IP_{det} \rightarrow V$ and $V_o = out(x, a_1, a_2, ..)$
  if $x \in L$ then $\exists P$ that produces $a_1, a_2, ..$ s.t. $out(x, a_1, a_2, ..) = 1 \rightarrow y = a_1, a_2, ..$ and $V_o(x,y) = 1$
  if $x \notin L$ then $\forall P \quad out(x, a_1, a_2, ..) = 0 \rightarrow \forall y \quad V_o(x,y) = 0$

Definition 7.4 (Introduce random bits $V \in \{0,1\}^*$).
(We only use polynomially many of them)

1. *r private to $V \rightarrow P$ is deterministic*
2. *r public to V and $P \rightarrow P$ is deterministic but knows the coins of V*

Definition 7.5. *IP*
Let V be a randomized and polynomial time verifier V, let P be a prover and $r \in \{0,1\}^*$
$$q_1 = V(x,r), \quad a_1 = P(x, 1, q_1)$$
$$...$$
$$q_i = V(x, i, r_1, ..., r_{i-1}), \quad a_i = P(x, i, q_1, ..., q_i)$$
Voutput $out(V,P)(x,r) = out(x, a_1, a_2, .., r)$

Definition 7.6.
$IP = \{L : \exists V$ a randomized and polynomial time verifier with a polynomial number of messages s.t.
$$x \in L \Rightarrow \exists P \text{ s.t. } Pr_r(< P, V > (x,r) \text{ accepts}) = 1$$
$$x \notin L \Rightarrow \forall P \;\; Pr_r(< P, V > (x,r) \text{ accepts}) \leq 1/2\}$$

7.2 First examples

We look at problems in coNP.

7.2.1 Non-isomorphic graphs

Definition 7.7.
Let $G = (V, E)$ or $V = \{1, ..., n\}$ a graph and $\pi \in \mathcal{S}_n$,we define $\pi(G) = (V, E')$ the permuted graph s.t. : $(u,v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E'$.
$G_1 \cong G_2$ if it exists $\pi \in \mathcal{S}_n$ s.t. $\pi(G_1) = G_2$.
$GNI = \{(G_1, G_2) : G_2 \ncong G_2\}$

Theorem 7.8. *$GI \in NP$ and $GNI \in coNP, IP$*

Proof: Just give the permutation. □

Definition 7.9.
$H = G_b$ permuted by π, P computes bit c
output $= 1$ if $b = c$ or output $= 0$ if $b \neq c$.

Lemma 7.10. *If $(G_1, G_2) \in GNI$ then the set of permuted graphs $O_1 = \{G_1$ permuted by $\pi : \pi \in \mathcal{S}_n\}$ and $O_2 = \{G_2$ permuted by $\pi : \pi \in \mathcal{S}_n\}$ are disjoint. That means that either $H \in O_1$ or $H \in O_2$.*

Definition 7.11.
Let $P(H) = c$ s.t. $H \in O_c$. Since $O_1 \cap O_2 = \emptyset$, $P(H)$ is well-defined and $P(H) = b$. Therefore $Pr(output = 1) = 1$.

Lemma 7.12. *If $(G_1, G_2) \notin GNI$ then G_1, G_2 are isomorphic. Therefore $O_1 = O_2$.*
So H is a random graph of $O_1 = O_2$.
As a result, $\forall P \rightarrow Pr_r[out(V, P)(x, r) = 0] = \frac{1}{2}$.

Definition 7.13.
$IP(k) = IP$ with only k messages.

Theorem 7.14. *$\forall$constant k, $IP(k) \subseteq IP(2)$*

Theorem 7.15. *$\forall$constant k, $IP[k+1] \subseteq IP[k]$*

Theorem 7.16. *$IP = PSPACE$ with poly many messages.*

We will prove a restricted version of that theorem.

Definition 7.17. *$\sharp SAT_D = \{(\varphi, k)$ where $\varphi =$ 3-SAT formula and $k =$ number of positive assignments to $\varphi\}$.*

Theorem 7.18. *$\sharp SAT_D \in IP$*

7.2.2 Proof of $\sharp SAT_D \in IP$

Arithmetization. Consider a formula $\varphi = (0,1)^n \rightarrow (0,1)$ with n variables. We want to construct in polynomial time a low degree polynomial R_φ in n variables s.t.

$$\forall a \in \{0,1\}^n, \quad R_\varphi(a_1, a_2, .., a_n) = \varphi(a_1, a_2, .., a_n).$$

Construction by induction over any field:

- $x \rightarrow x$
- $\overline{x} \rightarrow 1 - x$
- $\overline{\varphi} \rightarrow 1 - R_\varphi$

- $\varphi_1 \wedge \varphi_2 \rightarrow \varphi_1\varphi_2$
- $\varphi_1 \vee \varphi_2 \rightarrow 1-(1-\varphi_1)(1-\varphi_2)$

Lemma 7.19.
$\deg R_\varphi \leq 3m$ *where* m *is the number of clauses in* φ.
$\forall a \in \{0,1\}^n, \quad \varphi(a) = R_\varphi(a)$.
We can compute a representation of R_φ *in linear time.*

We now consider the problem of checking that $\sum_{x_1,\ldots,x_n\in\{0,1\}} p(x_1,x_2,\ldots,x_n) = c \mod q$, where p is some polynomial of degree at most d. Then $\sharp SAT_D$ reduces to this problem by letting $p = R_\varphi$ and $q > 2^n$ (since the number of solutions of φ is at most 2^n).

Sumcheck protocol.

Definition 7.20 (IP protocol for $\mathit{Sumcheck}_{q,n}(p,c)$**).**

- *p a polynomial with n variables and c a natural integer.*
- *If $n = 1$, check that $p(0)+p(1) = c$ (if $\neq$ reject, otherwise accept)*
- *If $n > 1$, ask from the prover the polynomial* $p'(x) = \displaystyle\sum_{x_2,\ldots,x_n\in\{0,1\}} p(x,x_2,\ldots,x_n)$.
- *Check that $p'(0)+p'(1) = c$ (if $\neq$ reject, otherwise continue)*
- *Choose at random $r \in \mathbb{Z}_q$ and execute* $\mathit{Sumcheck}_{q,n-1}(p(r,\ldots),p'(r))$.

Theorem 7.21. *If* $\displaystyle\sum_{x\in\{0,1\}^n} p(x) = c \mod q$ *then* $\mathit{Sumcheck}_{q,n}(p,c)$ *accepts.*
Otherwise it rejects with probability at least $1-\frac{nd}{q}$, *where* $d = \deg p$.

Proof:
Case $\sum_{x\in\{0,1\}^n} p(x) = c \mod q$**.**
The proof is also by induction on n. If $n=1$ we have $p(0)+p(1)=c$, therefore $< P,V > (p,c)$ accepts.
Otherwise:

$$\begin{aligned} p'(0)+p'(1) &= \sum_{x_2,\ldots,x_n\in\{0,1\}} p(0,x_2,\ldots,x_n) + \sum_{x_2,\ldots,x_n\in\{0,1\}} p(1,x_2,\ldots,x_n) \\ &= \sum_{x_1,\ldots,x_n\in\{0,1\}} p(x_1,\ldots,x_n) = c. \end{aligned}$$

And by induction $\mathit{Sumcheck}_{q,n-1}(p(r,\ldots),p'(r))$ accepts so $< P,V > (p,c)$ accepts.

Case $\sum_{x\in\{0,1\}^n} p(x) \neq c \mod q$**.**
The proof is also by induction on n. If $n=1$ the verifier always rejects, therefore the result is true.

Let $n > 1$ be an integer. If $p'(x) = \sum_{x_2,...,x_n \in \{0,1\}} p(x, x_2, ..., x_n)$ (ie P is the honest prover) then $p'(0) + p'(1) \neq c$ so the verifier always rejects.
Otherwise $p'(x) \neq \sum_{x_2,...,x_n \in \{0,1\}} p(x, x_2, ..., x_n)$ and we deduce:

$$\begin{aligned}
& \Pr(\mathit{Sumcheck}_{q,n}(p, c) \text{ accepts}) \\
\leq\ & \Pr_r\Big(\sum_{x_2,...,x_n \in \{0,1\}} p(r, x_2, ..., x_n) = p'(r)\Big) \\
& + \Pr_r \Big(\mathit{Sumcheck}_{q,n-1}(p(r, ...), p'(r)) \text{ accepts } \text{ and } \sum_{x_2,...,x_n \in \{0,1\}} p(r, x_2, ..., x_n) \neq p'(r))\Big).
\end{aligned}$$

The first probability term is upper bounded by $\frac{d}{q}$ using the Shwartz-Zippel lemma, and the second probability term by $\frac{d(n-1)}{q}$ using the induction hypothesis. Which shows the induction hypothesis for n and completes the proof. $\square$

Corollary 7.22. $\overline{3 - SAT} \in IP$

Proof: Let q be a prime number $> 2^n$. Then just run $\mathit{Sumcheck}_{q,n}(P_\varphi, 0)$.
φ not satisfiable $\Rightarrow \mathit{Sumcheck}_{q,n}(P_\varphi, 0)$ accepts.
φ satisfiable $\Rightarrow \Pr(\mathit{Sumcheck}_{q,n}(P_\varphi, 0) \text{ rejects}) \geq 1 - \dfrac{3mn}{2^n}$. $\square$

7.3 Program checking

Definition 7.23.
T is a computational task.
A checker for T is a poly time and randomized algo C s.t. given any program P satisfies :

1. *if P is correct then $\forall y \rightarrow P(y) = T(y)$ and $Pr(C^p accepts P(x)) = 1$*
2. *if $P(x) \neq T(x)$ then $Pr(C^p rejects P(x)) \leq \frac{1}{2}$*

Complexity of C

- *the number of calls to P*
- *runtime complexity of C (where each call to P has zero cost)*
- *we want the number of calls to be small*
- *we want runtime complexity to be negligeable to the runtime complexity of any correct program*