Motivation: Provers and Verifiers

Recall: languages in NP admit short, easy-to-check membership certificates

**NP membership checking as an interaction of two parties:**
- The Prover produces a certificate (proof of membership) that it claims to be valid
- The Verifier validates the certificate to decide upon acceptance

**Can we generalise this idea?**
- A (untrusted) Prover tries to convince the Verifier of membership
- Verifier sceptically checks the Prover’s arguments before making a decision
- The interaction might involve several rounds of communication
- The Prover might have unbounded computational power, but the Verifier should operate in P

For which languages can such a polytime Verifier ensure that it can be convinced of membership exactly for the words that really are in the language?

Example: Graph Isomorphism

We consider (undirected) graphs over a set of numbered vertices 1, 2, ..., n.

Two graphs are isomorphic if one can be obtained from the other by a bijective renaming (permutation) of vertices.

**Observations:**
- **Graph Isomorphism** is in NP (certificate: renaming)
- There are $n!$ many potential permutations, so exhaustive checking requires exponential time
- However, **Graph Isomorphism** is not known (or believed) to be NP-hard
Graph Non-Isomorphism

There does not seem to be a short certificate for this, but there is an interactive protocol:

**Protocol:** given non-isomorphic graphs $G_1$ and $G_2$
- **Verifier:** randomly select $i \in \{1, 2\}$; randomly permute vertices of $G_i$ to obtain a new graph $H$; send $H$ to the Prover
- **Prover:** determine which $G_j$ ($j \in \{1, 2\}$) the graph $H$ is isomorphic to; send $j$
- **Verifier:** accept if $i = j$, else reject

**Analysis:** The Prover can ensure acceptance for non-isomorphic graphs, but for isomorphic graphs it can only achieve acceptance with probability $0.5$ (which can be reduced further by repeating the interaction several times)

Zero-Knowledge Proofs

Running the previous protocol is interestingly uninformative:
- The Verifier can be convinced that $(G_1, G_2) \in \text{GRAPH NON-ISOMORPHISM}$
- But the Verifier learns nothing about the reasons
- In particular, the Verifier would not be able to prove this to anybody else

This is called a zero-knowledge proof.

Note: The mathematical property that characterises such proofs formally is that the Verifier could have produced the whole interaction all by itself, without the assistance of a Prover. This would not convince the Verifier, of course, but would not be distinguishable otherwise.

Making interactive proofs formal (1)

The interaction can be viewed as a sequence of messages $m_1, m_2, \ldots, m_k$, followed by a sequence of messages $m_k, m_{k+1}, \ldots, m_{k+m}$, followed by a function $P : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

**Definition 26.3:** A polynomial verifier $V$ with bound $p$ and a prover $P$ accept a word $w$ with probability $\Pr[V \leftrightarrow P \text{ accepts } w] = \Pr_{r \in \{0,1\}^{p|w|}}[V \leftrightarrow P](w, r) = \text{accept}$. 

Note: Polynomial verifiers could, for example, use messages to store the number of available steps that remain, and reject when this is used up.
The class IP

We can now formally define a class of languages that are accepted by polytime Verifiers using interactive proofs:

**Definition 26.4:** A language $L$ is in IP if there is a polynomial verifier $V$ such that, for every word $w$:

1. If $w \in L$ then there is a prover $P$ with $\Pr[V \leftrightarrow P \text{ accepts } w] \geq \frac{2}{3}$.
2. If $w \notin L$ then for all provers $\tilde{P}$ we have $\Pr[V \leftrightarrow \tilde{P} \text{ accepts } w] \leq \frac{1}{3}$.

In words:

- there is a “good” prover $P$ that can convince $V$ to accept $w \in L$ with high probability
  (note that the existence of one good prover for each $w \in L$ implies that there is one globally good prover)
- not even a “bad” prover $\tilde{P}$ can convince $V$ to accept words $w \notin L$ with more than a low probability

**Theorem 26.6:** $NP \subseteq IP$.

**Proof:** Use definition of NP via polynomial-time verifiers.

**Theorem 26.7:** $BPP \subseteq IP$.

**Proof:** Verifier can solve BPP problems without talking to Prover.

Obvious sub-classes of IP

Some observations are straightforward:

**Theorem 26.8:** $IP \subseteq PSpace$.

**Proof:** Consider $L \in IP$ with polynomial verifier $V$. For any word $w$, let

$$\Pr[V \text{ accepts } w] = \max_P \Pr[V \leftrightarrow P \text{ accepts } w].$$

Then $\Pr[V \text{ accepts } w] \geq \frac{2}{3}$ if $w \in L$ and $\Pr[V \text{ accepts } w] \leq \frac{1}{3}$ otherwise.

**Goal:** Compute the value of $\Pr[V \text{ accepts } w]$ in PSpace.

**Notation:**

- Let $M_j$ abbreviate a message sequence $m_1 \# m_2 \# \cdots \# m_j$
- $(V \leftrightarrow P)(w, r, M_j) = \text{ accept if } (V \leftrightarrow P)(w, r) = \text{ accept for a message sequence } m_1 \# m_2 \# \cdots \# m_j \text{ that extends } M_j \text{ (in particular: } M_j \text{ is possible with } r, V \text{ and } P)$
- $\Pr[(V \leftrightarrow P) \text{ accepts } w \text{ starting from } M_j] = \Pr_{r \in \{0,1\}^*}(V \leftrightarrow P)(w, r, M_j) = \text{ accept}$
- $\Pr[V \text{ accepts } w \text{ starting from } M_j] = \max_r \Pr[(V \leftrightarrow P) \text{ accepts } w \text{ starting from } M_j]$

Probabilistic interactions

The definition of IP uses probabilistic computations

- The Verifier is a polynomially time-bounded probabilistic TM
- The Prover does not use randomness (and including it would not change IP)
- As discussed for BPP, we can amplify probabilities; in particular, the bounds $\frac{2}{3}$ and $\frac{1}{3}$ are not essential to the definition

The use of randomness in the Verifier is important for expressive power:

**Theorem 26.5:** Let $IP_d$ be the restriction of IP that is obtained when requiring $V$ to be deterministic (ignoring the random bits). Then $IP_d = NP$.

The proof is not hard (exercise; or see Arora & Barak, Lemma 8.4)

A superclass of IP

Interestingly, we can use another well-known class to capture IP from above:

**Theorem 26.8:** $IP \subseteq PSpace$.
Theorem 26.8: IP ⊆ PSpace.

Proof (cont.): What we seek is \( \Pr[V \text{ accepts } w] = \Pr[V \text{ accepts } w \text{ starting from } M_0] \), where \( M_0 \) is the empty message sequence.

We define numbers \( N[M_j] \) recursively, with the longest possible sequences as base case:

1. If \( M_j \) is not a 1 or \( M_j = m_1 \# \cdots \# m_0 \), where \( M_0 \) is the empty message sequence.
2. Else, if \( j \) is odd and \( M_j = m_1 \# \cdots \# m_0 \), then
   - If \( m_j = \) then \( N[M_j] = 1 \)
   - If \( m_j = \) then \( N[M_j] = 0 \)
   - If \( m_j \neq \) and \( m_j = \) then \( N[M_j] = \max_{m_{j+1}} N[M_j # m_{j+1}] \)
3. Else, if \( j \) is even, then \( N[M_j] = \text{wt-avg}_{m_j} N[M_j # m_{j+1}] \)
   - where \( \text{wt-avg}_{m_j} N[M_j # m_{j+1}] = \sum_{m_j} \Pr_{r \in \{0, 1\}^{|r|}} [V(w, r, M_j) = m_{j+1}] \cdot N[M_j # m_{j+1}] \)

In all cases, \( m_{j+1} \) ranges over (a superset of) the messages possible at this step (which can be assumed to be of polynomial length, and are therefore bounded).

Note 1: Case 2.3 corresponds to best possible answer of any Prover

Note 2: Case 3 corresponds to probability-weighted average for given Verifier

This finishes the proof of the theorem. □

Markus Krötzsch, 3rd Feb 2020 Complexity Theory slide 16 of 31
The power of the prover

Our definition of IP allows Prover to have unlimited computational power (possibly even uncomputable behaviour).

However, our proof of IP ⊆ PSpace showed that the optimal Prover output for any given Verifier can be computed in polynomial space, so we get:

**Corollary 26.9:** The class IP remains the same if the Prover is required to compute its responses in polynomial space.

Solving \( \#\text{SAT}_D \) in IP

We consider a formula \( \varphi \) of size \( n \) and with \( m \) propositional variables \( x_1, \ldots, x_m \).

For \( 1 \leq i \leq m \), let \( f_i : \{0, 1\}^i \rightarrow \mathbb{N} \) be the function that maps \( (a_1, \ldots, a_i) \) to the number of satisfying assignments of \( \varphi \) with \( x_1 = a_1, \ldots, x_i = a_i \).

- Then \( f_0() \) is the solution to \( \#\text{SAT} \).
- We find \( f_i(a_1, \ldots, a_i) = f_{i+1}(a_1, \ldots, a_i, 0) + f_{i+1}(a_1, \ldots, a_i, 1) \)

# The power of IP

So far, we know that IP contains NP, BPP, but also **Graph Non-Isomorphism**, which is not known to be in either class.

As we will see, IP can do much more. We start with the following problem:

**\( \#\text{SAT} \)**

- **Input:** A propositional logic formula \( \varphi \).
- **Problem:** The number of satisfying assignments of \( \varphi \).

**Note:**
- \( \#\text{SAT} \) is not a decision problem. Let \( \#\text{SAT}_D = \{\langle \varphi, k \rangle \mid \text{the solution of } \#\text{SAT} \text{ on } \varphi \text{ is } k \} \) be the corresponding decision problem.
- Computing \( \#\text{SAT} \) solves propositional satisfiability as well as unsatisfiability.
- Indeed, it is complete for the powerful class \( \#P \).

**Solving \( \#\text{SAT}_D \) in IP**

**Theorem 26.10:** \( \#\text{SAT}_D \in IP \)

We consider a formula \( \varphi \) of size \( n \) and with \( m \) propositional variables \( x_1, \ldots, x_m \).

For \( 1 \leq i \leq m \), let \( f_i : \{0, 1\}^i \rightarrow \mathbb{N} \) be the function that maps \( (a_1, \ldots, a_i) \) to the number of satisfying assignments of \( \varphi \) with \( x_1 = a_1, \ldots, x_i = a_i \).

- Then \( f_0() \) is the solution to \( \#\text{SAT} \).
- We find \( f_i(a_1, \ldots, a_i) = f_{i+1}(a_1, \ldots, a_i, 0) + f_{i+1}(a_1, \ldots, a_i, 1) \)

**Protocol:** to check if \( \langle \varphi, k \rangle \in \#\text{SAT}_D \)

- **P:** send \( f_0() \) to V
- **V:** check if \( f_0() = k \) and reject if this fails
- For \( i = 1, \ldots, m \):
  - **P:** send \( f_i(a_1, \ldots, a_i) \) to V for all \( (a_1, \ldots, a_i) \in \{0, 1\}^i \)
  - **V:** check, for all \( \bar{a} \in \{0, 1\}^{i-1} \), if \( f_{i-1}(\bar{a}) = f_i(\bar{a}, 0) + f_i(\bar{a}, 1) \), reject if not
- **V:** check if, for all \( (a_1, \ldots, a_m) \in \{0, 1\}^m \), \( f_m(a_1, \ldots, a_m) = 1 \) if and only if \( \{x_1 \mapsto a_1, \ldots, x_m \mapsto a_m\} \) is a satisfying assignment for \( \varphi \); accept iff

This protocol does not show \( \#\text{SAT}_D \in IP \):
- It requires exponential time to perform exponentially many checks.

However, the protocol is otherwise correct:
- If \( k \) is the correct result, a truthful Prover can convince the Verifier.
- If \( k \) is not correct, not even a mischievous Prover can convince the verifier (exercise: why?)
Arithmetisation

To reduce the number of messages and checks, we use arithmetisation.

Let $\varphi$ be transformed into an arithmetic expression $\Phi$ by replacing subexpressions:

- $\alpha \land \beta$ becomes $\alpha \beta$
- $\neg \alpha$ becomes $(1 - \alpha)$
- $\alpha \lor \beta$ becomes $1 - (1 - \alpha)(1 - \beta)$

Some observations:

- $\Phi$ is a multivariate polynomial function over variables $x_1, \ldots, x_n$
- The degree of $\Phi$ is bounded by the size $n$ of $\varphi$
- The value of $\Phi$ for inputs $x_i \in \{0, 1\}$ is also in $\{0, 1\}$, and corresponds to the valuation of $\varphi$ on the corresponding truth values
- We can evaluate $\Phi$ over an arbitrary field

Example 26.11: For a prime number $p$, the algebra of natural numbers $\{0, 1, \ldots, p-1\}$ and where $+$ and $\cdot$ are addition and multiplication modulo $p$ is a finite field. This field is denoted $\text{GF}(p)$.
The main insight about IP is as follows:

**Theorem 26.12: IP = PSpace**

Proof: We have already shown IP ⊆ PSpace. For the converse, we adopt our proof of #SAT, ∈ IP to show that TrueQBF ∈ IP. This suffices (why?).

Consider a QBF of the form \( \psi = \forall x_1. \exists x_2. \forall x_3. \cdots \exists x_m. \varphi[x_1, \ldots, x_m] \) (this is w.l.o.g. - why?). Using the arithmetisation \( \Psi \) of \( \varphi \), we find \( \psi \in \text{TrueQBF} \) if

\[
\sum \prod \sum \Phi(a_1, \ldots, a_m) = 1 \tag{2}
\]

where \( \sum_{a \in \{0,1\}} P(a) = P(0) + P(1) = 1 - (1 - P(0))(1 - P(1)) \).

We would like to verify (2) using similar ideas as for #SAT, ∈ IP.

Showing IP = PSpace

We write \( \exists x_1. R_1 x_2. R_2 x_1. R_2. \exists x_3. R_3 x_1. R_3. R_4. \cdots \exists x_m. R_m. \Phi(x_1, \ldots, x_m) \) as \( O_1 \psi_1. O_2 \psi_2. \cdots. O_{\ell} \psi_\ell. \Phi(x_1, \ldots, x_m) \), where \( O_i \in \{\exists, \forall, R\} \) and \( y_i \in \{x_1, \ldots, x_m\} \).

Verifier picks a prime \( p > n^4 \) (for \( n \) the size of \( \psi \)); we calculate in GF\( (p) \).

**Protocol:** to check \( K = O_1 \psi_1. O_2 \psi_2. \cdots. O_{\ell} \psi_\ell. g(b_1, \ldots, b_\ell) \mod p \) where \( O_i \psi_i. O_{\ell-1} \psi_{\ell-1}. \cdots. O_1 \psi_1. g(b_1, \ldots, b_\ell) \) is a polynomial in \( \ell \) variables that has a polynomial-size representation and polynomial degree

- \( V \): if \( k = 0 \), verify \( g(b_1, \ldots, b_\ell) = K \) and reject or accept accordingly; else, ask \( P \) for a representation of \( O_2 \psi_2. \cdots. O_{\ell} \psi_\ell. g(b_1, \ldots, b_\ell) [y_1 \mapsto \text{undef}] \)
- \( P \): send a polynomial \( \tilde{h}(y_1) \)
- \( V \): check if \( \tilde{h} \) is polynomially sized and of degree \( \leq m \); check if \( K = O_1 \psi_1. \tilde{h}(y_1) \); reject if any of these fail; pick a random \( b \in \text{GF}(p) \) and send \( b \) to \( P \)
- Recursively use the same protocol to verify \( \tilde{h}(b) = O_2 \psi_2. \cdots. O_{\ell} \psi_\ell. g(b_1, \ldots, b_\ell) [y_1 \mapsto b] \mod p \)
Explanations

The following notes may help to understand the protocol.

- The function $O_1y_1. \cdots . O_ky_k.g$ is a function on variables $x_1, \ldots, x_\ell$
  - Variables $x_i \ (i > \ell)$ are bound by $\exists$ or $\forall$, hence eliminated
  - Variables $x_i \ (i \leq \ell)$ may still occur in $R$ operators, but they do not remove them
- $O_2y_2. \cdots . O_ky_k.g$ is a function on variables $x_1, \ldots, x_\ell, x_{\ell+1}$ if $O_1 \in \{\exists, \forall\}$
- $O_2y_2. \cdots . O_ky_k.g(b_1, \ldots, b_\ell)[y_1 \mapsto \text{undef}]$ denotes the function over $y_1$ obtained by ignoring the binding $b_i$ for $y_1 = x_i$ (only relevant if $O_1 = R$)
- $O_2y_2. \cdots . O_ky_k.g(b_1, \ldots, b_\ell)[y_1 \mapsto b]$ denotes the function over $y_1$ obtained by redefining the binding $b_i$ for $y_1 = x_i$ to be $b$ (only relevant if $O_1 = R$)
- The check $K = O_1y_1. \tilde{h}(y_1)$ is evaluated as required for $O_1$

Finishing the proof

**Theorem 26.12: IP = PSpace**

**Proof:** Summary of approach:

- The problem is arithmetised and extended with degree-reduction operators
- A prime $p > n^4$ is chosen to define a filed $GF(p)$ for calculations
- A protocol is followed to verify the arithmetisation yields $1 = 1$

As in the case of $\#SAT_d$, the Prover’s chances of fooling the Verifier as small:

- Wrong claims require to send wrong polynomials $\tilde{h}(y_1)$
- It is unlikely that $V$ picks a random value $b$ on which $\tilde{h}(p)$ agrees with the correct function’s value ($p > n^4$ suffices here since the degree of the functions are small)

This finishes the proof. □

Summary and Outlook

Interactive proofs enable probabilistic machines to solve problems beyond NP

**Graph Non-Isomorphism** has an interesting interactive zero-knowledge proof protocol

IP = PSpace

What’s next?

- Summary & consultation
- Examinations