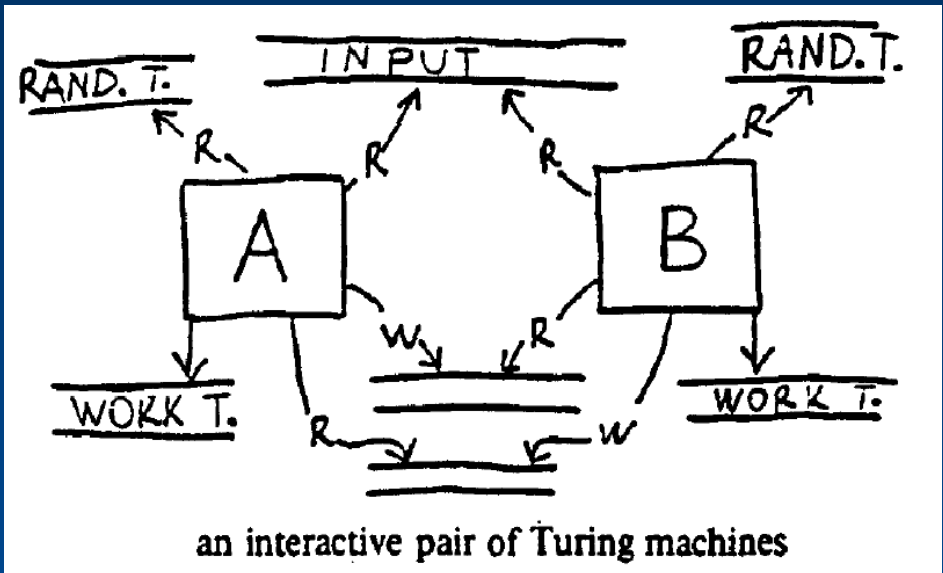
Διαλογικά συστήματα αποδείξεων (Interactive proof systems)



an interactive pair of Turing machines

Κρυπτογραφία & Πολυπλοκότητα
καθ. Στάθης Ζάχος
παρουσίαση: Νίκος Λεονάρδος

Εισαγωγή

Proof Systems:

- Η **απόδειξη** είναι μια διαδικασία που σκοπό έχει να πείσει κάποιον για την ισχύ μιας πρότασης (π.χ. $x \in L$).
- Το NP ως σύστημα αποδείξεων.

Interactive Proof Systems:

- Ενισχύουμε το NP με **interaction** και **randomization** και φτάνουμε στα Interactive Proof Systems.
- Graph non-isomorphism is in IP, Graph 3 coloring is in IP.
- $IP \subseteq PSPACE$

Arthur Merlin Games:

- Ορισμός των Arthur-Merlin Games ως combinatorial games.
- Η ιεραρχία που ορίζουν καταρρέει στο 2ο επίπεδο, δηλαδή,

για k > 2 AM = AM[k] = MA[k+1]

- **IP = AM** ως προς Language Recognition.

IP = PSPACE

Proof Systems and NP

- Μπορεί κανείς να φανταστεί ένα σύστημα αποδείξεων ως μια διαλογική διαδικασία μεταξύ ενός **Prover P** και ενός **Verifier V.**
- Ο **P** θέλει να πείσει τον **V** για την ισχύ μιας πρότασης.
- Ο **V** θέλει να αποδεχθεί έναν ισχυρισμό μόνο εάν είναι αληθής.

- Μία γλώσσα L είναι στο NP αν και μόνο αν για κάθε στοιχείο της L υπάρχει πολυωνυμικού μήκους απόδειξη ότι $x \in L$ ή πιο τυπικά:
- **$L \in NP$** <u>αvv</u> υπάρχει relation R_L recognizable in polynomial time & polynomialy balanced τέτοιο ώστε: **$L = \{x : (x,y) \in R_L\}$**

Μπορούμε να δούμε τον τελευταίο ορισμό ως εξής:

- Υπάρχει ένας παντοδύναμος **Prover**.
- Ο **P** θέλει να πείσει τον **Verifier** ότι **$x \in L$**
- Στέλνει στον **V** τεκμήριο **y**, πολυωνυμικά φραγμένο ως προς **x**
- Ο **Verifier** πρέπει να ελέγξει το τεκμήριο και να αποφανθεί σε πολυωνυμικό χρόνο αν αποδέχεται ή απορρίπτει την είσοδο.

Proof Systems and NP

NP Proof System for SAT

- Ο **Prover** θέλει να πείσει τον **Verifier** ότι η παρακάτω formula ανήκει στο SAT: $(x \vee y \vee z') \wedge (x' \vee y') \wedge z'$
- Ο **Prover** στέλνει στον **Verifier** μια ανάθεση αληθοτιμών στις μεταβλητές, που να ικανοποιούν τη formula, π.χ. $(x,y,z)=(F,T,F)$.

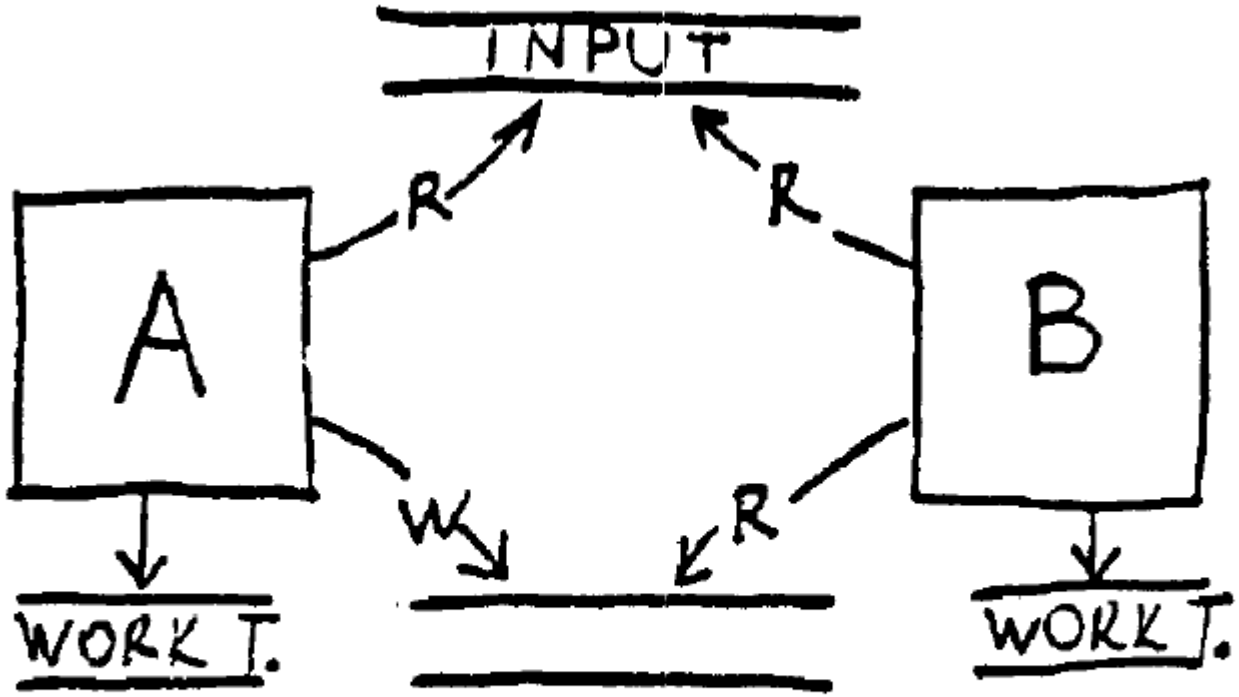
Κάτι τέτοιο είναι εύκολο για τον παντοδύναμο **Prover**, αν βέβαια μια τέτοια ανάθεση υπάρχει.

- Ο **Verifier** ελέγχει αν η ανάθεση που έλαβε ικανοποιεί τη formula

Ιδιότητες του NP Proof System:

- **Efficiency**: Η στρατηγική του **Verifier** είναι αποδοτική.
- **Correctness** requirements:
  - **Completeness:** Για μια αληθή πρόταση υπάρχει στρατηγική που να οδηγεί σε αποδοχή.
  - **Soundness:** Για μια ψευδή πρόταση δεν υπάρχει στρατηγική που να οδηγεί σε αποδοχή.

The NP Proof System



The *NP* proof-system

Interactive Proof: Παράδειγμα

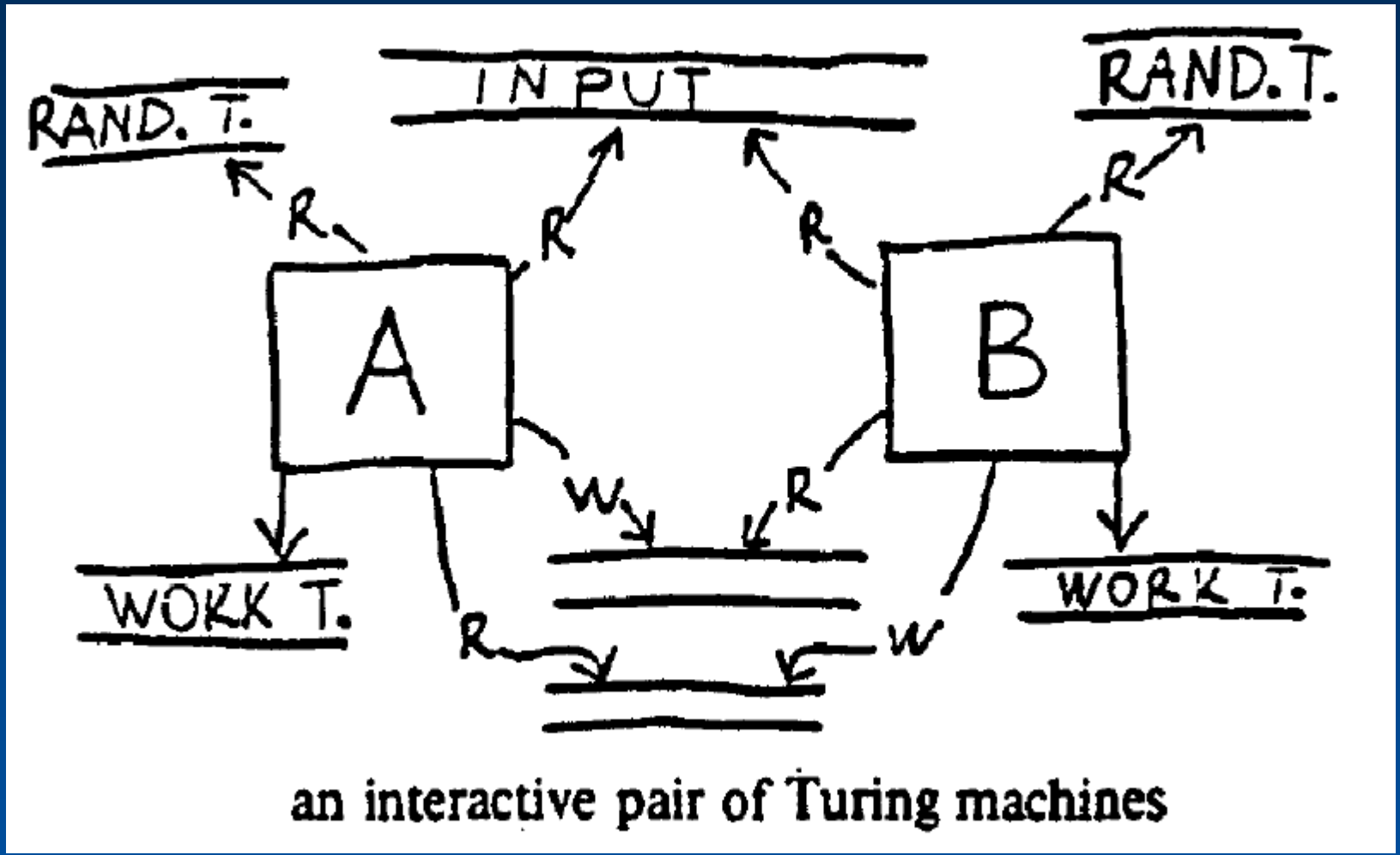
Πώς θα μπορούσαμε να πείσουμε κάποιον που πάσχει από αχρωματοψία, ότι δύο όμοιες μπίλιες έχουν διαφορετικό χρώμα;

- Έστω ότι ο **Prover** υποστηρίζει ότι οι μπίλιες έχουν διαφορετικό χρώμα (η μία μαύρο και η άλλη κόκκινο).
- Ο **Verifier** κρατάει τη μαύρη μπίλια με το δεξί χέρι και την κόκκινη με το αριστερό. Με τυχαίο τρόπο (κορώνα - γράμματα) και κρυφά από τον **Prover** αποφασίζει αν θα κάνει swap τις μπίλιες ή όχι. Το αποτέλεσμα είναι ότι ο **Verifier** γνωρίζει αν έκανε το swap ή όχι ενώ ο **Prover** μπορεί μόνο να μαντέψει, με πιθανότητα **1/2**.
- Ο **Verifier** αποκαλύπτει τις μπίλιες. Αν ο **Prover** αποτύχει να βρει σε πιο χέρι είναι η μαύρη και σε ποιο η κόκκινη, ο **Verifier** απορρίπτει με βεβαιότητα, ενώ αν ο **Prover** τα καταφέρει, ο **Verifier** πείθεται με πιθανότητα **1/2**.
- Το παραπάνω πρωτόκολλο μπορεί να **επαναληφθεί πολυωνυμικά πολλές φορές** $p(|x|)$ και έτσι η πιθανότητα να μαντέψει ο **Prover** όλα τα τυχαία bits του **Verifier** γίνεται $2^{-p(|x|)}$.

Τα νέα συστατικά της απόδειξης

- **Interaction:** Χωρίς δυνατότητα επικοινωνίας ο **Prover** θα έπρεπε να σκεφτεί μία γραπτή απόδειξη, κάτι που δεν φαίνεται να είναι δυνατόν.
- **Randomness:** Αν ο **Verifier** ήταν ντετερμινιστικός, ο **Prover** θα μπορούσε να τον κοροϊδέψει κοιτάζοντας τον κώδικά του.
- **Secrecy (private coins vs. public coins):** Ο **Verifier** έκρυψε το random bit του από τον **Prover** ώστε να ικανοποιείται η ιδιότητα **Soundness**. Αξιοσημείωτο είναι το γεγονός ότι κάτι τέτοιο δεν είναι απαραίτητο! Όπως θα αποδείξουμε λίγο αργότερα, υπάρχει αποδεικτικό σύστημα για το παραπάνω πρόβλημα, όπου ο **Verifier** δείχνει όλα τα random bits του στον **Prover**.

An interactive pair of Turing machines



an interactive pair of Turing machines

["The knowledge complexity of Interactive Proof Systems", 1985, Shafi Goldwasser (MIT), Silvio Micali(MIT), Charles Rackoff (Toronto)]

Interactive Proof System

Ένα Διαλογικό Σύστημα Αποδείξεων (**Interactive Proof System**) για μία γλώσσα L είναι ένα παιχνίδι μεταξύ ενός **Prover** και ενός **Verifier**. Το input είναι κοινό και επικοινωνούν βάσει ενός πρωτοκόλλου που ικανοποιεί τις παρακάτω προϋποθέσεις:

- **Efficiency**: Η στρατηγική του **Verifier** είναι ένας **Probabilistic Polynomial-Time** υπολογισμός.
- **Correctness** requirements:

  - **Completeness:** Υπάρχει στρατηγική για τον **Prover**, τέτοια ώστε για κάθε $x \in L$, όταν "παίζουν" με **κοινό input x**, να οδηγεί τον **Verifier** σε αποδοχή με πιθανότητα **τουλάχιστον 2/3**.

  - **Soundness:** Για κάθε $x \notin L$, όταν "παίζουν" με **κοινό input x**, κάθε στρατηγική για τον **Prover** πείθει τον **Verifier** με πιθανότητα **το πολύ 1/3.**

Παρατήρηση: ενώ περιορίζουμε τους υπολογισμούς του **Verifier** σε πολυωνυμικό χρόνο, ο **Prover** θεωρούμε ότι έχει απεριόριστη υπολογιστική δύναμη.

Η ιεραρχία IP

- Η κλάση **IP** περιέχει όλες τις γλώσσες που έχουν Διαλογικό Σύστημα Απόδειξης.
- Ο αριθμός των μηνυμάτων που ανταλλάσσονται μεταξύ του **Prover** και του **Verifier** κατά τη διάρκεια του πρωτοκόλλου ονομάζεται **αριθμός γύρων** (**number of rounds**) του συστήματος.
- Για κάθε συνάρτηση **r(.) : N -> N**, η κλάση **IP(r(.))** περιέχει όλες τις γλώσσες που έχουν Διαλογικό Σύστημα Απόδειξης, στο οποίο, για κοινό input **x**, πραγματοποιούνται το πολύ **r(|x|)** γύροι.
- Για μία οικογένεια συναρτήσεων R, ορίζουμε την κλάση $IP(R)=U_{r\in R}IP(r(.))$.

Παρατήρηση 1: Επειδή, εξ ορισμού, ο **Verifier** είναι περιορισμένος σε πολυωνυμικό χρόνο, **IP = IP(poly)**.

Παρατήρηση 2: Θα μπορούσαμε να θέσουμε περιορισμούς και στον αριθμό των **random bits** που έχει στη διάθεσή του ο **Verifier**, και να ορίσουμε κλάσεις **IP(b(.), r(.))**, όπου ο αριθμός των τυχαίων bits του **Verifier** για input x είναι μικρότερος από b(|x|). **IP(b(.)) = IP(b(.), poly)**

Σχόλια 1

- **NP ⊆ IP**
- Στον ορισμό των **IP** συστημάτων απαιτήσαμε την ύπαρξη ενός **Prover** που όταν $x \in L$ πείθει τον **Verifier** με πιθανότητα τουλάχιστον 2/3. Αν απαιτήσουμε **perfect completeness**, δηλαδή, όταν $x \in L$ ο **Prover** να μπορεί σε κάθε περίπτωση να πείσει τον **Verifier** (πιθανότητα αποδοχής ίση με 1), αποδεικνύεται ότι ο ορισμός είναι ισοδύναμος.

Θεώρημα: "Αν μία γλώσσα έχει Διαλογικό Σύστημα Απόδειξης, τότε έχει με **perfect completeness**."

- Όμοια μπορεί κανείς να απαιτήσει **perfect soundness**, δηλαδή, όταν $x \notin L$ ο **Prover** να μην μπορεί σε καμία περίπτωση να πείσει τον **Verifier** (πιθανότητα αποδοχής ίση με 0). Σε αυτή την περίπτωση το σύστημα ανάγεται στο **NP proof system**. Πράγματι, για $x \in L$ ο **Prover** μπορεί να βρει μία σειρά τυχαίων bits για τα οποία μπορεί να οδηγήσει τον **Verifier** σε αποδοχή (τέτοια bits υπάρχουν λόγω completeness). Αν ο **P** στείλει αυτά τα bits μαζί με τις δικές του απαντήσεις στον **V**, εκείνος μπορεί να ελέγξει σε πολυωνυμικό χρόνο την εγκυρότητά τους και αν επιπλέον οδηγούν σε αποδοχή, να αποδεχθεί.

Σχόλια 2

Κανείς ίσως αναρωτηθεί κατά πόσο τα **interaction** και **randomization** είναι απαραίτητα συστατικά ενός Διαλογικού Συστήματος Αποδείξεων

- Αν έχουμε μόνο **interaction**, δηλαδή αν ο **Verifier** είναι ντετερμινιστικός, τότε υπάρχει ένας **Prover** που γνωρίζει τις κινήσεις του **Verifier** και μπορεί να του στείλει τις δικές του απαντήσεις σε ένα μήνυμα. Άλλωστε, ένας ντετερμινιστικός **Verifier** πρέπει πάντα να απορρίπτει μία ψευδή είσοδο, πράγμα που σημαίνει **perfect soundness** και όπως είδαμε κάτι τέτοιο ανάγει το σύστημά μας στο **NP proof system.**
- Αν έχουμε μόνο **randomization**, τότε παίρνουμε την κλάση **IP[1]** και δεν είναι παρά μία πιθανοτική (και πιθανόν ισχυρότερη) εκδοχή του NP.

Graph isomorphism

- Δύο γράφοι $G_1=(V_1,E_1)$ και $G_2=(V_2,E_2)$ ονομάζονται **ισομορφικοί** όταν υπάρχει **1-1 και επί** συνάρτηση $\pi:V_1->V_2$ τέτοια ώστε:

$$(u,v) \in E_1 \text{ αν και μόνο αν } (\pi(u),\pi(v)) \in E_2$$

Η συνάρτηση π ονομάζεται ισομορφισμός μεταξύ των G_1 και G_2.

- Αν δεν υπάρχει καμία συνάρτηση με τις παραπάνω ιδιότητες, οι γράφοι ονομάζονται **μη-ισομορφικοί**.
- Ορίζουμε τη γλώσσα **GNI** ως εξής:

$$GNI = \{(G_1,G_2): G_1 \text{ και } G_2 \text{ είναι μη-ισομορφικοί}\}$$

- Το **GI (graph isomorphism)** είναι στο **NP**. Ενδιαφέρον έχει το γεγονός ότι δεν ξέρουμε αν είναι και **NP-hard**
- Το **GNI (graph non-isomorphism)** δεν γνωρίζουμε αν είναι στο **NP**. Θα δούμε όμως, ότι το **GNI** έχει Interactive Proof System και μάλιστα $GNI \in IP[2]$

Graph non-isomorphism is in IP[2]

Common Input: Δίνονται δύο γράφοι, G_1 και G_2

Verifier(1ο μήνυμα)

Επιλέγει **τυχαία** έναν γράφο.

Με **τυχαίο** τρόπο κατασκευάζει γράφο H ισομορφικό με αυτόν που επέλεξε.

Στέλνει τον H στον **Prover**

Prover (2ο μήνυμα)

Completeness: Αν οι δύο γράφοι είναι **μη-ισομορφικοί** μπορεί να βρει με ποιον από τους δύο είναι ισομορφικός ο γράφος H και έτσι να στείλει στον **Verifier** την σωστή απάντηση (με **βεβαιότητα**)

Soundness: Αν οι γράφοι είναι **ισομορφικοί**, μπορεί να μαντέψει τη σωστή απάντηση με **πιθανότητα** $\frac{1}{2}$. (το πλήθος των μεταθέσεων που οδηγούν από τον G_1 στον H είναι το ίδιο με το πλήθος αυτών που οδηγούν από τον G_2 στον H).

Ο V ελέγχει την απάντηση εύκολα, αφού **γνωρίζει τα random bits**

Graph non-isomorphism is in IP[2]

Το **κοινό input** είναι δύο γράφοι, έστω $G_1=(\{1,\ldots,n\},E_1)$ και $G_2=(\{1,\ldots,n\},E_2)$

Ο **Verifier**

- επιλέγει **τυχαία** $i \in \{1,2\}$
- με επίσης **τυχαίο** τρόπο υπολογίζει μετάθεση π του $\{1,\ldots,n\}$.
- εν συνεχεία, εφαρμόζοντας την π στον γράφο i υπολογίζει τον γράφο

$$H = (\{1,\ldots,n\}, \{(\pi(u),\pi(v)):(u,v)\in E_i\})$$

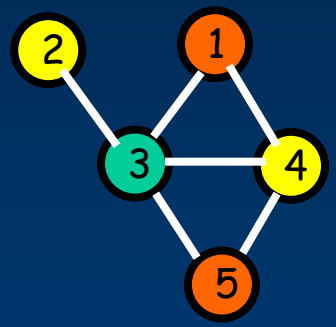
- στέλνει τον γράφο H στον **Prover**

Ο **Prover** στέλνει $j \in \{1,2\}$ στον **V**

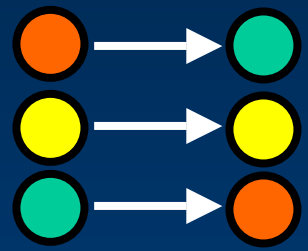
Ο **Verifier** αποδέχεται <u>αν και μόνο αν</u> $i=j$.

Graph 3 Coloring is in IP, ZK

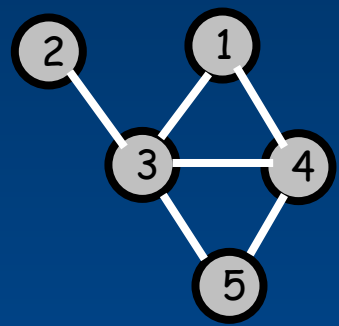
Ο **Prover** γνωρίζει χρωματισμό του γράφου με **3** χρώματα



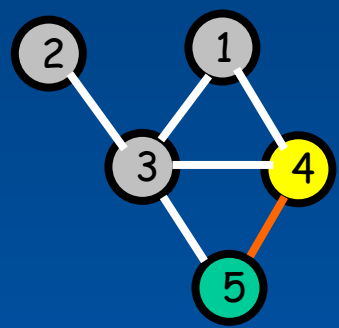
Επιλέγει **τυχαία μετάθεση** χρωμάτων



Κλειδώνει τους φρεσκοβαμμένους κόμβους σε αδιαφανή κουτιά και στέλνει το γράφο στον **Verifier**



Ο **Verifier** διαλέγει στην **τύχη** μία ακμή και ζητάει από τον **Prover** τα κλειδιά. Βλέπει τα χρώματα των κόμβων και αν είναι διαφορετικά αποδέχεται, ειδάλλως απορρίπτει.



Graph 3 Coloring is in IP, ZK

Completeness: Αν ο γράφος είναι **3-colorable**, ο **Prover** μπορεί πάντα να πείσει τον **Verifier**

Soundness: Αν ο γράφος δεν είναι **3-colorable**, τότε υπάρχει τουλάχιστον μία ακμή με το ίδιο χρώμα στα άκρα της. Ο **Verifier** μπορεί να διαλέξει αυτή την ακμή με **πιθανότητα 1/|E|.**

Zero Knowledge

Μπορεί ο **Verifier** μετά το τέλος του πρωτοκόλλου να υπολογίσει τον χρωματισμό του γράφου με 3 χρώματα;

Ο **Verifier** το μόνο που έβλεπε σε κάθε γύρο ήταν ένα **τυχαίο** ζευγάρι χρωμάτων. Αυτό όμως το κάνει και μόνος του!

Το πρωτόκολλο για το **Graph non-isomorphism** είχε αυτή την ιδιότητα;

Strong and weak definitions of IP

Definitions WITH Perfect Completeness

completeness: $x \in L \rightarrow$ Prob[Verifier Accepts] = 1

soundness: Για κάθε $x \notin L$, όταν "παίζουν" με **κοινό input x**, κάθε στρατηγική του **Prover** πείθει τον **Verifier** με πιθανότητα **το πολύ:**

- $2^{-p(|x|)}$
- ε, $\varepsilon \in (0,1)$
- $1 - 1/p(|x|)$

Definitions WITHOUT Perfect Completeness

completeness: ο **Verifier** αποδέχεται με πιθανότητα **τουλάχιστον** C

soundness: ο **Verifier** αποδέχεται με πιθανότητα **το πολύ** S

Τα παρακάτω ζεύγη (C,S) δίνουν ισοδύναμους ορισμούς:

- $(1-2^{-p(|x|)}, 2^{-p(|x|)})$
- $(2/3, 1/3)$
- $(\frac{1}{2} + \varepsilon, \frac{1}{2} - \varepsilon)$, με $\varepsilon < ½$
- $(p + \varepsilon, p - \varepsilon)$, με $p \in (0, 1)$ και $0 < \varepsilon < \min\{ p, 1 - p \}$
- $(p + 1/p(|x|), p - 1/p(|x|))$, με $p \in (0, 1)$

Πόσο δυνατός πρέπει να είναι ο Prover;

Στον ορισμό των IP συστημάτων αναφερόμαστε σε Prover με απεριόριστη δύναμη. Πόση δύναμη είναι πραγματικά απαραίτητη;

Γνωρίζοντας την στρατηγική του **Verifier**, μπορούμε να προσομοιώσουμε τον **Optimal Prover** σε **πολυωνυμικό χώρο.**

Θεώρημα: "Η βέλτιστη στρατηγική για τον **Prover** μπορεί να υπολογιστεί σε **πολυωνυμικό χώρο**."

Πόρισμα: **IP ⊆ PSPACE**

Τελικά, μπορούμε να προσομοιώσουμε κάθε **Interactive Proof** σε **πολυωνυμικό χώρο**

- Κάθε στιγμή ο **Optimal Prover** πρέπει να διαλέγει την στρατηγική που έχει την μεγαλύτερη πιθανότητα να καταλήξει σε αποδοχή.
- Για να το κατορθώσει αυτό, πρέπει να ανατρέξει σε όλους τους πιθανούς μελλοντικούς διαλόγους με τον **Verifier** και να επιλέξει την καλύτερη.
- Μπορεί να γίνει σε **πολυωνυμικό χώρο.**

Simulation of the Optimal Prover

- Έστω $F(\beta_1, \pi_1, \ldots, \pi_{i-1}, \beta_i)$ η πιθανότητα ένας διάλογος που ξεκινάει με $\beta_1, \pi_1, \ldots, \pi_{i-1}, \beta_i$ να οδηγήσει σε αποδοχή. Όπου β_i και π_i τα i - οστά μηνύματα που έστειλαν ο **Verifier** και ο **Prover** αντίστοιχα.
- Έστω **r** όλα τα τυχαία bits του **Verifier**.
- Έστω $R_{\beta 1, \pi 1, \ldots, \pi i-1, \beta i}$ το σύνολο όλων των **r** σε συμφωνία με τον διάλογο $\beta_1, \pi_1, \ldots, \pi_{i-1}, \beta_i$.
- Έστω $V(r, \pi_1, \ldots, \pi_{i-1})$ το μήνυμα β_{i+1} που έστειλε ο **Verifier**.
- Θα δείξουμε ότι η **F** μπορεί να υπολογιστεί χρησιμοποιώντας **πολυωνυμικό χώρο** και ότι για κάθε i, μπορεί να βρεθεί ένα π_i που μεγιστοποιεί την πιθανότητα **F**.

$$F(\beta_1, \pi_1, \ldots, \pi_{i-1}, \beta_i) = \max_{\pi_i} \frac{1}{|R_{\beta_1, \pi_1, \ldots, \beta_i}|} \sum_{r \in R_{\beta 1, \pi 1, \ldots, \beta i}} F(\beta_1, \pi_1, \ldots, \pi_i, V(r, \pi_1, \ldots, \pi_i))$$

Simulation is in PSPACE

- Για κάθε π_i:
  - Για κάθε δυνατή random sequence **r**:
    - ελέγχουμε αν $r \in R_{\beta 1, \pi 1, \ldots, \pi i-1, \beta i}$, προσομοιώνοντας τον **Verifier**
    - Υπολογίζουμε το $\beta_{i+1} = V(r, \pi_1, \ldots, \pi_i)$, επίσης προσομοιώνοντας τον **Verifier**
    - Υπολογίζουμε αναδρομικά το $F(\beta_1, \pi_1, \ldots, \pi_i, \beta_{i+1})$

Η αναδρομή μπορεί να ολοκληρωθεί σε πολυωνυμικό χώρο, διότι το βάθος της αναδρομής φράσσεται από το πλήθος των γύρων, που είναι πολυωνυμικό.

Επιπλέον, η πιθανότητα που μας γυρίζει κάθε αναδρομική κλήση, μπορεί να αποθηκευτεί με έναν αριθμητή και έναν παρονομαστή, με αναπαραστάσεις πολυωνυμικού μήκους.

$$F(\beta_1, \pi_1, \ldots, \pi_{i-1}, \beta_i) = \max_{\pi_i} \frac{1}{|R_{\beta_1, \pi_1, \ldots, \beta_i}|} \sum_{r \in R_{\beta_1, \pi_1, \ldots, \beta_i}} F(\beta_1, \pi_1, \ldots, \pi_i, V(r, \pi_1, \ldots, \pi_i))$$

IP ⊆ PSPACE

- Έστω **L in IP:**
- Τότε υπάρχει **IP** για την **L:**
- Μπορούμε να προσομοιώσουμε τον **Prover** αυτού του συστήματος σε **πολυωνυμικό χώρο**.
- Για κάθε δυνατό random sequence του **Verifier** προσομοιώνουμε το διάλογο μεταξύ **Prover** και **Verifier.** Εφικτό, σε **πολυωνυμικό χώρο**.
- Στο τέλος μετράμε πόσοι διάλογοι οδήγησαν σε αποδοχή και αν είναι το λιγότερο **2/3** αποδεχόμαστε.
- Τελικά, **αποδεχόμαστε** αν και μόνο αν **x ∈ L**
- Συμπέρασμα: **IP ⊆ PSPACE**

Combinatorial Games

Τα παρακάτω ορίζουν ένα **Combinatorial Game**:

- Έστω $D_1, D_2, \ldots, D_t$ μη κενά, πεπερασμένα σύνολα
- Έστω συνάρτηση f ορισμένη στο καρτεσιανό γινόμενο $dom(f)=D_1 x \ldots x D_t$ και με $ran(f)=\{0,1\}$ (pay off function)
- Δύο παίχτες **A** (Arthur) και **M** (Merlin) **παίζουν εναλλάξ**. Στην **i-οστή** κίνηση ο παίκτης που έχει σειρά επιλέγει $x_i \in D_i$
- Το παιχνίδι **τελειώνει** μετά την **t-οστή** κίνηση
- Ο παίχτης **M κερδίζει** αν $f(x_i,\ldots,x_t)=1$
- **history** = $(x_i,\ldots,x_t)$
- **game space** = dom(f)
- **game size** = $\log_2|dom(f)|$
- Το παιχνίδι ορίζεται από ένα ζεύγος (f,Q), όπου παίζει πρώτος ο παίκτης Q (Q=A ή Q=M)

Arthur-Merlin Games

Τα **Arthur-Merlin Games** είναι **combinatorial games** που επιπλέον ισχύουν τα παρακάτω:

- Οι κινήσεις του **A** είναι **τυχαίες** (δεν ενδιαφέρεται για νίκη)
- Οι κινήσεις του **M** είναι **βέλτιστες** (έχει απεριόριστη δύναμη)
- Για κάθε input x ισχύει ένα από τα παρακάτω:

  (α) **W(x) > 2/3**, ή

  (β) **W(x) < 1/3**

όπου W(x) η πιθανότητα του M να νικήσει.

- Η **γλώσσα** που αναγνωρίζει ένα τέτοιο παιχνίδι αποτελείται από τις συμβολοσειρές x για τις οποίες ισχύει το (**α**).

Ιεραρχία AM

AM(t(n)) είναι το σύνολο των γλωσσών που αναγνωρίζονται από παιχνίδια Arthur-Merlin μήκους **t(n)**, όπου ο A παίζει πρώτος. Ομοίως ορίζουμε **MA(t(n))**, **AM(Poly)=MA(Poly)=U{AM(n^k:k>0)}**, **AM(3)=AMA**, **MA(1)=M** κ.λπ.

["Trading Group Theory for Randomness", 1985, Laszlo Babai (Budapest, Chicago)]

Συμβολισμοί

- Έστω συνάρτηση f που παίρνει πραγματικές τιμές στο D=dom(f):

$$\textbf{average operator :} \quad Ax\, f(x) = \sum_{x \in D} \frac{f(x)}{|D|}$$

$$\textbf{maximum operator :} \quad Mx\, f(x) = \max\{ f(x) \mid x \in D \}$$

Σε ένα παιχνίδι Arthur-Merlin **(f,M)**, η πιθανότητα να κερδίσει ο **M** είναι:

$$Mx_1 Ax_2 Mx_3 \ldots Qx_t\ f(x_1, x_2, \ldots, x_t)$$

και αν παίζει ο **A** πρώτος:

$$Ax_1 Mx_2 Ax_3 \ldots Qx_t\ f(x_1, x_2, \ldots, x_t)$$

- Έστω **p** η πιθανότητα να κερδίσει ο **M** σε ένα παιχνίδι Arthur-Merlin. Ορίζουμε την **αβεβαιότητα** του παιχνιδιού (**uncertainty** of the game) ως εξής:

$$unc(f,Q) = \min\{p, 1-p\}$$

(η πιθανότητα να χάσει ο παίκτης με το πλεονέκτημα)

Απαιτήσεις κατά την προσομοίωση

Σκοπός είναι η προσομοίωση δεδομένου παιχνιδιού (f,Q) με ένα άλλο (f*,Q*) λιγότερων κινήσεων, ώστε να ικανοποιούνται τα εξής:

(α) Και στα δύο παιχνίδια έχει το **πλεονέκτημα** ο ίδιος παίχτης

(β) Ο χώρος του παιχνιδιού (**game space**) δεν αυξάνει σημαντικά

(γ) Η **αβεβαιότητα** αυξάνεται ή δεν μειώνεται σημαντικά

(δ) Η **f*** πρέπει να υπολογίζεται εύκολα από την **f**

Τηρώντας τις παραπάνω απαιτήσεις μπορούμε να προσομοιώσουμε κάθε παιχνίδι AM[κ], όπου κ σταθερά κ>=2, με ένα παιχνίδι AM. Η ιεραρχία AM καταρρέει στο 2ο επίπεδο. Συγκεκριμένα:

AM = AM[κ] = MA[κ+1], για κάθε σταθερά κ>=2

Ενίσχυση του πλεονεκτήματος

Σε συγκεκριμένο παιχνίδι **Arthur-Merlin (f,Q)** ο ένας παίκτης έχει το πλεονέκτημα (αν $x \in L$ ο Merlin). Μπορούμε να αυξήσουμε αυτό το πλεονέκτημα βάζοντας τους παίκτες να παίξουν το παιχνίδι εν παραλλήλω σε **k** σκακιέρες.

Πιο τυπικά ορίζουμε (f^k, Q) ως εξής:

$dom(f) = D_{1k} \times \ldots \times D_{tk}$ και

$f^k(x_{11}, \ldots, x_{tk})$ είναι η πιθανότητα να κερδίσει ο Μ **περισσότερα από τα μισά παιχνίδια**.

Αποδεικνύεται ότι αν η αβεβαιότητα στο πρώτο παιχνίδι ήταν μικρότερη από **1/3** τότε στο δεύτερο είναι μικρότερη από **$(8/9)^{k/2}$**.

Εναλλαγή παικτών

Μπορούμε να προσομοιώσουμε κάθε ΜΑ παιχνίδι με ένα ΑΜ

Το πρόβλημα που πρέπει να ξεπεράσουμε για αυτήν την προσομοίωση, είναι ότι δίνοντας στον Α την πρώτη κίνηση, ο Μ έχει τη δύναμη να ανατρέψει το πλεονέκτημα υπέρ του.

Ξεπερνάμε αυτή την δυσκολία βάζοντας τον Α να ξεκινήσει με m ανεξάρτητες κινήσεις και αναγκάζοντας τον Μ να κερδίσει τις περισσότερες από αυτές με μία μόνο κίνηση.

- Έστω (f, M) με $dom(f) = X \times Y$ ένα ΜΑ παιχνίδι
- Μπορούμε να το προσομοιώσουμε με ένα παιχνίδι (f^*, A) όπου ο Α παίζει πρώτος με $dom(f) = Y^m \times X$.

Μείωση των γύρων κατά 1

AM[κ] = AM[κ+1]

Αποδεικνύεται ότι κάθε **(g,A)** παιχνίδι **k+1** γύρων μπορούμε να το προσομοιώσουμε με ένα παιχνίδι **(G,A) k** γύρων.

Ο τρόπος που γίνεται αυτό είναι ίδιος με τον τρόπο που εναλλάξαμε προηγουμένως τις κινήσεις στο MA παιχνίδι.

Πάλι το παιχνίδι γίνεται σε m παράλληλες σκακιέρες:

$dom(G) = (D_1 \times D_3^m) \times D_2 \times \ldots \times \prod D_i^m$

Οι παρενθέσεις δείχνουν ότι μετά την εναλλαγή των κινήσεων συμπτύσσουμε τις δύο συνεχόμενες κινήσεις του A σε μία δημιουργώντας έτσι παιχνίδι με μία κίνηση λιγότερο.

Ο **M** πρέπει να κερδίσει τα **περισσότερα** από τα **m** παιχνίδια:

$w_i=(x_1, x_2, x_{3i}, x_{4i}, \ldots x_{ti})$

AM[2] = AM[κ] = MA[κ+1]

A M A M A M A M A M A M A... Q
A A M M A M A M A M A M A... Q
A M A M A M A M A M A M... Q

Επαναλαμβάνοντας για σταθερό αριθμό φορών τα παραπάνω επιτυγχάνουμε το ζητούμενο.

Δεν μπορούμε να κάνουμε το ίδιο για απεριόριστο αριθμό γύρων γιατί το μέγεθος του παιχνιδιού τετραγωνίζεται κάθε φορά που κάνουμε την εναλλαγή.

Private Coins vs. Public Coins

Arthur-Merlin Games & Interactive Proofs

Ομοιότητες:

- διαλογικά
- πιθανοτικά
- ο ένας παίχτης έχει απεριόριστη δύναμη (Merlin, Prover)
- ο άλλος είναι περιορισμένων δυνατοτήτων (Arthur, Verifier)

Βασική Διαφορά:

- Στα Arthur-Merlin Games οι τυχαίες επιλογές του Arthur είναι **γνωστές** στον Merlin, ενώ αντίθετα στα Interactive Proofs οι τυχαίες επιλογές του Verifier γίνονται **κρυφά** από τον Prover. Συγκεκριμένα, στα Arthur-Merlin Games ο Arthur δεν κάνει τίποτε άλλο από το να στέλνει τα random bits του στον Merlin.

Συνεπώς, για κάθε $f=O(poly)$ ισχύει $AM(f) \subseteq IP(f)$

IP = AM ως προς Language Recognition

Για κάθε πολυώνυμο Q ισχύει IP[Q] ⊆ AM[Q+2]

Γενική ιδέα απόδειξης:

Θέλουμε να προσομοιώσουμε τυχαίο IP σύστημα με ένα AM παιχνίδι. Η ιδέα στην απόδειξη αυτού του θεωρήματος είναι να στείλει ο A τα random bits του στον M, αλλά με τέτοια μορφή ώστε να τον αναγκάσει να εκτελέσει για λογαριασμό του τις πράξεις που θα εκτελούσε ο Verifier του IP συστήματος. Στη συνέχεια ο M στέλνει αυτούς τους υπολογισμούς μαζί με την απάντησή του και ο A ελέγχει την εγκυρότητά τους.

["Private Coins vs. Public Coins in Interactive Proof Systems", 1986, Shafi Goldwasser (MIT), Michael Sipser (MIT)]

Quantified Boolean Formulas

- Έχουμε ήδη δείξει ότι **IP** $\subseteq$ **PSPACE**
- Για να δείξουμε ότι και **PSPACE** $\subseteq$ **IP**, αρκεί να δείξουμε ότι ένα **PSPACE-Complete** πρόβλημα έχει **Interactive Proof System**

Quantified Boolean Formulas

- Το σύνολο των **Quantified Boolean Formulas** ορίζεται ως το **κλείσιμο** (**closure**) του συνόλου των boolean variables x_i και των αρνήσεών τους $\neg x_i$ υπό τις πράξεις $\wedge$ (and), $\vee$ (or), $\forall x_i$ (universal quantification) και $\exists x_i$ (existential quantification)
- Μία **QBF** ονομάζεται **κλειστή** (**closed**) όταν όλες οι μεταβλητές της είναι bounded από κάποιον ποσοδείκτη. Οι κλειστές QBF μπορεί να είναι είτε **True**, είτε **False**.
- Μία **κλειστή QBF** ονομάζεται **απλή** (**simple**) όταν μεσολαβεί το πολύ ένας καθολικός ποσοδείκτης μεταξύ κάθε εμφάνισης μίας μεταβλητής και του σημείου quantification της

π.χ. $\forall x_1 \forall x_2 \exists x_3 [(x_1 \vee x_2) \wedge \forall x_4 [(x_2 \wedge x_3 \wedge x_4)]$ είναι απλή

$\forall x_1 \forall x_2 [(x_1 \wedge x_2) \; \forall x_3 [(\neg x_1 \wedge x_3)]$ δεν είναι απλή

Simple QBF's

- Το **QBF** είναι **PSPACE-Complete**.

Θεώρημα 1: Κάθε **QBF** μεγέθους **n** μπορεί να μετατραπεί σε **ισοδύναμη simple QBF** με μέγεθος **πολυωνυμικό ως προς n**.

Παράδειγμα: Παρακάτω η έκφραση **Q** δεν περιέχει ποσοδείκτες

QBF: $\exists x \forall y \exists z \forall s \exists t\ Q(x,y,z,s,t)$

simple QBF: $\exists x_0 \forall y_0 \exists x_1 (x_1 = x_0) \wedge \exists z_0 \forall s_0 \exists x_2 \exists y_1 \exists z_1$

$(x_2 = x_1) \wedge (y_1 = y_0) \wedge (z_1 = z_0) \wedge \exists t_0\ Q(x_2, y_1, z_1, s_0, t_0)$

Με την παραπάνω διαδικασία η μετασχηματισμένη QBF είναι προφανώς **simple**. Επιπλέον, αν **n** ήταν το μέγεθος της πρώτης QBF, τότε το μέγεθος της simple QBF θα είναι $O(n^2)$.

- Έστω η γλώσσα **L = {x : (x is simple QBF) ∧ (x is true)}**

Λόγω των παραπάνω, αν κατασκευάσουμε **Interactive Proof System** για την γλώσσα **L** θα έχουμε αποδείξει ότι: **IP=PSPACE**

["IP=PSPACE", 1992, Adi Shamir (Israel)]

Arithmetization of QBF

Με κάθε **closed QBF B** αντιστοιχίζουμε μία **αριθμητική έκφραση A**, κάνοντας τους παρακάτω συντακτικούς μετασχηματισμούς:

$x_i \longrightarrow z_i \ , \ z_i \in Z$

$\neg x_i \longrightarrow 1 - z_i$

$\wedge \longrightarrow$ * , integer multiplication

$\vee \longrightarrow$ + , integer addition

$\forall x_i \longrightarrow \Pi_{z_i \in \{0,1\}}$

$\exists x_i \longrightarrow \Sigma_{z_i \in \{0,1\}}$

$x_i \approx x_j \longrightarrow (x_i \wedge x_j) \vee (\neg x_i \wedge \neg x_j)$

Arithmetization of QBF: Παράδειγμα

Έστω για παράδειγμα η παρακάτω **TRUE QBF**:

$$B = \forall x_1 \exists x_2 [(x_1 \wedge x_2) \vee \exists x_3 (\neg x_2 \wedge x_1)]$$

Η αριθμητική της έκφραση είναι:

$$A = \prod_{z_1 \in \{0,1\}} \sum_{z_2 \in \{0,1\}} [(z_1 \cdot z_2) + \sum_{z_3 \in \{0,1\}} (1 - z_2) \cdot z_3]$$

και η τιμή της είναι **2**.

Θεώρημα 2: Μια **closed QBF B** είναι **TRUE** αν και μόνο αν η τιμή της **αριθμητικής έκφρασής της A**, είναι **μη-μηδενική**. (απόδειξη με επαγωγή στη δομή της B)

Το μέγεθος της αριθμητικής έκφρασης

Πόσο μεγάλη μπορεί να γίνει η τιμή της αριθμητικής έκφρασης μίας closed QBF;

Θεώρημα 3: Έστω **B** μία **closed QBF** μεγέθους **n**. Η τιμή της αριθμητικής της έκφρασης **A** είναι $O(2^{2^n})$.

Απόδειξη: Για κάθε sub-expression **B'** της **B** ας είναι **u(B')** η μέγιστη τιμή που μπορεί να πάρει η αριθμητική έκφραση της **B'** αν αντικατασταθούν οι ελεύθερες μεταβλητές με 0 και 1.

(1) Αν η **B'** είναι x_i ή $\neg x_i$, τότε **u(B') = 1**

(2) Αν η **B'** είναι $B'' \vee B'''$, τότε $u(B') <= u(B'') + u(B''')$

(3) Αν η **B'** είναι $B'' \wedge B'''$, τότε $u(B') <= u(B'') \cdot u(B''')$

(4) Αν η **B'** είναι $\exists x_i B''$, τότε $u(B') <= 2 \cdot u(B''')$

(5) Αν η **B'** είναι $\forall x_i B''$, τότε $u(B') <= [u(B''')]^2$

Μπορεί και τόσο πολύ: π.χ. $B = \forall x_1 \forall x_2 \cdots \forall x_{n-1} \exists x_n (\neg x_n \wedge x_n)$

Είναι $u(\exists x_n (\neg x_n \wedge x_n)) = 2$ και τη διπλασιάζει κάθε $\forall$

mod p

Παρατήρηση: Η τιμή μίας αριθμητικής έκφρασης μπορεί να γίνει πολύ μεγάλη. Αυτό μας ενοχλεί διότι ο **Verifier** δεν μπορεί να χειριστεί αριθμούς $O(2^{2^n})$. Για να λειτουργήσει το πρωτόκολλο θα πρέπει να βρεθεί πρώτος **p** κατάλληλος ώστε να γίνουν οι πράξεις **modulo p**.

Θεώρημα 4: Έστω **B** μία **closed QBF** μεγέθους **n**. Υπάρχει πρώτος **p** πολυωνυμικού μήκους ως προς **n** τέτοιος ώστε $A \neq 0 (\text{mod} p)$ <u>αν και μόνο αν</u> η **B** είναι **TRUE**

Απόδειξη:

(**if**) Έστω ότι ο **A** είναι **μη-μηδενικός** ακέραιος. Αν είναι **0** modulo κάθε πολυωνυμικού μήκους πρώτο, τότε είναι **0** modulo το γινόμενο όλων αυτών. Από το θεώρημα των πρώτων ($\pi(n) \sim n/\ln n$)

αυτό το γινόμενο είναι $\Omega(2^{2^{nd}})$ για κάθε σταθερά **d**. Άτοπο, γιατί το **A** είναι μη-μηδενικό και $O(2^{2^n})$.

Functional Form

Ορίζουμε ως **Functional Form** A' μίας αριθμητικής έκφρασης A, το πολυώνυμο μίας μεταβλητής που προκύπτει από την αφαίρεση του αριστερότερου $\Pi_{z_i \in \{0,1\}}$ ή $\Sigma_{z_i \in \{0,1\}}$ από την A.

Παράδειγμα:

$$B = \forall x_1 [\neg x_1 \vee \exists x_2 \forall x_3 (x_1 \wedge x_2) \vee x_3]$$

$$A = \prod_{z_1 \in \{0,1\}} [(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2 + z_3)]$$

$$A' = [(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2 + z_3)]$$

Ο **Prover** μπορεί να εκφράσει το A' ως πολυώνυμο. Σε αυτή την περίπτωση είναι $q(z_1) = z_1^2 + 1$

Polynomials of Functional Forms

Θεώρημα 5: Αν η **B** είναι **simple** και μεγέθους **n**, τότε ο βαθμός του πολυωνύμου $q(z_1)$ που προκύπτει από τη **Functional Form** της **A** είναι **O(n)**.

Απόδειξη:

- Σε subexpressions χωρίς ποσοδείκτες, ο βαθμός του z_1 είναι μικρότερος από το μήκος της subexpression
- Τα αθροίσματα δεν αλλάζουν το βαθμό του πολυωνύμου
- Τα γινόμενα διπλασιάζουν το βαθμό, αλλά επειδή η **B** είναι **simple** τέτοιος διπλασιασμός μπορεί να συμβεί μόνο μία φορά

Παρατήρηση: Προσθέτοντας μεταβλητές μπορούμε να μειώσουμε το βαθμό του πολυωνύμου στο **3** και έτσι να αρκούν **4** αριθμοί για να το περιγράψουμε. Π.χ. η $B = \forall x B'$, όπου η **x** εμφανίζεται **k** φορές στη **B'**, μπορεί να γραφεί:

$$\forall x_1 \exists x_2 \cdots \exists x_k \, (x_1 \approx x_2 \approx \ldots \approx x_k) \wedge B''$$

Παράδειγμα

common input:

$$B = \forall x_1 \left[\neg x_1 \vee \exists x_2 \forall x_3 (x_1 \wedge x_2) \vee x_3 \right]$$

$$A = \prod_{z_1 \in \{0,1\}} \left[(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2 + z_3) \right]$$

$$A' = \left[(1 - z_1) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (z_1 \cdot z_2 + z_3) \right]$$

Στέλνει ο P στον V στο 1ο βήμα:

$$q(z_1) = z_1^2 + 1$$

$$a = q(0) \cdot q(1) = 2$$

Verifier:

- ελέγχει αν $a = q(0) \cdot q(1) = 2$
- στέλνει τυχαίο $z_1 = 3$ στον Prover (2ο βήμα)

Παράδειγμα

Υπολογισμοί που κάνουν και οι δύο:

$$A'(z_1 = 3) = (1-3) + \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (3z_2 + z_3) = A_1 + A_2$$

$$A = \sum_{z_2 \in \{0,1\}} \prod_{z_3 \in \{0,1\}} (3z_2 + z_3)$$

$$a = q(z_1 = 3) - a_1 = 10 - (-2) = 12$$

$$A' = \prod_{z_3 \in \{0,1\}} (3z_2 + z_3)$$

Στέλνει ο P στον V στο 3ο βήμα: $q(z_2) = 9z_2^2 + 3z_2$

και πάλι από την αρχή...

Verifier:

- ελέγχει αν $a = q(0) + q(1) = 9 + 3 = 12$
- στέλνει τυχαίο $z_1 = 2$ στον Prover (4ο βήμα)

Παράδειγμα

Υπολογισμοί που κάνουν και οι δύο:

$$A'(z_2 = 2) = \prod_{z_3 \in \{0,1\}} (6 + z_3) = A_1 + A_2$$

$$A = \prod_{z_3 \in \{0,1\}} (6 + z_3)$$

$$a = q(z_2 = 2) - a_1 = 9 \cdot 4 + 3 \cdot 2 = 42$$

$$A' = 6 + z_3$$

Στέλνει ο P στον V στο 5ο βήμα: $q(z_3) = z_3 + 6$

Verifier:

- ελέγχει αν $a = q(0) \cdot q(1) = (6+0)(6+1) = 42$
- επιλέγει τυχαίο $z_3 = 5$ και καθώς στο νέο A το A_2 είναι κενό, ελέγχει μόνος του εάν το $A'(z_3 = 5) = (6 + 5) = 11$ είναι ίσο με $a = q(z_3 = 5) = 5 + 6 = 11$

Το πρωτόκολλο

common input: αριθμητική παράσταση A μίας **simple QBF**

Ο **Prover** στέλνει στον **Verifier** την τιμή $a = A \bmod p$ (όπου p κατάλληλος πρώτος) και θέλει να τον πείσει ότι $A \neq 0 (\bmod p)$. Αυτό το κατορθώνει μειώνοντας σε κάθε βήμα το μέγεθος της A. Η A σε κάθε ενδιάμεσο βήμα διαιρείται σε A_1+A_2 ή $A_1 \cdot A_2$. Η A_1 είναι αριθμητική παράσταση υπολογίσιμη από τον **Verifier** και με τιμή a_1, ενώ η A_2 είναι μία κλειστή αριθμητική έκφραση και αρχίζει με τον αριστερότερο ποσοδείκτη της A.

εν συνεχεία επαναλαμβάνουν τα παρακάτω:

- Αν η A_2 είναι κενή, ο **Verifier** αποδέχεται αν και μόνο αν $a = a_1$
- Αν η A_1 δεν είναι κενή, αντικαθίσταται η A από την A_2 και το a από $a - a_1 \bmod p$ ή $a / (a_1 \bmod p)$. Αν $a_1 = 0 \pmod p$ ο **Verifier** σταματάει και αποδέχεται αν και μόνο αν $a = 0 \pmod p$.
- Διαφορετικά, ο **Prover** στέλνει το πολυώνυμο $q(z_i)$ στον **Verifier** και αυτός ελέγχει αν $a = q(0) + q(1)$ ή $a = q(0).q(1)$. Στέλνει τυχαίο $r \in Z_p$ στον **Prover** και αντικαθίστανται τα A και a από $A'(z=r) \pmod p$ και $q(r) \pmod p$ αντιστοίχως.

Completeness & Soundness

completeness: Όταν η τιμή της QBF είναι TRUE, ο **Prover** δεν έχει παρά να ακολουθήσει το πρωτόκολλο και εν τέλει θα πείσει τον **Verifier**.

soundness: Αν ο **Prover** δώσει ψεύτικη τιμή **α** στον **Verifier**, θα πρέπει να δώσει και ψεύτικο πολυώνυμο **q**, ώστε να περάσει τους ενδιάμεσους ελέγχους του **Verifier** (**α=q(0)+q(1)** ή **α=q(0).q(1)**).

Όμως, όταν ο **Verifier** εκτιμήσει για τυχαίο $r \in Z_p$ την τελική αριθμητική έκφραση **A'** μόνος του, για να παραπλανηθεί θα πρέπει το πολυώνυμο του **Prover** να συμπίπτει με την **A'** στο τυχαίο **r**. Τα **A'** και **q** είναι πολυώνυμα βαθμού το πολύ **3** και επομένως συμπίπτουν σε **3** το πολύ σημεία (εκτός αν ταυτίζονται). Το **r** αντιθέτως μπορεί να πάρει οποιαδήποτε τιμή στο **Zp** και αφού η τιμή του **p** είναι εκθετική ως προς την είσοδο η πιθανότητα λάθους του πρωτοκόλλου είναι εκθετικά μικρή.

παρατήρηση 1: το πρωτόκολλο είναι **Public Coin**
παρατήρηση 2: το πρωτόκολλο που περιγράφηκε δεν χρειάζεται να επαναληφθεί, είναι όμως από μόνο του πολυωνυμικού μήκους.