Jean Goubault-Larrecq

Randomized complexity classes

Today: Shamir's theorem

Today

- The classes **ABPP**, **IP**
- Easy: **ABPP** $\subseteq$ **IP** $\subseteq$ **PSPACE**
- Hard (Shamir's theorem): **ABPP** = **IP** = **PSPACE**

$\mathsf{ABPP} \subseteq \mathsf{IP} \subseteq \mathsf{PSPACE}$

ABPP, IP

- **ABPP** $\stackrel{\text{def}}{=}$ **AM**[poly] = {languages recognizable by an A-M protocol with **polynomially many** rounds}
- **IP** $\stackrel{\text{def}}{=}$ **IP**[poly] = {languages recognizable by an interactive proof with **polynomially many** rounds}
- **Beware**: Merlin must provide answers y of size polynomial in $n \stackrel{\text{def}}{=} \text{size}(x)$, **not** in the size of the history

The subtlety with answer sizes

- Imagine Merlin were allowed to answer y of size $|\text{history}|^2$ (and Arthur is lazy, and $|r|=n$, to make things simpler)
- $|x\#q_1\#r_1| = 2n+2$
- $|x\#q_1\#r_1\#y_1| = (2n+2)+1+(2n+2)^2 = 4n^2+6n+7 \geq 4n^2$
- $|x\#q_1\#r_1\#y_1\#q_2\#r_2\#y_2| \geq (4n^2)^2 = 16n^4$
- …
- $|x\#q_1\#r_1\#y_1\#\ldots\#q_k\#r_k\#y_k| \geq 2^{2\text{^}k}n^{2\text{^}k}$
- polynomial if k constant,
  doubly exponential if k=poly(n)

The subtlety with answer sizes

- Instead, Merlin must answer y of size $\leq q(n)$ [q **polynomial**]
  Arthur also runs $\mathcal{A}(x\#q_1\#r_1\#y_1\ldots,r)$ in time $\leq q(n)$
  hence uses up $\leq q(n)$ random bits, produces question of size $\leq q(n)$
- $|x\#q_1\#r_1| \leq n+2q(n)+2$
- $|x\#q_1\#r_1\#y_1| \leq n+3q(n)+3$
- $|x\#q_1\#r_1\#y_1\#q_2\#r_2\#y_2| \leq n+6q(n)+6$
- …
- $|x\#q_1\#r_1\#y_1\#\ldots\#q_k\#r_k\#y_k| \leq n+3k\,q(n)+3k$
- **polynomial** if k=poly(n)

ABPP ⊆ PSPACE

- We start with the relatively simple inclusion **ABPP ⊆ PSPACE**
- Let $L \in$ **ABPP**, decided in $R(n)$ rounds, random tape size $=q(n)$, lazy Arthur
- Idea: **count** the number of lists of random strings $r_1, r_2, \ldots, r_{R(n)}$ that lead to acceptance
- That must be $\geq 2/3.2^{R(n)q(n)}$ or $\leq 1/3.2^{R(n)q(n)}$:
  accept if larger than $1/2.2^{R(n)q(n)}$, reject otherwise
- Answers by Merlin are **guessed**.
- Hence L is in **NPSPACE**, therefore in **PSPACE** (Savitch).
  See lecture notes for details.

ABPP ⊆ PSPACE: alternate argument

- Let $L \in \mathbf{ABPP}$, defined by formula
  $\mathrm{E}r_1, \exists y_1, \mathrm{E}r_2, \exists y_2, \ldots, \mathrm{E}r_k, \exists y_k, P(x,r_1,y_1,\ldots,r_k,y_k)$ $[k=R(n)]$
  namely this is ≥⅔ if $x \in L$, ≤⅓ if $x \notin L$

- Hence
  $F(x) \stackrel{\text{def}}{=} \Sigma r_1, \max y_1, \Sigma r_2, \max y_2, \ldots, \Sigma r_k, \max y_k, P(x,r_1,y_1,\ldots,r_k,y_k)$
  is $\geq ⅔.2^{R(n)q(n)}$ if $x \in L$, $\leq ⅓.2^{R(n)q(n)}$ if $x \notin L$

- We accept if $F(x) \geq ½.2^{R(n)q(n)}$, we reject otherwise

- Note that we can compute $F(x)$ in poly space:
  — $2R(n)$ words r_i, y_i, of size $\leq q(n)$
  — $P(x,r_1,y_1,\ldots,r_k,y_k)$ poly time, hence poly space
  — Intermediate counters $\leq 2^{R(n)q(n)}$, hence of size $\leq R(n)q(n)$.

IP ⊆ PSPACE

- Let now $L \in \mathbf{IP}$, decided in $R(n)$ rounds, random tape size $=q(n)$
  Arthur no longer lazy: $q_i \stackrel{\text{def}}{=} \mathcal{A}(x\#q_1\#r_1\#y_1\#\ldots\#y_{i-1},r_i)$, size $\leq q(n)$
- If we **count** the number of lists of random strings $r_1, r_2, \ldots, r_{R(n)}$
  that lead to acceptance, and Merlin guesses y_i,
  then y_i may **depend on** $r_1, r_2, \ldots, r_i$
  — but it is only allowed to depend on (x and) $q_1, q_2, \ldots, q_i$
- Instead, we count the # of lists of **random questions** $q_1, q_2, \ldots, q_{R(n)}$
  — it is just that they are not **uniformly** random;
  we weigh each of them with the number of random strings that give rise to those questions: see lecture notes for details

IP ⊆ PSPACE: alternate argument

- Let $L \in \mathbf{IP}$, similarly as for **AM**, we can show that L is defined by a formula
  $E' q_1, \exists y_1, E' r_2, \exists y_2, \ldots, E' q_k, \exists y_k, \Pr_{r_1,\ldots,r_k}(P(x, r_1, y_1, \ldots, r_k, y_k) = 1) \quad [k = R(n)]$
  where $E' q_i$ is average over questions q_i,
  with probability card $\{r_i \mid \mathcal{A}(x\#q_1\#r_1\#y_1\#\ldots\#y_{i-1}, r_i) = q_i\} / 2^{q(n)}$
- This formula is $\geq 2/3$ if $x \in L$, $\leq 1/3$ if $x \notin L$
- Hence
  $F(x) \stackrel{\text{def}}{=} \Sigma q_1, \max y_1, \Sigma q_2, \max y_2, \ldots, \Sigma q_k, \max y_k, (\Sigma r_1, \ldots, r_k, P(x, q_1, r_1, y_1, \ldots, q_k, r_k, y_k))$
  (where the final sum ranges over random strings r_i yielding the correct questions q_i)
  is $\geq 2/3 . 2^{R(n) q(n)}$ if $x \in L$, $\leq 1/3 . 2^{R(n) q(n)}$ if $x \notin L$ [$q(n) \stackrel{\text{def}}{=}$ question size, now]
- We accept if $F(x) \geq 1/2 . 2^{R(n) q(n)}$, we reject otherwise
- Note that we can compute $F(x)$ in poly space, as previously.

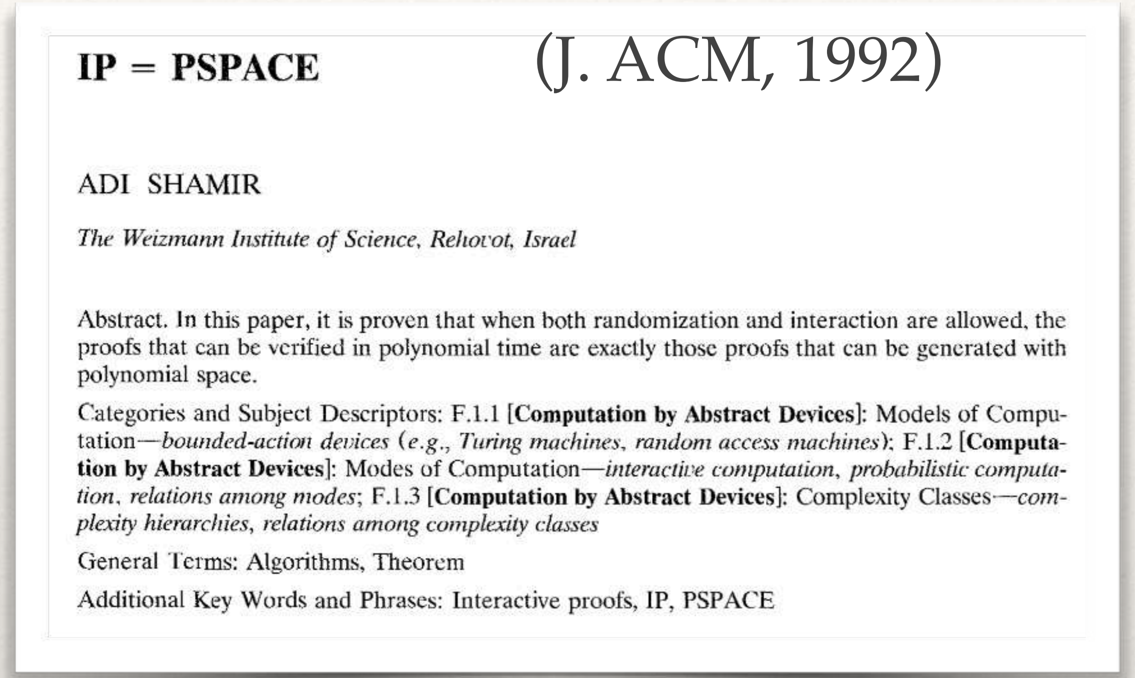
The easy direction

- **Prop. ABPP ⊆ IP ⊆ PSPACE**
- We have just sketched proofs of **IP** ⊆ **PSPACE**
- **ABPP** ⊆ **IP** is because **AM**[$f(n)$] ⊆ **IP**[$f(n)$] for any f:
  given $L \in$ **AM**[$f(n)$] decided by a lazy Arthur,
  an **IP**[$f(n)$] protocol for f computes $q_i \stackrel{\text{def}}{=} \mathcal{A}(x\#q_1\#r_1\#y_1\#\ldots\#y_{i-1},r_i)$
  as r_i, simply. □

The hard direction:
PSPACE $\subseteq$ ABPP

Shamir's theorem

(J. ACM, 1992)


IP = PSPACE

ADI SHAMIR

The Weizmann Institute of Science, Rehovot, Israel

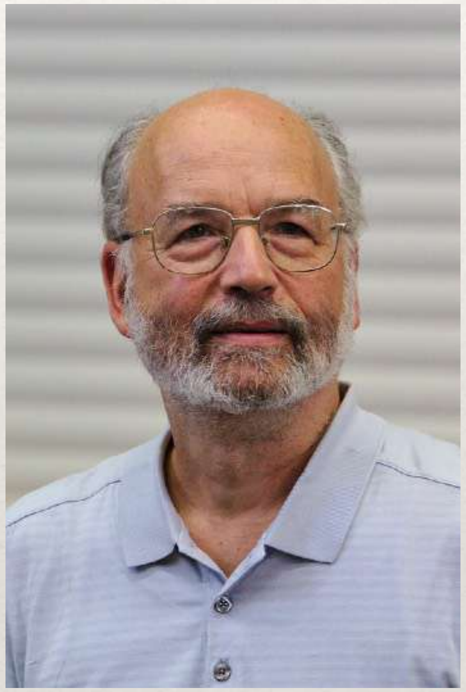
Abstract. In this paper, it is proven that when both randomization and interaction are allowed, the proofs that can be verified in polynomial time are exactly those proofs that can be generated with polynomial space.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*bounded-action devices* (*e.g., Turing machines, random access machines*); F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*interactive computation, probabilistic computation, relations among modes*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes—*complexity hierarchies, relations among complexity classes*

General Terms: Algorithms, Theorem

Additional Key Words and Phrases: Interactive proofs, IP, PSPACE

Adi Shamir



Par Erik Tews — Travail personnel, CC BY-SA 3.0,
`https://commons.wikimedia.org/w/index.php?curid=28572036`

Shamir shows **PSPACE ⊆ ABPP**,
which entails **IP=PSPACE**

Building on a series of previous ideas by
Lund, Feige, and others

Alexander Shen

I will really describe A. Shen's simplified proof

Александр Ханиевич Шень

IP = PSPACE: Simplified Proof

(J. ACM, 1992)

A. SHEN

Academy of Sciences, Moscow, Russia, CIS

Abstract. Lund et al. [1] have proved that PH is contained in IP. Shamir [2] improved this technique and proved that PSPACE = IP. In this note, a slightly simplified version of Shamir's proof is presented, using degree reductions instead of simple QBFs.

Categories and Subject Descriptors: F. 1. 2 [**Computation by Abstract Devices**]: Modes of computation—*Alternation and nondeterminism; probabilistic computation*: F.1.3 [**Computation by Abstract Devices**]: Complexity classes—*relation among complexity classes*; F.4.1 [**Mathematical Logic and Formal Languages**]; Mathematical Logic—*proof theory*

General Terms: Theory

Additional Key Words and Phrases: Interactive proofs, PSPACE

General idea of the proof

- We will show that **QBF** is in **ABPP**
- For this, we will **arithmetize** the evaluation of QBF formulae

$$\forall X_1, \exists X_2, \forall X_3, \exists X_4, \ldots, \forall/\exists X_k, G(X_1,X_2,\ldots,X_k)$$

conjunction of 3-clauses

- by evaluating them as **polynomials**

which will act as **error-correcting** codes (but don't worry about that)

- … mod p
- because (low degree) polynomials provide proofs that are checkable with just **one random sample** (see next slides)

Polynomials mod p

Polynomials mod p

- Let p be prime: $K \stackrel{\text{def}}{=} \mathbb{Z}/p\mathbb{Z}$ is a **field**.
- $K[X_1,\ldots,X_m]$ = {polynomials $\Sigma_{n_1\ldots n_m} a_{n_1\ldots n_m} X_1^{n_1}\ldots X_m^{n_m}$ on m variables with coefficients $a_{n_1\ldots n_m}$ in K}

  sum of **monomials**
- For every polynomial P, one can **evaluate** P on an m-tuple $(v_1, \ldots, v_m)$ in K^m, yielding a value $P(v_1, \ldots, v_m)$ in K
- This defines a **function** $[\![P]\!] : K^m \to K$ (a so-called **polynomial function**)

Polynomials and polynomial functions

- One should (in principle) not confuse **polynomials** P with **polynomial functions** $[\![P]\!]$.
- For example, $X_1^p - X_1$ and 0 are distinct polynomials, which define the same function (Fermat's little theorem)
- However, there is no ambiguity if P has low degree: for two polynomials P, Q in **one variable** X_1, if $\deg(P)$, $\deg(Q) < p$, then $[\![P]\!]=[\![Q]\!]$ iff $P=Q$
- Equivalent to: if $\deg(P) < p$, then $[\![P]\!]=0$ iff $P=0$ because $P\neq0$ implies P has $\leq \deg(P)$ roots (**Lagrange**)

The Schwartz-Zippel Lemma

- This generalizes to multivariate polynomials.
- For $P \in K[X_1,\ldots,X_m] \stackrel{\text{def}}{=} \Sigma_{n_1\ldots n_m}\, a_{n_1\ldots n_m}\, X_1^{n_1}\ldots,X_m^{n_m}$
  the **total degree** $\deg(P) \stackrel{\text{def}}{=} \max \deg(a_{n_1\ldots n_m}\, X_1^{n_1}\ldots,X_m^{n_m})$
  where $\deg(a_{n_1\ldots n_m}\, X_1^{n_1}\ldots,X_m^{n_m}) \stackrel{\text{def}}{=} n_1+\ldots+n_m$ if $a_{n_1\ldots n_m}\neq 0$
  $\stackrel{\text{def}}{=} 0$ otherwise
- A **root** of P is an m-tuple $(v_1, \ldots, v_m)$ such that $P(v_1, \ldots, v_m)=0$
- **Theorem** (Schwartz 1980, Zippel 1979). Let $K \stackrel{\text{def}}{=} \mathbb{Z}/p\mathbb{Z}$, $m\geq 1$.
  Every $P \in K[X_1,\ldots,X_m]$ such that $P\neq 0$ has $\leq \deg(P).p^{m-1}$ roots.

The Schwartz-Zippel Lemma

- **Theorem** (Schwartz 1980, Zippel 1979). Let $K \stackrel{\text{def}}{=} \mathbb{Z}/p\mathbb{Z}$, $m \geq 1$. Every $P \in K[X_1,\ldots,X_m]$ such that $P \neq 0$ has $\leq \deg(P).p^{m-1}$ roots.
- By induction on m. We write P as a **univariate** polynomial in X_m, with coefficients in $K[X_1,\ldots,X_{m-1}]$:
$$P = Q_d\, X_m{}^d + Q_{d-1}\, X_m{}^{d-1} + \ldots + Q_1\, X_m + Q_0,$$
where $Q_d, Q_{d-1}, \ldots, Q_1, Q_0 \in K[X_1,\ldots,X_{m-1}]$ and $Q_d \neq 0$
- **Base case**: $m=1$, this is Lagrange.

The Schwartz-Zippel Lemma

- **Theorem** (Schwartz 1980, Zippel 1979). Let $K \stackrel{\text{def}}{=} \mathbb{Z}/p\mathbb{Z}$, $m \geq 1$.
Every $P \in K[X_1,\ldots,X_m]$ such that $P \neq 0$ has $\leq \deg(P).p^{m-1}$ roots.
- **Induction case** $m \geq 2$. $P = Q_d\, X_m{}^d + Q_{d-1}\, X_m{}^{d-1} + \ldots + Q_1\, X_m + Q_0$,
where $Q_d, Q_{d-1}, \ldots, Q_1, Q_0 \in K[X_1,\ldots,X_{m-1}]$ and $Q_d \neq 0$
- Note: $\deg(P) \geq \deg(Q_d)+d$. We count the roots $(v_1,\ldots,v_m)$ of P:
  - either $(v_1,\ldots,v_{m-1})$ is a root of Q_d: $\leq \deg(Q_d).p^{m-2}$ possible $(m-1)$-tuples,
times p possible values for v_m
  - or it is not: at most p^{m-1} possible $(m-1)$-tuples,
times $\leq d$ possible roots v_m (for each fixed $(m-1)$-tuple $(v_1,\ldots,v_{m-1})$)
- **Total**: $\leq \deg(Q_d).p^{m-2}.p + p^{m-1}.d = (\deg(Q_d)+d).p^{m-1} \leq \deg(P).p^{m-1}$. □

Polynomial identity testing

- **Theorem** (Schwartz 1980, Zippel 1979). Let $K \stackrel{\text{def}}{=} \mathbb{Z}/p\mathbb{Z}$, $m \geq 1$. Every $P \in K[X_1,\ldots,X_m]$ such that $P \neq 0$ has $\leq \deg(P).p^{m-1}$ roots.
- Consequence (**polynomial identity testing, PIT**):
  Given $P \in K[X_1,\ldots,X_m]$ with $d \stackrel{\text{def}}{=} \deg(P) < p$,
  if $P \neq 0$ then $\Pr_{v_1,\ldots,v_m \in K}(P(v_1,\ldots,v_m)=0) \leq d/p$.
- Hence the problem:
  INPUT: $P \in K[X_1,\ldots,X_m]$ with $d \stackrel{\text{def}}{=} \deg(P) < p/2$,
  QUESTION: $P \neq 0$?
  is in **RP**.

a « low degree polynomial »

provided evaluation of P can be done in **polynomial time**…

Complexity of arithmetic operations

Complexity of arithmetic operations

- Given numbers a, b of size $\leq f(n)$, in binary
- $a+b$: time $O(f(n))$, result size $\leq f(n)+1$
- $a.b$: time $O(f(n)^2)$, result size $\leq 2f(n)$
  [can be improved: Karatsuba $O(f(n)^{\log 3/\log 2})$, Toom-Cook $O(f(n)^{1+\varepsilon})$, Schönhage-Strassen $O(f(n) \log f(n) \log \log f(n))$)]
- a^b: result size $= b.\text{size}(a)$
  exponential in size(b)
  Hence no matter which algorithm we choose to implement a^b, running time will be exponential

Fast exponentiation

```
let rec pow(a,b)=
    if b=0
       then 1
    else let (b',lsb) = b divmod 2 in
         let r = pow(a,b') in
         let r2 = r*r in
         if lsb=0
            then r2
         else r2*a
```

- … this is why we turn to **mod** p operations

Complexity of operations mod p

- If p is of size $\leq f(n)$, then **all** numbers mod p are of size $\leq f(n)$
- Only new operation: x mod p
  Here is an easy way
  (assuming a on $\leq k$ bits, and $p \geq 1$;
  more efficient: see Montgomery representation):

```
r := x;
let q = p<<(k-1) in
for i=1 to k: (* Inv: q=p2^(k-i),r<2q,r=x mod p *)
    if r≥q then r -= q; (* r<q,r=x mod p *)
    q >>= 1;
```

- in time $O(k\, f(n))$. In practice, $x=ab$ has size $k = 2f(n)$.
  Hence ab mod p: time $O(f(n)^2)$ [same as for ab],
  but **size remains $\leq$ size(p) $\leq f(n)$**
- Hence any polynomial computation involving $A(n)$ additions and $M(n)$ multiplications mod p takes time time $O(A(n)f(n)+M(n)f(n)^2)$: **polynomial** if $A(n)$, $M(n)$, $f(n)$ are polynomial.

Complexity of operations mod p

- Any polynomial computation involving $A(n)$ additions and $M(n)$ multiplications mod p takes time time $O(A(n)f(n) + M(n)f(n)^2)$: **polynomial** if $A(n)$, $M(n)$, $f(n)$ are polynomial.
- Hence evaluating $P(v_1,\ldots,v_m)$ where $P \in K[X_1,\ldots,X_m]$, $K \stackrel{\text{def}}{=} \mathbb{Z}/p\mathbb{Z}$
  takes polynomial time if:
  (1) size(p)=$f(n)$ is polynomial
  (2) m is polynomial
  (3) P has polynomially many non-zero monomials

  P has polynomial size
- When m=1, (3) is equivalent to: deg(P) is **polynomial**
  (In general, #monomials is exponential = $O(\deg(P)^m)$)

Polynomials and polynomial expressions

- Until now, polynomials were given **explicitly**, as lists of monomials
- We will deal with **polynomial expressions**, namely expressions that **simplify** to polynomials
- E.g., $(x+1)(2y+3)^2$: needs 2 additions and 3 products simplifies to $4xy^2+4y^2+6xy+6y+9x+9$,
  which needs 5 additions and 9 products (and is larger!)
- Expressions will use extra operations: $\vee$, $\wedge$, $\neg$, $\forall$, $\exists$, $\underline{\mathrm{R}}$

Finding prime numbers (1/3)

- How do we find a prime number p of $f(n)$ bits?
- **Theorem (Bertrand's postulate, Chebyshev 1899).** For every natural number $N \geq 1$, there is at least one prime number p such that $N < p \leq 2N$;
  in fact there are strictly more than $N/(3 \log(2N))$

Theorem 5.7 (Bertrand's Postulate). *For any positive integer m, we have*

$$\pi(2m) - \pi(m) > \frac{m}{3\log(2m)}.$$

Victor Shoup. *A Computational Introduction to Number Theory and Algebra.* (Beta version 4.) `https://shoup.net/ntb/`

- Then rejection sampling + primality testing

Finding prime numbers (2/3)

- So $>2^{f(n)}/(3\,(f(n)+1)\log 2)$ primes of [exactly] $f(n)$ bits, out of $2^{f(n)-1}$ $f(n)$-bit numbers

Theorem (Bertrand's postulate, Chebyshev 1899). For every natural number N, there is at least one prime number p such that $N < p \leq 2N$; in fact there are strictly more than $N/(3\log(2N))$

Theorem 5.7 (Bertrand's Postulate). *For any positive integer m, we have*

$$\pi(2m) - \pi(m) > \frac{m}{3\log(2m)}.$$

Victor Shoup. *A Computational Introduction to Number Theory and Algebra.* (Beta version 4.) https://shoup.net/ntb/

- $\Pr_{p,\text{ of } f(n) \text{ bits}}(p \text{ is prime}) > 2/(3\,(f(n)+1)\log 2)$
- Hence rejection sampling will find an $f(n)$-bit prime number in at most $3/2 \log 2\,(f(n)+1)$ tries on average
- Primality checking is poly time [Agrawal,Kayal,Saxena 2002]
- Hence, if $f(n)$ is polynomial, then finding an $f(n)$-bit prime number can be done in **average polynomial time**

Finding prime numbers (3/3)

Hence, if $f(n)$ is polynomial, then finding an $f(n)$-bit prime number can be done in **average polynomial time**

- Imagine we can find an $f(n)$-bit prime number in average time $p(n)$
- By simulating this computation for $2p(n)$ steps,and failing if timeout is reached, either:
  — we obtain an $f(n)$-bit prime number in time $O(p(n))$
  — or we fail, with probability $\leq 1/2$
- Repeating this process while it fails, and at most $q(n)$ [polynomial] times, either:
  — we obtain an $f(n)$-bit prime number in time $O(q(n)p(n)\log n)$
  — or we fail, with probability $\leq 1/2^{q(n)}$

Drawing random numbers mod p

- Let p be an $f(n)$-bit prime number
- To draw v mod p at random **uniformly**: **rejection sampling** again
- stops in ≤2 iterations **on average**
- With a timeout of 4 iterations, we obtain a random v mod p in time $4f(n)$, or we fail with probability $\leq 1/2$
- Repeating this process while it fails,
  and at most $q(n)$ [polynomial] times, either:
  — we obtain an $f(n)$-bit random v mod p in time $O(q(n)f(n)\log n)$
  — or we fail, with probability $\leq 1/2^{q(n)}$

Arithmetization

Arithmetizing formulae

- We will interpret QBF formulae F as **polynomial expressions** $F(X_1,\ldots,X_m)$ (we will **not** simplify them as polynomials)
- … in such a way that for all **Booleans** $v_1,\ldots,v_m$,
  $F(v_1,\ldots,v_m)$ is the value of $F[X_1:=v_1,\ldots,X_m:=v_m]$
  (and is in particular Boolean; we let false=0, true=1)
- $P \wedge Q \stackrel{\text{def}}{=} P.Q \qquad \neg P \stackrel{\text{def}}{=} 1-P \qquad P \vee Q \stackrel{\text{def}}{=} 1-(1-P)(1-Q)$

Arithmetizing formulae

- $P \wedge Q \stackrel{\text{def}}{=} P.Q$ $\neg P \stackrel{\text{def}}{=} 1-P$ $P \vee Q \stackrel{\text{def}}{=} 1-(1-P)(1-Q)$
- **Example**: $(X_1 \wedge \neg X_2) \vee X_3 = 1-(1-X_1.(1-X_2))(1-X_3)$
- For a 3-clause C, $\deg(C) \leq 3$, constant size (counting the size of variables as one)
- For a set [conjunction] G of k 3-clauses,
  $\deg(G) \leq 3k$, size $O(k)$

k=poly(n), good!

Arithmetizing QBF formulae

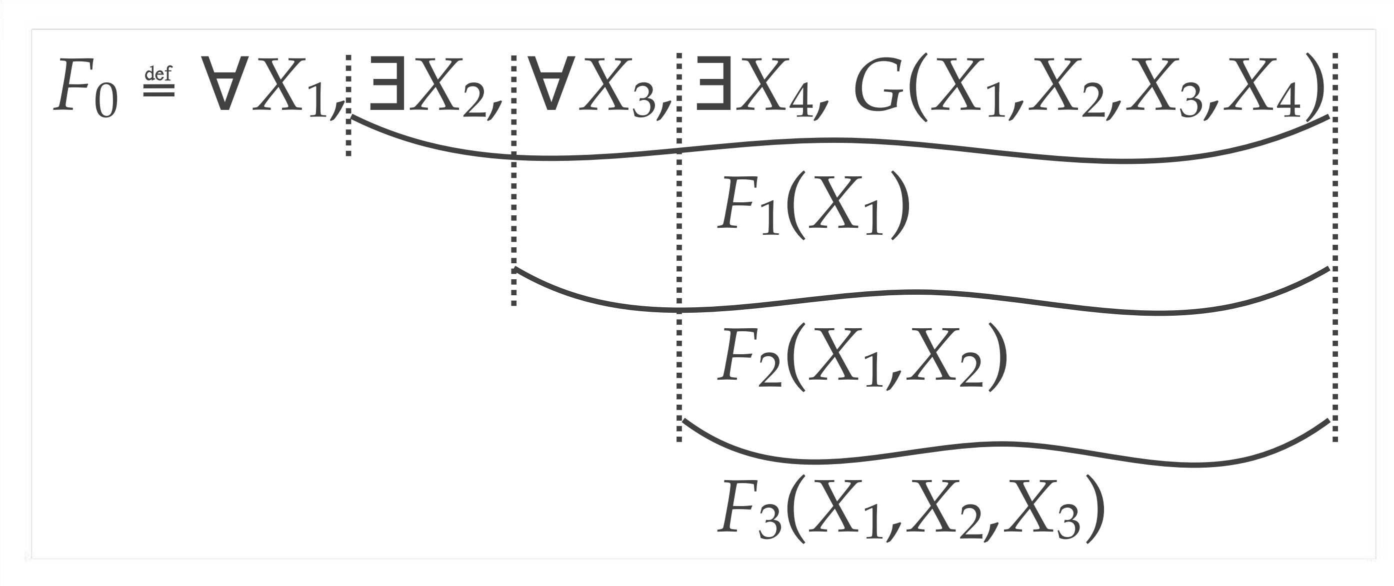
- $P \wedge Q \stackrel{\text{def}}{=} P.Q \qquad \neg P \stackrel{\text{def}}{=} 1-P \qquad P \vee Q \stackrel{\text{def}}{=} 1-(1-P)(1-Q)$
- $\forall X.P \stackrel{\text{def}}{=} P[X:=0] \wedge P[X:=1] \quad \exists X.P \stackrel{\text{def}}{=} P[X:=0] \vee P[X:=1]$
- Each quantifier **doubles** both the degree and the size
- For a set [conjunction] G of k 3-clauses,
  $\deg(G) \leq 3k$, size $O(k)$
- $\forall X_1, \exists X_2, \forall X_3, \exists X_4, \ldots, \forall/\exists X_m, G(X_1,X_2,\ldots,X_m)$
  degree: $2^m 3k$, size $O(2^m k)$

exponential: no problem for Schwartz-Zippel (take $f(n)$ polynomial > $m \log_2 (3k)$),
but will cause a **size** problem later (solved by Shen's trick, see later)

An **ABPP** game to decide QBF

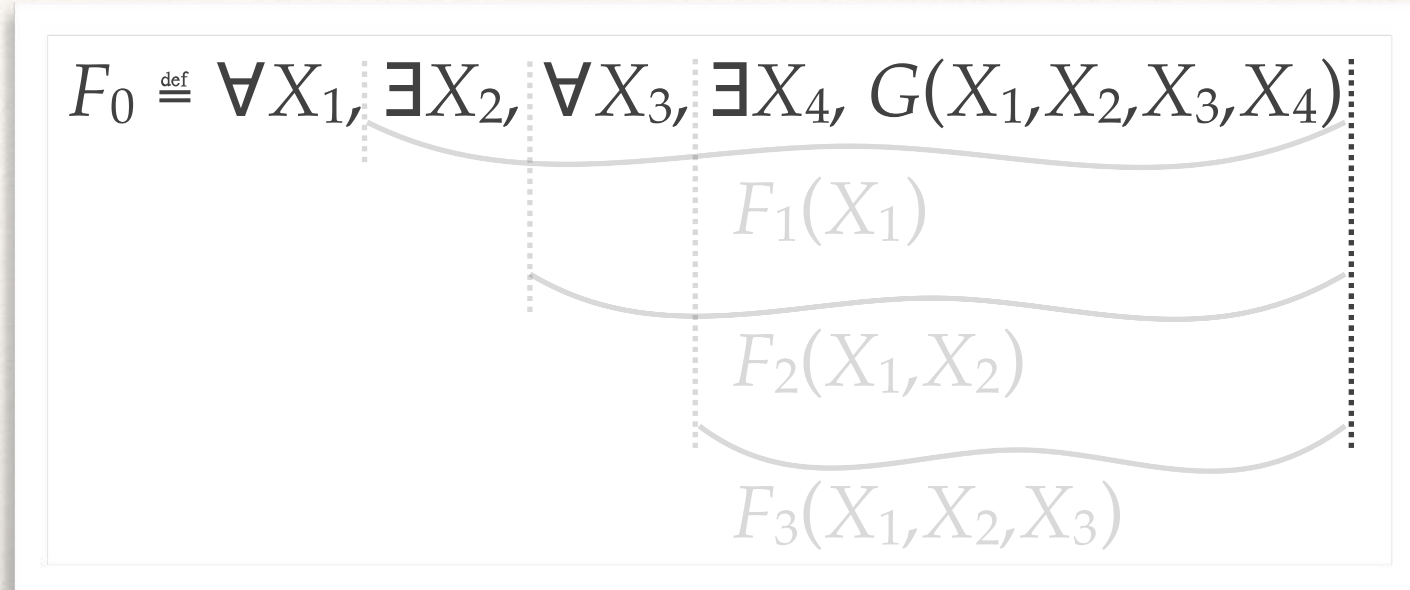
- We first assume that the max degree $d_{\max}$ of all polynomials we need to handle is **polynomial** (instead of $2^m 3k$)…
- This is wrong, but will be solved by Shen's trick later
- We let Arthur check that
  $$\forall X_1, \exists X_2, \forall X_3, \exists X_4, \ldots, \forall/\exists X_m, G(X_1,X_2,\ldots,X_m) = 1$$
  by asking Merlin for polynomials representing certain subformulae (~error-correcting codes), and checking them using Schwartz-Zippel
- There will be m rounds
- Let me explain this with m=4…

An ABPP game to decide QBF



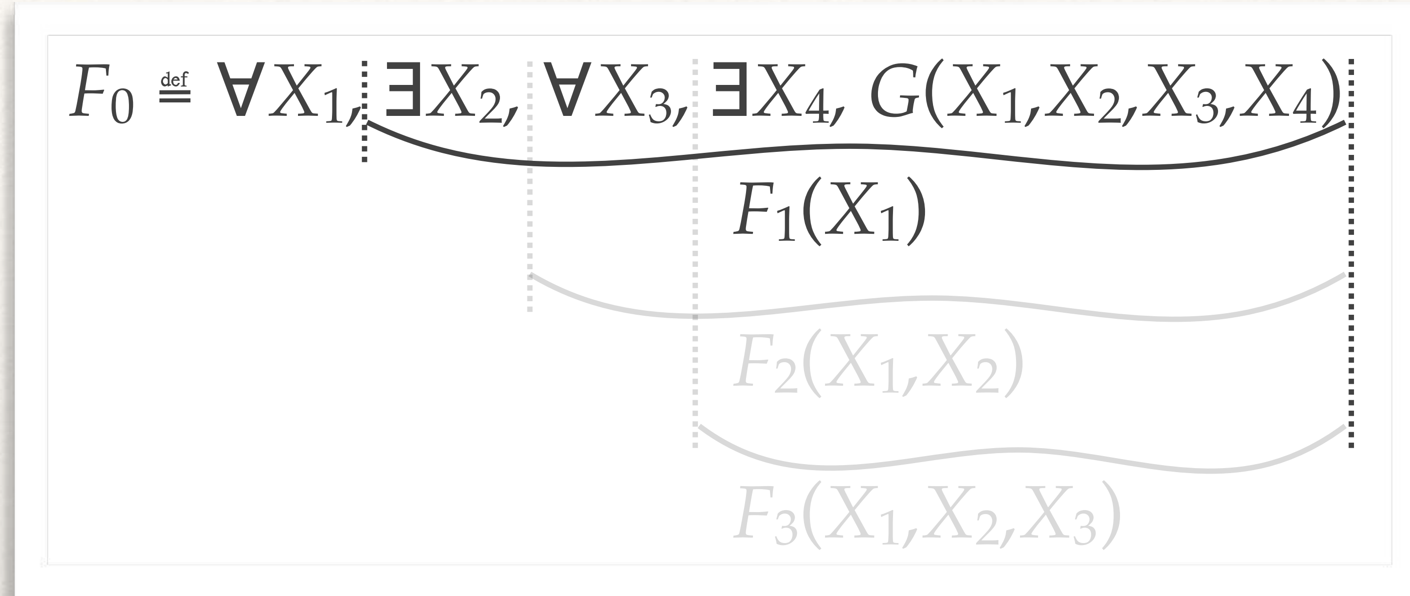
- At each point of the game, we will have a polynomial expression F (… with **no** variable) and an **objective** value w, and Arthur wishes to check whether $[\![F]\!]=w$.
- Initially, $F=F_0$, $w=w_0 \stackrel{\text{def}}{=} 1$

An ABPP game to decide QBF



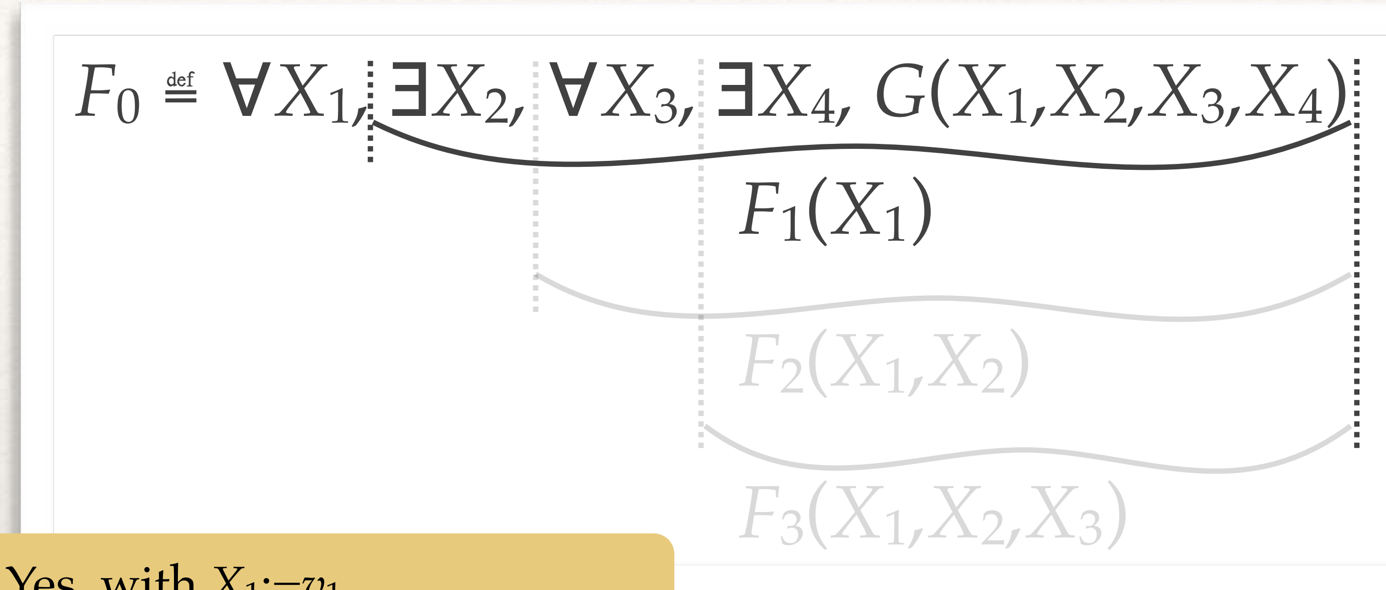
- Initially, $F=F_0$, $w=w_0 \stackrel{\text{def}}{=} 1$
- Arthur cannot check whether $[\![F_0]\!]=w_0$ (F_0 is too large)
- Merlin gives a polynomial (not a polynomial expression) $P_1(X_1)$, claiming that:
  — $[\![P_1(X_1)]\!] = [\![F_1(X_1)]\!]$
  — $[\![\forall X_1, P_1(X_1)]\!] = w_0$
- Since $d_{\max}$ is (assumed) polynomial, and $P_1(X_1)$ is **univariate**, $P_1(X_1)$ has **polynomial size**

An ABPP game to decide QBF



- Initially, F=F_0, w=$w_0 \stackrel{\text{def}}{=} 1$
- Merlin gives $P_1(X_1)$, claims:
  — $[\![P_1(X_1)]\!] = [\![F_1(X_1)]\!]$
  — $[\![\forall X_1, P_1(X_1)]\!] = w_0$
- Arthur checks that
  $[\![\forall X_1, P_1(X_1)]\!] = w_0$ by verifying that $P_1(0).P_1(1) = w_0$
  … admittedly, it is **very** easy for a dishonest Merlin to pass this test
- In order to check $[\![P_1(X_1)]\!] = [\![F_1(X_1)]\!]$,
  Arthur draws v_1 mod p uniformly, and needs to check $P_1(v_1)$=$F_1(v_1)$,
  by Schwartz-Zippel (on one variable), this is a **reliable** test
- Now $F = F_1(v_1)$, w=$w_1 \stackrel{\text{def}}{=} P_1(v_1)$

An ABPP game to decide QBF

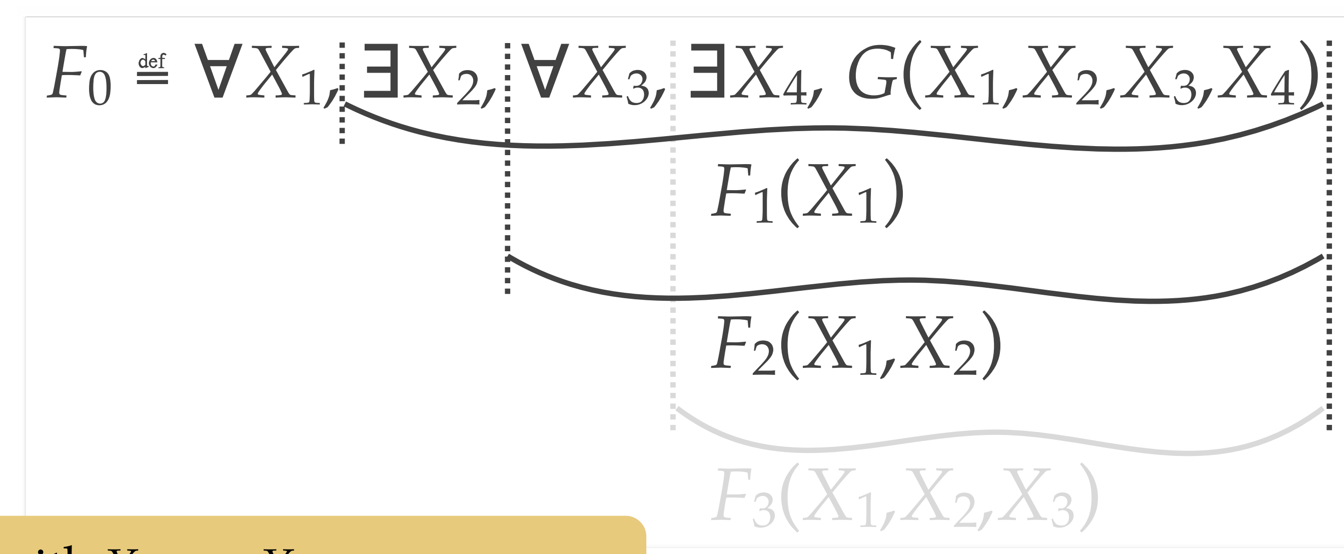


- Now $F = F_1(v_1)$, $w{=}w_1 \overset{\text{def}}{=} P_1(v_1)$
- Merlin gives $P_2(X_2)$, claims:
  — $[\![P_2(X_2)]\!] = [\![F_2(v_1,X_2)]\!]$
  — $[\![\exists X_2, P_2(X_2)]\!] = w_1$

Yes, with $X_1{:=}v_1$
Note that $P_2(X_2)$ is univariate, too.

- Arthur checks that
  $[\![\exists X_2, P_2(X_2)]\!] = w_1$ by verifying that $1–(1–P_2(0))(1–P_2(1)) = w_1$
- In order to check $[\![P_2(X_2)]\!] = [\![F_2(v_1,X_2)]\!]$,
  Arthur draws v_2 mod p uniformly, and needs to check $P_2(v_2){=}F_2(v_1,v_2)$,
  by Schwartz-Zippel (on one variable), this is a **reliable** test
- Now $F = F_2(v_1,v_2)$, $w{=}w_2 \overset{\text{def}}{=} P_2(v_2)$

An ABPP game to decide QBF

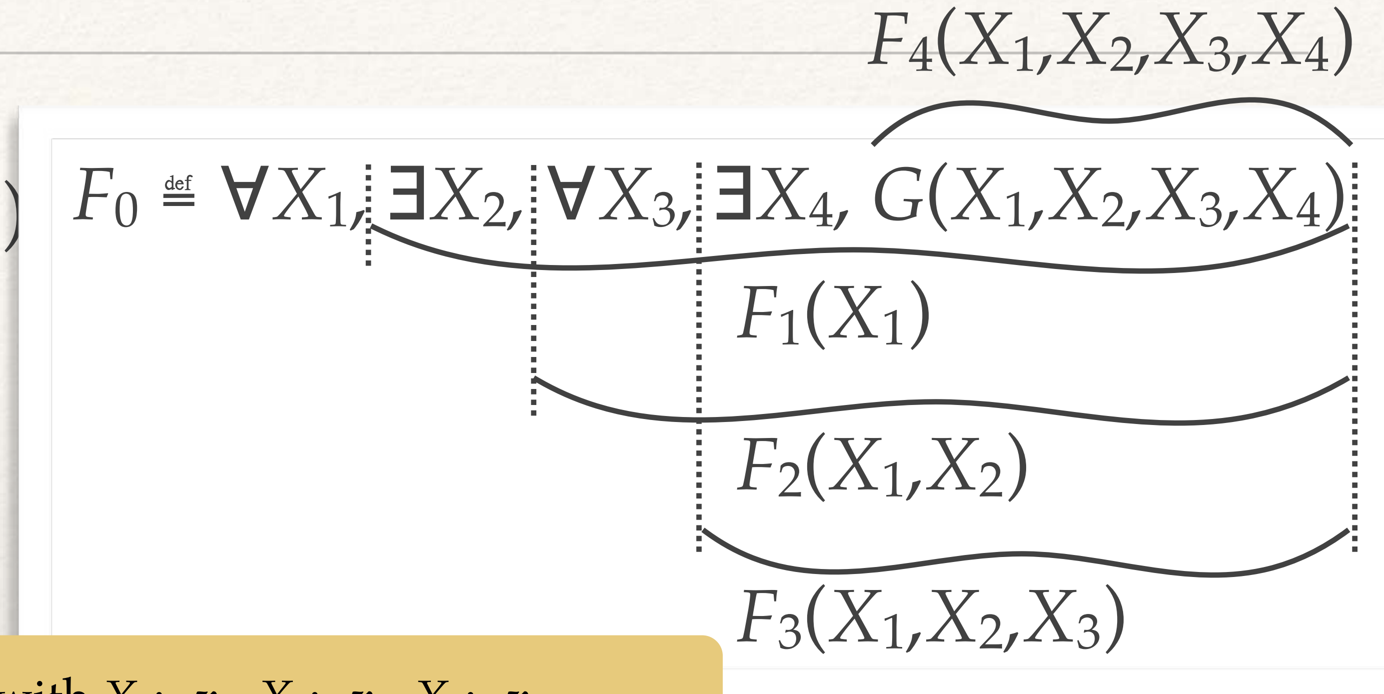


- Now $F = F_2(v_1,v_2)$, $w{=}w_2 \stackrel{\text{def}}{=} P_2(v_2)$
- Merlin gives $P_3(X_3)$, claims:
  — $[\![P_3(X_3)]\!] = [\![F_3(v_1,v_2,X_3)]\!]$
  — $[\![\forall X_3, P_3(X_3)]\!] = w_2$

Yes, with $X_1{:=}v_1$, $X_2{:=}v_2$
Note that $P_3(X_3)$ is univariate, too

- Arthur checks that
  $[\![\forall X_3, P_3(X_3)]\!] = w_2$ by verifying that $P_3(0)P_3(1) = w_2$
- In order to check $[\![P_3(X_3)]\!] = [\![F_3(v_1,v_2,X_3)]\!]$
  Arthur draws v_3 mod p uniformly, and will check $P_3(v_3){=}F_3(v_1,v_2,v_3)$,
  by Schwartz-Zippel (on one variable), this is a **reliable** test
- Now $F = F_3(v_1,v_2,v_3)$, $w{=}w_3 \stackrel{\text{def}}{=} P_3(v_3)$

An ABPP game to decide QBF



- Now $F = F_3(v_1,v_2,v_3)$, $w{=}w_3 \stackrel{\text{def}}{=} P_3(v_3)$
- Merlin gives $P_4(X_4)$, claims:
  — $[\![P_4(X_4)]\!] = [\![F_4(v_1,v_2,v_3,X_4)]\!]$
  — $[\![\exists X_4,\ P_4(X_4)]\!] = w_3$

Yes, with $X_1{:=}v_1$, $X_2{:=}v_2$, $X_3{:=}v_3$
Note that $P_4(X_4)$ is univariate, too

- Arthur checks that
  $[\![\exists X_4,\ P_4(X_4)]\!] = w_3$ by verifying that $1–(1–P_4(0))(1–P_4(1)) = w_3$
- In order to check $[\![P_4(X_4)]\!] = [\![F_4(v_1,v_2,v_3,X_4)]\!]$
  Arthur draws v_4 mod p uniformly, and will check $P_4(v_4){=}F_4(v_1,v_2,v_3,v_4)$,
  by Schwartz-Zippel (on one variable), this is a **reliable** test
- … and Arthur can do this by himself, since $F_4{=}G$. □

Error bounds

- If F_0 is true, then Merlin simply gives the simplified form of $F_k(v_1,v_2,\ldots,v_{k-1},X_k)$ for $P_k(X_k)$, at each turn k
- Arthur will **always** accept in the end, in that case

$$F_4(X_1,X_2,X_3,X_4)$$

$$F_0 \stackrel{\text{def}}{=} \forall X_1, \exists X_2, \forall X_3, \exists X_4, G(X_1,X_2,X_3,X_4)$$

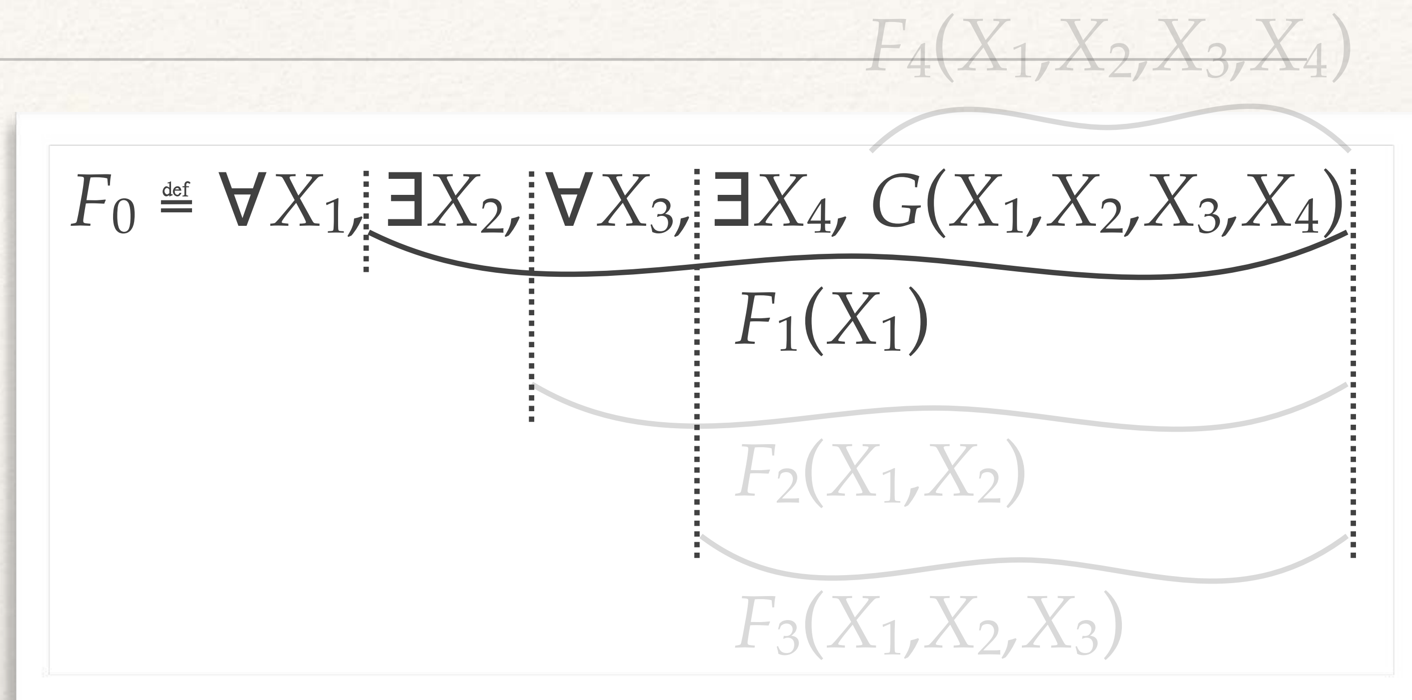
$$F_1(X_1)$$

$$F_2(X_1,X_2)$$

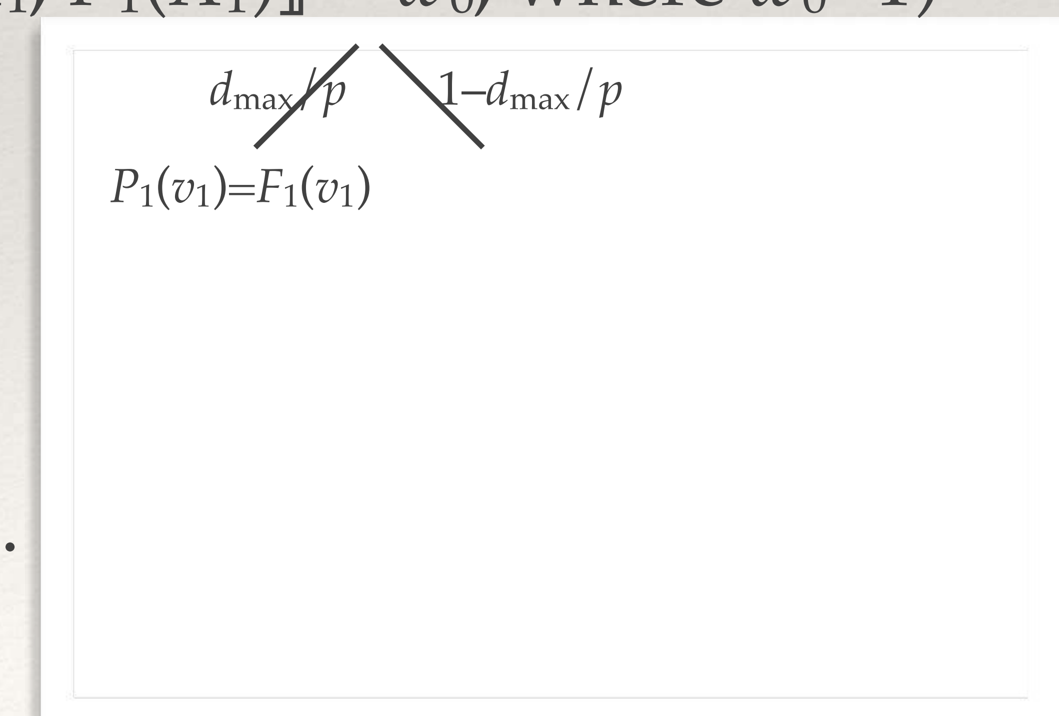
$$F_3(X_1,X_2,X_3)$$

Error bounds

- If F_0 is false, how can Merlin play (i.e., cheat) so as to force Arthur to eventually accept?



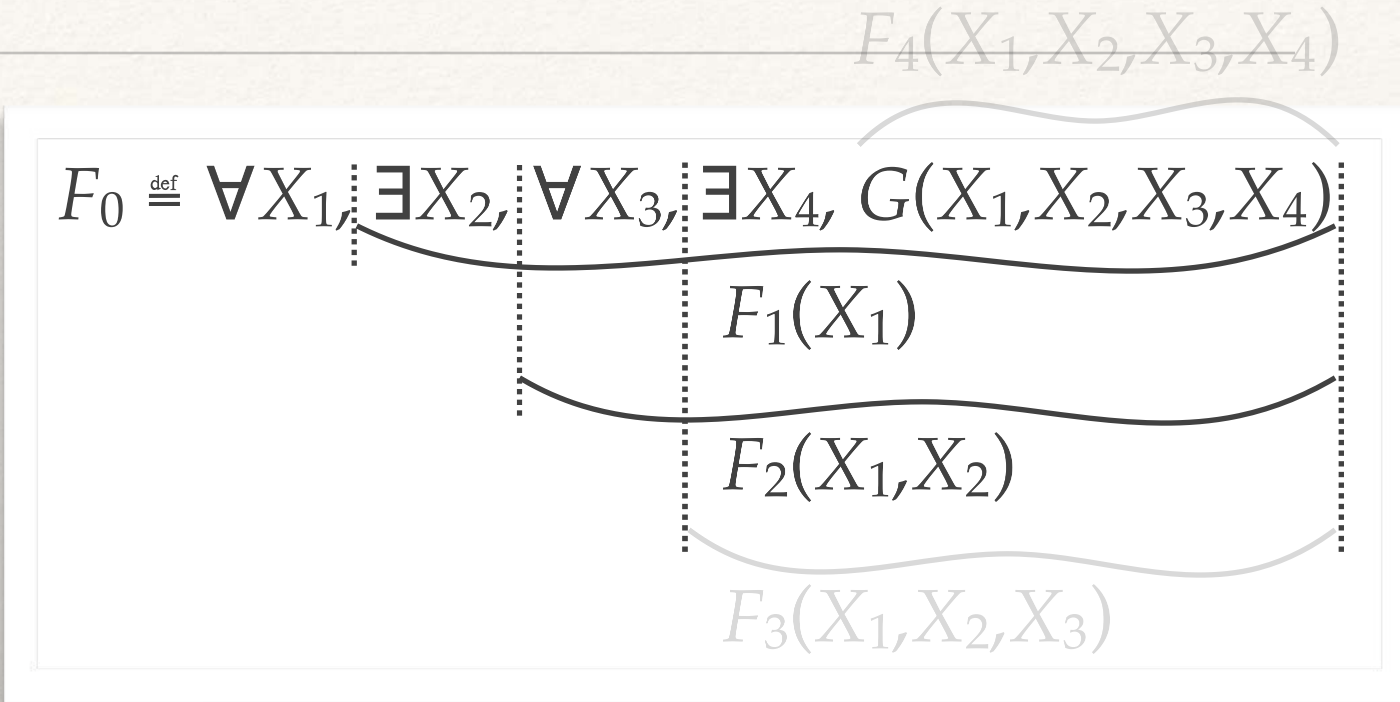
- **Round 1:** $P_1(X_1) \neq F_1(X_1)$ [as polynomials]
  since $[\![\forall X_1, P_1(X_1)]\!]=1$ (Arthur checks $[\![\forall X_1, P_1(X_1)]\!] = w_0$, where $w_0=1$)
  but $[\![\forall X_1, F_1(X_1)]\!]=[\![F_0]\!]=0$



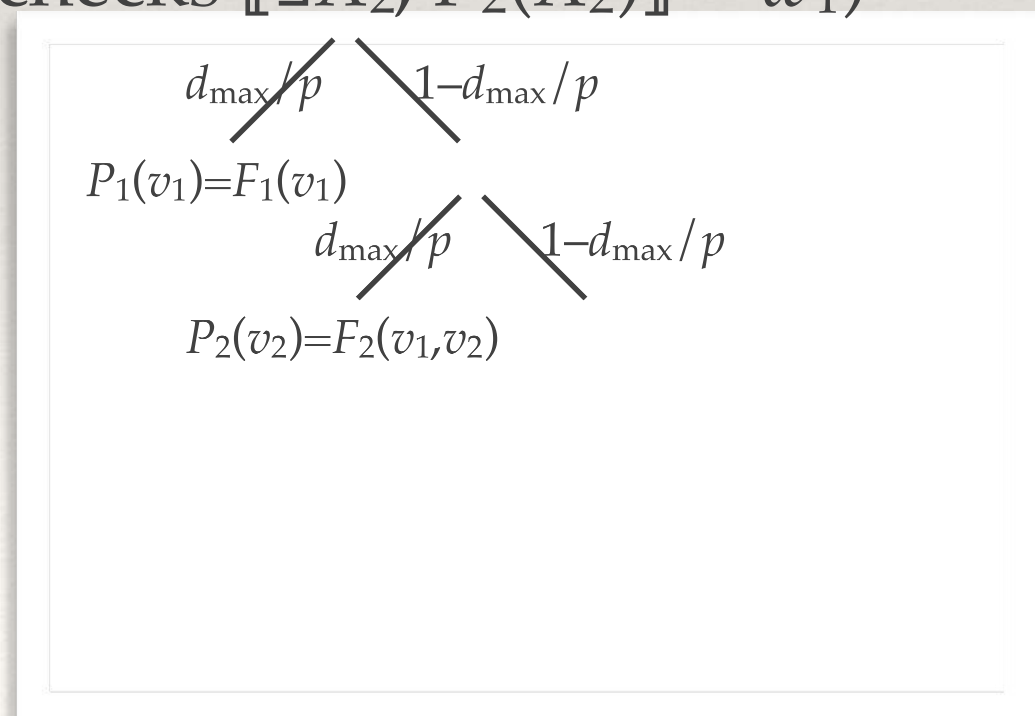
- With prob. $\leq d_{max}/p$ over v_1 (Schwartz-Zippel), $P_1(v_1)=F_1(v_1)$

- Otherwise, $F_1(v_1) \neq w_1$, where $w_1 \stackrel{\text{def}}{=} P_1(v_1)$, so…

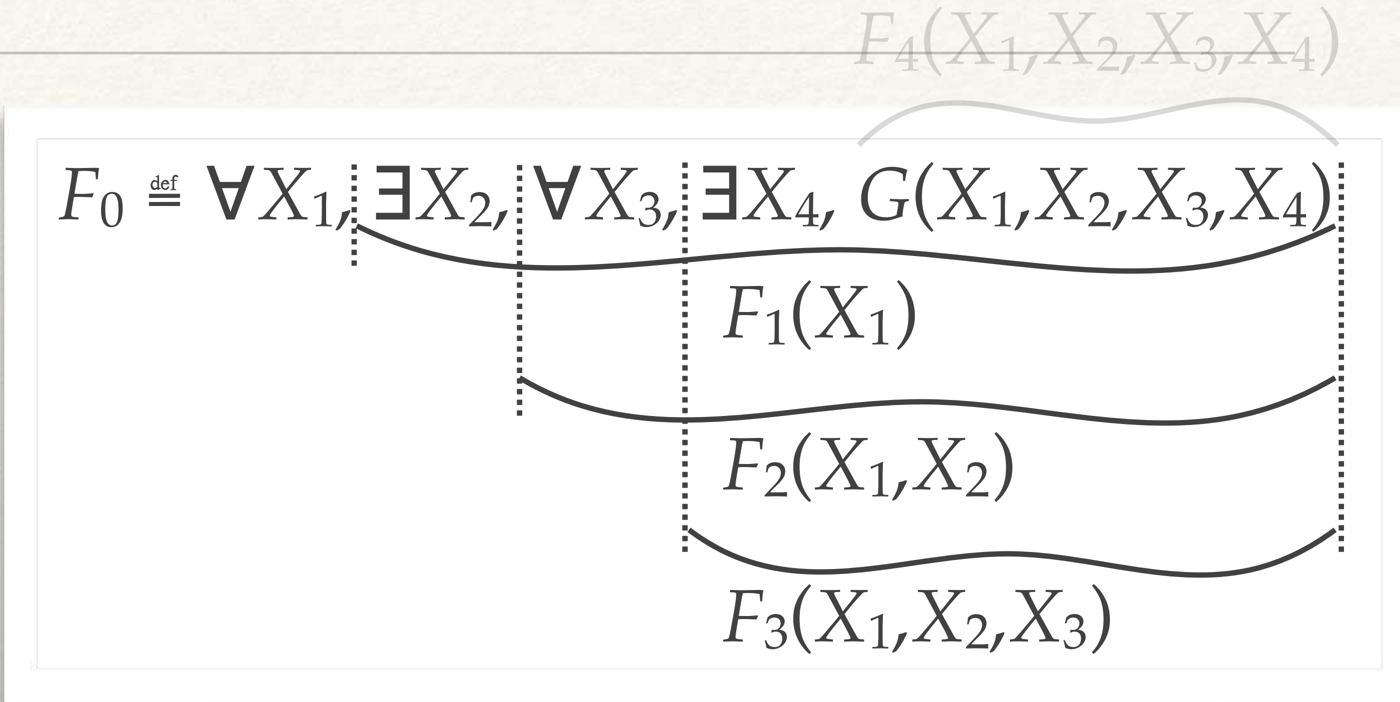
Error bounds



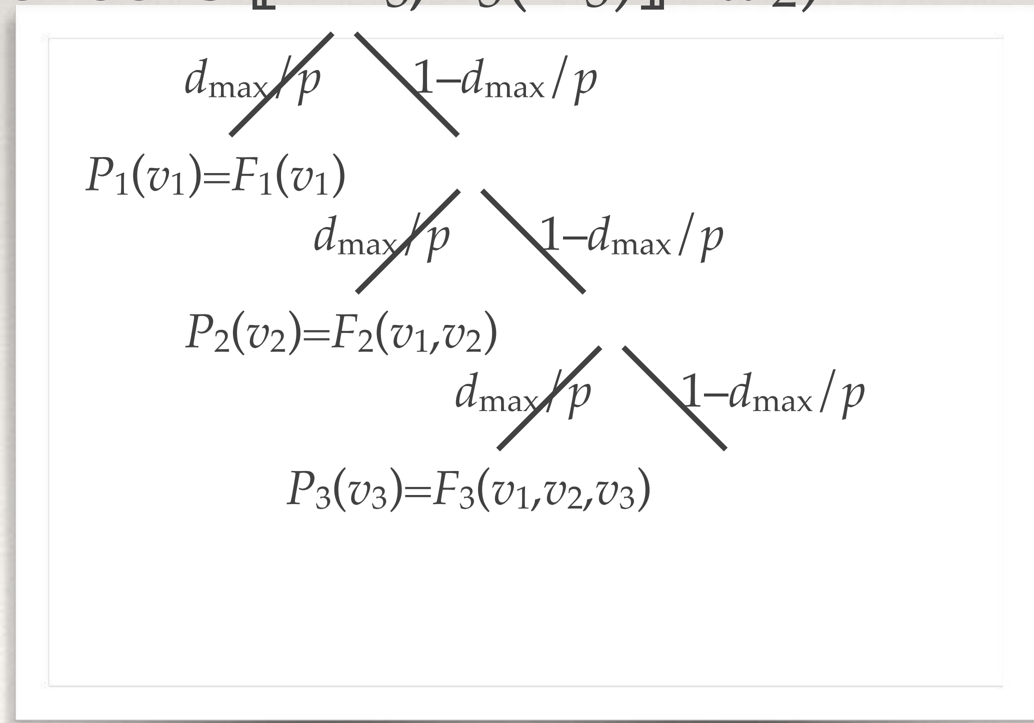
- If F_0 is false, how can Merlin play so as to force Arthur to eventually accept?
- Recap: now $F_1(v_1) \neq w_1$ $[w_1 \stackrel{\text{def}}{=} P_1(v_1)]$
- **Round 2:** $P_2(X_2) \neq F_2(v_1,X_2)$ [as polynomials] since $[\![\exists X_2, P_2(X_2)]\!] = w_1$ (since Arthur checks $[\![\exists X_2, P_2(X_2)]\!] = w_1$)
  but $[\![\exists X_2, F_2(v_1,X_2)]\!] = F_1(v_1) \neq w_1$
- With prob. $\leq d_{max}/p$ over v_2 (Schwartz-Zippel), $P_2(v_2) = F_2(v_1,X_2)$
- Otherwise, $F_2(v_1,v_2) \neq w_2$, where $w_2 \stackrel{\text{def}}{=} P_2(v_2)$, so…



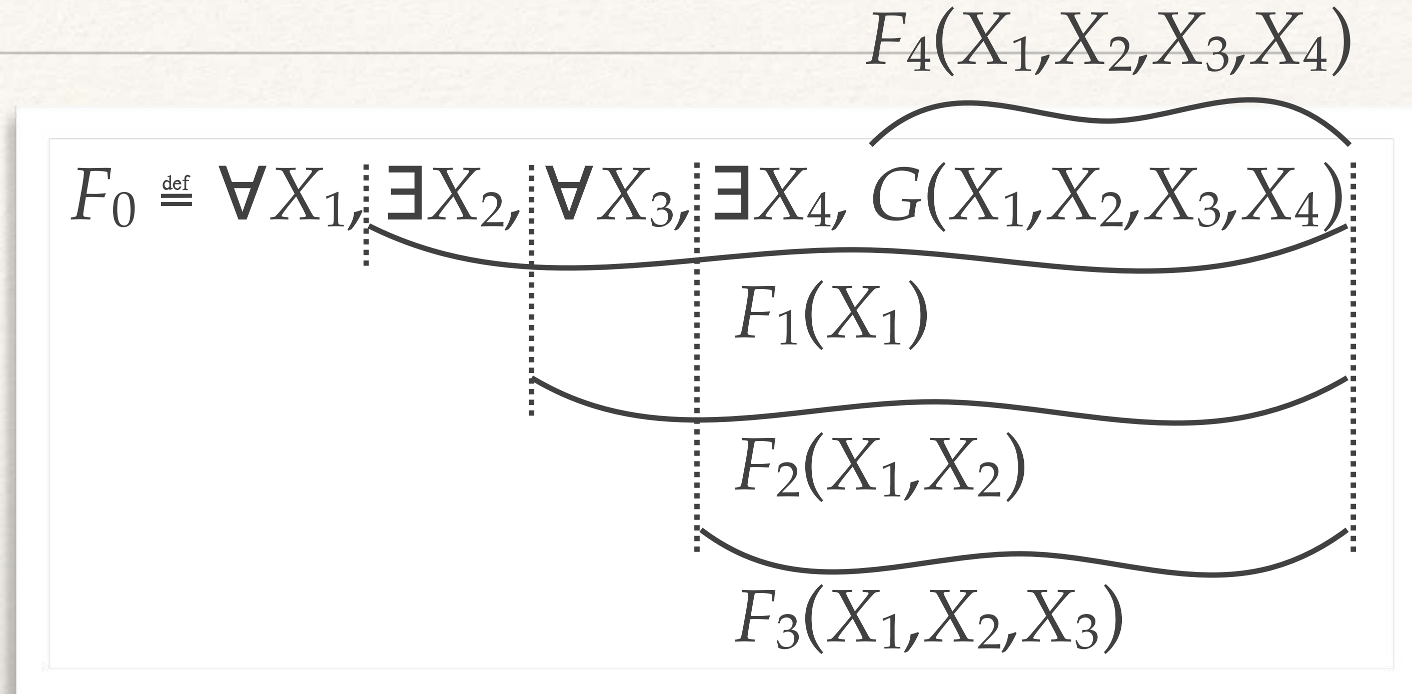
Error bounds



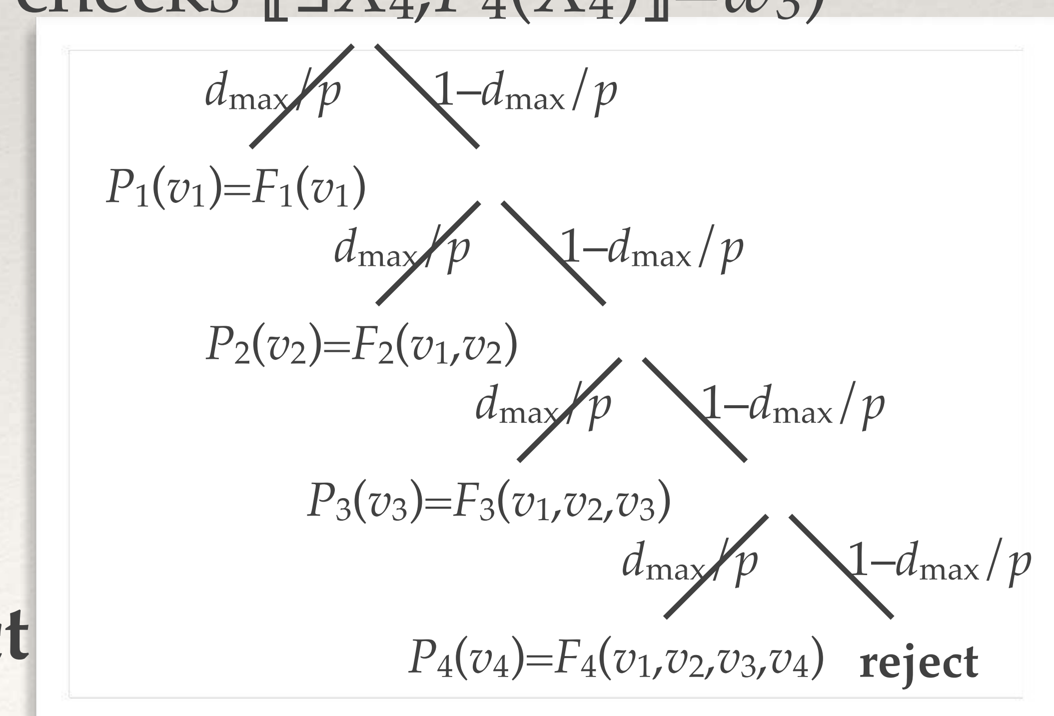
- If F_0 is false, how can Merlin play so as to force Arthur to eventually accept?
- Now $F_2(v_1,v_2) \neq w_2$ [$w_2 \stackrel{\text{def}}{=} P_2(v_2)$]
- **Round 3:** $P_3(X_3) \neq F_3(v_1,v_2,X_3)$ [as polynomials] since $[\![\forall X_3, P_3(X_3)]\!] = w_2$ (since Arthur checks $[\![\forall X_3, P_3(X_3)]\!] = w_2$)
  but $[\![\forall X_3, F_3(v_1,v_2,X_3)]\!] = F_2(v_1,v_2) \neq w_2$
- With prob. $\leq d_{\max}/p$ over v_3 (Schwartz-Zippel), $P_3(v_3) = F_3(v_1,v_2,v_3)$
- Otherwise, $F_3(v_1,v_2,v_3) \neq w_3$, where $w_3 \stackrel{\text{def}}{=} P_3(v_3)$, so…



Error bounds

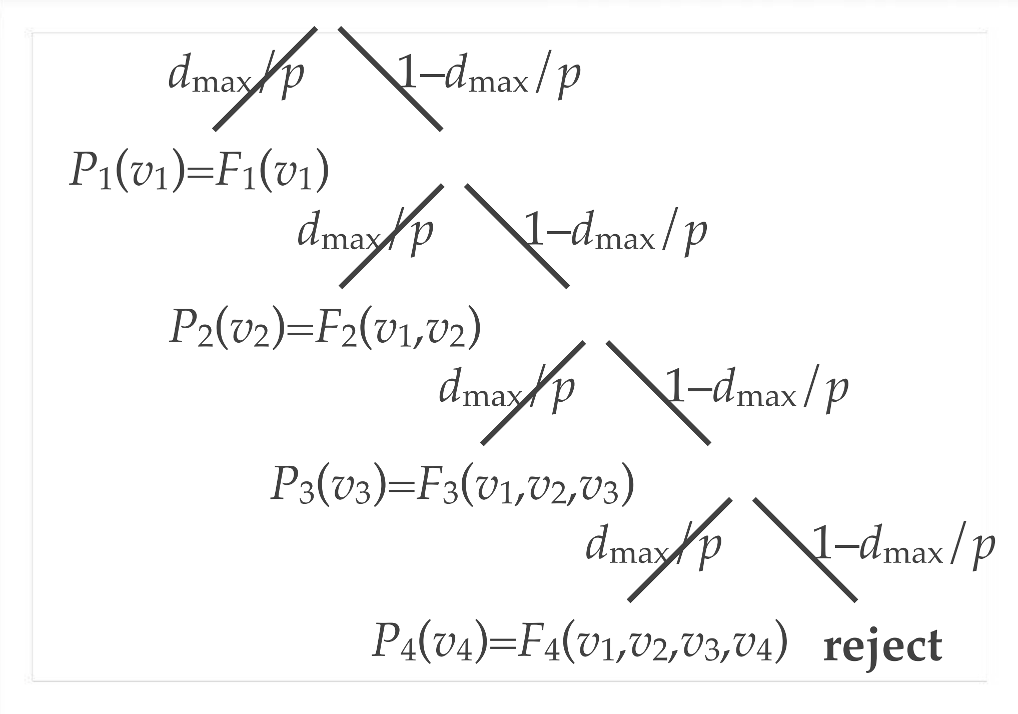


- If F_0 is false, how can Merlin play so as to force Arthur to eventually accept?
- Now $F_3(v_1,v_2,v_3)\neq w_3$ $[w_3 \stackrel{\text{def}}{=} P_3(v_3)]$
- **Round 4:** $P_4(X_4)\neq F_4(v_1,v_2,v_3,X_4)$ [as polynomials] since $[\![\exists X_4, P_4(X_4)]\!]=w_3$ (since Arthur checks $[\![\exists X_4, P_4(X_4)]\!]=w_3$)
  but $[\![\exists X_4, F_4(v_1,v_2,v_3,X_4)]\!]=F_3(v_1,v_2,v_3)\neq w_3$
- With prob. $\leq d_{max}/p$ over v_4 (Schwartz-Zippel), $P_4(v_4)=F_4(v_1,v_2,v_3,v_4)$
- Otherwise, $F_4(v_1,v_2,v_3,v_4)\neq w_4$, where $w_4 \stackrel{\text{def}}{=} P_4(v_4)$, but Arthur will then **reject**



Error bounds

- If F_0 is false, then probability of acceptance is $\leq 4d_{\max}/p$
- That was for m=4 quantified variables
- In the general case, $F_0 = \forall X_1, \exists X_2, \forall X_3, \exists X_4, \ldots, \forall/\exists X_m,$ $G(X_1,X_2,\ldots,X_m)$ and prob. of acceptance $\leq m d_{\max}/p$
- But all that works in poly time only if $d_{\max}$ is polynomial in n…



Shen’s trick

Shen's trick: degree reduction

- Given $P \in K[X]$, let

  $\underline{R}X, P(X) \stackrel{\text{def}}{=} AX+B$

  where $B \stackrel{\text{def}}{=} P(0)$

  $A \stackrel{\text{def}}{=} P(1)–P(0)$

New « quantifier » $\underline{R}$ (reduction).
Beware that $\underline{R}X, P(X)$ still depends on X

$RX, P(X)$ is really $P(X)$ mod $(X^2–X)$

- At the Boolean level, $\underline{R}$ is a no-op:

  $\underline{R}X, P(X)$ and $P(X)$ have the **same** values on X=0 or 1

- … but the degree of $\underline{R}X, P(X)$ is **at most one** (in X)

Shen's trick: using R

- Instead of checking whether the polynomial expression
  $$\forall X_1, \exists X_2, \forall X_3, \exists X_4, \ldots, \forall/\exists X_m, G(X_1,X_2,\ldots,X_m)$$
  evaluates to 1,
- we consider the polynomial expression
  $$\forall X_1, \underline{\mathrm{R}}X_1,$$
  $$\exists X_2, \underline{\mathrm{R}}X_1, \underline{\mathrm{R}}X_2,$$
  $$\forall X_3, \underline{\mathrm{R}}X_1, \underline{\mathrm{R}}X_2, \underline{\mathrm{R}}X_3,$$
  $$\exists X_4, \underline{\mathrm{R}}X_1, \underline{\mathrm{R}}X_2, \underline{\mathrm{R}}X_3, \underline{\mathrm{R}}X_4,$$
  $$\ldots$$
  $$\forall/\exists X_m, \underline{\mathrm{R}}X_1, \underline{\mathrm{R}}X_2, \ldots, \underline{\mathrm{R}}X_m, G(X_1,X_2,\ldots,X_m)$$
- That has now $m+m(m+1)/2$ quantifiers instead of m (polynomial)

Testing $\underline{\mathrm{R}}$ probabilistically

- Instead of just ∀ and ∃ rounds, there are now also $\underline{\mathrm{R}}$ rounds
  They are dealt with in a very similar way:

- Imagine $F_k(X) = \underline{\mathrm{R}}X, F_{k+1}(X)$ [just showing var. X for clarity]
  and Arthur wishes to check $F_k(v_k)=w_k$ [current objective]

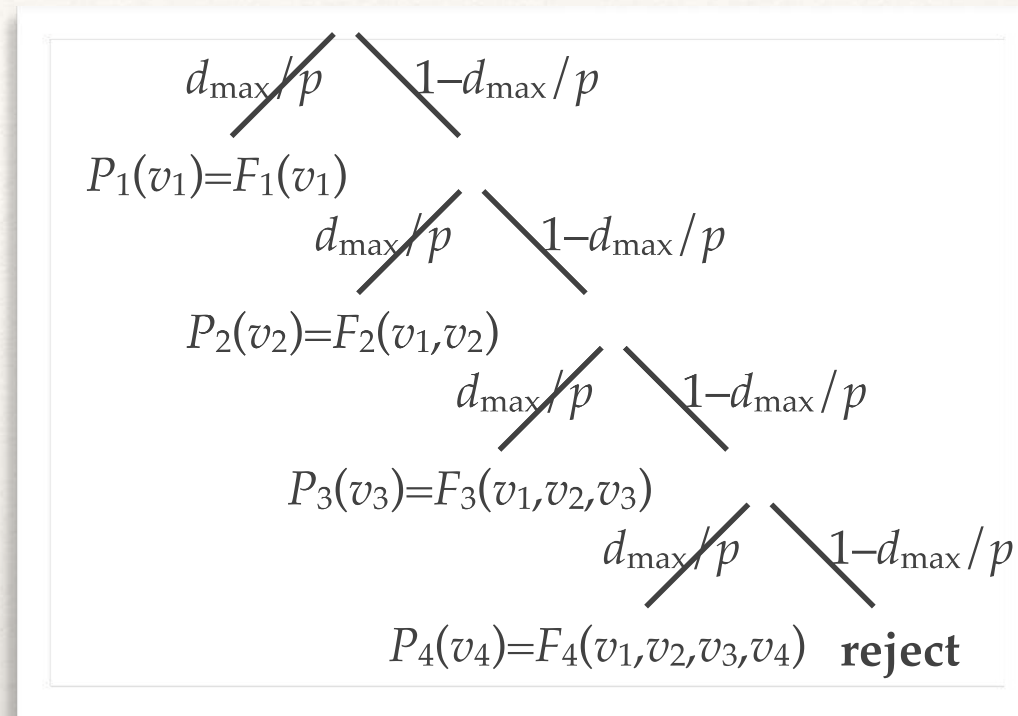
- Merlin provides univariate polynomial $P_{k+1}(X)$, claims:
  — $[\![P_{k+1}(X)]\!] = [\![F_{k+1}(X)]\!]$
  — $[\![\underline{\mathrm{R}}X, P_{k+1}(X)]\!](v_k) = w_k$

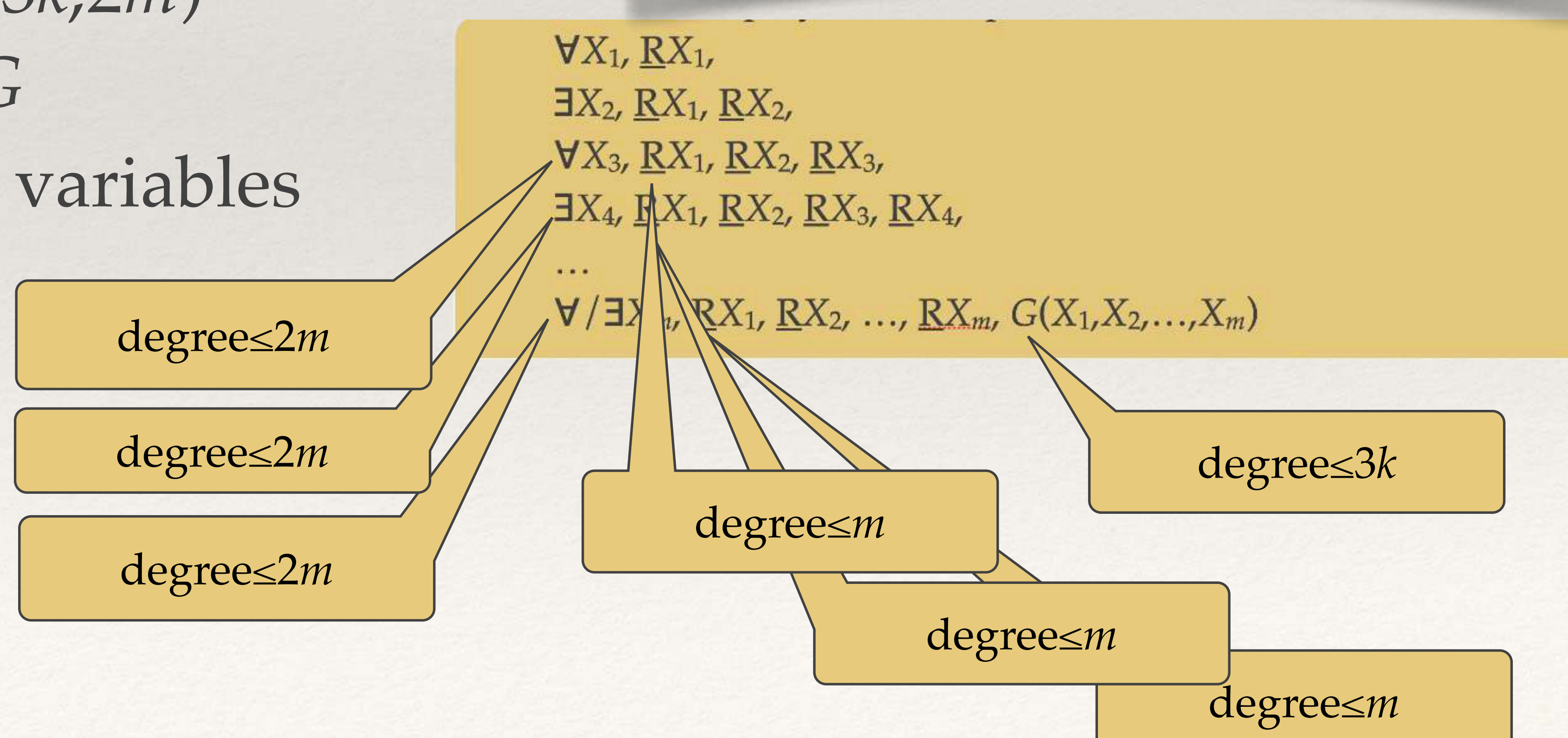
- Arthur checks $[\![\underline{\mathrm{R}}X, P_{k+1}(X)]\!](v_k) = w_k$, i.e., $Av_k+B=w_k$,
  where $B \stackrel{\text{def}}{=} P_{k+1}(0)$, $A \stackrel{\text{def}}{=} P_{k+1}(1)–P_{k+1}(0)$

- … then goes on to the next round by drawing v_{k+1} mod p,
  with the goal of checking $F_{k+1}(v_{k+1})=w_{k+1}$, where $w_{k+1} \stackrel{\text{def}}{=} P_{k+1}(v_{k+1})$

Error bounds, and $d_{\max}$

- If F_0 is false, then probability of acceptance is ≤ #quantifiers.$d_{\max}/p$
- Now #quantifiers = $m+m(m+1)/2$
- and (new!) $d_{\max}$ is **polynomial** in n…
- precisely, at most $\max(3k,2m)$
  where $k \stackrel{\text{def}}{=}$ #clauses in G
  $m \stackrel{\text{def}}{=}$ #quantified variables
  … **linear** in size(F_0)





The final adjustments (1/3)

- If F_0 is false, then probability of acceptance is $\leq \#\text{quantifiers}.d_{\max}/p$
  We need to make that $\leq 1/2^{q(n)}$, for an arbitrary polynomial $q(n)$
  Let us aim for $1/2^{q(n)+1}$, really (we will see why later)

- $d_{\max} \leq \max(3k, 2m) \leq 3n$, $\#\text{quantifiers} = m + m(m+1)/2 \leq (n^2+3n)/2 \leq 2n^2$ [if $n \geq 1$], so we require:

$$p \geq 2^{q(n)+1}.6n^3$$

- Let us draw p at random on $f(n)$ bits [in poly time], where

$$f(n) \stackrel{\text{def}}{=} q(n) + \lceil 3 \log_2 n + \log_2 6 \rceil + 2$$

  … failing with probability $\leq 1/2^{q(n)+2}$

- If that did not fail, then

$$p \geq 2^{f(n)-1} \geq 2^{q(n)+1}.6n^3, \text{ as required}$$

Repeating this process while it fails,
and at most $q(n)$ [polynomial] times, either:
— we obtain an $f(n)$-bit prime number in time $O(q(n)p(n)\log n)$
— or we fail, with probability $\leq 1/2^{q(n)}$

The final adjustments (2/3)

- During the whole game, we will draw numbers mod p
  #quantifiers $= m+m(m+1)/2 \leq 2n^2$ times

- Each time, this may fail,
  and we arrange the probability of
  failure to be $\leq 1/(2n^2 \,.\, 2^{q(n)+2})$,
  viz. $\leq 1/2^{q'(n)}$, where $q'(n)$ is some polynomial $\geq q(n)+2+\log_2(2n^2)$

> Repeating this process while it fails,
> and at most $q(n)$ [polynomial] times, either:
> — we obtain an $f(n)$-bit random v mod p in time $O(q(n)f(n)\log n)$
> — or we fail, with probability $\leq 1/2^{q(n)}$

- Hence the total probability of failure is at most:
  — $1/2^{q(n)+2}$ when drawing p
  — $1/2^{q(n)+2}$ for the $\leq 2n^2$ draws of numbers mod p
  hence at most $1/2^{q(n)+1}$

The final adjustments (3/3)

- The total probability of failure is at most $1/2^{q(n)+1}$
- In case of failure, Arthur immediately **accepts**.
  This way,
  - if F_0 is true, then if Merlin plays honestly,
    Arthur will eventually accept, either because the game goes as planned, or because some failure occurs
  - if F_0 is false, then whatever strategy Merlin uses,
    acceptance occurs only if failure (prob. $\leq 1/2^{q(n)+1}$)
    or if game goes on as planned
    but Arthur does not detect Merlin's cheating
    (prob. $\leq 1/2^{q(n)+1}$ as well, by our choice of p)
  - … hence with probability $\leq 1/2^{q(n)}$. $\square$

Conclusion

- We have proved:

and with **perfect soundness**! no error if $x \in L$

Theorem. QBF is in **ABPP**.

- Since QBF is **PSPACE**-complete, and since **ABPP** is **closed under poly time reductions**,

Corollary. PSPACE ⊆ ABPP

- With the previous result **ABPP ⊆ IP ⊆ PSPACE**:

- **Corollary** (Shamir's theorem). **ABPP = IP = PSPACE.**

and every PSPACE language has an ABPP protocol with **perfect soundness**

Next time…

Next time

- A glimpse at the Arora-Safra theorem

  $$\mathbf{NP} = \mathbf{PCP}(O(\log n), O(1), O(1))$$

- … specially its relationship to the hardness of **approximation** problems