Automatizability and Interpolation

Paul Beame

University of Washington

PCMI 2000 Thurs July 27

Review proof system definition

- A proof system for a language **L** is a polynomial time algorithm **V** s.t.
  - for all inputs **x**
    - **x**∈**L** iff ***there exists a string P s.t. V accepts input (x,P)***
    -
  - think of **P** as a *proof* that **x** is in **L** and **V** as a proof *verifier*

Complexity of proof systems

- **Defn:** The **complexity** a proof system **V** is a function **f:N→N** defined by

$$\mathbf{f(n)} = \max_{\mathbf{x}\in\mathbf{L},|\mathbf{x}|=\mathbf{n}} \ \min_{\mathbf{P}:\ \mathbf{V}\text{ accepts }(\mathbf{x},\mathbf{P})} |\mathbf{P}|$$

  - i.e. how large **P** has to be as a function of **|x|**
  - **V** is **polynomially-bounded** **iff** its complexity is a polynomial function of **n**

- Definition says **nothing** about how costly it is to find short proofs!
  - lower bounds are even stronger that way

Automatizability (sic)

- **Defn:** Given a proof system **V** for **L** and a function **$f: N \times N \to N$** we say that **V** is **f(n,S)-automatizable** iff there an algorithm **A_V** s.t.
  - given any input **x** with **$|x|=n$**, if **$x \in L$**, **A** outputs a proof **P** in **V** of this fact in time at most **f(n,S)** where **S** is the size of the shortest proof in **V** that **x** is in **L**

- We say that **V** is **automatizable** iff it is **f(n,S)**-automatizable for some f that is **$n^{O(1)}S^{O(1)}$**
  - **i.e., can find a proof in time polynomial in the size of the smallest one**

Width & Automatizability

- **Theorem** [BW]: **Every Davis-Putnam (DLL)/tree-like resolution proof of F of size S can be converted to one of width** $\lceil \log_2 S \rceil + w(F)$

- **Corollary** [CEI][BP][BW]: **Tree-like resolution is** $S^{O(\log n)}$**-automatizable**

- **Proof:** There are only $2^{\log S}\binom{n}{\log S} = n^{O(\log S)}$ clauses of length at most $\log S$. Run breadth-first resolution only deriving clauses of width $\log S$. Can keep space requirements down by making it a recursive search.

Width, Resolution, and PCR

- **Theorem** [BW] **Every resolution proof of F of size S can be converted to one of width**

$$O(\sqrt{n \log S}) + w(F)$$

- **Corollary: General resolution is** $2^{O(\sqrt{n \log S} \log n)}$**-automatizable**

- **Theorem: Tree-PCR and PCR are** $S^{O(\log n)}$**-automatizable and** $2^{O(\sqrt{n \log S} \log n)}$**-automatizable respectively**
  - **There are roughly n^d monomials of degree at most d & Groebner-basis like algorithm does linear algebra in that basis**

Interpolation

- Given formulas
  - **A(x,z)** in variables **x** and **z**
  - **B(y,z)** in variables **y** and **z**

- **Defn:** If $A(x,z) \lor B(y,z)$ is a tautology then an **interpolant C** is a function s.t.
  - **for any truth assignment ζ to z**
    - **$C(\zeta)=0$ implies $A(x,\zeta)$ is a tautology**
    - **$C(\zeta)=1$ implies $B(y,\zeta)$ is a tautology**

- Also dual form if $A(x,z) \land B(y,z)$ is unsatisfiable

Interpolation - origin of the name

- Given formulas
  - **A(x,z)** in free variables **x** and **z**
  - **B(y,z)** in free variables **y** and **z**

- **Theorem:** [Craig] If **A(x,z)**®**B(y,z)** is a tautology then there is an interpolant **C** with only free variables **z** such that **A(x,z)**®**C(z)** and **C(z)**®**B(y,z)**.

  - i.e. given Ø**A(x,z)**Ú**B(y,z)**: **C(z)**®**B(y,z)**, Ø**C(z)**®Ø**A(x,z)**

Feasible Interpolation

- **Defn:** Given a propositional proof system **V** and a function **$f:N \rightarrow N$** we say that **V** has **f-interpolation** iff **given an unsatisfiable formula of the form**
  $A(x,z) \wedge B(y,z)$ with proof size S in V
  there is a circuit of size at most $f(S)$ computing
  an interpolant C for $A(x,z) \wedge B(y,z)$;
  i.e. that says which of $A(x,z)$ or $B(y,z)$ is false

- **V has feasible interpolation iff f is polynomial**

- **V has monotone f-interpolation iff whenever the variables z occur only negatively in B and only positively in A, the circuit C is a monotone circuit.**

Automatizability & Interpolation

- **Lemma:** [Impagliazzo,BPR] If **V** is automatizable then **V** has feasible interpolation

- Proof: Let f be the polynomial function such that V is f-automatizable and $\mathbf{A_V}$ be the associated algorithm.
- Given unsatisfiable $\mathbf{A(x,z) \wedge B(y,z)}$ and an assignment z to **z**:

Automatizability & Interpolation

- **Lemma:** [Impagliazzo,BPR] If **V** is automatizable then **V** has feasible interpolation

- Proof: Let f be the polynomial function such that V is f-automatizable and A_V be the associated algorithm.
- Given unsatisfiable $A(x,z) \wedge B(y,z)$ and an assignment z to z:
  - **Run A_V on input $A(x,z) \wedge B(y,z)$ to a proof P of size $S' \le f(S)$ where S is the size of its optimal proof in V**

Automatizability & Interpolation

- **Lemma:** [Impagliazzo,BPR] If **V** is automatizable then **V** has feasible interpolation

- Proof: Let f be the polynomial function such that V is f-automatizable and A_V be the associated algorithm.
- Given unsatisfiable $A(x,z) \wedge B(y,z)$ and an assignment z to z:
  - **Run A_V on input $A(x,z) \wedge B(y,z)$ to a proof P of size $S' \leq f(S)$ where S is the size of its optimal proof in V**
  - **Run A_V on input $A(x,z)$ for $f(S')$ steps**
    - **if it finds a proof output 0**
    - **else output 1**

Automatizability & Interpolation

- **Lemma:** [Impagliazzo,BPR] If **V** is automatizable then **V** has feasible interpolation

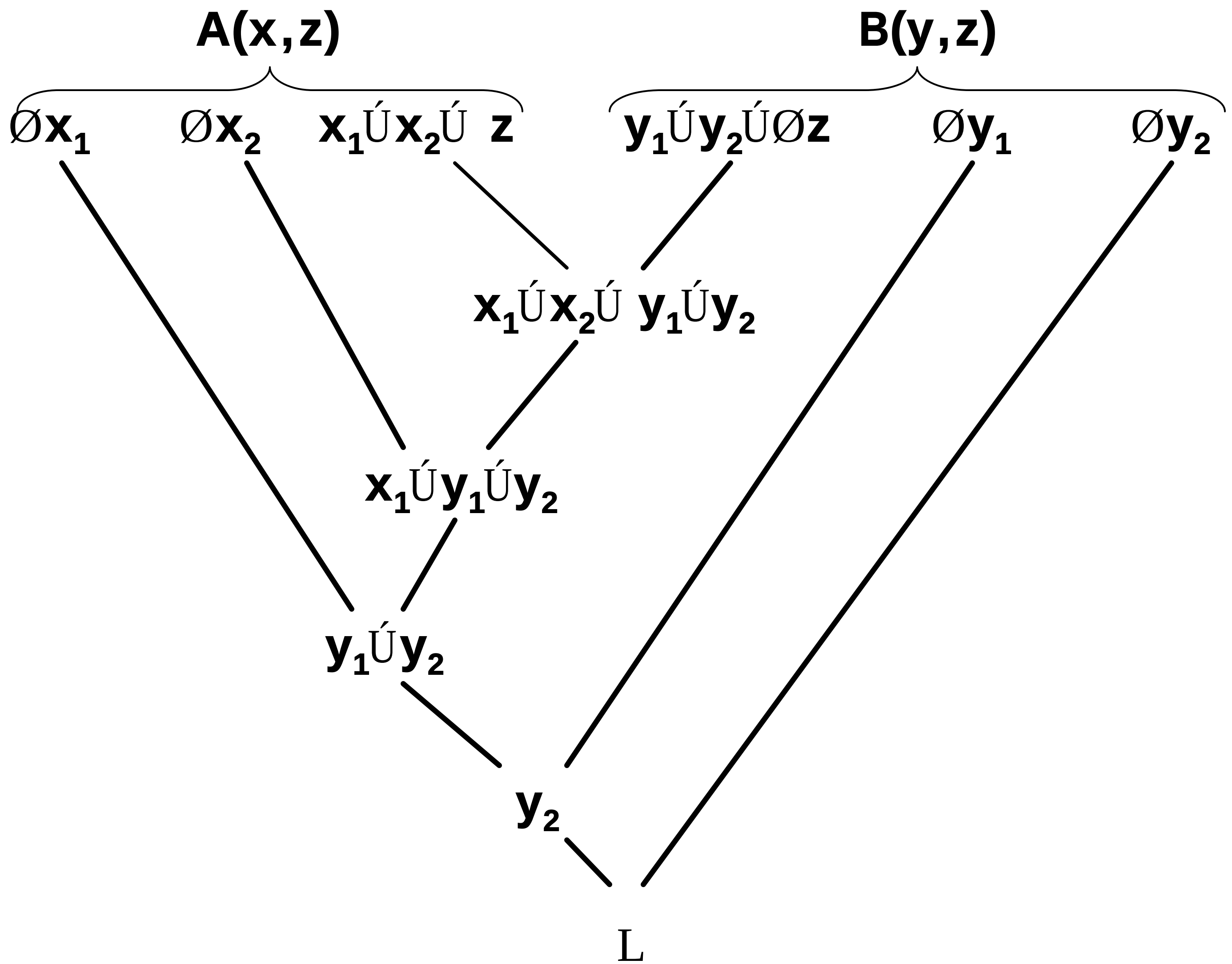
- Proof: Let f be the polynomial function such that V is f-automatizable and A_V be the associated algorithm.
- Given unsatisfiable $A(x,z) \wedge B(y,z)$ and an assignment ζ to **z**:
  - **Run A_V on input $A(x,z) \wedge B(y,z)$ to a proof P of size $S' \leq f(S)$ where S is the size of its optimal proof in V**
  - **Run A_V on input $A(x,\zeta)$ for $f(S')$ steps**
    - **if it finds a proof output 0**
    - **else output 1**
- **Note that if $B(y,\zeta)$ has satisfying assignment σ then plugging σ,ζ into the proof P yields a proof of size S' of unsatisfiability of $A(x,\zeta) \wedge B(\sigma,\zeta)$ which is $A(x,\zeta) \wedge 1$**

Interpolation and Resolution

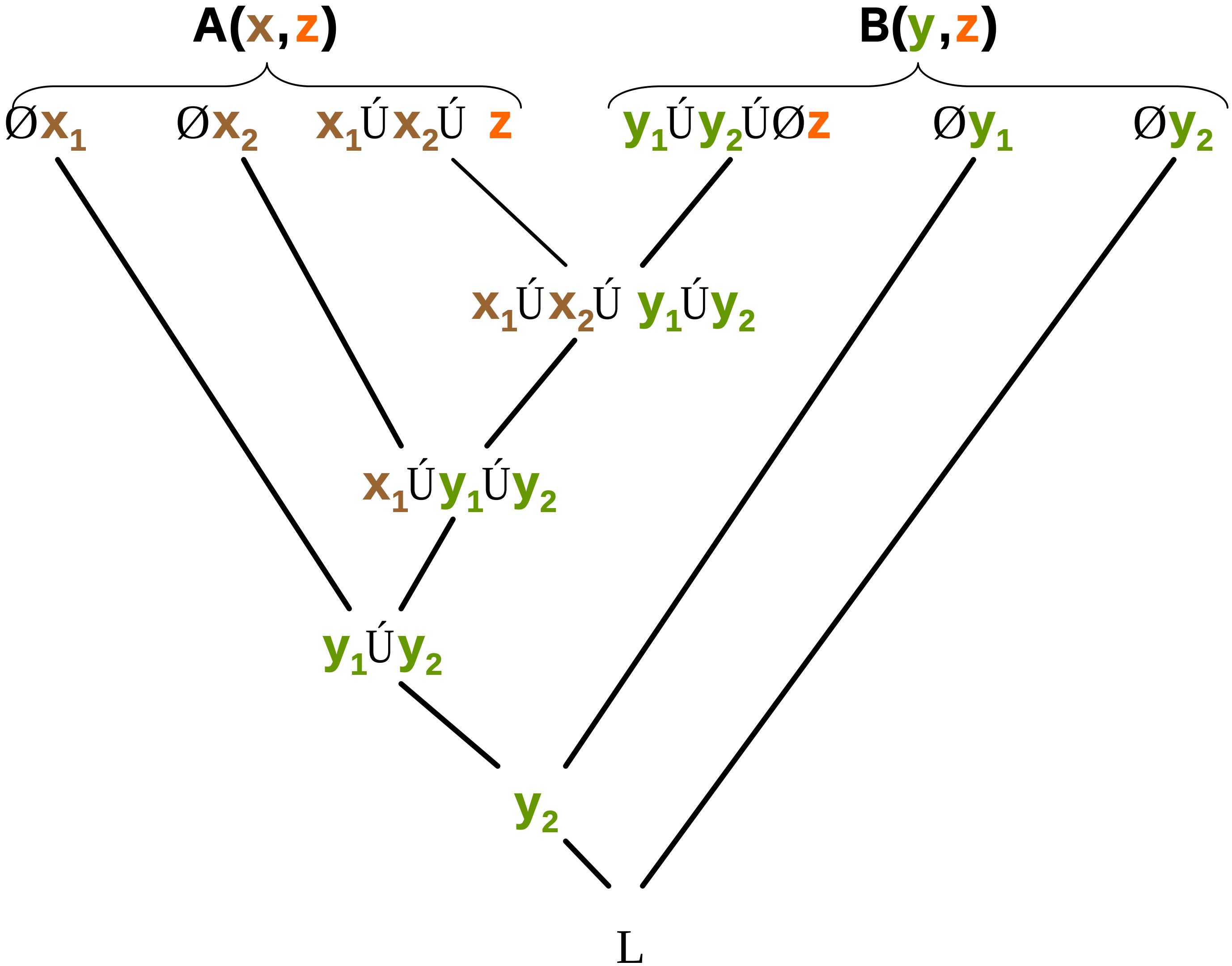
- **Theorem:** [Krajicek] Resolution has feasible (monotone) interpolation.

- **Proof idea:** structure of proof allows one to decide easily which clauses cause unsatisfiability

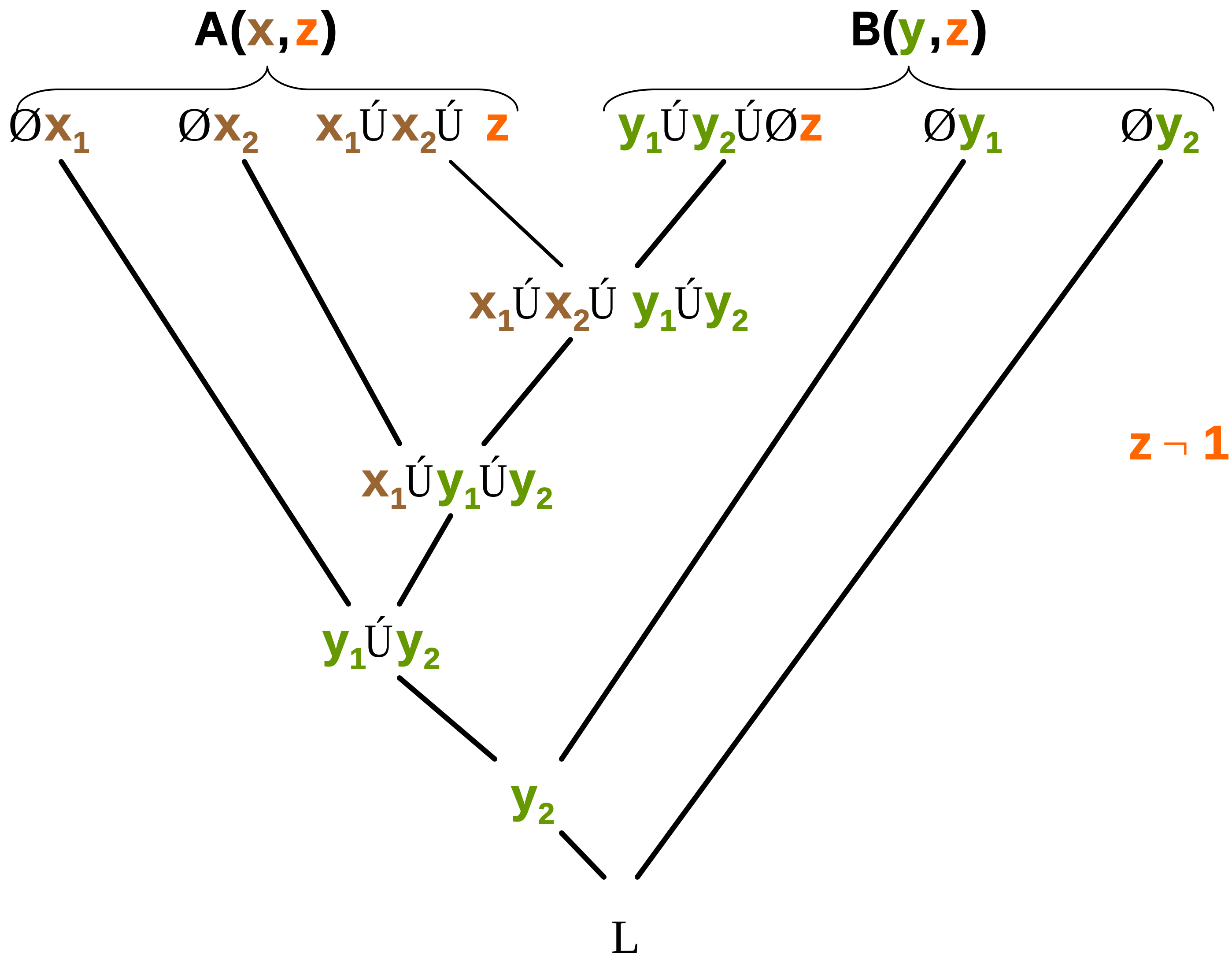
Interpolation for Resolution



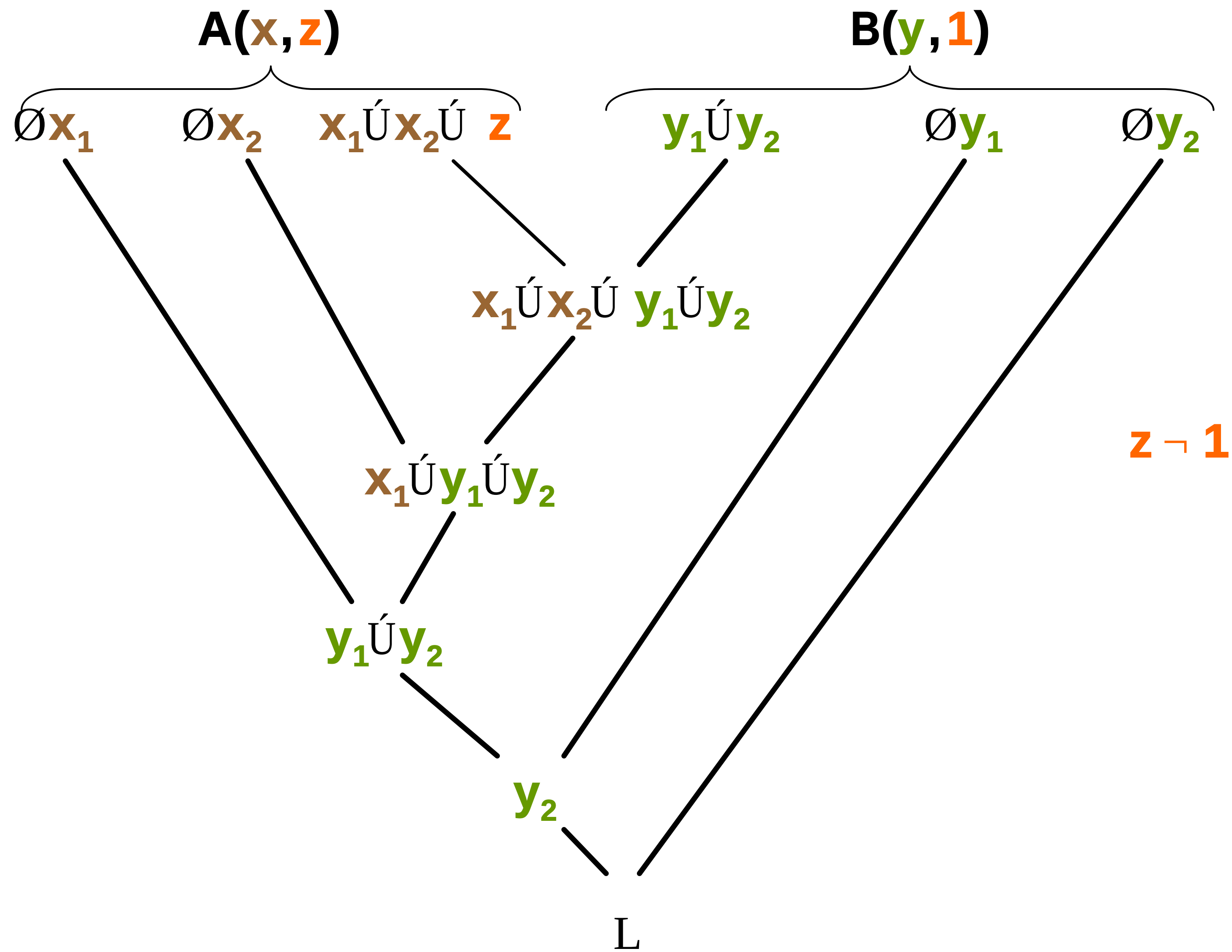
Interpolation for Resolution



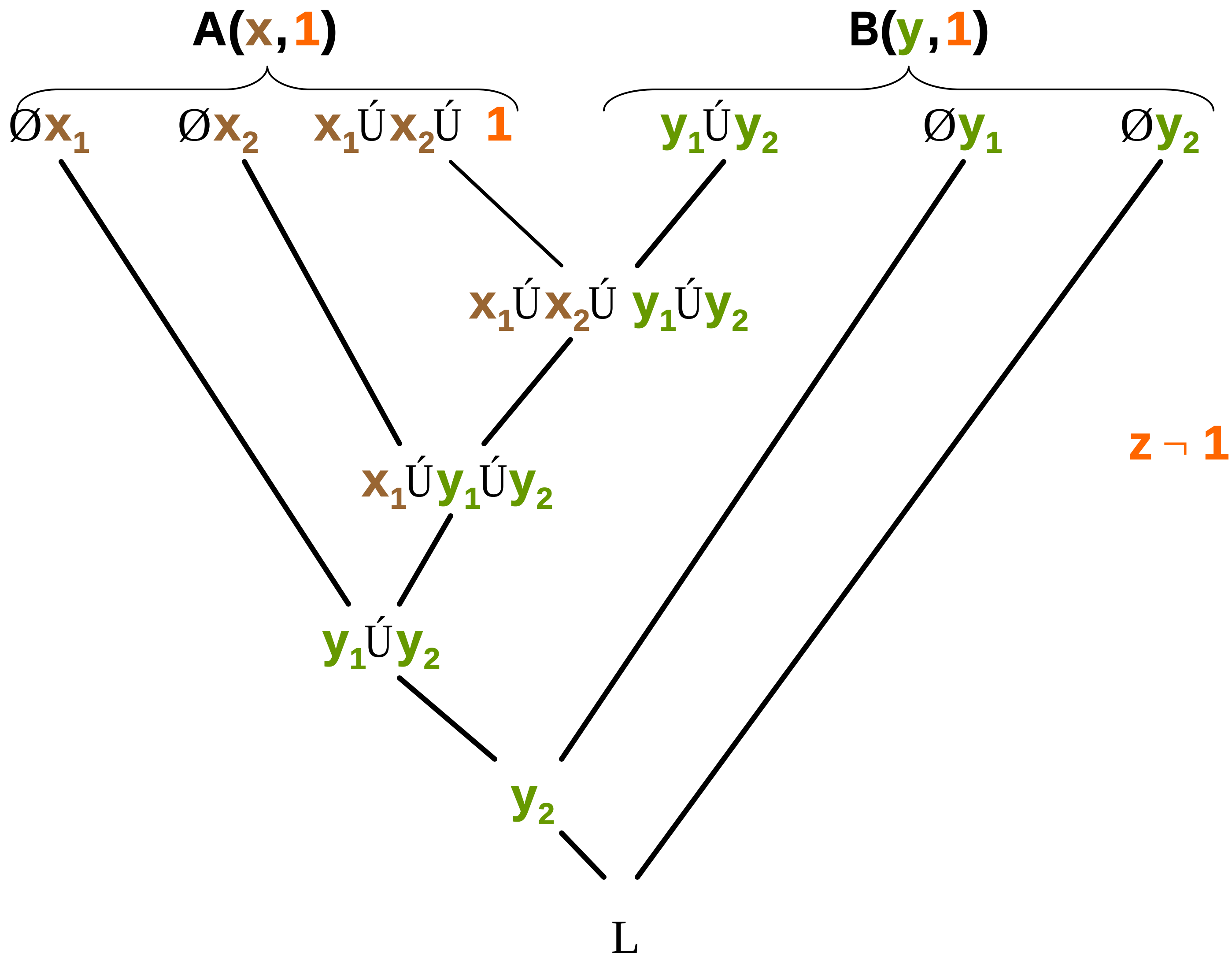
Interpolation for Resolution



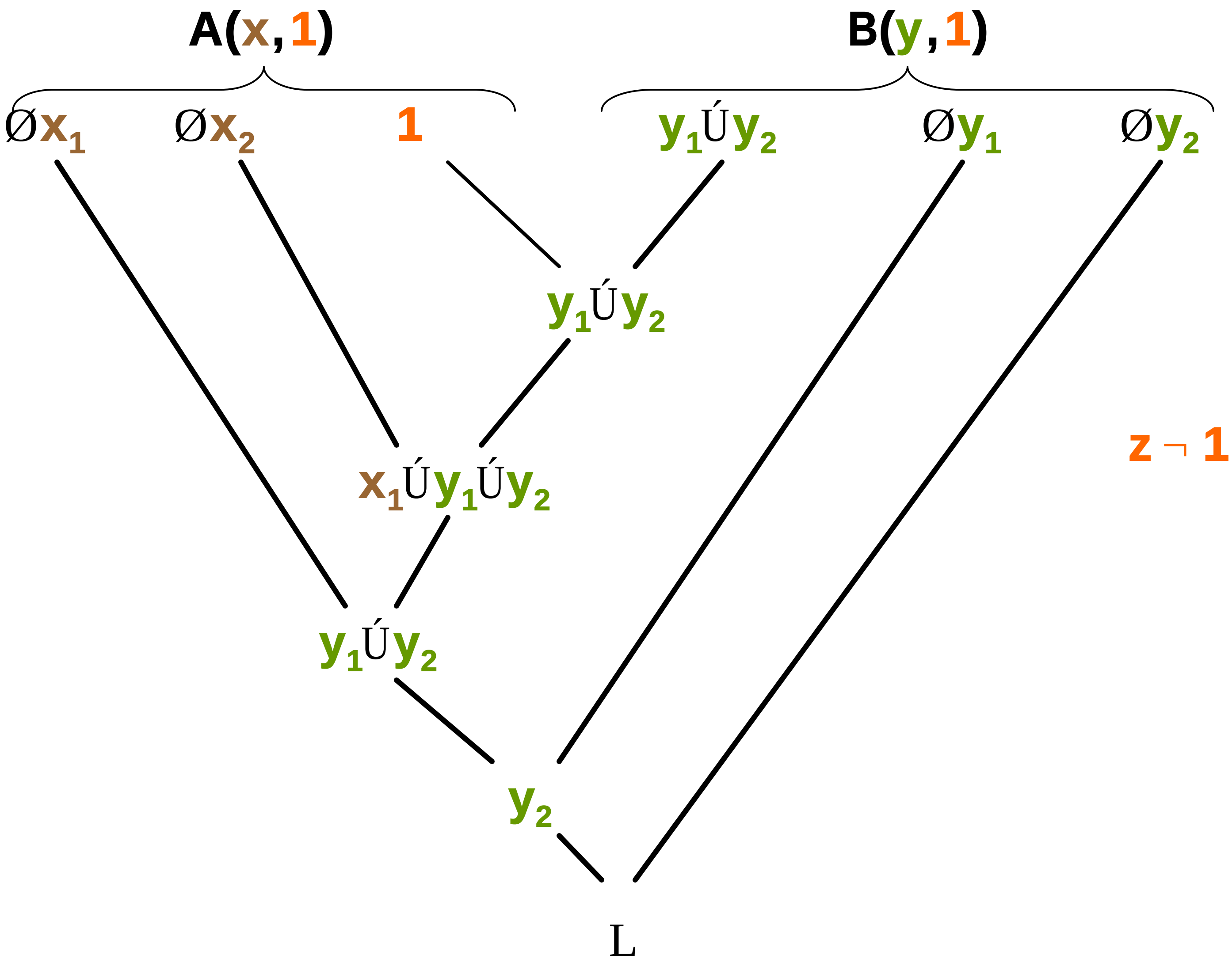
Interpolation for Resolution



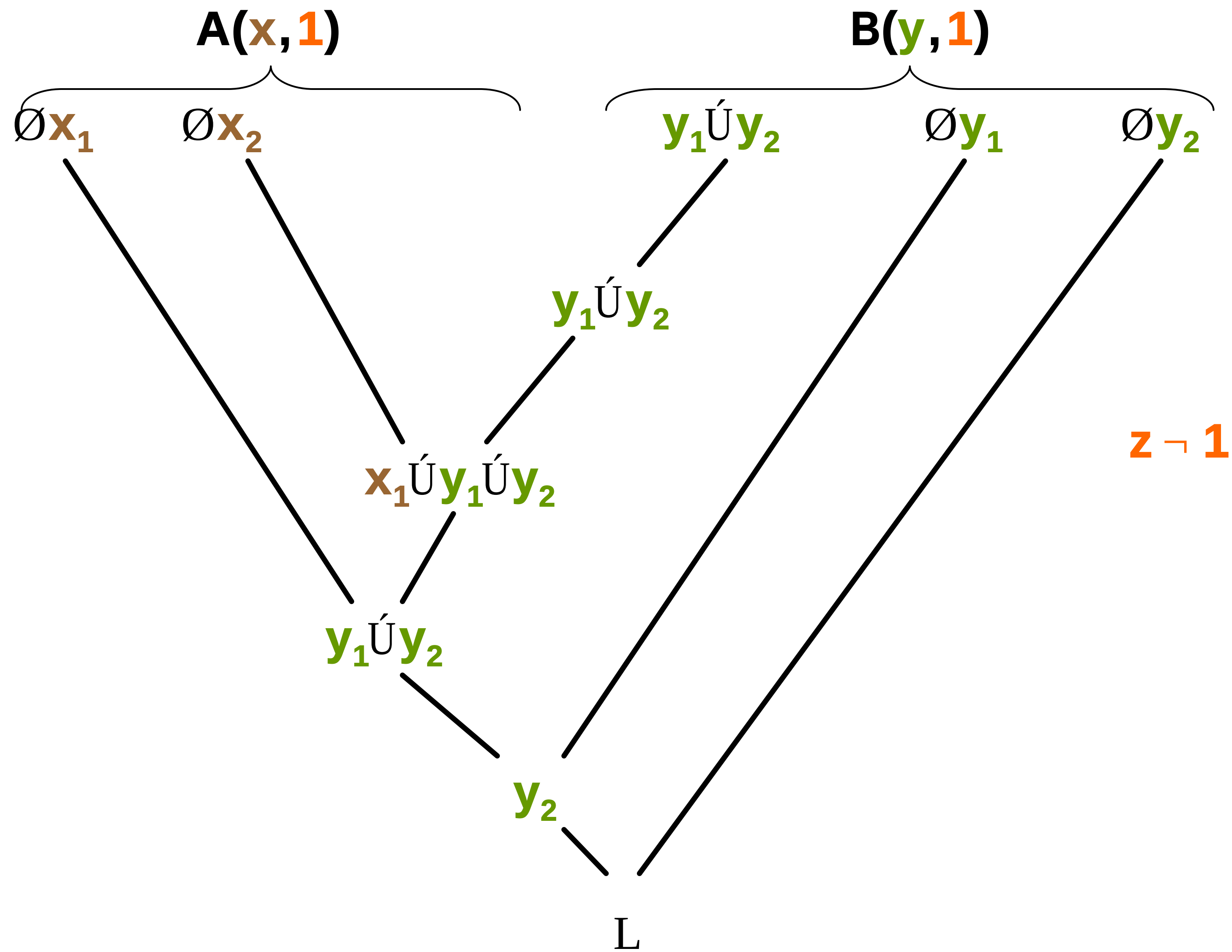
Interpolation for Resolution



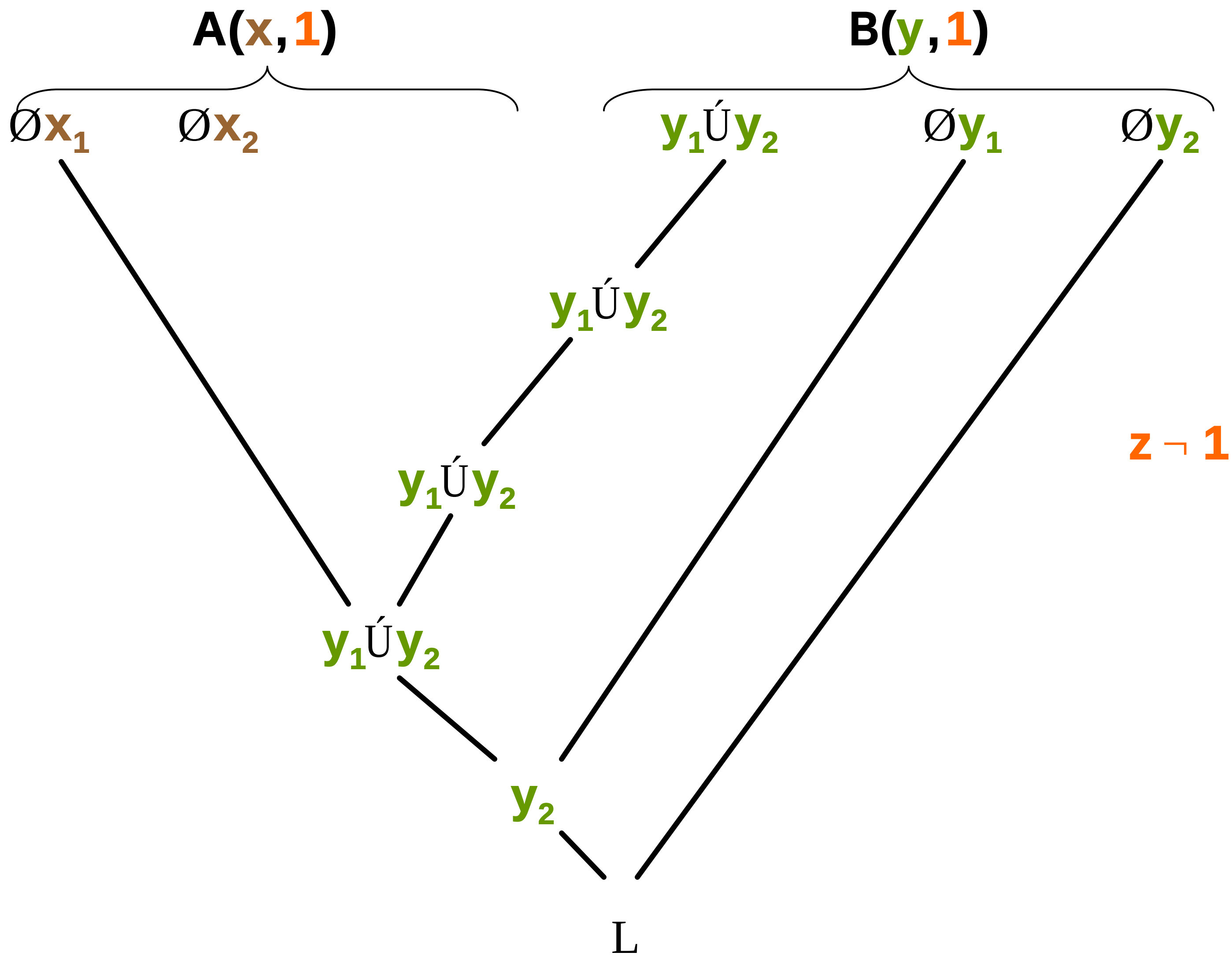
Interpolation for Resolution



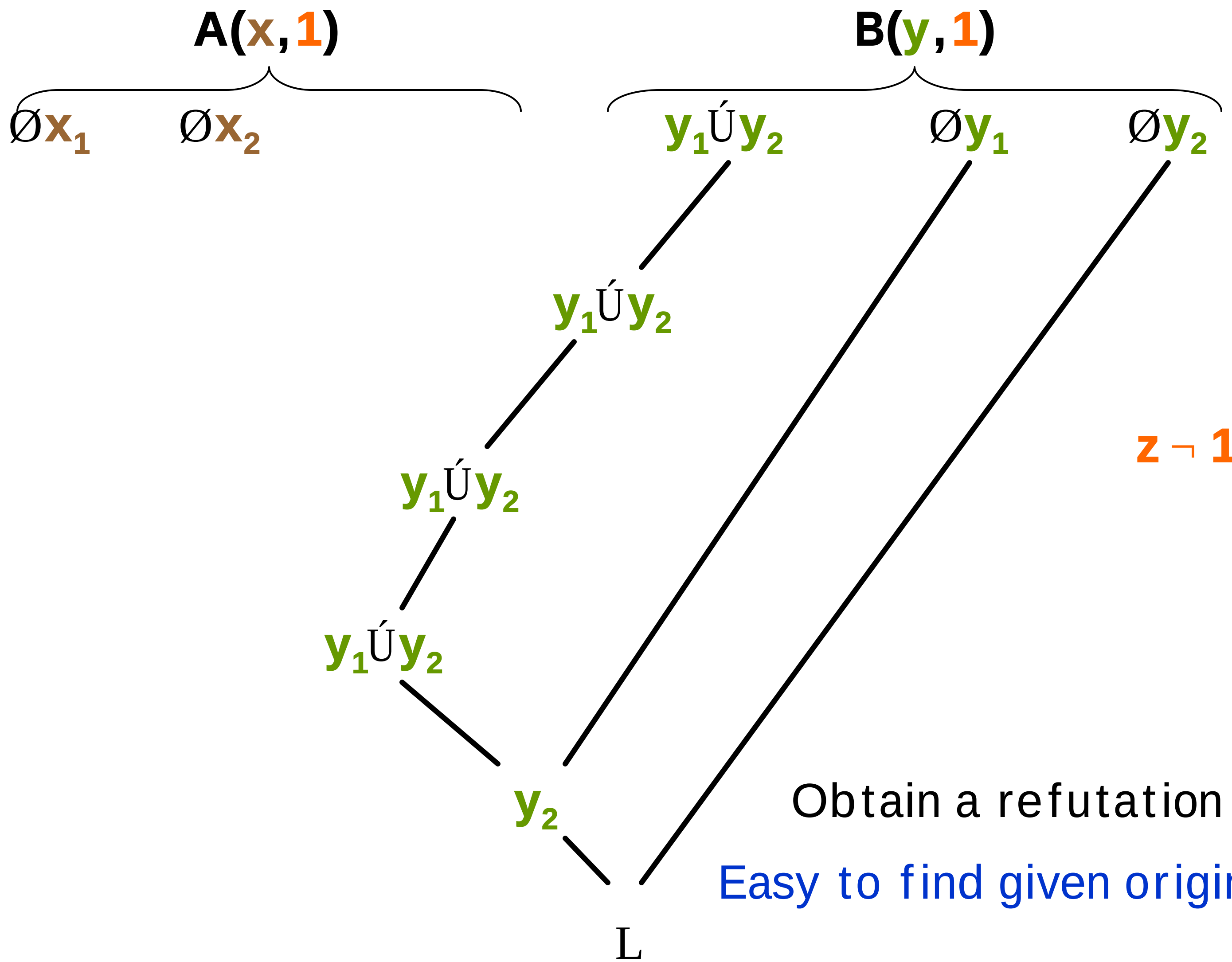
Interpolation for Resolution



Interpolation for Resolution



Interpolation for Resolution



Obtain a refutation of B(y,1)

Easy to find given original proof

Interpolation and Lower Bounds

- General proof strategy:
  - Given
    - a class of circuits for which one has lower bounds
    - a proof system whose interpolants are in the class
  - Build
    - a formula whose interpolant will be a circuit for a hard problem in the circuit class

Interpolation and Lower Bounds

- **Theorem:** If proof system **V** has feasible interpolation and **NPË P/poly** then **V** is not polynomially bounded

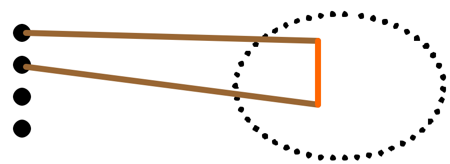
- **Theorem:** [BPR] Any proof system **V** that has monotone feasible interpolation is not polynomially bounded

Interpolation & NP vs P/poly

- **Proof sketch:** Suppose that **V** has feasible interpolation and is polynomially bounded with bound **p**
- Consider formula $\mathbf{A(x,z) \wedge B(y,z)}$ where
  - **z represents a CNF formula**
  - **A(x,z) says that assignment x satisfies z**
  - **B(y,z) says that y - of length p(|x|) - is a proof in V that z is unsatisfiable**
- Feasible interpolation for this formula will give a polysize circuit for deciding satisfiability

Clique-coloring formulas

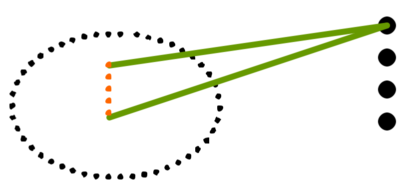
- Formula $A(x,z) \wedge B(y,z)$ where
  - z are the $n(n-1)/2$ variables representing an n node graph $G(z)$
  - $A(x,z)$ is the statement that $G(z)$ has a k-clique

k 

$(\bigvee_v x_{iv})$ $(\neg x_{iv} \vee \neg x_{ju} \vee z_{uv})$

$(\neg x_{iv} \vee \neg x_{jv})$ $(\neg x_{iu} \vee \neg x_{iv})$

  - $B(y,z)$ is the statement that $G(z)$ is $(k-1)$-colorable

 k-1

$(\bigvee_i y_{vi})$ $(\neg z_{uv} \vee \neg y_{ui} \vee \neg y_{vi})$

$(\neg y_{vi} \vee \neg y_{vj})$

Interpolation examples

- **Theorem:** [Krajicek] Resolution has feasible (monotone) interpolation.
- **Theorem:** [Pudlak 95] Cutting Planes has feasible (monotone) interpolation where the interpolants are circuits over the real numbers
  - Also extended monotone lower bounds for clique to real circuits
- **Corollary:** Any Cutting Planes proofs of clique-coloring formulas are exponential
- **Theorem:** Polynomial calculus has feasible interpolation

Limitations of Interpolation

- Theorem: [KP] If one-way functions exist then Frege systems do not have feasible interpolation.

- Theorem: [BPR, Bonet et al] If factoring Blum integers is hard then any proof system that can polynomially simulate TC^0-Frege, or even AC^0-Frege does not have feasible interpolation

Proof idea

- Suppose one has a method of **key agreement**
  - Given two people, one with **x** and another with **y**
  - via exchanging messages, they agree on a secret key **key(x,y)** so that even listening to their conversation without knowing **x** or **y** it is hard to figure out what even a bit of **key(x,y)** is
- The common variables **z** will represent the transcript of their conversation
  - **A(x,z)** will say that the player with **x** correctly computed its side of the conversation and the last bit of **key(x,y)** is **0**
  - **B(y,z)** will say that the player with **y** correctly computed its side of the conversation and the last bit of **key(x,y)** is **1**

Connections with proof systems

- Must encode the computation of each player in such a way that the proof system can prove given **x** and **z** what the value of the bit is
- Can extend **x** by helper extension variables to make the task easier
  - The actual proof uses Diffie-Hellman secret key exchange which is as hard as factoring
  - That requires powering which is not in TC^0 but the extension variables make it easy enough to prove

The Interpolation Line