Arithmetic circuits and polynomial replacement systems
(work in progress)

P. McKenzie (Montréal)
H. Vollmer (Würzburg)
K. Wagner (Würzburg)

Caveat to the Web slides

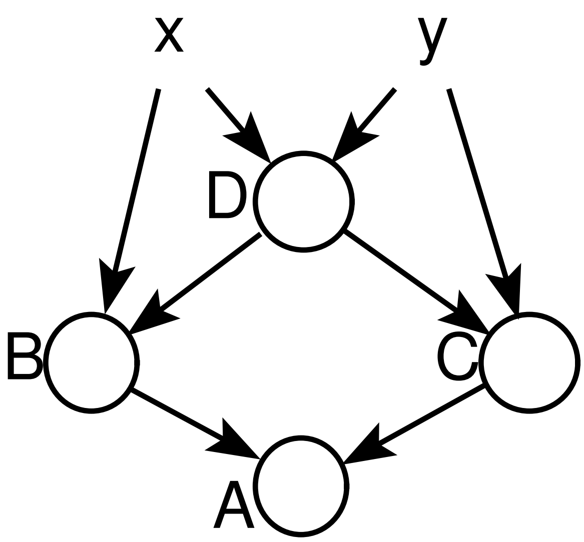
(A few slides, including motivation, are missing which I did not know how to convert from Powerpoint to Postscript, and then to include here.)

(Some overlays, and the hand-drawn colors, for example the AND and OR labels of various diagrams, are sometimes missing as well.)

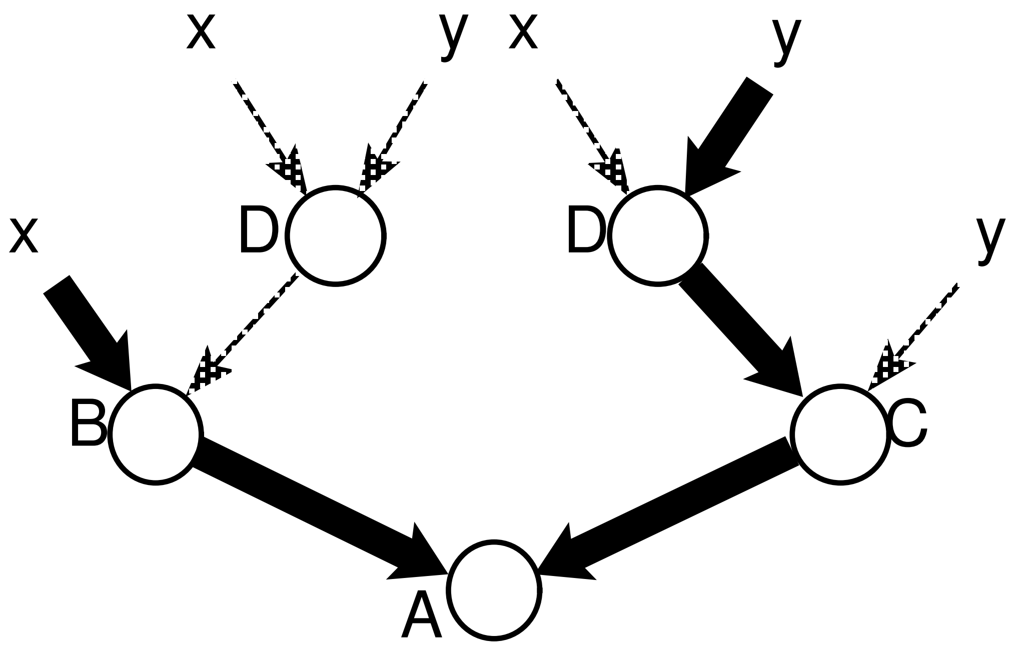
(Motivation is from #P, #L, #LOGFCL, #NC1, #AC0)

(Next circuit has AND at bottom and OR elsewhere.)

Proof trees by example

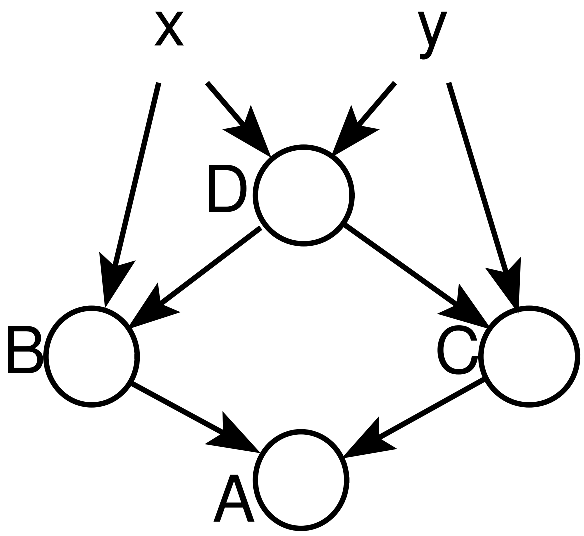


Must first unwind circuit into a tree:

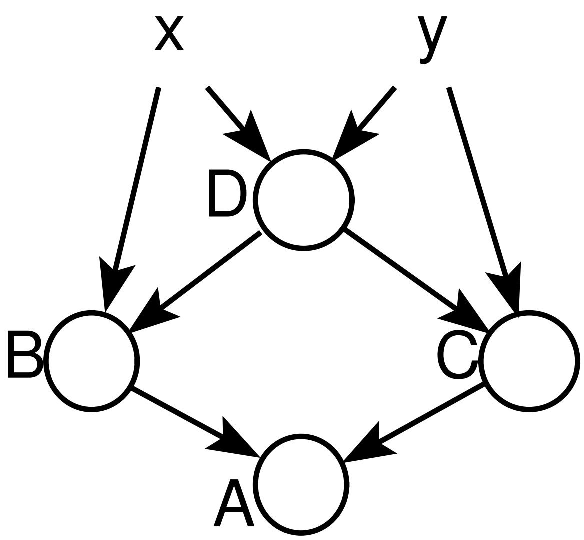


Thick arcs form a proof tree on $x = y = 1$

Known: the number of proof trees in



is given by the $\{+, \times\}$-arithmetic circuit

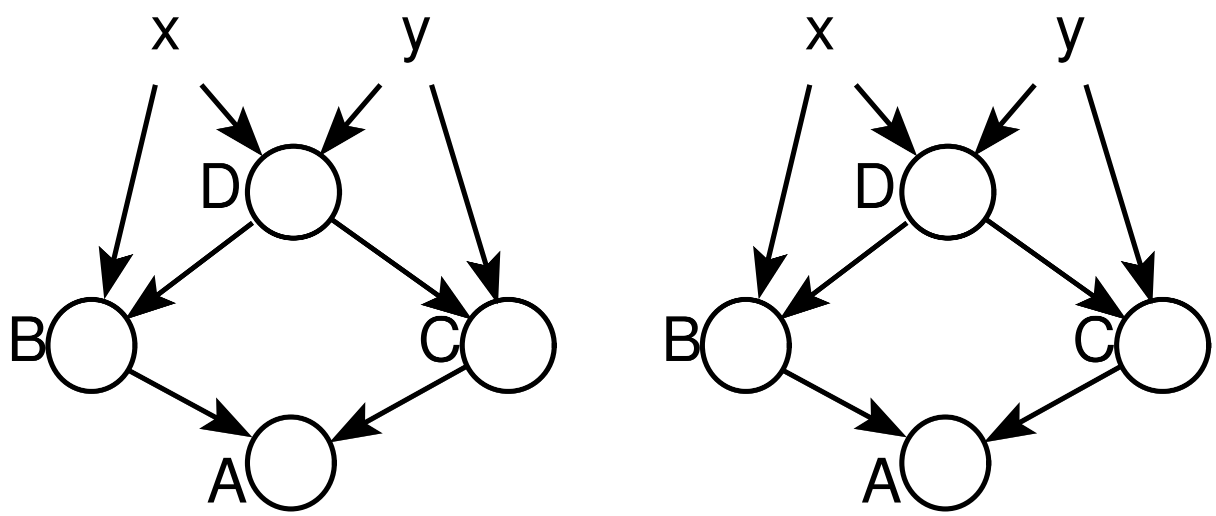


To count proof trees

- Replace $\vee$ with $+$ and $\wedge$ with $\times$
- View circuit as straight-line program
- Compute polynomial “top down”

- Evaluate program at desired input

Proof trees in our example



Straight-line program: $\begin{bmatrix} D & \to & x+y \\ C & \to & D+y \\ B & \to & x+D \\ A & \to & BC \end{bmatrix}$

Counting:

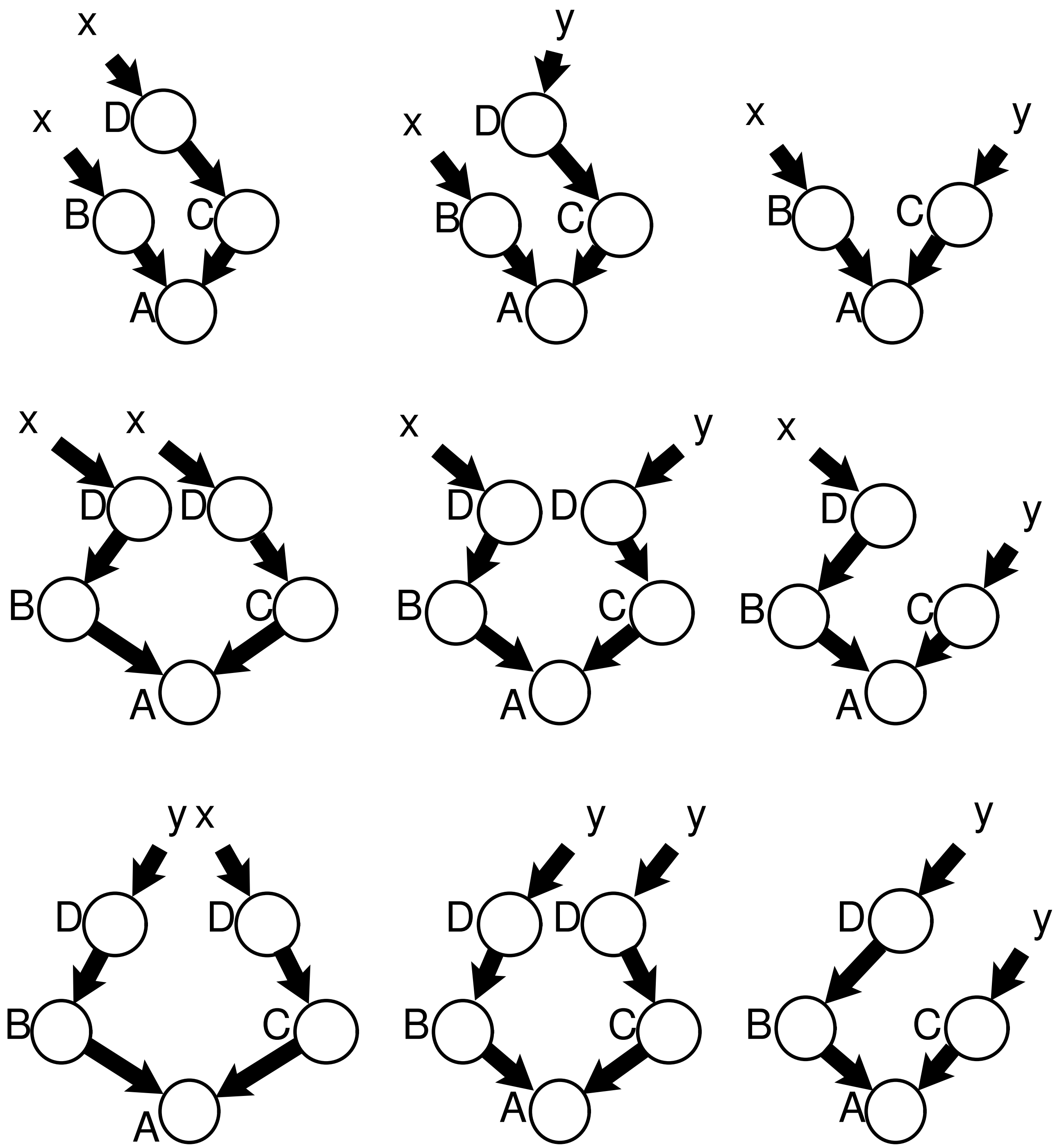
$$\begin{array}{rcl} A & \to & BC \to (x+D)C \\ & \to & (x+D)(D+y) \\ & & \\ & \to & (2x+y)(x+2y) \\ & & \\ & \to & 9 \text{ on input } x=y=1 \end{array}$$

Our starting point

What counting classes arise if we do not unwind circuits into trees before counting their proofs?

Natural notion of proof <u>circuits</u>

- how to count them?
- is their count expressible as small $\{+, \times\}$-circuits?



The 9 proof trees on input $x = y = 1$

(Overlay to explain which two proof trees amongst the 9 proof trees on the previous slide do not give rise to a valid proof circuit.)

To count proof **circuits**

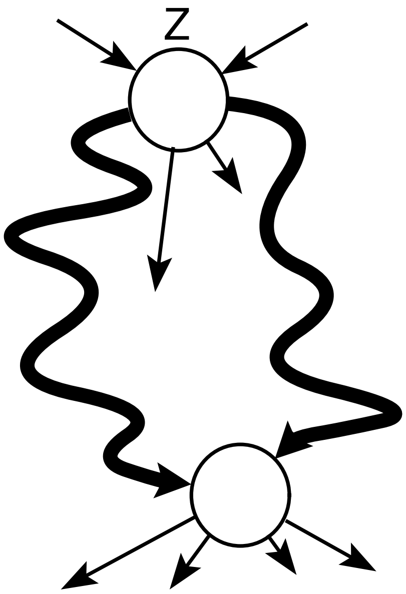
- Replace $\vee$ with $+$ and $\wedge$ with $\times$
- View circuit as straight-line program
- Compute polynomial "top down"

  but knock down any nontrivial power Z^i of a variable to Z before replacing Z

- Evaluate program at desired input

(Overlay slide 6 in order to explain how the “knock down exponents” algorithm yields 7 proof circuits in our example, as D^2 becomes D in the straight-line program evaluation.)

Why does this work?



Because Z^2 reflects two distinct paths from Z into an AND gate $\Rightarrow$ the contribution should be Z alone.

NB: replace Z only after applying $Z^2 \rightarrow Z$ to completion, and only when no descendant of Z remains.

Counting proof circuits...

...is in #P:

guess at an OR gate

recurse at an AND gate

never explore twice $\Rightarrow$ poly time.

...cannot be #P-hard under many-one polytime reduction, unless P=NP, since for any NP machine M:

$$
\begin{aligned}
x \in L(M) \ \text{ iff } \ & \#\text{ACCEPT}(M, x) > 0 \\
\text{ iff } \ & \#\text{PFCIRCUITS}(C, y) > 0 \\
\text{ iff } \ & \text{the circuit } C \text{ accepts } x
\end{aligned}
$$

for "$f_M : x \mapsto \langle C, y \rangle$" the reduction.

Theorem: Counting proof circuits is #P-complete under <u>Turing</u> reduction.

Cor 1. Our "exponents knock down" algorithm for counting proof circuits cannot work in poly time unless P=NP.

Cor 2. No poly size $\{+, \times\}$-algebraic circuit expresses in general the number of proof circuits of a boolean circuit C unless P=NP.

Proof of **2**: C has at most $2^{|C|}$ proof circuits, and evaluating a $\{+, \times\}$-circuit modulo $2^{|C|}$ could be done in P.

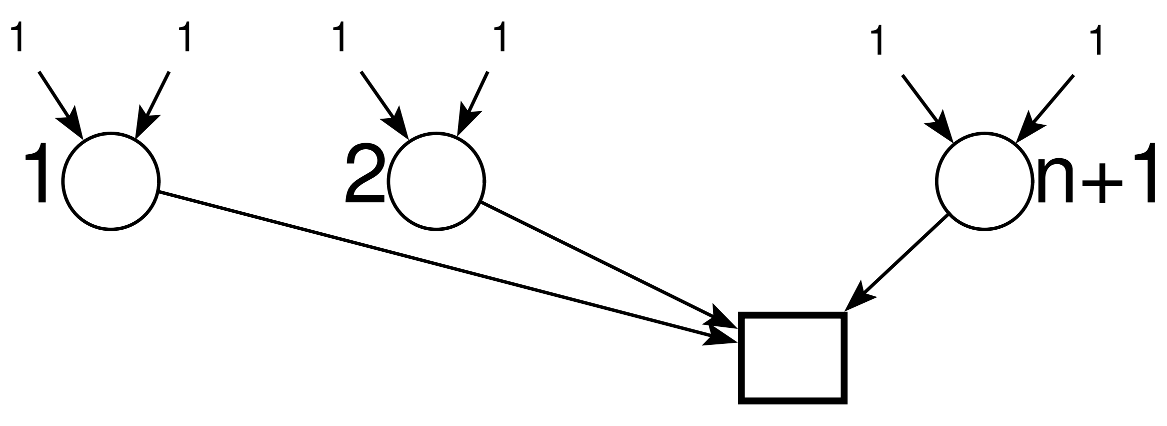
#P-hardness proof:

From Valiant's #3SAT:

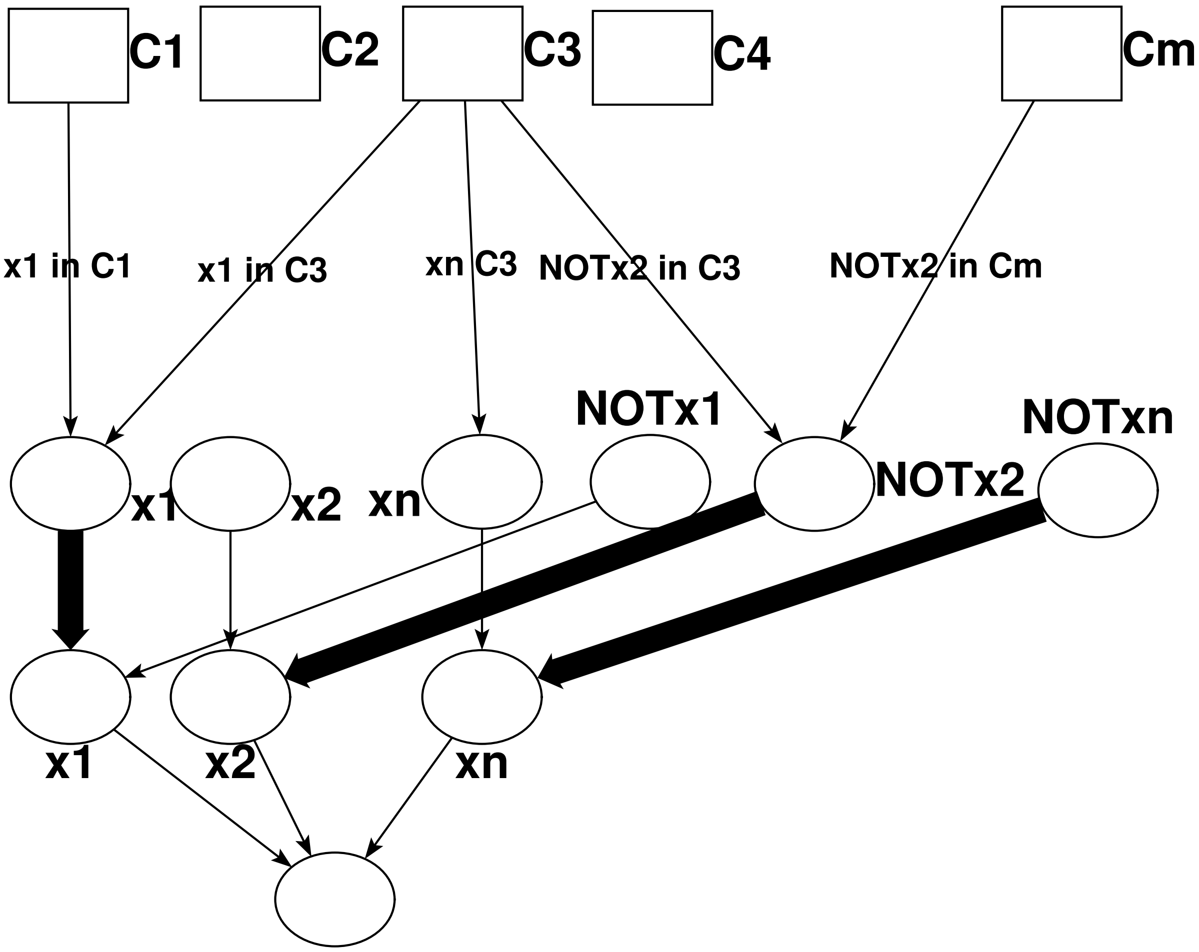
clauses $C1, C2, \ldots, Cm$,

variables $x1, x2, \ldots xn$.

Will use gadgets



which have 2^{n+1} proof circuits.



Thick is a boolean assignment.

Thick generates $2^{k\cdot(n+1)}$ proof circuits iff **thick** satisfies exactly k clauses.

Thick satisfies all 3SAT clauses $\Rightarrow$

contributes $2^{m\cdot(n+1)}$ proof circuits.

Thick $\neg$satisfy all 3SAT clauses $\Rightarrow$

contribution $\leq 2^{(m-1)\cdot(n+1)}$.

If b boolean assignments satisfy the 3SAT instance:

$$
\begin{aligned}
b \cdot 2^{m\cdot(n+1)} &\leq \#\text{proof circuits} \\
&\leq b \cdot 2^{m\cdot(n+1)} + \\
&\quad\ (2^n - b) \cdot 2^{(m-1)\cdot(n+1)} \\
&< (b+1) \cdot 2^{m\cdot(n+1)}.
\end{aligned}
$$

Our counting example: alternate viewpoint

Replacement rules: $\begin{bmatrix} D & \to & x+y \\ D^2 & \to & D \\ C & \to & D+y \\ C^2 & \to & C \\ B & \to & x+D \\ B^2 & \to & B \\ A & \to & BC \\ A^2 & \to & A \end{bmatrix}$

A yields a <u>set</u> of polynomials in $\mathbb{N}[x, y]$:
$\{(2x+y)(x+2y), x^2+3xy+x+y+y^2\}$.

Largest evaluation = # proof trees
smallest evaluation = # proof circuits.

PRS

A **polynomial replacement system** is $S = \big(\{x_1, \ldots, x_n\}, \{x_{n+1}, \ldots, x_m\}, q, R\big)$ where

- $\{x_1, \ldots, x_n\}$ are *terminal variables*,

- $\{x_{n+1}, \ldots, x_m\}$ are *nonterminal vars*,

- $q \in \mathbb{N}[x_1, \ldots, x_m]$ is *start polynomial*,

- R is a finite set of *replacement rules*, i. e. pairs of polynomials in $\mathbb{N}[x_1, \ldots, x_m]$.

How are rules applied?

Most general form:

$$p_1 \underset{S}{\Rightarrow} p_2 \quad \text{iff} \quad \begin{array}{l} \exists (p_3, p_4) \in R, \\ \exists p_5(\overline{x}, y) \in \mathbb{N}[x_1, \ldots, x_m, y] \\ \text{such that} \\ p_1(\overline{x}) = p_5(\overline{x}, p_3(\overline{x})) \text{ and} \\ p_2(\overline{x}) = p_5(\overline{x}, p_4(\overline{x})). \end{array}$$

Restricted form:

$$p_1 \underset{S}{\rightarrow} p_2 \quad \text{iff} \quad \begin{array}{l} \exists (p_3, p_4) \in R, \\ \exists p_5(\overline{x}), p_6(\overline{x}) \in \mathbb{N}[x_1, \ldots, x_m] \\ \text{such that} \\ p_1(\overline{x}) = p_5(\overline{x}) \cdot p_3(\overline{x}) + p_6(\overline{x}) \\ p_2(\overline{x}) = p_5(\overline{x}) \cdot p_4(\overline{x}) + p_6(\overline{x}). \end{array}$$

Lemma: The transitive reflexive closures of both forms are equivalent.

Of interest

PROBLEMS:

$$poly(S) = \Big\{p(x_1, \ \ldots \ , x_n) \ \Big| \ \exists p' \text{ s.t. } q \overset{*}{\underset{S}{\Rightarrow}} p' \text{ and } p(x_1, \ldots, x_n) = p'(x_1, \ldots, x_n, a_1, \ldots, a_m) \text{ for all } a_1, \ldots, a_m \in \mathbb{N}\Big\}.$$

$$range(S) = \{p(a) \mid p \in poly(S), a \in \mathbb{N}^n\}$$

$$eval(S) = \{(a, p(a)) \mid p \in poly(S), a \in \mathbb{N}^n\}.$$

PRS FAMILIES: simple, acyclic, deterministic, idempotent.

Simple and deterministic and acyclic:

$\{+, \times\}$-circuits over $\mathbb{N}$

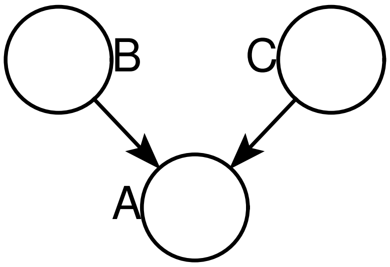
Simple and idempotent and acyclic:

"min $eval(S)$" counts proof circuits.

Simple and acyclic:

$\{\cup, +, \times\}$-circuits over $\mathbb{N}$

i.e. $eval(\cdot) \equiv_m^{\text{log}}$ $\mathbb{N}$-CIRCUIT$\{\cup, +, \times\}$

since 

simulated by $\begin{array}{ccc} A & \to & B \\ A & \to & C. \end{array}$

Note [StMe73]: $\mathbb{N}$-EXPRESSION$\{\cup, +, \times\}$ is NP-complete.

Complexity

S simple and deterministic

- $\forall S, poly(S)$ is P-complete.

In P: canonize S off line:

remove $A \overset{*}{\underset{S}{\Rightarrow}} \alpha A \beta$

remove $A \overset{*}{\underset{S}{\Rightarrow}} A + 0, A \overset{*}{\underset{S}{\Rightarrow}} 0 + A$

remove $A \overset{*}{\underset{S}{\Rightarrow}} A \times 1, A \overset{*}{\underset{S}{\Rightarrow}} 1 \times A$

Then a bounded number of equivalent SLPs

remain, since S is fixed.

Hardness: add to S a prs for CVP.

- $poly(\cdot) \in$ co-RP.

Straight-line program equivalence [IbMo83].

S simple and deterministic (cont)

- $\exists S, range(S)$ is NP-complete.

$range(\cdot) \in$ NP: remove cycles; input has n bits.

NP-hard: from the language QUADRATIC of $(a, b, c) \in \mathbb{N}^3$ s.t. $ax^2 + by = c$ for $x, y \in \mathbb{N}$. Fix

$$\langle \cdot, \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

an injective map. Then $(a, b, c) \in$ QUADRATIC

iff $\exists x, y, \langle a, b, ax^2 + by \rangle = \langle a, b, c \rangle$

iff $\exists u, v, x, y, \langle u, v, ux^2 + vy \rangle = \langle a, b, c \rangle$.

Take $\langle \alpha, \beta, \gamma \rangle = \gamma + (\beta + \gamma)^2 + (\alpha + \beta + \gamma)^3$.

- $eval(\cdot)$ is P-complete.

Note: artificial upper bound since a circuit C can have $2^{2^{|C|}}$ proof trees.

S simple and acyclic

- $\forall S, poly(S)$ is P-complete.

  In P: $poly(S)$ finite, rest as before.

- $poly[\cdot]$ NP-hard, in $\exists \cdot$ co-RP $=$ MA.

  NP-hardness: from SUM-OF-SUBSET. Rules

$$\begin{array}{rcl} x_k & \to & x_{k-1} \\ x_k & \to & x_{k-1} + b \end{array}$$

  nondeterministically add b to subset.

- $eval[\cdot]$ NP-complete.

  Again SUM-OF-SUBSET.

S simple and acyclic (cont)

- $eval(\cdot) \equiv_m^{\mathsf{log}} \mathbb{N}$-CIRCUIT$\{\cup, +, \times\}$.

Complexity: **OPEN**

$eval(\cdot) \in$ PSPACE.

Succinct $eval(\cdot)$ PSPACE-hard.

More open questions

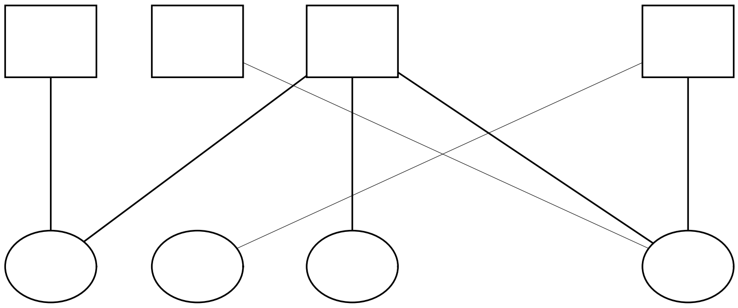
- Useful (beyond unification)?

- Expressibility as various prs? Links with Valiant's definability? Analogy with Valiant?

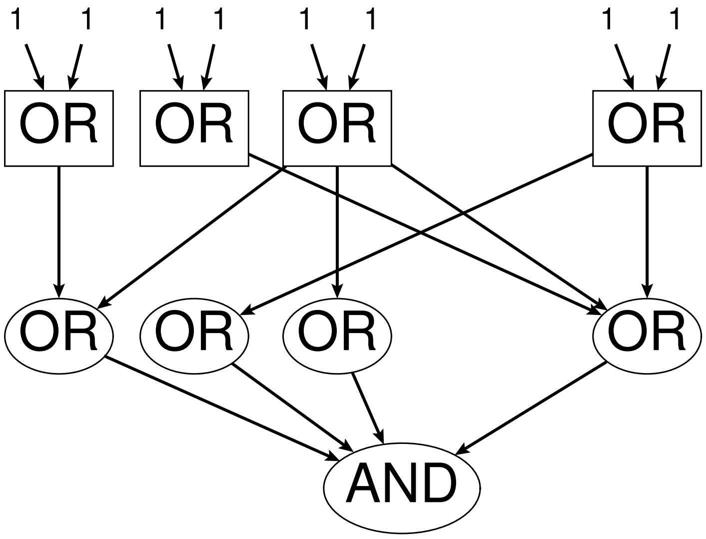
- Multivariate rules, acyclic prs, etc.

- The lurid case of depth-3 circuits versus the "$Z^2 \to 0$" rule...

The case of depth-3 circuits: details



From given $2n$-node bipartite graph, construct



The prs using rules $Z^2 \to 0$ instead of $Z^2 \to Z$ computes $2^n\cdot$ (# perfect matchings).

Yet: is computing the number of proof circuits for depth-3 circuits #P-hard?