Designing Efficient Span Programs

Robert Špalek

joint work with Ben Reichardt

Designing span programs

- **Def:** A *span program* P is: [Karchmer & Wigderson '93]
  - A *target vector* t over **C**,
  - *Input vectors* v_j each associated with a conjunction of literals from $\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$
  - Span program P computes $f_P: \{0,1\}^n \rightarrow \{0,1\}$, $f_P(x) = 1 \Leftrightarrow$ t lies in the span of { true v_j }
- **Given:** a Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$

  Task 1: design a span program P computing f
- Last talk: this gives a bounded-error quantum algorithm for f

  [Reichardt & Š, STOC'08]

Efficiency of span programs

- **Def:** *witness size* of P on input x:

length of the shortest vector satisfying certain linear constraint

$$\mathrm{wsize}_S P(x) = \begin{cases} \min_{|w\rangle : A\Pi|w\rangle = |t\rangle} \| S|w\rangle \|^2 & \text{if } f_P(x) = 1 \\ \min_{\substack{|w'\rangle : \langle t|w'\rangle = 1 \\ \Pi A^\dagger |w'\rangle = 0}} \| S A^\dagger |w'\rangle \|^2 & \text{if } f_P(x) = 0 \end{cases}$$

  - S is a diagonal matrix of complexities of the inputs (we typically consider all ones),
  - A is the matrix of input vectors of P,
  - Π is the projector onto true columns on input x
- The running time of our quantum alg. is O(max$_x$ wsize P(x))
- **Task 2:** optimize the witness size of P by tuning its "free coefficients", ideally to match the lower bound for f

  (we use *adversary lower bounds* for comparison [Ambainis'03])

Span programs based on formulas

Using DNF formulas

- Expand the function f as a DNF formula

  Ex: $\text{Equal}_n(x_1 \ldots x_n) = (x_1 \wedge \ldots \wedge x_n) \vee (\neg x_1 \wedge \ldots \wedge \neg x_n)$

- Form a span program with 1 row of ones, and with columns corresponding to the clauses, labeled by the conjunctions of the input variables and their negations

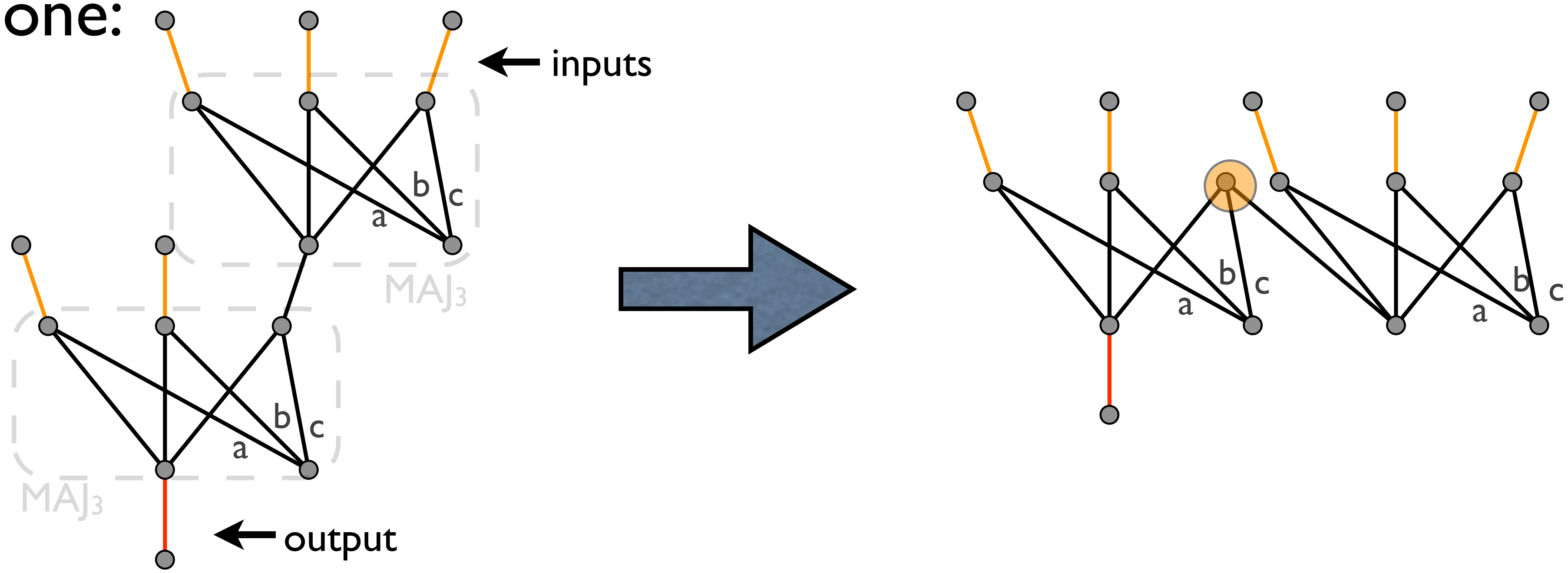
target	$x_1 \ldots x_n$	$\neg x_1 \ldots \neg x_n$
1	1	1

  - By definition, this span program evaluates f
  - It may not be efficient, but one can optimize its weights.

    This way, for example, one gets an optimal $O(\sqrt{n})$ algorithm for {AND, OR} formulas of size n
    [Ambainis, Childs, Reichardt, Š & Zhang, FOCS'07]

Formulas that are not in DNF

- If the formula consists of more complicated sub-formulas, we can use composition of their corresponding span programs
- Connect the output edge of one gadget with an input edge of another one:



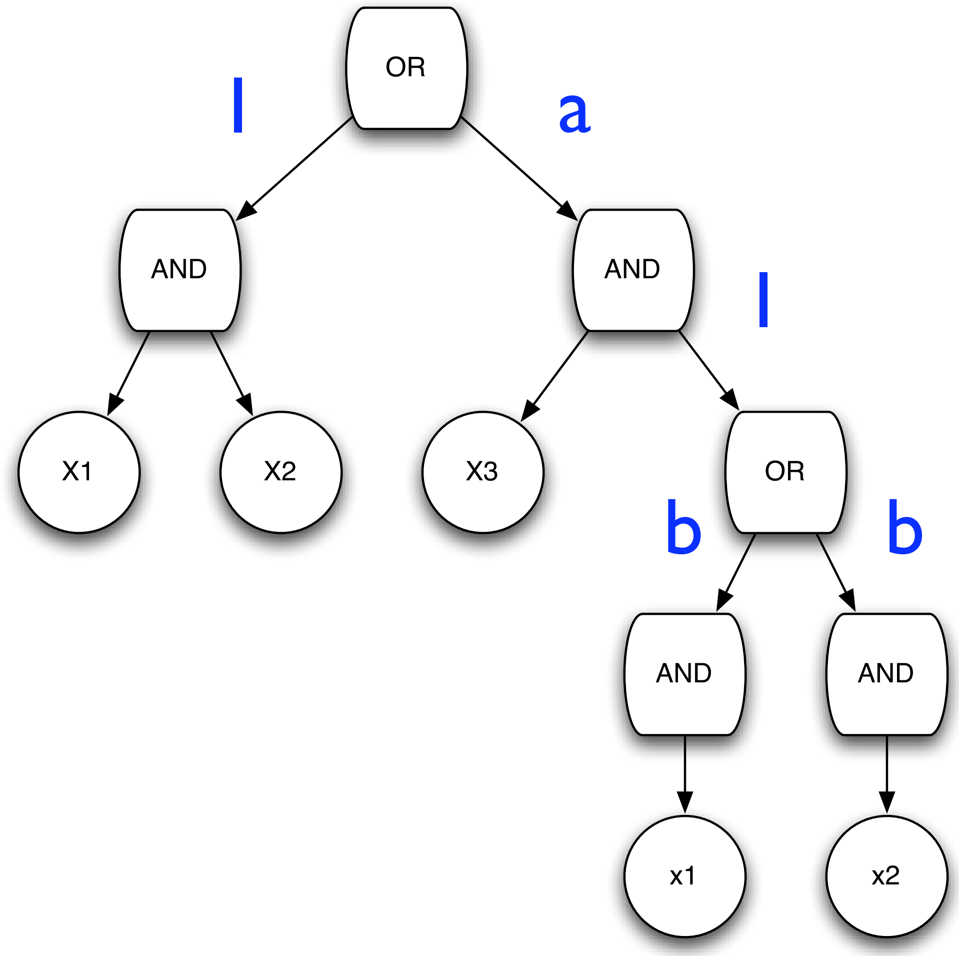
- The span program then looks like this:

t	x_1	x_2	1	x_3	x_4	x_5
1	1	1	1	0	0	0
0	a	b	c	0	0	0
0	0	0	1	1	1	1
0	0	0	0	a	b	c

Running example: Majority of 3

- $\mathrm{Maj}_3(x_1,x_2,x_3) = (x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$
- Since this is not in a DNF form, we need to compose two span programs, one for $(x_1 \wedge x_2)$ and one for $((x_1 \vee x_2) \wedge x_3)$.
- If we set the weights according to [ACRŠZ'07], we get witness size $\sqrt{5}$.
  - These weights would be optimal if the formula was a **read-once** formula on bits, i.e. $(x_1 \wedge x_2) \vee ((x_4 \vee x_5) \wedge x_3)$.

    However, we are promised that $x_4=x_1$ and $x_5=x_2$, so some inputs don't appear.
- If we optimize the weights under this promise, the witness size is $\sqrt{(3+\sqrt{2})} < \sqrt{5}$.



x_1x_2	x_3	x_1	x_2
1	a	0	0
0	1	b	b

$$a = \sqrt{\frac{1+\sqrt{2}}{2}} \qquad b = \frac{1}{\sqrt[4]{2}}$$

Going beyond formulas

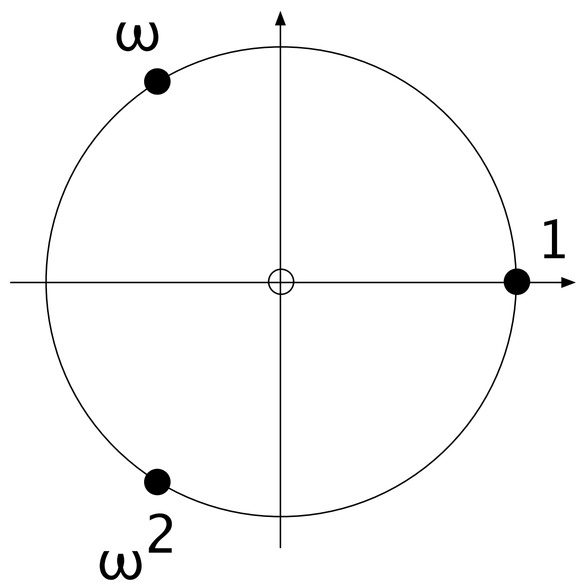
and respecting function symmetries

Symmetric span program for Maj3

- Maj3 is a *symmetric function*: one can permute the input bits any way and the function value stays the same. However, the span program doesn't treat the input bits symmetrically.
- Consider span program:

target	x_1	x_2	x_3
1	1	1	1
0	1	ω	ω^2

$$\omega = e^{2\pi i/3}$$



- If any two input bits are one, we get the same true column vectors $(1, 1), (1, \omega)$, modulo a unitary transform. The program is symmetric with respect to all minimal 1-inputs.
- Simple computation shows that witness size of this span program is 2, matching the adversary bound, and hence it is optimal

Automorphisms of Maj_3

- *Automorphism* = combination of a permutation and bit-flip of some input bits, preserving the function value

  Automorphism group = group of all automorphisms
- The automorphism group of Maj_3 is $\mathbf{S_3}$.
  - Since we are permuting bits which can only have one of two values {0,1}, in each input string there will always exist two bits of the same value (pigeonhole principle).
  - We don't care whether these two equal bits are swapped ⇒

    we can apply an extra transposition to each permutation σ from S_3 to make σ **even**, with no effect to the input.
  - Therefore the "*symmetry group*" of our interest isn't S_3, but $\mathbf{A_3=Z_3}$, i.e. one can get to any input string with the same number of ones by using just rotation.

Using representation theory

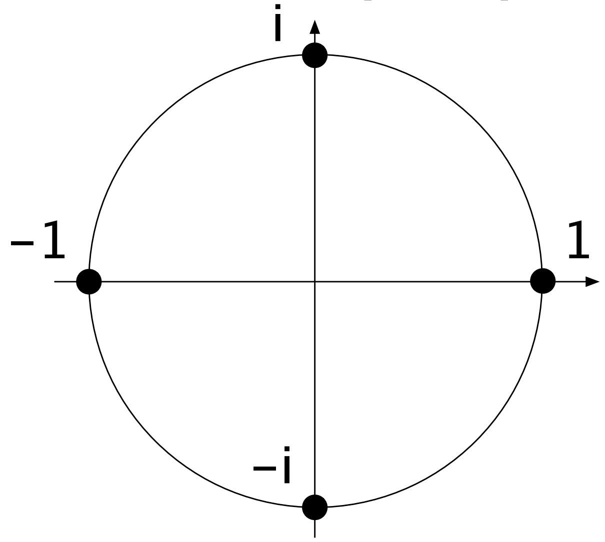
- The optimal span program for Maj_3 is

target	x_1	x_2	x_3
1	1	1	1
0	1	ω	ω^2

- Note that it consists of two *representations of* Z_3, 1 and ω^i, one in each row
- **Q:** Is this a coincidence or can we find a similar pattern for more interesting functions?

Threshold 2 of 4

- Generalization of Maj_3 to more input bits
- A natural span program is



target	x_1	x_2	x_3	x_4
1	1	1	1	1
0	1	i	-1	-i

- However, this span program is not symmetric, because the sub-matrix corresponding to input 1100 is very different from the sub-matrix for input 1010.

  Indeed, its witness size is $\sqrt{8} > \sqrt{6}$

- Need to go to **higher dimensions** to achieve symmetry

Using different rings than C

- A complex number a+bi can be though of as a pair of real numbers (a, b), or as a 2x2 matrix

a	-b
b	a

  This matrix representation preserves summation and multiplication.

- One can represent a complex span program by a real one with the same witness size. For example, for Maj_3:

target	x_1	x_2	x_3
1	1	1	1
0	1	ω	ω^2

→

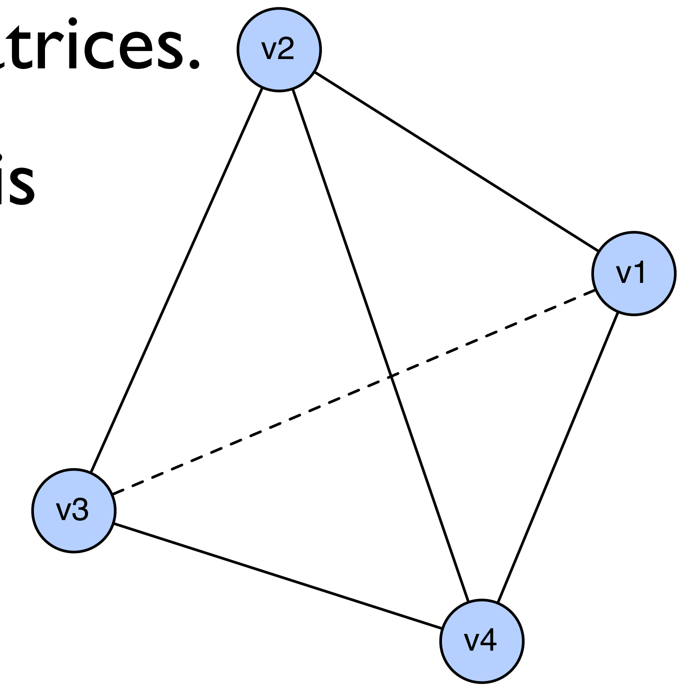
target	x_1	x_1	x_2	x_2	x_3	x_3
1	1	1	1	1	1	1
0	1	0	-1/2	$-\sqrt{3}/2$	-1/2	$\sqrt{3}/2$
0	0	1	$\sqrt{3}/2$	-1/2	$-\sqrt{3}/2$	-1/2

- Can we do the opposite, i.e. extend **C**? Yes.

Simplex with 4 vertices

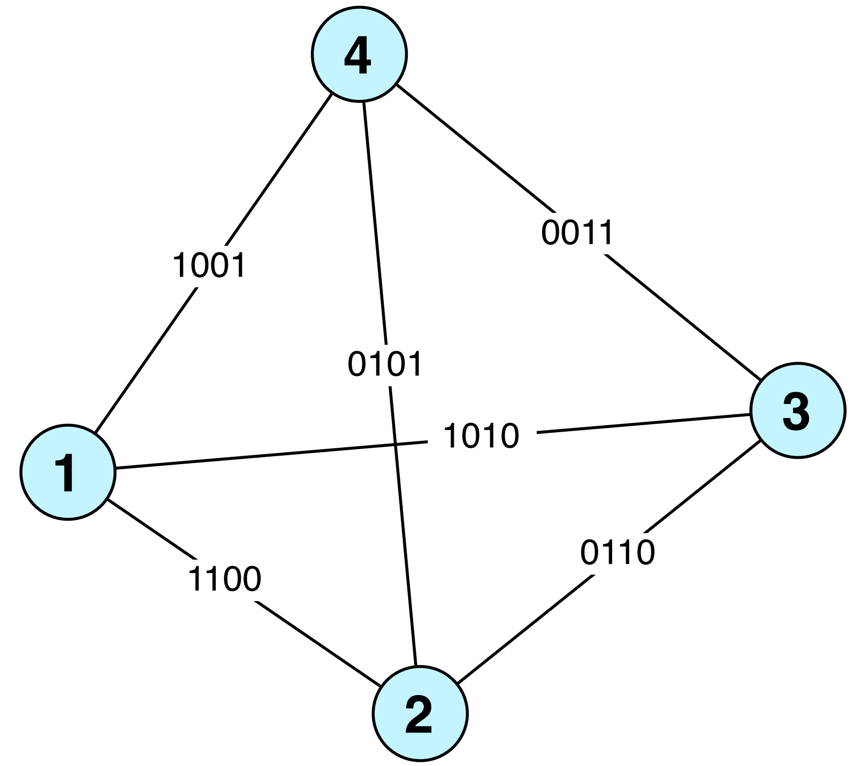
- In 3 dimensions, we can take 4 vertices of the regular tetrahedron. All its edges are symmetric to each other.
- How to embed such a simplex into a span program?
  - We go from the ring **C** of complex numbers into an extension ring of quaternions/unitary matrices.
  - In this extension ring, the span program is

target	x_1	x_2	x_3	x_4
1	1	1	1	1
0	v_1	v_2	v_3	v_4



where v1, v2, v3, v4 are elements of the extension ring corresponding to the vertices of the simplex.

Constructing the extension ring



- The automorphism group of $Thr_{2\text{ of }4}$ is S_4, and we can again restrict our attention to $\mathbf{A_4}$, the subgroup of even permutations
- This is the group of orientation-preserving *symmetries of tetrahedron*. Its generators are:
  - Define a clock-wise rotation “around” each vertex, labeled by the fixed input bit.
  - Represent the rotations in SO(3) ⊆ SU(2).

  The elements $v_{1..4}$ of the span program will be the 2x2 representations of these rotations.

optimal wsize = √6

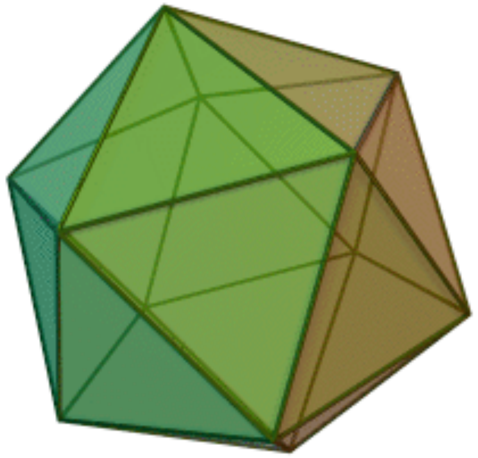
target	x_1	x_1	x_2	x_2	x_3	x_3	x_4	x_4
1	1	1	1	1	1	1	1	1
0	1	1	1	-1	i	-i	i	i
0	i	-i	i	i	1	1	1	-1

2x2 unitary matrix

Threshold 2 of 5

- Similarly to $\mathrm{Thr}_{2\text{ of }4}$, one can use the vertices of a regular 5-simplex in 4 dimensions.
- They can also be represented by 2x2 complex matrices and give an optimal span program with witness size √8, of the form

target	x_1	x_2	x_3	x_4	x_5
1	1	1	1	1	1
0	v_1	v_2	v_3	v_4	v_5



- The symmetry group of $\mathrm{Thr}_{2\text{ of }5}$ is $\mathbf{A_5}$, which is the group of orientation-preserving *symmetries of the icosahedron*. It can also be embedded into SO(3) ⊆ SU(2).
- **Q:** Can we think of the 2x2 matrices corresponding to the 5 vertices of the simplex as generators of this group?

$\text{Threshold}_{M \text{ of } N}$

- Span programs for M=2 and N≥5:
  - **Q:** Can we analyze them in a closed form?
- Span programs for M=3 and N=5:
  - **Q:** Can we get a symmetric span program of the form

target	x_1	x_2	x_3	x_4	x_5
1	1	1	1	1	1
0	v_1	v_2	v_3	v_4	v_5
0	w_1	w_2	w_3	w_4	w_5

    where v_i, w_i are 2x2 unitary matrices? What would the pairs of matrices correspond to on the icosahedron?
- **Q:** Can we solve in a closed form the general case?
- **Q:** Do we actually need the smaller "symmetry group" $\mathbf{A_n}$ instead of $\mathbf{S_n}$? It arises very naturally for Maj_3.

Going beyond permutation symmetry

Automorphism groups with flipping input bits

Ambainis function

- **Def:** $A(x_1,x_2,x_3,x_4)$ is true iff the input bits are sorted, i.e.

  $x_1 \le x_2 \le x_3 \le x_4$ or $x_1 \ge x_2 \ge x_3 \ge x_4$

- Its automorphism group is generated by

  $A(x_1,x_2,x_3,x_4) = A(x_2,x_3,x_4,\neg x_1)$

  i.e. one can rotate the whole string to the left while flipping the new last bit

- Simplest non-monotone function with unknown witness size.

  $\deg(A)=2$, $\mathrm{Adv}(A)=2.5$, $\mathrm{Adv}^{\pm}(A)=2.51353$

  $\mathrm{wsize}(A) \le 2.77394$ using optimized weights for formula
  $(x_1 \wedge ((x_2 \wedge x_3) \vee (\neg x_3 \wedge \neg x_4))) \vee (\neg x_1 \wedge ((\neg x_2 \wedge \neg x_3) \vee (x_3 \wedge x_4)))$

0- and 1- inputs of the Ambainis function

- If we order both x_i's and $\neg x_i$'s in one line, then the "symmetry group" of this function is $\mathbf{Z_8}$
- **Idea:** it may be worthy to look at span programs consisting of representations of Z_8.
  - If a span program is generated this way, then it is sufficient to verify just 2 input strings 0000 and 0010. All other inputs are unitarily related due to the rotational symmetry.
- Cannot make it work yet due to some "little" details

	x_1	x_2	x_3	x_4	$\neg x_1$	$\neg x_2$	$\neg x_3$	$\neg x_4$
0000					✔	✔	✔	✔
0001				✔	✔	✔	✔	
0011			✔	✔	✔	✔		
0111		✔	✔	✔	✔			
1111	✔	✔	✔	✔				
1110	✔	✔	✔					✔
1100	✔	✔					✔	✔
1000	✔					✔	✔	✔
0010			✔		✔	✔		✔
0101		✔		✔	✔		✔	
1011	✔		✔	✔		✔		
0110		✔	✔		✔			✔
1101	✔	✔		✔			✔	
1010	✔		✔			✔		✔
0100		✔			✔		✔	✔
1001	✔			✔		✔	✔	

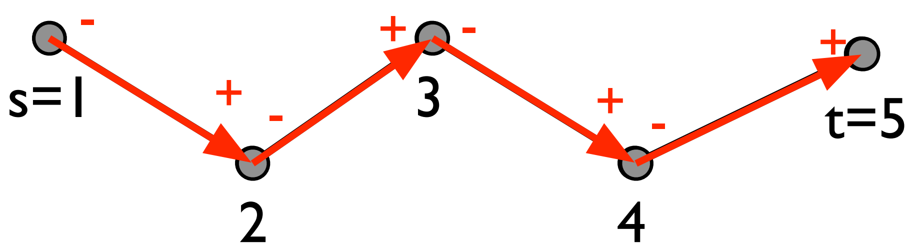
Non-constant size span programs

Examples of large span programs

- **Ex:** Undirected s-t connectivity
  - G a graph with marked source and sink vertices
  - $|t\rangle = |\text{sink}\rangle - |\text{source}\rangle$

    $|v_{(i,j)}\rangle = |i\rangle - |j\rangle$

t	$e_{(1,2)}$	$e_{(2,3)}$	$e_{(3,4)}$	$e_{(4,5)}$	$e_{(1,3)}$	$e_{(1,4)}$	$\cdots$
-1	-1	0	0	0	-1	-1	
0	1	-1	0	0	0	0	
0	0	1	-1	0	1	0	$\cdots$
0	0	0	1	-1	0	1	
1	0	0	0	1	0	0	

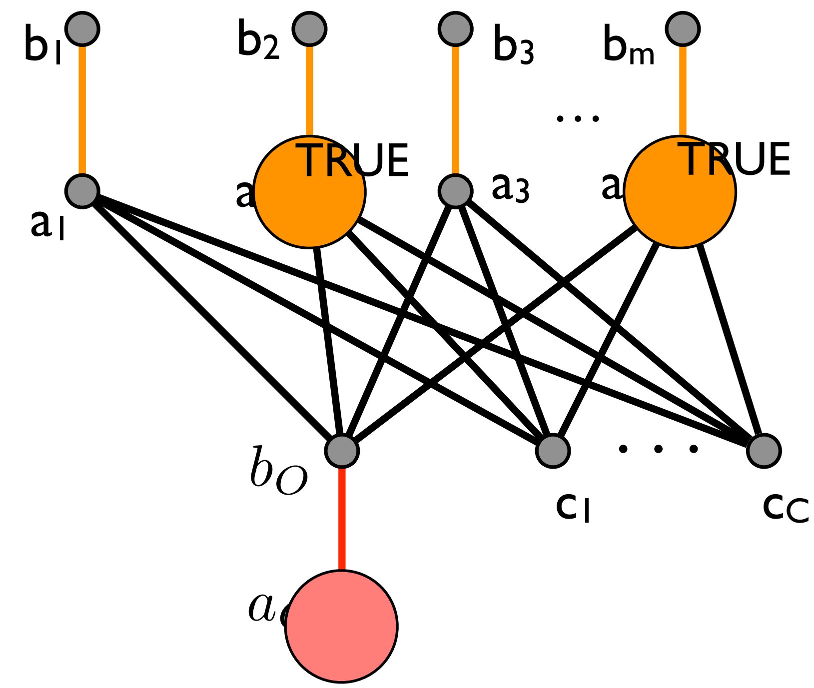


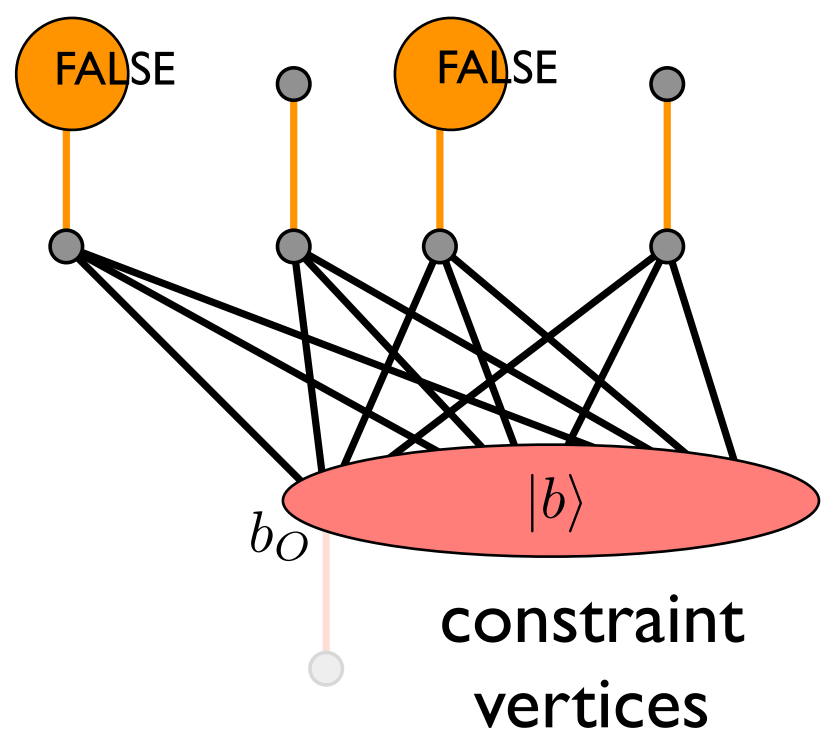
Linear combinations of the edge vectors correspond to paths in the graph.

Large overhead of span programs

- On true inputs, the span program has negligible *overhead*, because all amplitude of 0-eigenvalue eigenvectors is put on the true inputs and on the output vertex
- **Problem:** On false inputs, the span program puts amplitude also on its *constraint vertices*, and hence it decreases the overlap of the eigenvector with the root vertex.

  This is negligible for span programs with O(1) constraints, but may not be for larger ones.





The trouble of unbalanced inputs

3-bit functions

Fully solved for balanced inputs

Gate	Adversary lower bound
0	0
x_1	1
$x_1 \wedge x_2$	$\sqrt{2}$
$x_1 \oplus x_2$	2
$x_1 \wedge x_2 \wedge x_3$	$\sqrt{3}$
$x_1 \oplus x_2 \oplus x_3$	3
$x_1 \oplus (x_2 \wedge x_3)$	$1 + \sqrt{2}$
$x_1 \vee (x_2 \wedge x_3)$	$\sqrt{3}$
$(x_1 \wedge x_2) \vee (\overline{x_1} \wedge x_3)$	2
$x_1 \vee (x_2 \wedge x_3) \vee (\overline{x_2} \wedge \overline{x_3})$	$\sqrt{5}$
$\mathrm{MAJ}_3(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$	2
$\mathrm{MAJ}_3(x_1, x_2, x_3) \vee (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3})$	$\sqrt{7}$
$\mathrm{EQUAL}(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3})$	$3/\sqrt{2}$
$(x_1 \wedge x_2 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_2})$	$\sqrt{3 + \sqrt{3}}$

The highlighted functions require span programs.
The rest follows from composition.
http://www.ucw.cz/~robert/papers/gadgets/

Partially unbalanced inputs

- Assume the input complexities of 2 inputs are equal and the third one is arbitrary

  Ex: $\mathrm{Maj}_3(x_1, x_2, y_1 \vee y_2)$

- Maj_3 is relatively easy to compute and is optimal

target	x_1	x_2	x_3
1	α	α	$\sqrt{(1/2+\beta\alpha^2)}$
0	i	$-i$	2α

$$\beta = \frac{\mathrm{wsize}(x_3)}{\mathrm{wsize}(x_1)}$$

$$\alpha = \frac{1}{2\sqrt{2}}\sqrt{\sqrt{8+\beta^2}-\beta}$$

$$\mathrm{wsize}(P) = \frac{1}{2}(\sqrt{8+\beta^2}+\beta)\mathrm{wsize}(x_1)$$

- Equal_3 needs to be solved separately in 3 intervals of β. One gets 3 different expressions that smoothly connect. The solution is optimal, too.

$$\{\beta+\sqrt{2-\beta^2}, \qquad \sqrt{\frac{3}{2}(2+\beta^2)}, \qquad 1+\beta\}$$

$$\sqrt{2/5} \qquad\qquad 2$$

Partially unbalanced inputs

- **Def:** If-then-else: $(x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$
- If wsize(x_2)=wsize(x_3), then there exists a natural optimal span program with witness size wsize(x_1) + wsize(x_2)
- Otherwise one can easily prove obvious bounds on wsize
  - upper bound wsize(x_1) + max(wsize(x_2), wsize(x_3))
  - lower bound wsize(x_1) + min(wsize(x_2), wsize(x_3))
- For input complexities $(1, 1, \sqrt{2})$, we only know

min	Adv	Adv±	wsize	max
2	2.18398	2.20814 < 2.22833		2.41421

Completely unbalanced inputs

- **Ex:** $\mathrm{Maj}_3(x_1,\ x_2 \vee x_3,\ x_4 \oplus x_5)$ or, more generally,

  $\mathrm{Maj}_3(x_1, x_2, x_3)$ with input complexities $(1, \alpha, \beta)$
- We don't know their witness size
  - The exact solution of the adversary bound leads to a cubic polynomial.
  - We don't have a candidate solution of the span program yet.

Analyzing candidate span programs

Software packages for computing the witness size and the adversary bound

Adversary bound

- Matlab program using SeDuMi finds the optimal adversary bound using semidefinite programming

  http://www.ucw.cz/~robert/papers/adv/

  - Works with non-Boolean inputs and promise functions
  - **Input:** list of 0- and 1-inputs, and the input complexities
  - **Output:** Adv and $\mathrm{Adv}^{\pm}$, and the adversary matrices
- **Ex:** Maj_3
  - X = [0, 0, 0; 0, 0, 1; 0, 1, 0; 1, 0, 0]
    Y = [0, 1, 1; 1, 0, 1; 1, 1, 0; 1, 1, 1]
    costs = [1, 2, 3]
    [Gamma, D] = madv(X,Y, costs)
    Gamma1 = Gamma .* mat(D(:,1))
    norm(Gamma)/norm(Gamma1)

Witness size

- Mathematica module computes the witness size of a given span program. It can also perform limited optimization of the program.

- **Input:**
  span program template
  - target vector
  - input vectors, possibly with some coefficients as free variables
  - (conjunctions of) input bits corresponding to the vectors

- **Output:**
  - which function is computed by this program
  - the witness size for each input
  - the best assignment of the weights.
    Both symbolical and numerical optimization are available (the latter being much faster).

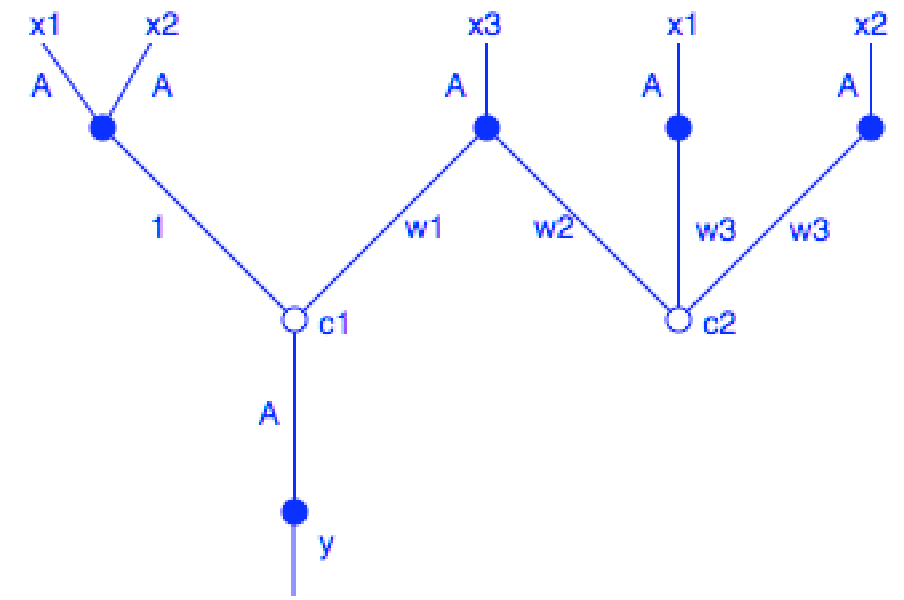
Witness size

Ex: Maj3 using NAND formula

```
n = 3;
inputs = {{y}, {x1, x2}, {x3}, {x1}, {x2}};
m = {{A, 1, w1, 0, 0},
     {0, 0, w2, w3, w3}};

rout = computeSolution[m, inputs];
evaluations = evaluateInputs[];
evaluations // MatrixForm

optimizeWeights[evaluations, {w1, w2, w3}]
```



Function computed is 831

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ \frac{w2^2+2\,(1+2\,w1^2)\,w3^2}{2\,A^2\,(w2^2+2\,w3^2)} & \frac{w2^2+4\,w1^2\,w3^2}{2\,A^2\,w2^2} & \frac{1+w1^2}{A^2} & \frac{A^2\,(w2^2+w3^2)}{w1^2\,w3^2} & \frac{1+w1^2}{A^2} & \frac{A^2\,(w2^2+w3^2)}{w1^2\,w3^2} & 2\,A^2 & \frac{2\,A^2\,(w2^2+2\,w3^2)}{w2^2+2\,(1+2\,w1^2)\,w3^2} \end{pmatrix}$$

$$\left\{\left\{\sqrt{3+\sqrt{2}},\ 2.10100298961545\right\},\ \left\{w1 \to -\sqrt{\frac{1}{2}\left(1+\sqrt{2}\right)},\ w2 \to 1,\ w3 \to \frac{1}{2^{1/4}}\right\}\right\}$$

Summary

1. Span programs design
   - based on formulas
   - based on the "symmetry group"
     - using extension rings
     - using representation theory
   - symmetry group with allowed bit flips
2. Non-constant size span programs
3. Trouble with unbalanced inputs
4. Our software packages

Q&A