Quantum algorithms through composition of graphs

Arjan Cornelissen[1]

[1]Simons Institute, University of California, Berkeley, California

October 8th, 2025

Quantum algorithms

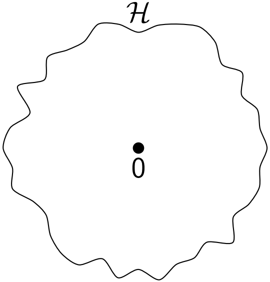
Quantum algorithms

Quantum algorithm:

Quantum algorithms

Quantum algorithm:

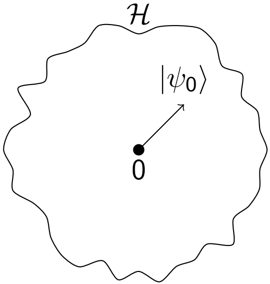
1. *State space:* Hilbert space $\mathcal{H}$.



Quantum algorithms

Quantum algorithm:

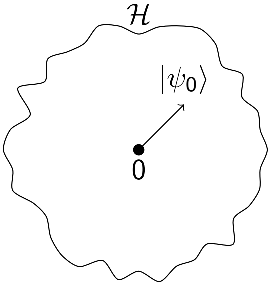
1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.



Quantum algorithms

Quantum algorithm:

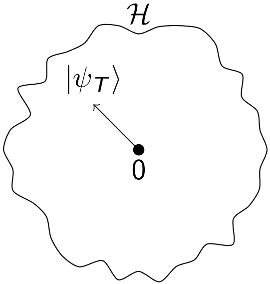
1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.
3. *Quantum operations:*
   unitaries $U_1, \dots, U_T \in \mathcal{L}(\mathcal{H})$.



Quantum algorithms

Quantum algorithm:

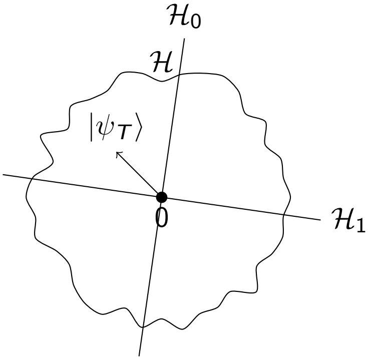
1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.
3. *Quantum operations:*
   unitaries $U_1, \dots, U_T \in \mathcal{L}(\mathcal{H})$.



Quantum algorithms

Quantum algorithm:

1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.
3. *Quantum operations:*
   unitaries $U_1, \dots, U_T \in \mathcal{L}(\mathcal{H})$.
4. *Quantum measurement:* $\{o_1, \dots o_n\}$
   $\mathcal{H}_{o_1}, \dots, \mathcal{H}_{o_n} \subseteq \mathcal{H} : \bigoplus_o \mathcal{H}_o = \mathcal{H}$.
   $\mathbb{P}[\text{output } o] = \|\Pi_{\mathcal{H}_o} U_T \cdots U_1 |\psi_0\rangle\|^2$.



$$\mathbb{P}[\text{output } o] = \|\Pi_{\mathcal{H}_o} U_T \cdots U_5 U_4 U_3 U_2 U_1 |\psi_0\rangle\|^2.$$

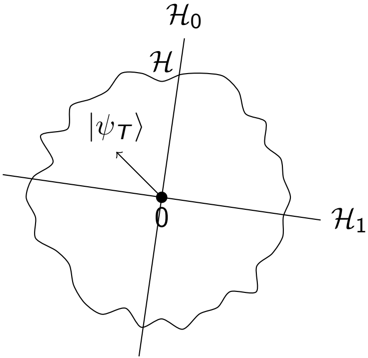
Quantum algorithms

Quantum algorithm:

1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.
3. *Quantum operations:*
   unitaries $U_1, \dots, U_T \in \mathcal{L}(\mathcal{H})$.
4. *Quantum measurement:* $\{o_1, \dots o_n\}$
   $\mathcal{H}_{o_1}, \dots, \mathcal{H}_{o_n} \subseteq \mathcal{H} : \bigoplus_o \mathcal{H}_o = \mathcal{H}$.
   $\mathbb{P}[\text{output } o] = \|\Pi_{\mathcal{H}_o} U_T \cdots U_1 |\psi_0\rangle\|^2$.

Quantum query algorithm: $f : \mathcal{D} \to \Sigma$.

1. *Input oracle:* $\forall x \in \mathcal{D}$, unitary $O_x \in \mathcal{L}(\mathcal{H})$.
2. *Success probability:*
   $\forall x \in \mathcal{D}, \mathbb{P}[\text{output } f(x)] \geq 2/3$.



$\mathbb{P}[\text{output } o] =$
$\|\Pi_{\mathcal{H}_o} U_T \cdots U_5 U_4 U_3 U_2 U_1 |\psi_0\rangle\|^2$.

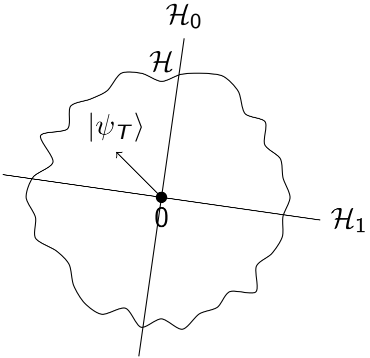
Quantum algorithms

Quantum algorithm:

1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.
3. *Quantum operations:*
   unitaries $U_1, \dots, U_T \in \mathcal{L}(\mathcal{H})$.
4. *Quantum measurement:* $\{o_1, \dots o_n\}$
   $\mathcal{H}_{o_1}, \dots, \mathcal{H}_{o_n} \subseteq \mathcal{H} : \bigoplus_o \mathcal{H}_o = \mathcal{H}$.
   $\mathbb{P}[\text{output } o] = \|\Pi_{\mathcal{H}_o} U_T \cdots U_1 |\psi_0\rangle\|^2$.

Quantum query algorithm: $f : \mathcal{D} \to \Sigma$.

1. *Input oracle:* $\forall x \in \mathcal{D}$, unitary $O_x \in \mathcal{L}(\mathcal{H})$.
2. *Success probability:*
   $\forall x \in \mathcal{D}, \mathbb{P}[\text{output } f(x)] \geq 2/3$.

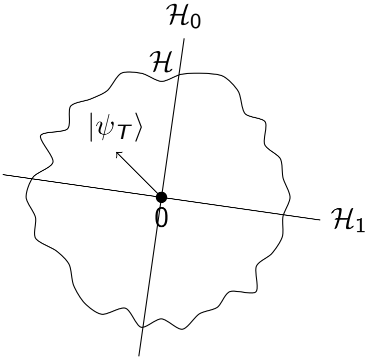


$\mathbb{P}[\text{output } o] =$
$\|\Pi_{\mathcal{H}_o} U_T \cdots U_5 O_x U_3 O_x U_1 |\psi_0\rangle\|^2$.

Quantum algorithms

Quantum algorithm:

1. *State space:* Hilbert space $\mathcal{H}$.
2. *Initial state:* $|\psi_0\rangle \in \mathcal{H}$, $\||\psi_0\rangle\| = 1$.
3. *Quantum operations:*
   unitaries $U_1, \dots, U_T \in \mathcal{L}(\mathcal{H})$.
4. *Quantum measurement:* $\{o_1, \dots o_n\}$
   $\mathcal{H}_{o_1}, \dots, \mathcal{H}_{o_n} \subseteq \mathcal{H} : \bigoplus_o \mathcal{H}_o = \mathcal{H}$.
   $\mathbb{P}[\text{output } o] = \|\Pi_{\mathcal{H}_o} U_T \cdots U_1 |\psi_0\rangle\|^2$.



$\mathbb{P}[\text{output } o] =$
$\|\Pi_{\mathcal{H}_o} U_T \cdots U_5 O_x U_3 O_x U_1 |\psi_0\rangle\|^2$.

Quantum query algorithm: $f : \mathcal{D} \to \Sigma$.

1. *Input oracle:* $\forall x \in \mathcal{D}$, unitary $O_x \in \mathcal{L}(\mathcal{H})$.
2. *Success probability:*
   $\forall x \in \mathcal{D}, \mathbb{P}[\text{output } f(x)] \geq 2/3$.

Quantum query complexity: $\mathrm{Q}(f; O_x)$:

1. Minimum number of oracle calls.

Quantum algorithmic frameworks (for boolean functions)

Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm
for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.

Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.

$\boxed{\text{Quantum algorithm}}$

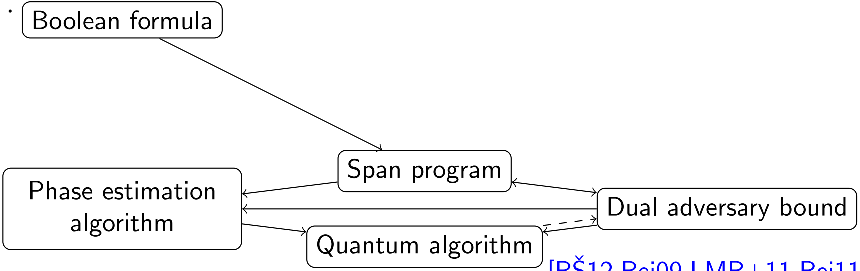
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[RŠ12,Rei09,LMR+11,Rei11]

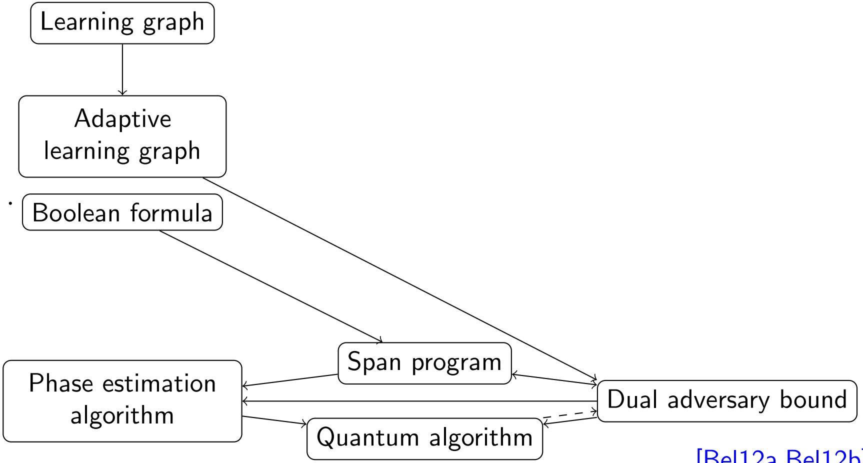
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[Bel12a,Bel12b]

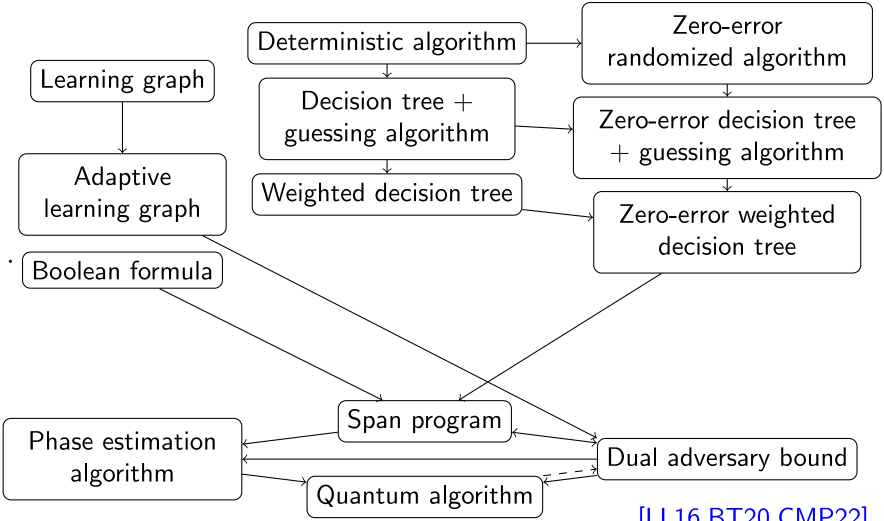
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[LL16,BT20,CMP22]

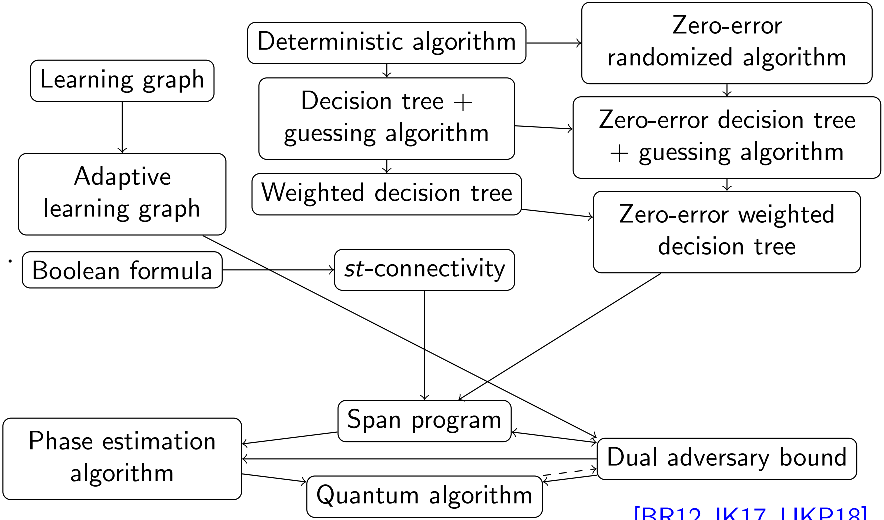
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[BR12,JK17,JJKP18]

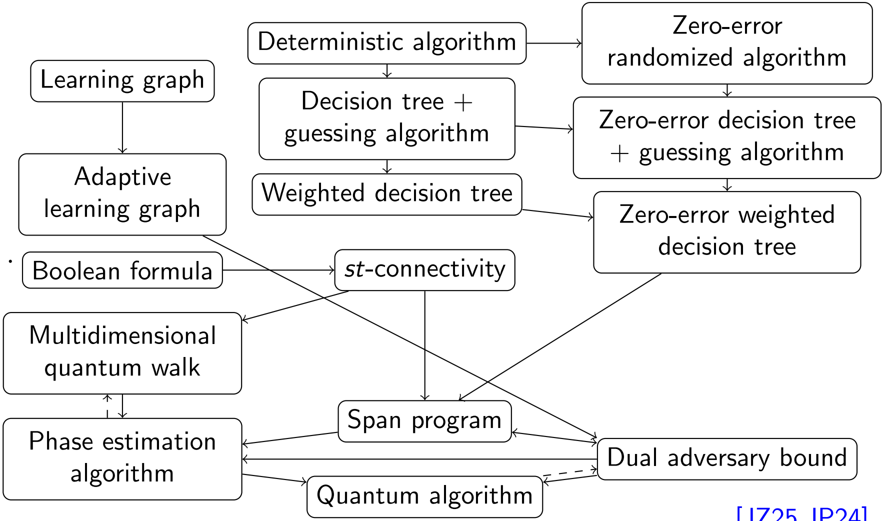
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[JZ25,JP24]

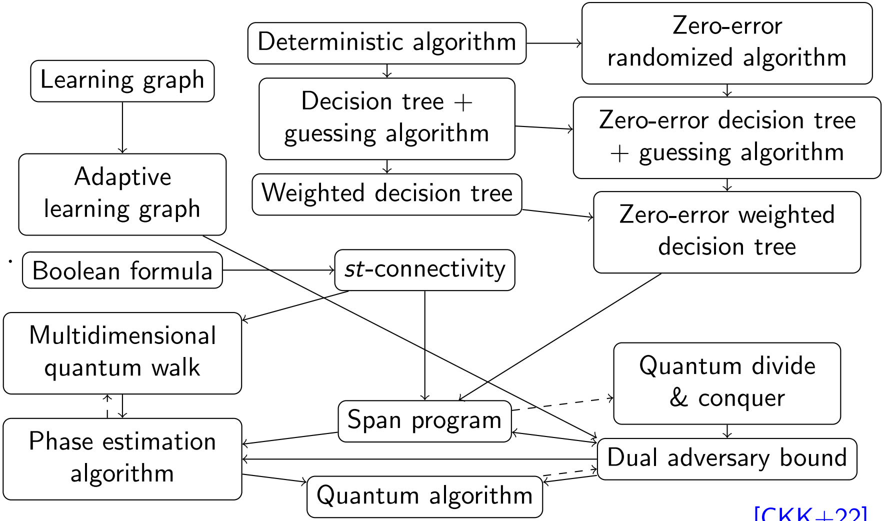
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[CKK+22]

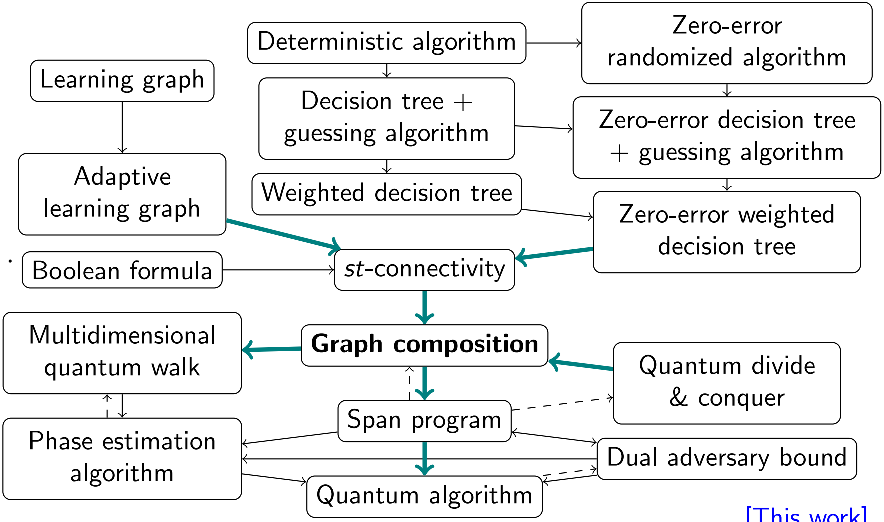
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

1. Define object L.
2. Convert object into quantum algorithm.



[This work]

Span programs [RŠ12,Rei09,Rei11]

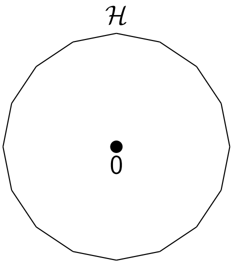
Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

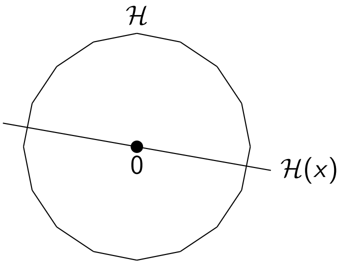
1. *Hilbert space:* $\mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

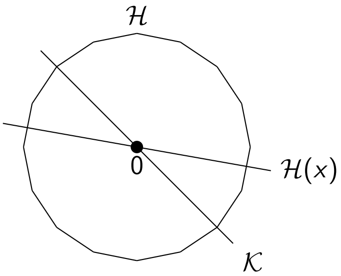
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

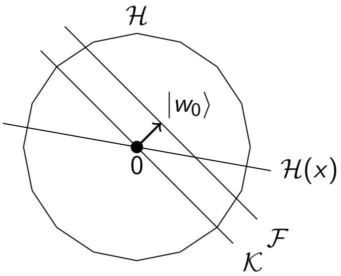
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.



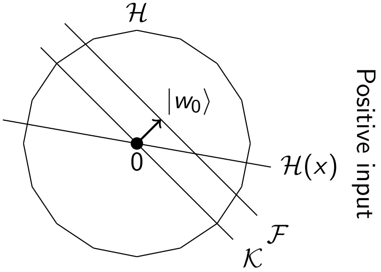
Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Positive vs. negative inputs:

1. $f : \mathcal{D} \to \{0,1\}$, $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.



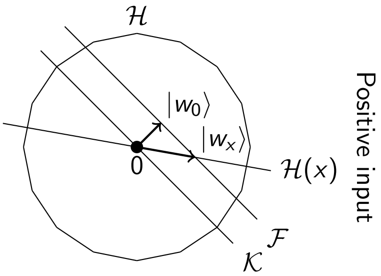
Span programs [RŠ12,Rei09,Rei11]

Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Positive vs. negative inputs:

1. $f : \mathcal{D} \to \{0,1\}$, $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. $w_+(x,\mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{F} \cap \mathcal{H}(x)\}$.



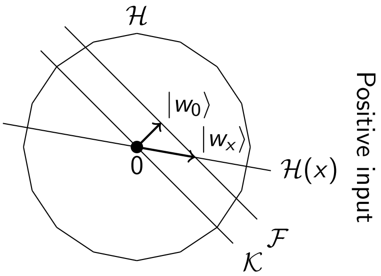
Span programs [RŠ12,Rei09,Rei11]

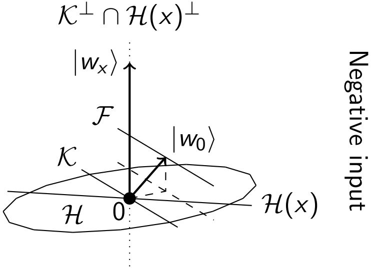
Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Positive vs. negative inputs:

1. $f : \mathcal{D} \to \{0,1\}$, $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. $w_+(x, \mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{F} \cap \mathcal{H}(x)\}$.
3. $w_-(x, \mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w_0|w\rangle = 1\}$.





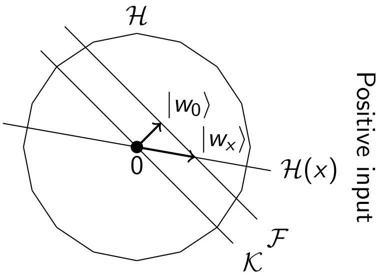
Span programs [RŠ12,Rei09,Rei11]

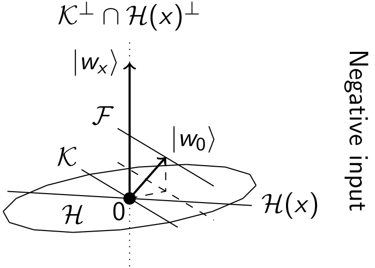
Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Positive vs. negative inputs:

1. $f : \mathcal{D} \to \{0,1\}$, $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. $w_+(x,\mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{F} \cap \mathcal{H}(x)\}$.
3. $w_-(x,\mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w_0|w\rangle = 1\}$.
4. $C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}$.





Span programs [RŠ12,Rei09,Rei11]

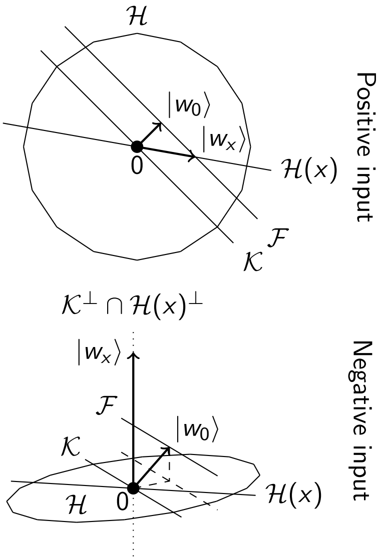
Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

Positive vs. negative inputs:

1. $f : \mathcal{D} \to \{0,1\}$, $f(x) = 1 \Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$.
2. $w_+(x,\mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{F} \cap \mathcal{H}(x)\}$.
3. $w_-(x,\mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w_0|w\rangle = 1\}$.
4. $C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}$.

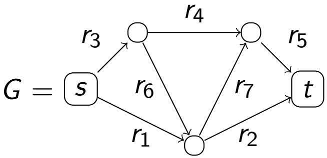
Thm: $Q(f; 2\Pi_{\mathcal{H}(x)} - I) = O(C(\mathcal{P}))$ [Rei11].



Electrical networks and span programs [BR12, JK17, JJKP18]

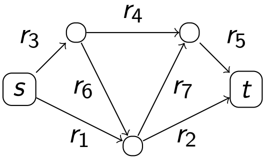
Electrical networks and span programs [BR12, JK17, JJKP18]

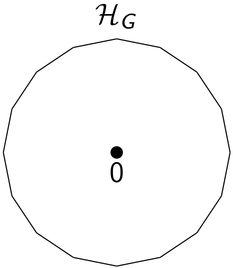
Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.



Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$.

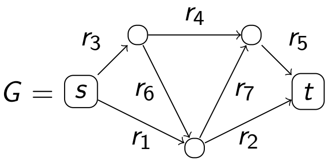
$G =$ 

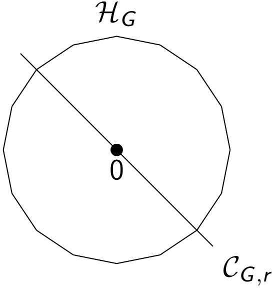


Electrical networks and span programs [BR12, JK17, JJKP18]

Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$.
2. *Circulation:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = 0$.
   Circulation space: $\mathcal{C}_{G,r} \subseteq \mathcal{H}_G$.

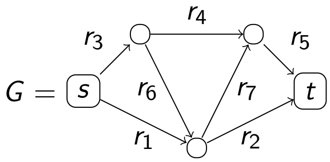


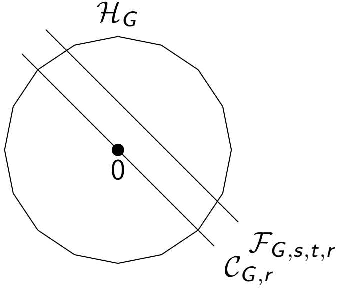


Electrical networks and span programs [BR12, JK17, JJKP18]

Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$.
2. *Circulation:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = 0$.
   Circulation space: $\mathcal{C}_{G,r} \subseteq \mathcal{H}_G$.
3. *Unit st-flow:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = \delta_{v,s} - \delta_{v,t}$.
   Unit st-flow subspace: $\mathcal{F}_{G,s,t} \subseteq \mathcal{H}_G$.

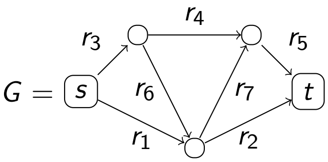


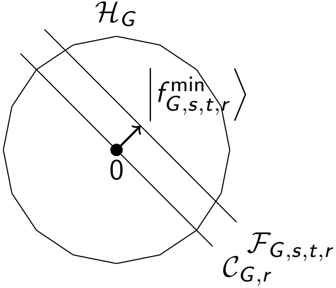


Electrical networks and span programs [BR12, JK17, JJKP18]

Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e}\, |e\rangle$.
2. *Circulation:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = 0$.
   Circulation space: $\mathcal{C}_{G,r} \subseteq \mathcal{H}_G$.
3. *Unit st-flow:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = \delta_{v,s} - \delta_{v,t}$.
   Unit st-flow subspace: $\mathcal{F}_{G,s,t} \subseteq \mathcal{H}_G$.
4. *Effective resistance:* $R_{G,s,t,r} := \| |f^{\min}_{G,s,t,r}\rangle \|^2$.

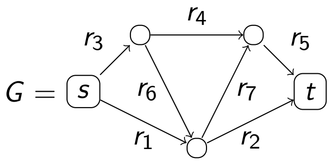


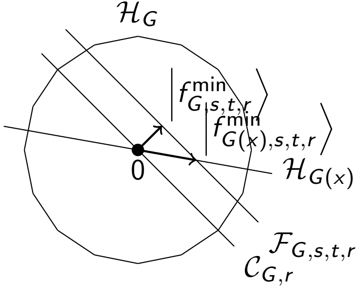


Electrical networks and span programs [BR12, JK17, JJKP18]

Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$.
2. *Circulation:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = 0$.
   Circulation space: $\mathcal{C}_{G,r} \subseteq \mathcal{H}_G$.
3. *Unit st-flow:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = \delta_{v,s} - \delta_{v,t}$.
   Unit st-flow subspace: $\mathcal{F}_{G,s,t} \subseteq \mathcal{H}_G$.
4. *Effective resistance:* $R_{G,s,t,r} := \| |f^{\min}_{G,s,t,r}\rangle \|^2$.
5. *Subgraph:* $x \in \{0,1\}^E \mapsto G(x) \mapsto \mathcal{H}_{G(x)} \subseteq \mathcal{H}_G$.

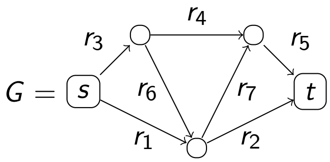


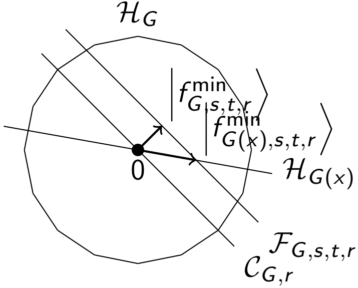


Electrical networks and span programs [BR12, JK17, JJKP18]

Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$.
2. *Circulation:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = 0$.
   Circulation space: $\mathcal{C}_{G,r} \subseteq \mathcal{H}_G$.
3. *Unit st-flow:* flow f with $\forall v \in V$,
   $\sum_{v \in N^+(v)} f_e - \sum_{v \in N^-(v)} f_e = \delta_{v,s} - \delta_{v,t}$.
   Unit st-flow subspace: $\mathcal{F}_{G,s,t} \subseteq \mathcal{H}_G$.
4. *Effective resistance:* $R_{G,s,t,r} := \| |f^{\min}_{G,s,t,r}\rangle \|^2$.
5. *Subgraph:* $x \in \{0,1\}^E \mapsto G(x) \mapsto \mathcal{H}_{G(x)} \subseteq \mathcal{H}_G$.





st-connectivity span program: $(\mathcal{H}_G, x \mapsto \mathcal{H}_{G(x)}, \mathcal{C}_{G,r}, |f^{\min}_{G,s,t,r}\rangle)$.

Graph compositions [This work]

Graph compositions [This work]

Graph composition:

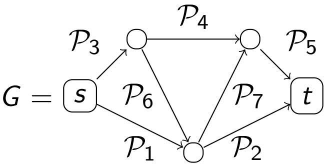
1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Graph composition:



Graph compositions [This work]

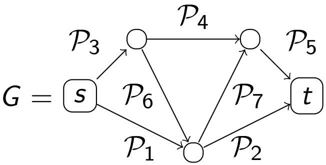
Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \||w_0^e\rangle\|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \||w_0^e\rangle\|^2$.
5. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

Graph composition:



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

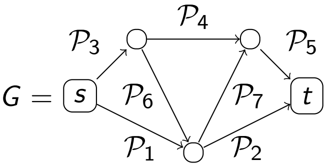
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \| |w_0^e\rangle \|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \| |w_0^e\rangle \|^2$.
5. $|w_0\rangle = \mathcal{E}(|f^{\min}_{G,s,t,r}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Graph composition:



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

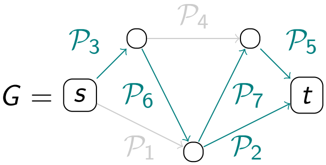
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \||w_0^e\rangle\|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \||w_0^e\rangle\|^2$.
5. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R_{G,s,t,r^-}^{-1}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

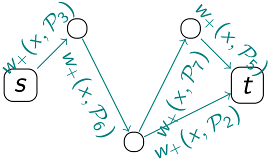
1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \| |w_0^e\rangle \|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \| |w_0^e\rangle \|^2$.
5. $|w_0\rangle = \mathcal{E}(|f^{\min}_{G,s,t,r}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:

$G =$



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

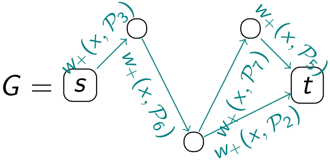
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \||w_0^e\rangle\|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \||w_0^e\rangle\|^2$.
5. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R_{G,s,t,r^-}^{-1}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

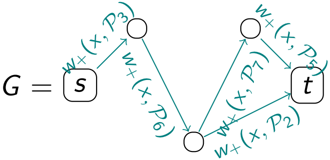
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \| |w_0^e\rangle \|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \| |w_0^e\rangle \|^2$.
5. $|w_0\rangle = \mathcal{E}(|f^{\min}_{G,s,t,r}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

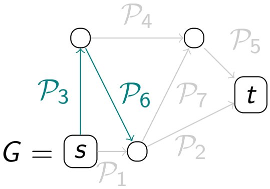
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative witness size $w_-(x, \mathcal{P})$:



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

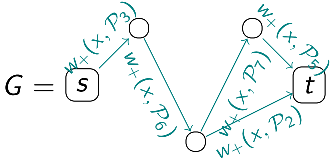
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \| |w_0^e\rangle \|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \| |w_0^e\rangle \|^2$.
5. $|w_0\rangle = \mathcal{E}(|f^{\min}_{G,s,t,r}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

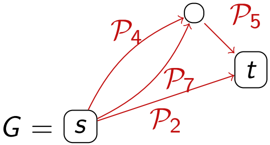
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative witness size $w_-(x, \mathcal{P})$:



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

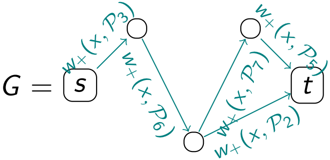
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \||w_0^e\rangle\|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \||w_0^e\rangle\|^2$.
5. $|w_0\rangle = \mathcal{E}(|f^{\min}_{G,s,t,r}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

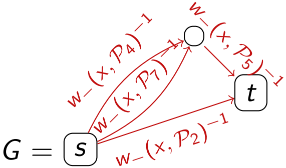
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative witness size $w_-(x, \mathcal{P})$:



Graph compositions [This work]

Graph composition:

1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

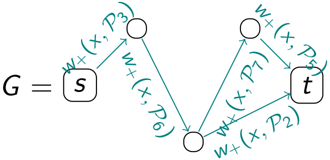
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

1. $\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}_e$
2. $\mathcal{H}(x) = \bigoplus_{e \in E} \mathcal{H}_e(x)$
3. $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}, |e\rangle \mapsto |w_0^e\rangle \,/\, \||w_0^e\rangle\|$.
4. $\mathcal{K} = \mathcal{E}(\mathcal{C}_{G,r}) \oplus \bigoplus_{e \in E} \mathcal{K}_e$, with $r_e = \||w_0^e\rangle\|^2$.
5. $|w_0\rangle = \mathcal{E}(|f^{\min}_{G,s,t,r}\rangle)$.

Main theorem: For all $x \in \mathcal{D}$,

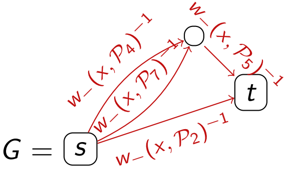
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



$$w_+(x, \mathcal{P}) = R_{G,s,t,r^+}.$$

Negative witness size $w_-(x, \mathcal{P})$:



$$w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}.$$

Path-cut theorem

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

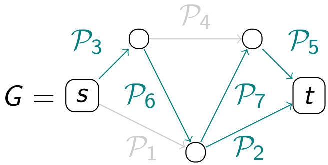
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:

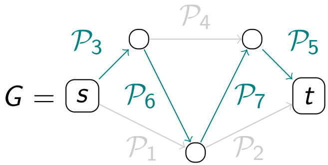


Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

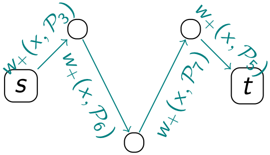
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



Path-cut theorem

Positive input:

$G =$ 

Theorem: For all $x \in \mathcal{D}$,

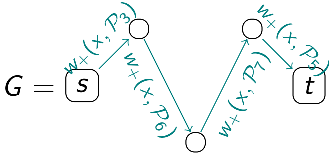
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



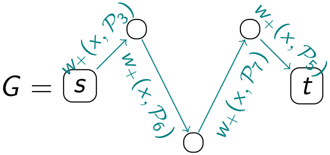
$$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e).$$

Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

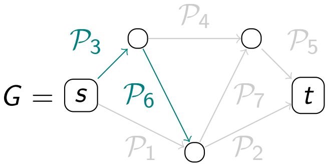
1. Let P be a path from s to t: $w_+(x,\mathcal{P}) \leq \sum_{e \in P} w_+(x,\mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x,\mathcal{P}) \leq \sum_{e \in C} w_-(x,\mathcal{P}_e)$.

Positive input:



$w_+(x,\mathcal{P}) \leq \sum_{e \in P} w_+(x,\mathcal{P}_e)$.

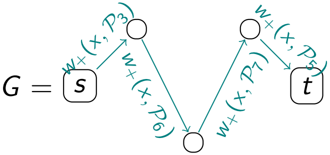
Negative input:



Theorem: For all $x \in \mathcal{D}$,

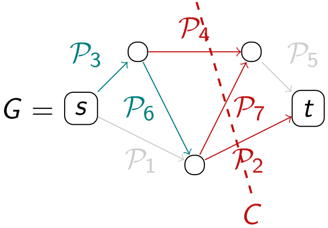
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.

Negative input:

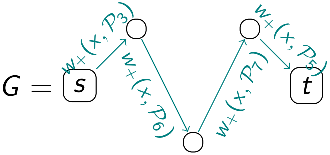


Path-cut theorem

Theorem: For all $x \in \mathcal{D}$,

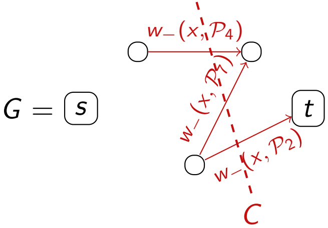
1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.

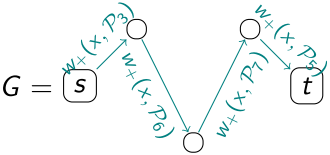
Negative input:



Theorem: For all $x \in \mathcal{D}$,

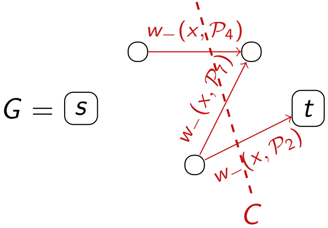
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



$$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e).$$

Negative input:



$$w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e).$$

Path-cut theorem

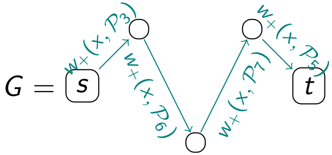
Theorem: For all $x \in \mathcal{D}$,

1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Properties:

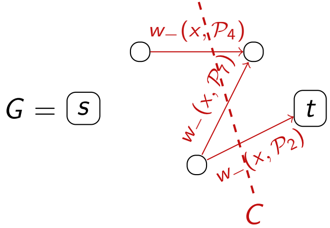
1. Simpler (less-powerful) version.
2. Still powerful enough for many applications.

Positive input:



$$w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e).$$

Negative input:



$$w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e).$$

Example: OR-function

Example: OR-function

Trivial span program: $(\alpha > 0)$

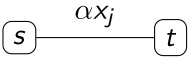
1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

Example: OR-function

Trivial span program: $(\alpha > 0)$

1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

Trivial span program for x_j:



$(\alpha > 0)$.

Example: OR-function

Trivial span program: $(\alpha > 0)$

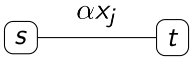
1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

The OR-function:

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

Trivial span program for x_j:



$(\alpha > 0)$.

Example: OR-function

Trivial span program: $(\alpha > 0)$

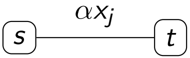
1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

The OR-function:

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

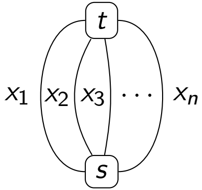
$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

Trivial span program for x_j:



$(\alpha > 0)$.

Graph composition for OR_n:



Example: OR-function

Trivial span program: $(\alpha > 0)$

1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

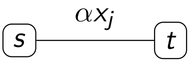
The OR-function:

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

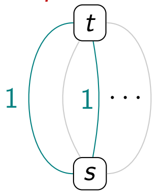
2. $w_+(x) = \frac{1}{|x|} \leq 1$.

Trivial span program for x_j:



$(\alpha > 0)$.

Graph composition for OR_n:



$x = 1010\cdots0 \Rightarrow w_+(x) = \frac{1}{2}$

Example: OR-function

Trivial span program: $(\alpha > 0)$

1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

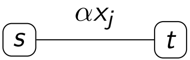
The OR-function:

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$

$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$

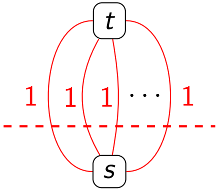
2. $w_+(x) = \frac{1}{|x|} \leq 1$.
3. $w_-(x) = n$.

Trivial span program for x_j:



$(\alpha > 0)$.

Graph composition for OR_n:



$x = 0000\cdots 0 \Rightarrow w_-(x) = n$

Example: OR-function

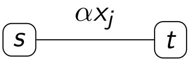
Trivial span program: $(\alpha > 0)$

1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

The OR-function:

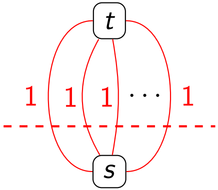
1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|} \leq 1$.
3. $w_-(x) = n$.
4. $C(\mathcal{P}) = \sqrt{n}$.

Trivial span program for x_j:



$(\alpha > 0)$.

Graph composition for OR_n:



$x = 0000\cdots 0 \Rightarrow w_-(x) = n$

Example: OR-function

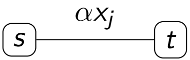
Trivial span program: $(\alpha > 0)$

1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

The OR-function:

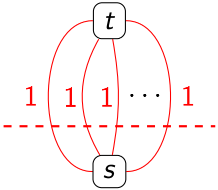
1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$
$$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|} \leq 1$.
3. $w_-(x) = n$.
4. $C(\mathcal{P}) = \sqrt{n}$.
5. $\Rightarrow \mathrm{Q}(\mathrm{OR}_n) \in O(\sqrt{n})$.

Trivial span program for x_j:



$(\alpha > 0)$.

Graph composition for OR_n:



$x = 0000 \cdots 0 \Rightarrow w_-(x) = n$

Example: OR-function

Trivial span program: $(\alpha > 0)$

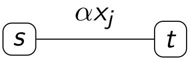
1. $x_j : \{0,1\}^n \to \{0,1\}$, $x \mapsto x_j$.
2. $w_+(x, x_j) = \alpha$, if $x_j = 1$.
3. $w_-(x, x_j) = 1/\alpha$, if $x_j = 0$.

The OR-function:

1. $\mathrm{OR}_n : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{OR}_n(x) = \begin{cases} 1, & \text{if } |x| \geq 1, \\ 0, & \text{if } |x| = 0. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|} \leq 1$.
3. $w_-(x) = n$.
4. $C(\mathcal{P}) = \sqrt{n}$.
5. $\Rightarrow Q(\mathrm{OR}_n) \in O(\sqrt{n})$.

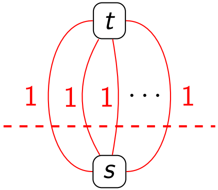
Quadratic speed-up for search.

Trivial span program for x_j:



$(\alpha > 0)$.

Graph composition for OR_n:



$x = 0000\cdots0 \Rightarrow w_-(x) = n$

Example: Threshold function

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$

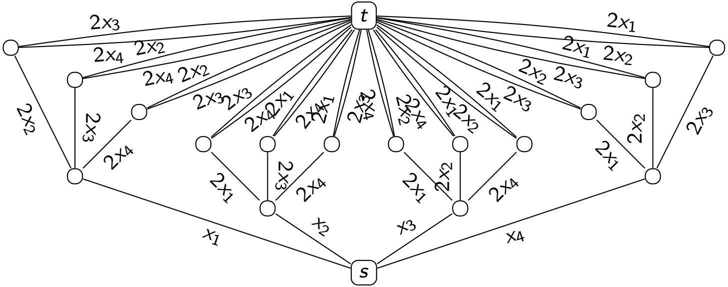
Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$

   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$

Graph composition for Th_4^3*:*

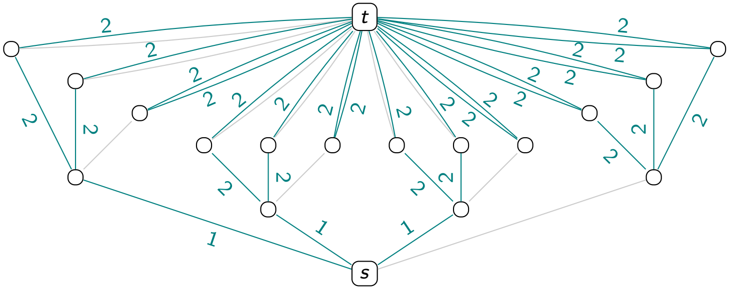


Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$

Graph composition for Th_4^3:



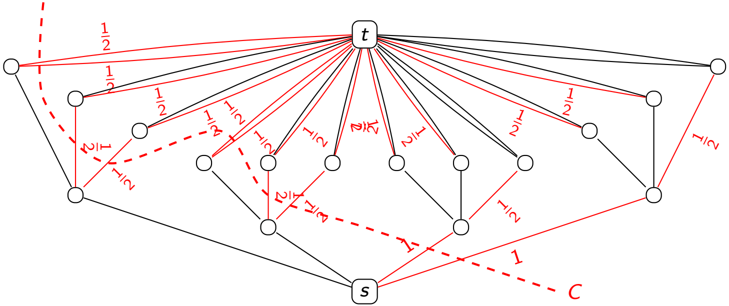
$x = 1110 \Rightarrow w_+(x) = 1$

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$

Graph composition for Th_4^3*:*



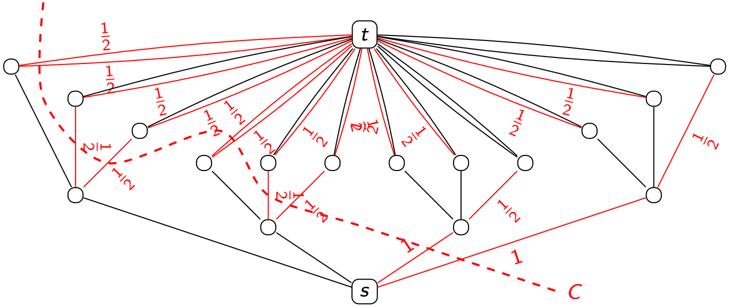
$x = 1100 \Rightarrow w_-(x) = 6$

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$
4. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.

Graph composition for Th_4^3*:*



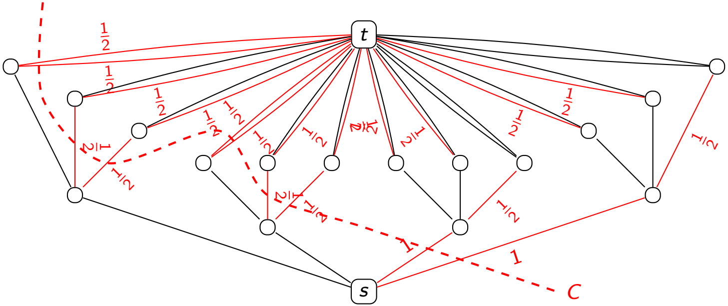
$x = 1100 \Rightarrow w_-(x) = 6$

Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$
4. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow Q(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

Graph composition for Th_4^3*:*



$x = 1100 \Rightarrow w_-(x) = 6$

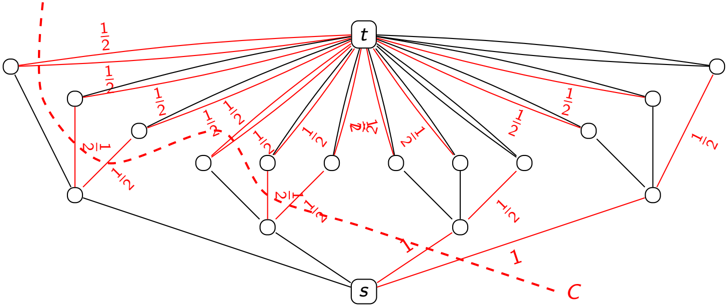
Example: Threshold function

The threshold function: $(k \in [n])$

1. $\mathrm{Th}_n^k : \{0,1\}^n \to \{0,1\}$
   $$\mathrm{Th}_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{if } |x| < k. \end{cases}$$
2. $w_+(x) = \frac{1}{|x|-k+1}$
3. $w_-(x) = \frac{k(n-k+1)}{k-|x|}$
4. $C(\mathcal{P}) = \sqrt{k(n-k+1)}$.
5. $\Rightarrow Q(\mathrm{Th}_n^k) \in O(\sqrt{k(n-k+1)})$.

Known to be optimal!

Graph composition for Th_4^3*:*



$x = 1100 \Rightarrow w_-(x) = 6$

Example: the $\Sigma^*20^*2\Sigma^*$-problem

Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

Example: the $\Sigma^*20^*2\Sigma^*$-problem

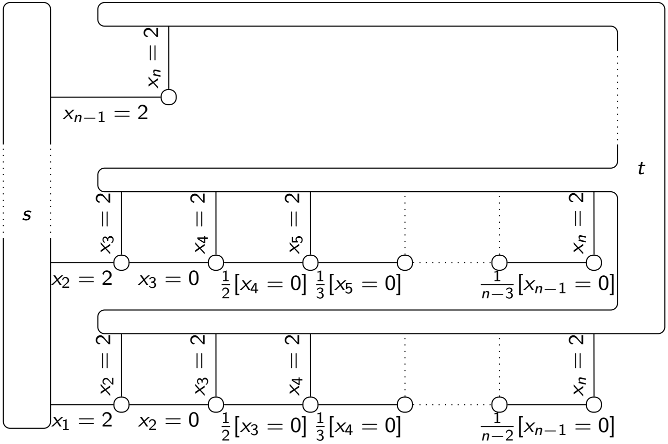
$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.

Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

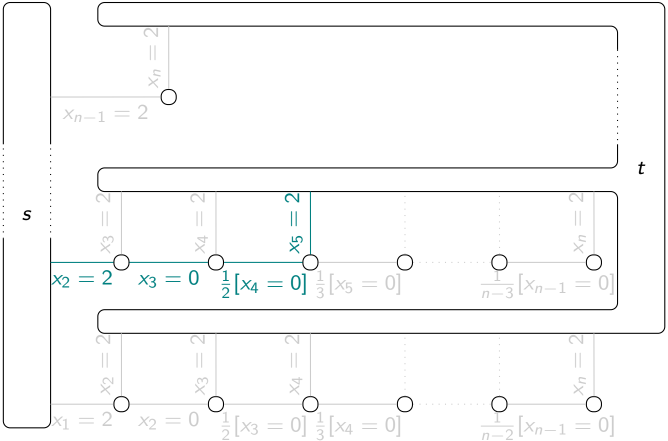
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.



Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0,1,2\}$, $f : \Sigma^n \to \{0,1\}$.

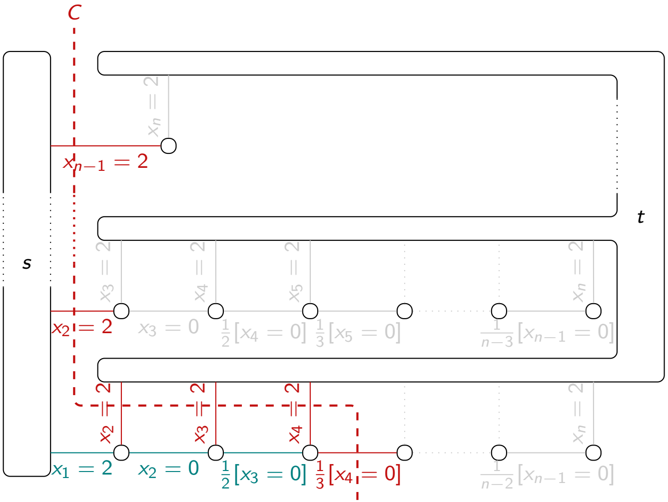
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.



Example: the $\Sigma^*20^*2\Sigma^*$-problem

$\Sigma = \{0,1,2\}$, $f : \Sigma^n \to \{0,1\}$.

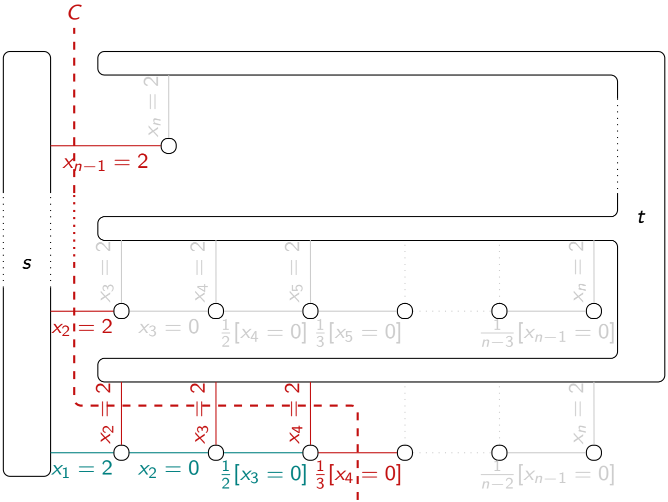
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.
3. Let x be a negative instance.
   $x = 2\underbrace{001}_{\ell_1=3}102\underbrace{0001}_{\ell_2=4}002\underbrace{001}_{\ell_3=3}\cdots$
   $\Rightarrow w_-(x,\mathcal{P}) \leq n + \sum_{j=1}^{k} 2\ell_j \in O(n)$.



Example: the $\Sigma^*20^*2\Sigma^*$-problem

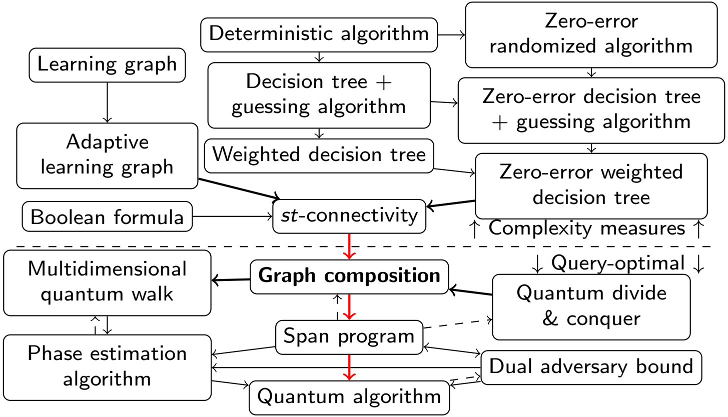
$\Sigma = \{0,1,2\}$, $f : \Sigma^n \to \{0,1\}$.

1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.
3. Let x be a negative instance.
   $x = 2\underbrace{001}_{\ell_1=3}102\underbrace{0001}_{\ell_2=4}002\underbrace{001}_{\ell_3=3}\cdots$
   $\Rightarrow w_-(x,\mathcal{P}) \leq n + \sum_{j=1}^{k} 2\ell_j \in O(n)$.
4. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.

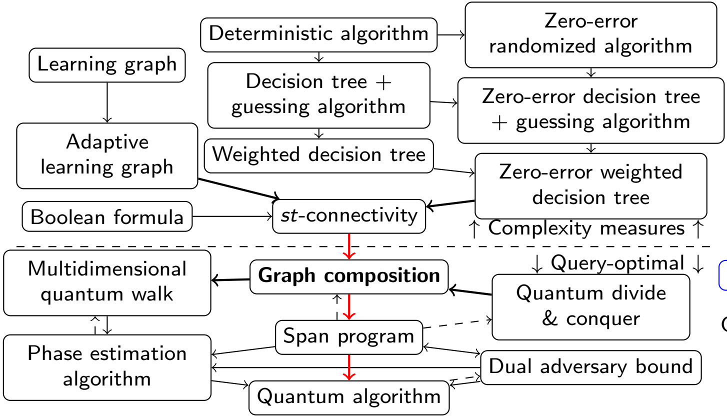


Summary (I/II)

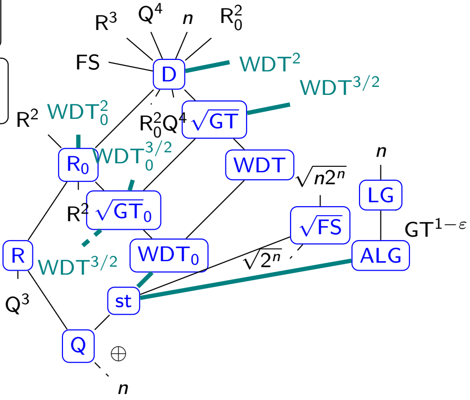
Relations between algorithmic frameworks



Relations between algorithmic frameworks



Relations between complexity measures



Summary (II/II)

Summary (II/II)

Graph composition:

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.

Examples:

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.
2. In the paper:
   1. Pattern matching.
   2. $\mathrm{OR} \circ \mathrm{pSEARCH}$.
   3. Dyck-language recognition with depth 3.
   4. 3-increasing subsequence.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.
3. *Subsequent developments:*

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.
2. In the paper:
   1. Pattern matching.
   2. $\mathrm{OR} \circ \mathrm{pSEARCH}$.
   3. Dyck-language recognition with depth 3.
   4. 3-increasing subsequence.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.
3. *Subsequent developments:*
   1. Time-efficient implementation.

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.
2. In the paper:
   1. Pattern matching.
   2. $\mathrm{OR} \circ \mathrm{pSEARCH}$.
   3. Dyck-language recognition with depth 3.
   4. 3-increasing subsequence.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.
3. *Subsequent developments:*
   1. Time-efficient implementation.
   2. Generalization to switches.

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.
2. In the paper:
   1. Pattern matching.
   2. $\mathrm{OR} \circ \mathrm{pSEARCH}$.
   3. Dyck-language recognition with depth 3.
   4. 3-increasing subsequence.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.
3. *Subsequent developments:*
   1. Time-efficient implementation.
   2. Generalization to switches.
   3. Inclusion of quantum walk algorithms.

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.
2. In the paper:
   1. Pattern matching.
   2. $\mathrm{OR} \circ \mathrm{pSEARCH}$.
   3. Dyck-language recognition with depth 3.
   4. 3-increasing subsequence.

Summary (II/II)

Graph composition:

1. *Definition:*
   1. *st*-connectivity with edge span programs.
2. *Analysis:*
   1. Exact witness characterization using effective resistances.
   2. Path-cut theorem: weaker but easier to apply.
3. *Subsequent developments:*
   1. Time-efficient implementation.
   2. Generalization to switches.
   3. Inclusion of quantum walk algorithms.

Examples:

1. In this talk:
   1. OR, $\mathrm{Threshold}$.
   2. The $\Sigma^*20^*2\Sigma^*$-problem.
2. In the paper:
   1. Pattern matching.
   2. $\mathrm{OR} \circ \mathrm{pSEARCH}$.
   3. Dyck-language recognition with depth 3.
   4. 3-increasing subsequence.

Thanks for your attention!

`ajcornelissen@outlook.com`

References (I/III)

[AGJ21] Simon Apers, András Gilyén, and Stacey Jeffery. A unified framework of quantum walk search.

[Bel12b] Aleksandrs Belovs. Span programs for functions with constant-sized 1-certificates.

[Bel12a] Aleksandrs Belovs. Learning-graph-based quantum algorithm for k-distinctness.

[BR12] Aleksandrs Belovs and Ben W Reichardt. Span programs and quantum algorithms for *st*-connectivity and claw detection.

[BT20] Salman Beigi and Leila Taghavi. Quantum speedup based on classical decision trees.

[CKK+22] Andrew M Childs, Robin Kothari, Matt Kovacs-Deak, Aarthi Sundaram, and Daochen Wang. Quantum divide and conquer.

[CMP22] Arjan Cornelissen, Nikhil S Mande, and Subhasree Patro. Improved quantum query upper bounds based on classical decision trees.

[JJKP18] Michael Jarret, Stacey Jeffery, Shelby Kimmel, and Alvaro Piedrafita. Quantum algorithms for connectivity and related problems.

References (II/III)

[JK17] Stacey Jeffery and Shelby Kimmel. Quantum algorithms for graph connectivity and formula evaluation.

[JP24] Stacey Jeffery and Galina Pass. Multidimensional quantum walks, recursion, and quantum divide & conquer.

[JZ25] Stacey Jeffery and Sebastian Zur. Multidimensional quantum walks, with application to k-distinctness.

[LL16] Cedric Yen-Yu Lin and Han-Hsuan Lin. Upper bounds on quantum query complexity inspired by the elitzur–vaidman bomb tester.

[LMR+11] Troy Lee, Rajat Mittal, Ben W Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion.

[Rei09] Ben W Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function.

[Rei11] Ben W Reichardt. Reflections for quantum query algorithms.

References (III/III)

[RŠ12] Ben Reichardt and Robert Špalek. Span-program-based quantum algorithm for evaluating formulas.

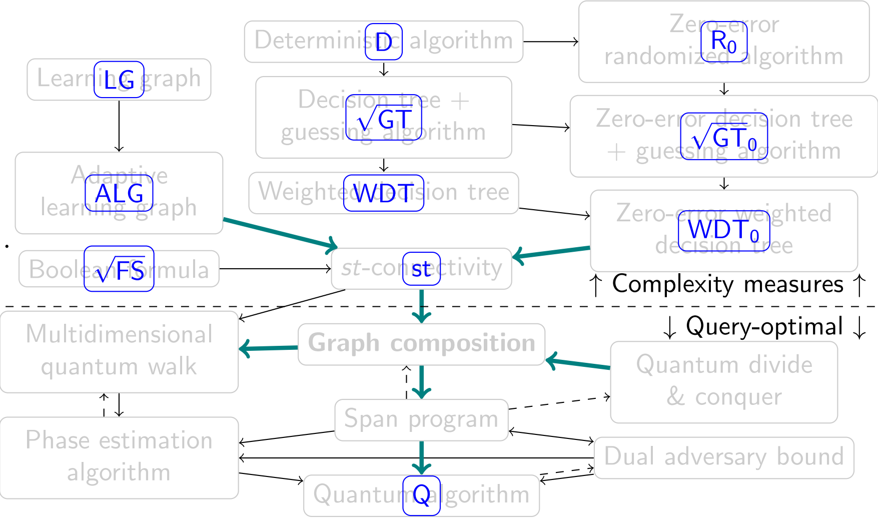
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

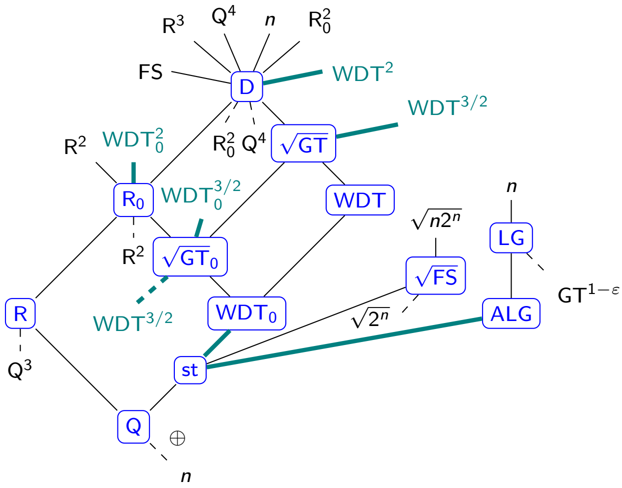
1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.
3. $O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle$.

Framework:

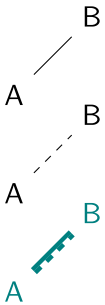
1. Define object L.
2. Convert object into quantum algorithm.



Complexity measure relations for total boolean functions



Legend:

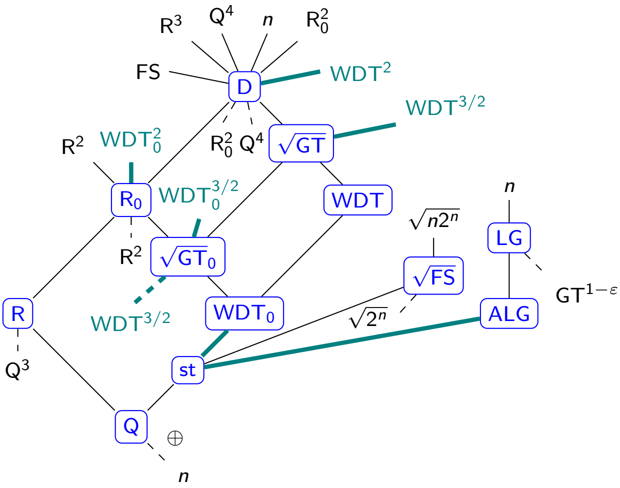


$\forall f : \{0,1\}^n \to \{0,1\}$
$A(f) \in \widetilde{O}(B(f))$

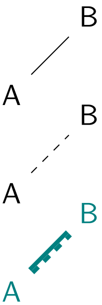
$\exists f : \{0,1\}^n \to \{0,1\}$
$A(f) \in \widetilde{O}(B(f))$

New in this work

Complexity measure relations for total boolean functions



Legend:



$\forall f : \{0,1\}^n \to \{0,1\}$
$A(f) \in \widetilde{O}(B(f))$

$\exists f : \{0,1\}^n \to \{0,1\}$
$A(f) \in \widetilde{O}(B(f))$

New in this work

Open questions:

1. Separation between Q and st?
2. Can we prove $R \in \widetilde{O}(st^2)$?