Quantum algorithms through composition of graphs

Arjan Cornelissen[1]

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

March 26th, 2025

SIMONS INSTITUTE for the Theory of Computing

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. $\mathcal{D} \subseteq \{0, 1\}^n$.

Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.

Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.

Quantum algorithm

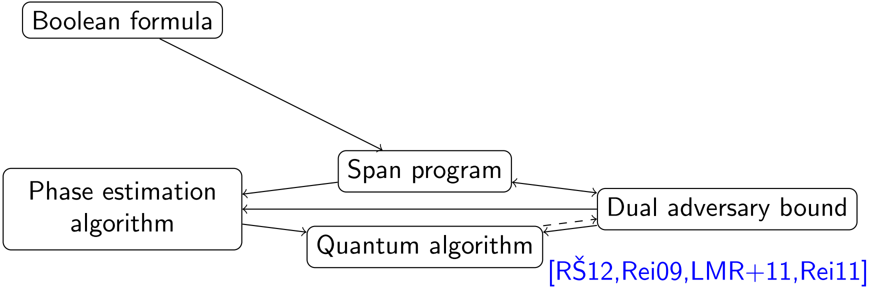
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



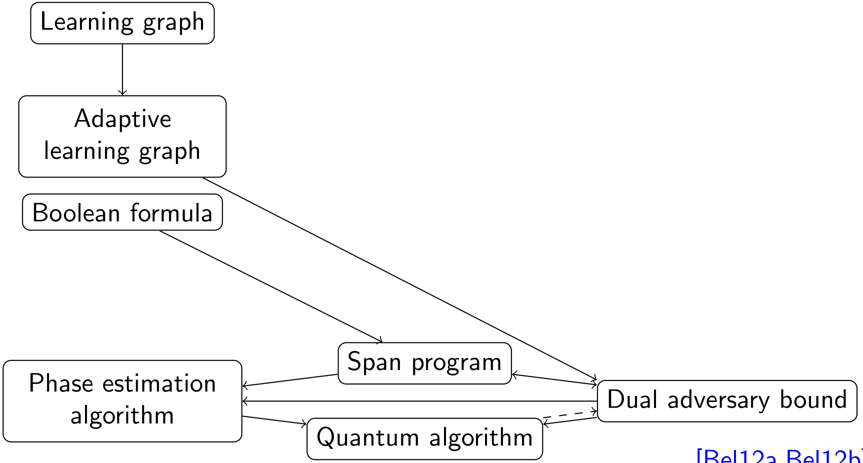
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



[Bel12a,Bel12b]

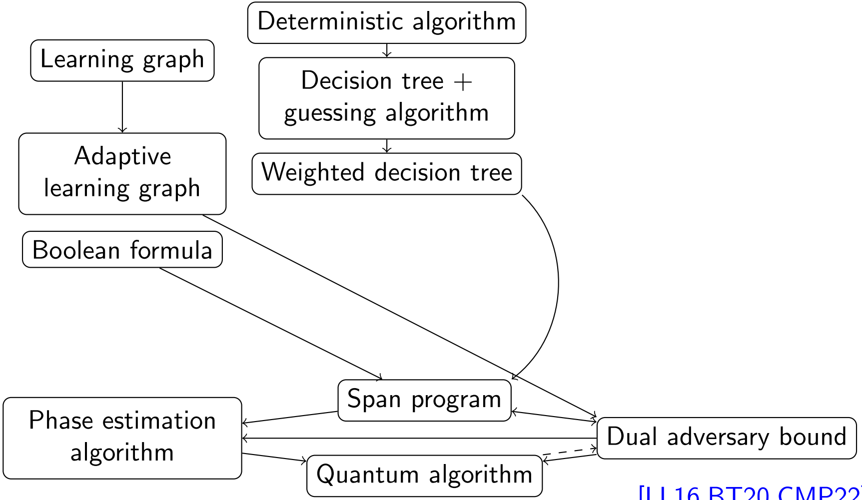
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



[LL16,BT20,CMP22]

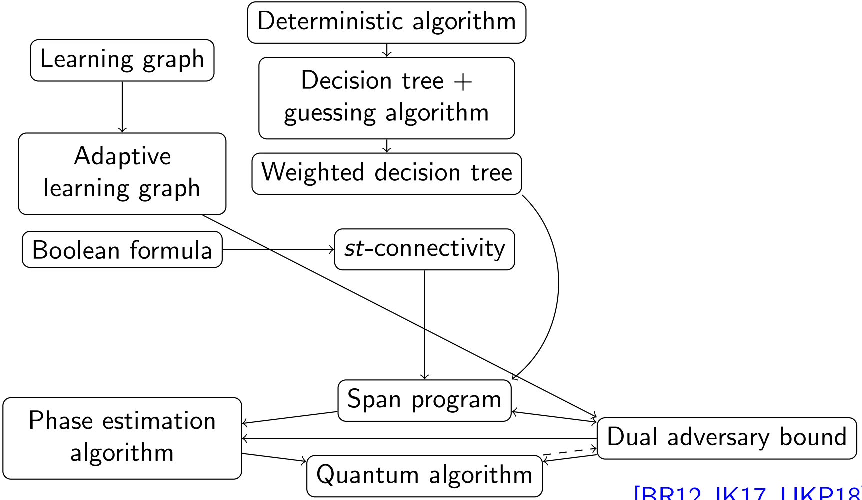
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



[BR12,JK17,JJKP18]

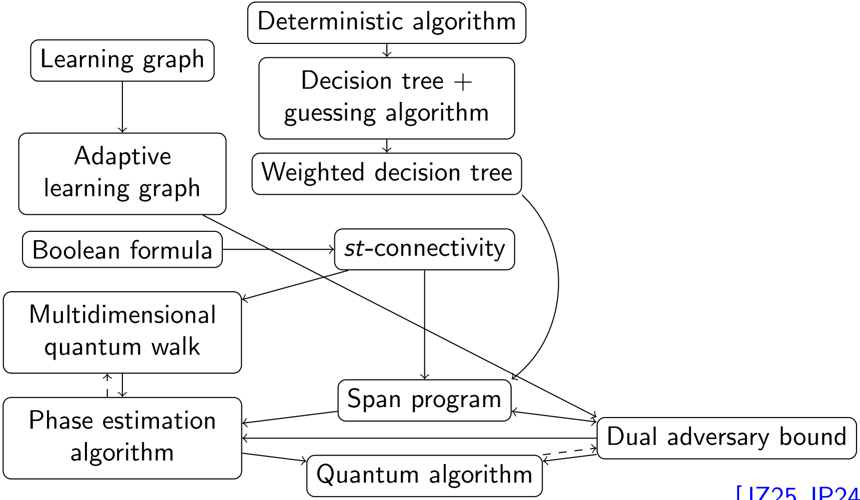
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



[JZ25,JP24]

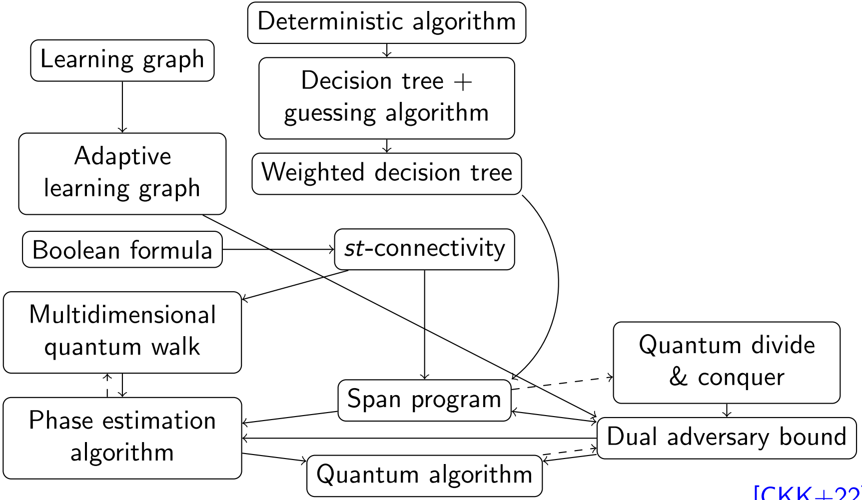
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



[CKK+22]

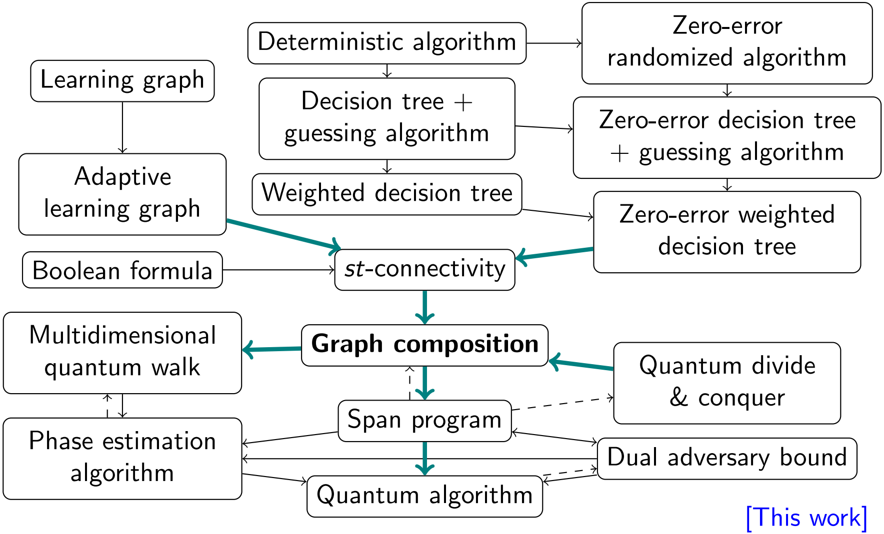
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



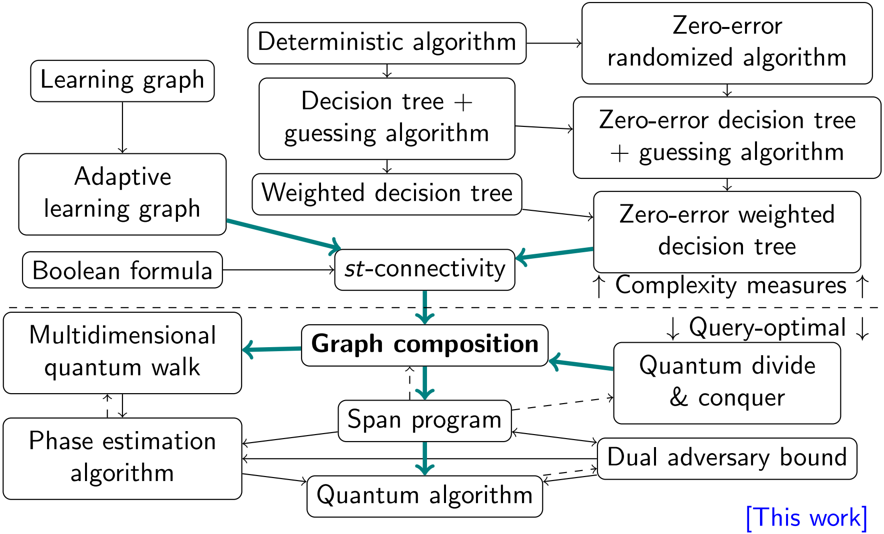
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

1. $f : \mathcal{D} \to \{0, 1\}$.
2. $\mathcal{D} \subseteq \{0, 1\}^n$.

Method: Two types of frameworks:

1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



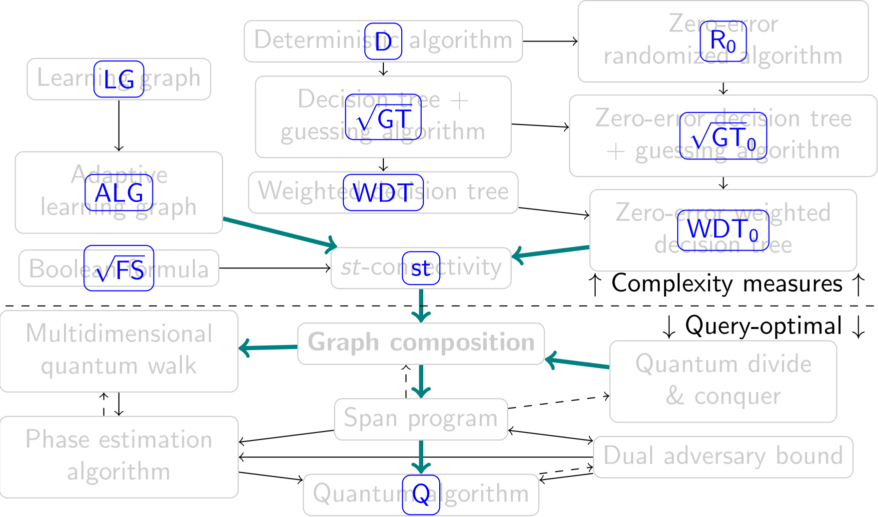
Quantum algorithmic frameworks (for boolean functions)

Goal: Design algorithm for boolean function f:

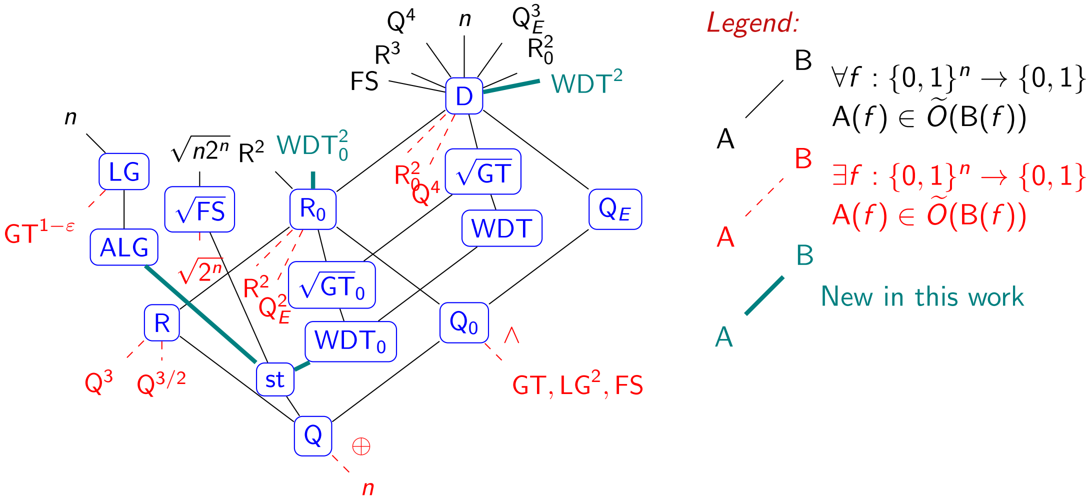
1. $f : \mathcal{D} \to \{0,1\}$.
2. $\mathcal{D} \subseteq \{0,1\}^n$.

Method: Two types of frameworks:

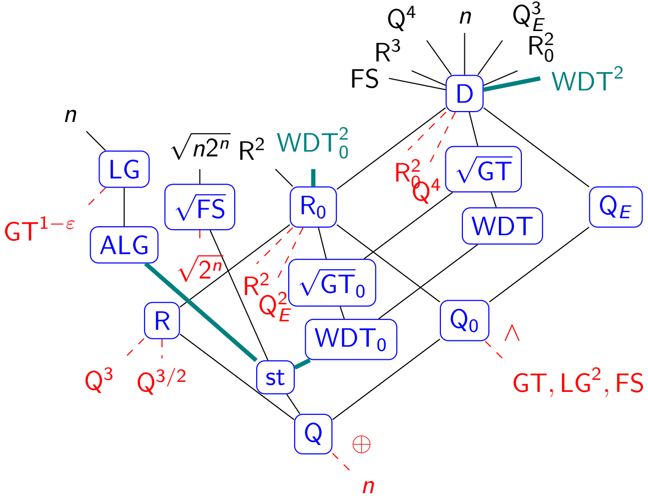
1. Quantum walks. Unification: [AGJ21].
2. Span programs / adversary bound. Unification: **[This work]**.



Complexity measure relations for total boolean functions



Complexity measure relations for total boolean functions



Legend:

B

A

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

B

A

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

B

A

New in this work

Open questions:

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

Span programs [RŠ12,Rei09,Rei11]

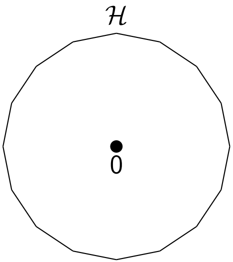
Span programs [RŠ12,Rei09,Rei11]

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 $\mathcal{D}$.

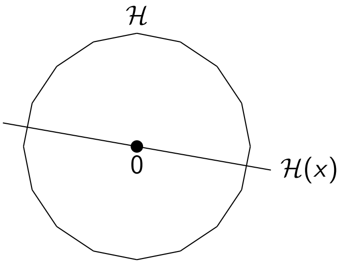
1. *Hilbert space:* $\mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

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

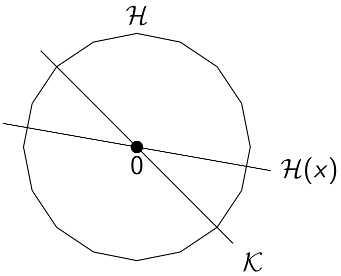
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.



Span programs [RŠ12,Rei09,Rei11]

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

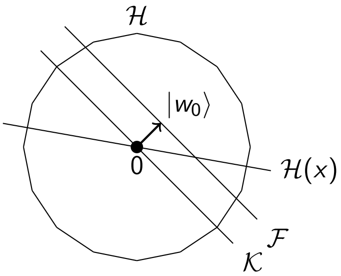
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.



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:* $|w_0\rangle \in \mathcal{K}^\perp$.



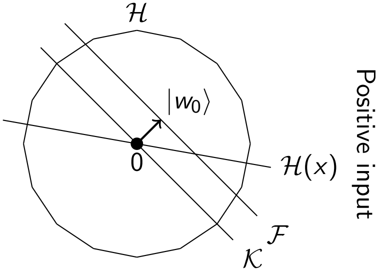
Span programs [RŠ12,Rei09,Rei11]

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



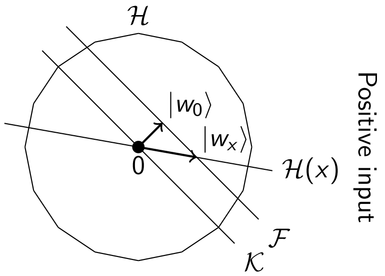
Span programs [RŠ12,Rei09,Rei11]

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. $w_+(x, \mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{F} \cap \mathcal{H}(x)\}$.



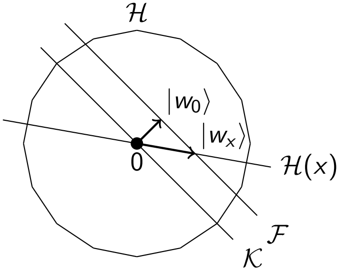
Span programs [RŠ12,Rei09,Rei11]

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

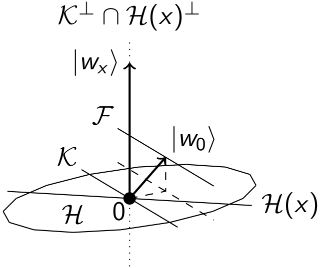
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

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. $w_-(x, \mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w_0|w\rangle = 1\}$.



Positive input



Negative input

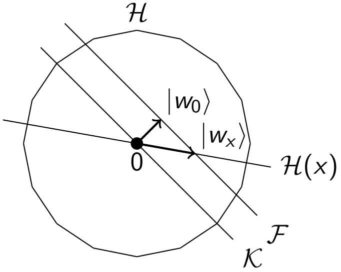
Span programs [RŠ12,Rei09,Rei11]

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

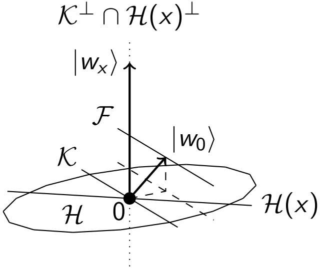
1. *Hilbert space:* $\mathcal{H}$.
2. *Input-dependent subspace:* $\forall x \in \mathcal{D}, \mathcal{H}(x) \subseteq \mathcal{H}$.
3. *Input-independent subspace:* $\mathcal{K} \subseteq \mathcal{H}$.
4. *Initial vector:* $|w_0\rangle \in \mathcal{K}^\perp$.

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. $C(\mathcal{P}) = \sqrt{\max_{x \in f^{-1}(0)} w_-(x,\mathcal{P}) \cdot \max_{x \in f^{-1}(1)} w_+(x,\mathcal{P})}$.



Positive input



Negative input

Span programs [RŠ12,Rei09,Rei11]

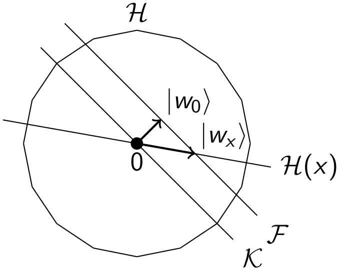
Span program: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ on $\mathcal{D}$.

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:* $|w_0\rangle \in \mathcal{K}^\perp$.

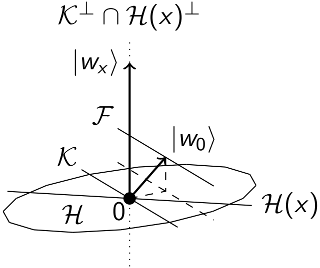
Positive vs. negative inputs:

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].



Positive input

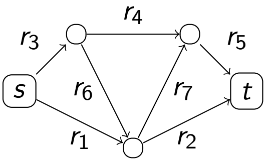


Negative input

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

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

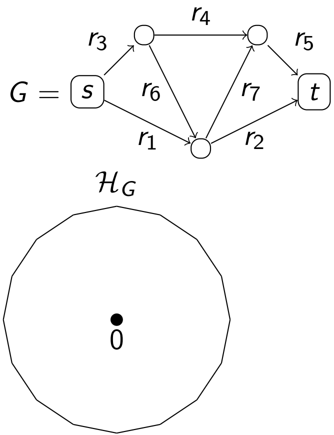
Graph $G = (V, E)$, resistances $r : E \to [0, \infty]$, $s, t \in V$.

$G =$ 

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

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

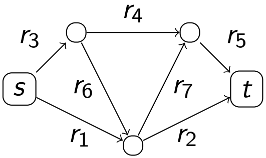
1. *Flow:* $f : E \to \mathbb{C}$.
   Flow space: $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$,
   $f \mapsto |f_{G,r}\rangle = \sum_{e \in E} f_e \sqrt{r_e} \, |e\rangle$.

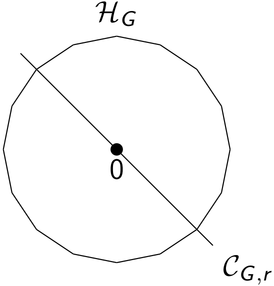


Electrical networks and span programs [BR12, JK17, JJKP18]

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: $\mathcal{C}_{G,r} \subseteq \mathcal{H}_G$.

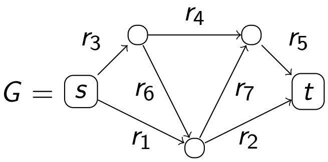
$G =$ 

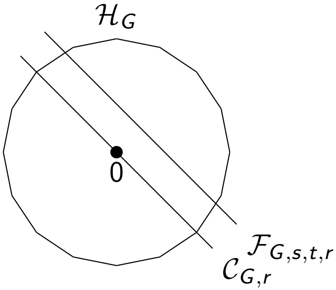


Electrical networks and span programs [BR12, JK17, JJKP18]

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: $\mathcal{F}_{G,s,t} \subseteq \mathcal{H}_G$.

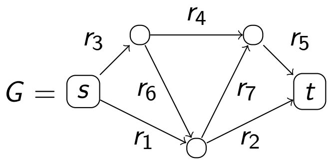


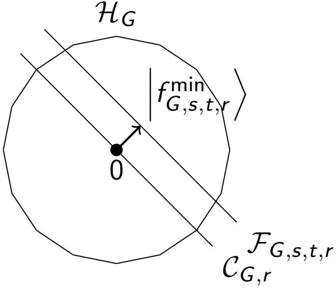


Electrical networks and span programs [BR12, JK17, JJKP18]

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:* $R_{G,s,t,r} := \||f^{\min}_{G,s,t,r}\rangle\|^2$.

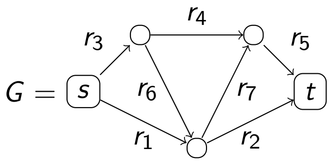


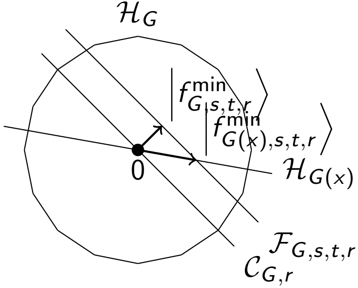


Electrical networks and span programs [BR12, JK17, JJKP18]

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:* $x \in \{0,1\}^E \mapsto G(x) \mapsto \mathcal{H}_{G(x)} \subseteq \mathcal{H}_G$.



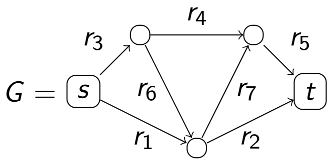


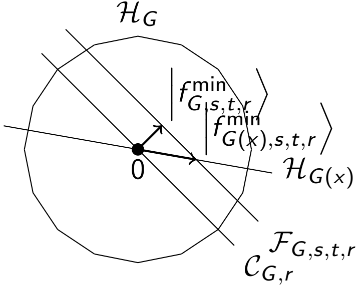
Electrical networks and span programs [BR12, JK17, JJKP18]

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:

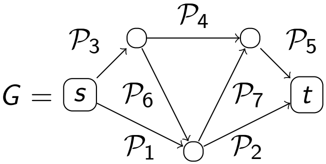
1. Undirected graph $G = (V, E)$.
2. Edge span programs $(\mathcal{P}_e)_{e \in E}$ on $\mathcal{D}$.

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]

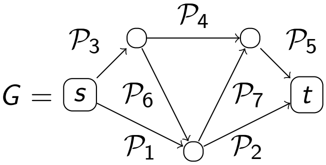
Graph composition:

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 $\mathcal{D}$.

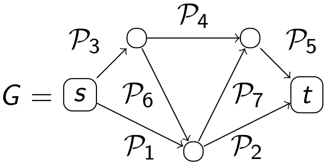
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

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 $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

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 $\mathcal{D}$.

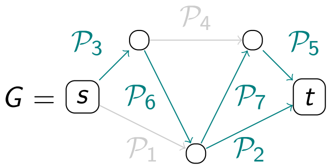
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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 $\mathcal{D}$.

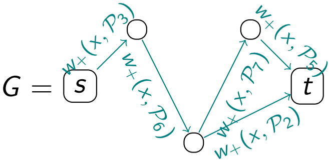
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

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 $\mathcal{D}$.

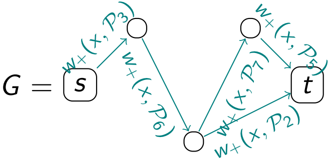
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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:



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

Graph compositions [This work]

Graph composition:

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

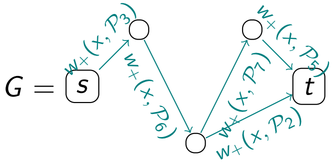
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

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

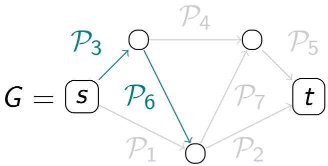
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R_{G,s,t,r^-}^{-1}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



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

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 $\mathcal{D}$.

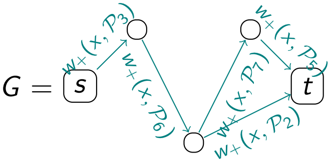
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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. $|w_0\rangle = \mathcal{E}(|f_{G,s,t,r}^{\min}\rangle)$.

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

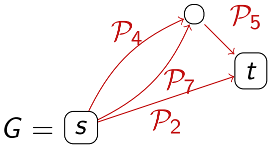
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R_{G,s,t,r^-}^{-1}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

Positive witness size:



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

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 $\mathcal{D}$.

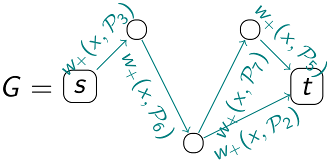
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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 $x \in \mathcal{D}$,

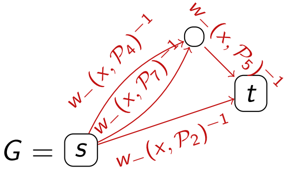
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

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 $\mathcal{D}$.

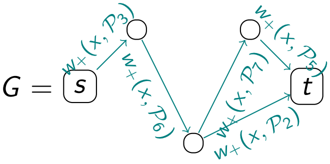
Formally: Span program $\mathcal{P}$ on $\mathcal{D}$:

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 $x \in \mathcal{D}$,

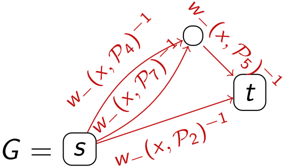
1. $w_+(x, \mathcal{P}) = R_{G,s,t,r^+}$ with $r_+(e) = w_+(x, \mathcal{P}_e)$.
2. $w_-(x, \mathcal{P}) = R^{-1}_{G,s,t,r^-}$ with $r^-(e) = w_-(x, \mathcal{P}_e)^{-1}$.

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

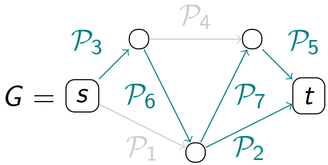
Path-cut theorem

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

Positive input:



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

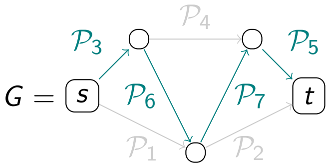
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Path-cut theorem

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

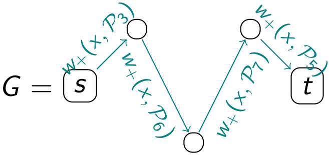
1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



Path-cut theorem

Positive input:



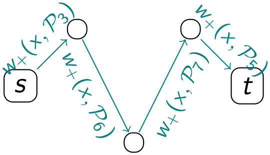
Theorem: For all $x \in \mathcal{D}$,

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:

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

$G =$ 

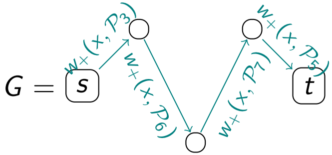
$$w_+(x,\mathcal{P}) \leq \sum_{e\in P} w_+(x,\mathcal{P}_e).$$

Path-cut theorem

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

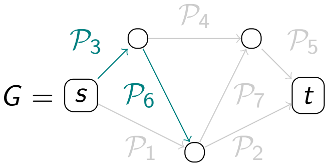
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



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

Negative input:

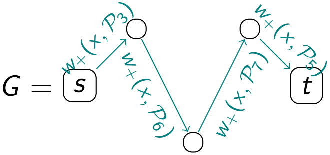


Path-cut theorem

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

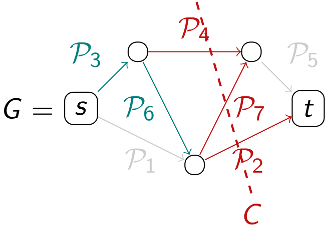
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



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

Negative input:

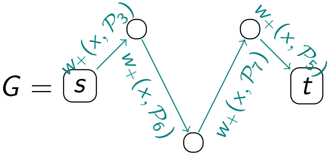


Path-cut theorem

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

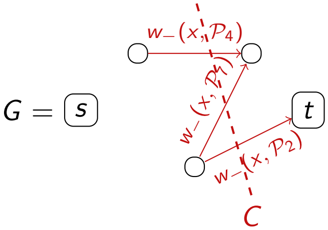
1. Let P be a path from s to t:
   $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t:
   $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



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

Negative input:

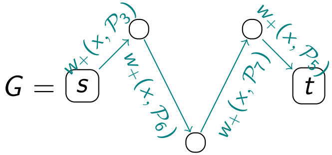


Path-cut theorem

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

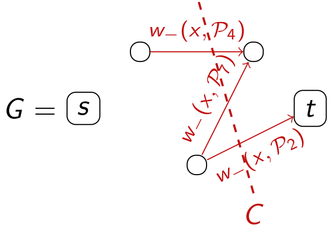
1. Let P be a path from s to t: $w_+(x, \mathcal{P}) \leq \sum_{e \in P} w_+(x, \mathcal{P}_e)$.
2. Let C be a cut between s and t: $w_-(x, \mathcal{P}) \leq \sum_{e \in C} w_-(x, \mathcal{P}_e)$.

Positive input:



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

Negative input:



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

Path-cut theorem

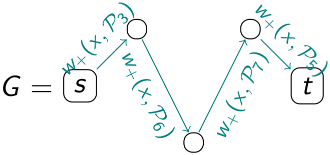
Theorem: For all $x \in \mathcal{D}$,

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:

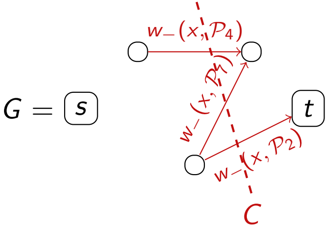
1. Simpler (less-powerful) version.
2. Still powerful enough for many applications.

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: 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

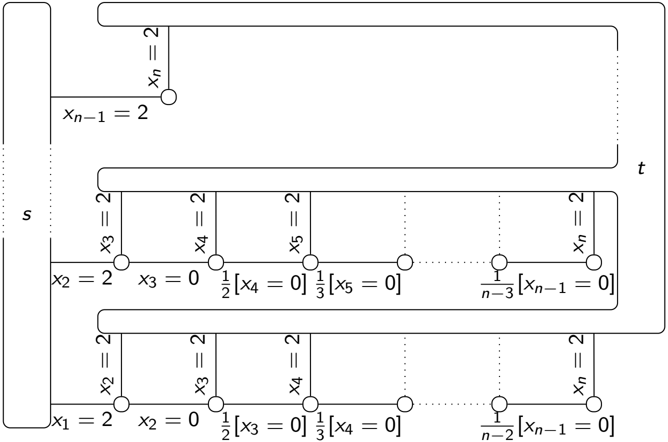
$\Sigma = \{0, 1, 2\}$, $f : \Sigma^n \to \{0, 1\}$.

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

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

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

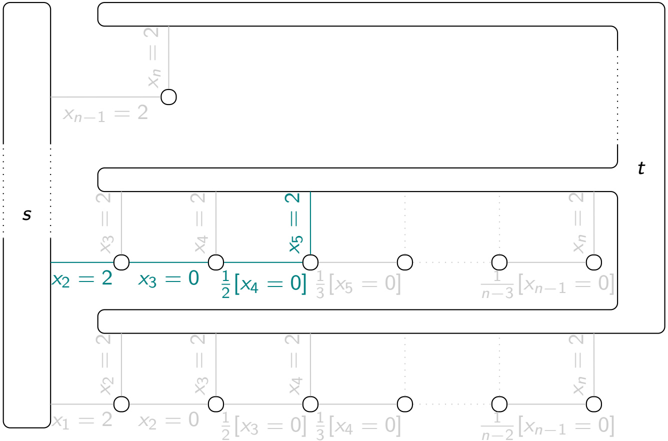
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.



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

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

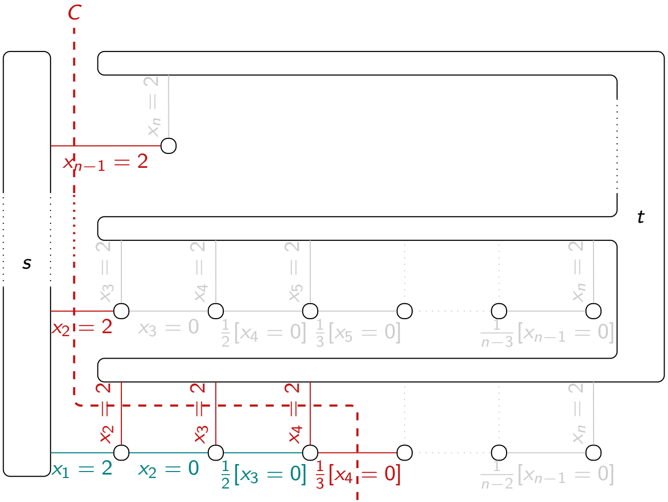
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.



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

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

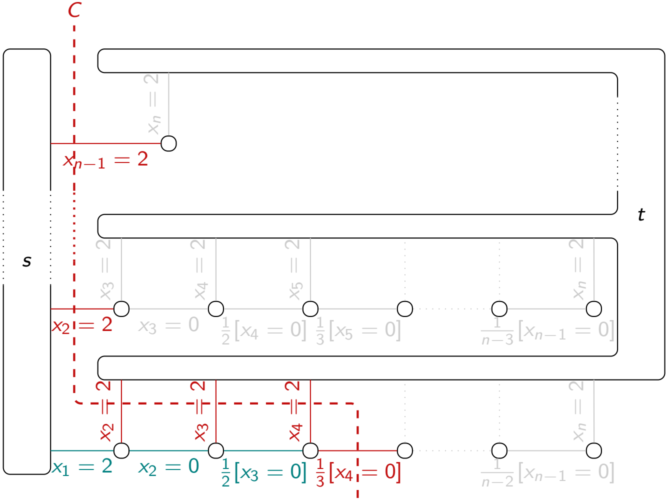
1. $f(x) = [x \in \Sigma^*20^*2\Sigma^*]$.
2. Let x be a positive instance.
   $x = \cdots 0102\underbrace{000000}_{\text{length } \ell}2100\cdots$
   $\Rightarrow w_+(x,\mathcal{P}) \leq 1 + \sum_{j=1}^{\ell} \frac{1}{j} + 1 \in O(\log(n))$.
3. Let x be a negative instance.
   $x = 2\underbrace{001}_{\ell_1=3}102\underbrace{0001}_{\ell_2=4}002\underbrace{001}_{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



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. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.



(Time-efficient) Implementation (I/II)

(Time-efficient) Implementation (I/II)

Three operations: (each is called $O(C(\mathcal{P}))$ times)

1. $2\Pi_{\mathcal{H}(x)} - I = \bigoplus_{e \in E}(2\Pi_{\mathcal{H}^e(x)} - I)$.
2. $2\Pi_{\mathcal{K}} - I = -(2\Pi_{\mathcal{C}_{G,r}} - I) \bigoplus_{e \in E}(2\Pi_{\mathcal{K}^e} - I)$.
3. $C_{|w_0\rangle} : |\perp\rangle \mapsto |w_0\rangle \,/\, \||w_0\rangle\|$.

(Time-efficient) Implementation (I/II)

Three operations: (each is called $O(C(\mathcal{P}))$ times)

1. $2\Pi_{\mathcal{H}(x)} - I = \bigoplus_{e \in E}(2\Pi_{\mathcal{H}^e(x)} - I)$.
2. $2\Pi_{\mathcal{K}} - I = -(2\Pi_{\mathcal{C}_{G,r}} - I) \bigoplus_{e \in E}(2\Pi_{\mathcal{K}^e} - I)$.
3. $C_{|w_0\rangle} : |\perp\rangle \mapsto |w_0\rangle \,/\, \||w_0\rangle\|$.

Bottleneck: Implementation of $R_{\mathcal{C}_{G,r}} := 2\Pi_{\mathcal{C}_{G,r}} - I$.

(Time-efficient) Implementation (I/II)

Three operations: (each is called $O(C(\mathcal{P}))$ times)

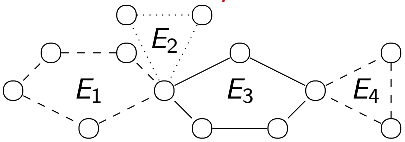
1. $2\Pi_{\mathcal{H}(x)} - I = \bigoplus_{e \in E}(2\Pi_{\mathcal{H}^e(x)} - I)$.
2. $2\Pi_{\mathcal{K}} - I = -(2\Pi_{\mathcal{C}_{G,r}} - I) \bigoplus_{e \in E}(2\Pi_{\mathcal{K}^e} - I)$.
3. $C_{|w_0\rangle} : |\perp\rangle \mapsto |w_0\rangle \,/\, \||w_0\rangle\|$.

Bottleneck: Implementation of $R_{\mathcal{C}_{G,r}} := 2\Pi_{\mathcal{C}_{G,r}} - I$.

Decompositions:

1. *Tree decomposition:* $\mathcal{C}_{G,r} = \bigoplus_{j=1}^{k} \mathcal{C}_{G|_{E_j}, r|_{E_j}}$.

Tree decomposition:



$E = E_1 \sqcup E_2 \sqcup E_3 \sqcup E_4$

(Time-efficient) Implementation (I/II)

Three operations: (each is called $O(C(\mathcal{P}))$ times)

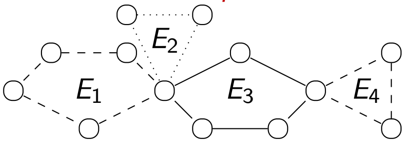
1. $2\Pi_{\mathcal{H}(x)} - I = \bigoplus_{e \in E}(2\Pi_{\mathcal{H}^e(x)} - I)$.
2. $2\Pi_{\mathcal{K}} - I = -(2\Pi_{\mathcal{C}_{G,r}} - I) \bigoplus_{e \in E}(2\Pi_{\mathcal{K}^e} - I)$.
3. $C_{|w_0\rangle} : |\bot\rangle \mapsto |w_0\rangle / \||w_0\rangle\|$.

Bottleneck: Implementation of $R_{\mathcal{C}_{G,r}} := 2\Pi_{\mathcal{C}_{G,r}} - I$.

Decompositions:

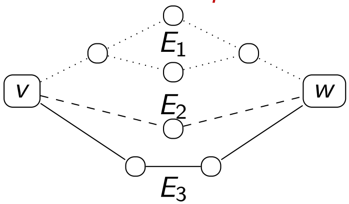
1. *Tree decomposition:* $\mathcal{C}_{G,r} = \bigoplus_{j=1}^k \mathcal{C}_{G|_{E_j}, r|_{E_j}}$.
2. *Parallel decomposition between v and w:*
   $\mathcal{C}_{G,r} = \bigoplus_{j=1}^k \mathcal{C}_{G|_{E_j}, r|_{E_j}} \oplus \mathcal{E}(\mathcal{C}^\perp)$, with
   1. $\mathcal{C} = \mathrm{Span}\{\sum_{j=1}^k \frac{1}{\||f^{\min}_{G|_{E_j}, v, w, r|_{E_j}}\rangle\|} |j\rangle\} \subseteq \mathbb{C}^k$.
   2. $\mathcal{E} : |j\rangle \mapsto \frac{|f^{\min}_{G|_{E_j}, v, w, r|_{E_j}}\rangle}{\||f^{\min}_{G|_{E_j}, v, w, r|_{E_j}}\rangle\|}$.

Tree decomposition:



$E = E_1 \sqcup E_2 \sqcup E_3 \sqcup E_4$

Parallel decomposition:



$E = E_1 \sqcup E_2 \sqcup E_3$

(Time-efficient) Implementation (I/II)

Three operations: (each is called $O(C(\mathcal{P}))$ times)

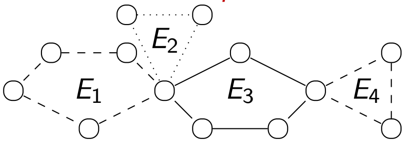
1. $2\Pi_{\mathcal{H}(x)} - I = \bigoplus_{e\in E}(2\Pi_{\mathcal{H}^e(x)} - I)$.
2. $2\Pi_{\mathcal{K}} - I = -(2\Pi_{\mathcal{C}_{G,r}} - I)\bigoplus_{e\in E}(2\Pi_{\mathcal{K}^e} - I)$.
3. $C_{|w_0\rangle} : |\perp\rangle \mapsto |w_0\rangle \,/\, \||w_0\rangle\|$.

Bottleneck: Implementation of $R_{\mathcal{C}_{G,r}} := 2\Pi_{\mathcal{C}_{G,r}} - I$.

Decompositions:

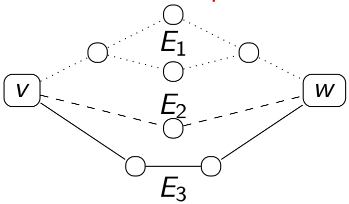
1. *Tree decomposition:* $\mathcal{C}_{G,r} = \bigoplus_{j=1}^k \mathcal{C}_{G|_{E_j},r|_{E_j}}$.
2. *Parallel decomposition between v and w:* $\mathcal{C}_{G,r} = \bigoplus_{j=1}^k \mathcal{C}_{G|_{E_j},r|_{E_j}} \oplus \mathcal{E}(\mathcal{C}^\perp)$, with
    1. $\mathcal{C} = \mathrm{Span}\{\sum_{j=1}^k \frac{1}{\||f^{\min}_{G|_{E_j},v,w,r|_{E_j}}\rangle\|}|j\rangle\} \subseteq \mathbb{C}^k$.
    2. $\mathcal{E} : |j\rangle \mapsto \frac{|f^{\min}_{G|_{E_j},v,w,r|_{E_j}}\rangle}{\||f^{\min}_{G|_{E_j},v,w,r|_{E_j}}\rangle\|}$.
3. Always possible to decompose.

Tree decomposition:



$E = E_1 \sqcup E_2 \sqcup E_3 \sqcup E_4$

Parallel decomposition:



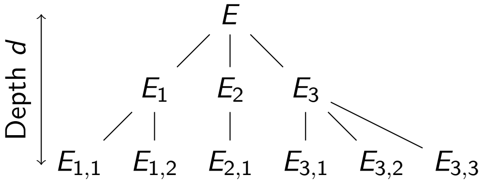
$E = E_1 \sqcup E_2 \sqcup E_3$

(Time-efficient) Implementation (II/II)

(Time-efficient) Implementation (II/II)

1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\dots,j_d} = \{e\}$.

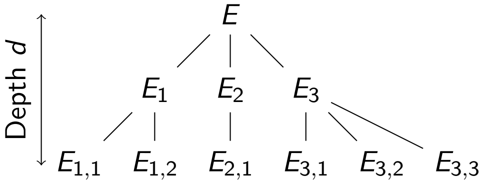
Tree-parallel decomposition tree



(Time-efficient) Implementation (II/II)

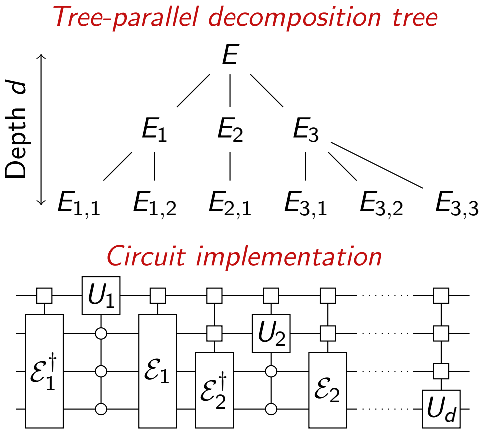
1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\dots,j_d} = \{e\}$.
   2. Embed $|e\rangle = |j_1, j_2, \dots, j_d\rangle$.

Tree-parallel decomposition tree



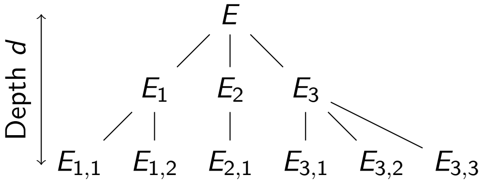
(Time-efficient) Implementation (II/II)

1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\ldots,j_d} = \{e\}$.
   2. Embed $|e\rangle = |j_1, j_2, \ldots, j_d\rangle$.
2. *Circuit implementation of $R_{\mathcal{C}_{G,r}}$:*

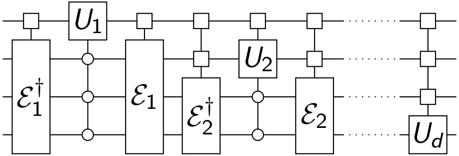


1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\dots,j_d} = \{e\}$.
   2. Embed $|e\rangle = |j_1, j_2, \dots, j_d\rangle$.
2. *Circuit implementation of $R_{\mathcal{C}_{G,r}}$:*
   1. Every $\mathcal{E}_j$ is a state-preparation operation.

Tree-parallel decomposition tree



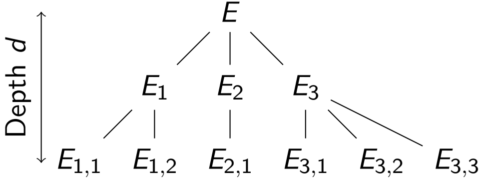
Circuit implementation



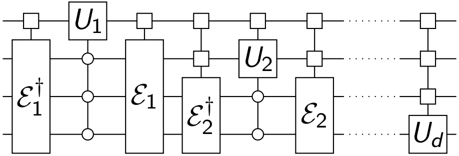
(Time-efficient) Implementation (II/II)

1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\dots,j_d} = \{e\}$.
   2. Embed $|e\rangle = |j_1, j_2, \dots, j_d\rangle$.
2. *Circuit implementation of $R_{\mathcal{C}_{G,r}}$:*
   1. Every $\mathcal{E}_j$ is a state-preparation operation.
   2. Every U_j is:
      1. Identity for a tree decomposition.
      2. Reflection through a 1D subspace for parallel decomposition.

Tree-parallel decomposition tree



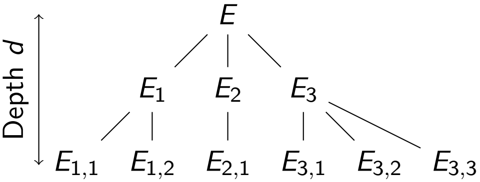
Circuit implementation



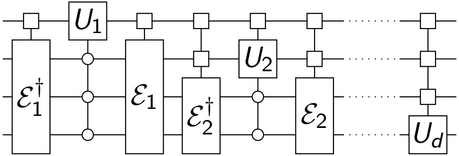
(Time-efficient) Implementation (II/II)

1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\dots,j_d} = \{e\}$.
   2. Embed $|e\rangle = |j_1, j_2, \dots, j_d\rangle$.
2. *Circuit implementation of $R_{\mathcal{C}_{G,r}}$:*
   1. Every $\mathcal{E}_j$ is a state-preparation operation.
   2. Every U_j is:
      1. Identity for a tree decomposition.
      2. Reflection through a 1D subspace for parallel decomposition.
   3. Both can be implemented with KP-trees:
      1. $\widetilde{O}(\log|E|)$ time,
      2. $\widetilde{O}(|E|)$ bits of QROM.

Tree-parallel decomposition tree



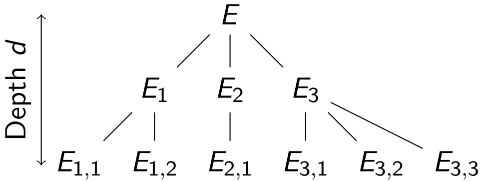
Circuit implementation



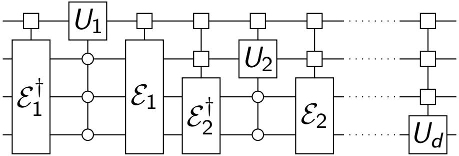
(Time-efficient) Implementation (II/II)

1. *Tree-parallel decomposition tree:*
   1. Every leaf is a single edge, i.e., $E_{j_1,\ldots,j_d} = \{e\}$.
   2. Embed $|e\rangle = |j_1, j_2, \ldots, j_d\rangle$.
2. *Circuit implementation of $R_{\mathcal{C}_{G,r}}$:*
   1. Every $\mathcal{E}_j$ is a state-preparation operation.
   2. Every U_j is:
      1. Identity for a tree decomposition.
      2. Reflection through a 1D subspace for parallel decomposition.
   3. Both can be implemented with KP-trees:
      1. $\widetilde{O}(\log|E|)$ time,
      2. $\widetilde{O}(|E|)$ bits of QROM.
   4. *Total cost:*
      1. $\widetilde{O}(d\log|E|)$ time,
      2. $\widetilde{O}(d|E|)$ bits of QROM.

Tree-parallel decomposition tree

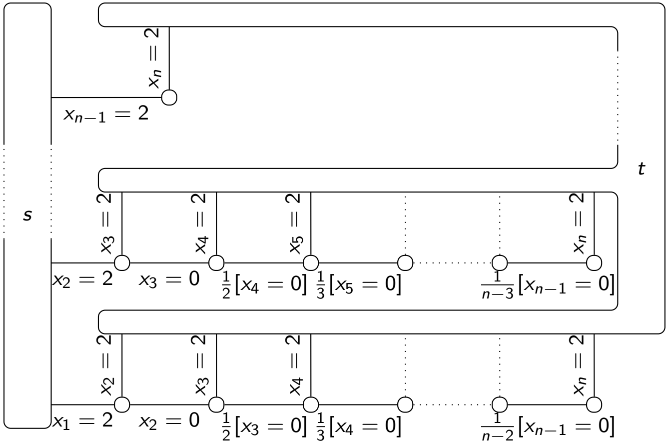


Circuit implementation



Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

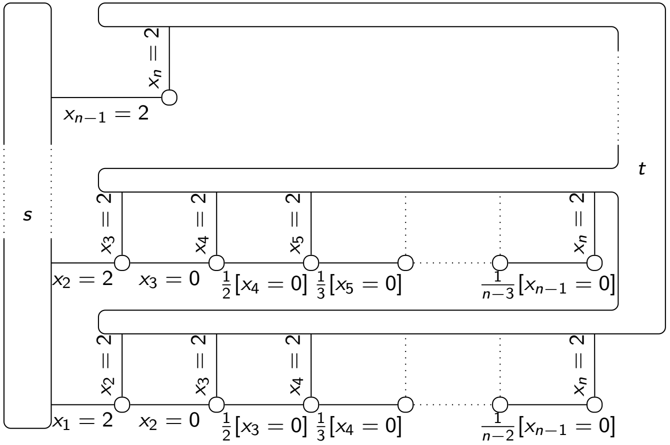
Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem



Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

Analysis:

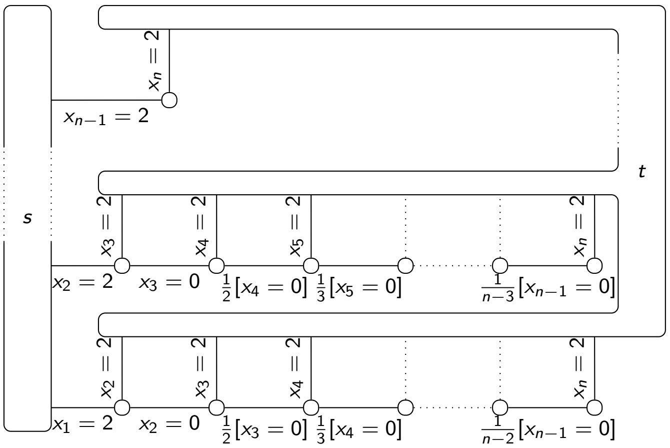
1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.



Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

Analysis:

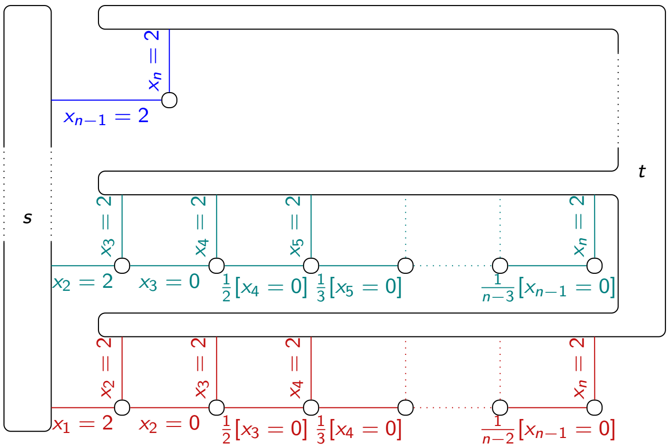
1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.



Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

Analysis:

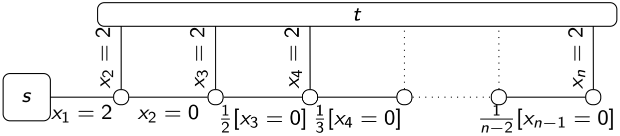
1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.



Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

Analysis:

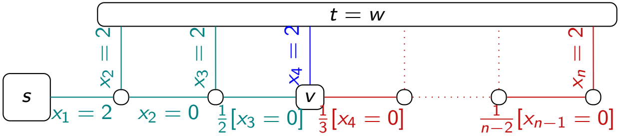
1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.



Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

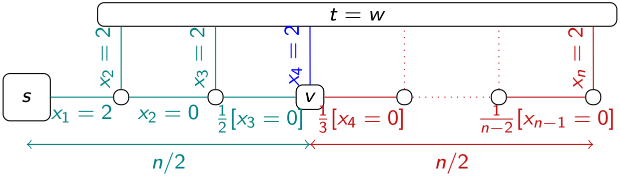
Analysis:

1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.



Analysis:

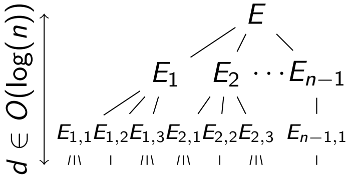
1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.

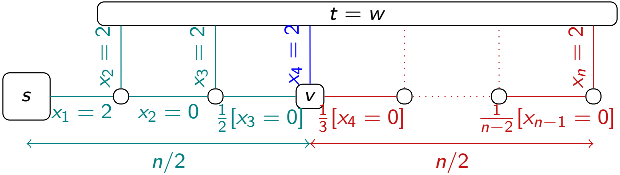


Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

Analysis:

1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.
3. *Tree-parallel decomposition:*

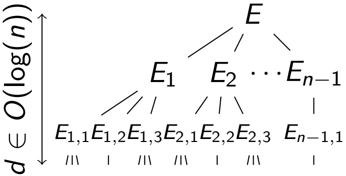


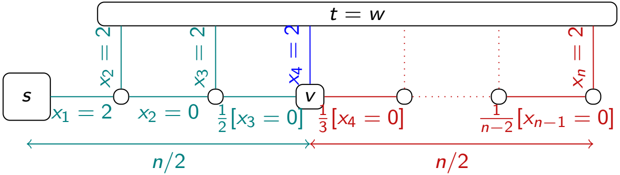


Example: Time-efficient implementation of the $\Sigma^*20^*2\Sigma^*$-problem

Analysis:

1. $C(\mathcal{P}) \in O(\sqrt{n\log(n)})$.
2. $|E| \in O(n^2)$.
3. *Tree-parallel decomposition:*





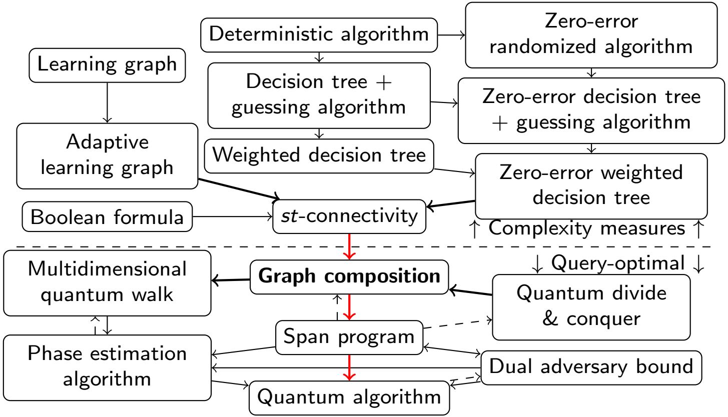
Total cost:

1. $O(\sqrt{n\log(n)})$ queries.
2. $\widetilde{O}(\sqrt{n})$ time.
3. $\widetilde{O}(n^2)$ bits of QROM. (Further ad-hoc improvements possible).

Summary (I/II)

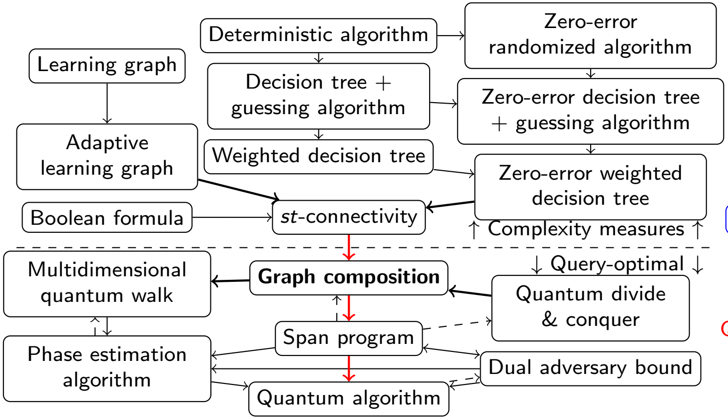
Summary (I/II)

Relations between algorithmic frameworks

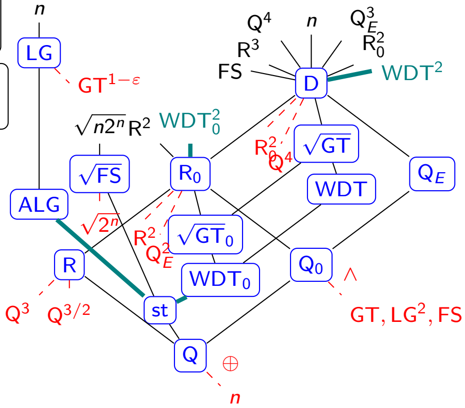


Summary (I/II)

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.
3. *Time-efficient implementation:*
   1. Tree-parallel decomposition.
   2. Efficient implementation using QROM.

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. *Time-efficient implementation:*
   1. Tree-parallel decomposition.
   2. Efficient implementation using QROM.

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.
3. *Time-efficient implementation:*
   1. Tree-parallel decomposition.
   2. Efficient implementation using QROM.

Examples:

1. In this talk:
   1. 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.
3. *Time-efficient implementation:*
   1. Tree-parallel decomposition.
   2. Efficient implementation using QROM.

Examples:

1. In this talk:
   1. 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. *Time-efficient implementation:*
   1. Tree-parallel decomposition.
   2. Efficient implementation using QROM.

Examples:

1. In this talk:
   1. 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.