Catalytic Computing Between L and P

Ian Mertz
December 6, 2019
What is catalytic computing?

input:
0 1 0 0 1 1 0 0 0 1 1 0

machine

output:

work:
1 1 1 1 1 1 1 1
What is catalytic computing?

**Input**

```
0101010101010
```

**Machine**

```

```

**Output**

```
```

**Work**

```
111111
```

**Catalytic**

```
00010110010101111110110
```
Tree Evaluation Problem

```
    O
   /|
  / |\
 /  | \
 O   O
 |   |  
|   |   
5  2  1  3  2  7  3  6
```

$\text{TEP}_{h,k}$

$h = 3$

$k = 7$
Pebbling
Pebbling
Pebbling
Pebbling
Pebbling

$\Omega(h \cdot \log k)$

$O(h + \log k)$
Restrictions (read-once)
Restrictions (thrifty)
Barrington's Theorem
Barrington's Theorem

$NC'$

$h = O(\log n)$
Barrington's Theorem

$NC^1$ can be computed by a layered width 5 poly-size deterministic branching program.
Ben-Or/Cleve '92

$\#NC'(R)$

$h = O(\log n)$

$R = F_7$

Diagram with nodes labeled 5, 2, 1, 3, 2, 7, 3, 6.
Ben-Or/Cleve '92

Register program $P(1, 2 \ldots s)$

1. $R_1 \leftarrow R_1 + x_1 \cdot R_2$
2. $R_4 \leftarrow R_4 + R_2 \cdot R_3$
3. $R_1 \leftarrow R_1 + 8 \cdot x_3$
$\#NC'(R)$ can be computed by a poly-size register program over $R$ with 3 registers.
Ben-Or/Cleve '92

Ω(\log n)
Ben-Or/Cleve '92

Proof:
Ben-Or/Cleve '92

Proof:

\[ P(1, 2, 3) \]
\[ 1. R_1 \leftarrow R_1 + 3R_2 \]
Ben-Or/Cleve'92

Proof:

\[ P(1, 2, 3) \]

1. \( P_i(1, 2, 3) \)

\[ R_i \leftarrow R_i + f_i \cdot R_2 \]
Ben-Or/Cleve '92

Proof:

1. $P_1(1, 2, 3)$

2. $P_2(1, 2, 3)$

$$R_1 \gets (R_1 + f_1 \cdot R_2) + f_2 \cdot R_2$$
Ben-Or/Cleve '92

Proof:

\[ P(1, 2, 3) \]

1. \[ P_i(1, 3, 2) \]

\[ R_i \leftarrow R_i - f_i \cdot R_3 \]
Ben-Or/Cleve '92

Proof:

\[ P(1, 2, 3) \]
1. \[ P_1(1, 3, 2) \]
2. \[ P_2(3, 2, 1) \]

\[ R_3 \leftarrow R_3 + f_2 \cdot R_2 \]
Proof:

\[ P(1, 2, 3) \]
1. \[ P_1(1, 3, 2) \]
2. \[ P_2(3, 2, 1) \]
3. \[ P_1(1, 3, 2) \]

\[ R_i \leftarrow (R_i - f_i \cdot R_3) + f_i \cdot (R_3 + f_2 \cdot R_2) \]
Ben-Or/Cleve '92

Proof:

\[ P(1, 2, 3) \]
1. \[ P_1^{-1}(1, 3, 2) \]
2. \[ P_2(3, 2, 1) \]
3. \[ P_1(1, 3, 2) \]
4. \[ P_2^{-1}(3, 2, 1) \]
Catalytic + TEP

#NC'(R) can be computed by a poly-size register program over R with 3 registers.
Catalytic + TEP

Given register programs $P_1$, $P_2$, where $P_i$ transparently computes $f_i$, there exists a register program $P$ which transparently computes $f_1 \cdot f_2$ using the same space and at most 4 program calls.
Catalytic + TEP

Given register programs $P_i$, $P_2$, where $P_i$ transparently computes $F_i$, there exists a register program $P$ which transparently computes $\sum_{j,k} F_{i,j} \cdot f_{2,k}$ using the same space and at most 4 program calls.
Catalytic + TEP

Given register programs $P_1, \ldots, P_d$ where $P_i$ transparently computes $f_i$, there exists a register program $P$ which transparently computes $\prod_{i=1}^d f_i$ using the same space and at most $d^2$ program calls.
Catalytic + TEP

Main Theorem [Buhrman + '14]:
Catalytic Logspace contains \( TC' (\geq NL) \).
Catalytic + TEP

Main Theorem [Cook-Mertz '19]:

$\text{TEP}_{h,k}$ can be simulated by a register program with $(\frac{k}{eh}+1)^{2h}$ instructions and $(\frac{k}{eh}+1)^{3eh}$ boolean registers.
Catalytic + TEP

\((2 + 3\varepsilon) h \cdot \log \left( \frac{k}{\varepsilon h} + 1 \right) \) vs \(h \cdot \log k\)
Catalytic + TEP

\((2 + 3\varepsilon) h \cdot \log \left( \frac{k}{\varepsilon h} + 1 \right) \text{ vs } h \cdot \log k\)

- better for \( h = \omega(k^{\theta + \varepsilon}) \)
Catalytic + TEP

\[(2+3\varepsilon) h \cdot \log \left( \frac{k}{e^h+1} \right) \text{ vs } h \cdot \log k\]

- better for \( h = \omega(k^{1/2+\varepsilon}) \)
- not read once

1. \( P_1^* (1,3,2) \)
2. \( P_2 (3,2,1) \)
3. \( P_1 (1,3,2) \)
4. \( P_2^* (3,2,1) \)
Catalytic + TEP

\[(2 + 3\epsilon) h \cdot \log \left(\frac{k}{eh} + 1\right) \text{ vs } h \cdot \log k\]

- better for \( h = \omega(k^{\frac{1}{2}} + \epsilon) \)
- not read-once or thrifty

\[
[f = x] = \sum [f(y_2) = x] [f = y][f = z]
\]
Open problems
Open problems

1) \( P \) computes \( \sum_{i} \prod_{j} f_{i,j} \) using \( t(n) < 2^n \) program calls

\[ \text{poly}(n) \Rightarrow \text{TEP} \in (\log h)^{O(h)} \]

\[ O(1) \Rightarrow \text{TEP} \in L \]
Open problems

1) $P$ computes $\sum_j \prod_i f_{i,j}$ using $t(n) < 2^n$ program calls

2) use lemmas to simulate - $AC^1$ in non-catalytic $L$
Open problems

1) $P$ computes $\sum \prod f_{i,j}$ using $t(n) < 2^n$ program calls

2) use lemmas to simulate
   - $\text{AC}^1$ in non-catalytic $L$
   - $\text{TC}^1$ in $VP$