# Local Reductions

September 2014

Emanuele Viola

Northeastern University

Papers:

Local Reductions
with Hamid Jahanjou and Eric Miles

Succinct and explicit circuits for sorting and connectivity
With Hamid Jahanjou and Eric Miles

Short PCPs with projection queries
With Eli Ben-Sasson

Theorem [Cook, Levin]: 3SAT is NP-complete

Theorem [Cook, Levin]: 3SAT is NP-complete

$\forall$ M $\in$ NTIME(t) $\exists$ reduction R : $\forall$ x

R(x) = $\varphi \in$ 3SAT $\leftrightarrow$ M(x) = 1
R runs in time poly(t) (t = poly(n), t = $2^n$ etc.)

Applications require to optimize (by themselves or both)

- | $\varphi$ |

- “Complexity” of R

- Optimizing | φ | (70s - 80s)
[…, Pippenger Fischer, Gurevich Shelah,...]

$| \varphi | = t \log^{O(1)} t$

- Optimizing complexity of R.

If reduction has resources polynomial in t,
it is almost trivial

Our focus: resources << t

## Clause-explicit R

$R(i,x)$ = i-th clause of $\varphi$ , e.g. $( y_{15} \vee \neg y_7 \vee \neg y_8 )$

$|i| = \log |\varphi|$

We will ignore x and focus on the map as a function of i, though dealing with x is not easy.

Why care about explicitness?

# Explicit R

- Succint-sat NEXP complete

  $t = 2^n$ , $|\varphi| = \text{poly}(t)$, R(i) run in time poly(|i|)

- Lower bounds for SAT

[Van Melkebeek, Fortnow, Lipton, Vigas]

$t = \text{poly}(n)$, $|\varphi| = t \log^{O(1)} t$, R(i) in time poly(|i|), space O|i|

- Williams lower bounds from SAT/derandomization

Lower bound against C (e.g., C = $ACC^0$ ), can use

  $t = 2^n$ , $|\varphi| = t \log^{O(1)} t$, R(i) computable by C

# Explicit R

$| \varphi | = \text{poly}(t), \; R \in AC^0$
[Arora Steurer Wigderson] (or folklore)

$| \varphi | = t \log^{O(1)} t, \; R \in NC^1$
[Ben-Sasson, Goldreich, Harsha, Sudan, Vadhan] (2005)

Note: Williams $ACC^0$ lower bound uses workaround due to absence of more efficient reductions.

More efficient reductions "hard (perhaps impossible)"

Consequent drawbacks to be discussed shortly

Theorem [Jahanjou Miles V.]
Reduce NTIME(t) to 3SAT via reduction R :

- $|\varphi| = t \log^{O(1)} t$

- Each output bit of R(i) depends on O(1) bits of i.
  (A.k.a. local, $NC^0$, junta).

Note: $R(i) = ( y_{15} \vee \neg y_7 \vee \neg y_8 )$

$|y_{15}| = \log t = |i|$ bits; each bit depends on O(1) bits of i.

Note: Local R cannot even compute $i \rightarrow i+1$

# Outline

Intro

Consequences of local reductions

Proof of local reductions

PCP reductions

## Warm-up consequence:

SUCCINCT-3SAT, SUCCINCT-3COLOR, etc. remain NEXP complete even on instances represented by $NC^0$ circuits

Slightly better $ACC^0$ lower bound

## Consequence: Tighter connection between SAT algorithms and lower bounds

NOTE: “lower bound” throughout means for $f \in NEXP$ or $E^{NP}$

[W] gives lower bounds against size s, depth d from SAT algorithm for size $s^c$ , depth $c\, d$

We only require SAT algorithm for size $c\, s$, depth $d + c$.

This (and refinements) gives several new connections for classes of interest:

## For each, new lower bound from SAT algorithm.

- Linear-size circuits
- Linear-size log-depth circuits [Valiant 1977]
- Linear-size series-parallel circuits [Valiant 1977]
- Quasi-polynomial SYM-AND circuits

These can be related to assumptions about kSAT

- [W] Exponential-time hypothesis [Impagliazzo Paturi] false

  => linear-size circuits lower bound

Our proof from previous result: Apply Cook-Levin. ■

- [JMV] Strong Exponential-time hypothesis false

  => linear-size series-parallel circuits lower bound

- [JMV] $n^c$ - SAT in time $2^{n - \omega(n/\log\log n)}$

  => linear-size log-depth circuits lower bound

# Some tighther results [Ben-Sasson V., JMV]

- Unbounded-depth circuits:
  Lower bound for depth d <= SAT for depth d+1.

- Recall for general circuits a 3n lower bound is unknown.

3n lower bound from 3SAT in TIME$(1.07)^n$

non-boolean 3n lower bound from 3SAT in TIME$(1.10)^n$

Record: TIME$(1.34)^n$

## Do we simplify the proof [W] that NEXP is not in $ACC^0$ ?

- Recall that [W] uses as black-box previous reductions

- If instead use as black-box ours, the proof is more direct.

- In fact, for this application it suffices $R \in AC^0$
  Much easier to establish.

Independently, Kowalski and Van Melkebeek proved $R \in AC^0$

# Outline

Intro

Consequences of local reductions

Proof of local reductions

PCP reductions

# Background

We reduce NTIME(t) to CIRCUIT-SAT C :

(1) $| C | = t \log^{O(1)} t$

(2) Given index i to gate, R(i) outputs type, and children with constant locality

Pippenger Fischer oblivious simulation gives (1), but (2) hard

Use alternative [Van Melkebeek], based on sorting networks
(The idea of sorting is from Gurevich Shelah)

Strangely little known!?

Rediscovered by "mini-poly-math" class project at NEU

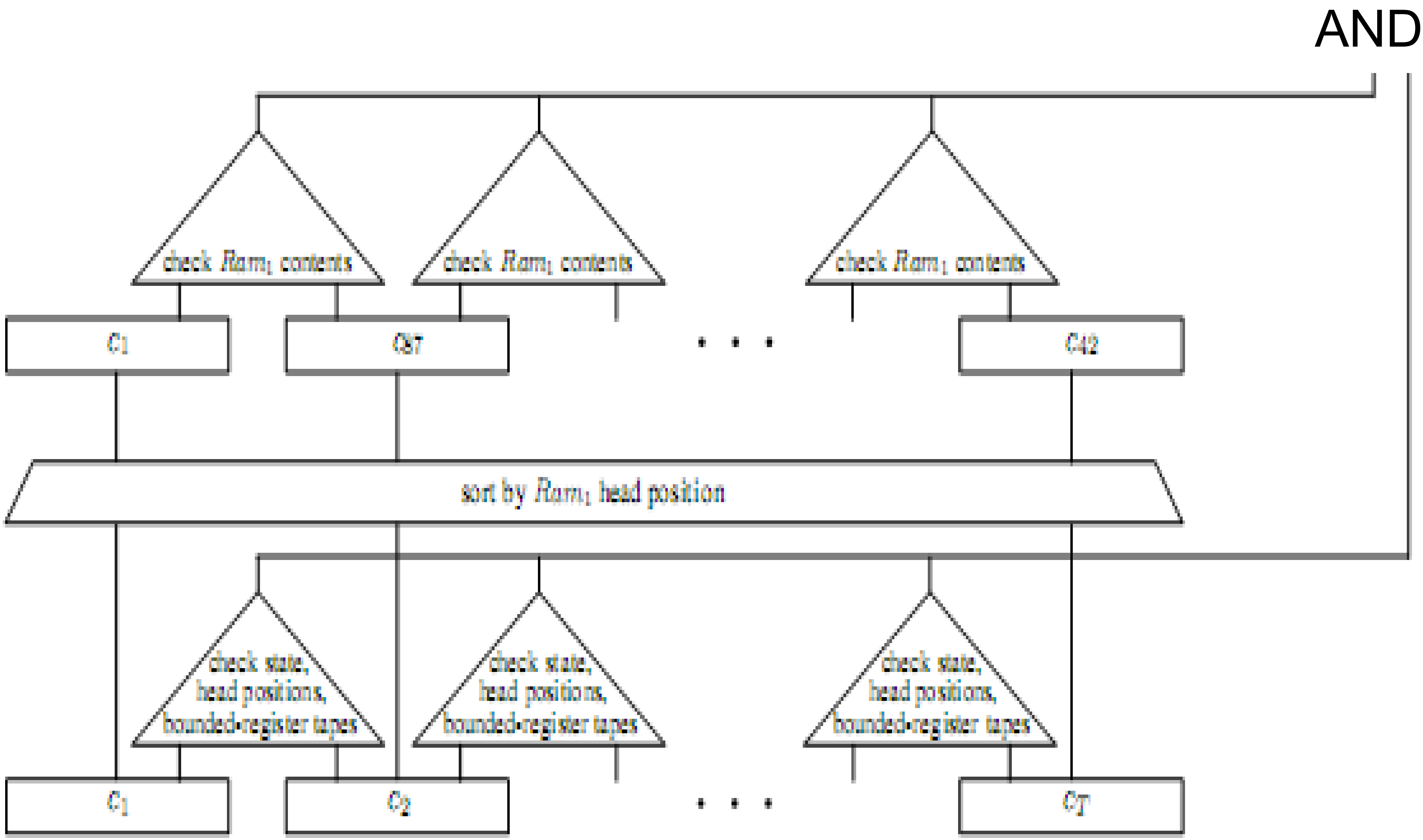



Figure 1: Each of the $T$ configurations has size $O(\log T)$. The checking circuits have size $\text{poly}\log T$. The sorting circuits have size $\tilde{O}(T)$. $k$ is a constant. Hence overall circuit has size $\tilde{O}(T)$.


Figure 1: Each c [illegible] g [illegible] s size $O($[illegible]$T)$. The checking circuits have size poly log $T$. The s [illegible] cuits have [illegible] ze $\tilde{O}(T)$. $k$ is a constant. Hence overall circuit has size $\tilde{O}(T)$.

Sorting network.

This can be done quite efficiently, but O(1) locality unknown
[Separate write-up, all that you need for $AC^0$ reduction]

For constant locality, we instead use routing networks,
as in PCP literature since Polischuck and Spielman

With De Buijin graphs, computation very simple:
children of **i** are

**i** XOR CONSTANT

(**i** rotated) XOR constant

Check circuits:

Easy to obtain R running in linear space (= log |C| space).

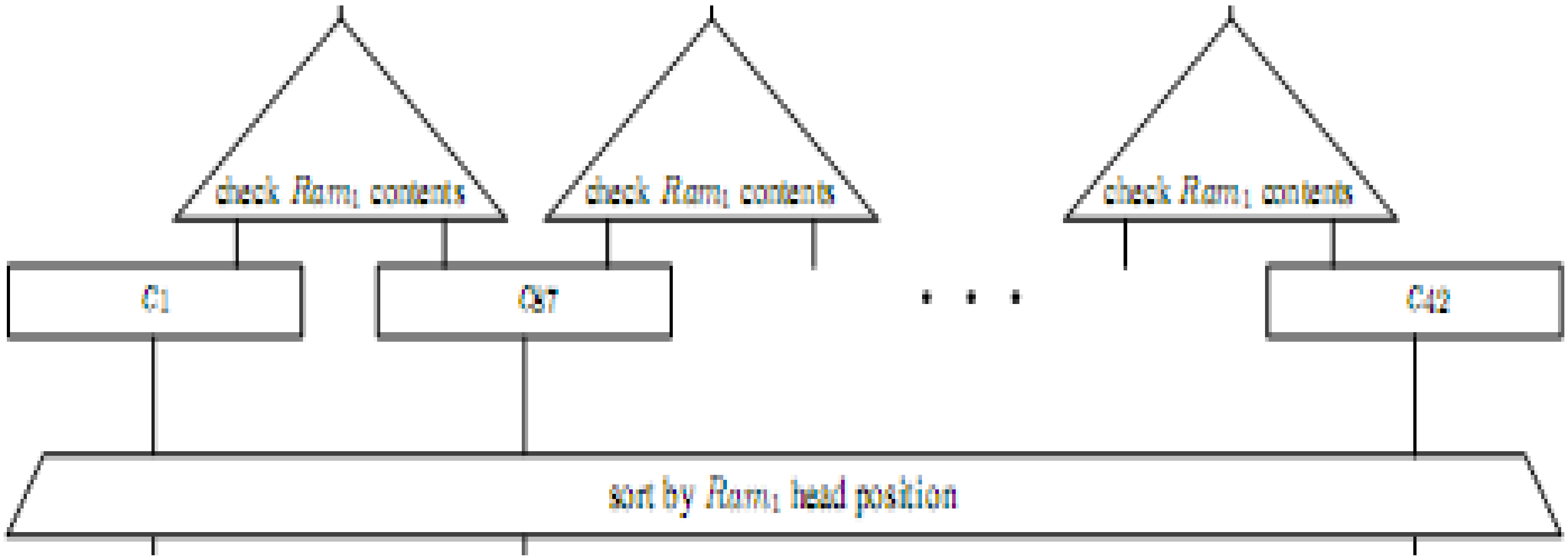
Theorem [JMV] For every C with linear-space R
there is equivalent C', | C' | = poly |C|, with local R

Technique [Ruzzo]
New gates of C' are configurations of linear-space R.

But Ruzzo does not aim or prove constant locality.

Obtaining that is not trivial, as you can't check if a configuration is valid.

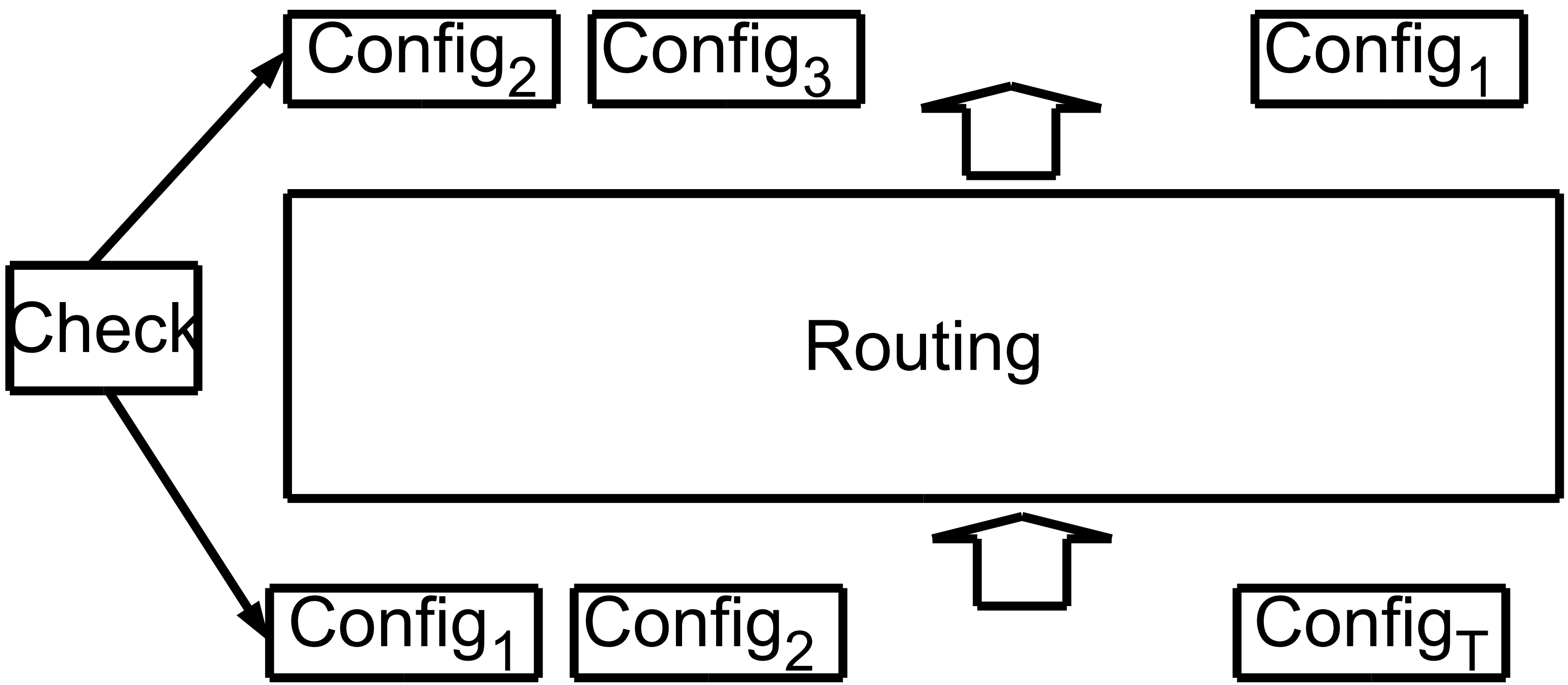
Problem: Given index to i-th configuration, need to compute index to (i+1) configuration




Recall you cannot even compute i → i+1

Problem: Given index to i-th configuration, need to compute index to (i+1) configuration

Solution: Use routing networks in a different way. Instead of output of network being sorted order, it will be "successor" function.



# Outline

Intro

Consequences of local reductions

Proof of local reductions

PCP reductions

> 10-year old problem:

## MAX-3SAT in time $2^n / n^{\omega(1)}$ ?

Equivalently, SAT of $\text{MAJ-AND}_3$ circuits

Bottleneck for Williams' approach based on SAT algorithms.
Needed for $TC^0$ , threshold of threshold, etc.

Note: This is for size $n^3$ , much of what we saw earlier was for size $O(n)$.

Derandomization comes to rescue.

MAJ-AND$_3$ and some other classes, can be derandomized.

This suffices for lower bounds [W], using PCP reductions.

Same considerations made earlier about Cook-Levin:

1) more efficient reduction => tighter connection

2) [W, Santhanam W] need workaround due to INefficiency of reductions.

- [Ben-Sasson, Goldreich, Harsha, Sudan, Vadhan]

Explicit PCP with $t \log^{O(1)} t$ constraints, many queries

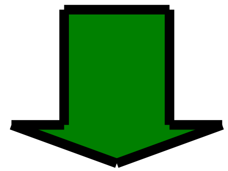
- [Mie] Improves queries to O(1).

Theorem: [Ben-Sasson V.]
Variant of [BGHSV] PCP:
given index to constraint, variables (a.k.a. queries) are projections.
Postprocess is a CNF [easy]

Note: Projection queries were used in concurrent [W] lower bound for $AC^0$ SYM from #SAT. By above enough to derandomize (or SAT)

Consequence:

Derandomizing (unbounded fan-in) depth d+2 circuits



lower bound for depth d

Example: depth-2 threshold lower bound still open.

## Question:

Improve number of queries to O(1), matching [Mie]

How efficient PCP reductions? Constant locality?