Some classes between NC^1 and LogCFL

Meena Mahajan

The Institute of Mathematical Sciences

Chennai, India

meena@imsc.res.in

joint work with

Nutan Limaye, Antoine Meyer, B. V. Raghavendra Rao

Circuit Classes: NC

NC: families of circuits

- polynomial size
- polylogarithmic depth (short circuits)
  ($O(\log^i n)$ for NC^i)
- AND_2 and OR_2 gates
- $0, 1, x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}$ at leaves.

NC = efficiently parallelizable problems in P.

NC^1 = Alternating Logarithmic Time

Circuit Classes: SC

SC: families of circuits

- polynomial size
- polylogarithmic width (thin circuits) ($O(\log^i n)$ for SC^i)
- AND_2 and OR_2 gates
- $0, 1, x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}$ at leaves.

SC = problems in P with polylog-space (and simultaneously poly time) algorithms.

SC^1 = Deterministic Log Space

Branching Programs

Branching programs:
layered directed graphs

- polynomial size
- edges labeled from $1, x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}$.
- designated start and finish nodes.

Equivalent to Skew circuits:
every AND gate has at most one non-leaf input.

BWBP: Constant-width skew circuits

Complexity Classes

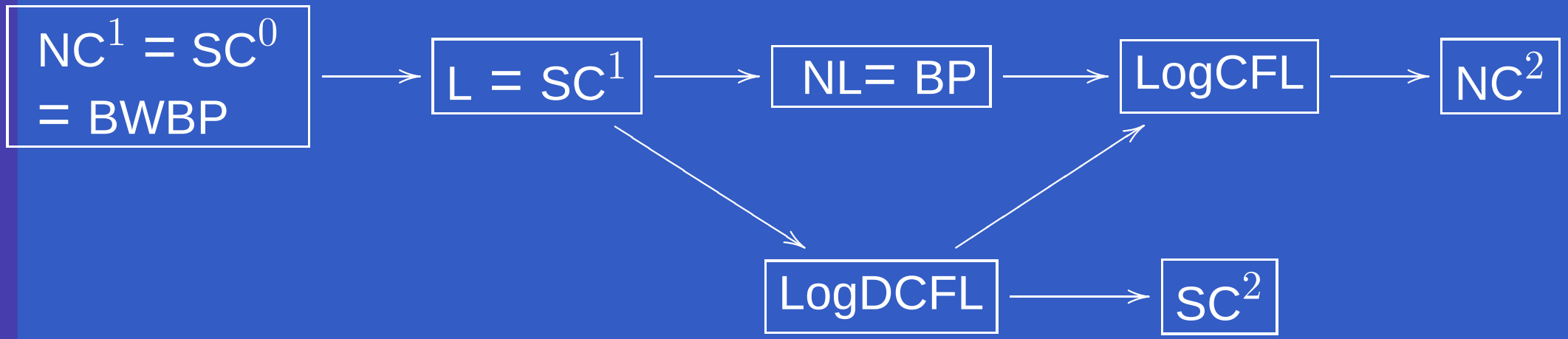
L: deterministic log space

NL: non-deterministic log space

LogDCFL: L-reducible to some deterministic context-free language DCFL

LogCFL: L-reducible to some context-free language CFL

The landscape between NC^1 and NC^2



Barrington
Sudborough
Ruzzo
Venkateswaran

Cook

Degree of a circuit

- Algebraic degree of associated polynomial
  - Degree of a leaf = 1.
  - Degree of $\text{OR}(g_1, g_2)$ = $\max\{$ Degree(g_1), Degree(g_2) $\}$
  - Degree of $\text{AND}(g_1, g_2)$ = Degree(g_1) + Degree(g_2)
  - Degree of circuit = max degree of any node in it.
- Degree of SC^0 circuits can be exponential.
- Degree of NC^1 circuits bounded by a polynomial.
- Degree of BP bounded by a polynomial.
- LogCFL equals polynomial size circuits of polynomial degree equals SAC^1. *Ruzzo,Venkateswaran*

Restricting the degree

- Define small SC:
  sSC^i = SC^i circuit of polynomial degree

Restricting the degree

- Define small SC:
  sSC^i = SC^i circuit of polynomial degree
- sSC is in LogCFL.

Restricting the degree

- Define small SC:
  sSC^i = SC^i circuit of polynomial degree
- sSC is in LogCFL.
- BWBP $\subseteq$ sSC^0 $\subseteq$ SC^0 $\subseteq$ BWBP
  i.e. degree bound not a restriction for SC^0.

Restricting the degree

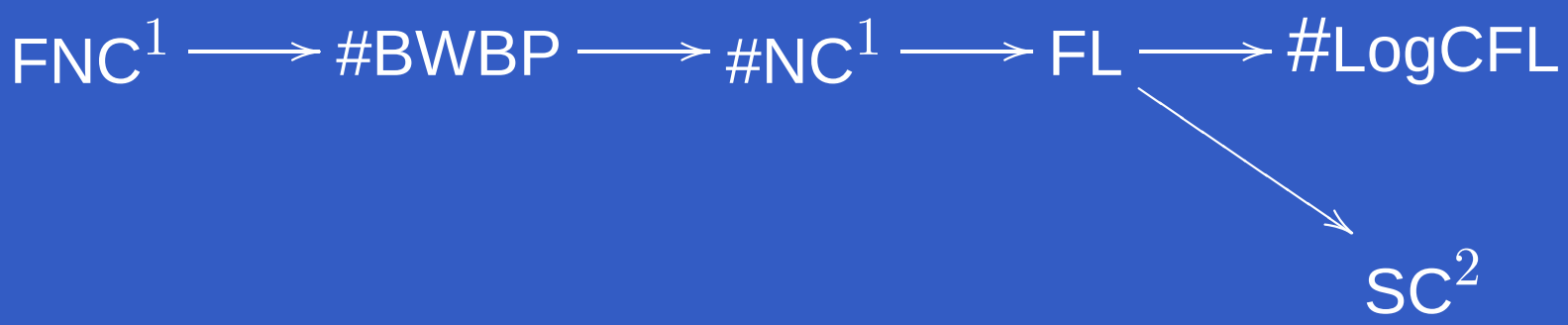
- Define small SC:
  sSC^i = SC^i circuit of polynomial degree
- sSC is in LogCFL.
- $\mathrm{BWBP} \subseteq \mathrm{sSC}^0 \subseteq \mathrm{SC}^0 \subseteq \mathrm{BWBP}$
  i.e. degree bound not a restriction for SC^0.
- $\mathrm{sSC}^0 \subseteq \mathrm{sSC}^1 \subseteq \mathrm{SC}^1$
  i.e. sSC^1 is sandwiched between NC^1 and L.

Counting classes

- Arithmetizing a circuit: Replace
  - every AND gate by a $\times$ gate;
  - every OR gate by a $+$ gate;
  - leaf-level negation $\overline{x_i}$ by $1 - x$.
- Arithmetizing a branching program: Count the number of $s \longrightarrow t$ paths.
- Arithmetizing a nondeterministic class: Count the number of accepting computations.

Relationships among counting classes

Equivalences do not necessarily carry over to arithmetizations.



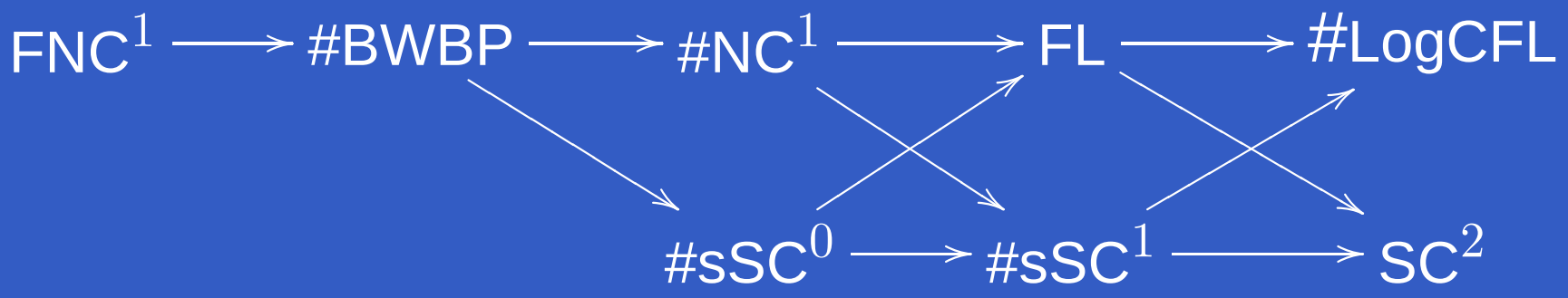
Barrington

Caussinus, McKenzie, Thérien, Vollmer

Allender & Chiu, Davida, Litow

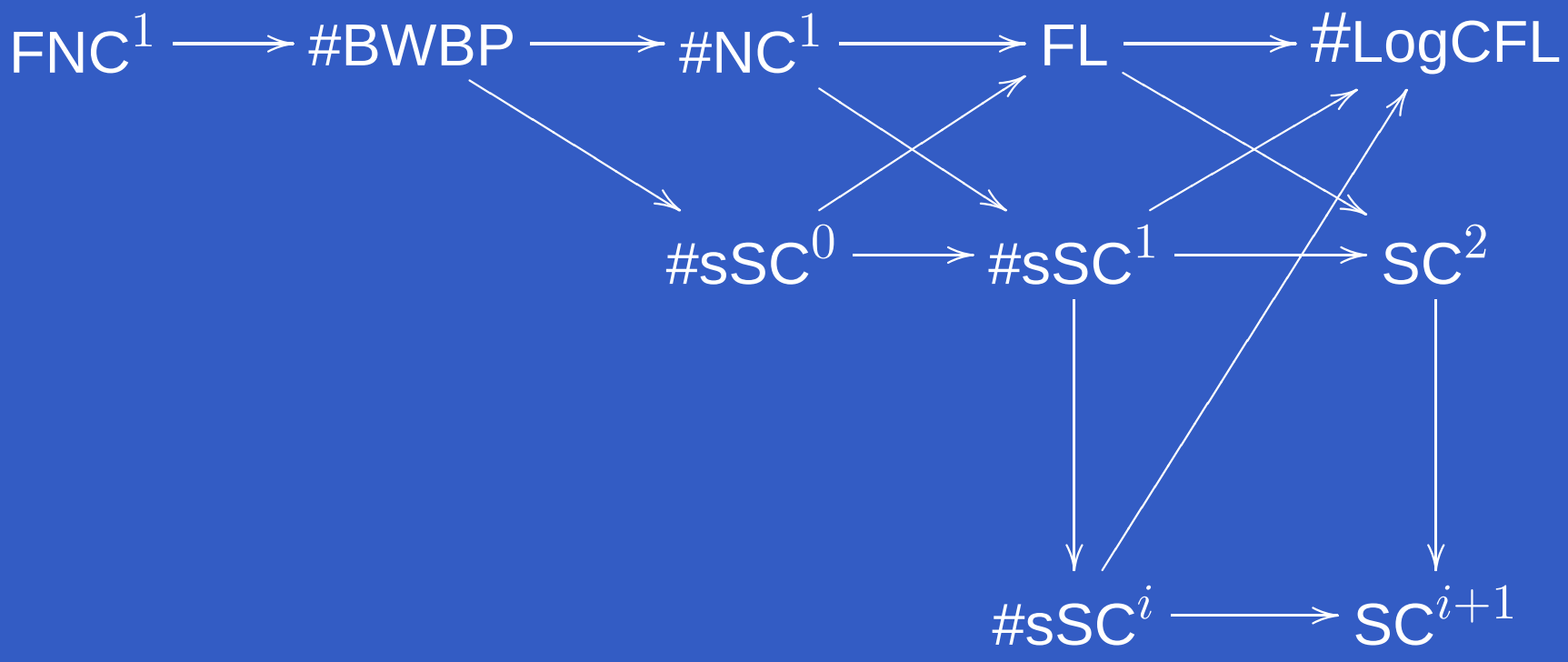
Relationships among counting classes

Equivalences do not necessarily carry over to arithmetizations.

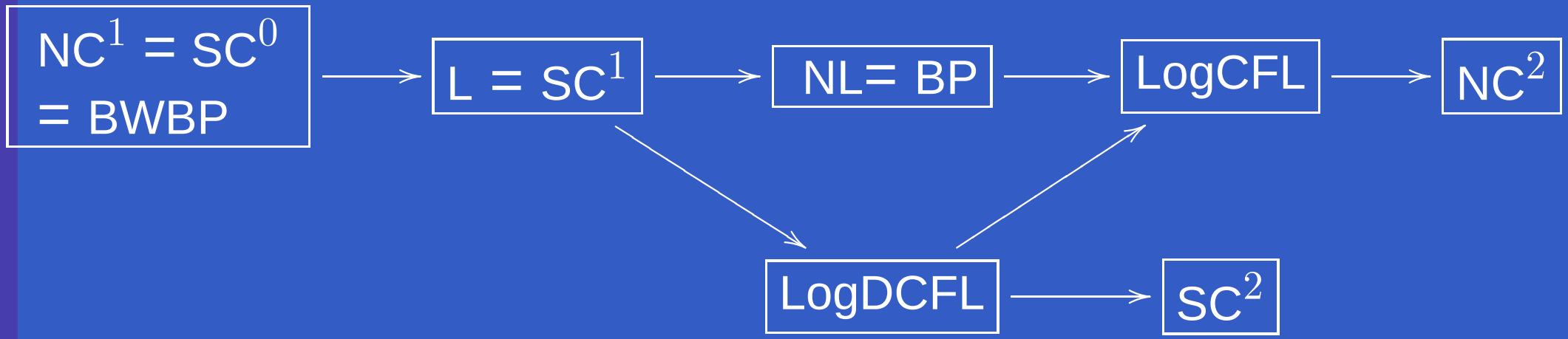


Relationships among counting classes

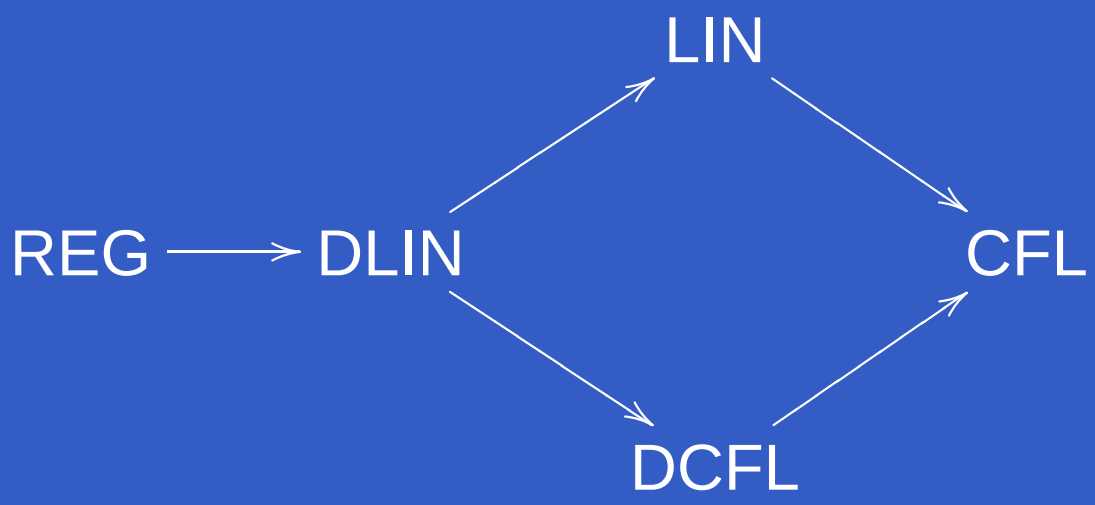
Equivalences do not necessarily carry over to arithmetizations.



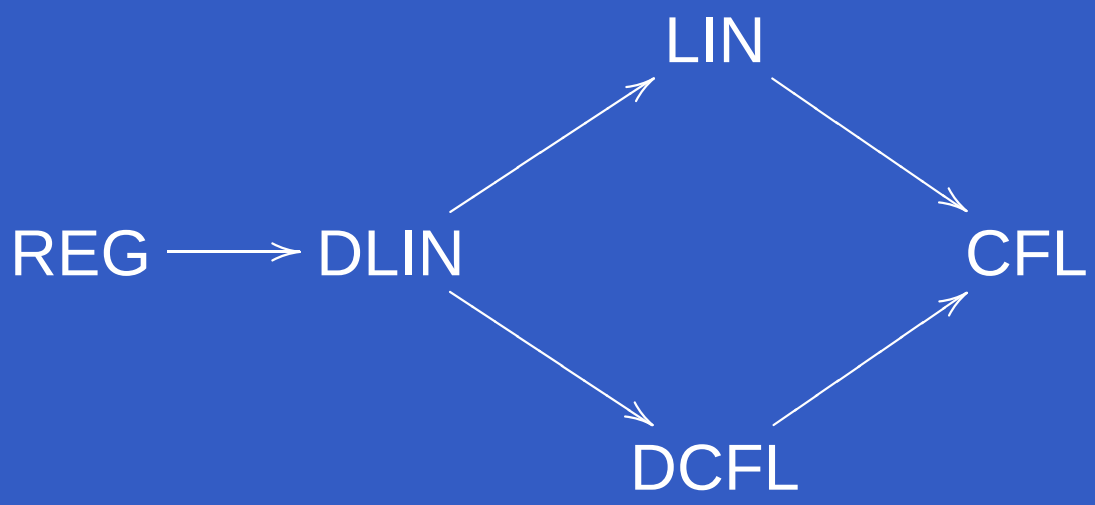
The landscape between NC^1 and NC^2



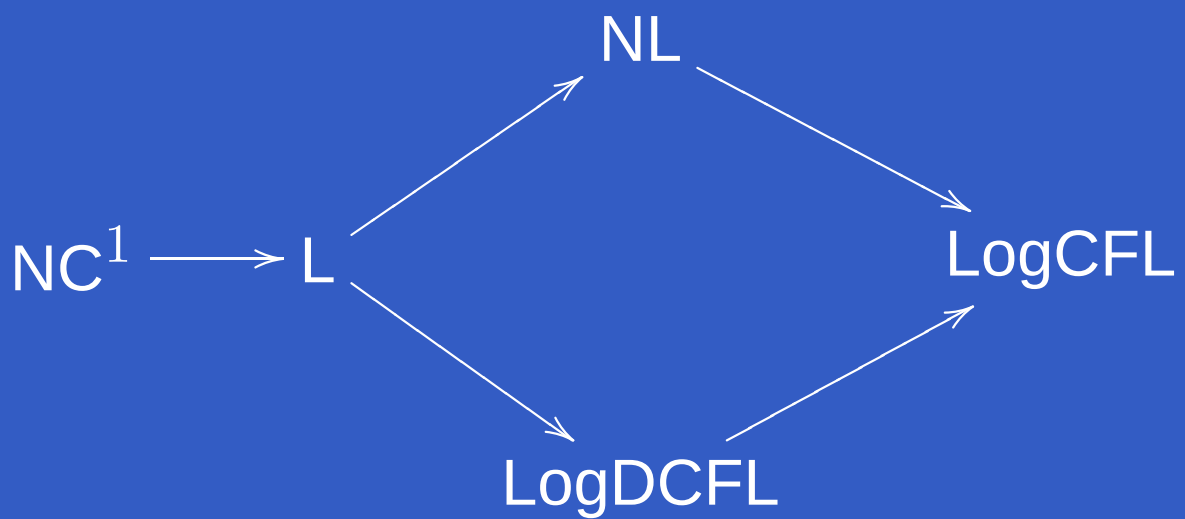
A language-theoretic view



A language-theoretic view



Applying closure under reasonable reductions:



Barrington

Lange

A little more than REG

Visibly pushdown automata:

ϵ-moves-free PDA whose stack access is based on a Σ-partition.

$$
\begin{aligned}
a \in \Sigma_+ &: \quad p \longrightarrow_a qA \\
a \in \Sigma_- &: \quad pA \longrightarrow_a q \\
a \in \Sigma_0 &: \quad p \longrightarrow_a q
\end{aligned}
$$

A little more than REG

Visibly pushdown automata:

ϵ-moves-free PDA whose stack access is based on a Σ-partition.

$$
\begin{aligned}
a \in \Sigma_+ &: \quad p \longrightarrow_a qA \\
a \in \Sigma_- &: \quad pA \longrightarrow_a q \\
a \in \Sigma_0 &: \quad p \longrightarrow_a q
\end{aligned}
$$

Balanced parentheses are VPLs.

$\Sigma_+ = \{ (\}$

$\Sigma_- = \{) \}$

What is known about VPL's and VPA's?

- First studied under the name Input-driven languages. *Mehlhorn*

What is known about VPL's and VPA's?

- First studied under the name Input-driven languages. *Mehlhorn*
- VPL is incomparable with DLIN.

  Balanced parentheses are VPLs, not in DLIN.
  $\{wcw^R \mid w \in \{a,b\}^*\}$ is in DLIN, not in VPL.

What is known about VPL's and VPA's?

- First studied under the name Input-driven languages. *Mehlhorn*
- VPL is incomparable with DLIN.

  Balanced parentheses are VPLs, not in DLIN.
  $\{wcw^R \mid w \in \{a,b\}^*\}$ is in DLIN, not in VPL.
- VPA can be determinized. *Alur, Madhusudan*
  So REG $\subset$ VPL $\subset$ DCFL (proper containments).

What is known about VPL's and VPA's?

- First studied under the name Input-driven languages. *Mehlhorn*
- VPL is incomparable with DLIN.

  Balanced parentheses are VPLs, not in DLIN.
  $\{wcw^R \mid w \in \{a,b\}^*\}$ is in DLIN, not in VPL.
- VPA can be determinized. *Alur, Madhusudan*
  So REG $\subset$ VPL $\subset$ DCFL (proper containments).
- VPL are in NC^1. *Dymond*
  Crucial ingredients:
  - Boolean Formula evaluation in NC^1. *Buss*
  - stack-height after reading i bits computable in $O(\log n)$ depth.

Counting class?

#NFA= #BWBP. What about #VPA?

Counting class?

#NFA= #BWBP. What about #VPA?

#VPA= #BWBP

Beyond VPA

- VPA: Σ-partition dictates stack movement.

Beyond VPA

- VPA: Σ-partition dictates stack movement.
- $Q \times \Sigma$-partitioned PDA?

Beyond VPA

- VPA: Σ-partition dictates stack movement.
- $Q \times \Sigma$-partitioned PDA?

  Doesn't make sense, because Q-partitioned PDA can simulate all PDA.

Beyond VPA

- VPA: Σ-partition dictates stack movement.
- $Q \times \Sigma$-partitioned PDA?

  Doesn't make sense, because Q-partitioned PDA can simulate all PDA.
- $Q \times \Sigma$-partitioned DPDA?

Beyond VPA

- VPA: Σ-partition dictates stack movement.
- $Q \times \Sigma$-partitioned PDA?

  Doesn't make sense, because Q-partitioned PDA can simulate all PDA.
- $Q \times \Sigma$-partitioned DPDA?

  More powerful than VPA.
  Equal(a, b), $\{wcw^R \mid w \in \{a, b\}^*\}$, can be accepted.

Beyond VPA

- VPA: Σ-partition dictates stack movement.
- $Q \times \Sigma$-partitioned PDA?

  Doesn't make sense, because Q-partitioned PDA can simulate all PDA.
- $Q \times \Sigma$-partitioned DPDA?

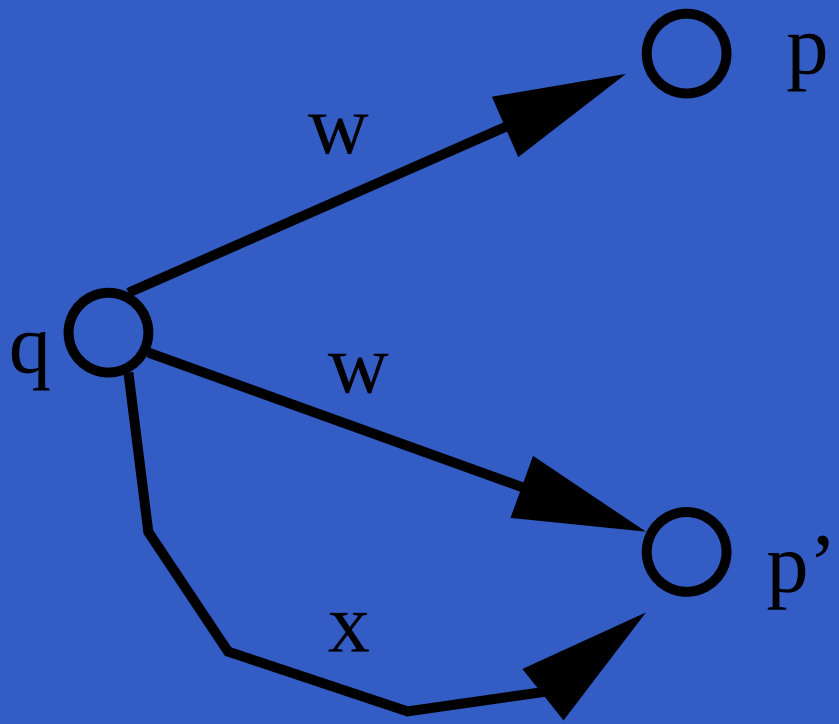
  More powerful than VPA.
  Equal(a, b), $\{wcw^R \mid w \in \{a, b\}^*\}$, can be accepted.
- What does the partitioning give us?
  Or, what are we looking for from the partitioning?
  The ability to quickly compute stack-height: Exactly what is needed for deciding membership in NC^1.

Synchronised PDA

G_P: infinite configuration graph of PDA

$g : V(G_P) \longrightarrow \mathbb{Z}$

T: deterministic transducer



g(p) = g(p') =

T(w) = T(x)

Framework of SPDA *Caucal*

P is synchronised by T if in G_P,

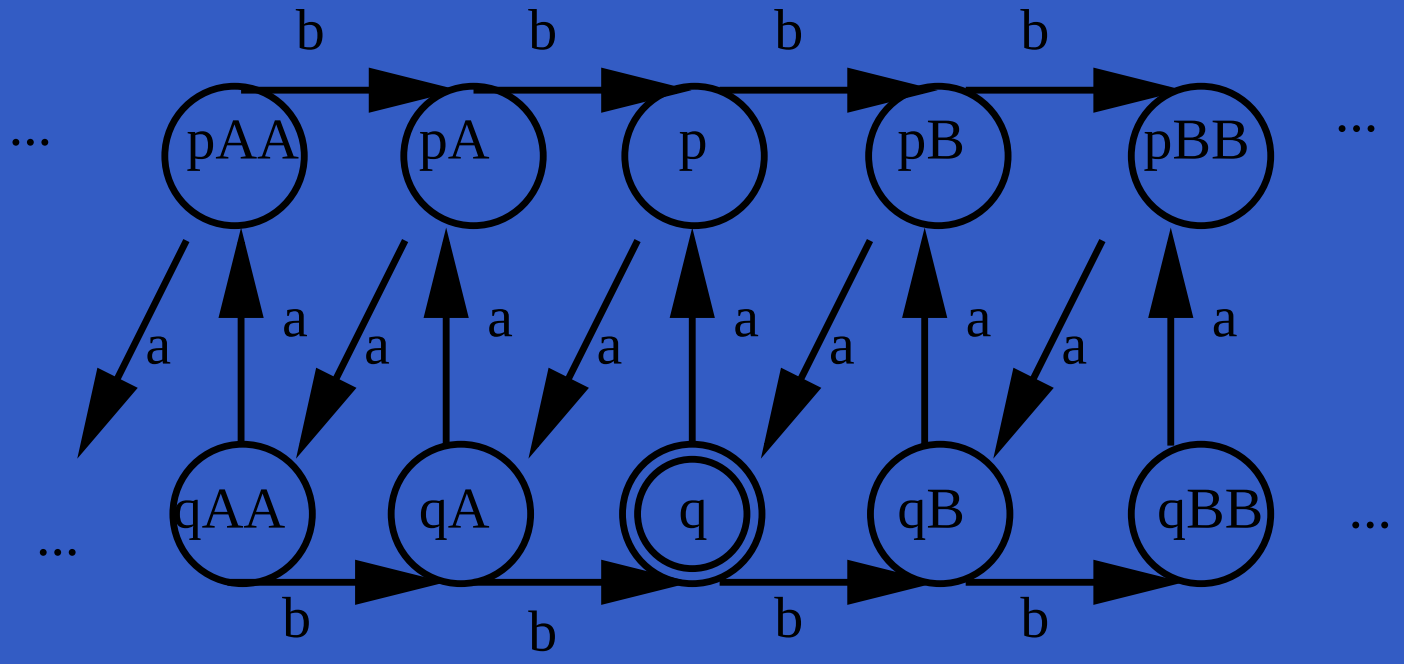
- $\exists g : V(G_P) \longrightarrow \mathbb{Z}$
- For each node u, all strings ρ labeling a path (from Init) to u have the same $T(\rho)$.
- For each string σ, all paths in G_P from Init labeled σ end in nodes u with $g(u) = T(w)$.
- G_P can be generated in a regular fashion respecting growing values of $|g(.)|$.

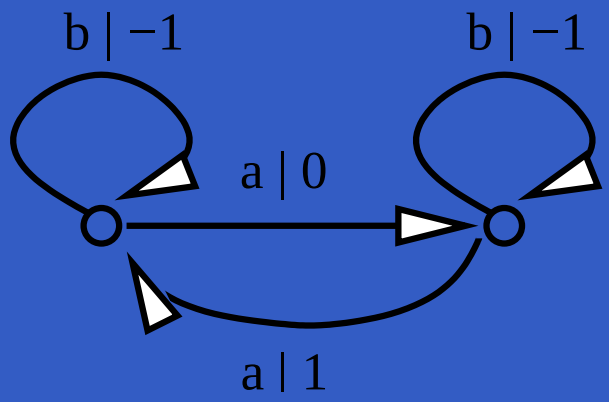
VPA: special case where T has one state, and outputs in $-1, 0, 1$.

For each fixed T, PDA synchronised by T form a Boolean algebra inside DCFL.

An example SPDA

PDA accepting $\{w \mid |w|_a = 2|w|_b\}$





Stack-synchronised PDA

- SPDA (P,T) with mapping $g : V(G_P) \longrightarrow \mathbb{N}$ being the stack height of a configuration.

Stack-synchronised PDA

- SPDA (P,T) with mapping $g : V(G_P) \longrightarrow \mathbb{N}$ being the stack height of a configuration.
- Despite non-determinism, stack-height profile on all runs of P for the same string looks the same.

Stack-synchronised PDA

- SPDA (P,T) with mapping $g : V(G_P) \longrightarrow \mathbb{N}$ being the stack height of a configuration.
- Despite non-determinism, stack-height profile on all runs of P for the same string looks the same.
- Stack-height on any prefix can be computed by running T and adding its outputs; i.e. within NC^1.

Stack-synchronised PDA

- SPDA (P,T) with mapping $g : V(G_P) \longrightarrow \mathbb{N}$ being the stack height of a configuration.
- Despite non-determinism, stack-height profile on all runs of P for the same string looks the same.
- Stack-height on any prefix can be computed by running T and adding its outputs; i.e. within NC^1.
- Hence, reduction to VPL ... languages accepted by stack-synchronised PDA are in NC^1.

Stack-synchronised PDA

- SPDA (P,T) with mapping $g : V(G_P) \longrightarrow \mathbb{N}$ being the stack height of a configuration.
- Despite non-determinism, stack-height profile on all runs of P for the same string looks the same.
- Stack-height on any prefix can be computed by running T and adding its outputs; i.e. within NC^1.
- Hence, reduction to VPL ... languages accepted by stack-synchronised PDA are in NC^1.
- Added bonus: reduction is parsimonious; so the arithmetization is #BWBP.

What is in Stack-synchronised PDA?

- $Q \times \Sigma$-partitioned *weak* DPDA are stack-synchronised PDA.

What is in Stack-synchronised PDA?

- $Q \times \Sigma$-partitioned *weak* DPDA are stack-synchronised PDA.
- Determinism/partitioning is not crucial. If there is a certain equivalence relation on the states of any weak PDA, it still is a stack-synchronised PDA.

What is in Stack-synchronised PDA?

- $Q \times \Sigma$-partitioned *weak* DPDA are stack-synchronised PDA.
- Determinism/partitioning is not crucial. If there is a certain equivalence relation on the states of any weak PDA, it still is a stack-synchronised PDA.
- The equivalence relation: States p, q of a PDA P are $Q\Sigma$-equivalent, denoted $p \sim q$, if whenever $pw \xrightarrow{a} p'w'$ and $qv \xrightarrow{a} q'v'$,

  it holds that $p' \sim q'$, $|w'| - |w| = |v'| - |v|$.

What else can we hope to capture?

- All weak SPDA?
- All SPDA?
- Exploit the power of NC^1 better; the transducer output can be computed in TC^0.
- For Dymond's algorithm, it suffices to consistently compute, in NC^1, the stack-height of any accepting run. Stack-synchronisation is not necessary. For what PDA is this possible?

Thank you!