

# An Algebraic Perspective on Boolean Function Learning

---

Ricard Gavaldà<sup>1</sup> Denis Thérien<sup>2</sup>

<sup>1</sup>LARCA research group  
Univ. Politècnica de Catalunya  
Barcelona, Spain

<sup>2</sup>School of Computer Science  
McGill University  
Montréal, Canada

---

ALT'09, Porto, october 5th 2009

# Introduction

We can learn boolean functions represented in many ways:

Conjunctions,  $k$ -CNF,  $k$ -DNF, monotone DNF, Deterministic Finite Automata,  $k$ -term DNF,  $k$ -decision lists, read-once formulas, bounded rank decision trees, constant-degree polynomials, sparse polynomials, threshold gates, decision trees, CDNF formulas, multisymmetric concepts, conjunctions of Horn clauses,  $O(\log n)$ -term DNF, nested subspaces, counter languages, OBDD, Multiplicity (Weighted) Automata, . . .

# Introduction

## Programs over monoids

- ... *yet another* representation of boolean functions!!

yes, but

- gives context: detailed, deep taxonomies of monoids
- highlights a few unnoticed learnable classes
- suggests limits of current techniques

# Summary

- Membership queries: algorithm for  $MOD_p \circ MOD_m$  circuits
- Equivalence queries: decision lists over constant-degree polynomials over  $\mathbb{F}_p$
- Membership + Equivalence:  
Maximal class of functions learnable as Multiplicity Automata
- Unifies many known results
- Does not capture: monotonicity, threshold circuits, read- $k$  conditions, sensitivity to variable ordering

Background: Algebra and circuits

# Semigroups

- A *semigroup* is set with a binary, associative operation
- A *monoid* is a semigroup with an identity
- A *group* is a monoid where each element has an inverse
  
- A monoid  $A$  *divides* a monoid  $B$  if  $A$  is a homomorphic image of a subsemigroup of  $B$
- An *aperiodic* (aka group-free) monoid is one that is divided by no nontrivial group

# Monoid products

- The direct product of  $A$  and  $B$ ,  $A \times B$  is defined by

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1 \cdot a_2, b_1 \cdot b_2)$$

- A semidirect product of  $A$  and  $B$  is defined by choosing a function  $f : A \times B \rightarrow A$  and

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1 \cdot f(a_2, b_1), b_1 \cdot b_2)$$

- The *wreath product* of  $A$  and  $B$ , denoted  $A \star B$ , generalizes semidirect product by accounting for all choices of  $f$

# Decomposition theorem

## Theorem [Krohn-Rhodes 62]

1. Every finite semigroup  $M$  divides a wreath product of finite simple groups and copies of the flip-flop monoid\*
2. Only finite simple groups are required if  $M$  is a group
3. Only flip-flop monoids are required if  $M$  is aperiodic

\* A particular 3-element aperiodic monoid

# Boolean functions

Functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$

- AND, OR, NOT, threshold gates
- Generalized  $MOD_m$  gates

$$MOD_m^A(x_1, \dots, x_n) = 1 \quad \text{iff} \quad \left(\sum_{i=1}^n x_i\right) \in A$$

- Decision lists, decision trees
- Deterministic Finite Automata
- Weighted Automata or Multiplicity Automata over rings  
 $M(x_1, \dots, x_n) =$  sum over all paths consistent with  $x_1 \dots x_n$   
of product of labels in path

## Programs over monoids

- An instruction over a monoid  $M$  is a triple  $(i, u, v)$   
Interpreted as “read  $x_i$  and emit  $u$  if  $x_i = 0$ ,  $v$  if  $x_i = 1$ ”
- A *program* over  $M$  is a sequence of instructions  
 $L = (l_1, \dots, l_s)$  plus an accepting set  $A \subseteq M$

$$(L, A)(x) = \begin{cases} 1 & \text{if } \prod_{i=1}^s l_i(x) \in A, \\ 0 & \text{otherwise.} \end{cases}$$

## Programs over monoids (2)

- Each program  $P$  over  $M$  computes a boolean function  $B(P)$
- $B(M)$  is the set of boolean functions computed by programs over  $M$
- For a class of monoids  $\mathcal{M}$

$$B(\mathcal{M}) = \bigcup_{M \in \mathcal{M}} B(M)$$

## From monoids to boolean functions

Division: If  $M_1$  divides  $M_2$  then  $B(M_1) \subseteq B(M_2)$

Direct product:

$$\begin{aligned} B(M_1 \times M_2) &\equiv \text{boolean combinations of } B(M_1) \text{ and } B(M_2) \\ &\equiv NC^0 \circ (B(M_1) \cup B(M_2)) \end{aligned}$$

Wreath product: For  $G$  a group,

$$B(M \star G) = B(M) \circ B(G)$$

## Examples

Classical examples [Barrington 87, Barrington-Thérien 89]:

<b>Monoidland</b>	<b>Circuitland</b>
all monoids	$NC^1$
any nonsolvable group	$NC^1$
Abelian groups	boolean combinations of MOD gates
solvable groups	poly-size, constant-depth circuits made of MOD gates
aperiodic monoids	poly-size constant-depth circuits made of AND, OR, NOT gates

For learning we should remain well below  $NC^1$

## Dramatis personae, groups

Group	Description
Abelian groups	direct products cyclic groups
$G_p$ or $p$ -groups	groups of cardinality $p^k$
Nilpotent groups	direct products of $p$ -groups
Solvable groups	wreath product of cyclic groups

## Dramatis personae, groups

<b>Groupland</b>	<b>Circuitland</b>
Abelian groups	$MOD_m$ , degree 1 polynomials over $Z_m$
$G_p$ or $p$ -groups	$MOD_{p^k} \circ NC^0$ , $MOD_p \circ NC^0$ , constant degree polynomials over $\mathbb{F}_p$
Nilpotent groups	$MOD_m \circ NC^0$ , constant degree polynomials over $Z_m$
Solvable groups	constant-depth, poly-size modular circuits

## Dramatis personae, aperiodic

**DA** monoids:  $(stu)^k t (stu)^k = (stu)^k$  for some  $k$

In circuitland [GT03]:

$$B(\mathbf{DA}) = \bigcup_k \text{rank-}k \text{ decision trees}$$
$$B(\mathbf{DA}) \circ NC^0 = \bigcup_k k\text{-decision lists}$$

Borderline of expressivity in several contexts (descriptive complexity, communication complexity)

(Almost) nothing between  $B(\mathbf{DA})$  and DNF, in monoidland

## Membership queries

## Negative results

### Fact [GT06]

Learning programs over  $M$  requires  $2^n$  Membership queries if

- $M$  is *not* a group
- or  $M$  is a *nonsolvable* group

Reason: Can compute singletons in polynomial size

# What about solvable groups?

Two subclasses of solvable groups *provably* weaker than  $NC^1$ :

- Nilpotent groups
  - Equivalent to polynomials of constant degree over some  $Z_m$
  - Includes Abelian groups and  $G_p$
- $G_p \star Abelian$ 
  - Equivalent to depth-2,  $MOD_p$ -of- $MOD_m$  circuits

## Group lower bounds

If  $G$  nilpotent, any two programs of length  $s$  over  $G$  differ on some assignment of weight  $c_G$  [PT88]

If  $G \in G_p \star \textit{Abelian}$ , any two programs of length  $s$  over  $G$  differ on some assignment of weight  $c_G \log s$  [BST89]

Learning strategy:

- 1 Ask Membership queries with all assignments of weight  $c_G$  (or weight  $c_G \log s$ )
- 2 Build *unique* program consistent with the answers

Part 2 is a purely computational problem

# Abelian groups

## Theorem

If  $G$  is Abelian, then  $B(G)$  is learnable from Membership queries in  $n^{O(1)}$  time

Equivalent to  $MOD_m$  gates and degree-1 polynomials over  $Z_m$

Open: extend to degree- $O(1)$  polynomials (= nilpotent groups)

## $G_p \star \text{Abelian}$

### Theorem

If  $G \in G_p \star \text{Abelian}$ , then  $B(G)$  is learnable from Membership queries in  $n^{O(\log s)}$  time

Equivalent to  $MOD_p$ -of- $MOD_m$  circuits

Known to be learnable in time  $(n + s)^{O(1)}$  from Membership *and* Equivalence queries [BBTV97]

## Equivalence queries

$$DL \circ MOD_p \circ NC^0$$

### Theorem [from known results]

Decision lists having constant-degree polynomials over  $\mathbb{F}_p$  at the nodes are learnable from  $n^{O(1)}$  Equivalence queries

Combine:

- Tricks to make  $MOD_p$  gates 0/1-valued [Fermat,BT94]
- Subspace learning algorithm [HSW87]
- Decision list / nested difference algorithm [R87,HSW87]
- Composition theorem

# $DL \circ MOD_p \circ NC^0$

$DL \circ MOD_p \circ NC^0$  subsumes:

- $DL \circ MOD_p$ : nested differences of linear subspaces of  $\mathbb{F}_p$
- $DL \circ NC^0$ :  $k$ -DL, so rank- $k$  DT's,  $k$ -CNF and  $k$ -DNF
- $MOD_p \circ NC^0$ : constant-degree polynomials over  $\mathbb{F}_p$
- strict width-2 branching programs [BBTV97]

*Note:* All these classes are nonuniversal

# Algebraic equivalent

## Theorem

1.  $DL \circ MOD_p \circ NC^0 = B(\mathbf{DA} \star G_p)$
2.  $\bigcup_m DL \circ MOD_m \circ NC^0 = B(\mathbf{DA} \star Nilpotent)$

Hence  $B(\mathbf{DA} \star G_p)$  learnable from  $n^{O(1)}$  Equivalence queries  
With Equivalence queries,  $B(\mathbf{DA} \star Abelian)$  learnable iff  
 $B(\mathbf{DA} \star Nilpotent)$  learnable

# What's the ceiling?

- If  $M \in \mathbf{DA} \star G_p$  then  $M$  is *not* universal
- If  $M \notin \mathbf{DA} \star \text{Nilpotent}$  then  $M$  is universal <sup>1</sup>
- For  $M$  in between, we don't know; basic first question

---

<sup>1</sup>subtle lie here; see proceedings

## Membership and Equivalence queries

# Multiplicity Automata

## Theorem [BBBKV00]

Functions  $\Sigma^* \rightarrow \mathbb{F}_p$  computed by Multiplicity Automata over  $\mathbb{F}_p$  are polynomial-time learnable from Membership and Equivalence queries.

Subsumes, besides DFA:

- polynomials over  $\mathbb{F}_p$
- unambiguous DNF (hence decision trees and  $k$ -term DNF)
- $MOD_p$ -of- $MOD_m$  circuits

# Algebraic characterization

## $LG_p \textcircled{m} \mathbf{Com}$ [Weil 87]

The value of  $m_1 \dots m_s$  can be determined by counting mod  $p$  the number of factorizations of the form  $a_0 L a_1 L a_2 \dots a_{k-1} L a_k$ , for  $L$  a commutative language (bool comb of)

## Theorem

1.  $B(LG_p \textcircled{m} \mathbf{Com})$  is polynomially simulated by MA over  $\mathbb{F}_p$
2. unambiguous DNF, polynomials, and  $MOD_p$ -of- $MOD_q$  circuits are polynomially simulated in  $B(LG_p \textcircled{m} \mathbf{Com})$

# Order sensitivity

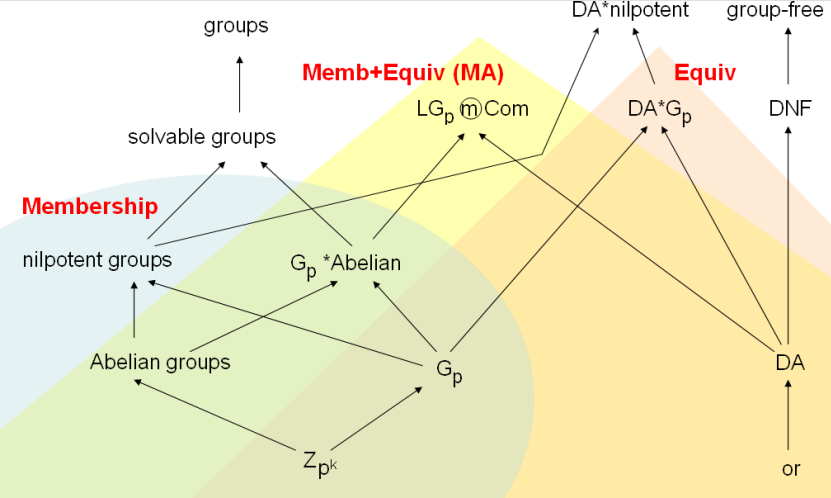
## Conjecture

$LG_p \textcircled{m} \mathbf{Com}$  is the largest class of monoids that is polynomially simulated by MA

Intuition: If  $M \notin LG_p \textcircled{m} \mathbf{Com}$  there is a function  $f \in B(M)$  such that  $f$  has MA of size  $poly(n)$  but the smallest MA for some  $f(x_{\pi(1)}, \dots, x_{\pi(n)})$  has size  $2^{\Omega(n)}$

There is an explicit characterization [TT07] of monoids *not* in  $LG_p \textcircled{m} \mathbf{Com}$

# In summary



# Conclusions

- Many learning results can be unified into 3 algorithms for learning large classes of monoids
- Extending to larger classes seems to require either proving new lower bounds or learning DNF
- Open problem: Efficiently learn one  $MOD_m \circ NC^0$  gate