
THE SPECTRUM PROBLEM FROM A PROGRAMMING PERSPECTIVE

Neil D. Jones (DIKU, University of Copenhagen)

LCC workshop 2006 at Seattle

Workshop: **50 years of the Spectrum Problem**, held in Oxford summer 2005.

This is essentially my talk from that workshop.

Currently being written: a joint paper on the topic, by Arnaud Durand, Neil Jones, Johann Makowsky, Malika More.

THE SPECTRUM PROBLEM

- ▶ Posed by Scholz in 1952; first steps around 1956; grew over the years, and led to the LCC field
(thanks to joint paper authors for newer context!)
- ▶ Resolution in early 1970s of a 20-year open problem
by a characterisation that was a product of its time
- ▶ Several independent discoveries (Jones and Selman, Fagin, Bennett?, Ritchie?, Christen,...)
- ▶ Roots in
formal language theory (fairly new in 1960s) and
computational complexity (just then emerging)
- ▶ Neil Immerman: “the beginning of finite model theory and descriptive complexity”

WHO SOLVED IT?

Several independent solutions...

- ▶ **Jones & Selman in 1970 or 1971** (appeared in 1972 STOC)
- ▶ Fagin's thesis (Berkeley 1973): discovered the solution independently (main emphasis on **“generalized spectra”**)
- ▶ Fagin: **Bennett proved it first, but didn't publish**
(a comment later retracted)
- ▶ Rumor that **Dennis Ritchie solved it too**
- ▶ **Germany**: Rödding and Schwichtenberg found early (1972) upper bounds on complexity of spectra;
- ▶ Christen found an exact characterisation a bit later, independently of work in the US

STRUCTURE OF THE TALK

1. 1970s: The **spectrum problem**

- (a) **Background:** Scholz, Mostowski, Asser, Bennett, . . .
- (b) Similar open problems in **formal language theory**
- (c) **The characterisation:** $\text{Spec} = \text{NE} = \text{NTIME}(2^{O(|n|)})$
- (d) Ideas of the Jones & Selman reasoning

2. My research shifted direction the 1990s, to study the **expressive power of read-only programs**

- (a) A **folklore theorem** (stemming from Cook's 2DPDA)
- (b) **First-order** read-only programs
- (c) **Higher-order** read-only programs
- (d) Primitive recursion versus tail- and general recursion

Plus ça change, plus c'est la meme chose

PART 1: THE SPECTRUM PROBLEM

Let ϕ be a closed first-order formula in

- ▶ predicate names P_1, \dots, P_ℓ (may include =)
- ▶ with arities k_1, \dots, k_ℓ

Definition An **interpretation** (or “structure”) is a tuple:

$$I = \langle A; \mathcal{P}_1, \dots, \mathcal{P}_\ell \rangle$$

with $\mathcal{P}_i \subseteq A^{k_i}$ for $i = 1, \dots, \ell$ (= must be identity)

Definition I is a **model** of ϕ , written $I \models \phi$ if ... (as usual)

EVALUATING $I \models \phi$ A LA TARSKI

Let $I = \langle \mathbb{N}_n; \mathcal{P}_1, \dots \rangle$ be an interpretation, and let environment ρ bind free variables of formula ϕ . **Define $I \models \phi$, $I \models_\rho \phi$ by:**

$I \models \phi$ iff $I \models_{\square} \phi$ for closed ϕ

$I \models_\rho P(x_1, \dots, x_k)$ iff $(\rho(x_1), \dots, \rho(x_k)) \in \mathcal{P}$

$I \models_\rho \neg\phi$ iff $I \models_\rho \phi$ is false ϕ :

$I \models_\rho \phi_1 \wedge \phi_2$ iff $I \models_\rho \phi_1$ and $I \models_\rho \phi_2$

$I \models_\rho \forall x \phi$ iff $I \models_{\rho[x \mapsto i]} \phi$ (for every $i \in \mathbb{N}_n$)

MORE BASICS

Definition The **spectrum** $\text{spec}(\phi)$ is
the set of cardinalities of finite models of ϕ :
$$\text{spec}(\phi) = \{n \mid I \models \phi \text{ for some } I = \langle A; \dots \rangle \text{ with } n = \#A\}$$

Without loss of generality:

- ▶ An n -element interpretation is over $\mathbb{N}_n = \{0, 1, \dots, n - 1\}$.
- ▶ Formulas have no function symbols.
- ▶ Logical operators are only \wedge, \neg, \forall .
- ▶ Formulas may contain $0, x + 1, x < y, \max$ (interpreted as $n - 1$).

SCHOLZ'S PROBLEM (1952): CHARACTERISE SPECTRA

$\text{Spec} = \{ \text{spec}(\phi) \mid \phi \text{ is a first-order formula} \}$

- ▶ One spectrum is **a set of natural numbers**
- ▶ Thus Spec is a **set of sets** of natural numbers.
- ▶ Given ϕ , the set $\text{spec}(\phi)$ is clearly decidable:
- ▶ **To decide $n \in \text{spec}(\phi)$: there are only finitely many n -element interpretations $\langle A; \mathcal{P}_1, \dots, \mathcal{P}_\ell \rangle$ over ϕ .**

Given n , **enumerate all interpretations; evaluate ϕ on each.**

- ▶ Scholz **possibly expected** something like:
 - **Spec is the smallest set of sets containing ... and**
 - **closed under operations op_0, op_1, \dots, op_k**

i.e., analogous to definition of primitive recursive functions.

PROPERTIES OF SPECTRA KNOWN AROUND 1960

- ▶ If S_1, S_2 are spectra, then both $S_1 \cup S_2$ and $S_1 \cap S_2$ are spectra. Easy to see.
- ▶ Much less clear: if S is a spectrum, then **is $\mathbb{N} \setminus S$ a spectrum?**
- ▶ Let ε_k^* = the sets whose characteristic functions lie in Grzegorzczuk's class ε_k . Known (Asser, Mostowski 1950s):

$$\varepsilon_2^* \subseteq \text{Spec} \subset \varepsilon_3^*$$

- ▶ **Not known:** whether

$$\varepsilon_2^* = \text{Spec}$$

This **all looked very familiar**: My M.S. and Ph.D. work (University of Western Ontario 1967) was about the **context-sensitive languages**:

the same properties and same open questions.

ABOUT NUMBERS AND STRINGS

Identify: a set $N \subseteq \mathbb{N}$ with its set of **binary representations**

$$\mathit{bin}(N) = \{\mathit{binary_representation}(n) \mid n \in N\} \subseteq \{0, 1\}^*$$

Notational convention: we write

- ▶ n for the number of elements in an interpretation,
- ▶ $|n|$ for the number of bits in $\mathit{binary_representation}(n)$
- ▶ **We ignore leading zeroes.**

Simple fact: for any $n \in \mathbb{N}$:

$$|n| \approx \log(n)$$

FORMAL LANGUAGE THEORY

Language: a set of strings $L \subseteq \{0, 1\}^*$.

The Chomsky hierarchy (1960s):

Class of sets	Machine characterisation	Relation to class below	Closed under		
			\cup	\cap	\setminus
Rec. enumerable	Turing machine	\supset	Yes	Yes	No
Context-sensitive	$\text{NSPACE}(x)$	\supset	Yes	Yes	??
Context-free	nondeterm. PDA	\supset	Yes	No	No
Regular	Finite automata		Yes	Yes	Yes

▶ **Big gaps** between classes

▶ **Diagonalisation** proves Rec. enumerable \supset Context-sensitive

SPECTRA AND RELATED CLASSES

Class of sets	Complexity class	Relation to class below	Closed under		
			\cup	\cap	\setminus
Spectra	$\text{NTIME}(2^{O(x)})$ <small>(new 1970s)</small>	\supseteq (=?)	Yes	Yes	??
Aux. SPACE $ n $ PDA	$\text{DTIME}(2^{O(x)})$ <small>(Steve Cook 1970s)</small>	\supseteq (=?)	Yes	Yes	Yes
Context-sensitive	$\text{NSPACE}(x)$	\supseteq (=?)	Yes	Yes	Yes(!) <small>(I&Sz 1980s)</small>
ϵ_2^*	$\text{DSPACE}(x)$	\supseteq (=?)	Yes	Yes	Yes
Rudimentary	??		Yes	Yes	Yes

► **Small gaps** between classes

► **Diagonalisation doesn't work** to prove \supsetneq . **Open questions!**

HOW MANY BITS ARE THERE IN AN INTERPRETATION ?

Let ϕ be a first-order formula in

▶ predicate names P_1, \dots, P_ℓ (may include =)

▶ with arities k_1, \dots, k_ℓ

▶ and let $\mathbb{N}_n = \{0, 1, \dots, n - 1\}$

Definition An **interpretation** is a tuple:

$$I = \langle \mathbb{N}_n; \mathcal{P}_1, \dots, \mathcal{P}_\ell \rangle$$

with $\mathcal{P}_i \subseteq \mathbb{N}_n^{k_i}$ for $i = 1, \dots, \ell$ (= must be identity)

How many bits to write down $I = \langle \mathbb{N}_n; \mathcal{P}_1, \dots, \mathcal{P}_\ell \rangle$?

Easy to see:

$$n^{k_1} \times n^{k_2} \times \dots \times n^{k_\ell} = n^{k_1 + k_2 + \dots + k_\ell}$$

SPECTRUM TESTING BY TURING MACHINE

Testing $n \in \text{spec}(\phi)$, for a fixed formula ϕ . Read n .

1. Choose interpretation $I = \langle \mathbb{N}_n; \mathcal{P}_1, \dots, \mathcal{P}_\ell \rangle$
(use nondeterminism)
2. Evaluate ϕ over I , i.e., decide $I \models \phi$.
3. Accept n iff $I \models \phi$.

Storage usage: How many bits to write interpretation I ?

$$n^{k_1+k_2+\dots+k_\ell} \approx 2^{|n|(k_1+k_2+\dots+k_\ell)}$$

where $|n|$ is the number of bits to write down n .

Storage usage: **exponential in the input length.**

Time for **fixed** ϕ : **polynomial in n** , so **exponential in $|n|$.**

A TURING MACHINE COMPUTATION

Turing machine has form $Z = (Q, \Sigma, q_0, Accept, \delta)$

Form of a configuration:

$$(q, a_1 a_2 \dots a_p, j)$$

where

$$q \in Q \quad = \textit{control state}$$

$$a_1 a_2 \dots a_p \in \Sigma^* \quad = \textit{tape contents}$$

$$1 \leq j \leq p \quad = \textit{scanning position}$$

A Turing machine computation on input n has form:

$$comp = \#config_0\#config_1\# \dots \#config_m\#$$

- ▶ $config_0$ contains input n ;
- ▶ $config_m$ is accepting; and
- ▶ $config_i \vdash config_{i+1}$ for $0 \leq i < m$

SIZE OF A TURING MACHINE COMPUTATION

Let Turing machine Z (nondeterministic) run in time $2^{c|n|}$.

How many bits in an entire Turing machine computation

$$comp = \#config_0\#config_1\# \dots \#config_m\#$$

where $m \leq 2^{c|n|}$?

Clearly $|config_i| \leq 2^{c|n|}$ for each i .

Total length:

$$|comp| \leq 2^{c|n|} \times 2^{c|n|} = 2^{2c|n|}$$

Thus **exponential in the length of the input**, i.e.,
polynomial in the value of the input.

1. ORDER LOGIC SIMULATION OF A TURING MACHINE

Let Turing machine Z (nondeterministic) run in time $2^{c|n|}$.

Idea: use a predicate name P of sufficiently large arity k .

Think of the interpretation \mathcal{P} as a bit sequence

$$\mathcal{P}(0, \dots, 0), \dots, \mathcal{P}(n-1, \dots, n-1)$$

coding a Turing machine computation on input n .

Write formulas to

- ▶ Check the syntax of the alleged Turing machine computation
- ▶ Check the forms of the initial and final configurations
- ▶ Check that $config_i \vdash config_{i+1}$ for $0 \leq i < m$

Technical parts:

- ▶ Access to the j -th bit of the computation
- ▶ Comparing the corresponding bits of $config_i$ and $config_{i+1}$

HOW WE ACTUALLY DID IT: 1

- ▶ Turing machine Z (nondeterministic):

input a number n .

Suppose Turing machine Z runs in time $2^{c|n|} = n^c$.

- ▶ Show equivalent: that

spectrum automaton M_n accepts input value n .

HOW A “SPECTRUM AUTOMATON” WORKS

1. Is there an accepting computation of length $2^{c|n|} = n^c$.
2. The can be written in (roughly) n^{2c} bits.
3. Guess (nondeterministically) a $2c$ -dimensional hypercube of bits $H : n^{2c} \rightarrow \{0, 1\}$.
4. Apply a multihead k -dimensional finite automaton to H :
 - ▶ at any moment a head has **one scanning position**, given by a $2c$ -tuple (i_1, \dots, i_{2c}) with each $0 \leq i_j \leq n - 1$
 - ▶ Finite-state control (**one head is enough (?)**)
 - ▶ One-step head actions:
accept, or
test: is scan head at boundary in dimension $i?$, or
move to an adjacent position in the hypercube

HOW WE ACTUALLY DID IT: 2

- ▶ Simulate the “spectrum automaton” by a first-order formula:
 - One-step state transition
 - plus transitive closure

PART 2: READ-ONLY COMPUTATION (1990s)

Later, I shifted research areas, from logical topics to programming languages, semantics,...

Aim: to understand powers of programming language **control structures**, e.g., **while** versus **for** loops.

Approach: relate subrecursive programming languages to known complexity classes.

For example there was a

Folklore theorem:

$L \in \text{LOGSPACE}$ iff L is accepted by a

deterministic multihead read-only Turing machine.

LOGSPACE AND PTIME CHARACTERISED BY READ-ONLY PROGRAMMING LANGUAGES

Read-only functional programs:

- ▶ Input data in $\{0, 1\}^*$, i.e., a **Boolean list**
- ▶ Current computational state:
several suffixes x_1, x_2, \dots, x_k of original input $a_1 a_2 \dots a_n$.
Implicit stack implementing recursive calls.
- ▶ Operations `hd`, `tl` **but no cons** (**cons allocates memory!**)
- ▶ Control: `if-then-else`, function calls (**general recursion**)
- ▶ **Tail recursion**: **no computation after** a recursive function call:

$f(\dots) = \dots$ *if ... then* $f(+++)$ **OK!**
else $g(f(---))$ **NOT OK!**

LOGSPACE AND PTIME CHARACTERISED BY READ-ONLY PROGRAMMING LANGUAGES

Two results (Theoretical Computer Science 1999)

1. $L \in \text{LOGSPACE}$ iff L is decidable by a read-only **tail-recursive functional program**.
2. $L \in \text{PTIME}$ iff L is decidable by a read-only **general-recursive functional program**.

Roots of the proofs

- ▶ 1: based on the folklore theorem.
- ▶ 2: based on Cook's result on **auxiliary pushdown automata**.
Idea: replace Cook's explicit stack by the use of recursion.
 - Operation "move tape head right" implemented by **t1**.
 - **Programming tricks** to express "move tape head left".

THE EXPRESSIVE POWER OF HIGHER-ORDER TYPES

Results (Journal of Functional Programming, 2001)

- ▶ Characterised decision powers of **higher-order** read-only programs
- ▶ Two-dimensional classification: first, by **data order**
(first-order programs = data order 0)
- ▶ Second, by **control structure**:
 - general recursion,
 - tail recursion
 - primitive recursion(called `foldr` in functional programming)

SURPRISE

All unawares, I was again doing finite model theory!

(Thanks to Bruce Kapron for seeing this.)

- ▶ Boolean list input = interpretation of a unary predicate
- ▶ Read-only restriction amounts to computing a “global function” on a finite algebra
- ▶ Primitive recursive control: studied by Gurevich in 1983
- ▶ What’s new in the programming approach?
Control by general recursion and tail recursion.

A special case is `foldr`, i.e.,

primitive recursion on Boolean lists.

POWER OF HIGHER-ORDER PROGRAMS

Programs	1. order	2. order	3. order ...	LIMIT
Read-write	REC. ENUM	REC. ENUM	REC. ENUM	REC. ENUM ...
	Kleene			
Primitive recursive	PRIM.REC.	PRIM ¹ REC.	PRIM ² REC. ...	System T
	Gödel			Gödel
Read-only (RO)	PTIME	EXPTIME	EXP ² TIME ...	ELEMENTARY
	Jones	Jones	Jones	
Tail recursive RO	LOGSPACE	PSPACE	EXPSPACE ...	ELEMENTARY
	Jones	Jones	Jones	
Primitive rec. RO	LOGSPACE	PTIME	PSPACE ...	ELEMENTARY
	Gurevich	Gurevich	Goerdt, Seidl	

Odd fact: the bottom row grows “**half as fast**” as rows above.
Difference: the bottom row has **limited control structure:**
for loops instead of **while** loops (unlimited recursion).

You pay a price for disciplined programming control structures.

SOME CONSEQUENCES

Read-only programs	1. order	2. order	3. order ...	LIMIT
General recursive	P _{TIME}	EXPTIME	EXP ² TIME ...	ELEMENTARY
Tail recursive	LOGSPACE	PSPACE	EXPSPACE ...	ELEMENTARY
Primitive recursive	LOGSPACE	P _{TIME}	PSPACE ...	ELEMENTARY

▶ A long-standing open problem

Is general recursion **more powerful than** primitive recursion or tail recursion or first-order read-only programs?

Equivalent by table to:

Is P_{TIME} a proper superset of LOGSPACE?

▶ For 2. order programs: general recursion **IS** more powerful than primitive recursion since

$$\text{EXPTIME} \neq \text{P}_{\text{TIME}}$$

Maybe a problem with **deriving programs from proofs?**

BRINGING SOME LOGICAL THREADS TOGETHER

Spectrum problem (Scholz; Jones+Selman; Christen)

Given: FOL formula ϕ

To decide: $\{\#I \mid I \models \phi\}$

Complexity: NE in $|\#I|$

Generalised spectra (Fagin)

Given: formula $\exists T \phi$

To decide: $\{\mathcal{T} \mid [T \mapsto \mathcal{T}] \models \phi\}$

Complexity: NPTIME in $|\mathcal{T}|$

Model-checking (Clarke, Emerson, Vardi,...)

Given: temporal formula ϕ , e.g., in CTL

To decide: $\{\mathcal{M} \mid \mathcal{M} \models \phi\}$

Complexity: PTIME in $|\mathcal{M}|$

Read-only programs (Gurevich, Goerd, Seidl, Jones)

Given: read-only program p

To decide: $\{d \mid \llbracket p \rrbracket(d) = 1 \text{ and } d \in \{0, 1\}^*\}$

Complexity: LOGSPACE...ELEMENTARY (depends on order, control structure)