# Avraham “Abe” Bernstein: CV Abbreviated

Expert in Software Algorithm Design, Domain Specific Languages, & Anti-Reverse Engineering (aka Obfuscation)

Avraham Bernstein

2021-10-28

email: Avraham DOT Bernstein AT gmail DOT com   
tel/whatsapp: +972-54-6410955   
geolocation: Jerusalem 9727433 Israel   
  
cv-full-html-link: [Most recent version of full CV in HTML format](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html)  
cv-abbrev-docx-link: [Most recent version of abbreviated CV in DOCX format](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Abbrev.docx)  
  
Copyright © 2021, Avraham Bernstein, Israel. All rights reserved.   
License: CC-BY-4.0: [Creative Commons Attribution 4.0 International](https://spdx.org/licenses/CC-BY-4.0.html)



Secure[1](http://www.avrahambernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "fn1) watermarked photo of the author from 2017: Avraham Bernstein

My profession is a computer scientist and inventor. I have a long and successful track record inventing and implementing [algorithms](https://en.wikipedia.org/wiki/Algorithm) in a wide range of application domains such as automotive, pay TV, [VLSI](https://en.wikipedia.org/wiki/Very_Large_Scale_Integration) [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) design, shop floor production control, [bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics), accessibility, and telecommunications. I am hired to solve challenging and interesting problems that have greatly benefited my employers and clients. A common technique that I use is developing [DSLs](https://en.wikipedia.org/wiki/Domain-specific_language) (Domain-Specific Language) and their associated [compilers](https://en.wikipedia.org/wiki/Compiler). And I have unique expertise in the subspeciality of cybersecurity related to anti-[reverse engineering](https://en.wikipedia.org/wiki/Reverse_engineering), aka [obfuscation](https://www2.cs.arizona.edu/~collberg/Teaching/553/2011/Resources/obfuscation.pdf).

I invent software [algorithms](https://en.wikipedia.org/wiki/Algorithm). Typically for all my inventions I must implement a working POC (Proof Of Concept), and often I also implement a MVP (Minimum Viable Product). I am a “hands-on” technology expert who comfortably “swims” from the highest conceptual level down to the lowest level of “bits and bytes”.

For many projects I invent a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) (Domain-Specific Language), that [compiles](https://en.wikipedia.org/wiki/Compiler) into working software. Where feasible I design the [DSLs](https://en.wikipedia.org/wiki/Domain-specific_language) to be [declarative](https://stackoverflow.com/questions/1619834/what-is-the-difference-between-declarative-and-procedural-programming-paradigms) - as opposed to procedural. [DSLs](https://en.wikipedia.org/wiki/Domain-specific_language) greatly simplify the problem space especially for domain experts who are not software professionals. And [DSLs](https://en.wikipedia.org/wiki/Domain-specific_language) greatly help to rigorously specify testable software architectures and specifications.

I am an [expert generalist](https://simplicable.com/new/expert-generalist) and an [autodidact polymath](https://iamautodidact.com/how-to-become-an-autodidact-polymath-the-complete-guide/) [2](http://www.avrahambernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "fn2). My “secret weapon” is a subscription to the [O’Reilly Safari](https://www.oreilly.com/library/view/learning-java-4th/9781449372477/pr02s07.html) technical library. I am able to synthesize my knowledge. I am often able to apply the skills and knowledge that I acquired in one domain, and successfully apply them to another. To get to this level of expertise so quickly, I have often relied upon domain experts who mentored me. In return, I have mentored many others.

I attended university in Toronto Canada where I received a Master’s degree in economics and applied mathematics. My thesis was the economically efficient design of a [hydroelectric dam](https://en.wikipedia.org/wiki/Hydroelectricity) using computer simulation of water flow. At the age of 14, my first major programming project was to program a perfect game of 3D X’s and O’s (aka [Qubic](https://img0.etsystatic.com/053/1/11127679/il_570xN.771759904_fqi9.jpg)) played on a stack of four transparent 4x4 boards. I wrote the program in Fortran on a [mini-computer](https://en.wikipedia.org/wiki/IBM_1130) that had just 64 KB RAM including the space required for the O/S.

Like everyone else in the software development business, I started off as a junior programmer. Later I progressed to manage teams and small departments, and finally to the level of CTO for 3 different small firms - where I was a partner in one of them. But first and foremost I am a master software engineer. Like Picasso loved to paint his whole life, I still love designing and inventing software algorithms. As you can see from my recent experience described immediately below, even though I am in my 60’s, I am still at the top of my game. Note my recent patents. I decided that I prefer working on very challenging and interesting problems alone, or at most managing a very small team, where I shine as a mentor. I do not want to take a standard (Hebrew: Stam) senior software developer position, working on sundry bug fixes and minor feature additions. Ideally, for a small firm I should be working on special projects (possibly as a contractor), and for a large firm I should report to the CTO.

The following are many of the application domains in which I have worked:

1. automotive: 2018-20: [Argus](https://argus-sec.com/)

(a) I worked on greatly increasing the efficiency of vehicle software updates. My IP (Intellectual Property) enabled my employer to set up a business unit in this field.   
(b) I invented and patented algorithms in the fields of [delta compression](https://en.wikipedia.org/wiki/Delta_encoding) and assembly “hot patching”.   
(c) I developed legal and safe workarounds to the fundamental [Misra C](https://en.wikipedia.org/wiki/MISRA_C) restrictions on the use of dynamic memory allocations and variable length structures. These restrictions forced most software architects and programmers into abandoning many of the software advances made in the past 20 years.

(d) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "automotive)

1. Internet and satellite pay-TV: 2004-17: [Viaccess-Orca/Orange](https://www.viaccess-orca.com/), [NDS](https://en.wikipedia.org/wiki/Synamedia)

(a) cybersecurity research  
(b) anti-[reverse engineering](https://en.wikipedia.org/wiki/Reverse_engineering), aka [obfuscation](https://www2.cs.arizona.edu/~collberg/Teaching/553/2011/Resources/obfuscation.pdf)   
(c) “lightweight” cryptography, where I am continuing to do my own independent research until today for the purpose of possibly building my own [obfuscating compiler](https://github.com/JonathanSalwan/Tigress_protection)

(e) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "pay-tv)

1. miscellaneous recent compiler projects: 2020-21: [Qedit](https://qed-it.com/), stealth mode cybersecurity startup

(a) patching [x86-64](https://en.wikipedia.org/wiki/X86-64) assembly code  
(b) [WASM](https://webassembly.org/) (Web Assembly)

(c) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "misc-compiler)

1. [**bioinformatics**](https://en.wikipedia.org/wiki/Bioinformatics): 2009: [Syntezza](https://www.syntezza.com/)

(a) invented [PCR](https://en.wikipedia.org/wiki/Polymerase_chain_reactio) algorithms

(b) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "bioinformatics)

1. accessibility: 1988-present: Cubital (defunct), Virtouch (defunct), and continuing independent research until today

(a) invented algorithms and applications for the blind, [dyslexics](https://en.wikipedia.org/wiki/Dyslexia), and quadriplegics

(b) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "accessibility)

1. online privacy & personal cybersecurity: 2000-present: [Canary Mission](https://canarymission.org/), and continuing independent research until today

* Internet hygiene for the masses

1. transportation vehicle guidance: 2010: Telequest (defunct)

* invented algorithms for guiding vehicles through urban traffic (similar to what [Waze](https://www.waze.com/) does)

1. telecommunications: 1998-2004: Phasecom-Vyyo (defunct), Jolt (defunct), TMT (defunct), [NDS](https://en.wikipedia.org/wiki/Synamedia)

(a) hybrid protocols, operations research, algorithms, QA, [DSL](https://en.wikipedia.org/wiki/Domain-specific_language), communications laboratory architecture

(b) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "telecommunications)

1. **Mil-spec QA: 1994-95:** [Elop/Elbit](https://elbitsystems.com/about-us-quality-assurance-elbit-systems-electro-optics-elop/), Pitkha (defunct)

(a) invented a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) that was used to implement thousands of mil-spec QA tests for the Israeli manufacturered fire control subsystem of the US Army’s [BlackHawk](https://en.wikipedia.org/wiki/Sikorsky_UH-60_Black_Hawk) helicopter

(b) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "milspec-qa)

10.[**VLSI**](https://en.wikipedia.org/wiki/Very_Large_Scale_Integration) [**CPU**](https://en.wikipedia.org/wiki/Central_processing_unit) design: 1991-93 & 1998: [DSPG](https://www.dspg.com/), Pitkha (defunct), Fourfold (defunct)

(a) invented a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) that could extremely accurately simulate a [DSP](https://en.wikipedia.org/wiki/Digital_signal_processor) [CPU](https://en.wikipedia.org/wiki/Central_processing_unit), along with designing a software development toolchain including C compiler, linker, assembler, and debugger   
(b) wrote a C compiler for a 128 core stack based [CPU](https://en.wikipedia.org/wiki/Central_processing_unit)

(c) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "vlsi-cpu-design)

1. shop floor production control: 1989-91: [Iscar](https://www.iscar.com/newarticles.aspx/countryid/1/newarticleid/259), DEC (defunct)

(a) invented a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) that could configure and fully automate a factory that produced thousands of different kinds of cutting blades

(b) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "sfpc)

1. 3D printing: 1987-88: Cubital (defunct)

(a) senior software engineer

(b) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "3d-printing)

1. robotics: 1986 & 1996-97: Orisol (defunct), [Optimet/Ophir](https://www.optimet.com/about_ophir_optronics.php)

(a) invented a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) based on [AutoCAD](https://en.wikipedia.org/wiki/AutoCAD) for high speed leather sewing workstation with live video feedback   
(b) invented a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) in order to automate use of an [interferometer](https://en.wikipedia.org/wiki/Interferometry)-like workstation (based upon [conoscopic probe](https://www.deepdyve.com/lp/emerald-publishing/conoscopic-probes-are-set-to-transform-industrial-metrology-dL3FC3ukUZ) technology) enabling it to be used on a factory floor   
(c) invented a [DSL](https://en.wikipedia.org/wiki/Domain-specific_language) used to implement a 3D graphics toolkit based upon [OpenGL](https://en.wikipedia.org/wiki/OpenGL) in order to visualize and manipulate the huge clouds of potentially millions of data points resulting from the probe measurements

(d) [More details ...](http://www.AvrahamBernstein.com/cv/AvrahamBernstein-CV-Full.html" \l "robotics)

1. avionics: 1984-85: DSI (defunct?), [Elta/IAI](https://www.iai.co.il/about/groups/elta-systems)

* junior programmer for the radar on the [Lavi fighter plane](https://en.wikipedia.org/wiki/IAI_Lavi)

1. data collection terminals: 1983 & 1986: DSI (defunct?), [Elta/IAI](https://www.iai.co.il/about/groups/elta-systems), Elde (defunct)

* my first embedded software project in which I implemented a micro kernel for the [Intel 8080](https://en.wikipedia.org/wiki/Intel_8080) under the guidance of a senior mentor

## Computer Languages & Packages

| Expertise | Language |
| --- | --- |
| 5/5 | C99, C11, [gcc](https://gcc.gnu.org/), clang, bash, awk, markdown, pandoc, Linux User Mode API & CLI |
| 4/5 | make, [sed](https://en.wikipedia.org/wiki/Sed), Python, [pyexpander](https://pyexpander.sourceforge.io/), [TCL](https://en.wikipedia.org/wiki/Tcl), [tinyc](https://en.wikipedia.org/wiki/Tiny_C_Compiler), Zim wiki, MkDocs static web site generator |
| 3/5 | C++, HTML5, Javascript, [XML](https://en.wikipedia.org/wiki/XML), JSON, Yaml, [ELF](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format), [Lief](https://github.com/lief-project/LIEF), [Zydis](https://github.com/zyantific/zydis), [Pydis](https://github.com/novogen/pydis), [SrcML](https://www.srcml.org/), [Beautiful Soup](https://beautiful-soup-4.readthedocs.io/en/latest/), [Jinja2](https://jinja2docs.readthedocs.io/en/stable/), Reveal.js, git, Jira, Excel, Visual Studio, [libvirt](https://en.wikipedia.org/wiki/Libvirt), [VirtualBox](https://en.wikipedia.org/wiki/VirtualBox), Windows User Mode API |
| 2/5 | CMake, [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics), Golang, [LLVM](https://llvm.org/), Antlr, bison, flex, Assembler, [Forth](https://en.wikipedia.org/wiki/Forth_(programming_language)), Prolog, Lisp, Fortran, C#, Java, J (i.e. neo-APL), Numpy, OpenGL, [SNMP](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol), TCP/IP, UDP, vim |

My expertise level ranking is based upon how frequently I use these languages, and also based upon my personal preferences. But since I write compilers, I am usually able to become proficient in any computer language that my job requires within 1-2 weeks.

### Language Research Interests

1. [Nim](https://nim-lang.org/) and [Python 3.10 structural pattern matching](https://www.python.org/dev/peps/pep-0635/) are interesting due to the way they elegantly handle ASTs ([Abstract Syntax Tree](https://en.wikipedia.org/wiki/Abstract_syntax_tree)) which are useful for writing compilers, refactoring, and obfuscation patterns.
2. A deeper understanding of [WASM](https://webassembly.org/) (Web Assembly) is interesting (1) in order to bring C binary performance into Web browser Javascript applications, and (2) as a virtual machine that can allow compiled C/Golang/Nim/Rust functions to be dynamically loaded into binary applications instead of [FORTH](https://en.wikipedia.org/wiki/Forth_(programming_language)) whose stack programming paradigm is uncomfortable/unfamiliar for many programmers.