Avraham “Abe” Bernstein | CV-Lite

Author: *Avraham "Abe" Bernstein*
Email: *Avraham DOT Bernstein PLUS cv AT gmail DOT com*
Tel/Whatsapp: *+972.54.641-0955*
City: *Jerusalem 9727433 ISRAEL*
Time Zone: *UTC +02:00/+03:00 (winter/summer)*
Shabbat Observant: *Not accessible electronically nor engaging in any business activities from Fri. evenings (Jm. time) beginning 1 hour before sunset until Sat. night 1 hour after sunset, nor on Jewish holidays*
CV: *Full HTML, Full PDF, Lite HTML, Lite PDF*
WWW: *https://www.avrahambernstein.com*
Linkedin: *https://www.linkedin.com/in/avrahambernstein/*
Last Update: *2025-07-28*

1. Summary

Senior software architect and polymath with over 40 years of innovation in algorithm design, compiler construction, and obfuscation tools across industries including cybersecurity, automotive, accessibility, and bioinformatics. Adept at designing domain-specific languages, anti-reverse engineering, and building MVPs and prototypes for CTO teams. Holds multiple patents and thrives on technically challenging projects.

NOTICE for pedantic HR types: This CV is slightly unusual. It is 7 pages long instead of 3, OMG! It represents my 40+ years experience as a polymath in a wide range of domains at an expert level, and includes many inventions, many of which are recent. Normally the inventions require a few lines of explanation in order to show their signficance. Therefore this CV is quite different than someone's who has been working in the same field using the same tools for 10-15 years without producing any inventions, where a 2 page CV would suffice. When am I able to present my CV directly to a potential employer with a strong technical background, I have never encountered even a single one who was not extremely impressed by my professional accomplishments, nor who gave me any negative feedback on account of its length. And I am 69, which means it will be more likely for me to be hired as a contractor or a consultant, instead of as a regular employee.

2. Core Skills & Tools

- **Languages:** C, Python, Bash, HTML, Markdown, XML, YAML
- **Technologies:** srcML (commercial license), Beautiful Soup, Jinja2, Pyexpander, WASM, Linux, ELF, GCC, Clang
- **Domains:** Compiler Design, Obfuscation = anti-reverse engineering design, Cybersecurity, Embedded Systems, Accessibility, Automotive Software, Bioinformatics
- **Other:** Domain Specific Languages (DSL), Code Refactoring, Reverse Engineering, Factory Automation

3. Recent Experience

2025: Independent: Founder & Principal Engineer Compiler & Obfuscation Tools Development

- Designing commercial-grade obfuscating compilers for C/C++ and Web Assembly (WASM).
- Implementing tools for static and dynamic code obfuscation, encrypted string handling, name mangling, and ELF DSO export masking.
- Leveraging srcML and Beautiful Soup for automated C source code refactoring

2022-25: Aurora Labs, Tel Aviv: Senior Software Architect in CTO office for Digital Automotive Industry

- **Invented a patent-pending** algorithm to reduce RAM usage and boost compression during FLASH updates. The key to the patent was the use of buffered I/O windows, similar to the *Linux* file system, which eliminated the need to allocate a pre-image in RAM the size of the FLASH sector especially when the size of the *DMA* write region is typically only 256 bytes. Thus we are able to transfer larger FLASH chunks which typically results in significantly better compression. For small FLASH devices, in the range of 64-128 KB, compression could often be improved by up to 25%. And the board's original FLASH driver could be utilized without any modification.
- Reduced CPU (**6x**) and RAM (**2x**) usage compared to the company's original core source code refactoring product for embedded automotive C code via the use of srcML and Beautiful Soup. The original version was a hand coded compiler in *Python*. Its purpose was to automatically reduce the time of an automobile software update (via naive *LZMA* compression) from a few hours to as low as 15 minutes. I had a great deal of co-operation from the company's chief scientist, Carmit Sahar, who designed the original version.

2021: Morphisec, Beer Sheva: Anti-Reverse Engineering Modifications to *Linux x64* Libc Kernel

- The challenges were (1) not to change the addresses of any of the *libc* functions, and (2) to automatically modify the kernels of the many different *Linux* distributions. Implementation was via the *Python* version of the ZyDis *x64* disassembler.

2021: Qedit, Tel Aviv: WASM Cybersecurity Consultant for Financial Industry

2017-20: *Argus Cyber Security* (now PlaxidityX), Tel Aviv: Senior Researcher for Digital Automotive Industry

- **Patented** a modification to the mini-bsdiff algorithm in order to minimize FLASH memory usage in automotive software updates for devices which ostensibly did not have sufficient FLASH to implement the algorithm. I was in contact with Colin Percival, the original inventor of the *bsdiff* algorithm.
- Architect of embedded software update driver using mini-bsdiff and xz compression.

4. Earlier Roles (Selected)

2013-17: Viaccess-Orca (subsidiary of Orange FR), Ra'anana: Cybersecurity Obfuscation Manager for Internet TV industry

2013 part-time: *NVT*, US (defunct): CTO of Agritech Startup for Cassava Production in Nigerian Jungle

2012: *Telequest*, Jerusalem: VP R&D, Automated Vehicle Navigation To Find Optimal Routes in City Traffic

2011: Syntezza Bioscience, Jerusalem: Bioinformatics consultant

- In just 3 months I **invented** an AI threshold algorithm for PCR Ct detection, and a noise-reduction technique for a MRSA diagnostic kit. When I first began work, their kit was a complete failure. Even though I had zero background in microbiology, I discovered that the PCR control test tubes were configured improperly in all of their 800 hospital samples, for which I was able to correct after-the-fact even though the samples were long ago destroyed. And I discovered that their preliminary chemistry was not able to filter the inhibitors in their nasal samples.
- Even after correcting for the systematic noise, at least 50+% of the data points were inhibited, so the graph of the assays were definitely not *sigmoidal* (i.e. flat "S" shaped). Their graph looked more like *Rorchach* ink blots. While the graph of the pure *MRSA* colonies were linear, also definitely not *sigmoidal*. The fundamental assumption of *PCR* mathematics is that the shape of the assay function must be *sigmoidal*, so it seemed that we faced a catastropic failure. But my threshold algorithm handled them just fine. I was mentored by the company's consultant, Tzachi Bar, who invented the most commonly used public domain algorithm used in most *PCR* devices. **My algorithm saved the company's business unit from bankruptcy, and my algorithm won a MRSA detector "shoot-out" against the big pharma labs at the world famous St. George's medical school in London.**

2005-10: *NDS* (now Synamedia), Jerusalem: Internet TV Industry:

1) Cybersecurity Research:

- I showed how Digital Rights Management (DRM) protection schemes that relied upon date or playback count restrictions could easily be broken by garden variety virtual machines.
- I showed how the QEMU virtual machine, that allowed for assembly instructions to be emulated, could be used to break the host machine's random number generator, which allowed multiple confederate hosts to generate the exact same card ID, which enabled massive leeching of content from the content servers.
- I penetrated a confederate network of card ID sharing, and showed how all the confederates could easily be knocked out.

2) Satellite Content Servers Automated Testing:

- I **invented** a Domain Specific Language (DSL) (implemented in TCL on *Windows NT*) for simulation of the satellite.
- I used the *DSL* to generate flaws in the satellite's control program that disabled the satellite.
- I supplied the *DSL* to all developers to run sanity tests *before* checking in their code.
- I studied the company's bug database, and found that half of their

development effort was spent fixing bugs. The company had over 1000 developers! So I created a simple development protocol that knocked out over half of the bugs on the developer's desktop *before* submitting code to the QA group. The problem was that most check-ins were from make files with tens of thousands of warnings. The silver bullet was insisting upon check ins of make files without any warnings.

2002-03: *Virtouch*, Jerusalem (defunct): VP R&D, Accessibility Technology

- **Invented** an SVG web browser based solution enabling the blind to interpret images, maps, and mathematical functions via the PC's sound card and a consumer grade graphics tablet.

1999-2004: *Vyyo*, Jerusalem (defunct): for RF wireless cable modem industry

- Manager offline software tools: network management system (NMS), embedded SNMP agent, highly efficient ARP table and embedded hash table algorithm, and much more efficient modem speed testing via a Fibonacci search algorithm instead of the industry standard binary search algorithm.
- Architect of super-efficient cable modem testing laboratory. Reduced H/W costs by a factor of **16x** via multiplexing with a programmable layer-2 switch.
- **Invented** a hybrid Ethernet/Dial-up network protocol for cable modems, via dynamic modifications to the edge router's ARP table where the *edge-router* was the interface to both the dial-up and cable modem Ethernet networks.

1998: *Fourfold*, Jerusalem (defunct): Software architect of a modified GCC massively parallel compiler tool chain for a Forth-like CPU with unlimited registers.

1990-97: *Pitkha*, Jerusalem (defunct): CEO & Domain Specific Language (DSL) Architect

1. **1996-97:** 2D conoscopic holographic laser metrology device for Optimet-Ophir (subsidiary of MKS-Newport, US), Jerusalem
    - **Invented** a *DSL* (implemented in the *Windows NT* implementation of TCL) used to scan objects mounted on a XY jig. Due to lighting conditions on the object, different regions required different scanning speeds.
    - In theory, a scan returned a cloud of tens of millions of XYZ points. The most difficult problem was the creation of an easy to use language that generated life-like images from these points. Note that this project was implemented before the mature VTK image visualization toolkit became available. I **invented** my own, but simpler, version of *VTK* based upon a *TCL* wrapper for OpenGL.
2. **1996:** mil-spec testing laboratory for *Elop* (subsidiary of Elbit), Rehovot
    - **Invented** a *DSL*, a *BASIC*-like language (implemented in C using lex and yacc before TCL became available in *Windows NT*), that was used to reduce 2 meters of mil-spec testing documentation for the *Black Hawk* weapons targeting system into a *manageable and easily modifiable* set of requirements. The *DSL* had to support various H/W drivers for the test equipment used in the lab. The *DSL* language was simple enough to use that the system engineer, who had no programming background, could create *ad hoc* tests. Tests could run

unattended for up to 100 hours without any memory leaks.

3. **1992-95: Invented** a *DSL* (implemented in C using lex and yacc) that was the backbone for a software development tool chain for the DSPG PINE CPU:
    - The tool chain included a clock accurate CPU simulator, debugger with a programmable I/O port simulator (BTW still, in 2025, not yet found on most debuggers or emulators), assembler, disassembler, and overlay linker. (Note that this project was implemented before the stable version GCC 2.95 became available). Initially the purpose of the *DSL* was to resolve the problem of register usage restrictions between adjacent assembly instructions. The VLSI architects regularly changed the restrictions which made it extremely difficult to create a hand crafted assembler in a reasonable period of time. A register restriction violation required the manual insertion of a *NOP* into the instruction stream. (Today, i.e. 2025, modern CPU architectures can automatically "stall" the pipeline).
    - The *DSL* implemented a language that completely described the CPU's instruction set. Changes to the register restrictions were trivial to implement, and the assembler could be recompiled within an hour. **The assembler ran over 100x faster than the VLSI simulated assembler. The assembler ran for over 100 hours without any memory leaks. The assembler allowed the architects to regularly run instruction set sanity tests as soon as the architecture was updated. The complete tool chain was used by application programmers to build and test complete applications 6 months before the final production of the physical CPU.**

1990-91: Invented a *DSL* to completely automate a metal blade production factory at Iscar-Matkash, Tefen

- The plant included the following types of objects:
    - various types of workstations with multiple stands where pallets for parts and raw materials were placed, and from where the end product was removed
    - automatically guided robotic vehicles delivered pallets to and from the workstation stands
    - some pallets were transported by conveyor, and others placed on stacks
    - the storage region had room for hundreds of pallet stands
    - work-in-progress parts that were stored for too long had to oiled in order to prevent rust
    - tools that required maintenance, e.g. cutting tools, recorded their consumption before being sent for repair (e.g. sharpening)
    - etc.

- My job along with my co-architect, David Goldstein, was to automatically orchestrate the plant via S/W. We had a (then) powerful *VAX/VMS* computer at our disposal with a relational database. The project manager insisted that the main programming language be *Pascal*.

- We came to the project with no knowledge of factory automation. So we started with a month long mentoring course from an industrial engineer.

- Our solution was to design a *DSL* that described every single object (i.e. a few 100) in the factory's "object kingdom", and how they interacted with one another. The *DSL* had to configurable by the factory engineer who was not a software engineer, while the clerical staff required a GUI window into the DSL for making on-the-fly configurations and viewing the status of the factory.
- It became immediately obvious that the S/W required object oriented techniques to implement. At that time *object oriented programming (OOP)* was in its infancy, and barely used in industry. There was no Internet yet at that time, but literature was available in the Hebrew Univ computer science library. Recall that we had a hard requirement to use the procedural Pascal language.
- *VAX/VMS* had a command line utility for designing command "ensembles" which could be executed via callbacks to a compilable language, e.g. Pascal. We used these ensembles to define every type of object, along with their attributes, supported by the our DSL. Today, i.e. 2025, I would use *XML* instead (which had not yet been invented).
- There were tens of thousands object instances stored in a database on disk - including their configuration and current status. The database regularly stored status updates. The database was constantly scanned in order to implement the next task that our scheduler required. In the event that the main computer shut down, either expectedly or unexpectedly, the database allowed for a smooth restart/recovery.
- **After about a year, we implemented a working factory!**

1988-89: *Cubital* (subsidiary of Scitex), Herzliya:

- Early 3D printing (stereo-lithography) R&D.
- PC accessibility **invention** for quadriplegics: Invention of virtual keyboard using a telescopically modified CRT light pen, which could support a 800 mm. distance from display, combined with a sip and puff switch. Our first subject was a quadriplegic polio victim, with the light pen connected to her head via a women's hairband, who could type 30 characters per minute, and who productively worked as a book editor. The virtual keyboard could be configured so that only a single selected key was displayed on the screen, where the light pen hovered, which allowed over 95% of the screen to remain unobstructed. Too bad I didn't patent this virtual keyboard invention.

5. Patents & Inventions

Multiple patents pending and granted in fields including FLASH compression, software obfuscation, automotive updates, bioinformatics, and accessibility devices.

6. Education

1979: *York University*, Toronto Canada: MA Economics, minor in Applied Mathematics

- Simulated hydroelectric dam in FORTRAN for major project

1973-78: *University of Toronto*, Canada: Undergraduate School of Arts & Sciences, Rotman Graduate School of Management, Graduate School of Engineering

7. Teaching & Mentorship

Part-time instructor and mentor in various technical domains. Strong advocate of pairing with domain experts to accelerate onboarding and innovation.

8. Portfolio & More:

See details in Full CV.