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: *2025-07-23*

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, reverse engineering, and building MVPs and prototypes for CTO teams. Holds multiple patents and thrives on technically challenging projects.

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 WebAssembly (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 automatic 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.
- Optimized CPU and RAM usage, by a factor of 5x, of the company's core source code refactoring product for embedded automotive C code via srcML and Beautiful Soup.

2021: Morphisec, Beer Sheva: Anti-Reverse Engineering Modifications Of Linux Kernel

2021: Qedit, Tel Aviv: WASM Cybersecurity Consultant

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.
- 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, Vehicle Navigation

- Developed traffic jam reduction algorithms for automotive navigation, an early (failed) competitor to Waze.

2011: Syntezza Bioscience, Jerusalem: Bioinformatics consultant

- In just 3 months I invented an AI-driven PCR threshold algorithm and noise-reduction techniques for a MRSA diagnostic kit. When I first began work with this company, their kit was a complete failure. Even though I had zero background in micro-biology, I discovered that the PCR "control" test tubes were configured improperly in all of their 800 hospital samples, for which I able to correct after-the-fact even though the samples were long destroyed. And I discovered that their preliminary chemistry was not able to filter the inhibitors in their nasal samples. I invented a new extremely accurate algorithm for "Ct" detection in spite of 50+% of inhibited data points, and which worked on pure MRSA colonies, both of which resulted in non-sigmoidal shaped graphs of the underlying biological assays which was the fundamental assumption of industry standard PCR algorithms. The 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: Cybersecurity Researcher for Internet TV industry

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

- Invented web browser based solution enabling the blind to interpret images and maps via a graphics tablet and sound card.

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.

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

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

Invented Domain Specific Languages (DSLs) for:

1. **1996-97:** 2D conoscopic holographic laser metrology device for Optimet-Ophir (subsidiary of MKS-Newport US), Jerusalem

    - The DSL (implemented in TCL) was used to scan objects mounted on a XY jig. Due to lighting conditions on the object, different regions required different pre-defined 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

    - The DSL, a BASIC-like language (implemented in C using lex and yacc before TCL became commonly available) 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 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:** completely automated factory for Iscar-Matkash, Tefen

    - The plant included the following types of objects:

        - various types of workstations that had multiple stands where parts and raw materials were placed, and the end product was removed
        - robotic arms automatically inserted parts and raw materials into the station, and removed the end product
        - robotic automatically guided vehicles delivered and removed pallets from the workstations to the correct stand, and moved them to the conveyor system or to the storage area
        - some pallets were transported by conveyor
        - some pallets were placed on stacks
        - pallets were temporarily stored in a storage area made up of hundreds of 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)
    - My job along with my co-architect was to automatically orchestrate the plant via S/W. We had a (then) powerful VAX/VMS computer at our disposal with a relational database. Our liaison from DEC insisted that the main programming language be Pascal.
    - We came to the project with no knowledge of factory automation. So we started with month long mentoring from an industrial engineer who did.
    - Our solution was to design a domain specific language (DSL) that described every single object (i.e. a few hundred 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 just being developed in research/university settings. The first version of C++ was just released, and SNOBOL was an OOP research prototype. 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 (2025), 30 years later, 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!
4. **1990-92:** The DSL (implemented in C using lex and yacc) was the backbone for a software development toolchain for the DSPG PINE CPU:
    - The toolchain included a CPU simulator, debugger with a programmable I/O port simulator (BTW still, in 2025, not found on most debuggers or emulators), assembler, disassembler, and overlay linker. (Note that this project was implemented before the stable GCC 2.95 became available). Initially the purpose of the DSL solved the problem of the registry usage restrictions between adjacent assembly instructions that the VLSI architects changed regularly which made it extremely difficult to create a hand crafted assembler in a reasonable period of time. A restriction violation required the manual insertion of a NOP into the instruction stream.

    - The DSL implemented a language that completely described the CPU's instruction set. Changes to the 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 toolchain was used by application programmers to build and test complete applications 6 months before the final production of the physical CPU.

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, 800 mm. distance from display, combined with a sip and puff switch. Enabled a quadriplegic polio victim, with the light pen connected to her head via a women's hairband, to type 30 characters per minute to become 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.