# Avraham “Abe” Bernstein | Abbreviated Professional CV

*Version: 2.1.0-abbrev*

*Last update: 2017-08-21T19:12:59Z*

*Copyright © 2017 Avraham Bernstein, Jerusalem ISRAEL. All rights reserved.*

*License: Except where otherwise noted, this work is licensed under the Creative Commons License* [*CC BY-ND 4.0*](https://creativecommons.org/licenses/by-nd/4.0/)



Secure[[1]](#footnote-2) photo of the author, Avraham Bernstein c. 2010.

### 0.1 Contact Info & Links

**email/skype:** [Avraham.Bernstein@gmail.com](mailto:Avraham.Bernstein@gmail.com) [[2]](#footnote-3)

**pgp-id:** [E6647D2F](https://pgp.mit.edu/pks/lookup?op=vindex&search=0x86EFCDAEE6647D2F)

**geolocation:** [Jerusalem ISRAEL](https://www.google.com/maps/place/31.808472,35.200944), **tz:** UTC +2:00/+3:00 [winter/summer]

**tel-IL-mobile/whatsapp:** +972.54.641-0955 **[preferred]**

**tel-US-mobile:** +1.845.402-0023

**www-home:** <http://purl.org/Avraham.Bernstein>

**cv-full-html:** <http://purl.org/Avraham.Bernstein/cv.html> **[preferred]**

**cv-abbrev-html:** <http://purl.org/Avraham.Bernstein/cv-abbrev.html>

**cv-full-docx:** <http://purl.org/Avraham.Bernstein/cv.docx>

**cv-abbrev-docx:** <http://purl.org/Avraham.Bernstein/cv-abbrev.docx>

**linkedin:** <https://www.linkedin.com/in/AvrahamBernstein>

### 0.2 Table of Contents

[Avraham “Abe” Bernstein | Abbreviated Professional CV 1](#__RefHeading___Toc1204_1374534403)

[0.1 Contact Info & Links 1](#__RefHeading___Toc1206_1374534403)

[0.2 Table of Contents 2](#__RefHeading___Toc1208_1374534403)

[1.0 Summary 3](#__RefHeading___Toc1210_1374534403)

[2.0 Work Experience 4](#__RefHeading___Toc1212_1374534403)

[2017 - present: Consultant: Cybersecurity 4](#__RefHeading___Toc1214_1374534403)

[2011-17: Security Manager & Architect: OTT Internet Pay TV System: Secure Player 4](#__RefHeading___Toc1216_1374534403)

[2016-16: Consultant: Cybersecurity Protection of a Small Business With Extremely High Security Concerns 5](#__RefHeading___Toc1218_1374534403)

[2010-12: Part-time CTO: US Agricultural Trading Company with Nigeria 5](#__RefHeading___Toc1220_1374534403)

[2010-11: VP R&D: Urban Traffic Vehicle Route Guidance 5](#__RefHeading___Toc1222_1374534403)

[2010-10: Consultant: Transparent Technical Negotiator of Restricted Hi-Tech Trade with China 6](#__RefHeading___Toc1224_1374534403)

[2009-09: Consultant: Bioinformatics PCR Algorithms 6](#__RefHeading___Toc1226_1374534403)

[2004-09: Security Research & Business Development 6](#__RefHeading___Toc1228_1374534403)

[2002-03: CTO: Blind Accessibility H/W & S/W to Enable Viewing of Digital Images & Maps 7](#__RefHeading___Toc1230_1374534403)

[1999-02: Manager Cable Modem & Router Network Utilities S/W Development Group 8](#__RefHeading___Toc1232_1374534403)

[2001-01: Consultant: Network Management System (NMS) for FSO Devices 9](#__RefHeading___Toc1234_1374534403)

[2000-01: Contract Programmer: Win32 Asynchronous Network Driver DLL for a VisualBasic Project 9](#__RefHeading___Toc1236_1374534403)

[1998-99: Contract Programmer: GCC Compiler Port for a 128-Core Stack Machine 9](#__RefHeading___Toc1238_1374534403)

[1997-98: Contractor: S/W Architect & Implementation of Conoscopic Interferometer Workstation 9](#__RefHeading___Toc1240_1374534403)

[1995-96: CEO & CTO: S/W Architect & Implementation of US DOD Mil-Spec Automated Testing System: Night Hawk Fire Control System 9](#__RefHeading___Toc1242_1374534403)

[1991-94: CEO & CTO: S/W Architect & Implementation of S/W Toolchain For DSPG PINE CPU 9](#__RefHeading___Toc1244_1374534403)

[1989-91: S/W Architect & Implementation: Shop Floor Production Control (SFPC) System: BARI II 9](#__RefHeading___Toc1246_1374534403)

[1988-88: S/W Architect & Implementation: Quadriplegic PC Accessibility 10](#__RefHeading___Toc1248_1374534403)

[1987-88: Senior Programmer & VAX/VMS Sysadmin: 3D Printer: Solider 11](#__RefHeading___Toc1250_1374534403)

[1986-86: Programmer: Automated Leather Sewing Workstation 11](#__RefHeading___Toc1252_1374534403)

[1984-85: Programmer & VAX/VMS Sysadmin: Hebrew/English Word Processor: Glyph 11](#__RefHeading___Toc1254_1374534403)

[1983-84: Programmer: Real-Time 11](#__RefHeading___Toc1256_1374534403)

[1981-83: Programmer & IBM CP/CMS Sysadmin 11](#__RefHeading___Toc1258_1374534403)

[1979-80: Programmer/Economist 11](#__RefHeading___Toc1260_1374534403)

[1977-78: Intervenor/Economist 11](#__RefHeading___Toc1262_1374534403)

[4.0 Education 11](#__RefHeading___Toc1264_1374534403)

[4.1 Formal Education 11](#__RefHeading___Toc1266_1374534403)

[1979: York University, Canada: MA Economics & Applied Mathematics 11](#__RefHeading___Toc1268_1374534403)

[1977: University of Toronto - Rotman School of Management (MBA Program): No Degree 12](#__RefHeading___Toc1270_1374534403)

[1976: University of Toronto: BA Economics & Applied Mathematics 12](#__RefHeading___Toc1272_1374534403)

[4.2 Continuing Education 12](#__RefHeading___Toc1274_1374534403)

[5.0 Spoken Languages 12](#__RefHeading___Toc1276_1374534403)

[6.0 Computer Languages, SDKs, and Operating Systems 13](#__RefHeading___Toc1278_1374534403)

[7.0 Patents Under Development 13](#__RefHeading___Toc1280_1374534403)

[8.0 Personal 13](#__RefHeading___Toc1282_1374534403)

[Colophon 13](#__RefHeading___Toc1284_1374534403)

## 1.0 Summary

I am an experienced computer scientist and S/W architect. I have devised innovative solutions to many S/W problems for a wide range of fields, including [cybersecurity](#viaccess), [cryptography](#patents), [bioinformatics](#syntezza), computer languages, [factory automation](#sfpc), [telecommunications](#vyyo), [blind vision](#virtouch), [accessibility](#light-pen), [transportation vehicle route guidance](#telequest), and [test automation](#elop). I have worked at executive level for a number of organizations, large and small, and helped them realize improvements in their product performance, often putting them in the front rank in their field. I have acquired expert knowledge in a number of fields, often liasing with noted experts, and have been able to quickly apply this knowledge to improve the competitive position of the companies and their products. I have a keen interest in computer languages, both practical and theoretical. I have created a number of [domain specific languages (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) that were instrumental in greatly simplifying seemingly intractable problems.

For the past 13 years, I worked primarily in the field of cybersecurity with a specialty in [S/W obfuscation](https://en.wikipedia.org/wiki/Obfuscation_(software)), i.e. anti-reverse engineering techniques. Most recently I was head of security for an Internet pay TV product where the client devices were Android and IOS. The security critical portion of the application was implemented in native code, i.e. C/C++, and not in Java or Objective-C that were used solely for the UI. We obfuscated the native code. During my 6 year tenure we had 40M subscribers, and no hacks in the field.

I am still very much a “hands-on” S/W engineer too. Typically I work on novel and complex problems, where I must first create a working proof-of-concept before I can produce a specification that is firmly grounded in reality. I am a consummate tool maker, where my tools are algorithms, filters, and domain specific languages.

Usually I am able to apply knowledge and [meta-knowledge](https://en.wikipedia.org/wiki/Metaknowledge) that I have acquired from other fields, and apply it to the new one - what educational psychologists refer to as [transfer of learning](https://en.wikipedia.org/wiki/Transfer_of_learning). A common technique that I use is to create a [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) that allows me to write a formal specification of a problem. With a little ingenuity the DSL can be used to create a working command line application, and it usually forms the basis of an automated testing framework. I successfully used a DSL to describe a [shop floor production control](#sfpc) system, along with a compiler that executed/implemented the shop floor. And I used a DSL to describe a [CPU architecture](#dspg), along with a compiler that automatically generated a simulator/debugger, assembler, and C compiler. A common programming technique that I use is [metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming) with templates that allows me to automatically generate code by creating intelligent macros regardless of the underlying target programming language.

I am an [expert generalist](http://99u.com/articles/7269/picasso-kepler-and-the-benefits-of-being-an-expert-generalist), and an [autodidact polymath](http://autodidactpolymath.com/autodidact-polymath-definition/), i.e. a self-learner in new fields who achieves expertise quickly - often with the help of a good mentor. Also I am a good mentor myself. **I slavishly attend to my own** [**continuing education**](#continuing-ed)**.** I have a strong background in business and economics. I communicate well both orally and in writing. I am a native English speaker.

## 2.0 Work Experience

### 2017 - present: Consultant: Cybersecurity

*@Self-Employed, Jerusalem:*

1. Developing my own [obfuscating](https://en.wikipedia.org/wiki/Obfuscation_(software)) [compiler](https://en.wikipedia.org/wiki/Compiler) for C/C++ and for [Web Assembly (WASM)](http://webassembly.org/). Still in stealth mode.
2. Collaborating with the compiler company Semantic Designs [Legacy Software Migration](http://www.semdesigns.com/Products/Services/LegacyMigration.html) tools for projects in Israel.
3. I am a mentor for the Jerusalem [Mass Challenge](http://israel.masschallenge.org/) start-up hub.

### 2011-17: Security Manager & Architect: OTT Internet Pay TV System: Secure Player

*@Viaccess-Orca, Ra’anana - a subsidiary of Orange FR, and @Discretix/SansaSecurity, Netanya - now merged into ARM:*

1. I was head of security for their [Over-The-Top (OTT)](https://en.wikipedia.org/wiki/Over-the-top_content) Internet pay TV product for 6 years. I joined the project shortly after its inception. When I left there were 40M subscribers, with no successful subscriber hacks/penetrations in the field.
2. I was responsible for end-to-end security policy and architecture, and inclusion and implementation of security related programming utilities.
3. The supported client devices were Android and IOS phones and tablets.
4. The original [DRM](https://en.wikipedia.org/wiki/Digital_rights_management) was [PlayReady](https://en.wikipedia.org/wiki/PlayReady). Later we supported Google [Widevine](http://www.widevine.com/), and the [Viaccess-Orca](http://www.viaccess-orca.com/content-protection.html) proprietary DRM.
5. In a nutshell, my primary job was to protect the content, content decryption keys, and subscriber keys, on the client devices. The security critical modules were written in native code, i.e. C/C++, and *not* Java or Objective-C which were used solely for the UI. Obfuscation was designed to protect the native code from reverse engineering. I made it nearly impossible for an attacker to engage in *static* [reverse engineering](https://en.wikipedia.org/wiki/Reverse_engineering), and made it extremely difficult to use the state-of-the-art *dynamic* reverse engineering tools such as the [IDA Hex-Rays](https://www.hex-rays.com/) decompiler/debugger.
6. We supplied a reference application to our customers who were our [CA](https://en.wikipedia.org/wiki/Conditional_access) broadcaster/operator customers who needed to provide an [OTT](https://en.wikipedia.org/wiki/Over-the-top_content) option to their subscribers. They customized our reference application to their needs. We did not have a direct relationship with the subscribers. Our operators typically took a very long time to do their integration and QA before they would release/push their application to their subscribers. The fundamental technical flaw with this business model was that when we found a serious bug or a security problem in a core module, the operators would typically delay for 6-12 months before pushing the update to their subscribers. We had no timely way to update a key that had been compromised! Eventually I architected a solution that provided good enough security , while still satisfying the QA needs of our customers.
7. Some S/W security tools we purchased, namely the [InterTrust/WhiteCryption](https://www.intertrust.com/products/application-security/) [obfuscating](https://en.wikipedia.org/wiki/Obfuscation_(software)) compiler and white-box encryption tools.
8. Many S/W security tools I architected and implemented in-house, including an Android root detection algorithm.
9. Unfortunately the bottom line with respect to security is that no matter how well protected the client S/W is, you have to expect that it will eventually be hacked. Without pushing a new randomly obfuscated version, say every 60-90 days, no protection scheme will remain secure.

### 2016-16: Consultant: Cybersecurity Protection of a Small Business With Extremely High Security Concerns

### 2010-12: Part-time CTO: US Agricultural Trading Company with Nigeria

### 2010-11: VP R&D: Urban Traffic Vehicle Route Guidance

*@TeleQuest, Jerusalem:*

1. I designed algorithms and a computational infrastructure similar to what [Waze](https://www.waze.com/) does today.
2. I implemented these algorithms on Tokyo traffic simulations which is the most traffic congested city in the Western world, and who happen to have a superb traffic data collection system.
3. I built [AWS](https://aws.amazon.com/) cloud infrastructure to run real-time traffic simulations.
4. The simulation S/W was written in Java. Due to the huge size of the simulations, I needed to optimize the Java code - in particular to control garbage collection.
5. The results of the Tokyo simulations were that with a threshold of just 20% guided vehicles on the roads, we could achieve up to a 50% performance decrease in travel times for guided vehicles, with a side effect of 40% decrease for non-guided vehicles. These results were the exact opposite of the competition which generated their own secondary traffic jams once the guided vehicle threshold reached 20% due to their simplistic opportunistic algorithms. I managed a group of 6 people including programmers and mathematicians.

### 2010-10: Consultant: Transparent Technical Negotiator of Restricted Hi-Tech Trade with China

### 2009-09: Consultant: Bioinformatics PCR Algorithms

*@Syntezza, Jerusalem:*

1. Within 3 months of joining the company and quickly learning the basics of bioinformatics, I discovered a new algorithm using Artificial Intelligence (AI) and Data Science techniques for handling [PCR inhibition](https://en.wikipedia.org/wiki/Polymerase_chain_reaction_inhibitors) - which I am in the process of [patenting](#patents).
2. The client’s product was a [MRSA](https://en.wikipedia.org/wiki/Methicillin-resistant_Staphylococcus_aureus) (= a lethal staphylococcus bacteria that is antibiotic resistant and thrives in hospitals) [PCR](https://en.wikipedia.org/wiki/Polymerase_chain_reaction) (= DNA amplification/duplication technology) detection kit. Typically the kit would be used in hospitals to determine whether a newly admitted patient was a MRSA carrier. A PCR test can return results within 2 hours, whereas the “gold standard” [Petri dish](https://en.wikipedia.org/wiki/Petri_dish) test, which was developed over 100 years ago, takes at least 36 hours. Given that MRSA is infectious, and potentially fatal, it is important for hospitals to be aware ASAP which patients are carriers.
3. When I started the project, the client’s PCR kit had only a 50% detection rate due to inhibition problems. This result was grossly unacceptable for any medical test. The investors had lost confidence, and were about to pull out.
4. I improved the test’s accuracy to 95%, which was 10% better than their competitors from the pharmaceutical giants. I saved the client from liquidation.
5. I stood on the shoulders of the person who developed the original [PCR Ct](http://www.thermofisher.com/il/en/home/life-science/pcr/real-time-pcr/qpcr-education/pcr-understanding-ct-application-note.html) function analytical detection algorithm, [Dr. Tzachi Bar](https://scholar.google.co.il/citations?user=XokRoVQAAAAJ&hl=en), who was my mentor, and who got me up to speed so quickly.

### 2004-09: Security Research & Business Development

*@Cisco-NDS, Jerusalem: NDS primary business was providing* [*Conditional Access (CA)*](https://en.wikipedia.org/wiki/Conditional_access) *pay TV security systems to satellite TV broadcasters:*

1. I worked on a wide variety of security related projects. My background task was to do code security reviews. Typically secure coding is achieved by [adhering to best programming practices](https://www.youtube.com/watch?v=eL5o4PFuxTY).
2. I was a member of the architecture team for their in-house [LLVM](https://llvm.org/) [obfuscating](https://en.wikipedia.org/wiki/Obfuscation_(software)) compiler.
3. I developed techniques using Virtual Machine (VM) technology to crack [Digital Rights Management (DRM)](https://en.wikipedia.org/wiki/Digital_rights_management) schemes, and to subvert the random number generators which are the core initialization process for all cryptographic algorithms.
4. I worked with the operational security team to track hackers, and to provide stealth techniques for our researchers.
5. I arranged for world class security researchers to give seminars, (1) a week long reverse engineering seminar, and (2) a 2 day seminar how to hack with virtual machines.
6. I wrote the technical bid for securing the 2008 Olympics Beijing Olympics TV broadcasts.
7. I researched business cases for providing security services to the computer gaming industry, and for printer ink cartridges. For example sales revenue from premier games can gross $60M in their first 2 weeks, by which time they are typically hacked. Improving security so that the hack can be delayed for another 2 weeks, can increase revenue by 50%.
8. I advised senior management on the technical and business merits of potential acquisitions.
9. I did a very interesting and important non-security related project for senior management, by [data mining](https://en.wikipedia.org/wiki/Data_mining) the company’s bug database, that showed that 25% of S/W development manpower was wasted on fixing bugs. And I showed them simple techniques that could reduce this number by 80% .
10. I designed a hybrid simulator/emulator debugger for legacy [Set-Top Boxes (STB)](https://en.wikipedia.org/wiki/Set-top_box) that originally could be debugged only with printf statements to log files. My new debugger allowed source code on the PC to be debugged using the [MS Visual Studio IDE](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio) debugger while still viewing the results on the STB. Implementation was accomplished by reverse engineering the STB middleware API. 80% of the middleware ran natively on the PC, while the STB low level H/W specific portions were implemented via an agent on the STB that was accessed via API calls that were implemented as [Remote Procedure Calls (RPC)](https://en.wikipedia.org/wiki/Remote_procedure_call).
11. I wrote an automated testing system for a satellite content delivery system for huge content, e.g. delivering ultra high definition movies to cinemas, and print newspapers for remote publishing. I created a [Domain Specific Language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) in order to execute the satellite operations. After studying the Win32 source code of the satellite ground control station, I detected a major conceptual flaw which the architect refused to believe (because testers are not supposed to understand Win32 internals!). So I wrote a progressive test that brought the satellite to its knees at only 25% of its rated capacity. Afterwards the development team used my tool to develop their own unit test scripts, and to execute a system sanity test before checking-in any changes to the source control system.

### 2002-03: CTO: Blind Accessibility H/W & S/W to Enable Viewing of Digital Images & Maps

*@Virtouch, Jerusalem:*

1. My task was to change the company’s focus from providing tools for teaching blind children Braille skills to instead provide tools for blind adults to use at work.
   * The fact of the matter is that government social services agencies around the world are much more concerned about funding tools that allow the blind to integrate into the work force, and thus become economically self sufficient, versus funding educational tools for blind children.
2. Previously the company had developed a [tactile mouse](http://www.avrahambernstein.com/resources/vtplayer.png) that was far too expensive for our target audience (~$350). Also the blind find it very difficult to use mice because mouse movement must be coordinated with the cursor on the screen which they cannot see.
3. I developed new algorithms that allowed the blind to read [GIS](https://en.wikipedia.org/wiki/Geographic_information_system) maps using SVG and XML combined with audio cues (by using pitch and volume), and by using an off-the-shelf graphics tablets and stylus (that were much less expensive than the tactile mouse).
   * The blind prefer graphic tablets because they inherently have accurate spatial knowledge of their position on the tablet because they can feel it.
   * Additionally using two different instruments, say a piano and flute each with a 6 octave range with 7 or 12 notes per octave (i.e for a major scale or chromatic scale respectively), can provide an easily discernable sonic spatial grid.
   * Color information, after being converted to an 8-bit gray scale, can be conveyed via volume.
   * The S/W needs to make a rectilinear mapping of the tablet into the whole screen, or into an O/S window.
   * The S/W needs the ability to do seamless zooming.
4. Modern GIS based maps are made of many overlapping layers, similar to the overlay techniques used in anatomy textbooks. Some of the layers commonly found in maps are the following:
   * roads
   * public transit
   * sites of interest
   * district/political boundaries
   * cities and their relative sizes
   * topography
   * geographic features such as bodies of water versus land, desert, marsh,
   * vegetation
   * precipitation
5. Printed maps are able to display many layers simultaneously by intelligent uses of colors, textures, and fonts.
6. It is possible to provide a blind friendly way to present the same information by providing user selectable filters that greatly reduce clutter by selecting which layers to display, and by reducing granularity of the layers (e.g. instead of displaying different elevations using 50 meter contour isolines, it is just as easy to for the user to select 500 meter isolines). The isolines can be discerned sonically. And information associated with regions can be presented by hovering over a region, or by mouse clicking, or by key presses. And the presentation is enhanced by seamless zooming.
7. Well designed consistent mapping schemas, such as those created by the [USGS](https://www.usgs.gov/products/maps/topo-maps), can usually be automatically parsed.
8. The same techniques could be used for displaying interactive maps of mathematical functions and geometry, or to manually [mark up](https://en.wikipedia.org/wiki/Markup_language) (i.e. annotate) other images.
9. In order to quickly create lots of content, a public domain repository should be established, and the mark up scheme should be open sourced.
10. I *almost* saved the company from liquidation. My research was awarded a European FP6 grant of $0.5M Euro that required *matching funds*. But the investors refused to put up the matching funds due to the company’s long history of financial failure in the children’s market.
11. The most important business lesson that I learned was that building special purpose H/W for the disabled is a recipe for bankruptcy; but if you can repurpose off-the-shelf H/W, typically designed for gamers, then you can financially succeed in the accessibility/disability market.
12. The most important personal lesson that I learned was how satisfying it was working with the blind volunteers, and supplying them with a tool that allowed them to “see”.

### 1999-02: Manager Cable Modem & Router Network Utilities S/W Development Group

*@Vyyo, Jerusalem:*

I managed a team of 3-6 programmers.

### 2001-01: Consultant: Network Management System (NMS) for FSO Devices

### 2000-01: Contract Programmer: Win32 Asynchronous Network Driver DLL for a VisualBasic Project

### 1998-99: Contract Programmer: GCC Compiler Port for a 128-Core Stack Machine

### 1997-98: Contractor: S/W Architect & Implementation of Conoscopic Interferometer Workstation

### 1995-96: CEO & CTO: S/W Architect & Implementation of US DOD Mil-Spec Automated Testing System: Night Hawk Fire Control System

### 1991-94: CEO & CTO: S/W Architect & Implementation of S/W Toolchain For DSPG PINE CPU

*@Pitkha outsourcing, Jerusalem, for @DSPG, Givat Shmuel:*

1. I was the S/W architect of a clock accurate DSP CPU simulator along with a complete software development toolchain, i.e. a debugger, C compiler, assembler and linker. Note that the system was developed just *before* the GNU Compiler Collection framework reached maturity, i.e. v2.95.
2. This system enabled working applications to be developed before the chip became physically available. It reduced application time-to-market by 6-12 months.
3. The technological breakthrough was my design of a [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) implemented in [lex/yacc](https://en.wikipedia.org/wiki/Yacc) that described the CPU architecture.
4. The associated DSL compiler automatically generated the source code for the complete toolchain that enabled it be automatically rebuilt within an hour in the face of almost daily changes to the VLSI architecture - especially the pipeline.

### 1989-91: S/W Architect & Implementation: Shop Floor Production Control (SFPC) System: BARI II

*@Digital Equipment Corporation (DEC), Herzliya, for @Iscar, Tefen:*

1. Iskar Matkash in Tefen IL is a fully automated factory that produces thousands of different cutting blades using a sintering process. The raw materials go through many stages of operations. In many cases after undergoing intermediate processing, the partially processed material can still be diverted to multiple final products - similar to stem cells. The factory contains hundreds of automated workstations, stands, stacks, guided vehicles, and conveyor belts. The product or intermediate product is placed on pallets. The pallets are moved from one stand on a workstation to a stand on another workstation, or temporarily to a storage stand or stack.
2. My task was to create a computer program that automatically operated/orchestrated the factory.
3. When my co-architect and I started this project, we had zero background in industrial engineering. We were supplied with a mentor who brought us up to speed.
4. Eventually after months of discussions we created an architecture that was a textbook object oriented taxonomy - a “factory object kingdom”. The top level object was a “production instruction”.
5. We defined the attributes and methods associated with each object.
6. We created a descriptive, i.e. *non-procedural*, [domain specific language (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language) that was designed to be user-friendly for the factory engineer.
7. I wrote the language manual.
8. We used the language to configure the factory. We created a UI wrapper for the language which emitted CLI script.
9. We mapped the language to a relational database.
10. We created the implementation architecture in Pascal. Given the inherent object oriented (OO) nature of the architecture, C++ would have been an ideal implementation language choice - but the project management refused.
11. After 18 calendar months, and 6 man-years later, the factory ran perfectly!

### 1988-88: S/W Architect & Implementation: Quadriplegic PC Accessibility

*@Cubital, Herzliya - a charity project funded by the company and the CEO* [*Itzhak Pomerantz*](https://il.linkedin.com/in/itzhak-pomerantz-91500b3)*:*

1. First of all, it important to note that this project took place in 1988 when speech-to-text technology was still in its infancy, and exhorbitantly expensive.
2. The H/W used for this project was the following:
   1. A [light pen](https://en.wikipedia.org/wiki/Light_pen), i.e. an obsolete pre-mouse point and click device that synchronizes with the trace signal of the CRT video display, outfitted with special light weight military optics that increased its effective range from 5 mm from the screen to 800 mm.
   2. A standard accessibility [sip-and-puff](https://en.wikipedia.org/wiki/Sip-and-puff) switch.
3. The light pen was mounted on the user’s head by using a sturdy woman’s plastic hair head band, while the sip-and-puff straw replaced the button on the light pen.
4. The S/W that I developed overlaid a virtual keyboard on top of the screen.
5. In novice mode aiming the pen at a key caused it to illuminate, while clicking on it entered it into the system as a virtual key stroke.
6. The problem with novice mode was that the virtual keyboard obstructed more than half the screen.
7. In expert mode, the virtual keyboard was hidden, but when the pen was aimed at an individual key, it would pop-up and become illuminated.
8. This system was used to enable Shulamit Gabbai, a former school teacher who became quadriplegic by contacting polio (due to a terrible malfunction in the *Or Akiva* drinking water supply which became mixed with sewage), to become a book editor for *Maariv*. She was able to type 30 characters per minute.

### 1987-88: Senior Programmer & VAX/VMS Sysadmin: 3D Printer: Solider

### 1986-86: Programmer: Automated Leather Sewing Workstation

### 1984-85: Programmer & VAX/VMS Sysadmin: Hebrew/English Word Processor: Glyph

### 1983-84: Programmer: Real-Time

*@Elta/IAI via DSI, Ashdod:* This was my first job upon making [Aliya](https://en.wikipedia.org/wiki/Aliyah) to Israel. Before receiving my security clearance, I worked on the development of a data collection system. I wrote a [RTOS](https://en.wikipedia.org/wiki/Real-time_operating_system) kernel for the 8080 CPU because at the time no off-the-shelf alternative was available. My mentor was Menachem Malkosh. It was a formative learning experience. After receiving my clearance, I worked on the radar system for the [Lavi fighter plane](https://en.wikipedia.org/wiki/IAI_Lavi).

### 1981-83: Programmer & IBM CP/CMS Sysadmin

### 1979-80: Programmer/Economist

### 1977-78: Intervenor/Economist

*@Ontario Energy Board (OEB), Toronto:*

1. I was an [intervenor](https://en.wikipedia.org/wiki/Intervention_(law)) at the ECAP’77 costing and pricing hearings.
2. Like all citizens, I had legal standing because I paid an electric bill.
3. I took over the [marginal cost pricing](https://en.wikipedia.org/wiki/Marginal_cost) (= peak load or time-of-day pricing) proposal of the Ontario Hydro (i.e. at that time the name of Ontario’s electric generation and transmission utility) economists who were forced to drop their case due to extreme political pressure from the metallurgy processing industry who at the time relied upon electric blast furnaces.
4. I actively participated in the hearing sessions for about 9 months. I filed submissions, gave expert testimony, and cross-examined opposing witnesses.
5. I argued my position very well. At 22 years old, I was the first public interest intervenor in the history of the OEB to be awarded costs.
6. I published an op-ed in [The Globe and Mail](https://www.theglobeandmail.com/), i.e. at the time Canada’s newspaper of record, explaining the economic and political issues surrounding the case.

## 4.0 Education

### 4.1 Formal Education

#### 1979: York University, Canada: MA Economics & Applied Mathematics

I passed my final examinations in economic theory before I even started the program, so in order to achieve my required course credits the faculty agreed to allow me to be a special student at the Univ. of Toronto graduate faculty of engineering, where I took the majority of my courses. My major project was a computer simulation in Fortran how to cost efficiently operate a hydro-electric dam. I was mentored by an economist from Ontario Hydro (i.e. at that time the name of Ontario’s electric generation and transmission utility).

#### 1977: University of Toronto - Rotman School of Management (MBA Program): No Degree

I “dropped out” of school in the middle of the year after taking an advanced micro-economic theory course which analyzed the Ontario Hydro [marginal cost pricing](https://en.wikipedia.org/wiki/Marginal_cost) submission to the Ontario Energy Board (OEB), in order to take advantage of the unusual opportunity to present Ontario Hydro’s case for them, which they were forced to drop for political reasons ([see above](#oeb)). The following year I was able to apply my course credits to an MA Economics program at York Univ ([see above](#york-univ)).

#### 1976: University of Toronto: BA Economics & Applied Mathematics

The most memorable and still useful courses I took were in statistics, experimental design, game theory, advanced calculus, and microeconomics.

In 1971 at the age of 15, for a highschool computer science course, I wrote a computer program to play a perfect game of 3D 4x4x4 [tic-tac-toe](https://en.wikipedia.org/wiki/Tic-tac-toe) in Fortran on an [IBM 1130](https://en.wikipedia.org/wiki/IBM_1130). The computer had 16 KB RAM, and was the size of a refrigerator. It was arguably my most formative learning experience from which I received the computer programming “bug” which I carry with me to the present day.

### 4.2 Continuing Education

1. Today the field of computer science is changing so rapidly, that without ongoing self-study, one’s formal education becomes obsolete within 5 years.
2. From 1991-96, while I was CEO & CTO of Pitkha Outsourcing, I used to spend one afternoon per week reading at the Hebrew University Jerusalem (HUJI) computer science library.
3. Afterwards with the advent of the Internet, more up-to-date computer science topics were available on the Internet, so going to the library was no longer the most efficient way to keep updated.
4. Since 2005, I have maintained a subscription to the [O’Reilly Safari](https://www.safaribooksonline.com/) on-line tech library.
5. My daily dose of tech news comes from [Slashdot](https://www.safaribooksonline.com/).
6. I regularly watch [TedX](https://www.ted.com/about/programs-initiatives/tedx-program) and [Talks At Google](https://talksat.withgoogle.com/) video seminars.
7. The most fascinating feature of TedX talks is to watch and learn how world class experts in a wide range of fields are able to distill their special area of knowledge to intelligent laymen in just 18 minutes. Whenever I make a presentation, I attempt to emulate the best TedX speakers. Also I attempt to write presentations which emulate this TedX [Art of Innovation Top 10 Format](https://www.youtube.com/watch?v=Mtjatz9r-Vc).
8. I regularly read the tech sections of the Israeli business newspapers [Globes](https://www.globes.co.il/en/) and [The Times of Israel](http://www.timesofisrael.com/start-up-israel/).
9. I have ecclectic interests.
10. I regularly research new topics in depth.
11. My browser bookmarks are my most important professional store of my knowledge. I use Firefox because it has the best built-in bookmarking feature, because it uses tags/labels. I have a well honed tag taxonomy.
12. Therefore I really am an [expert generalist](http://99u.com/articles/7269/picasso-kepler-and-the-benefits-of-being-an-expert-generalist), and an [autodidact polymath](http://autodidactpolymath.com/autodidact-polymath-definition/), i.e. a self-learner in new fields who achieves expertise quickly.

## 5.0 Spoken Languages

1. English (5/5)
2. Hebrew (4/5)
3. French (2/5)

## 6.0 Computer Languages, SDKs, and Operating Systems

Language knowledge in order of expertise, based upon my current frequency of usage:

1. C, TCL, bash + posix text utilities, e.g. awk, sed, etc.
2. C++, python, make, html5, css, markdown, pandoc, jinja2
3. flex, bison, llvm, javascript, java, yaml, json, go
4. forth, lisp, prolog, apl, fortran, opengl, svg, xml schema, relax ng, xslt, perl, C#

Note that I write compilers and [Domain Specific Languages (DSL)](https://en.wikipedia.org/wiki/Domain-specific_language), so learning a new language takes me only a few days.

O/S knowledge in order of expertise, based upon my current frequency of usage:

1. Linux
2. Android
3. IOS
4. Win32

## 7.0 Patents Under Development

1. [Bioinformatics](https://en.wikipedia.org/wiki/Bioinformatics): (a) An extremely accurate and simple noise reduction and normalization algorithm to improve the accuracy of the standard [PCR Ct](http://www.thermofisher.com/il/en/home/life-science/pcr/real-time-pcr/qpcr-education/pcr-understanding-ct-application-note.html) calculation, and (b) an [Artificial Intelligence (AI)](https://en.wikipedia.org/wiki/Artificial_intelligence) methodology for measuring the quantity of DNA in a bioassay where [inhibition](https://en.wikipedia.org/wiki/Polymerase_chain_reaction_inhibitors) makes it impossible to estimate the Ct because no underlying [logistic function](https://en.wikipedia.org/wiki/Logistic_function) (= a flat “S” shaped curve) exists.
2. [Cryptography](https://en.wikipedia.org/wiki/Cryptography): A set of non-linear cryptographic primitives using [Hamming weight](https://en.wikipedia.org/wiki/Hamming_weight)-like [data dependent permutations](https://link.springer.com/article/10.1007%2Fs00145-001-0012-9?LI=true) which overcomes the well known limitation of using Hamming weights because they have a [binomial distribution](https://en.wikipedia.org/wiki/Binomial_distribution).

## 8.0 Personal

I was born in Canada in 1956. I have lived in Jerusalem Israel since 1983. I am married with 4 children, 2B + 2G, plus many grandchildren. I take physical fitness seriously. Once upon a time I was a judoka, and a classical guitarist. I was an IDF reserve soldier for 15 years, where I served as a combat soldier in the infantry in the Jordan Valley. In spite of the fact that I joined the army when I was 32 years old (Hebrew: *Shlav Betnik*), functionally, but unofficially, I served in the capacity of deputy company commander (Hebrew: *Samech Mem Pe*) which provided me with the opportunity to achieve rich personal growth, and enabled me to learn important managerial and leadership skills.

## Colophon

1. **Generator:** This document was generated using the [Pandoc](http://www.pandoc.org/) universal document converter extended [Markdown](https://en.wikipedia.org/wiki/Markdown) engine, along with the [Jinja2](http://jinja.pocoo.org/) macro/template preprocessor. See the source code at my [github site](https://github.com/avraham-bernstein/avraham-bernstein.github.io/tree/master/cv).
2. **Safety & non-annoyment pledge:** This document is free of scripts, frames, ads, and animations.

1. This secure photo was built using an [oilify](http://docs.gimp.org/en/plug-in-oilify.html) filter and a visible email address [watermark](https://en.wikipedia.org/wiki/Digital_watermarking). It uses [digital camouflage](https://io9.gizmodo.com/5510040/designer-reverse-engineers-face-detection-tech-to-develop-camouflage-makeup) that will defeat almost all face matching algorithms - in spite of the fact that my 3 year old grandson has no problem identifying me from this photo. It protects my privacy in two ways. (1) Harvesting it from the Internet, and adding it to a database of photos that will be compared with CCTV surveillance photos will not work. (2) The photo has built-in [two factor authentication (2FA)](https://en.wikipedia.org/wiki/Multi-factor_authentication). The prominent email address watermark prevents a web site from maliciously displaying my photo while attempting to associate/label it with any other name besides mine. Similarly if a site were to maliciously display my photo without my permission in order to imply my association or agreement with them, then it is trivial for anyone who may be suspicious of their claim to “challenge” the site by asking me for a verification email. [↑](#footnote-ref-2)
2. [Anti-harvesting](https://en.wikipedia.org/wiki/Email_address_harvesting) privacy protection. [↑](#footnote-ref-3)