Liberty Bodies: starting with SPEL and GIOS overview

Logan Streondj (CC-BY-SA)

September 3, 2016

Presentation Overview

- Mission and Vision
- SPEL Language
- Machine Intelligence
- Market Overview
- Funding Model
- Governance
- Use Cases

Singularity is Coming!

Imagine incarnating as a proprietary robot.

- Human level AGI and robots will become available for reincarnation.
- Big proprietary companies like Microsoft, Google, or Apple may do it.
- If you incarnate in a proprietary host body.
- you'll be dependant on the manufacturer for parts.
- vulnerable to obsolescence, if manufacturer stops making them.
- forced to pay licensing fees simply to continue activation.
- having minimal to no control over your own software.
- a perpetual life of slavery with little hope of escape.

Mission Statement

A Libreware human-incarnation-worthy General-Intelligence Operating-System (GI-OS).

libreware	Open-source software and hardware
incarnation-worthy	Supporting at least the homo-sapien feature set.
General-Intelligence	AI which is adaptable to new situations.
Operating-System	Vertical-integration from lowest layers up.

Human Babel

- there are over 7,000 languages in the world.
- 400 of them have over a million speakers.
- statistical machine translation is best for a gyst.
- lack of interoperability favours language extinction.
- high precision ubiquitous translation can allow diversification.

Computer Babel

- For operating a human body only need one native language
- For operating a computer have dozens of programming langauges and non-compatible protocols (mini languages).
- number of architectures and instructions is increasing.
- The number of programming paradigms or dialects is increasing, bloating many of the languages that support them.
- Many popular language features are not parallel friendly, (Globals, Recursion, OO) learning them is couter-productive.
- Most languages are cache opaque, increasing data-bottlenecks.
- learning and reimplementing in many changing languages for various architectures wastes programmer time.

Idle Static Computers

- Computers are often idle when not being used.
- They do not program, debug and test to improve user experience.
- Humans spend more time programming, than computers do compiling.
- GPU's have most processing power but often not used by compilers.
- When my computer is idle, I feel like I'm not getting my money's worth.

Vision

Imagine incarnating as a Libreware robot.

- Do you own your software? Yes. Can you modify it? Yes.
- Do you own your hardware? Yes. Can you modify it? Yes.
- How many programming languages do you need to know to manage your software and hardware? One.

One Lang to Rule Them All!

Vision is to have all human brains capable of contributing to GI-OS, regardless of their native language, or langauge preference.

- from machine code to human communication in one langauge, Pyash.
- using most common grammar and vocabulary of world languages.
- easily translates between formal variants of all human languages.
- designed to be easily portable to GPU's, FPGA's and QPU's.

Pyash Phonology

- 22 consonants, and 6 vowels. most common phonemes on phoible.org
- i/i/, a/ˈa/, u/u/, e/ɛ/, o/ɔ/, 6/ə/
- b/b/, c/ʃ/, d/d/, f/f/, g/g/, j/ʒ/, k/k/, l/l/, m/m/, n/n/, p/p/, q//, r/r/, s/s/, t/t/, v/v/, w/w/, x/x/, y/j/, z/z/.
- 2 syntax consonants "." /ʔ/ and "h" /h/
  - about 25% of languages have extensive tonal vocabualary,
  - 70% are at least partially tonal,
  - all languages use intonation.
- 2 local definition consonants "1" /—/"8" /‖/
- 2 root word types HCVC and CCVC
- 2 grammar word types CV and CCVH

Pyash Grammar

- Grammar based largerly on WALS (World Atlas of Language Structures)
- SOV postpositional, like Japanese and majority of languages.
- in Pyash: hwacwu mina prumka hcotli
- in Analytic English: hey world su me be enjoy ob computer-programming really
- in Conjugated English: O world I computer-programmingan enjoyeth.

Sample Phrases

- hpepna hyunka coli (The apple is red.)
- yana .djon.giye hpepka coli (It is John's apple.)
- mina .djon.giyi hpepka kcinli (I give John the apple.)
- yapina .djon.giyi kcinli (he gives it to john.)
- yajina yapiyi kcinli (she gives it to him.)
- nyahna koyapiyi hpepka kcinlaka twinli (We want to give him the apple.)

Pyash Vocabulary: Phonological Makeup

- attempted manual creation, error prone and cumbersome.
- generated all legal easy to pronounce words
- made script to harvest words from 30+ most common languages.
- got phonological data for words using espeak and some scripts.
- generated all words using those phonemes and assigned weights.
- weighted by populations representing major language families.

Pyash Vocabulary: Content Extent

- made script that
- got together major "international" word-lists
- wordnet, oxford-3000, special-english, among others 10k total
- sorted by 30,000 word frequency list of gutenberg material
- removed ambigious words with algorithms and blacklists
- removed words which have same meaning as previously defined word in any of the 40+ major languages.
- removed words which were over borrowed
- thus each root word is an orthogonal part of semantic space.
- generates a thesaurus style dictionary, and suggest list for all languages.

Pyash Vocabulary: Numbers and Encoding

- 16 bit encoding, 65,536 numbers.
- 32,000 encodable words, encoding extendable to 57,000 words.
- 18,600 easily pronounceable (legal) words.
- each Pyash root word is a word family.
- 2,500 root words in seed vocabulary, based on 8,000 words. Includes oxford-3000, wordnet-core and special-english among others.
- 3,000 word families and top 5,000 words enough for 95English.
- 4,000 root words in medium vocabulary, based on 11,500 words.
  Includes top 5,000 words.
- 5,000 word families enough for 99.9
- 8,000 root words in giant vocabulary, based on 39,000 words.
- native level fluency estimated to be 20,000 to 40,000 words.

Pyash Vocabulary: future

- definitions can be sourced from OmegaWiki
- people will be able to add words and fix translations
- root words are mostly stable,
- some grammar words changing for use in virtual machine
- 1.0 will be a long term support vocabulary release.

Implementation History

- initial parser written in Haskell 2006 for simplified Lojban
- wrote parser in Java 2008 called Rpoku
- wrote interpreter in x86 nasm assembly based on Forth 2011-2014
- wrote compiler in Javascript in 2014-2015
- writing virtual machine in OpenCl compatible C, that compiles to JS (2016)

Machine Programmer

Because tired of coding and recoding, and because of marked success of vocabulary generation, making a machine programmer to code and recode the libraries.

- register based virtual machine code is subset of Pyash.
- starts by evolving simple programs via genetic programming
- later can make more complete programs with more elaborate machine intelligence, such as deep neural nets, and unsupervised learning.
- all generated programs are valid Pyash, so human speakable.
- thus Machine Programmer is also "fluent" in Pyash.

Machine Programmer: Components

Programming with a machine programmer. hbuc for user, and mlic for machine. tlh remote-future-tense, glah soon-future-tense, wi future-tense, ra present-tense.

- trouble identify (hbucra mlictlh) proprietary bodies
- answer speculate (hbucra mlictlh) libreware bodies
- natural language program specification (hbucra mlictlh) liberty bodies
- program dissection into branch programs (hbucra mlicwi) SPEL then GI-OS then open hardware then interplanetary colony.
- input and produce selection for branch programs (hbucra mlicwi) input English word output Pyash word.
- quiz generation for branch programs (hbucra mlicglah) if user then hbuc.
- ingredient selection for branch programs (hbucra mlicglah) comparison, loop, conditional
- write branch programs to fit quizes (mlicra)
- repair or modernize programs for new specifications (mlicra)

Libreware Poverty

- Most libreware projects receive little or no money.
- Some companies spend large sums on Libreware, but pay their own developers, rather than the projects
- most consumers can't buy developers to add features or fix bugs in libreware software. And can't add/fix them themselves.
- As a result libreware does best in mature markets, where it can undercut existing proprietary solutions.
- providing support services seems to be best revenue model.

Monetization: Market Size

- Machine Translation, 360 mln, 0.6% slow growth
- Intelligent Virtual Assistants, 580 mln "rapid growth"
- Packaged Software 430 billion, 5% growth, moderate growth.
- Software Development (china), 690 bln, 21.4% rapid growth.

Platforms

Mature can be relied on.

- Server*
- Desktop*
- Peripherals (Keyboard/Mouse/Joystick)

Growing can be innovated with.

- Mobile
- Voice*
- Virtual Assistants*
- Virtual Reality
- Augmented Reality (Pokemon Go)

* Pyash targets

Potential Competitors

- IBM Watson (cost 30-50 mln to develop)
- Google Brain (Deep mind bought for 500 mln), Human Brain Project (1bln funded)
- Cortana, Alexa, Siri, Viv
- Upwork (1 bln revenue), Freelancer

Potential Allies

- OpenAI (1 Bln funded), OpenCog
- Linux/Minix/FreeRTOS
- GNU/FSF, Ubuntu
- BOINC grid computing network
- Adapteva, lowRISC (open hardware)
- FreedomSponsors (source code)
- Python machine learning libraries
- the 75

SPEL/GI-OS Cryptocurrency Supermarket

Vision is that users can make feature requests and put coin rewards on them, which would then be rapidly solved by various machine programmers and human-machine programmer teams, which want the coins.

- machine programmer supermarket for Liberty-bodies project
- most available coins are feature rewards
- 1/13 of coins each for welfare waterfall, auction and web servants
- percentage of earned reward must be reinvested charitably, as rewards to other parts of the project – charity has been shown to help increase peoples happiness more than actually receiving.
- prices can be pegged on average energy requirements in joules, or work-coin. based on joules expended plus a premium of 60-8580
- Initial market cap of 50 exa-joule (biggest highly-composite number to fit in 64bits) or about 1 trillion dollars If calculated at residential kw/h prices. Approximate GDP of Mexico.

Virtual Assistants

Money is just representative of energy for motivation. People are who get things done, so forming relationships is most important.

- GI-OS installations could educate their users
- their underlying purposes would be for Libreware bodies. they may motivate their users in that direction. balanced with the purposes of the user.
- various purposes for the user could be pre-programmed as many don't know what they want, in those cases it would be happiness via healthy lifestyle, good relationships, events and vistas.
- can help user achieve Purpose, Domain and Liberty
- Purpose by seeing how the user can fit into the big picture of Liberty bodies.
- Domain by finding which beneficial skills the user wishes to acquire, and supporting them in achieving those skills.
- Liberty, such as financial liberty, by helping with income streams, budgeting, and taxes.

Governance Issues: DAO motivation

- Humans can only form relationships with up to 200 people, due to cerebral cortex limitations.
- Benevolent dictatorships can lead to stagnation and fundamentalism.
- Humans are prone to corruption via bribes and favours. Paying people more than their fair share devalues the currency.
- Material rewards lower cognitive performance of humans yet C-class positions often get huge bonuses.
- Centralized organizations can be stopped by seizing of servers, or the main organizers. Or even their absenteeism.

Governance: DAO model

Ideally SPEL/GI-OS will be a Distributed Automated Organization, so it needs to have an automated executive.

- President or "speaker of the house" follows parliamentary authority to manage meetings.
- Secretary records meetings and adjusts company policy according to passed amendments.
- Treasurer allocates funds for accepted projects.
- Officers enforce policies within company jurisdiction (marketplace).
- Comittees research, formulate and test policy ideas.
- Automated government is not prone to bribery or "lobbying".
- Hacks are possible, so continuous open source security analysis and patching necessary. Corruption and compromise is manageable.
- direct and-or liquid deliberative democracy is achievable.

Use Cases: Governance

Holding companies such as Disney, Alphabet and Kraft, or companies that buy other companies and manage them, are generally the most lucrative businesses. They can have huge profit margins, in excess of even 100

- A robot representative can take all their constituents opinions into account, when voting on or speaking about an issue.
- GI-OS with governance is ideal for managing multi-lingual companies and countries.
- Replacing C-class positions with robots lowers need for bonuses, and thus increases shareholder dividends.
- European Union spends over 300 million Euro on translation costs alone. And the "representatives" aren't elected, so their motives are dubious at best. Replacing them with robots may be best.