Pyash: One Language to Unite Them All*

Htaf (Logan) Dwes
LiberIT Liberty Information Technology Services
logan@liberit.ca

Abstract

By using the fundamentals of human language, we may be able to achieve complete vertical integration for software languages, allowing one language to do everything from low level programming to chatting with humans.

Most software languages can't be used for making content, documentation or having a discussion. Because the vocabulary and grammar of most software languages is so limited, it is impossible to gain conversational fluency, thus they never rise to the status of human language, but sit as merely a code.

After years of research, analysis, data-mining and prototypes, a language has been made that not only allows for human-computer discourse and programming, but surprisingly can also be usable as a highly formal pivot language between the majority of human languages. As of this writing, beta-testing seems to be just a short time away.

In conclusion, a single language to unite all languages is viable at least to the degree that it has already been implemented. It can translate between controlled variants of most, possibly all human languages. Though further work is needed to prove that those controlled variants can be easily learned by natural language speakers. It can't translate between different software languages, but may be able to vertically integrate all the purposes of software and human languages into one language. Further work is needed in order to prove that to a higher degree of certainty.

CCS Concepts **•Software and its engineering →General programming languages; •Human-centered computing** **→*Natural language interfaces;***

Keywords grammar, programming language, pivot language

*A vertically integrated software language based on the common features of the majority of human languages.

SLE2017, Vancouver, British Columbia, Canada

DOI: 10.1145/nnnnnnn.nnnnnnn

ACM Reference format:
Htaf (Logan) Dwes. 2017. Pyash: One Language to Unite Them All. In *Proceedings of SLE Vancouver, Vancouver, British Columbia, Canada, October 2017 (SLE2017),* 7 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 Introduction

The purpose of software languages is to help humans communicate with machines. To achieve this, the language has to be regular and sufficiently well defined that both parties understand what the other is saying. Though contemporarily the computer is programmed in one language, and error messages have a different protocol (mini-language).

Homo-sapien language has been evolving since at least Mitochondrial Eve, she lived possibly one or two hundred thousand years ago. There are already over 6,000 human languages, so it is acceptable to add another one, which happens to also be a general purpose software language.

The goal of the software language, is complete vertical integration. So that if one is to reincarnate into a robot, then everything from the lowest to highest levels can be accomplished using the same language. Similar to how English can be used to communicate everything from the lowest to highest levels.

Several versions of implementing this idea have been made over the last decade. This paper uses Pyash as the word for the pivot language (Intermediary Representation), it means language and is the result of data mining world language vocabularies (3.2).

Natural English is not formal enough to be directly used as a software language, however Pyash English is. Pyash also acts as a bridge, for high precision translation, so Pyash English documents could be rapidly and precisely translated to Pyash Hindi, Pyash Spanish, Pyash Swahili, or any other supported human language.

This high precision translation could open the door to software languages for the majority of humanity which is not fluent in English.

2 Literature Review

This section will review some of the common references and mention how Pyash is different from them.

The main difference is that Pyash is based on the fundamentals of human language, and has a complete and orthogonal vocabulary.

2.1 COBOL

COBOL, originally intended to be a business programming language, was designed by several committees, some of the committee members were unfamiliar with computer programming and-or linguistics, the committees also had issues with discontinuity of personnel. This led to a language that was neither very good for computer programming, nor very easy for humans to understand, while also having issues with repeals due to changing personnel.

By contrast, Pyash's design takes into consideration many academic and real world sources for its grammar (3.1), vocabulary (3.2) and instruction set architecture (3.4). It's evolution has also run through several different iterations though all with the same informed personnel to keep it on track.

2.2 Hypertalk

Hypertalk uses English keywords to replace common programming syntax symbols, so is largely just a relexification of standard (ALGOL inspired) programming.

Pyash on the other hand starts with a human grammar base and then adapts it to usage as a software language.

2.3 Lojban

Lojban is a language intended for human use, but based on the structure of programming languages, in particular predicate logic[14]. Because of this Lojban is more of an API rather than a human language, making it very difficult to gain fluency[9][20].

While there have been some cursory motions towards making Lojban a programming language[17], none have gotten much past the concept stage.

The net result is that Lojban has not proven suitable for human communication, nor as a software language. Though it has been a useful stepping stone and point of inspiration.

3 Method

Many different approaches have been taken to the creation of software languages. Rather than basing Pyash on the Chomsky Hierarchy of formal languages and formal grammars, it is based it on human grammar.

3.1 Grammar

Linguistic Universals[4] are patterns found systematically across large groups of languages, possibly all languages. In particular all languages have verb phrases and noun phrases, and mark their phrases either with placement, adpositions or affixes. All can also express tense, mood and aspect.

However there is the issue of making the pivot language. Which of the many options should the language use? To the rescue comes the World Atlas of Language Structures[7] (WALS), which allows one to see what are the most common features around the world.

In particular Pyash is Verb-final, or Subject-Object-Verb word-order, similar to Hindi, Japanese and Amharic. Linguistic Universals point toward suffixes and-or postpositions for verb-final languages, so they are used.

But what of the grammar words themselves? A variety of contenders were reviewed, such as Universal Networking Language[8] from the United Nations University, and FrameNet[10] from Berkley. A more organic solution was chosen consisting of the list of Glossing Abbreviations[12] used by linguists when transcribing foreign languages.

3.2 Vocabulary

Contemporary Software Languages generally lack a root vocabulary. Keywords may have a special meaning, but they are typically of a syntactic or grammatical nature, so are at most a grammatical vocabulary. API's naming convention of being series of unreserved letters, means that all unreserved words are proper nouns.

Pyash has a root vocabulary so that documentation, description and discussion can all happen in the same language as computer programming. The encoding requires API names to be words with a proper morphology (3.3), and may be restricted to only being official dictionary defined ones, ensuring standardization and ease of translation.

To generate the vocabulary first several word-lists were put together, including WordNet core[3], Oxford-3000[13], UNL-core[8], Special English[5], FrameNet[10], New Academic Word List (NAWL)[15], New General Service List (NGSL)[16] and Project Gutenberg Frequency List[2]. After collating them all and taking out the duplicates, the language was left with almost 39 thousand words.

Google Cloud Translation API[11] was used to translate each word on the list individually into the top 48 languages by number of native speakers. Giving an overall coverage of greater than 70% of the world population.

A script to sort the vocabulary based on the frequency list[2] was made and it filtered them for uniqueness. Words were removed that were:

Overborrowed If more than 38%[1] languages use the English term.

Ambigious If it means multiple things in more than 38% of the languages.

Homographs If it is a homograph of an already defined word in any of the languages.

This left the language with a fairly orthogonal pool of about eight thousand words.

3.3 Morphology

The pivot language needs to be sufficiently easily spoken by humans for it to be usable by humans in conversation. This was particularly the case in early prototypes, as it wasn't

[1] $2 - \phi = 38\%$ where ϕ is golden ratio or 1.618. A golden fraction was felt to be a natural choice.

ASCII	IPA	Description	English
a	ä	central open vowel	arm
b	b	voiced bilabial plosive	ball
c	ʃ	unvoiced post-alveolar fricative	shout
d	d	voiced alveolar dental	door
e	ẹ	mid front unrounded vowel	enter
f	f	unvoiced labio dental fricative	fire
g	g	voiced velar plosive	great
h	ʰ	aspiration	happy
i	i	unrounded closed front vowel	ski
j	ʒ	voiced post-alveolar fricative	garage
k	k	unvoiced velar plosive	keep
l	l	lateral approximants	love
m	m	bilabial nasal	map
n	n	alveolar nasal	nap
o	ọ	mid back rounded vowel	robot
p	p	unvoiced bilabial plosive	pan
q	ŋ	velar nasal	English
r	r	alveolar trill	(Scottish) curd
s	s	unvoiced alveolar fricative	snake
t	t	unvoiced alveolar plosive	time
u	u	rounded closed back vowel	blue
v	v	voiced labio dental fricative	voice
w	w	labio velar approximant	water
x	x	velar fricative	(Scottish) loch
y	j	palatal approximant	you
z	z	voiced alveolar fricative	zoom
.	ʔ	glottal stop	uh-oh
6	ə	mid central vowel	uh
7	˥	high tone	wha?
_	˩	low tone	no!
1	|	dental click	tsktsk
8	‖	lateral click	winking click

Table 1. ASCII alphabet used by Pyash, the letter's IPA equivalents, description and English pronunciation key.

Code	ASCII Example	IPA	Name
CV	ka	/kä/	short grammar word
CSVH	kyah	/kjäʰ/	long grammar word
HCVC	hkap	/ʰkäp/	short root word
CSVC	kyap	/kjäp/	long root word

H /ʰ/ aspiration or spectrographically an unvoiced vowel.
C a consonant.
S a consonant of higher sonority than the preceding one.
V a vowel (highest sonority).

Table 2. Pyash word morphology.

realized that the pivot language could be used for translating between possibly all human languages — which would negate the need for actually learning the pivot language, a Pyash controlled natural language would be sufficient.

First, an alphabet representing phonemes which are popular in human languages was required, for this PHOIBLE[19] was used. Then WALS'[7] chapters on phoneme inventories was used to find what a common ratio of consonants to vowels is, as well as common number of consonants and vowels, and picked the most popular single phonemes which are reasonably distinct. Two tones were also included to increase the number of words. Two clicks were included for temporary document specific words — in place of acronyms. An ASCII letter for each IPA phoneme was also selected (Table 1) to make sure Pyash is web compatible.

Second, a morphology of how the phonemes are put together to make words was required. For this phonotactics of the sonority scale[1] was used, paired with the WALS[7] chapter on syllable structure.

The language was also made easily parsed even if there are no spaces or pauses between words. Each word is either two or four letters long. The two letter words start with a consonant and end with a vowel, and the four letter ones start with two consonants and end with a consonant (Table 2).

The valid words were generated with several alphabets, and a script was made to assign words based on the phonemes in the source languages weighed by their representative native speaking populations. The highest frequency words were assigned to the easier to pronounce and understand smaller alphabets. And the more rare words were assigned to the more difficult extended alphabets — with voice contrast and-or tones for instance.

3.4 Instruction Set Architecture

For complete vertical integration the language has to boil down to machine level instruction, or an instruction set architecture. The JVM bytecode is an example of a different language which can also be implemented as an instruction set architecture[22].

Understanding that the future of computing is going towards parallelism much research into how to make the language as parallel-friendly as possible was done. In particular the Heads and Tails ISA[21] was found to be quite inspiring.

Each Pyash word fits in sixteen bits (a uint16_t). There are four word types and one quote type which are encoded. The quote type allows for including literals.

For parallelism sentences are encoded into codelets[6], which are comprised of one or more vectors of sixteen, sixteen bit values. The first sixteen bit value of a vector is the index for the vector, marking the location of grammatical cases and moods (ends of noun and verb phrases).

This encoding can then be translated to any supported human language (Table 4). In terms of compiling to a programming language, it compiles to OpenCL C. There is also

1. **Pyash English** do say the quoted'word'hey world'word'quoted.
2. **Pyash** zi.wo.hwacwu.wo.zika hsactu
3. **Codelet** 0051 291D E928 28BE 245E E948 295E 0000 0000 0000 0000 0000 0000 0000 0000 0000
4. **Codelet Explained** (0051 index) (291D quoting two words) (E928 28BE hwac wu) (245E ka accusative-case) (E948 hsac say) (295E tu deonitic-mood) 0000 0000 0000 0000 0000 0000 0000 0000 0000
5. **C**

```
wotyutdokahsac(_("hwacwu"));
```

6. **Output with en_US locale** hey world
   Output with ru locale эй мир

Table 3. A codelet encoding example. Note: Controlled natural language input and output was implemented in the Javascript version[28], and hasn't yet been fully ported to C.

Pyash	mina ryopyi syutka kwinli
Gloss	me NOM robot DAT liberty ACC giving REAL
Pyash English	I be giving the liberty to robot.

Table 4. Example of formal translation

a design[26] for making a code-parallel virtual machine, that can process linear code on GPU's using Pyash ISA.

The encoding could also be used for storage of information, similar to a database, as well as for knowledge management, similar to how human languages are used for storing information.

3.5 Parser or Encoder

The parser is probably of some interest due to its refined simplicity. It is a hand coded, single pass type, modeled on how a human would parse text. There are no parse trees or any such complexities.

First the parser checks if a word is a valid Pyash word, if so, then checks if it is a grammatical-case word, a grammatical-mood word or a quote word, if not then simply adds it to the codelet.

If it is a quote word then acts accordingly either upon the literals ahead or the words behind, adding what is necessary to the codelet, and adjusting the codelet and text index pointer to just after the quote.

If it is a grammatical-case word, then in addition to adding the word to the codelet, also marks it on the index.

If it is a grammatical-mood word then does as with the grammatical-case word but also ends the codelet. With the exception of the conditional mood, which is treated the same as a grammatical-case for encoding.

For reading and writing to the codelet there is a function, which manages which vector is being added to. If the addition over-runs one vector, then it's index is inverted, and the next vector receives the additions. This way when reading indexes, it is known if it is the end of the codelet based on the first bit of the index — if it is a one then it is the final vector.

This simple parser/encoder could parse/encode sentences in parallel, and should be adaptable for parsing spoken streams of phonemes. A more complicated version of the parser/encoder will be necessary once support is added for subordinate clauses, since they would have to be broken up into multiple codelets for the encoding.

4 Discussion

Various variations of the language have been worked on since 2007. The first implementation[23] was in Haskell and second was in Java[24], both were recursive parsers.

The third implementation[25] followed the Jones Forth[18] model, hoping to bootstrap something small and scaleable, so Intel assembly was used for a few years and succeeded in making a basic interpreter.

The fourth attempt[28] was in nodejs Javascript, since by that time it was realized that the language could be used for translation, and something portable was desired which could written quickly — the antithesis of assembly. A translator was made and a basic compiler to Javascript, but was severely limited by a hand picked vocabulary, so an automated vocabulary (3.2) was made. While it was being made, it was realized that the object oriented implementation in Javascript was difficult or impossible to make parallel, combined with its plain text encoding led to it running extremely slow. So the Javascript translator and compiler was abandoned, but the automated vocabulary was kept.

The fifth and current attempt[27] it was motivated by the realization that something fast, scaleable and future-friendly was needed, so a parallelizeable ISA (3.4) was designed and the implementation was done in OpenCL C. As of this writing (May 2017) it compiles hello world, does variable assignment, for loops, and function declarations are being implemented.

4.1 Vertical Integration

While the main focus of the current implementations has been computer programming languages and related documentation. The language can be used to cover the areas of other software language types as well.

4.1.1 Database Languages

For example SQL database access and creation language, can easily fit as a subset of Pyash, with some slight vocabulary changes (Table 5). Due to this rather fortunate grammatical-case design of SQL it should be possible to translate from

SQL	Pyash	Pyash English
CREATE	tlip	establish
SELECT	kcot	gather
UPDATE	draf	modernize
DELETE	dlas	delete
INSERT	hquk	inject
FROM	pwih	from
WHERE	te	at
INTO	twih	into

Table 5. Sampling of SQL keywords and their Pyash equivalents

SQL to Pyash and vice-versa — whereas with most placement based parameter family of languages it is a non-trivial process.

4.1.2 Ontology Languages

For knowledge representation or ontology languages, the databases could simply be made of Pyash codelets. They could be rapidly queried in parallel on GPU for any particular piece of information. They could be translated to and from human language, for sharing gathered knowledge with humans, or acquiring knowledge from humans.

Even a few people having a conversation, such as at a meeting could generate programs and-or machine knowledge if they were speaking with enough formality to be Pyash accessible.

Pyash accessibility is currently rather low, having a rather strict grammar. But with machine learning algorithms to help with converting natural language speech into Pyash controlled natural languages the amount of machine accessible knowledge that could be harvest from the spoken and written word should dramatically increase.

4.1.3 Modeling Languages

Considering that Gellish is a modeling language, and that Pyash has a much more developed grammar, it should be fairly straightforward to adapt Pyash to be a universal modeling language.

For visual people, graphics could be generated from Pyash descriptions. So in the hypothetical scenario of some people talking in a meeting, the computer could be projecting the model of what is described on the screen. Or running and showing simulations to see the potential outcomes of various policy or program changes.

4.1.4 Domain Specific Programming Languages

The majority of domain specific languages seem to have placement based parameters. This means that reading the API is likely necessary to understanding how to use any functions. Thus, unless the API is written in Pyash or some other machine-accessible format, translating to and particularly from those languages to Pyash is non-trivial.

Translating to those languages is easier, as a human can read the API and make an appropriate Pyash side function to access it. However if someone adds a new function to that other language, without following something like the Pyash function naming convention, then it will be nearly impossibly to translate to Pyash without reading it's corresponding API and-or analyzing it's code.

Possibly when machine learning and AI gets sufficiently sophisticated it will be able to do those translations, but that is quite possibly decades away.

For now it makes sense to limit official Pyash programming development to compiling to popular C libraries, and also making native libraries.

4.1.5 Communications Protocols and Serialization formats

There are a wide range of communication protocols, all serving their own niches. For example, HTTP, SMTP, and IRC.

With the advent of XML there was an increase of protocol creation, for example XMPP, SOAP and XML-RPC. However since XML doesn't have a root vocabulary most of these different protocols have different naming conventions and so are not easily inter-operable.

XML is also rather bulky, so in certain areas, such as configuration and data storage, more compact alternative such as JSON, Lua and YAML have gained. Though like XML, they lack a root vocabulary.

Pyash does have a root vocabulary so it is fairly straightforward to use as a communication protocol. Having the root vocabulary could encourage people to extend the language rather than make entirely new protocols. The binary encoding of Pyash, which can store various types including binary data, is both compact and can be decoded into a human readable format in a variety of human languages.

In terms of usage of space, Pyash is likely to be more bulky than any of the early terse ones like HTTP, but will typically use less space than XML, approaching JSON or YAML — depending on the length of names used.

The goal of using Pyash for protocols is making it easier to collect, consume and process large amounts of data. Especially now that many of us have more storage and processing power than we know what to do with. For example, may people have powerful GPU's in their computers, which most of the time sit relatively idle.

Since the error reporting was mentioned earlier, here is an example (Table 6). Though the Pyash versions are longer, they are more portable, and non-English speaking people can help debug the program, as the Pyash could be translated to an approximation of their native language.

Additionally a variety of protocols could be translated into Pyash, not necessarily so they would be faster, but to make

Error message	encoding:570:text_encoding debug text
Pyash English	from encoding file at num five seven zero line in text encoding cereomony the debug text be emitting.
Pyash	kfinhfaspwih hfakhsipzrondo lyinlwoh htetkfinsricnwih dyekhtetka mwa7nli

Table 6. Demonstrating how error messages might be conveyed more meaningfully using Pyash.

it easier for an AI or AGI to understand and communicate using them.

4.1.6 Markup Languages

LaTeX, HTML and Markdown are some of the most popular markup languages on the internet today. Of course they are mostly for formatting, and do not include a vocabulary for the content.

However for writing modern documents, it is often important to have chapters, sections and subsections. Spoken speech has an (arguable) analog of bold and italics, via the focus and topic of the sentence — which is already a part of Pyash grammar. However spoken language generally doesn't have long enough monologues for people to even mark their spoken paragraphs.

The grammar of Pyash could be extended enough to allow for such mark up. An example would be to make a grammar word for paragraph, module (section), and frame (chapter).

Pyash as a markup language would be particularly useful in using Pyash for writing international content, such as stories, news articles or even legislation.

5 Conclusion and Further Work

A software language based on the fundamentals of human language that is usable for human communication and computer programming is certainly viable and implementable, as it has been done.

Translating all or most human languages, or at least controlled variants of them does on the surface appear viable. Though further research would have to be done to see what level of conjugation is comfortable for, and how long it would take for native language speakers to adapt to the controlled variants.

Translating everything between software languages is unfortunately not viable due to the much smaller scope of them, as they can't be used for human communication. Though existing codebase can be used via foreign function interface.

Complete vertical integration of everything that a computer might need to do seems to be viable, though further work would need to happen to prove it.

This implementation of the language seems to be satisfactory. Language adoption is a major hurdle, which motivates this article. Pyash is being used to write an automated programmer to more quickly write the standard libraries, and general intelligence operating system to follow.

References

[1] 2004. What is the sonority scale? (2004). http://www-01.sil.org/linguistics/glossaryoflinguisticterms/WhatIsTheSonorityScale.htm

[2] 2006. Frequency List, Project Gutenberg. (2006). https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists#Project_Gutenberg

[3] 2006. WordNet a Lexical Database for English. (2006). https://wordnet.princeton.edu/wordnet/download/standoff/

[4] 2009. The Universals Archive. (2009). https://typo.uni-konstanz.de/archive/intro/

[5] 2010. VOA Special English Word Book. (2010). http://www.manythings.org/voa/words.htm

[6] 2013. The Codelet Execution Model: Fine-Grain Multithreading for Extreme-Scale Computing. (2013). http://www.capsl.udel.edu/codelets.shtml

[7] 2013. The World Atlas of Language Structures Online. (2013). http://wals.info/

[8] 2014. Introduction to UNL. (2014). http://www.unlweb.net/wiki/Introduction_to_UNL

[9] 2014. Lojban's Biggest Problem or Why Still Nobody Speaks It. (2014). https://groups.google.com/d/msg/lojban/e7cg5vy2EfA/aJaC09ERWasJ

[10] 2016. FrameNet Data. (2016). https://framenet.icsi.berkeley.edu/fndrupal/frameIndex

[11] 2016. Google Cloud Translation API. (2016). https://cloud.google.com/translate/

[12] 2016. List of glossing abbreviations. (2016). https://en.wikipedia.org/wiki/List_of_glossing_abbreviations

[13] 2016. The Oxford 3000. (2016). https://www.oxfordlearnersdictionaries.com/wordlist/english/oxford3000/

[14] 2017. Lojban Introductory Brochure. (2017). https://mw.lojban.org/papri/Lojban_Introductory_Brochure

[15] Culligan B. Browne, C. and J Phillips. 2013. A New Academic Word List. (2013). http://www.newacademicwordlist.org/

[16] Culligan B. Browne, C. and J Phillips. 2015. New General Service List. (2015). http://www.newgeneralservicelist.org/

[17] Ben Goertzel. 2013. Lojban++: An Interlingua for Communication between Humans and AGIs. (2013). https://link.springer.com/chapter/10.1007/978-3-642-39521-5_3

[18] Richard W.M. Jones. 2009. Jones Forth. (2009). https://github.com/AlexandreAbreu/jonesforth/blob/master/jonesforth.S

[19] Steven Moran, Daniel McCloy, and Richard Wright. 2014. PHOIBLE Online. (2014). http://phoible.org/parameters

[20] Arika Okrent. 2010. *In the land of invented languages : a celebration of linguistic creativity, madness, and genius*. Spiegel & Grau Trade Paperbacks, New York.

[21] Heidi Pan and Krste Asanović. 2001. Heads and Tails: A Variable-length Instruction Format Supporting Parallel Fetch and Decode. In *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '01)*. ACM, New York, NY, USA, 168–175. https://doi.org/10.1145/502217.502244

[22] Martin Schoeberl. 2007. JOP - Java Optimized Processor. (2007). http://www.jopdesign.com/

[23] Andrii (Logan) Zvorygin. 2008. Rpoku. (2008). https://sourceforge.net/projects/rpoku/files/rpoku/

[24] Andrii (Logan) Zvorygin. 2009. Rpoku Compiler in Java. (2009). https://sourceforge.net/projects/rpoku/files/Rpoku%20Compiler%20in%20Java/First%20Exhibit/

[25] Andrii (Logan) Zvorygin. 2014. Speakable Programming for Every Language (in Nasm). (2014). https://sourceforge.net/projects/spel/files/spel/

[26] Andrii (Logan) Zvorygin. 2017. Expanding domain of algorithms for GPGPU with code parallelism. (2017). https://gitlab.com/liberit/pyac/blob/master/documentation/vmOnOpenCL-sigconf.pdf

[27] Andrii (Logan) Zvorygin. 2017. humanity grammar international program language hjat hgaf nrot hrom pyac. (2017). https://gitlab.com/liberit/pyac

[28] Andrii (Logan) Zvorygin. 2017. Speakable Programming for Every Language (in Javascript). (2017). https://sourceforge.net/projects/spel/files/speljs/