SPAL: Speech Program All Language

Logan Streondj

February 20, 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This is a human speakable programming language, geared for artificial general intelligence development.

1.1 Problem

1.1.1 Disglossia

Computer Languages

In order to program all levels of a modern computer, you need to know many different programming languages. Assembly/LLVM/SPIRV at the lowest level, C for system programming, C++/GTK/QT for graphical interface programming, Javascript/HTML/CSS/PHP/Perl for web programming, OpenCL/OpenMP/Pthread for parallel programming, bash/python/ruby/node for scripting, Java/C#/Lua for portable programming, ansible/chef/docker/kubernetes for administration, a host of different data storage formats XML/CSV/JSON/YAML/SQL and different documentation languages LaTeX/Doxygen/Markdown/TexInfo just to name a few. That doesn't even include statistics, audio, image and video processing languages.

Human Languages

You may have heard the story of Babel, and the story of Eve and the apple. The evidence goes back much farther, to Mitochondrial Eve, in thesecond last glaciation about 130 thousand years ago.

Mitochondrial Eve is the most successful mother, the mother all homo-sapiens share.

Eden was the great rift valley of Africa.

Mitochondrial Eve lived during times of famine where most women were likely too starved to be fertile. Eve was a clan leader and got enough food to reproduce, she had many daughters that became clan leaders.

Human language may have been perfected by Mitochondrial Eve, she helped her clan work together more efficiently than all the others during the population bottleneck, and thus came out the winner.

To this day we inherit our language brain centers from the mothers side.

Eve's daughters spread out, some went west, some, south, some east, some north.

Those that went South and East preserved clicks, like the Khoisan people of the kalahari. Those that went west into the jungle became the pygmies. Those that went north eventually became the farmers, the Bantu and peoples of the rest of the continents.

The language of Mitochondrial Eve can be reconstructed based on the common features of the oldest languages in the world, as well as our genetic predispositions to prefer certain forms of grammar.

The most common grammar form, and the one we are predisposed to is subject-object-verb (SOV), or head-final with postpositions and-or suffixes. Similar to Khoe (of the khoisan), Basque (the first homo-sapiens in Europe), Australian languages, Turkic (Central Asian), Uralic (North Eurasian),

Tibetan/Burmese (East Asian) and Proto-Indo-European (That conquered the world).

1.2 Paradigm

1.2.1 Easy to write bad code

In most, perhaps all contemporary languages it is easy for beginners to write bad code.

In assembly it is easy to write tangled spaghetti code. In C and C++ it is easy to have memory problems (buffer overflow, memory leaks, reading unassigned variables, etc) In Garbage collection languages it is easy to spend significant computer resources on allocating and deallocating memory. In Object Oriented languages it is easy to write unscalable code (any which uses objects). In functional programming languages it is easy to write memory bound code (non-tail recursive with lots of allocation and deallocation).

It often takes a lot of expertise to know the workarounds for the common programming traps, and even harder to apply them consistently.

1.2.2 Obsolete Non-Parallel Paradigms

Object Oriented, non-tail recursion and referentially opaque code are all obsolete considering that GPUs and parallel hardware are where processing power is growing the fastest.

It's easy to write bad code in many paradigms.

1.3 Inspiration

I was mentally projecting myself into a robot host body one day and realized that it would take a superhuman Artificial intelligence to be proficient in all of the languages and protocols of a modern computer and their interactions.

And I came to the realization that I wanted to be able to have access to all my knowledge and abilities with one language.

1.4 Answer

The answer I came up with is the speakable programming language.

1.4.1 Vocabulary

The root vocabulary was generated by taking the most frequently used thirty-eight thousand English words, translating them into the top thirty to forty human languages, and then removing words that were ambiguous and-or homophones.

This left a remainder of about eight thousand words, which were common to all languages and orthogonal (not overlapping in meaning).

This way you can use the root words of your preferred language to program, and they will be translated to all the other languages.

1.4.2 Grammar

SPAL currently uses Eve's grammar: SOV with postpositions and-or affixes is the grammar of Pyash the base language of SPAL.

Transferring it to other forms of grammar is fairly straight forward. And has been done with a former iteration of this language. That does however lead to a large number of variants.

So for the near future, will simply have Eve's grammar, and your preferred vocabulary for root words.

Many contemporary languages lost the nominative-accusative case distinction, and have grammar words which are used ambigiously. For example, the word ``with'' in English is used for comitative-case and instrumental-case.

So for the actual grammar words, have decided to go with abbreviated forms of the translations of glossing abbreviations. For example _com for comitative-case and _ins for instrumental case.

Because of knowledge bias, we'll have to work together to create documentation that is easy for beginners to understand.

1.4.3 Paradigm

The paradigm conforms to the JPL ten commandments[1].

- Restrict all code to very simple control flow constructs - do not use goto statements, setjmp or longjmp constructs, and direct or indirect recursion.
- All loops must have a fixed upper-bound. It must be trivially possible for a checking tool to prove statically that a preset upper-bound on the number of iterations of a loop cannot be exceeded. If the loop-bound cannot be proven statically, the rule is considered violated.
- Do not use dynamic memory allocation after initialization.
- No function should be longer than what can be printed on a single sheet of paper in a standard reference format with one line per statement and one line per declaration. Typically, this means no more than about 60 lines of code per function.
- The assertion density of the code should average to a minimum of two assertions per function. Assertions are used to check for anomalous conditions that should never happen in real-life executions. Assertions must always be side-effect free and should be defined as Boolean tests. When an assertion fails, an explicit recovery action must be taken, e.g., by returning an error condition to the caller of the function that executes the failing assertion. Any assertion for which a static checking tool can prove that it can never fail or never hold violates this rule. (I.e., it is not possible to satisfy the rule by adding unhelpful `` assert (true) '' statements.)
- Variables must be declared at the smallest possible level of scope. All arrays must have a max-length variable, and bounds of all new index points must be checked before a read or write operation occurs. If an array is uninitialized, there must be an initialized-length variable also, so uninitialized data is not read accidentally.
- The return value of non-void functions must be checked by each calling function, and the validity of parameters must be checked inside each function.
- The use of the preprocessor must be limited to the inclusion of header files and simple macro definitions.
- The use of pointers should be restricted. Specifically, no more than one level of dereferencing is allowed. Pointer dereference operations may not be hidden in macro definitions or inside typedef declarations. Function pointers are not permitted.
- All code must be compiled, from the first day of development, with all compiler warnings enabled at the compiler's most pedantic setting. All code must compile with these setting without any warnings. All code must be checked daily with at least one, but preferably more than one, state-of-the-art static source code analyzer and should pass the analyses with zero warnings.

Additionally some OpenCL restrictions.

- no return values.
- input parameters are constants.
- output parameters are pointers.
- functions that don't interact with environment are referentially transparent.

Those should all be taken care of automatically, when compiling from Pyash to OpenCL C. So while other programming features may be available, it would take extra effort to enable the program to use them, and thus to write bad code.

Part I

Core Language

Chapter 2

Phonology

There are two scripts for the SPEL core-language, one based on URL-compatible ASCII and one based on IPA. If you are unsure of how to pronounce a letter, then simply copy paste the IPA letter into wikipedia which will give ample explanation.

Phonemes are based on the most popular distinctive ones on phoibles http://phoible.org/ parameters plus two clicks.

See table 2 for the ASCII, IPA and their description.

2.1 Notes

Alignment the `h' or /ʰ/ is a semi silent h /h/, and is used mostly for alignment purposes. All words when written in text are either 2 or 4 glyphs long. However some root and grammar words are three letters, thus they need alignment. For 3 letter roots of the form CVC (consonant vowel consonant) the h prefixes the word, turing it into hCVC, for 3 letter grammar words of the form CCV, the h is suffixes it, turning it into CCVh. A simple way to remeber this is that all words must comply with the CCVC or CV form. So if a three letter word is missing one of those C's then replace it with an `h' to get proper alignment.

Glottal stops glottal stop `.' is only used for foreign quotes, such as that of proper names, as they don't necessarily conform to alignment rules

Tones Tones `7' and `_', are mostly for low frequency words

Clicks Clicks `1' or `8' are used for temporary words and variables, especially useful to make short forms of compound words which are often used in a text or flock of people. Other options for short-forms are acronyms which must comply with the phonotactic rules of the language and be grammatically marked as acronyms, and initialisms, which are foreign quotes as they don't fit the phonotactic rules.

2.2 Contribution

Currently the phonology is pretty much finished, however if there are some compelling arguments then it may still be modified.

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	ɐ̟	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	o̟	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

Chapter 3

Grammar

3.1 Composition

those marked with asterisk are mandatory for phrase formation.

Phonology

Grammar Words are of the form CV or CCVH where C is consonant, H is /h/ and V is vowel.

Root Words are of the form HCVC or CCVC where C is consonant, H is /h/ and V is vowel.

sentence emotions evidentials mood*

noun phrase (root or pro-form)* suffix case*

verb phrase (root xor copula)* (aspect or tense)*

adjective phrase (root or pro-form)* suffix adjective-marker* case
case needs to be marked if adjective comes after the head noun, or could be confused as part of a different noun phrase.

adverbial phrase (root or pro-form)* suffix adverb-marker*

3.2 Grammar Tree

3.3 Noun Classes

abstract-gender for thoughts and ideas

animate-gender for more active things, animacy intensifier

anthropic-gender for anatomically human-like things

augmentative greater in size or intensity

collective-noun noun taken as a collection of things, turns mountain into moutain range, or trees into forest, or shrub into shrubland

common-gender male or female gender, hermaphrodites and ambigious gender, such as bigender, trigender, pangender,etc

diminutive less in size or intensity

dual-number two of the noun

inanimate-gender for less active things, animacy lowerer

feminine-gender female gender

masculine-gender male gender

- noun
  - pro-form
    * person-deixis
    * time-deixis
    * space-deixis
    * interior-deixis
    * surface-deixis
    * under-deixis
    * discourse-deixis
    * social-deixis
    * amount-deixis
    * state-deixis
    * interrogative
    * pro-phrase
    * pro-sentence
  - suffixes
    * proximity
      · proximal
      · medial
      · distal
    * number
      · singular
      · dual
      · trial
      · paucal
      · plural
      · multal
      · collective
      · distributive
      · inclusive
      · exclusive
    * classifier
      · name
      · number
      · length
      · mass
      · time
      · electric current
      · temperature
      · amount-of-substance
      · luminous-intensity
    * animacy
    * volition
    * gender
    * specificity
    * definiteness
    * quantifier
      · assertive
      · elective
      · universal
      · negatory
      · alternative
  - adjective marker
  - adverb marker
  - case
- verb
  - aspect
  - tense
- correlative conjunction
  - conjunction (and)
  - inclusive disjunction (and-or)
  - exclusive disjunction (xor)
- particle
  - sentence-final-particle
    * mood
  - sentence-semi-final particle
    * emotions
    * evidentials
- subordinator
  - genitive
  - relativizer
- interjection

Table 3.1: Grammar Tree

mineral-gender natural forces, rocks, bodies of water, etc

neuter-gender non-reproductive entities, including asexual

paucal-number small amount of something, 8 bit value $(0 - 255)$

number moderate amount of something, 16 bit value between 256 and 65 thousand.

plural-number large amount of something, 32bit value, between 65 thousand and 4 billion.

multal-number giant amount of something, 64bit value, between 4 billion and 18 quintillion.

rational-gender for entities that have self-selected goal-oriented communication, such as humans, spiritual entities, aliens and machine intelligence

singular-number one of the noun

trial-number a few or three of the noun

vegetal-gender for plants, including power plants, solar panels, turbines, simple and complex machines, including primarily remote controlled ones

zoic-gender for animals, including basic robots, with some intelligence, semi-autonomous vacuum cleaners, aerial-vehicles. Capable of basic decisions but primarily rellying on instinct.

artistic-gender for arts, like martial arts, visual arts, calligraphy, architecture, generally pertaining to activities which are physical manipulations in specific manners

research-gender for fields of study, such as mathematics, physics, biology, and other -ologies, generally pertaining to exploration and acquisition of knowledge.

vehicle-gender for vehicles, typically those that are primarily for the purpose of transportiation. Includes space-ships and small asteroids/comets.

locality-gender for places, including homes, areas, lakes, and countries. Includes large asteroids, and common locations of a corporation, for-example school.

planetary-gender For planemo, or planetary-mass objects which are rounded by their own gravity, and major locations of a corportion, for-example local school board.

star-gender For stars and central locations of a corporation, for-example provincial school board.

galaxy-gender For galaxies and major divisions of a corporation or virtual world, for-example department of education.

universe-gender For universes, corporations and virtual worlds, for example a government.

3.3.1 grammatical number

1. singular
2. dual-number
3. trial-number
4. paucal-number
5. plural-number

3.3.2 noun classes for relative adjustment

- diminutive
- augmentative
- inanimate-gender
- animate-gender

3.3.3 noun classes by animacy

this is based on a thirteen chakra system with animist world view.

0. abstract-gender
1. mineral-gender
2. vegetal-gender
3. zoic-gender
4. rational-gender
5. artistic-gender
6. research-gender
7. vehicle-gender
8. locality-gender
9. planetary-gender
10. star-gender
11. galaxy-gender
12. universe-gender

3.3.4 noun classes regarding reproductive attributes

- neuter-gender
- male-gender
- female-gender
- common-gender
- anthropic-gender

3.4 Tense

past-tense things that happened

hesternal-tense yesterday

recent-past-tense

remote-past-tense

present-tense now, things that are happening

hodiernal-tense today

future-tense things that will happen

crastinal-tense tommorrow

soon-future-tense

remote-future-tense

3.5 Aspects

atelic-aspect a
cumulative-reference process for
SPEL it is like signal processing,
at any point it is still processing,
perhaps for parallel processes

cessative-aspect for ending process
for SPEL exiting process
for hardware description language
falling edge

completive-aspect completely and thoroughly
finished
for SPEL finished with no errors

continuative-aspect process started but
not active
for SPEL idle processes

delimitative-aspect temporary process

frequentive-aspect repetitive process
for SPEL can be used for servers/daemons

gnomic-aspect general truths
for SPEL defining functions

habitual-aspect habitual process
for SPEL can be provided services or features

inchoative-aspect begining of process
for SPEL loading process
for Hardware Description Layer signal rising edge

imperfective-aspect for SPEL a process which is ongoing
any partial process

momentane-aspect for things that happen suddenly or momentarily, like power surges and lightning bolts. For instance a clock-tick could be momentane and frequentive.

perfective-aspect any whole process
for SPEL a process which has completed

progressive-aspect for active process
for SPEL for active processes

prospective-aspect for processes that happen after
for SPEL queued processes

retrospective-aspect for processes that happen before
for SPEL prerequisite processes

telic-aspect quantized process
for processes where any of the parts are not the whole, only taken together is it the whole.
for SPEL this is processes that require sequential components of a different kind.

3.6 Grammatical Mood

admonitive-mood warning, I warn you that
for SPEL error messages

affirmative-mood agreeingly, I agree that
for SPEL selection of correct output, as per reinforcement learning or genetic algorithms

apprehensive-mood fearfully, I fear that
for SPEL throwing exceptions

assumptive-mood assumingly, I assume that
for SPEL assert statements

conditional-mood if such and such
for SPEL conditional clauses

commissive-mood I commit to, I promise that
for SPEL setting calendar events, and personal virtue goals, also for unit-tests
for Parliment, to send motion to comittee

benedictive mood blessings, I wish the blessing that
for SPEL increasing priority of a process

deductive-mood deductively, I deduce that
to mark conclusions through deductive inference

deliberative-mood shall I?, do you think that?
for SPEL asking user input

deonitic-mood I should, I ought to, I plan that
for SPEL pseudo-code

delayed imperative in future do that
for SPEL, scheduled jobs

desiderative-mood I want to, I desire that
for SPEL near term goal setting

dubitative-mood doubtfully, sarcastically, I doubt that
for SPEL for inverse assert statements

inductive-mood inductively, I derive that
to mark conclusions through inductive inference

inferential-mood for things which are infered based on premises.

epistemic-mood perhaps, I consider it possible that

Table 3.2: Aspect Tree

- state
    - perfective-aspect
        * momentane-aspect
        * completive-aspect
    - imperfective-aspect
        * continuous-aspect
        * progressive-aspect
        * delimitative-aspect
- occurence
    - gnomic-aspect
    - habitual-aspect
- part of time
    - inchoative-aspect
    - cessative-aspect
- relative time
    - retrospective-aspect
    - prospective-aspect

<h3>Lexical</h3>

- composition
    - atelic
    - telic
    - stative-verb
- causal
    - autocausative-verb
    - anticausative-verb

eventive-mood in the event that
for SPEL event catchers

gnomic-mood generally, In general I believe that
for SPEL function declaration

hortative should, I urge that

imperative-mood you must, I command that
for SPEL, imperative programming sentence

imprecative mood curse, curse that
for SPEL decreasing priority of a process, also for setting up security measures, such as firewalls, honey pots and others

indicative-mood indicating the real, I indicate that
for SPEL variable declaration

interoggative-mood questioningly, I question that
for SPEL search queries

irrealis-mood unreal sentence, it isn't real that
for SPEL comments

jussive-mood tell them to, I command them that.
for SPEL issuing commands to remote location

necessitative-mood I need that
for SPEL listing of required libraries

optative-mood I wish that
for SPEL long term goal setting

potential-mood possibly, I consider it possible that
for SPEL try statements

permissive-mood I permit that
for SPEL setting and limiting priviliges, also for SPARK style contracts
for Parliament set limits of debate

precative-mood I request that
for SPEL making network requests and pull requests
for Parliment amendments

prohibitive-mood don't, I forbid that
for SPEL, blocking certain things, or ignoring certain inputs

propositive-mood I suggest that, I propose that
for Parliment as a main motion starter in deliberative discussion

realis-mood It is real that

sensory-evidential-mood evidence I've experienced tells me that
for Parliament or Court, to bring evidence before the assembly, also to mark premises in logical arguments.

subjunctive-mood unreal clause

speculative-mood speculatively, I guess that

volitive-mood desires, wishes or fears

hypothetical-mood For things which aren't necessarily true, but could easily be true, from the speakers perspective.
also for spel catch statements

Table 3.3: Grammatical Mood Tree

- realis
    - indicative
    - evidential
    - energetic
- irrealis-mood
    - deontic
        * commissive
            · permissive
            · prohibitive
        * directive
            · imperative
            · hortative
            · precative
            · necessitative
            · jussive
        * volitive
            · desiderative
            · optative
            · apprehensive
            · benedictive
            · imprecative
    - epistemic
        * interrogative
        * speculative
            · assumptive
            · dubitative
            · potential
        * inferential
            · hypothetical
            · inductive
            · deductive
    - would be
        * conditional
        * eventive

Chapter 4

Dictionary

4.1 Emotions

4.1.1 By neutrotransmitter

acetylcholine alert attentive

dopamine pleasure

serotonin satisfaction

norepinephrine vigilance

gaba inhibition

glutamate excitement

oxytocin belonging

vasopressin territoriality

4.1.2 By brain Regions

amygdala fear, anger

thalamus drowsy, alert

hypothalamus reward, arousal

cingulate gyrus gumption perserverance

basal ganglia motivation

orbitofrontal cortex deliberative

prefrontal cortex equanimity self-control

ventral striatum goal direction

insula disgust

4.1.3 modifiers

- strong
- weak

4.1.4 from Wikipedia

Affection platonic love, oxytocin

Anger violent desire for immediacy, amygdala, andrenaline

Angst weak anxiety

Anguish strong anxiety

Annoyance weak anger

Anxiety fearful anticipation

Apathy non feeling

Arousal level of wakefulness thalamus

Awe surprise and fear

Boredom weak disgust

Confidence certainty

Contempt anger and disgust

Contentment satisfaction, serotonin

Courage equanimity surpassing fear

Curiosity seeking new information, dopamine

Depression strong sadness and rumination

Desire to want

Despair

Disappointment

Disgust insula

Distrust

Dread

Ecstasy

Embarrassment

Envy

Euphoria

Excitement

Fear

Frustration

Gratitude

Grief

Guilt

Happiness

Hatred

Hope

Horror

Hostility

Hurt

Hysteria

Indifference

Interest

Jealousy

Joy

Loathing strong disgust

Loneliness

Love

Lust

Outrage

Panic

Passion

Pity

Pleasure

Pride

Rage strong anger

Regret

Remorse

Sadness

Satisfaction

Schadenfreude

Self-confidence

Shame

Shock

Shyness

Sorrow

Suffering

Surprise

Trust

Wonder

Worry

Zeal

Zest

4.2 Prosody

4.3 Trochaic Rhythm

First syllable strongest, emphasis on odd syllables, grammar words always unemphasized.
http://wals.info/chapter/17

4.4 Espeak

Espeak unfortunately does not have trochaic rhythm support at this time (Feb 2017)

Part II

Compiler

Chapter 5

Specification

SPAL simple compile to OpenCL.

5.1 stages of compilation

1. natural language text (perhaps)
2. analytic language text
3. SPAL language text
4. SPAL encoded tiles
5. (OpenCL) C with SPAL names
6. (OpenCL) C with natural names (perhaps)

5.2 Method for implementation

In theory can use any language for implementation. Though ideally would be a version of C which is similar to the above, so it could then be recoded in SPAL.

5.3 answer verification

The agree debug library is OpenCL and holy ceremeny (pure function) compatible.

Ideally would have a way of listing many inputs and their corresponding outputs. If this could be fed to an OpenCL kernel that would be delicious.

The agree debug library can be the ``testing framework'' for SPAL programs. So each agree statement adds a line to the newspaper, after the program is complete it can list the statements in the newspaper, saying those are the tests that failed. Additionally could have a list of the number that have passed.

I'm thinking can save both the line number, and the amount that have passed in the first line of the newspaper. It can be an actual sentence, with two 16bit spaces for the values.

gzat na hnuc do lweh hnuc do mwah slak fa li

A newspaper until number with number succeeded.

A newspaper should be at least 16 sentences long, which is one page or 512 bytes, and less than or equal to 512 sentences, (32 pages), since that is the most that could fit in L1 memory with other processes.

Pyash	SPAL	C	file
kratta krathnimna li	cardinal _top cardinal name _nom _rea	int main () {	cardinal_name.c
swicta hnimna li	social _top name _nom _rea	void name () {	cardinal_name.c
hmasta hnimna li	mind _top name _nom _rea	inline void name (); inline void name () {	library_cardinal_name.h library_cardinal_name.c
krathmasta hnimna li	cardinal mind _top name _nom _rea	kernel void name () {	cardinal_name.cl
htipdoyu txikka hciccu	ten _num _ins indexFinger _acc down _con		if (i < 0xA) {
zrundofi	0 _num _return		return 0;
fe	_finally		}
hnimna tyindo cyah	name _nom three _num _cop		name = 3;
txikna zrondo cyah	indexFinger _nom zero _num _cop		i = 0;
htipdoyu txikka hciccu hyikdoyu plosliwa htekhromli	ten _num _ins indexFinger _acc down _con indexFinger _acc one _num _ins plus _rea _and library program _rea		for {;i < 0xA; ++i}{ library_program ();}

5.4 Memory

There is no dynamic allocation of memory, only static, until further notice.
   This is because historically dynamic allocation of memory has led to many memory leaks and other problems.

5.5 Control Flow

Instead of the traditional for loop that relies on variables defined outside it's bounds, the for loops in SPAL only contain a function, and a listener to see if it should break early.
   That way based on the size of the for loop, the compiler can assign it to run on single thread, multi thread CPU, or GPU.

5.6 translate all independentClauses to C

Any independent-clause can be turned into C.
   can be of the form:

```
sort1_case1_sort2_case2_verb_mood (sort1 name, sort2 name);
```

5.6.1 C Name Composition

For C, will need to include the types of the names in order to properly call functions, otherwise would have to have extra searching to locate which function is being refered to.
   This will make it a bit like Navajo or Swahili, where the noun class will be mandatory. So we should have easy grammar words for them,
   for names of things:

plu paucal-number 8bit

do number 16bit

pu plural-number 32bit

ml6h multal-number 64bit

ml6hhsosve multal-number sixteen vector, vector of 16 64bit values.

fe referrential, pointer

crih letter, char

crihfe letter referrential, char *

   It seems I would only need a hash table lookup for operating on the GPU, seems like most of the other stuff can be done with a few conditionals.

Chapter 6

Operation Template

6.1 overview

6.1.1 translation

An English programmer writes English text.
An English encoder encodes the English text to the Pyash medium code.
A Chinese translator decodes the Pyash medium code into Chinese text.
A Chinese programmer writes Chinese text.
A Chinese encoder encodes the Chinese text into the Pyash medium code.

6.1.2 Compiler

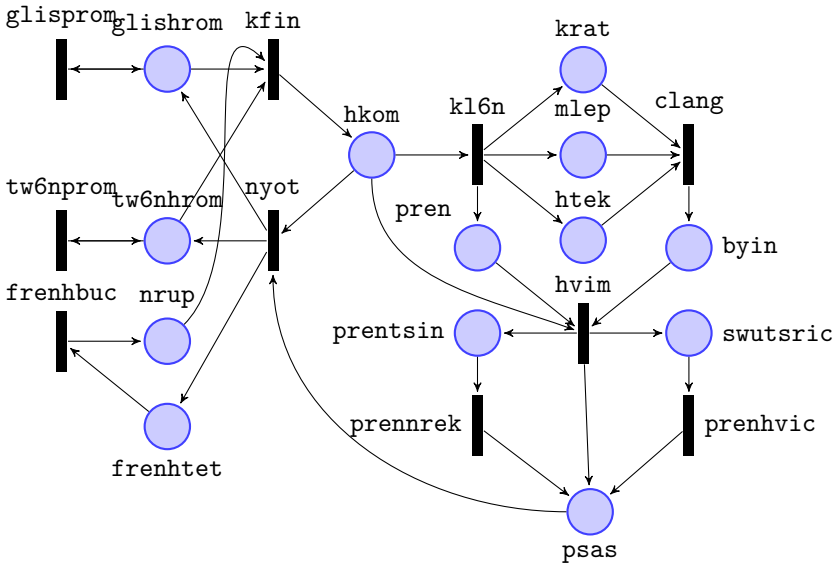
A code compiler from the medium code, to a cardinal ``.c'' file, library header file, and library ``.c'' files, as well as a kernel ``.cl'' file, and library file of intermediate code.

Clang compiler takes main and library ``.c'' files, and library header files, and produces an byin binary.

The byin binary operator sets up the constant stack, input data and makes writeable output data.

The host code starts the virtual machine kernel, and library kernels.

Figure 6.1: Compiler Petri Net



glisprom	English programmer
twynprom	Chinese programmer
glishrom	English program
tw6nhrom	Chinese program
frenhbuc	French user
frenhtet	French text
nrup	input
kfin	encoder
hkom	code
nyot	translator
kl6n	compiler
psas	produce or output
krat	cardinal file (main.c)
mlep	template file (lib.h)
htek	library file (lib.c)
pren	parallel file (kernels.cl)
hvim	runtime
prentsin	parallel knowledge (parallel da
prennrek	parallel workers
swutsric	holy ceremonies
prenhvic	parallel virtual-machine
psas	produce (output)

Chapter 7

Pyash Encoding

The virtual machine uses variable-length-instruction-word (VLIW), loosely inspired by head and tails instruction format (HTF). HTF uses VLIW's which are 128 or 256 bits long, however there can be multiple instructions per major instruction word.

7.1 VLIW's Head Index

The head is really a parse index, to show the phrase boundaries. In TroshLyash each bit represents a word, each of which is 16bits, when a phrase boundary is met then the bits flip from 1 to 0 or vice-versa at the phrase boundary word. index takes up the first 16bits of the VLIW. This would lead to 256bit (32Byte) VLIW's. The real advantage of the indexing occurs when there either multiple sentences per VLIW, or when there are complex sentences in the VLIW's. Having the VLIW's broken up into 32Byte chunks, makes it easier to address function entry points, which can be placed at the beginning of a VLIW. Can fit 16 VLIWS in a POSIX page, 128 VLIW's in a Linux page, so would only need 1 byte (8bits) for addressing functions that are within 1 page distance.

7.2 Word Compression

Now for the slightly more interesting issue of packing as many as 5 glyphs into a mere 16 bits. Why this is particularly interesting is that there is an alphabet of 32 glyphs, which would typically required 5 bits each, and thus 25bits in total. However the 16 bit compression is mostly possible due to the rather strict phonotactics of TroshLyash, as only certain classes of letters can occur in any exact place. The encoding supports 4 kinds of words, 2 grammar word classes and 2 root word classes. Where C is a consonant, T is a tone and V is a vowel, they are CVT, CCVT, and CVTC, CCVTC respectively.

7.2.1 CCVTC or CSVTF

I'll start with explaining the simplest case of the CCVTC word pattern. To make it easier to understand the word classes can call is the CSVTF pattern, where S stands for Second consonant, and F stands for Final Consonant. The first C represents 22 consonants, so there needs to be at least 5 bits to represent them. Here are the various classes

```
``C'' :``p'',``t'',``k'',``f'', ``s'',``c'',``x'', ``b'',``d'',``g'',``v'', ``z'',``j'',
    ``n'',``m'',``q'',``r'', ``l'',``y'',``w'',

``S'' ``f'',``s'',``c'',``y'', ``r'',``w'',``l'',``x'', ``z'',``j'',``v'',
```

``V'' ``i'',``a'',``e'',``o'',``u'',``6'',

``T'' ``7'',``_'',

``F'' ``p'',``t'',``k'',``f'', ``s'',``c'',``n'',``m''

, (can check the phonology page for pronunciation) C needs 5 bits, S would need 4 bits, however the phonotactics means that if the initial C is voiced, then the S must be voiced, thus ``c'' would turn into ``j'', ``s'' into ``z'' and ``f'' into ``v'', also none of the ambigiously voiced phonemes (l, m, n, q, y, w, r) can come before a fricative because they have a higher sonority, thus must be closer to the vowel. So S only needs 3 bits. V needs 3 bits T needs 2 bits and F needs 3 bits which is a total of 16 bits. 5+3+3+2+3 = 16 However there are other kinds of words also. we'll see how those work.

7.2.2 HCVTF

So here we have to realize that CVC or CVTC is actually HCVTF due to alignment. So what we do is make a three bit trigger from the first word, the trigger is 0, which can be three binary 0's, 0b000 3+5+3+2+3 = 16 H+C+V+T+C this does mean that now 0b1000, 0b10000 and 0b11000 is no longer useable consonant representation, however since there are only 22 consonants, and only 2 of those are purely for syntax so aren't necessary, so that's okay, simply can skip the assignment of 8, 16 and 24.

7.2.3 CSVT

This is similar to the above, except we use 0b111 as the trigger, meaning have to also skip assignment of 15, 23 and 31. $3+5+3+3+2=16$?+C+S+V+T

7.2.4 CVT

For this one can actually simply have a special number, such as 30, which indicates that the word represents a 2 letter word. $5+5+3+2+1$ $F+C+V+T+P$ what is PF P can be a parity-bit for the phrase, or simply unassigned.

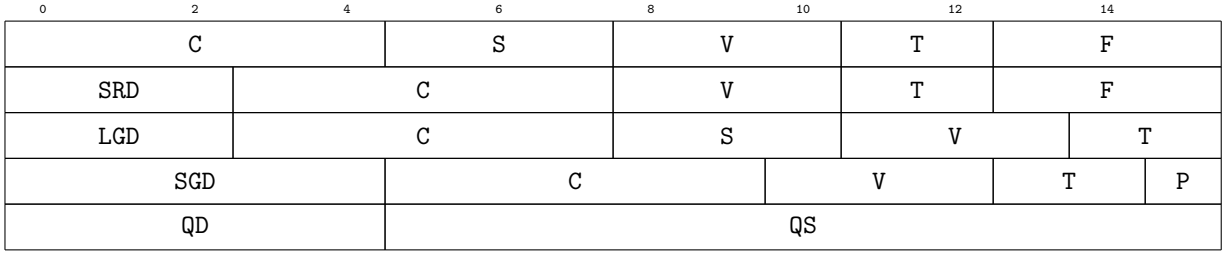
7.3 Quotes

Now with VM encodings, it is also necessary to make reference to binary numbers and things like that. The nice thing with this encoding is that we can represent several different things. Currently with the above words, we have 1 number undefined in the initial 5 bits. 29 can be an initial dot or the final one, can call the the quote-denote (QD), depending on if parser works forwards or backwards. Though for consistency it is best that it is kept as a suffix (final one), as most other things are suffixes. 5+3+8 = 16 Q+L+B QD has a 3 bit argument of Length. The Length is the number of 16bit fields which are quoted, if the length is 0, then the B is used as a raw byte of binary. Otherwise the B represents the encoding of the quoted bytes, mostly so that it is easier to display when debugging. The type information is external to the quotes themselves, being expressed via the available TroshLyash words. So in theory it would be possible to have a number that is encoded in UTF-8, or a string that is encoded as a floating-point-number. Though if the VM interpreter is smart then it will make sure the encoding is compatible with the type Lyash type, and throw an error otherwise.

7.4 Extension

This encoding already can represent over 17,000 words, which if they were all assigned would take 15bits, so it is a fairly efficient encoding. However the amount of words can be extended by increasing number of vowels, as well as tones. And it may even be possible to add an initial consonant if only one or two of the quote types is necessary. However this extension isn't likely to be necessary anytime in the near future, because adult vocabulary goes up to around 17,000 words, which includes a large number of synonyms. For instance the Lyash core words were generated by combining several different word-lists, which were all meant to be orthogonal, yet it turns out about half were internationally synonyms, so were cut down from around eight thousand to around four thousand words. It will be possible to flesh out the vocabulary with compound words and more technical words later on. Also it might make sense to supplant or remove some words like proper-names of countries.

7.5 Encoding Tidbit Overview



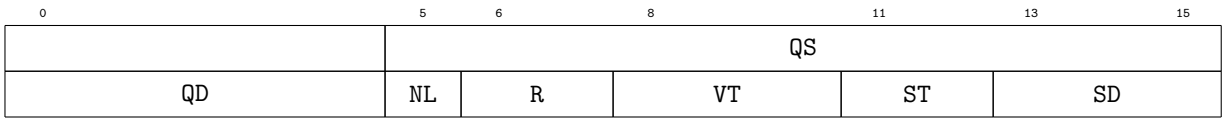
Legend C	Init
S	Seco
V	Vowe
T	Tone
F	Fina
SRD	Shor
LGD	Long
SGD	Shor
P	(opt
QD	Quot
QS	Quot

7.6 Table of Values

#	C	S	V	T	F
width	5	3	3	2	3
0	SRD	y /j/	i /i/	E	m /m/
1	m /m/	w /w/	a /ä/	MT ///	k /k/
2	k /k/	s z /s z/	u /u/	7 //	p /p/
3	y /j/	l /l/	e /e/	_ //	n /n/
4	p /p/	f v /f v/	o /o/		s /s/
5	w /w/	c j / /	6 /ə/		t /t/
6	n /n/	r /r/	4 // (U)		f /f/
7	LGD	x /x /	3 /æ/ (U)		c //
8	SRO				
9	s /s/				
10	t /t/				
11	l /l/				
12	f /f/				
13	c //				
14	r /r/				
15	LGO				
16	SRO				
17	b /b/				
18	g /g/				
19	d /d/				
20	z /z/				
21	j //				
22	v /v/				
23	LGO				
24	SRO				
25	q /ŋ/				
26	x //				
27	1 //				
28	8 //				
29	QD				
30	SGD				
31	LGO				

	blank means out of bounds
E	Error signal
U	unused
MT	middle tone, no marking
QD	quote denote
SGD	short grammar word denote
SRD	short root word denote
LGD	long grammar word denote
SRO	short root word denote overflow
LGO	long grammar word denote overflow

7.7 Quote Sort



7.7.1 definitions

QS quote sort

QD quote denote

NL name or literal bit

R region

VT vector thick

ST scalar thick

SD sort denote

16 tidbit					
5 tidbit	1 tidbit	2 tidbit	3 tidbit	2 tidbit	3 tidbit
QD	referential	region	vector	scalar thick	sort denote
definitions					
0	name	private	1	1 byte, _paucal_number	letter (s)
1	literal	worldwide	2	2 byte, _number	word (s)
2		preordained	4	4 byte, _plural_number	sentence (s)
3		coworker	8	8 byte, _multal_number	binary data
4			16		unsigned integer
5			U		signed integer
6			U		floating point
7			3		function

The quote denote is 5 bits long, leaving 11 bits. the next 2 bits is used to indicate bit thickness of quote scalar (s), the following 3 bits is used to indicate the magnitude of the vector (s), 1 bit for name or literal

letter l _letter

word word _word

phrase word _acc _phrase

sentence word _acc _rea _independent_clause

text

function

datastructure

named data type

unsigned integer one two three _number (291)

signed integer one two three _negatory_quantifier _num (-291)

floating_point_number two four _floating_point_num ten _bas one _neg _exponential _num (2.4)

fixed_point_number two _flo one _num (2.1)

rational one _rational three _num (1/3)

decimal number ten _bas one one _num (11)

hexadecimal number sixteen _bas eleven _num (11)

vector world _word _and _voc _word two sixteen word _vector (vector of 16 unsigned shorts each short containing a word, intialized to repeating sequence of ``hello _vocative_case'')

In the case of a refferential, or variable name, the name can be (up to) four words long, that way it fits in a 64bit area --- similar to a 64bit address.

	0 source-case	1 way-case	2 destination-case	3 location-case
0 base	nominative-case	instrumental-case	dative-case	accusative-case
1 space-context (x)	ablative-case	prosecutive-case	allative-case	locative-case
2 genitive-case	possessive-case	descriptive-case	possessed-case	relational-case
3 discourse-context	initiative-case	topic-case	terminative-case	vocative-case
4 social-context	causal-case	evidential-case	benefactive-case	comitative-case
5 surface-context (y)	delative-case	vialis-case	superlative-case	superessive-case
6 interior-context (z)	elative-case	perlative-case	illative-case	inessive-case
7 time-context (t)	initial-time	during-time	final-time	temporal-case

Table 7.1: grammtical-case number system

7.8 Independent-Clause Code Name

Decided to make the independant-clause code name actually a universal hash, based on the sorts, cases, aspects and mood of the sentence. It's easier that way.

The grammatical cases can have a table to make it easy to identify them.

Part III

Machine Intelligence

Chapter 8

Machine Programmer

8.1 Overview

A human programmer writes a function template, function suggestions and either provides a working function or sample input and output data.

An encoder encodes the function template and function suggestions into the intermediate representation (IR).

If the human programmer provides a working function, then the function profiler takes the function template IR and working function and generates the sample input and output data.

A population generator takes the function suggestions IR and input from /dev/random to create the population IR.

The population compiler converts the population IR into kernel or ``.cl'' files, one for each.

The population tester loads each population kernel, and streams the sample inputs through them, checking outputs for correctness, and produces the population fitness which includes fitness of all individuals.

The champion selector takes the fitness ratings, and the population IR, and outputs the champions.

The population mutator and recombiner takes the champions and function suggestions, then generates a new population IR.

An output generator takes the champions and outputs the best ones to a file.

Bibliography

[1] Gerard J. Holzmann. "The Power of Ten - Rules for Developing Safety Critical Code". In: *NASA/JPL Laboratory for Reliable Software* (). URL: http://pixelscommander.com/wp-content/uploads/2014/12/P10.pdf.