Lisp
Is a
Sanskrit
Parallel

This is a 'quick 'n' dirty' draft thought-note for review, especially aimed at Computer Science people.

(Fork it at https://github.com/lambdadi/sicp.)

CONTENTS

1 Preface

The following thought-note first sketches out direct parallels and then demonstrates them by citing examples.

SOURCES: All material and ideas in this note are drawn from five books:

i. ***Structure and Interpretation of Computer Programs*** (SICP) (2nd ed., as revised in 1990), ISBN 978-81-7371-527-3;
ii. ***Vedic Mathematics*** (Revised 1992), ISBN 81-208-0164-4;
iii. ***Scientific Nagari Phonography*** (Pilot Edition, 1994), Published by Institute of Typographical Research (Pune);
iv. ***वैदिक स्वरचिन्हे*** (Vedic Swarachinhey) (1st ed., Oct 2010, in Marathi), Sant Dnyaneshwar Ved-Vidya Pratishthan (Aurangabad); and
v. ***Sanskrit and Science*** (1st ed., 1984), Bharatiya Vidya Bhavan.

LANGUAGE: The precise and closely reasoned terminology of the SICP text book is directly used, but only as a device to ***spell out*** identical design and implementation features of both Lisp and Sanskrit. Not mere patterns, but actual features. CS people will recognize all the ideas written in the SICP book language.

EXAMPLES: Vedic Mathematics may be used to practically demonstrate all the direct parallels. To do a “Quick 'n' Dirty” elaboration, only two Vedic Math examples are used in this note.

NEXT STEPS: Many more subtle nuances are left for an expanded “Version 2”; to be completed (see also: APPENDIX B). The burden of iron-clad proof may be abstracted away to a “Version 3”.

NO CLAIM TO IP: None of the main ideas contained below originally occurred to me. I am only putting two-and-two together and sketching a picture. These connections are available for anyone to discover by choice and/or chance. So even (and especially) in the off chance this note becomes really significant, I disclaim ownership of everything herein. It should be for the world.

2 Striking Parallels

2.1 Standards First

Lisp is, first and foremost, *a standard*. And so is Sanskrit:

- ***Pre-existing, universal and near-unchanging atomic constructs*** underpin both standards. The symbolic logic of Lambda Calculus is to Lisp, what the phonetic logic of Vedic "Maheshwari" Phonography is to Sanskrit.
- ***Both standards were evolved*** to progressively control and manage complexities arising from the desire to achieve general-purpose as well as special-purpose knowledge processing at any scale.
- ***Both standards were codified*** to ensure undistorted and near-timeless propagation of _underlying knowledge processing principles_ without being burdened by dependence on any single - and by nature, arbitrary - language implementation.
- ***This holds true even though*** the Lisp standard evolved to fit the medium of essentially serial-processing hardware, whereas the Sanskrit standard evolved to fit the medium of essentially networked-processing wetware.

2.2 Computational Firepower

Both standards provide, in full measure, all three key mechanisms common to all powerful programming languages:

- ***Primitive expressions,*** which represent the simplest entities the language is concerned with,
- ***Means of Combination,*** by which Compound Elements are built, and
- ***Means of Abstraction,*** by which compound elements can be named and manipulated as units. Very strikingly, both foster Procedural Abstraction, Higher Order Procedures including General Methods of computation, Compound Data and Data Abstraction, all the way up to Metalingual Abstraction.
- ***Special comment:*** The notion of *List Processing* is embedded in the very core of the Vedic system, as will be illustrated in one of the Vedic Math examples to follow.

2.3 "Interpretedness"

Both standards call for implementation of _interpreted_ languages:

- ***Raw transliteration*** is practically possible *and* easy, implicitly, by virtue of such “interpretedness”. Lisp permits high-fidelity transliteration of Lisp code into all Lisps as well as virtually all other known languages – high-level to machine-level. Similarly Sanskrit permits phonetically perfect raw transliterations of all Sanskrit-standards-based language implementations (Hindi / Marathi / Tamil / Malayalam / Bengali etc.) as well as any other known spoken language – human or animal.
- ***Conversely, such a property further*** (profoundly) permits any computer/person to use suitably interpreted code, even though that

person/computer may be *completely blind* to the parent code's internal logic and/or multi-dimensional code expansions that can be generated by imposing other types of interpreters. By this power, and as long as the underlying symbolic/phonetic logic holds true, a self-describing code base may be accurately propagated and used in one way or another by many, if not most, computing systems across many generations.

- ***The fact of "interpretedness"*** stays true even though Computers and People, so far, possess fundamentally different operating environments. In other words we use different computational processes. On one hand, the processes and time spans of interpreter installation are different. Computers can be "trained" to interpret Lisp implementations even with one-click-installs. And people must be trained to use Sanskrit interpretations via repeated recitation of pre-defined material using pre-defined methods. But on the other hand, whereas one computer may operate with the Global Environment of only one Lisp interpreter at run time; one person may concurrently operate with compound (if not several) Sanskrit-based as well as non-Sanskrit-based interpreters at run time.
- ***Philosophically,*** users are allowed to modify the language and even the interpreter itself. In both cases, success is normally follows (say) one or two decades of broad and deep exposure to _solving real problems_ using at least one standard language implementation. (Interim newbie experiments may of course be conducted, but only in the seclusion of a favorite (and preferably far away) forest clearing.)

2.4 Dialectic Evolution

Last but not least, both standards have historically derived themselves from, and later spawned, dialectic evolution in highly similar ways:

- Each standard was invented with a goal to provide a near-perfect base for general-purpose knowledge processing, _in response to_ difficulties arising from idiosyncrasies of specific languages that had evolved till that time.
- The standard then spawned a fully general purpose, expressive, yet succinct classical implementation.
- Common Lisp is to the Lisp standard, what Vedic Sanskrit is to the Sanskrit Standard.
- Language evolution is driven strongly by the pursuit of Great Power of Expression with Great Succinctness, which is a cherished and often-stated ideal of both communities.
- Standardization enabled several brand new languages as well as derivative dialects, including general-purpose and/or special-purpose variants.
- Scheme, Clojure, Auto-Lisp etc. are to Lisp/Common Lisp what Pali, Prakrit, Marathi etc. are to Sanskrit/Vedic Sanskrit. Just like Clojure was invented to make esoteric Lisp cooperate with the common man's Java platform; Marathi was invented to translate esoteric Vedic know-how in a bid to force that community to cooperate with the common person.

3 Two Illustrative Examples

3.1 "Vargamula" Sutra

A Formula-Procedure to find Square Root by "Straight Squaring", *Vargamula* is taken from Chapter #34 of the Vedic Mathematics book, referenced in the preface.

To show operation of the *Vargamula Surta* and to prove its validity, I will also need to invoke and explain several other Sutras and their corollaries; so better the original book than I. But explanation of its construction itself gives enough insight for the purpose of this thought-note.

Vargamula's construction:

- ***It requires application of another sutra*** called *Dvandva Yoga*, or "The Duplex Combination Process".
- ***But Dvandva Yoga can be interpreted to mean*** "squaring" (a^2) or mean "cross-multiplication" (2ab) or mean *both* squaring *and* cross-multiplication (a^2 _and_ 2ab). *Dvandva Yoga* itself is a general-purpose procedure that invokes *Urdhva-Tiryagbhyam* in general, and in the event of special cases, invokes optimized procedures such as *Yavadunam Sutra*. The *Urdhva-Tiryagbhyam* procedure, in turn, is a general purpose division procedure that accepts *any* arbitrary real number, algebraic expression, partial fraction, and compound integral (having a numerator and denominator).
- ***So Vargamula must somehow abstract away such complexity*** and force *Dvandva Yoga* to generate new meaning i.e. cough up square-roots. This abstraction is achieved by forcing re-interpretation of input data and also invoking some cool number theory procedures.
- ***The input data re-interpretation rule says:*** "The given number is arranged as two-digit *groups* from right to left, and a single digit if left over at the left hand end is treated as a simple *group*. The number of digits in the square root will be the same

as the number of *digit-groups* in the given number and a single digit if any such there be. Thus if the square root contains n digits, the square must contain 2n or 2n-1 digits." This is nothing but ***treating input data as a List*** rather than a single atomic value.

- ***Still further, it invokes several number theory procedures namely:***
    - that 1, 5, 6 and 0 at the end of a number reproduce themselves as the last digits in its square;
    - that squares of the complements from ten have the same last digits i.e. the complementary pairs of 1^2 and 9^2; 2^2 and 8^2; 3^2 and 7^2; 4^2 and 6^2; 5^2 and 5^2; and 0^2 and 10^2 all have the same endings, namely, 1, 4, 9, 6, 5 and 0 respectively; and
    - that 2, 3, 7, and 8 are out of court altogether, as the final digits of a perfect square

Therefore, *Vargamula* is a compound procedure *and* it operates on compound data. And I suspect that because it encapsulates general-purpose nature of *Urdhva Tiryak*, *Vargamula* may also be extended to find square roots of any arbitrary mathematical function. This is nothing but procedural as well as data abstraction. At any rate, I can't help but feel that the idea of transforming a squaring procedure into a square root procedure at run-time is exactly similar to Lisp macro behavior.

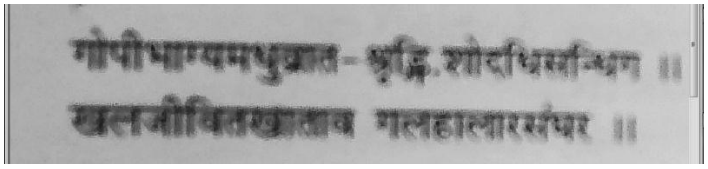
In summary, *Vargamula* deconstructed...

- ***Demonstrates*** procedural as well as Data abstractions
- ***Demonstrates List-processing*** of data where Lists can be of *arbitrary size*
- ***Demonstrates dynamic typing*** as it does not care if the input is whole or irrational
- ***Points at*** deep insight into orders of growth (Theta) of the processes generated by procedures, in time _and_ in space
- ***Hints that*** it is possible to demonstrate macro behavior, if the Sutra has a general enough scope;
- ***Uncovers*** one Extra-Important additional idea of, at least, the Vedic Math meta-Sanskrit: The Formula _is_ The Procedure. It appears to me that, in Vedic Sanskrit, it is not compulsory to think of imperative and declarative knowledge as always separate. This, I feel, may provide an order of magnitude more succinctness, as the task of translating between imperative and declarative formulations is rendered non-binding.

3.2 A Deceptively Innocuous "Shloka"

The Shloka (couplet/aphorism) is taken from page 348 of the Vedic Mathematics book referenced in the preface.

It is composed in अनुष्टुभ (Anushthub) meter.



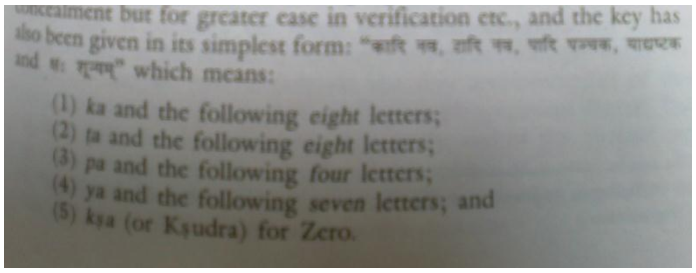
गोपीभाग्यमधुव्रात-शृङ्गिशोदधिसन्धिग ॥
खलजीवितखाताव गलहालारसंधर ॥

(This picture will be replaced by a better scan.)

GopiBhaagyamDhuvraat-ShrugdiShodadhiSandhiga|
KhalaJeevitaKhaataav GalahalaraSanghar||

This *Shloka* is so worded that it is an ode to Lord Sri-Krishna. A few clever wordplays also make it a praise of Lord *Sri-Shankara*. But now get this – if you apply the Vedic Alphabetic Code to it, the *Shloka* expands into the value of (π/10) accurate to *32 decimal places*. Blammo!

The book claims (but sadly does not explain) that it also holds a self-contained "Master Key" to expand its value to *any number of decimal digits*. Double-blammo! But it at least helps us decode the 32 digits of said *Shloka*. The key to this code is found in an aphorism that reads as:



concealment but for greater ease in verification etc., and the key has also been given in its simplest form: "कादि नव, टादि नव, पादि पञ्चक, याद्यष्टक and क्षः शून्यम्" which means:

(1) ka and the following *eight* letters;
(2) ṭa and the following *eight* letters;
(3) pa and the following *four* letters;
(4) ya and the following *seven* letters; and
(5) kṣa (or Kṣudra) for Zero.

(This picture will be replaced by a better scan.)

Ka-adi nav Ta-adi nav Pa-adi pancchak Ya-adyashtak Ksha shunyam cha|

So the expanded code is found to be based on *consonant sounds* and reads as follows:

1. ka, ta, pa and ya all **denote 1;**
2. kha,tha, pha and ra all **represent 2;**
3. ga, da, ba and la all **stand for 3;**
4. gha, dha, bha, and va all **denote 4;**
5. gna, na, ma and sa all **represent 5;**
6. ca, ta, and s'a all **stand for 6;**
7. cha, tha, and s.a all **denote 7;**
8. ja, da, and ha all **represent 8;**
9. jha and dha **stand for 9;** and
10. ks.a (or Ks.udra) **means Shunya or Zero!**

Vowels do not find a place in the composition, hence make no difference; and in conjunct consonants, the last consonant alone is to be counted.

So applying the code to the Shloka generates the list:
.3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3
2 3 8 4 6 2 6 4 3 3 8 3 2 7 9 2... **Yes, it is π/10!**
(I can't pronounce all of it properly beyond .3141, so I can't verify the remaining digits. But even just .3141 is so cool!)

Now while the book *doesn't* say it, I felt there could be a more profound idea embedded in it. So I let abstraction run amok and found that metaphysical interpretation yields an astonishing unification of all three results of the Shloka:

- ***Brahma, Vishnu, and Maheshwar*** respectively are "The Creator", "The Preserver", and "The Destroyer" of the entire known Universe.
- ***"Lord Sri-Krishna" is an encapsulation,*** in human form, of Lord Vishnu a.k.a. "The Preserver".
- ***"Lord Sri-Shankara" is an encapsulation,*** also in human form, of Maheshwar a.k.a. "The Destroyer".
- Now, there is ***no mention of Brahma*** anywhere in the Shloka. Indeed, his work begins only if there is no Universe.
- ***By way of this exclusion,*** the Shloka also might have something to do with the fact of an *existing* Universe.
- ***But what has π got to do*** with the two good fellows and the Universe?
- ***Vedic seers also stated that the shape*** of the circle permeates the entire Universe. *"Akhanda mandalakaram vyaptam yaen characharam";* they said. And now it gets weirder...
- ***Also the Vedic idea of the very Universe*** is that it is net-net *"Shunya"* i.e. Zero, which they also denote by a circle. (By the by, the circle also denotes the idea of *"Gati"* i.e. velocity. But I'll stop short of saying "planetary motion".)
- ***So the very notion of a circular shape*** is also a common factor.
- ***And lo and behold,*** one can indeed describe *any circle* (or sphere) of any scale at any point in the Universe, with *sufficiently accurate* value of π which, the claims says, the *Shloka* can generate.

So, the deconstruction of this Shloka...

- Demonstrate Primitive Expressions, Means of Combination, and Abstraction.
- Is a model of Succinctness and Expressiveness.
- Demonstrates "Interpretedness" and shows how "Sanskrits" use Multilingual Abstraction.
- Hints that Shlokas and Sutras can perhaps themselves be treated as data and used to further generate far more profound sets of abstractions. For more on this point, also see APPENDIX [A].

4 Conclusion

While I will attempt a "Version 2" subtitled "A More Generalist Elaboration", I *deeply* believe that actually a "Version 3: A Truly All-Encompassing Elaboration" needs to be written. I am not, as yet, capable of it but there are plenty who are. And I hope someone does it. And if they do, I hope that their work will be entitled *"**S**anskrit **I**nterpreted **C**omputation and **P**rogramming"*, in deference to the high standard of the modern work that so completely and delightfully imparts the reasoning necessary to express not only the poetic verisimilitude but also the hard-nosed engineering design features that are identically leveraged by the world of Lisp as well as the world of Sanskrit.

Thanks for reading so far!
- Aditya
https://github.com/lambdadi/sicp

APPENDIX

[A] Fantasy-filled Food For Thought

All CS people (particularly Lispers) know this well:
A particular procedure may require *itself* to be *augmented, at run time,* by another procedure or even a complete program, outside of itself, as a precondition for complete and/or error-free execution.

Now consider this:
It is a widespread and well known Sanskritic tradition to invoke the *name* of an appropriate "God" ***before commencing*** a certain type of task. Such an invocation is done in a very particular manner viz. "Aum <God X's name> Namaha".

This may readily translate to:

- ***Interrupt*** to Modify Global Environment *(procedure named "Aum")* +
- ***Load*** all known procedures associated with procedure-as-parameter *<God X's name>* +
- ***Pass parameter*** to modify behavior of procedure <God X's name>. In this case <*Namaha*> modifies it to *"I Invoke Thy Name and Commence."*

This is of course sheer fantasy. But just *suppose* one entertains the idea. Could it at least be in the *realm* of possibility?

Here is a fantasy-filled illustrative example:
By convention, one chants *"Aum Ganapatayay Namaha"* prior to commencing any study of *any* field of knowledge. And the plot thickens by way of this deconstruction:

Symbolically, the word *"Ganapati"* is associated with an *abstract being* capable of flawless and tireless capture of *all* knowledge. This expands into transcribing all audio-visual input into phonetically perfect inscribed output while, at the same time, committing the expanded contextual understanding to memory.

Procedurally, let me direct attention to the 6th Canto of the popular prayer-book, the *Ganapati Athava-Sheersham*. It is said to be the *beeja-mantram* (atomic representation) of the root sound *"Gam"* that creates the word *Ganapati*. Close analysis reveals that the linguistic and typographic key to Devnagri is coded therein. The literal translation of the canto is:

1. First recite the sounds (गणादीम् पूर्वमुच्चार्य), and only after, write them down (वर्णादीम् तदनंतरम्).
2. To make the *Mantram* whole, add to it that Nasal sign (अनुस्वारः परतरः), the basic element of which is a crescent (अर्धेन्दुलसितम्). Recite the *Mantram* in an ascending pitch (तारेण रुध्दम्).
3. And now, such is the graphic of your *Mantram*; (एतद् तवं अनुस्वरूपं)... The first element is the first part of the sound "G" (गकारः पूर्वरुपं) and the middle one is the verti-bar of the sound "a" (अकारो मध्यमरुपं) and the last element is the nasal sign (अनुस्वरक्ष्चान्त्यरुपं) with the top dot (बिन्दुरुत्तररुपं).
4. The final composition (संहिता) and its pronunciation (संधिः), is what is known as Ganesh Vidya (सा एषा गणेशविद्या).

This, in one short verse codifies the *entire system* of phonetics-based knowledge processing:

1. Recitation comes first, and only then may the written word be created.
2. Recitation needs the right intonation.
3. The written word will progress from left to right.
4. Only a combination of the written *along with* the spoken will capture all knowledge completely.

This hints that there *may* be several *Shloka-procedures* hidden in the *Ganapati* code base that could augment our embedded (neural network) interpreters at run time.

Process-wise the standard Vedic system of training also neatly follows the above model:

- Perform phonetically, metrically and rhythmically accurate chanting of text matter inscribed as verse
- Do this in combination with very specific hand motions to reinforce meter, rhythm, and inflections. By convention, only the dominant hand may be used (by dogma this has become the right hand).
- Furthermore, only after the student nails the recitation and also accurately commits the text matter to memory, may permission be given to inscribe. In the Vedic convention, inscription always *follows* mastery of recitation.
- Testing of learned material is frequently performed, typically via unscripted dialogue between the learner and the teacher. This, to my mind, is a way to enforce real-time analysis of the learner's run-time *eval/apply* capacity.

Now, does this training pattern not appear like ***the way to program a neural network*** capable of and indeed empowered by, concurrent use of "multiple intelligences"?

Imagine further, how it might look in a Lisp REPL?

```
(load "d:\\my-schemes\\aum-ganapatayey-namaha.scm")
; Pre-loads several procedures into current Global Environment, including "read-correctly" "say-precisely" "commit-to-memory" "do-till-eval-apply-mastered"
```

```
(learn-sanskrit book-1.papr book-2.papr book-3.papr)
; Depends on augmenting itself with pre-loaded "Ganesh" procedures for successful and error-free execution
```

```
(learn-law (attack-books 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19))
; Also depends on augmenting itself with pre-loaded "Ganesh" procedures for successful and error-free execution
; But it takes a long time to execute, so if you like you may also pre-load "d:\\my-schemes\\regular-coffee-break-generator.scm"
```

And finally Grok this: The very first paragraph of the SICP book reads thus... *"We are about to study the idea of a* ***computational process****. Computational processes are* ***abstract beings*** *(emphasis mine) that inhabit computers. As they evolve,* ***processes manipulate*** *other* ***abstract*** *things called* ***data****. The evolution of a process is directed by a pattern of rules called a program. People create programs to direct processes. In effect, we* ***conjure*** *the* ***spirits*** *of the computer with our* ***spells****.*

[B] Ideas for "Version 2: A More Generalist Elaboration"

This may possibly be written as follows:

- FIRST Explain, by *several* analogies, how complexity is controlled and managed. Use things commonly observable by physical senses - like Music and Traffic. Avoid difficult subject matter like Biology or Math.
- Using the analogy, define the core ideas of Primitives, Means of Combination and Means of Abstraction.
- For example one may use music to explain it thus:
    - ***Primitives:*** e.g. Sing and show the 7 musical notes Sa, Re, Ga, Ma, Pa, Dha, Ni, Sa. These, nobody can mess with, and so are standard.
    - ***Means of Combination:*** e.g. Sing and show how each next note must be at a fixed distance from the first note Sa. Show how harmonics may be created, but only in relation to the base notes. And so, all possible musical scale relationships may be created.
    - ***Means of Abstraction:*** e.g. One may represent a single note or combination of several notes by means of any abstract word (like "So long! Farewell! It's time to say adieu") as long as the word stays true to the note or series of notes.
    - Show how notes are primitives for words; words are primitives for rhymes; rhymes are primitives for musical pieces; musical pieces are primitives for complete Operas. And so, all music of any sophistication may be produced. Said another way, musical complexity is controlled and managed.
- NEXT use only the terminology of Primitives etc… And ***visually*** reinforce how they are a general pattern upon which all systems evolve to massive complexity. Add cells to generate tissue to generate organs to generate organ systems to generate people to generate families to generate villages to generate society to generate government to generate war and peace. Where no abstraction ever needs worry about the details of the abstraction that came before it.
- NEXT write some Lisp code ***in plain English*** and ***actually build*** upwards to abstractions. Introduce nuances (well-commented, of course).
- FINALLY reproduce the selfsame Lisp abstractions using Vedic Sanskrit standards

And so, clearly articulate how Lisp Is a Sanskrit Parallel.