The Computer Nonsense Guide

Live for the Swarm! 31.07.2019

Abstract

"An object is really a function that has no name and that gets its argument a message and then look at that message and decide what to do next." — Richard P. Gabriel

This guide is product of the efforts of many people too numerous to list here and the unique environment of our open-source community.

The work presented here confines itself primarily to the stabler parts of the system, and does not address the window system, user interface or application programming interfaces at all.

Spacebeam is a custom Debian stable (x86_64) distribution that enables end users easily build computational clusters, grid endpoints and visualization on tiled-displays; installation, consulting, and support is also available from community members.

We are an open-source research & development community that conducts multidisciplinary work on distributed systems, artificial intelligence and high-performance computing.

There are countless different kinds of use of all the things we call "signs", "words", "sentences". And this diversity is not something fixed, given once for all; but new types of language-games come into existence and others become obsolete and get forgotten.

Our Mission: provide tools inside a simple workspace for play, work and science!

Our Goal: a distributed AI toolkit and workspace environment for machines of all ages.

Core ideas

- Functions are a form of objects.
- Message passing and function calling are analogous.
- Asynchronous message passing is necessary for non-blocking systems.
- Selective receive allow to ignore messages uninteresting now.

Getting started

Your system need the latest release of Erlang, LuaJIT (with luarocks) and Singularity installed.

Installation

Then run this command:

```
luarocks install experience
```

For help, including a list of commands, run:

```
$ exp --help
```

Congratulations, you are jacked up and good to go!

Cognitive framework

Jean Piaget's theory of cognitive development proposes that humans cannot be given information, in which they immediately understand and use. Instead, learners must construct their own knowledge. They build their knowledge through experience. Experiences enable them to create mental models of the world. These models are changed, enlarged, and made more sophisticated through two complimentary processes: assimilation and accommodation.

Definition of intelligence

Piaget's definition of intelligence itself does not consist of an isolated cognitive process. It is not, one form of structure among others; it is the form of equilibrium towards which all the structures or cognitive processes tend.

Intelligence is only a generic term to indicate the equilibrium of cognitive processes.

He believed, human beings inherit a tendency to organize their intellectual processes and to develop particular adaptations to their environment.

Intellectual adaptation is also an exchange between a person and its environment and involves the same two processes of assimilation and accommodation found in biology.

Assimilation involves the person dealing with the environment in terms of its structures, while accommodation involves the transformation of its structures in response to the environment.

The particular ways in which an organism adapts and organize its processes depend also on its environment and learning history.

Cognitive constructivism

Cognitive constructivism is based on two different senses of construction. First, on the idea that people learn by actively constructing new knowledge, not by having information poured into their heads. Moreover, constructivism asserts that people learn with particular effectiveness when they are engaged in constructing personally meaningful artifacts (e.g. computer programs, animations).

View of learning

Because knowledge is actively constructed, learning is presented as a process of active discovery. The role of the instructor is not to drill knowledge into students through consistent repetition, or to goad them into learning through carefully employed rewards and punishments. Rather, the role of the teacher is to facilitate discovery by providing the necessary resources and by guiding learners as they attempt to assimilate new knowledge to old and to modify the old to accommodate the new.

What is an organization?

A monkey, a building, a drone: each is a concrete object and can be easily identified. One difficulty attending the study of organizations is that an organization is not as readily visible or describable.

Exactly what is an organization such as a business concern? It is a building? A collection of machinery? A legal document containing a statement of incorporation? It is hardly likely to be any of these by itself. Rather, to describe an organization requires the consideration of a number of properties it possesses, thus gradually making clear, or at least clearer, that it is.

The purposes of the organization, whether it is formal or informal, are accomplished by a collection of members whose efforts or behavior are so directed that they become coordinated and integrated in order to attain sub-goals and objectives.

Perception and behavior

All of us at some point or another have had the experience of watching another person do something or behave in a certain way, saying to ourselves, "She/he acts as if she/he thought, ... " and then filling in some supposition about the way the other person looked at things.

Simple as the statement "She acts as if she thought ... " may be, it illustrates two important points.

First, what the person thinks he sees may not actually exist.

They could act as if changes in methods as an attempt by management to exploit them.

As long as they had this attitude or belief, any action by management to change any work method would be met, at the very least, with suspicion and probably with hostility.

The second point is that people act on the basis of what they see.

In understanding behavior, we must recognize that facts people do not perceive as meaningful usually will not influence their behavior, whereas the things they believe to be real, even though factually incorrect or nonexistent, will influence it.

Organizations are intended to bring about integrated behavior. Similar, or at least compatible, perceptions on the part of organizational members are therefore a matter of prime consideration.

Clues

One of the first things we must recognize is that in learning about things we not only learn what they are, that is, that the round white object is for football, but we also learn what these things mean, that is, football is a sport that the USA men's team don't get and their woman counterpart have master perfectly.

Upon receiving a signal (sight of football) we perform an interpretative step by which a meaning is attached to it.

Many of these "meanings" are so common and fundamental in our understanding of the world that we fail to note them except under unusual circumstances.

One way these meanings are brought home to us is by meeting people from countries different from our own; many of the meanings which things have come from our culture, they are things all people within the culture share.

These common interpretations of things help enormously in communicating, but they sometimes make it difficult to set factors in perspective so that we can really understand the reasons for behavior.

Threshold of perception

We all, have certain things (stimuli) to which we are sensitized and that when these appear we are instantly alert and eager to examine them.

There are other stimuli of relative unimportant to us to which we do not pay as much attention and may, in effect, actually block out.

One way of viewing this subject is to suggest that we have thresholds or barriers which regulate what information from the outside world reaches our consciousness.

On some matters the barriers are high and we remain oblivious to them, but on others which are quite important to us we are sensitized and, in effect, we lower the barrier, permitting all the information possible concerning these matters to reach our consciousness.

Resonance

Related to this idea of sensitivity and selectivity is a phenomenon that might be called resonance.

Through experience and what we see ourselves to be, the understanding of a particular item of information may be very similar to that of others.

It is explained this way: since all the people inside a group look upon themselves as peers, they know what a change on the individual means in annoyance and inconvenience.

They can easily put themselves into his shows and, once having done so, probably feel almost as disturbed as he might be.

Internal consistency

One property of the images formed of the world around us is that they are reasonable, or internally consistent.

For instance, we may look at some draw on a page and see a rabbit. One portion along these lines might suggest a duck, but we do not have an image of something half rabbit and half duck.

If our first impression was a duck, we may never notice that a portion looks like a rabbit. We seem to tune out the elements that do not fit.

Dealing with conflict

Organizations that posses the capacity to deal adequately with conflict have been described as follows:

1. They posses the machinery to deal constructively with conflict. They have an structure which facilitates constructive interaction between individuals and work groups.
2. The personnel of the organization is skilled in the process of effective interaction and mutual influence (skills in group leadership and membership roles in group building and maintenance functions).
3. There is a high confidence and trust in one another among members of the organization, loyalty to the work group and to the organization, and high motivation to achieve the organization's objectives.

Confidence, loyalty, and cooperative motivation produce earnest, sincere, and determined efforts to find solutions to conflict. There is greater motivation to find constructive solution that to maintain an irreconcilable conflict. The solutions reached are often highly creative and represent a far better solution than any initially proposed by the conflicting interests.

The essence here is that out of conflict will come a new synthesis superior to what existed before and perhaps superior to any individual point of view existent in conflict.

Conflict, resting in part on different perspectives of what "ought" to be, is one of the avenues for opening new directions for the organization or one of the ways of moving in new directions. This is not only useful but also vital for organizational survival. The question, therefore, as we view conflict is not, "How to eliminate it?" but, "Is it conflict of such a type and within circumstances where it will contribute to rather than detract from organizational interest?"

Whether a conflict is good or bad for an organization, whether a conflict can be made useful for an organization, depends not so much on manipulating the conflict itself as on the underlying conditions of the overall organization. In this sense, conflict can be seen as;

1. a symptom of more basic problems which requires attention
2. an intervening variable in the overall organization to be considered, used, and maintained within certain useful boundaries.

Adaptation frequently proceeds through a new arrangement developing informally, which, after proving its worth and becoming accepted, is formally adopted. The first informal development, however, may be contrary to previously established procedures and in a sense a violation or a subversion of them; or the informal procedures may be an extension of a function for internal political purposes.

Programmed links

If the process had given instruction to report immediately on completion of the task, this instruction facilitates linking the completed act with the next one. Through an information transfer, we call this a programmed link.

The supervisor node may detect that something is wrong through another control cycle. It can then take corrective action by including or adding into this programmed link or perhaps by attacking on the more difficult problem of human apathetic attitudes and motivation.

Progression of goals

Organizations have progression of goals which result from a division of work.

A subdivide goal becomes the task of a process contained within a specialized organizational unit.

This nesting of goals is contained as part of the core organizational means-ends chain.

Needless to say, the hierarchy of control loops which are connected with the progression of goals may be handle in number of ways, regardless of how the elements are allocated, the important factor is that all elements must be provided for in some way. Hence, our model supplies an extremely useful tool in analyzing complex control situations by telling us what basic functions bust occur and in what sequence, even though initially we have no idea as to where or how they are executed in the organization.

Goals and feedback

The feedback loop containing information about organizational performance and conditions leads to definition of sub-unit goals or standards. It's important to show how a situation in one area could lead to modifications in a number of units at higher levels.

This even result in reformulating the basic goals of organizations. Feedback is essential to adequate goal formation.

Process of abstraction

"If a system is to be stable the number of states of its control mechanism must be greater than or equal to the number of states in the system being controlled." — William Ross Ashby

In trying to understand what is happening around us we are faced with a fundamental problem. In approaching any situation, the system trying to understand it, does not attempt to gather all information. Instead it selects certain facts and searchers for others.

This selection of some items and ignoring of others is a process of abstraction.

It is the abstracting form a real or if you will empirical situation the things seemingly most important to deal with.

In this process of abstraction and model building we deliberately select a few items, ignore may others, and then place the items chosen in a particular relationship to one another.

In doing so we are intentionally ignoring facts or relationships that can influence the type of situation under study.

The problem it to select the most meaningful elements and relationships and dropout the rest.

Those who use abstraction skillfully know well that they neither have all the facts nor have considered all the relationships bearing on the outcome of what they are analysing.

We do not use the abstractions from one situation in another setting without carefully examining the fit. Neither do we expect a model to handle all aspects of a situation.

We shall be dealing with many abstractions and models, not with the intention of exactly mirroring the real world but with the objective of clarifying our perception of its most essential features.

Abstractions and models are mechanisms for economizing both time and effort, but like any tool they must be used within their limits.

Model your goals

Taking the abstracted elements, a character with the flat tire begins to connect them into a pattern.

Better yet, he weaves them into a model of the confronting situation, which we can use both to understand his plight and figure out what to do about it.

The parts of this model would probably include, among other things, the flat tire, the image of the spare in the trunk, the telephone, the service station, a forthcoming business meting, etc.

A second model would contain the telephone, the service station, and the repairman there.

Finally, it concludes that it will call a cab and leave his wife to deal with the flat tire as best as she can.

These are extraordinarily elementary models, but they serve a very practical purpose.

With them the main character in our illustration can see the likely consequences of various courses of action.

We can find out these things by doing them directly by actually handling the tire and observing that we get dirty, or by calling the repairmen and waiting for him and learning that it takes too long.

In the age of big data; big models are good.

- For any given size of data, the bigger the model, the better it generalizes, provided you regularize well.
- This is obviously true if your model is an ensemble of smaller models.
- Adding extra models to the ensemble always helps.
- It is a good idea to try to make the data look small by using a big model.

By using the model, however, we can make some reasonable predictions about what will occur and therby accept or reject the choices open to us.

Several references have been made with the intent of this guide to provide conceptual tools for analysis. As with any other tool models, abstractions and generalizations are useful only when within their limitations.

Why Erlang helps?

"Any sufficiently complicated concurrent program in another language contains an ad hoc informally-specified bug-ridden slow implementation of half of Erlang." — Robert Virding

Erlang suits iterative development perfectly, the ecosystem offers a variety of languages with different focus all build in top of the BEAM and the OTP framework.

Let it crash!

Robust systems must always be aware of errors but avoid the need of error checking code everywhere.

We want to be able to handle processes crashes among cooperative processes.

- If one process crashes all cooperating processes should crash
- Cooperating processes are linked together
- Process crashes propagate along links

System processes can monitor them and rest them when necessary but sometimes we do need to handle errors locally.

Pattern matching

Functions use pattern matching to select clauses, this is a BIG WIN™

Supervision trees

A supervisor has a standard set of interface functions and include functionality for tracing and error reporting. Supervisors are used to build a hierarchical process structure called a supervision tree, a nice way to structure a fault-tolerant application.

- Supervisors will monitor their processes through links and trapping exists.
- Supervisors can restart the workers when they terminate.

On production, this usually means a fairly straight-forward combination of external process management, overload monitoring and proxying.

A supervisor is responsible for starting, stopping, and monitoring external processes. The basic idea of a supervisor is that it is to keep its processes alive by restarting them when necessary.

Fault-tolerance

Fault-tolerance is achieved by creating supervision trees, where the supervisors are the nodes and the workers are the leaves of this analogy. Supervisors on a particular level monitor and handle children in the subtrees they have started.

If any worker terminates abnormally, the simple supervisor immediately restart it. If the process instead terminate normally, they are removed from the supervision tree and no further action is taken.

Stopping the supervisor results in all the processes in the tree being unconditionally terminated. When the supervisor terminates, the run-time ensures that all processes linked to it receive an EXIT signal.

It is a valid assumption that nothing abnormal should happen when starting your system.

If a supervisor is unable to correctly start a process, it terminates all of its processes and aborts the startup procedure.

While we are all for a resilient system that tries to recover from errors, startup failures is where we draw the line.

The BEAM virtual machine

The virtual machine runs as one OS process. By default it runs one OS thread per core to achieve maximum utilization of the machine. The number of threads and on which cores they run can be set when the BEAM is started.

Erlang processes are implemented entirely by the VM and have no connection to either OS processes or OS threads. So even if you are

running a BEAM system of over one million processes it is still only one OS process and one thread per core, in this sense the BEAM is a "process virtual machine" while the Erlang system itself very much behaves like an OS and Erlang processes have very similar properties to OS processes.

- Process isolation
- Asynchronous communication
- Error handling, introspection and monitoring
- Predefined set of datatypes
- Immutable data
- Pattern matching
- Functional, soft real-time, reactive, message-passing system
- Modules as function containers and the only way of handle code

Inside the BEAM ecosystem, we just worry about receiving messages.

Load balancing

Compacting the load to fewer schedulers is usually better for memory locality, specially on hyperthreads, the primary process is in charge of balance the rest of the schedulers. The goal is to not overload any scheduler while using as little CPU as possible.

Process stealing

Used by artists of all types and computers alike, on the BEAM is the primary mechanism to load balance and spread processes.

- A scheduler with nothing runnable will try to "steal processes" from adjacent schedulers, then next beyond that.
- We only steal from run-queues, never running or suspended processes.
- Schedulers changes on other schedulers run-queues.
- Each scheduler has its own run-queue.
- Processes suspend when waiting for messages, this is NOT a busy wait.
- Suspended processes become runnable when a message arrives.

By this mechanism the BEAM suspend unneeded schedulers. Once every period of 20k function calls is reach a new primary process inside a node scheduler is chosen.

Functions and modules

Modules contain functions, its a flat module space with just functions they only exist in modules there are no dependencies between running modules they can come and go as they please.

Functions

Functions cannot have a variable number of arguments! Erlang/OTP assumes functions with same name but different arity, each function has only a fixed number of arguments.

Modules

Modules can have functions with the same name and different number of arguments (arity), inside the virtual machine they are different functions.

Modules can consist of

- Declarations
- Function definitions
- Macro definitions
- Compile time function definitions

Macros can be defined anywhere, but must be defined before used.

The system only has compile code there is no build-in interpreter just compile code in modules. Everything is in modules the module is the unit of code handling, you compile modules, load modules, delete modules, update modules, everything run though modules there are no living functions outside modules.

We can have multiple versions of modules in the system at the same time, all functions belong to a module, this handle of modules means there is no inter-module dependency of modules at all, they just come and go when the system is running.

In this sense a running BEAM instance has no notion of a system, and can be described more like a collection of running modules.

Lua in Erlang

"Scripting is a relevant technique for any programmer's toolbox." — Roberto Ierusalimschy

Luerl is an implementation of standard Lua 5.3 written in Erlang/OTP.

Lua is a powerful, efficient, lightweight, embeddable scripting language common in games, IoT devices, machine learning and scientific computing research.

It supports procedural, object-oriented, functional, data-driven, reactive, organizational programming and data description.

Being an extension language, Lua has no notion of a "main" program: it works as a library embedded in a host. The host program can invoke functions to execute a piece of Lua code, can write and read Lua variables, and call Erlang functions by Lua code.

Luerl is a library, written in clean Erlang/OTP. For more information, check out the get started tutorial. You may want to browse the examples source code.

Luerl goal

A proper implementation of the Lua language

- It SHOULD look and behave the same as Lua 5.3
- It SHOULD include the Lua standard libraries
- It MUST interface well with Erlang

Embedded language

Lua is an embeddable language implemented as a library that offers a clear API for applications inside a register-based virtual machine.

This ability to be used as a library to extend an application is what makes Lua an extension language.

At the same time, a program that uses Lua can register new functions in the Luerl environment; such functions are implemented in Erlang (or another language) and can add facilities that cannot be written directly in Lua. This is what makes any Lua implementation an extensible language.

These two views of Lua (as extension language and as extensible language) correspond of two kinds of interaction between Erlang and Lua. In the first kind, Erlang has the control and Lua is the library. The Erlang code in this kind of interaction is what we call application code.

In the second kind, Lua has the control and Erlang is the library. Here, the Erlang code is called library code. Both application code and library code use the same API to communicate with Lua, the so called Luerl API.

Modules, Object Oriented programming and iterators need no extra features in the Lua API. They are all done with standard mechanisms for tables and first-class functions with lexical scope.

Exception handling and code load go the opposite way: primitives in the API are exported to Lua from the base system C, JIT, BEAM.

Lua implementations are based on the idea of closures, a closure represents the code of a function plus the environment where the function was defined.

Like with tables, Luerl itself uses functions for several important constructs in the language. The use of constructors based on functions helps to make the API simple and general.

The result

Luerl is a native Erlang implementation of standard Lua 5.3 written for the BEAM ecosystem.

- Easy for Erlang to call
- Easy for Lua to call Erlang
- Erlang concurrency model and error handling

Through the use of the BEAM languages, Luerl can be augmented to cope with a wide range of different domains, creating a customized language sharing a syntactical framework.