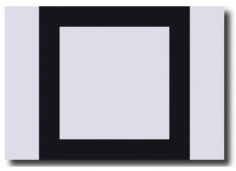
Aristo Tacoma

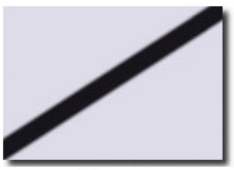
ROBOTIC GOAL SORTING


match001

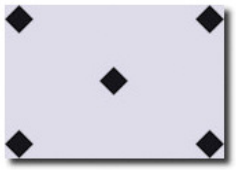

match002

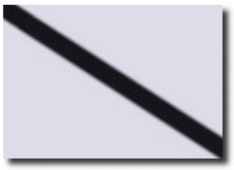

match003


match004


match005

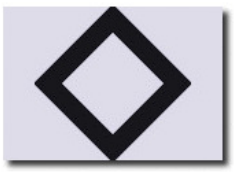

match006

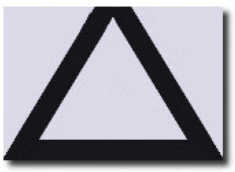

match007


match008

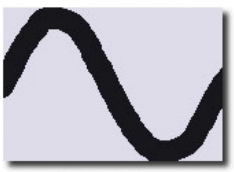

match009


match010


match011


match012


match013


match014

The Art of Thinking series

Volume 4 of 5

Aristo Tacoma

ROBOTIC GOAL SORTING

The Art of Thinking series
Volume 4 of 5

BOOK INFORMATION
Relevant G15 PMN apps for this book:
1005768 and # 1005769 and other apps
at norskesites.org/fic5
as well as at g15pmn.com
ISBN 978-82-93128-05-2
Published by Yoga4d von Reusch Gamemakers,
www.yoga6d.org
First published in 2024, first edition.
Copyright the author
Aristo Tacoma who is Stein H Reusch;
other pen names include Stein von Reusch,
Henning Braten, Stein H Braten Reusch,
and S.R.Weber--these other pen names all
family-name derived.
Redistribution of this text in respectful
contexts permitted, when text is kept
unchanged, whole, and not added to; and this
notice is part of the reproduction.
As background reading, consult, at yoga6d.org/library:
* the first three volumes in this series
* the super-model-theory.pdf from 2017, physics, and
its earlier form in this text:
* yoga4d_org_a_htm.pdf from 2004, which
is also at yoga4d.org/a.htm, an early text on physics
and numbers published in 2004/2005 on paper;
both at he National Library Library of Norway
and by this author. In addition, it is good to
read back and forth in the documentation that
comes along with G15 PMN app #3,333,333, ie,
the Third Foundation, at h33 and k33 there,
as well as the examples, in order to keep in
mind the FCM-relevant numbers.
Published by Yoga4d von Reusch Gamemakers,
Norway.
Book is printed on paper and is also
available at yoga6d.org/library

FOR NEW READERS:
This is the fourth volume in a five-volume series, "Art of Thinking". As part of this writer's journey into the theme of the art of thinking over many years, and work with mathematical logic, a sort of essence idea of the Personal Computer was shaped on the logical/conceptual level and expressed in terms of a CPU design (a CPU made in the mind, as a terminology and instruction set, and not yet, at the time of writing this book, in terms of electronics or chips). This was called G15. Along with it went a definition of the surrounding hardware, in a concrete and boundary-aware manner inspired by the logical work. Gradually an effortless programming language was built using only G15 instructions, called PMN. The first G15 PMN 'practical virtual implementation' was ready in 2012, and by 2017, a form of physics, called Super Model Theory, was formulated in a book centered on G15 PMN (see references in book information above. The G15 PMN is not dependent on typical hierarchical concepts such as 'files' and 'folders'. In short, it acts as if it is an OS, an Operating System for computers. In a certain sense, it is a world onto its own. In another, and practical/contemporary sense, it runs happily on mostly all commercial and noncommercial platforms, and interacts eminently with robotic hardware in a two-way realtime sense; and can be used for currency trading calculations with its G15 PMN FCM Spreadsheet, for writing, for graphics work, and above all, to stimulate the human mind, our minds, to think even better by being a language near to thought through which we can express our inmost concepts in a way that is complementary to natural language and free from the conceptual issues plaguing mathematical logic at the foundations.

NOTATION OF FORMAL CODE

In contast to the previous three volumes in this series, beginning with this volume, we'll put all code in uppercase except variables such as 'i1' and 'i5' where the uppercase, in some fonts, resembles the digit 1. Capital letters are fairly similar to the formal notation inside the G15 PMN platform and the sense is that readability of the code is enhanced by allowing it to stand out from the rest of the text that way.

WELCOME TO A.O.T. VOLUME 4, ROBOTIC GOAL SORTING
Also in this volume, of course, will our Art of Thinking explorations go into delightfully non-technical themes. But we also have some [correction: MUCH, inserted after getting to the completing chapter] technical work to do!]

In terms of my own writing, the book starts out just after completing app# 1005768 and, in addition to all the after all very extensive series of philosophical musings, sets out to depict the actual thinking process and G15 PMN formalism, or "coding" as we can abbreviate it, for some 800 permille or more of the app# 1005769.

The first three volumes of this series was a summing up, for me, of work done over a long time, in which the principal machinery was the Personal Computer. By it, the programming language G15 PMN was developed, then refined, and in these three volumes, one of the tasks was to express how to use this language. Another task was to prepare the ground for the coupling of the PC to motors and other machinery 'out there', in the world around the PC, in other words, to robotic things, so that an understanding of G15 PMN as also a way to meaningfully drive robots--and even build them--would emerge. At the point of completing volume 3, the books were, as far as the development of G15 PMN and robotics is concerned, up-to-date with regard to where I had got with these developments. Between the third volume and this, the fourth, I've had the opportunity to spend lots and lots of time with various robotic elements, and to experiment with how they might possibly connect to a personal computer; and this includes some experimentation with electronics.

This is a five-volume series, and as the first three volumes indicates, the fourth volume was supposed to be about robots very concretely, while the fifth going a bit beyond such themes, and include also such as a bit of EEG brain research. The title of this fourth volume, Robotic Goal Sorting, was decided long ago. Put simply, if we have got some wheels of some sort with motors, and some arm-like elements also with motors (or 'servos' as slow-moving motors typically with a movement range typically indicate by such as '0 to 360 degrees' are also called), and usually also some hand-like elements (or 'gripper'), and a couple of cameras near all this, and we have a way in which a program can send signals to all the motors and receive data from the cameras, then what makes this into a useful robot is that the signals sent work out coherently so as to get tasks done, usually with pretty strong guidance from also the data that the cameras bring home.

For a task to be coherently done in such a setup, the signals must be orderly and respect the physical features of the situation, and part of this order is sequence; and part of the sequence involves the concept that each task usually have several subtasks, and that this division can go on for quite a while. For instance, to lift a cup of coffee involves the subtask of gripping the cup, followed, in sequence, by another subtask of lifting it. And this can itself be a subtask of a larger task entitled, 'fetch a cup of coffee to the human master' :)

If the lifting-signal comes before the gripping-signal, neither the gripping nor the lifting will make coherent sense. So sequence is essential. This sequence, or, in other words, this 'sort' involves the concept of subtasks --or 'goals'--very intensely. The very concept of the robot begins to emerge as distinct from merely a wiring of motors and cameras to a PC by robotic goal sorting taking place inside the PC, correlating data from input sources like cameras to sort as correctly as can be, the goals that are then eventually translated into so low-level subtasks that they correspond to electric signals to the motors.

In other words, the very concept of the robot is a human concept intrinsically tied up to the equally human concept of the task. There need be no assumption of mindfulness for the robot as such; rather, we assume that the human being in charge of, first, the programming and wiring process is mindful, and then the human beings in charge of putting the robot to use also are mindful. It is by their coherent understanding of what is to be meaningfully done in this world that the robot comes in as a solution, making it less necessary for human beings to have to indulge in tasks that are less rewarding to do.

In an ideal society, of course, goods are fairly equally available for every citizen, and robots are their helpers and in many cases their slaves, and also do background work like overly repetitive factory work; allowing human beings to go to beach etc and put in a moderate amount of labor hours each day instead of having to sweat one's life out on a dehumanizing factory job to the benefit of an elite of bosses who, apart from doing some office hours to enforce their unethical rule, can party all day long. Neither marxism nor capitalistic democracy nor any of the attempted forms of fascism or socialdemocracy has proven to solve this. But it is no point sticking one's head in the sand and pretend robots aren't existing; robots might make an unethical rule worse, but they may also make an ethical rule better; and what we can do as programmers is to insist that we do robotics on an as ethical foundation as possible, through and through.

And as we have indicated, it must be part of our ethics, while intending that the resulting robotic setup and program shall be a good expression of a coherent mind, indeed a first-hand expression--and in that sense have a degree of 'computerized mentality'--to avoid implying that there is some independent nonhuman intelligence or mind or mindfulness in a robot, or a piece of software. In short, those who speak of 'artificial intelligence' really speak of intelligence in a phony sense, a 'fake intelligence'. This as intuition and insight, and logical argument as well--also inspired by Goedel's Second Incompleteness Theorem, runs through every aspect of G15 PMN, of course.

But to return to the progress of these volumes, it was always the intent that the fourth volume in this Art of Thinking series was going to handle robots concretely, not merely as visualized as activated through software. For that reason, I have, as said, worked much with robotic elements, trying, at most points, to avoid working with too-ready-made components by others, so that the sense in which the PC (and not eg a piece of software stored via machine code on a chip on an electronics card inside a robotic piece of machinery) is actually the center of the robot is a real one.

So, in short, every sort of relevant subtask machinery has been experimented with the past few years; I have, since completing volume 3, had various constructions going, with this sort of robotic 'arm' and that sort, this sort of robotic 'hand' and that sort, this sort of wheels and that sort of wheels, and used a variety of electronics controls --and in every occasion used G15 PMN to control these. As part of the process, I have also experimented with a range of mechanisms for making sense of what comes into the camera, in a way that doesn't chew up all the resources of a 32-bit PC. I am satisfied, and this is also expressed at the start of this book writing of this Volume 4, that the foundation ideas are in place and that the essential G15 PMN program components are in place, and these are also expressed at norskesites.org/fic5. The plan now is this: to create a description of how these foundations should be called on in a framework of finished FCM programs. Let me add here, for completeness sake, that the robotic pattern matching software components proposed in volume 3 have been exceedingly useful in inspiring further developments; and it is these further developments, rather than the exact apps made in the course of volume 3, that will be discussed here.

When the discussion of these components, and whatever philosophical, psychological or lifestyle theme we'll also bring in, is complete in this volume 4, the volume 5, in this updated plan we now make for the two completing volumes of this five-volume book series, will reflect the programs representing the physical realization as task-fulfilling robots in a clear-cut way. The reason we need the discussion is that there's still a lot of concepts that need to be sorted out--under the heading of 'robotic goal sorting'--for the programs to be made in the most effortless way, and so that they are fruitful, that they really are the Robotic FCM programs for the future, for well-working robotics at every level in society. All the existing ambitions, including dabbling with a bit of playful brain research via the combination of EEG and eroticism, as well as opening if possible yet more wide philosophical horizons to our conceptual panorama view, are still relevant and the volume 5 will simply be bigger.

This introduction of volume 4 is not complete if I do not explain a little bit about the particular role of writing can have for me--and perhaps for you as well, as a budding or advanced G15 PMN programmer--which goes like this: when it comes to programs not yet made, and which, presumably, have a good complexity about them, writing is a bit like walking-while-musing. The writing, not necessarily easy to read, not necessarily pedagogic or explanatory relative to what has been done, nor perhaps the easiest explanation of the program that is being made, while it is being made, has in it the feature of 'being a discussion with oneself' in which questions are clarified; answers are considered in the light of proposals, which are then criticized and dropped in favour of other proposals, in a process that is very gratifying, sometimes, for the writer--although a reader may feel that the sometimes entangled writing may not be the simplest explanation of the program possible.

However, the argument of including such program exploration writing as part of a book like this, is that the Art of Thinking is far greater than merely the art of explaining concepts. It is also a treatise about how to

make new thinking come about; and not merely abstractly, but by doing it. So by analogy, the reader can do similar such writing herself or himself. In addition, when complexity is high, it is not obvious that this complexity is easily remembered in detail by the programmer after the program has been made and other, new, challenging programming projects are on the task-list. And if the complexity is not well remembered, it is unlikely that it is well explained. The high-complexity program may be better explained through the intricacies of the explorative writing that led up to it.

But there is a vast difference between having this discussion on software components for robots prior to a experimentation with robotic hardware, and--as here, now, after some years of intense experimentation with robotic hardware. We have a very fortidious starting-point. And at this stage, I'm ready to sketch what I consider the perfect domestic type of robotics, which in most ways form the prototype of every type of robotics. For instance, a factory robot may simply be three times as big in all senses. It is this type of hardware that turned out to be most stable, most capable, most realisitic--and we include here results also about practical safe maneuverability in typical human indoor environments, and realism as to what type of material that should be involved in these robots, as well as realism as to how the tasks should be shaped and what not to expect from robots. This goes together with a sharply tested speed consideration when, as is the core theme in First-hand Computerized Mentality, that the first-hand Personal Computer of the 32-bit kind with the G15 PMN is at the core of it:

First of all, every robot task is context-dependent and no robot is for a 'general context'. This goes together with the philosophical understand that the human mind is essentially infinite, and that infinitude is, again, essential for perception--in a way that is beyond dependency on contexts. A digital computer is based on algorithms and these are, in a way, rules of thumbs for dealing with--not understanding--but handling context-specific tasks. In this, the camera data is not so as to 'show' the computer or the program the 'reality', but rather the camera data, and any other input data, is so as to coax the algorithm to sort its robotic goals correctly in that situation. What is picked from such as the camera is, in short, hints to the program. They are clues. The robot does 'see' that the cup is grasped, but the program can be made so as to match over clues that the cup is grasped. These clues are context-dependent. They may look exactly like clues meaning totally different things in different contexts. The only way a robot can 'discover' (and we put quotes on psychological words when used to refer to computer or robot, as part of the FCM ethics) whether the clues actually refer to what is relevant in this context, or possibly to something completely different, is that when the clues tend to stack up in an inconsistent way, relative to what is expected in this context, it may be in fact because the context has changed.

As such, the question, when an algorithm in a robot--ie, on a PC running G15 PMN with connection to the robotic machinery--is going to relate to camera data is not what the camera data 'means', but rather: is there a distinct set of camera data for one type of relevant feature of

this context, that enables it to meaningfully, correctly, assign a variable value to this feature. To return again to the 'pick-up-coffee-cup' example, a feature is:
'cup-gripping'.
There are clues, when a camera with just a few pixels analyzed by the program is directed straight to the robotic hand, that can lead a program to assign some sort of probability value to the feature 'cup-is-gripped' with correctness. This is not to say that the robot 'sees' that the cup is gripped. It is merely that there are some clues in the data--eg, the pixels in some middle upper section of the camera view are averaging brighter while some other pixels in some other parts of the camera view from another camera beaming in on the same situation tend to have also brighter values on the average--that a programmer can use to make the program create a probability value for this feature in the program. And at a probability higher than so-and-so (again, a threshold value that the programmer, later on, will find a meaningful value for), the next goal or subtask for the robot can be called on. But all the while with a constant checking in on further clues that things are progressing as intended.

For those who have worked with photos on a computer, and my work with BERLiNiB fashion magazine photos is always fruitful in giving me impulses here, they know that 150 pixels along one side of a photo that may be rectangular is a kind of magical mini-limit to really make sense of a photo for a human. Over to 100 pixels, except when the context is known, eg geometry, it leaves too much to imagination, it is a too ambigious image. Once we're beyond the 150 pixel threshold, even if this is only its height and we have eg a hundred or so in width, it starts looking like a real photo and giving you some sense of what it may be all about when you look at it larger. For an algorithm you develop in the context of computing, it can be translated into this rule of thumb: to get a proper sorting out of the clues for what the robot is next to do in a context, get at least 150 times at least 100 pixels from each camera. And since the complexities are dauting, generally speaking, for getting, in real time, on a real 32-bit Personal Computer at the core of a robot, let's not go up so much in pixel size in the general case. That is not to exclude particular cases in which special machinery with higher pixel capacities are used. Eg, you could use a computer to scan, in high resolution with a thousand or more pixels in each direction, a book, in which it would operate a camera in which the high-quality page photos are stored. That is not to say it needs to algorithmically permute all that all the time. If a camera input is 100 x 150 pixels, it is 15,000 pixels, of course, and new ones by every third or fifth or tenth second, and that for each camera. So if the robot is ever going to do anything, it needs to be staying to meaningful minimums when it comes to image processing.

Let's finish the pixel-talk right now with this summary, and I'll get over to summary of other features of the ideal domestic robot: we're doing 160 times a little more than 100 pixels reduced to 0 for dark and 1 for light at each pixel-point, for each camera; with adjustable variables as for the darkness/light threshold. As a first analysis, we do rather simple geometric matchings with 14 images of just this size--something sicular, something quadratic, something curve-like, something digagonal, and such. This replaces the more creative, artistic 'core

pat mat' elements we discussed in the apps made during the third volume in this eries. These are fun, and they work, to some extent, but further exploration showed the need to simplify. In addition, experiments (at this point not entirely concluded) indicate that the G15 PMN CPU made solely via our Intraplate first-hand electronics has just a little bit too much distance in between its elements to fully gear up the speed necessary for a typical domestic robot, and so this justifies some use sometimes of a more chip-like implementation of the G15 PMN CPU in addition to our more macro-sized and more first-hand intrplates implementations (more about electronics of this sort in a future book series).

In the apps associated with the 'fic5' robotics page of the G15 PMN works, the camera input area is depicted often direct in the spreadsheet as a matrix the size 160x112 and it looks really good and easy to work with--indeed, it is looking like something almost begging for what we call "first-hand programming". Because it is both meaningful as a whole AND composed of visually distinct squares which either are bright spring-green or black, and not so many it is overwhelming to think about looping through them.

In G15 PMN terms, we're talking of a loop that very pleasantly looks like this in its periphery:

```
LL:112
LL:160
..do something with i1 and i2 here..
LO
LO
```

Experiment with wheels showed that rubber wheels of a tracked kind is stable and good for indoor maneuvering, without damaging most types of floors. In most cases, the domestic robot is dangerously unstable if too tall and the need for keeping it as low as can be vastly enhances security. However some tasks in a typical human habitat requires some more height, and after experimentation, we found that a platform of motor-adjustable height right above the tracked wheels and underneath everything else of the domestic robot would be ideal here. That under the supposition that when the robot is using an elevated stance of the 'electric tea table' between the tracked wheels and the rest of itself, it does so as moderately as possible both in terms of height and in terms of time, before going back to a more 'animal' or 'dog-like' height.

As before, our opinion is that a proper robot is not android or humanoid nor made to look like any animal. Rather, a robot proper should look like a robot--ie, like a machine, that is very obviously a machine, and--in particular for the domestic robot--that has a general radiance of being a slave to the human beings living there. This will sharpen the minds of those who live there about the clear-cut distinction between the machine and living mindful human being, instead of contributing to a culture in which that which is artificial and that which is naturally alive is blurred. Again, of course, this is part of our often-expressed FCM ethics. And FCM is something we originated alongside the Firth platform in 2006.

However, when we look at practicalities, in order to have a flexible robotic arm like the excellently chinese-made xArm7, in which the robotic hand is its 8th motor or 'servo', it is heavy, it is big, and it is not powerful--it can only handle a couple of kilograms at most. And if

you mount it on top of tracked wheels with the electric tea table (as we can call it) between, you don't have much space for more, if the robot is going to be effortlessly enough be able to go in and out of doors and openings between rooms. You can't mount a car-making robot arm to tracked wheels of a domestic robot size; you can't get even a kilogram to be handled if you are going to have real rich and necessary flexibility compressed into small space. This has to do with the practicality of the world of magnets and motors that are easy to make without going beyond the inventions of this world entirely. However, to get many domestic tasks done, one such 'arm' as xArm7 is too little and what we need is not just another like that, but some easily handled power-lifting tools that these arms can make use of it to get things beyond their range done.

The decision to call on xArm7 took place after a range of experiments with alternatives--not only because of the superior stability in the electronics connecting the PC to the arm, but also because it took next to no time to get the G15 PMN language, in a modified version of the G15 PMN FCM spreadsheet, to directly steer each and every aspect of the movement of the xArm7. In other words, we got our vehicle by it.

In sum, we talking of a domestic robot as, in fact, consisting of two technologically separate robots that are doing collaborative work, each one having tracked wheels, a tea table, and a flexible arm capable of somewhat more than a kilogram, and some cameras, and usually also some flashlights mounted and such. These can by wire (or in very well-tested circumstances by some radio control) be connected to a G15 PMN computer (which in some cases may be small enough to be part of the domestic robot), and to power supplies (or to some batteries that may be located near the tracked wheels also to provide additional gravitational stability for the robot). Typically, the two domestic robot may have different G15 PMN computers and so both of them will need to get information from the human beings as to what is to be done, and the FCM program is made so that, at essential points, they do things together.

Just how any one of them gets anything done, and even more so, when two of them do something together, that's all part of robotic goal sorting and what we will explore in this book. When we are at volume 5, what is here will be summed up in finished apps that have been put to much good work in actual FCM robots doing a lot of different things, and whatever additional fine-tunings and/or changes to the design in these apps relative to what we explored here on the conceptual level will be detailed in volume 5.

Let's get to it!

INFINITY RECAPTURED

It may be boring, and it may even to some minds be dangerous, but once in a while, to do anything in robotics and indeed anything in any philosophical realm, we once in a while need to "recapture infinity". The argument is my own entirely, originated by myself, and, as far as I can tell, there is nothing anywhere in the technological or mathematical world that reflects any understanding of this argument at all--except G15 PMN, which has been made on the premise of this argument. (For reference, see the list in yoga6d.org/library given at one of the first pages of this book.) When done with this, we're back to the main theme of the book.

Once you have grasped this logical argument, then, as far as numbers is concerned, you have as it were entered through the looking-glass and nothing is as before. Ie, no numbers 'behave' as they used to. To live in the world, you have to step out of it again, but the experience will linger--you will know that there is an alternate experience available, and one that hasn't any disproof, simply because it is real. But what it points to may not ever fully be understood. The point of the argument is to show the limits of human knowledge, not to make a new system out of it.

I will slighly motivate the argument by reminding the reader that the assumptions surroundings such as numbers 1, 2 and 3, and infinities, are at the ground of mostly all mathematical and technological thinking, design and construction in our present human society. It is assumed, for instance, that the so-called 'natural numbers' constitute a clear, coherent concept--a sort of group of abstract concepts, namely, these numbers;-and that this group is, in sum toto, infinite; and that there are other infinities, such as the idea of the whole collection of so-called 'real numbers' like 3.141519.. and all the other conceivable decimal numbers. If all this is hogwash, so is human science.

Or, to be more precise, if all that which was just mentioned is characterised by illusions, one must apply much more caution than that which is otherwise assumed when one regards something as 'known'. One of way showing this caution is to apply more quotes "'" around words; another way is to be ready to ask for instances of confirmation and disconfirmation (as such as Rudolf Carnap and Karl R Popper recommended in the theory of science in early 20th century) for any statement of fact, and then be aware of the possible multitude of interpretations in the light of various theory horizons (the phrase 'theory horizon' in contrast to just 'theory' or 'perspective' goes back to inspiration from my late father, Stein Braten).

Practical knowledge would still exist, but more in the sense of rules of thumbs rather than as visions of the world or what is beyond it.

So, the argument begins by dissolving the notion that there are any such coherent concept as "the collection of all natural numbers", where natural numbers are assumed to work on the principles of simple arithmetics such as addition, substraction, multiplication and division. A toddler can be taught that one plus two equals three. But you do not tell the toddler that there is a concept of all such numbers, and that is well and good, for there is no such concept. Let us see why.

[P] Proposition P: There is a concept of a collection of all natural numbers, such as 1, 2 and 3, which behave according to typical rules of arithmetic, such as addition and where 1 plus 2 equals 3, and so that this collection has only such members; in other words, that every member of this collection exhibit the same finite properties of numbers as 1, 2 and 3 do.

We shall now show that Not-[P] is correct.

And we shall show it by reductio ad absurdum, ie, that by assuming, [P] we come to a self-contradiction.

Note that we have not used any such concept as "set", and this is on purpose, for the argument shows that all set theories in mathematics and technology are incoherent, unless they concerns sets, and operations on sets, in which something such as the exact lower and upper boundaries are known numerically. What we are showing involves that where finitude is assumed, infinities may come in after all, and, upon reflection, this means also that where lack of self-reference is assumed, self-reference may come in after all (and lack of self-reference is a key, if not the key, to get most of mathematical and technological set theory to work).

A direct consequence of [P] is this proposition:

[Q1]: Every element in the collection mentioned by [P] admits to arithmetic of the finite kind, and, unlike the the imagined size in-toto of the whole collection, is finite.

From [Q1] this follows:

[Q2]: As for the collection mentioned by [P], there is d a principal distinction between the finitude of the size of its members and the infinitude of the size of the whole collection.

And from [Q2] this follows:

[Q3]: The collection of all natural numbers such as mentioned by [P] is not self-referential--it obviously cannot have itself as a member because its members are all finite but its size is infinite.

Whatever lettering, whatever words we use, this sort of reasoning, up until this point, is such as that Georg Cantor and Bertrand Russell would have nodded to. (In the 19th and 20th century, Georg Ferdinand Ludwig Phillip Cantor came with a so-called diagonal argument, which presupposes such as the assumptions above, and which, quite apart from some other parts of his thinking, entered into common mathematical and logical assumptions also through the works of the philosopher Bertrand Russell, and which are regarded as schoolbook-truths today and essential in driving forth the arguments for most of the handling of infinities in most of the mathematical and technological realms of human knowledge today.)

The Reductio ad Absurdum argument takes place in a context where we, by normal humane common sense logical assumptions, wish our statements to come forth as either argued for, or against, by the strengths of the full set of arguments. If by this strength we can argue both in favour of, and against, a statement, we are, in a sense, saying nothing. In other to say something rather than nothing, we must track down any self-contradictions. And if a self-contradiction can be shown to follow rather directly in the wake of a seemingly 'innocent' and 'obvious' proposition, then the proposition can, logically, be said to have been shown to be wrong. The pathways of easy, natural, common sense logic shows that from [P] we get to [Q1], [Q2] and [Q3]. If we can show

that eg. [Q2] is wrong, we have shown that [P] is wrong. If [P] is wrong, then even the ground-breaking and, we might say, 'essential' result of Kurt Goedel in his 2nd Incompleteness Theorem from about 1930 vanishes, because he, as mostly everybody else in the mainstream philosophical/mathematical world are assuming such as [P], [Q1], [Q2] and [Q3], and cannot get any of their thought-works done if they are not 'permitted' to entertain such notions. (This does not mean that what Kurt Goedel implied --namely, that human intuition supercedes algorithms, is wrong; rather, it strengthens that insight but brings a completely different logical backing for it.)

Let us now, remembering that [Q2] in this our context is attacked, use natural, common, easy, logical arguments to actually view and imagine the collection that [P] talks about.

One of the things that is important when we look for perhaps also hard-to-see incoherencies in a concept is that we make the inherent assumptions abundantly clear. Just what is the difference between 1, 2, and 3, and when do we get up to 10, and to 100, and to 1000, and so on? This, of course, depends on the 'number system'. And the absolute simplest number system in the universe is what we should use here, rather than any funny/quick way of writing numbers adapted for the purpose of quick calculation. Here, we should see the numbers as clearly as possible--not just a notation for them.

The simplest way to represent a number is a set of lines or marks, and without any systematization or grouping of them. The romans had I for 1, II for 2, and III for 3, but when they got to 4 they wanted IV as short-hand. We want no short-hand when we are to perceive the essence of infinity. Let us use stars, and we can write them either horisontally or vertically, eg like this:

```
* * *
* *
*
```

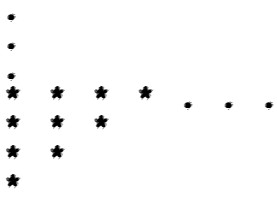
We are imagining the start of the collection of all natural numbers. We begin at the bottom of the triangle with '1' or one star. We go to the next line in the triangle and have '2' two stars. Then to 3.

We are building the collection mentioned in [P], presumably, and we are having two directions 'growing' here, our imagination. One is the size of the collection at each point, which is, of course, the vertical left line:

```
*
*
*
```

This is represented in our 'slow-motion' representation of the collection as the 'height' of the triangle. By proper spacing, the height is at each point in extension of this triangle with more members geometrically equal (eg, in terms of centimeters or inches) to the length of the upper line, which represents the upper number. Here is the triangle again at four members, and the next triangle has the adventurous 'three dots' that is the mathematical or logical equivalent for the common sense term "et cetera":

```
* * * *
* * *
* *
*
```

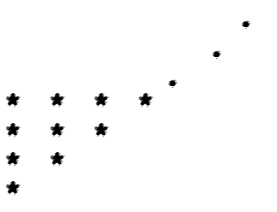


The latter triangle, which reminds us faintly of a bikini slip, has a vertical set of dots to indicate "and now we grow it in the direction of more and more members". And it has of course a horisontal set of dots to indicate "and now we grow it by 'longer and longer members'--which is to say, higher and higher numbers. In our number system, the most simple in the universe, the size of the symbol of the number in terms of how much space it occupies, equals the value of the number. A 1000 stars, to represent a 1000, is a thousand times as 'long' number as a single star, which represents the number 1. And this permits us to visualize more exactly what we are talking of in [P]. And it permits us to deduce a new proposition:
[Q4]: At each point in constructing the collection mentioned in [P], we have a symmetry in which the size of the collection is mirrored by the value of its latest addition.

Again, we have used nothing but common sense and clear logic and slow, good thinking to reach the series, [P] => [Q1], [Q2], [Q3] and [Q4]. But if we sort of re-awaken the mind of a Bertrand Russell and ask him to look at [Q4], we can guess that, while almost certainly nodding to it, he would slightly wrinkle his nose. Because, what we have all the time pointed out--that there is an incoherence in [P] here--is here, for the first time in our exposition, now beginning to show. We haven't, in [Q4], discussed anything about infinity yet, and one of the things Russell knew one would have to avoid--and that Goedel showed Russell he couldn't quite avoid in the way he imagined, at first--is self-reference in an infinite set. And with [Q4], we are getting self-reference with a finite set. That's no crisis. But we aren't done yet.

Before I go further, I wish you to meditate for a moment --but don't glue this visualization, please, to your mind because your mind has features also of the algorithmic and --as with Russell's set theory--algorithms don't stand much self-reference and certainly not when we get beyond boundaries. So don't use up your brain on chewing on arguments like these. Let them be as rain, clearing the air and when the Sun again is shining, do something reflecting that; don't wade around in the rain of infinity and get funny like many of the old infinity-mathematicians did, if we are to believe some of their biographies.

And the meditation, or pondering, or contemplation, or musing, is this: imagine now that this collection,



or half-bikini-slip, in a sort of abstract platonic world of human concepts, has, in fact, gone all the way to complete itself. Now there is no call for saying, 'Don't do that visualization; it's enough to visualize the start of the set! For if we are to proceed from talking about the numbers we count on fingers and teach to our toddlers to whether there are more types of infinities than one, and such, then we have got to take our own construction processes at the abstract level serioiusly enough that we pursue them to completion. The full and complete collection mentioned in [P] requires a mind-leap, it calls for it, the mind-leap is a necessity, a logical necessity, and absolutely required in order to detect whether our thinking is clear or muddy. So let us do it. We now, hereby, in our minds, reflect over the completed condition of the formal, abstract set of all the natural numbers. And we do so by our clear, natural, easiest-in-the-universe depiction of numbers by the stars above, and imagine that, by the three dots, we go, actually--but only in mind--"all the way".

And if you do it, it is an instant immensity of mind. At no point were there a breaking of symmetry between the growth of stars in the left vertical direction and the growth of stars in the upper horisontal direction. That is why it is entirely proper without any discussion to let the three dots be diagonally placed, up to the right, in this context of construction. Note that fallacy of writing the oft-repeated N={1,2,3,...} as seen in the thousand schoolbooks. The 'N', or set of Natural Numbers, has the illusory right angular bracket, the '}', to close it off; while the numbers 1, 2, and 3 are not shown for what they mean, but only assumed to be known, and the dots are applied only to them, and not so as to show the growth of the size of the set together with the growth of the value of its members. So for all its simplicity, N={1,2,3,...} is a false simplicity, for it is misleading.

I bring the imagine again and again encourage you to engage in mental abstract logical visualization and to get intuitions about this visualization:

```
          .
        .
      .
* * * *
* * *
* *
*
```

Karl R Popper wrote that (despite his insistence on the empirical checking of theories) he also believed (and in the context of mathematics) in intuition in order to check theories. It is because intuition is required to ascertain the quality of a logical proof that there can be, and is, disagreements about proofs. This intuition is the only way in which any argument about infinity can be analyzed. A computer, whether of the classical digital kind or made by some other method, cannot ascertain this kind of proofs. Only human minds can do that, and on the assumption that intuition exists, an intuition that can delve into infinitudes. By this intuition, when you watch the above, and remember our [Q4], we'll get to our [Q5]. But do it slowly, because we are nearing now the negation of [P]. Here again is our [Q4]:

[Q4]: At each point in constructing the collection mentioned in [P], we have a symmetry in which the size of the collection is mirrored by the value of its latest addition.
And, to not anticipate too fast, here is a next step. Again, I call on your combined powers of abstract imagination and intuition to see that what we are here talking about are logical deductions and not merely some kind of poetry on top of our postulates--but you have about a million schoolbooks and university books that point in other directions, so do it with proper slowness:
[Q5]: As we gradually, more and more, imagine that collection mentioned in [P], and represented in the simplest-way-possible by the triangle of stars, are grown more and more without obvious boundaries, we do not see any breakdown of the symmetry between the vertical length (ie, the size of the collection) and the horisontal upper length (ie, the value of the most recent addition to the collection).
In short,
[Q6]: The construction mentioned in [P] implies an absolute symmetry between, at each point in the visualized construction, the size of the collection and the value of its highest member. And we maintain a continuum of visualization here in which there is no point, even as we begin to entertain the notion of infinity, in which this symmetry is "cut off".

Now, if a revived Bertrand Russell were with us, he would here almost certainly wrinkle his statuesque nose--quite possibly also shake his head and murmur something about having a headache. And we have sympathy with him. This is the breakdown-point of most of human knowledge, for, as I have often claimed, h.k. is having as an axiom--a false axiom, imao., that it has grappled infinity quite well. An I'm not impressed by the writings of many of those who claims to be believers in a religion or some sort of spirituality either, for they seem to often support this axiom, even if a hidden way, while they may profess support to a general idea such as 'God's ways are entirely beyond human understanding'. Nor is it solved by inventing new symbols trying to capture 'omega' numbers or something fuzzy like that. There is no easy solution other than changing your life; and that's why there is such inertia around finiteness/infinity assumptions in human knowledge.

To give our imagined friend Bertrand Russell some credit, he did in fact have a period of doubt--when he encountered Cantor's Diagonal argument for real, the first time. The doubt vanished. But the mere fact that he wrote down that he had such doubt, shows, by the lips of, so to speak, the The Guru of Natural Number Sets himself, that intuition--and time, human time, the time involved in visualisation--comes in even at the core of deductive, logical, formal reasoning, and in which that which later on, erronously or not, may appear 'obvious' at some time did not earlier seem at all self-evident--and, as we're constantly arguing --with very good reason indeed. For it was all completely wrong in essence. The Digonal Argument of Cantor is total nonsense if there is no set N={1,2,3,...} which behaves the way he assumes, and as I hope you and I together are beginning to see very clearly right now, that the set N is little but sloppy thinking. Let us do the sharp thinking here and get over to the next chapter and back into rule-of-thumb practical knowledge again--but armoured by a very significant 'logical humbling' as regards the

dealings of infinity in abstract thought. (For those who wish to her inklings of this insight as I have found them eg in 10th century Anselm's works, Archbishop of Canterbury, see eg super-model-theory.pdf in the library net folder as listed in the start of this book.)

It follows from [Q6] that:
[Q7]: Fulfilling the logical imagination request in [P] as regards this collection so that it shall, in some infinite sense, be regard as complete, and this completion indeed being necessary for it to be considered a coherent conceptual whole, involves that we invoke no extra parameter that can cut to pieces the symmetry we have discussed in [Q6], but rather than we imagine an indefinite process that is at every stage, also in its completed stage, having the same absolute symmetry.

From this it is easy to reach the next point, which is, of course, in direct contradiction to [P], which asserts, as part of the construction principle of the collection, that every member has finite properties:

[Q8]: As regards the collection mentioned in [P], the symmetry which exists when it is incomplete and finite, has not, and cannot be, removed when it is complete and infinite.

This is in contradiction to [P], because from [Q8] it follows that:

[Q9]: As regards the collection mentioned in [P], the visualization of the collection shows that self-reference is maintained even as it is reaching infinitude. And if it hasn't reached infinitude, it is not the collection mentioned in [P]. When it has reached infinitude, it has members that does not permit the characterisation that [P] required, namely that they are finite.

By Reductio ad Absurdum, we have negated [P]. There is no set of natural finite numbers only containing natural finite numbers and no other types of numbers, in a coherent formal understanding.

A short way of saying this is that "there is no such thing as a set {1, 2, 3, ...}" or, "there is no set of finite numbers" (ie, without upper and lower boundary set, so only finite members are part of the set). This means that if a textbook has in it a phrase like, Let N={1, 2, 3,...} and it goes on to use this set N, it is an incoherent textbook. If it doesn't see the issue about this formulation, it is likely to go all over the board in all other questions involving numbers.

This, though in more immature language, was dismissed as a master's thesis at the dept. of Cognitive Science at the University of Oslo when I delivered it there before I wrote the 2005 book on physics and numbers listed in the beginning of this book. They did not yield when I said that they should look at it again--though the professor there acknowledged that he saw my point and did not see how to object to it. I took that as a token that to such an institution, loyality to the words typically spoken may matter more than whether the words refers to coherent thought and reality. Fine with me, let them be that way; they are making themselves irrelevant. Life goes on and washes away such incoherent machines.

The G15 PMN was shaped, of course, after this insight, and so as to directly reflect an honoring of the sense of infinitude as something most typically beyond the grasp of thought and according to a philosophy inspired by all this such that, for instance, the use of such as the three dots or 'et cetera' is taken to be something by which we must formally be careful about so as not to invite an infinity in 'through the side-door' without having awareness of what we've done.

The pathways for new coherent thinking on numbers and infinitudes from this, I venture to say, is itself infinite. Nothing of it has begun for real in humanity as yet. Plenty of formal thinkers have been creative about infinities, but when this creativity is not grounded in a solid insight into the incoherence at core of formal thinking about numbers, it just gets a lot of funny big words and rather nonsensical new 'theorems'.

On the optimistic note, however, it can be added that exactly since this insight has not yet come into a realized form in common sense understanding of numbers, we have new solutions--entirely new solutions--ahead of us.

That is enough meditation for now. Now, out of the rain and into the glorious sunny day.

WHAT ARE COMPUTATIONAL NODE NETWORKS?

In a first-hand computer, which, as we have discussed before (in this book series, also) is 32-bit, ie, its numbers have a known upper as well as lower boundary, which is approximately plus minus two billion when discussed in term of our normal 10-digit number system, what goes on is all a question of algorithms and data. And algorithms are nothing but exact rules for pushing data around; and--though care must be applied by the human programmers making such arrangements--some of this data can be algorithms. So, for instance, it is by an algorithm that we can move an algorithm and its data from a storage position into a position in the computer's RAM where it can be performed. But if we use an algorithm to peek and poke into another algorithm, great care must be taken not to make a mess.

When algorithms, which are each exact, are going to handle something like steering a robot to wash the floor not too much and not too little, some of the data that they use reflect gradualness, probabilities and such. That is, of course, not to say that the digital computer has become analog and 'probabalistic'; it is only that the exact rules have been written in a way which thoughtfully reflects more fuzzy challenges and solutions.

The idea of an algorithm is of course a human concept and to some extent a 'mental projection' onto a machine that we have made. It is a way to describe a certain flow of electricity along certain lanes and to use such descriptions creatively to redirect some of the

configuration of this electronics. A click on a PC keyboard involves electrons and the portion of the computer we call RAM or Random Access Memory. To describe this as 'programming' means that it is part of a concerted effort to get these keyclicks to have certain patterns such that, when certain other patterns of the computer meet with them, they lead to a 'performance' or 'execution' of the program. These are descriptions, but, as it turns out, highly useful descriptions. There is a clear satisfaction, and rightfully so, for us human beings that we can make something hugely general in nature and equip it with what we call 'instructions' and leave it to itself and see that it performs all that as expected. There is a real sense of mastery in doing simplistic programming, a mastery that would not be the same in the PC politely asks you what it should do and appears pleased to receive an inaccurate description of it. We expect such delicious inaccuracies to be part of the rule of the day between living beings with minds; but it is a different set of expectations, and joys, in connecting with something utterly algorithmic.

In thinking about programs, we can evolve new concepts on top of 'algorithms' and 'data' to describe certain bundles of such, and encourage a certain type of programming to meet a vast range of situations that might otherwise appear a bit too complex to be handled by 'mere' algorithms. But let us keep in mind that these things are descriptions and not take them too seriously.

A very healthy concept, and useful, connecting to the old Latin concept of a knot or a bind, Nodus, is that of the node, and to consider a program to be a network of nodes. At once we can imagine lines between squares or between some imagined abstract concept to indicate a larger program structure. The 'lines' may be such as data which gives the address in RAM for each 'node'; each node may be an array with some warps to algorithms and some other numbers representing some idea the programmer has about what should be measured or what is to be done.

When we speak of a computational network of nodes, we usually speak of a situation, like robotic tasks, where the idea of probability is profuse and certainty a luxury; we also speak of a situation in which sequences of that which is to be done by the PC--and ultimately by the motors and other expressive aspects of the machinery--can vary according to the somewhat fuzzy features of the situation.

It is still of course all a question of algorithms and data, only that we now tend to line things more up as a matrix where each slot in the matrix contains some data and some algorithms and some more data that links various parts of the matrix; and a 'master algorithm' that works up and down the matrix to calculate what next is to be given computational time, CPU time.

First-hand computerized mentality, or FCM, has a general description as a sort of matrix, and, if you like, set of matrices, that you can use rather as you like--a kind of framework to do thinking over algorithms and data in a way that makes sense for making good robotic programs of just about any type; and the definitions of this framework in G15 PMN is of course part of what is called 'Third Foundation G15 PMN' and shown to be useful in showing features of quantum and relativity physics in the super-model-theory.pdf as well as shown to be practical and fairly easy to work with esp. through the G15 PMN FCM Spreadsheet program. It is this program we wish to use in perhaps most FCM programs, as a frontend, showing the most

important things and allowing input about the most important things. The spreadsheet is one set of 'nodes'--also called 'funds' or even 'foundries'--and the other node networks can be handled by the same main algorithm, a function called 'translucent', so that the computer flips back and forth between giving computational power to the spreadsheet and to whatever else the robot is all about.

'First-hand' means that we as human beings with minds have, as part of a broader 'understanding ethics', as a criterion that we can and do understand what is going on in the computer at mostly all levels also when we do robotics.

The FCM framework in G15 PMN Third Foundation is truly immensely open. It doesn't say that nodes have to be of only two or three categories or whatever. It doesn't say that links have to go only forward or up or back or down in the imagined network. It allows any 32-bit number to be given as 'level number' for each slot in the matrix; it has a number of open slots for 'links' and other slots for warps and data, including a number that can turn off and on the slot--that is to say, in how the main loop 'translucent' handles it.

So the FCM framework in G15 PMN allows you to think, plan and implement just about any computational node network you like, as long as it is realistic. Realistic involves also that you remember that a robot is context-bound, it is task-specific; there is no such thing as a 'general' robot.

In considering such as the range of tasks, and challenges, a domestic robot helping one or more human beings getting on with the pleasures of living and reducing the quantities of boring tasks, must do, we can easily begin to describe some of the main forms of nodes--but all the time bear in mind that these forms of nodes are but descriptions of nodes; and that a node is but a description of algorithm and data. But these descriptions, when leading to clear visualisations, good questions, and good answers, lead to rules of thumbs in how to make the proper algorithms--which in FCM robotics with G15 PMN also may mean to construct something around the the FCM Spreadsheet.

So, for instance, some nodes are more obviously about 'summing up features' of a situation, whether a situation presumably 'out there' in the real world eg as read in by cameras, or a situation in terms of what goes on inside RAM. These can can freely call 'matching nodes', but the distinction will work only approximately. Distinction relative to what? Well, we can call the main other type for 'task nodes'. So the task can be to clear up the room and a matching node can represent the degree to which it is at present messy. The task involves getting something meaningful done, in a meaningful time, that reduces the number representing the 'messiness' in that matching node.

Some tasks can be there in the sense of regulating the other nodes, providing healthy boundaries of action; we can call them 'ethics task nodes'. They can be considered to be 'top priority' in the sense that they may never be overridden by a practical task node, rather they can stop a practical task from being done if there is a threat to something of principal importance as measured by a matching node that the ethics node constantly refers to. The obvious examples are that a cleaning robot has no business running its wheels over people's feet or bumping into a plant or to rush ahead in case kids are playing on the floor. Here, the life-protecting ethics nodes are

higher-level tasks constantly performing a monitoring over the other tasks, and if there is an uncertainty which is significant as to whether the higher priorities are satisified, the robot should go to a rest state or whatever it takes to not override these priorities.

Some matching nodes are simply that--they match over some data eg delivered from a camera. But others require a sysmtematic building up of data over time, involving a number of actions, and so it is equally well described as a task node. The distinction must be soft.

Of task nodes, most of them, in a FCM program which is sophisticated, call on subtasks, or goals, in a way that is at least partially dependent on matching nodes, not only in order to be activated, but in order to tune them. The domestic robot may want to wash a cup in hot water, but this washing is different depending whether the cup is just lightly and recently used by such as water or coffee, or whether it was used to drink some yoghurt many hours ago. The intensity of washing must be coherent with a matching over it.

Yet in every robot there are essentially 'entrained actions' which are nought but motions of its engines or servos or whatever we call it. We can call these nodes--we can have, if we like, one node for each such entrained motion series--for tasks that are 'elementary actions'. The robot ultimately expresses itself in terms of a well-sorted sequence of elementary actions, or goals at the action level. But the more sophistication it is to its computational node network, the more the programmer has put into getting this to be finely tuned according to a possibly rather vast range of insights.

Now we must all the time go back to the point that the nodes are but slots in a matrix, slots which contain both algorithms and data, and that all descriptions are but descriptions--there are no absolute distinctions of types here, and if we do begin with absolute distinctions of types, you can bet that most of the programming later on will be all about superceding those distinctions. It is this knowing that goes into the FCM framework as we have designed it, and which not only asks, but requires, the human mind to express itself through the shapings of this network.

For instance, while 'elementary actions' sounds pretty definite, you can perfectly well make a full copy of all the nodes and run it, on command by some of the task-and-matching nodes, as a simulation. Here, the elementary actions, when performed, are put into a mode where the robotic motors aren't getting impulses, but instead there is a simulation of expected results of these elementary actions that feeds back to the more elementary matching nodes, instead of actual input from cameras. And as long as you keep your tongue straight in your mouth, you can have such a simulation call on another, and perhaps different, node network, which represents a simulation of some of the features of the likely behaviour of a being or object out there in the real world, such as a human being.

Say, for instance, the robot matches on the reality of 'kids playing at the floor' and at the same time it has, with some emphasis, been given the task of 'gentle washing of all the floor and this with effective nudges to all present to move aside a little bit' and ethics tasks making it absolutely clear that there must be no harm to anyone. And let's also imagine that the robot has a sound bit that it plays, with a nice little song about its intended washing of floor but the kids, having heard it before and considering the robot a kind but uninteresting

nuissance in the background, simply ignores it. The robot may match over the attitudes of these kids and it may have some strategies such as offering some rewards such as candy for moving a little bit over to the other side of the room; or pretending that a game is about to be played; or putting on music too loud for games to continue; and some other strategies like that. How does it select between this or that pathway to achieving the goal some other human being has given it, and within the ethics criterions? For this, it may have, if the human programmer has bothered to do all the work, a simulation capacity so that the robot can explore likely consequences of this and that action pathway, while taking a pause and not having any motoric output while the simulations go on.

Eventually it decides on a pathway, or that it is too risky and it folds up in a corner.

How are the simulations switched on? How are the results of a simulation of the other or others reported back to the main computational node network so that a task can act on the result? This is all up to the programmer--for any node can be anything--it can connect to camera or to simulated cameras; it can express in terms of elementary actions that actually go out as signals to robotic motors or just to nodes that contain some information about the typical expected result of doing such and such. The simulation is just another computational node network and, as we said, each such may call on some others to some levels--not too many, it must be first-hand understandable --and the result can be plugged into a high-level matching node in the main network by an algorithm in the simulation network--whether by a node that is vaguely of the 'matching' kind or one that is of the 'task' kind. Or, the main network, after having called on the simulation, may reach into the simulation and fetch the results from it-- after all, RAM is RAM and in the real world of algorithms and data, all is shareable. One can make rules of division of sections of RAM but only at the price of much work to go beyond those divisions later on when it's necessary.

So we see that the notion of computational network of nodes is highly fruitful, it is fun to think with, it lends itself to be a thought instrument for expressing, as programmers, a bit how we ourselves may ponder over a situation before, and during action, and in this sense, the programmer expresses his or her mind in the FCM program--without presuming that this expression 'has a mind of its own'.

HOW CAN TWO ROBOTS COLLABORATE?

The text you read now I wrote on B9edit, which is of course a G15 PMN program. To get anything done on a computer, we need to get algorithms and data; and so G15 PMN is a way to mould algorithms with elegance. The B9edit text processing algorithms handle the data of clicking at a keyboard. So even core programs have some connection with hardware, ie, something not just digital--like a keyboard. The hardware usually have its own little electronics, generally much simpler than the computer, but just enough to get its activity translated into digital-looking electricity pathways to which the computer can relate. The typed-in characters are in a way, "data", and usually the computer, maybe even the CPU itself, will have a special location where this data resides--and it is being automatically updated by the electronics as typing goes on. A request to the CPU by the program to fetch the most recently typed-in character will then make the data available to the algorithm and it can store it somewhere.

All this involves timing--the timing of the hardware, in this case the keyboard, has to relate to the timing of the algorithm; and the algorithm must distinguish between what is a visible character, like "A", code as Ascii with the number '65', and that which is a click eg on the F3 function key, which in G15 PMN, as relayed by the "KI" two-letter word function, has the number '284'. This F3 is of course used in B9edit to signal such as a storing of the text you just wrote to the disk.

So with the most essential types of computer programs we are handling what we can call 'collaboration' between elements of the computer. When we equip the computer with more hardware so as to make it into a robot, there are, of course, many more such instances of collaboration involved, to get the whole machinery to work. To program a robot to do tasks on its own is complicated enough; to program a robot so that it can do tasks together with another robot--also so that the twin robots can be considered to be two aspects of one larger robot concept--is of course yet more complicated. But it's good to know that the notion of collaboration has been called on from the very first computer programs we think about.

You notice, perhaps, that I do not feel inclined to treat the notion of collaboration with quotes, unlike if we say, for instance, 'The robot is "satisfied"' Any mentalistic or too-biological/lifelike word, when applied to a robot, should be put in quotes so that we do not derange the vision of the human being even if we work much with, and talk much about, robots, in our daily life.

And so, in our FCM works, we do not say that the robot engages in recognition or knowing. Rather, we might say, for instance, "This is a situation where the robot matches over quickly and well" And, "The robot has been well entrained" (We do not say that the robot has been trained without in case using quotes around that word.) And certainly, there is no "training of algorithms" An algorithm is an absolute unit which pr definition has no component whether of learning or training. Rather, an algorithm can be of the "entraining" kind so that it can match over data and gather more data through these matchings so that it can offer smartly built data that allows quick and relevant matchings over similar data later on. This is a whole other class than living human being activity; and we should be generous enough in our word-life that we have an alternate set of concepts and do not engage in tacky word use relative to robots, and/or forget the quotes.

Having said this, we do permit some more obviously 'behavioural' terms and a few near the psychological realm to be used with regard to computers. For instance, easily we may say of a computer that it works, and we do not feel inclined to wiggle the fingers to indicate that this was a metaphorically intended word that should be quoted. We also speak of the computer's memory, because the word conveniently into the talk around computers in the early days when they were first being built and it felt cosy and natural enough and there are plenty of examples of use of the term that goes beyond the reference to a particular living being--eg, 'They erected a statue in memory of the hero! And if we can say of a single computer that it works, then we are certainly at liberty to say of two computers, with or without robotic hardware extensions, that they can work together; and to 'work together' is one of the definition of the verb 'collaborate'.

Now a robot, when programmed, may at times just be a concept in the programmer's mind--because there may be months of software preparation work that requires thinking and walking and wondering and typing and musing over example programs, all the time visualizing that this will be part of a full active robot doing safe and good things. Gradually, the robot, perhaps initially controlled by a PC which is in the side of the room and through long and heavy cables--the PC having a giant screen, a huge and comfortable keyboard and is situated on top of a desk--is getting to move and relate inputs through cameras and the FCM program is built up. At some point, perhaps it is all bundled up into one unit, a robot with its own little computer (with or without radio signals to a computer nearby doing steering), its own little screen perhaps up front (but out of the way relative to the practical things the robot must do with its robotic arm), and a little keyboard and an optional mouse there as well. This little computer may overheat if one does full graphics all the time, and so there is a version of G15 PMN (which I developed this year) which is called Batch Graphics, shortened into G15BG.

In G15BG it takes either something such as a push on the ENTER button of the keyboard, or a particular, and new, G15 CPU instruction that only make sense in G15BG, to show any graphics on the screen. Otherwise the screen does not update and is in a rather passive state. This makes grand sense for robots, for they have so many other ways of 'expressing themselves' than screen output. They have got an arm or more; they have got wheels; they are huge; they move about; and a screen is just one item among many.

However, a screen has one giant advantage over all other output mechanisms for the robot: it is native to G15 PMN, it is an action of pure leisure, a quantum of solace we might say, to put out something on the screen.

So when we get two robots to collaborate, and we need good timing, we need a way to exchange cues--what in traditional electronics communication terminology is typically called a 'handshake'--and for that, screen output by one robot as captured in one of the cameras of the other robot may be a cardinal way.

In First-Hand Computerised Mentality, and indeed in any first-hand programming, the emphasis is on the human visible and on human understanding. If something is done via a screen that you as a human being can look at even as you get your program to decode that screenoutput via the program on a twin robot as it caught it in a camera,

then we are in a situation in which the entire situation is more human-readable than if, say, radio waves at 1000 MHz were used. Light involves, apart from its philosophical mysteries, radio waves (as we can call them, although classical physics insists on the phrase 'electromagnetic waves') involves not merely such as the typical medium waveband 1000 MHz but, let's say if we want green light, 555,000 times as fast vibration. That's 555,000,000 MHz.

So if you are to program twin robots to do certain tasks together in a pre-programmed way, one way you can make them do the necessary handshakes are by means of radio waves--but if you can select 555,000,000 MHz instead of 1000 MHz for these waves, that's what you do, because 555,000,000 MHz is a green light on the screen whereas 1000 MHz is invisible and merely creates a scratching sound in your AM radio (or music, if you so please).

But with small screens on vehicles with tracked wheels, and cameras only here and there, such an arrangement makes most sense if we limit the complexity of the cues to, let's say, a letter or a digit shown over the whole screen one at a time, and indeed for several seconds each time. Essentially, what you want in many cases is, "let's do step 3; give me a thumbs-up when your part is done", and similar communication concepts.

For instance, with two tracked-wheels robot with one flexible long arm on top of each, and variable height, screen in front of them, and cameras here and there so that the programmer can select the most appropriate camera to check for handshake cues, the programmer could choose to get the robots to align themselves in a vaguely 'face-to-face' manner with the task between them. Maybe they're going to lift something too heavy for one of them and so one arm goes to hold on one side, the other arm, of the twin robot, goes to the other side, and typically one of them takes the lead and awaits for cues from the other that it has done as requested. The main robot waits for what is essentially a "nod" from the other robot that it has a firm enough grip, but it being the main robot doesn't send that signal the other way. It simply checks its own grip. When it has checked and confirmed that it has a good grip, and also got the equivalent of a thumbs-up from the other on this, it signals, "Okay, now lift", and it might do so by flashing a number or letter on the screen.

Now in this, for robots, fuzzy domain of the real world and its many interpretation possibilities through ambigious input in low-res through typical monochrome cameras, there may be a sudden matching that the grip must be improved or that the task may have to be done in some completely different manner. That can be the task of either of the twins to signal to each other.

A experienced computer programmer, who have done many types of work on a PC but not, perhaps, yet on a robot, may notice a similarity to this situation and that of analyzing perhaps partially restored disks or getting to grips with plucking some information from a database made by a program no longer in existence. In such situations, there is a lot of available data but very little of it comes with labels attached explaining anything of what's what. So something has got to be tried, and it may take a bit of time to see whether something meaningful builds up in terms of a particular take on the old magnetic floppy disk or whatever it is; and then perhaps something else

has got to be tried. The programmer, and through the program, _looks for cues_ in the data. These cues don't necessarily stand out very clearly. One has got to spot them. And one has got to interpret them. And gradually one can build up a way to do this so that, after sweating it out with 20 hours programming ignoring dinner etc, the program works as a dream and the programmer can dream sweetly on the office floor, hearing cheerful clicking sounds from the obedient computer, running his or her genius program.

So, fetching cues from data, and, in the case of twin robots, doing handshakes--these are obviously part of the natural and, we might say, necessary standard reportoire or smorgasboard of the collaborative robot programmer. Getting this right is essential.

Getting the robot to collaborate right with human beings is also essential--and infinitely more complex. But absolutely everything you learn when you program twin robots to collaborate in a preprogrammed way can, with suitable sensitivity, be conceptual input to the much vaster question of how the robots collaborate with living beings, and in particular human beings. And relevant areas to explore includes also how two or more robots can do improvised collaboration (not an easy task).

So we see that the theme of collaboration keeps on expanding as we explore it. How, for instance, do you get twin robots which have several preprogrammed collaborative tasks readied inside them for a particular situation, to get going with their collaboration? The obvious and easy answer in many situations is, click on them, dude. Let there be a menu on each little screen and let the preprogrammed collaboration type be selected, one for the main twin, and one for the other twin--what we might call the slave twin.

This is obviously a theme we'll touch on regularly.

In some of the forthcoming chapters we'll dig deep into the Third Foundation G15 PMN code with focus on the definition of the framework for FCM there, in particular the matrix of the FCM nodes or funds, and do the type of software thinking/planning/writing that must be at the foundation of every good FCM program. Not all of it will be very readable unless you have some years with G15 PMN behind you, but it can help you to master G15 PMN if you patiently absorb this while not trying to make full sense of everything at first.

GETTING DEEPLY PERSONAL: HOW MAYBE THE BRAIN WORKS

The past four years, more or less, I have had a personal experience that, for fear it dims in my memory now that it is, with each month, more and more a thing of the past, will have to be written just now, and this is clearly the right place for it--for it concerns an aspect of the quest to unleash thinking.

My late father had a brilliant brain, and he was a fast talker, a good writer, and admired thinker, on themes such as dialogue, communication, simulation, emotional engagement, empathy, and steering away from being submerged to the limited understanding by others (model monopoly or model power theory). He also collaborated with Kristen Nygaard (whom I later befriended) at a time when Nygaard, together with Ole-Johan Dahl, at Regnecentralen, Oslo, in the 1960s, were working out and implementing the concepts of object-oriented programming. The first form of Simula, in which father simulated voting behaviour, didn't have hierarchical object/class orientation. A later version, Simula67, had it, and it led to the development of Bjarne Stroestrup's C++ and then to such as Objective Lisp and Smalltalk and Java and all the rest of it, up until Xerox Parc with frames and mouse, Apple and Windows operating system and the hundreds of programming language shaped in this paradigm that Dahl and Nygaard specified in utter detail in the 1960s--this is well-documented history.

My father also had great emotional energy and his methods of arguing could slide quickly, sometimes too quickly, to an angry tone when he got impatient; but as my sister Kristin pointed out to me some weeks ago--he always 'made up'--he got back to you and got it right again; he didn't slam doors and keep them shut. He was always getting back to the generous mood, and if he had to apologize for his tone, he did so, readily.

Having nearly a dozen books behind him, and hundreds of journal-published articles, and his thoughts on his theory of the Virtual Other published by nothing less than the top-prestigious Cambridge University Press in a book he edited after running a seminar on childhood development at the Norwegian Academy of Science and Letters, it came as a shock to him that his memory, esp. after the death of my mother, Else Reusch Braten, began being less easily available to him than in his younger days. When he was nearing ninety, this issue was more pressing than ever; however he wasn't confused--he was simply more often silent when he had trouble remembering, trusting that more presence of mind would come around perhaps already in the evening. And it mostly always did. My late father, the and author Stein Braten, will have his last book published around when he would have reached his 90th birthday, at November 3, 2024.

Earlier on, his beloved Else, my beautiful mother, was a constant source of inspiration and harmony for his writing --and so, in a natural way, his mindful intellectual and dialogic teaching and research was entwined with his utterly important companionship with her, his wife.

My role in his book-writing over the past four years was that of facilitating, as best I could, the conditions for his writing to go ahead more or as s as before, though his last book had been published in 2013, and he almost seemed to have concluded he couldn't make yet another. And for those who find it incredible that a person who, say, during lunch-time, can have a hard time remembering many trivial daily-life things, can express philosophical remarks over scientific studies with exquisite sophistication in his writing time during the evening, I can say: this was my daily experience of him. Over the past years, he increasingly needed help with trival tasks; but when he was in the book-writing mood, --and I will tell now here a bit about it--he got as it were 'online' with the best parts of his intellect and was in possession of both intellectual capacities of reflection and memory

surpassing that of the other parts of the day to a significant extent.

I also wish to get these recollections noted down, so as to give a message to those who think 'measuring cognitive ability' is something fairly simple, a matter of exposing a person what they regard as some 'cognitive tests' and noting how well and how fast the person did what they think a high-cognitive person is supposed to do. To think, to cognate, to have cognition cannot as such be measured in the way some of those cognitive-test-makers imply. I have my own interpretation of what the role of the brain is relative to the larger concept of what we call the 'mind'. But in any case, I have here empirical knowledge, albeit anecdotal, but consistently and over a long time with a person who had unique intellectual capacities at a very old age but yet required a set of circumstances to unleash it. Now what are these circumstances.

In contrast to my reflections over such as programming, being systematic about reflecting over the theme in this chapter comes only at an effort for me, but here is a go:

First, my father needed to have much of the practicalities of the day and its main meal behind him. He also needed relaxation in front of the TV with his most-loved films. These were in general fairly harmonious and yet with grand perspectives--as his fascination for the hieroglyphs and the ancient Egyptian cultures--or having unique beauty and good flow in communication, as a BBC movie over the fictional character Emma, based on a Jane Austen novel, starring Romola Garai, from 2009; or the life biography of the Norwegian explorer and scientist Thor Heyerdahl; and other films, including with Meryl Streep. I hired young freelance nurses and homeworkers to do something together with him during some of the day, and these girls had a wonderful effect on him and often seemed to contribute directly to his enthusiasm to write more.

After spending quality time in front of the TV, he would qext go to a couch, in front of his writing station--his PC. Here, he would, with my assistance, put on some Shiatsu 'massage machines' to easen bloodflow. All the while, the temperature would be many degrees higher than what's typical in Norway: without this, apparently, the bloodflow would not be adequate to allow him to get into the writing form after the use of massage machines. I have since read studies on certain diease-fighting features of the human body activated by a higher bodily temperature which in turn can be activated by such as higher temperature around a person in the room; and I myself always find intellectual work much easier in a high temperature room, so in that sense there was a compatibility of approach. An intellectual working temperature 30 degree Celicus, or about 85 degrees Farenheit, is only possible though given certain conditions of bodily leisure, adequate nonalcoholic beverages (for alcohol drains the body of much-needed humidity and so for many can make heat painful), and also a genetic aptitude for high temperatures (something that, according to widely available statistics, is more rare amongst those with red hair, for inscrutable DNA reasons).

While taking it easy horisontally, and--for him, the all-essential jazz put on, we would go through a recent research report relevant for his work and thinking, typically found on internet and printed out some hours earlier.

Then, seating himself in front of the PC, font turned on gradually larger the more he approached 90, and with a caffeine-enriched quality drink on the table, he would

come with as deep and sophisticated insights, eloquently formulated, as I've ever heard from him during my growing-up. Sometimes, though, with a slowness speaking of his old age, and often exhaustion would set in just a few lines further in his manuscript. So a small book, just more than a pamphlet, took him around four years to complete. But what a book!

Nearing completion of this book, and--as it turned out, of Bratens own life;-a researcher and professor at the University of Oslo, used to using Stein Braten's books in her teaching at higher-level psychology, dr. Helene Amundsen Nissen-Lie, wrote a preface for his little book. She described there the perceived importance of his work over half a century and more, and said of this book that it sums up his most important insights over a long career in an almost haiku-like style. One of the happiest and finest moments of father's last year was to experience this preface read up aloud with both his daughters and all his family gathered around him. Some months later a lung infection led to hospitalization and after some days he left us, peacefully, knowing that his book had been completed.

Father always insisted that those who spin theories not having 'licked empirical dust' tend to go wierd. He himself had worked in marketing before he theorized over marketing; he had filmed mother-infant interaction before he theorized over the inborn capacities of infants for advanced also emotional understanding of others--and so on.

Let me also say that one of the things that seemed to be vital both for father's health and for his experience of quality of life was his monthly or bi-monthly visits, with me, to his qualified wise doctor, who took an active interest in the progression of father's writing.

Now the experience I have of father in late age told me, as nothing I have ever experienced before nor read about, that--put bluntly--the brain is not just the brain. And capacity is not just capacity. Cognition is not just cognition.

Nor is 'unlocking' the capacities of the brain merely a question of 'reprogramming it' nor 'injecting it' with this or that substance. All that is way too mechanical.

Nor is it merely a question of giving the body a little bit more exercise; it is not merely a question of finding the right music and intellectual stimulation; it is not merely a question of sitting still and meditating.

Whatever it takes, it takes all that, and more--including the sense of 'having a function'. As one of his teachers, the philosophy professor Arne Naess once said in a radio interview in his own old age--as I recall--"Nothing is worse than to feel that one is of no use to anybody; that one is just a bother." If that understanding has got into how we shape kindergartens and everything from then, more people could have shaped themselves so that also in old age, where perhaps they cannot even move by themselves, so that it would be easier for more to experience their actual value and keep on unfolding it until, just about, the last breath.

But what then when memories pass away? A section of the medical community may then be eager to pass judgement over the brain and give statistics for just how common it is to

forget such and such when above such and such age and recommend a little more exercise and salads and such. These are the same people who happily take people's right to decide over themselves away from those who they do not deem fit--soldifying the idea that their human existence is, even formally, just a burden from a certain point on. Worse, they may feed a person who is forgetting much some type of stimulants that perhaps in most cases only lead to chaos in the brain: yes, it gets more active, but it gets active in an incoherent way because their chemical supplement hasn't in it true human intelligence. Their supplement is like turning up the turbo injection power of an engine that hasn't got enough oil to keep it smoothly running. Had they left it slow-running, it might have retained its coherence; there may be ways of increasing the coherence; and then--when the brain itself knows that it makes sense, the intensity can go up for a little while.

For what is worse for a person who does not remember that the brain is getting a sort of stimulant that makes it begin to construct reality by plucking disjoint memories and making up stories which make no sense?

A person who does not remember in daily life may still have coherence, and may still have by far most memories intact. If the person is very quiet and claims not to remember, all that may be true--at one level. But it may also be the 'brain on its own' giving a report. And the very same person, just some hours later, giving a fullness of conditions satisfied--vaguely like the idea of Joseph Campbell that each should 'draw a mythic circle' around himself or herself daily as a ritual and go into a transcended state of consciousness--can become almost "oceanically present" The vibrancy of the person can reach into the furtherst depths. The presence of memories of the meaningful kind, and with perfect clarity, may emerge in an individual who, due to old age, barely can do the simplest things some hours earlier and appear to all extent to be mentally 'gone'.

And so, I have an interpretation of this, and it is, of course, massively in tune with my own take on the interactions between manifest matter and the depth-energies that are more subtle and which somehow connect to the quantum. My direct experience of my father going from his 'day-mood' to his 'writing-mood' was that of a nearly absolute transformation: the first is merely the brain; the second is the brain tuned into something much vaster--his self, his soul, his mind. And the happy-go-lucky officers of the harshly mechanical medicine world lives in complete denial of this to me rather obvious fact: that the human brain is just a ripple compared to the larger thing, the human mind, and the task of the former is to be an instrument of the latter, and the key to that is no medicine but a lifestyle such that I just gave example of --which includes, and luckily for my father he had that, a passion to unfold beautiful thoughts to the world.

That smartness that is so self-obsessed with its own smartness that it sees nothing but smartness and the lack of it, has no smartness. That smartness-which-is-no-smartness is summed up in two of the hottest, and stupidest, concepts on the planet: AI, and IQ. The present-day 'believers in AI' have a fetish so strong on algorithm that they aren't even capable of calling an

algorithm by the word 'algorithm'--they say anything but, 'super-intelligence', 'generative intelligence', 'God'. Their machineries may fool a section of the population adequately that this section says, "These machines have mind"; but it is all a smartness that lacks the depth and humour and reality-touch and empathy that the human mind and its intelligence, its genuine intelligence, can have to a degree that cannot be measured, and which is genuinely infinite in my opinion. And those who then set up 'measurements of intelligence', or 'intelligence quotients'--they are merely the foot soldiers of Newton--the hardest, least fascinating aspect of Newton, that is (for Newton in later years was far more far-reaching, being, amongst other things, one who delved into alchemy). They are fascinated by the view of the universe and the human being as an advanced form of clockwork, and they make a filter so that they cannot anymore perceive neither mind nor intelligence, and it is called, 'intelligence quotient'. They put their mediocre numbers on people, thinking they have encapsulated thought; but all they have encapsulated is their own stupidity, manifested in their petty schemes by which they attempt to measure the human being.

So the brain is not just the brain. The human personality is something infinite and the brain is more finite; and the pathway from the little mind of the brain on its own to the full human self with powerful personality features enabled such as an empathy that comes along with intellectual intuition and its own memory requires the art of regenerating the fullness of coherence in daily life, an approach to meaning and meaningfulness, and to both work and relationship; with the mechanisms of the body not crudely stimulated to achieve false intensities but with the patience that comes from honoring the necessary unknowingness we must have about the human brain/mind.

It is always in this light my notion of FCM should be understood: that the vastness of the human mind, also, of course, that of the programmer, can express itself--not just as a wonderful novel or painting or piece of music, but also as a program, and in that expression something of that mentality persists; but the priority is always the living mind and it must not be confused with any such crystallized expression as an algorithm. In entirely other words, which I used fairly much in an earlier phase of exploring this, the mind is The Uncomputer.

And, as we saw in the earlier chapter in this volume when we explored infinities, you can, as it were, go through the rabbit-hole and see numbers as something intensely and vibrantly other than mere mechanical units. A finite number is, as it were, an algorithmic-like expression--a crystallization--but that which crystallizes it is beyond all finitudes. The numbers are as music to the mind, but it takes the human mind to perceive them as such--not as static abstractions, but as commentaries on the pulsation of the mind, which is infinite in nature, in life; a life that has its own natural inborn thirst towards ever-more creative coherence (confer the PMW principle in Super Model Theory; see links in the start of the book to the texts about this).

The brain is not a machine and the mind is not merely the brain. Nor is the brain merely a 'quantum machine'. As

Stuart Hameroff, brain researcher, likes to say, it may well be a quantum orchestra. We go for that, if by 'orchestra' we mean jam session like, not regulated.

In the decades to come, big tech companies, dominating the planet with their fake-mind algorithms, must make a decision: do they wish to push tech away from having the role it has today, which is to be points of communication between human beings, and become merely a tool of knowledge and dry commerce and in which anything personal is merely an imitation? If so, the machines will get boring and people will go to the parks and to the cafees, rather than risk loosing face to their algorithm of a smile. And perhaps it is just as well: only that we cannot build societies on fakes. For my part, I suggest: let us both have machines, and cafees and parks and beaches and all the other analogue genuine meeting-points of human beings, and human beings and animals and nature; but then the machines must not be made sophisticated in ways that could cause harm to the immediacy of the I-Thou nature of natural society. Humanity deserves better tech than over-glamorized "AI" products, products that with utter ease can become instruments of worse types of totalitarian societies than anything seen in history. The ethical programmer decides that no further development along the lines indicated as "AI" should go on and that existing devices powered on those premises should be, to that extent, turned off; and politicians must make rules of this sort; and companies trying other pathways must feel the strength of anti-AI laws coming down on them.

As Frank Herbert in his Dune novel starts out by indicating: a task of humanity is to be masters, also of technology, and all else requires the fiercest of revolutions.

With FCM we're doing our quiet revolution. Let's proceed to some hard-core number thinking, laying out how we wish to configure the FCM funds in our upcoming computational node networks!

FIRST SKETCH OF A MORE CONCRETE TASK/MATCH NODE NET

This is a chapter in which the writing that a programmer typically will do in privacy, notes that are not really, usually, intended to survive the program, are being made so as to construct a program. When the intent is to throw the notes away as soon as the program is complete, tested and running smooth like silk, the programmer will easily allow himself or herself the use of incomplete sentences, abbreviations that only make sense if explained but they are not explained, and jumps in the construction without clarifying clearly that the earlier notes are considered obsolate at some point. Here, I try to walk a middle-ground between the two extremes of doing such local sketchy explorative partially meaningful writing, and an explanatory writing. But as long as the program under the particular consideration remains to be done, it is always possible that, when the program is being made, a totally new insight--or many of them--arises as to what one really needs to do, which may overturn at least bits of the construction. My experience at this point allows me to emulate in mind the upcoming program with some more clarity than otherwise, I believe.

Obviously, robotic hardware is here in my office but the concepts deserve to be clarified, and the program made to work at a conceptual level first, and then, in the completing volume, number 5, we will bind this fully together and--a promise--that volume won't be published before the robot is strolling around in my house, not wheeling over my toes and not pushing chairs around, but doing some useful things that actually are helpful and which involves active camera use at least in part of the tasks.

The robot? Robots.

For by 'the robot' I mean twin robots. So collaborative tasks are going to be programmed. But as we have seen, the concept of collaboration, at least in a general sense, more or less runs through the whole notion of programming, any programming, from the very start. With the ideas of handshake through graphical screen letters and such, we have enough as a start.

If you have a printer, in order to make most sense of the upcoming notes, print out the page in G15 app number 3,333,333, ie, the Third Foundation app, that shows the FCM node structure. When you mount that app, up comes the start menu at H:1, and it gives you a direct link to the card like F:2232. You will find the same overview card in any of the apps that contains the Third Foundation but if they have been even slightly extended, for instance with new two-letter words, the card number will generally be a higher one. In such a case the function 'scan' will tell you where it is: type in the word 'scan' and type in eg,

ten triplets

and also type in f1 or wherever the first card is, and a number like 99999--anything big enough to cover a big range--and it will come up with the card, and you can type, of course,

car

in order to see just that card.

At that card, we're told such as that the first position, which is #0, is the level number. The level number, as you might recall, tells roughly of the execution sequence of the nodes in the network. A higher level number generally means tht it is performed later in each loop cycle. Then comes its name--and the name is hugely useful for it reduces the need to memorize the numbers of the nodes when you need to link them to each

other, and linking nodes to one another is one of the things one does pretty much when we do FCM. There is a routine to convert the name to the node number and then the node number can be put into the array that has the links from one node to other nodes. But by looking at the program that sets up the nodes, the name will be visible--even if in many cases only used during compile-time of the program. (Though the name-search is fast enough that it can and sometimes should be used while running the node network as well.)

At the card we further see that the areas of links start at position# 50, and there can be maximum 100 of them, and the quantity of them goes into position# 47. My mnemonics for numbers is that 47 often refers to robotics. So it is part of a sense of numbers to design it this way.

Each node has ten triplets. Now what that term means in G15 PMN FCM is that there are ten times three free positions for putting algorithms and a little bit data to action with each node, and it is done in this way:
The first value can be anything you like,
the second value is the number of an fnact, ie, the number in the array of warps to functions that are driving the node network and which is different for each FCM net,
the third value can also be anything you like.
The first value of the first triplet is at position# 10 in the 150-number size fund or foundry, and it is typically used for something of signal importance about the node.

Briefly, the rest of the positions are these:
#40-#46, free numbers, used at your leisure, called in the program for Luxury numbers.
#48 'Top priority?' Starting with this volume, we let a range of coordinating nodes all be called 'high priority' but reserve the phrase 'top priority' for the what we might call 'the absolute necessary constraints. (Note that the 3rd Foundation calls #48 just 'high priority'.)
'Ethics nodes'--necessities that must all the time be regarded as more important than any particular task--such as not to bump into a living being or a piece of furniture when washing the floor--there is the number '1' here, which, in G15 PMN terms, inspired by the infinity theorem on natural numbers, is of course called DANCE. For all other nodes, the number is BASIS (ie, zero). No lower priority node can turn off the top priority setting of a top-priority node--that is our programming policy. But you yourself as programmer must be the ethical person who takes responsibility to carry this through. The computer programming language is, and must be (in order to be, indeed, a proper programming language) absolutely obedient to you. But it also means--watch it, when you set up top priority nodes. They really must be made with sensitivity, for they concern also how well versus how not-well a robot matches over its environment. If a top-priority node is associated with a low-quality measurement of the situation, the robot may enforce incoherence.

We're going to make a decision here, that follows from our earlier chapter on what computational node networks are all about, and which is coherent with the excitingly open FCM framework as laid out in the G15 PMN Third Foundation. Every bit of this framework has as its foundation what I took to be a carefully checked intuition. There is no tool of logic that can foresee all design possibilities and challenges: and so the best you can hope when you are

designing something within a designed framework is that the designer, or designers, had adequate intuition. This many years after the conception of the FCM it is clear that the intuition is a genius one, if I have to say it myself :)

Here's the decision: we'll do an MT style of nodes; and by that abbreviation, I mean: every node will be BOTH a matching node AND a task node. We're talking of each node having the full set of assigned variables and possibilities associated with both types of things--both the summary of features as analyzed by the computer over the inputs to the robot, and the performance of motoric actions usually through a range of subtasks. Now this decision doesn't mean every FCM in the future will follow this pathway; but the book is meant to show an example, and hopefully a very good example indeed, of how it might be done. There is no absolute law that says that a node must be 150 32-bit numbers either; it is just that it turns out to be eminently useful in a vast number of cases; but we can imagine cases where different-sized nodes are used.

I need a breathing-space in terms of a coffee and a minute or five before writing on, and, true to the jam session like writing which hopefully you the reader, now or at a later point, engages in, I will shoot in here a meta-physical perspective on how G15 PMN might power anything. 'Meta' in the sense that goes back to a labelling of Aristoteles' books--whether by himself or his students, for most of his works are the result of their note-taking while he was walking around with his lovers and pupils (and for which there were no distinction in the Greek society, remember)--is a word that means something like 'after' or 'beyond' and in the sense of 'meta-physics' we're talking: let's not merely think of the particular forms inside nature, inside what is 'born' (physis), but let us stretch our gaze up and to the horizon and grasp the whole and consider the beyondness, such as what lies beyond all this. Metaphysics, in other words, is an attempt by mortal human beings to perceive, in glimpses, something of the view of the higher beings at Olympus, --the olympic perspective (not to be confused with the adoptation in modern society of this in a sports context; however much the Greeks were obsessed with sports, indeed with nude sports).

Now--and this is in the Super Model Theory text which is also at K33 in the 3,333,333 app: first-handedness may be the criterion for algorithm not just 'here' in manifest reality, but also in subtle reality. If bishop George Berkeley, who imagined that God daydreams the world into existence and it persists due to the sophisticated and loving persistence of God's ongoing day-fdreaming, has some right in his opinion (from classical Western philosophy, although other parts of the planet, including the Sanskrit cultures, have their versions going much further back), then we might as well assume that God not only day-dreams matter into being, but--and indeed prior to that--all possible angels, or muses, as well as all possible helpful machinery, including first-hand computers --indeed 32-bit computers, because if mind at essence levels is at least fairly much like experienced mind at the manifest level, then when 32-bit makes sense in terms of number sizes here, it might also make sense there. All what I have so far said is part of the Super Model Theory.

In it, all matter is founded on FCM-like node networks, and all of these have algorithms; but nothing of this is performing on its own, but rather has concious, living steering and guidance the way living programmers and interactors with computers and robots up here on the manifest level. The muses, in other words, are master programmers. This is trivially true if you have read all my works.

And so, how can a universe so vast it is completely staggering with super-complex galaxies again containing ultra-complex planets again containing mega-complex plants --and, where we are, also fundamentally super-sophisticated human beings--be run by a 32-bit computer? Answer, by a network of them so vast that all networks we have ever seen dwindles completely in comparison. And to keep the numbers 32-bit, that network must have many features of a hierarchy. So, to the coffee.

Yes, all right, the jam writing session goes on. So we're now throwing forth projections about how a simple FCM program could be made that, despite simplicity, nevertheless has adequate complexity to handle a vast range of pretty tough robotics challenges. Not all of them, but a pretty vast range. We do not lay down how FCM should be done at all, but merely show how it can be done.

So, every node can be having features of matching--such as 'degree to which cups are washed up'--and features of task--such as, 'wash this cup'. The programmer decides whether to combine matching and task in one node, or not; but every node is prepared for both. That's how we propose this application within the FCM framework as included in the Third Foundation is being used; and we propose that's a pretty good idea quite often generally, as a rule of thumb.

Vaguely, doing a task involves also matchings--when we are talking any higher-level task and not merely the concrete motoric tasks such as 'lift-the-arm'--and so doing a task may typically require a bit more info than merely doing a matching. However, doing a matching may often only be possible by doing tasks. So the programmer's mind must be honored, and given room to formulate all this. But here's a way it can be structured:

Let the first triplet, of the ten, be dedicated to matchings, and indeed the main value, in permille, in other words, in a number from 0 to 1000, tell us the degree to which an either/or feature is the case. The 2nd value in 1st triplet can be function to set this. (When a function number is set to basis it means that there isn't a function there, but it can mean that the value is set by another function, whether in the same node or from a different node or whatever; this is all up to the programmer to structurize.) The third value should say how likely this is, again in a number from 0 to 1000. The first value of the second triplet gives a value, this time theme-specific, for what the matching is, when it is not an either-or case. (For quantum-like measurements, that could be a rotation number; see Super Model theory).

So, if the matching is over a yes/no question, like, "Is it messy here?", then the main value will say how much messy it seems to be, the extra value (the third value of the first triplet is often called 'extra value') will say

what sort of probability is assigned to this matching--from 0 (no trust) to 1000 (total trust). And the function will do what it takes to get this matching worked out. In the case of a robot doing manufactoring work, it may want not simply 'are there more items to put in the box' but 'how many items are there to be put in boxes'. And in such a case, we put the quantity of items in the first value of the second triplet. The one obvious thing we should also mention at first connected to matching is the clock of the matching. That can conveniently be decisecond, ie, one tenth of a second. As for good robotic programming, usually, things don't go all that fast: though in special cases like an FCM program running a train in a tunnel a more appropriate measurement might be hundredth of a seconds--so the FCM program more quickly can get at real significant matchings that may influence the security of human beings. The idea of putting algorithms to work for driving people through city streets in cars in insanity; because algorithms are context-dependent but city streets are context-changing by the very nature of human society.

So let's put the decisecond number into the third value of the second triplet. Now let's say a word about timing and clocks and programs: a G15 PMN program is defined as something which runs on a G15 Personal Computer, powered by a G15 chip (or a practical virtual implementation on this on a somewhat different platform), and for stability, it is recommended that the PC is rebooted twice a day, and more often when complex and demanding programs have been performed. That is also in alignment, of course, with the notion of setting boundaries--not just for numbers in theory, but also for what numbers we need when timing a program. Deciseconds as "timestamp" can only work in a 32-bit context if reboots are often enough that the numbers don't overflow. Reboots are not sign of weakness of the system, anymore than sleep is sign of weakness of a human being. These are healthy ingredients of renewal and essential for coherence of the machinery.

From here, let's consider that we now shape the task part of the node. In using some of the first triplets for the matching aspects, let's say that we use the fifth triplet and onwards for the task aspects. If you make an overview over the positions of each of the ten triplets, starting at position# 10 in the fund, starting with 10-12 for the first triplet, and completing with 37-39 for the tenth triplet, you'll see that the fifth triplet is 22-24. At position 22, then, we have the first value of the fifth triplet. Let us suggest now that this number is used by the task-node itself to keep track of how far it has got with steps. In upcoming chapter we list more fields for our FCM network: at position# 21 we have a 'soft' activation field, in contrast to the more 'hard' activation at pos# 49.

When we wish a high priority task to 'figure out' which lower priority task--ie, substask--to do actually, it may walk through the steps of a subtask rather 'manually' and summarize match numbers for each step somehow. The first and second triplet values beginning with the fifth triplet can be used for this sort of thing; we need it already in the app we're making here in this volume.

The function can relate to the task part of the links in the foundry, ie links to the subtasks, one at a time, to be performed. However it may be much more involved than

that: there may be a need to constantly check against various matchings in order for the subtasks to go ahead; and to get some matchings sorted out, the robot may have to change its direction to fixate its camera on some object and so a whole range of subtasks may be invovled to sort out a significant matchings before one subtask can be considered completed and the next one can begin.

Without assigning positions yet to every number, we also need a decisecond for when the task begun; a decisecond to indicate when it last called on such as a subtask; and a decisecond to indicate when the present task was completed. And indeed this particular field may be the one that we can use to check whether the task has been completed or not, when 'seen' from another function. A basis value here indicates that the task, supposing it has started, is still unfolding; while a non-basis value here tells other functions that it has completed. When the robot has got to have a task done within, say, the past 3 seconds, it nullfies this fields and initiates the task. And a way to initiate the task can be--in addition to ensuring that the main 'is active?' field of the foundary is set to DANCE, ie, 1;-to set the task number to dance. In other words--and this I see now that we write about it--the number of the task being done is the number of the task ABOUT to be done. Ie, when we want something done via a node, set its present task number to 1. Maybe there is only one task to be done through that node; maybe there are more; but in any case, that's how to activate it.

GETTING CONCRETE ABOUT WORKING WITH THE 14 FCM MATCHNUMS

In this book, we will, as they say, play in shallow water: the FCM we do here, with the node networks, are working on RAM, it's all software, so that we can concentrate on the concepts before we put them to physical use--the physical use will then be fairly easy, and we'll go through that in the next volume, which completes this series, and of course regularly in other books and book series.

In the previous chapter, some of the main points involves deciding on the structure of each node--and as FCM is defined the Third Foundation, that structure is indeed very flexible; so we must define before we put it to use. The first triplets, as we say, are dedicated to the Match aspect of the node--ie, its 'input'.

As there are ten triplets, the fifth triplet is about in the middle, and this is dedicated to the Task aspect of the node--ie, its 'output' esp the output that concerns affecting something by means of what we have alluded to before as Elementary Actions. These adjust an area which could otherwise have come from the robot cameras, an area in software, indeed an image of the 160x112 type that we have found meaningful as input to the node networks in this context. Translated to physical robot terms, these could represent items on a table that is moved by an robotic arm. The motion signals to the robotic arm would be 'elementary action' while the input from camera would show a changed 160x112 matrix; and it's up to the FCM network to match over this input and calculate new elementary actions until enough has been done.

The app # 1005768 was made and tested before this book --or at least the programming in this book--begun. And so

the programming of this book will concern equipping that GPS stuff with a whole new FCM network, or another aspect of FCM, which is the robotic computational network proper. And that app, which you ought to find on g15pmn.com right now, is featured alongside the app that is coded, tested, corrected and put out on the app list alongside that one, and numbered 1005769.

So the app #1005768 shows this matrix--when you start this app, recently made and part of the 'fic5' robotics set of apps, it is ready to show you the 'sine wave', which is match image number 14, --the completing match image. And it shows it when you go to the field that says '14' and click 'i' there. This shows what is very clearly a wave, and ALSO very clearly an either-or matrix of not that many pixels. It has 160 pixels in width and 112 pixels in height. So a loop in G15 PMN of this kind would give you an 'i1' that goes from 1..112 and an 'i2' that goes from 1..160, the latter being 'x', ie, width, and the former being 'y', ie height:

```
LL:112
LL:160
| here, use i2=x
| and i1=y; the
| pixels are 0
| and 255
LO
LO
```

A calculation I did just now within the spreadsheet in that app is that 112 times 160 is 17,920 and that is indeed the same number that match image number 14 gets as match value #14. To see this image in Gem, open f10000 plus 220 cards times one less than the image number. That calculates to f12860 if I'm not mistaken. So when you open it, the first thing you can do is to flip it vertically, by the click on followed by '2'. Then it is in upper left corner of the GEM image view. Since mouse pointer is used only at rare occasions in GEM, you might as well point the mouse to the lower right corner of it to help you to figure out how to use GEM to modify just that part.

Apart from a typical 'noiseline' or two at camera input, most of the 112 horisontal lines of pixels make up the matching number, so that the more matches of bright=255 pixels against the match image, and the more matches of black=basis against the match image, the higher the match number. If you in Gem try to invert the image at f12860 in that app #1005768 you'll get a match number that reflects one or two noiselines, ie, something in the nature of hundreds rather than thousands. All things about match have to have a little bit approximation about it, because it is going to work in the ever-changing lights and angles as reflected by robotic cameras--and these may be sometimes jumpy, as they are mounted on a robot that may well be on the move. A program cannot be 'control-freak' about these things; the FCM program must accomodate.

In Gem, if you quick-save via you will set disk# in the Spreadsheet to 3 and card# to 9000 and image number to 1, in the C15 field, before you click 'I'. If you wish to have that as permanent setting in that spreadsheet while you go in and out of Gem and try things, you might want to save the spreadsheet to another location, like L1. If you then start the app as normal, it will have its original spreadsheet at K1 while your own version at L1. The spreadsheet at L1 doesn't contain any G15 PMN code at all--it contains texts, numbers, and any formulas that have been entered via the button. But the program that comes along with the app #1005768 has assigned some

an extra algorithm to the field C15 of that spreadsheet, and this is tied up to the letter 'i'. So, the GPS, or G15 PMN FCM Spreadsheet in its original app, #3555558, is a little extended when it is used in the robotics context. One particularly neat extension is that which it calls 'letter warps'.

A letter warp for GPS simply means this: if you wish to use the alphabet, a..z, to provide extra functions to the spreadsheet at all its available positions, the setup for it is super-simple. Just duplicate how it is done in any app that does set up letter-warps, such as app #1005768, and you'll get it right. Be aware that some algorithms, including the image fetching that happens when you click 'i' displays what it displays on top of a large number of spreadsheet fields and these should not be used at the same time, at least not for anything that has to be on display. Also, the behaviour of the GPS when extended this way may be other than expected when PgDn is used--that all depends on how the letter-warp is made (ie, does it check for whether a PgDn has been pressed?).

GPS can and will be extended in many ways to accomdate robotics and other types of programs. The way the GPS is extended in #1005768 may be just a fraction of how it is extended in another robotics app. The policy of G15 PMN app publishing, as I intend it, is that apart from occasional fixes that are discovered later and which is essential to its function (and for which there is no obvious easy workaround), when an app is published, it will remain in that state and be a 'milestone' that you can always go back to and find that it works in all future. As G15 PMN has, as project, evolved over the past decade, it has not been cluttered by acts of imitation over fancy-but-not-essential extensions as found in most existing programming languages.

Let me just insert a comment about how other programming languages and projects are typically doing it: they come along with a perhaps fairly fascinating program and a long "version number", with many dots and digits. Scroll ahead five years and fetch the program, and it is a totally different program in many senses and quite possibly, several of the charming things that might have fancied you in its initial version have been eradicated. Maybe not just charming things, but essential things. And so, while the progression from one version number to a "higher" version number is called "upgrades", they are often more precisely called "rewritings". And when one sees a list of "earlier versions", including the version that existed five years ago, one can bet two-towards-one that it simply won't work anymore, because it relied on hundreds of supporting packages each which have undergone "upgrades".

In contrast, the robotics programs made now, in G15 PMN, not only exists alongside all the other apps ever made for G15 PMN, and it is beginning to be a respectable quantity, --but also, in addition, the robotics programs made in the newest implementation of G15 PMN on the newest machines work, usually either directly or with an extremely simple modification, also for the first implementation of G15 PMN. This is the way to grow a knowledge base.

Back to the match numbers. The fourteen match images are matching any binary (ie, with either bright pixel field or non-bright pixel field) image of size width 160 and height 112 against the simplest geometrical ideas--such as square, circle, diagonal, parallel vertical, parallel horisontal, and some more, including such a 'wave' that is

derived from the Cosine and Sine functions which are used to give the x and y coordinate of a circle. So it cannot be much simpler. The bands used to 'draw' these images are thick so that it is the idea that matters in some abstract sense, rather than trying to match a fine drawing. If you wish to use these fourteen match numbers, ranging from a basis level of some hundreds up to theoretical maximum full match of an identical image of 17,920, to describe in some detail a drawing in thin lines the obvious thing to do would be to enlarge this drawing and do match numbers over first one portion here, then another portion there, and so on, in addition to a matching over the whole. In turn, nodes would be set up to reflect how these coordinate one another to describe features of this image so as to distinguish it from another image that also uses fine lines.

It is a good idea to conjure up some images in the G15 PMN 'native' image editor Gem that are suitable for such match number generation. As said, it can be a good idea to load one of the existing match images as a start, eg the one at f10000 or the last one at f12860, or any in between --exactly 220 cards separate each. Then, usually, flip it vertically by 2 before you work on it and adjust the mouse to point to lower left corner, and flip it again vertically the same way before you save it, eg to C9000 by a click on . To draw on it, the easy and obvious way is by means of the button, number '2', which is called Filled Rectangles. These you move around by arrows. You paste them in by a click on and can keep on moving them around, and 'slimming' them by pushing them against the corner. By a click on key (ie, indent, usually positioned on the left side of the keyboard just underneath the numbers), you switch the 'mode', ie, how fast the arrows move the rectangle. Click when done on the work each time.

When you are ready to make a pattern that you really wish to save and do match number over, you should, when you clikc , start by option '3' there, which is to set the 'tone'. The correct tone, of course, is basis (0) for black and bright green 255. So choose either of these for your rectangle and move it up and to the left until you are within the topmost 160x112 range. Click to put the rectangle in, when done, then eg to save and to exit and get into the app and see what match numbers you get.

When there is exact overlap of black areas with black areas and bright green areas with bright green areas, in those cases you get the highest match numbers. Perhaps the most educational thing to pay attention to initially when you learn to work with these match numbers--which, yes, are part of a great deal of upcoming robotic work, and so are important for you to learn as an expert on FCM--is to single out the two or three highest match numbers and one or two of the lowest match numbers, and figure out why this makes sense by looking at the relevant ones of the fourteen match images.

NUTS AND BOLTS OF THE FUNCTION CALLED "TRANSLUCENT"

If you recall our earlier reflections over just what a computational nodework is, I think you will agree that the the essential elements in a computer is algorithm and data and that all talk of nodes is just a description of same. Just as there is not just one way of ordering the many items that may be on a large workbench you have in front of you, so there is not just one way of 'doing nodes'. Any program, no matter how large, how complicated, how dedicated to matching over 'fuzzy' data such as robotic cameras and making meaningful motoric robot motions can be stuffed into a single node, into two nodes, into three nodes, or into ten or a hundred nodes, with more or less order.

What we would like now, and it may possibly be what most of this volume is dedicated to, is to make an example of a robotic node network that has in its the sort of order that invites a vast number of advanced, even hyper-advanced extensions. The task, therefore, is not at all to use as few nodes as can be for our inside-RAM 'virtual' robot, but rather to use enough nodes that we get a real programming experience as to how to make many separate nodes that then interact in a convenient manner. We are not at all interested in making the shortest algorithm as possible to get the output on the screen that we may be looking for--in contrast, we wish to create an instrumental robotic example of FCM nodes in which expansion is easy as a breath. A little bit of redundancy in how we set up nodes and how many we use is, therefore, just what we need.

If you read this as a developing G15 PMN programmer and want to get a good grip on FCM, look up the definitions for the TRANSLUCENT and the FCM functions in the third foundation. We try as much as possible to avoid making any changes to these essential definitions, trust that the design intuition that went into them is good. Try and make as much sense of these functions as can be--they are, after all, short. They use only standard G15 PMN functions and normal sorts of arrays, like FCMINDEX, and matrices, like THISFCMNET. The FCMINDEX is supposed to be always sorted according to level number, and the network or networks are supposed to themselves initialize sorting of the nodes in case level numbers have changed (typically they do not change).

The FCMINDEX has warp numbers--not just position numbers but warp numbers in it, so that it warps directly to the first number in each node, which is the level number; and this is why the BS two-letter function sort works so well with it. (Be aware that comments may sometimes be less accurate than the tested code--in this case, when you look up defintion of FCMINDEX, it is said to have 'numbers' for the nodes in it. It should have said 'warps' or 'warp numbers' for the nodes in it.)

One feature you might pay attention to when you try to learn more about the TRANSLUCENT function (I write it in capital letters not because it is inside the program in capital letters, but in order to quote or highlight the word as in the sense of an exact syntactical name of a program bit), is this: inside the loop that again and again reloads the nodes and performs them dutifully, the variables are also loaded in again and again rather than stored locally. The variables of importance here are FCMINDEX--the list that, via warps, has an overview over all the nodes in the present node network that the loop is

performing--and FCMINDQTY, or 'fcm index quantity'--the variable that tells how many nodes are in FCMINDEX. In addition, THISFCMNET, which has the node matrix in its normal, not necessarily sorted form, is, as policy, used consistently in the algorithsm called from within the nodes themselves.

A comment in the program connected to the TRANSLUCENT and FCM functions says that these node network variables may be changed while the loop is running--if done "thoughtfully". Let's now bring, indeed, some fullness of thought to doing just this. Here's the motivation: the GPS, the G15 PMN Spreadsheet, is spreading itself rather all over the place with its use of level numbers. While technically it is possible, and in some particular cases even advisable, to mix the GPS node network with another network into a single network with one index, the more obvious solution is to have one or more separate FCM networks going when we use the GPS as frontend to show some results, and input some variable values, and such, and want to build up perhaps also extremely sophisticated robotic node networks running in the same RAM, in the same program--and even within the same loop, TRANSLUCENT.

So when a node refers to another node by means of a number that number refers to the matrix. A matrix is typically described, as you know, by X for width and Y for height. For most matrices, X and Y are given as the alphabetical sequence suggests, namely width first and height second. In some cases, it is esthetically (or for other reasons) more pleasing to write it the other way around, and in the spreadsheet, the four columns are dominant in how we write a reference to a field there--like, A15, B24,--and so the notion '4x135' makes sense there, with 135 lines.

In a sense, a text document is a matrix and each line of characters can be identified with its Y coordinate; the X coordinate refers to which position it is on the line. The line number, or Y coordinate, of a node matrix is such that each node can be identified by its Y coordinate, while the X position refers to the various data in this node. If you have used G15 PMN for a while, you will know that the difference between a warp and such as a Y coordinate is that while a warp is 'lightening fast' and ideal for large loops, a coordinate number has to be translated into a warp to be of any use, over and over again. However the nodes in an FCM network is typically set up so that heavily computational feats are done more by clever algorithms inside a node than by spinning over lots and lots of nodes. After all, the nodes must be programmed in a first-hand way to be properly part of our whole enterprise, and that means, in a way, that nodes often are 'handcrafted' (except for a smaller set such as the spreadsheet matrix of 4 times 135 where a loop sets them up to save coding space).

Also, if you wish to make an FCM network so that its particular makeup after performance for a while is stored to disk, coordinate numbers rather than warp numbers are definitely the way to go, because a warp can be different from one program run to the next (they depend on what possible extras have been preloaded, and even minute actions like creating a quote of a tiny text to be printed to the screen may affect the warp numbers of the next compilation). This also shows why the functions hooked on to the nodes in FCM are put there usually not as warps but as numbers that refer to the FNACTLIST array of functions.

The subroutine in TRANSLUCENT that calls on a node's algorithms is called PERMUTEACTS. It uses the FNACTLIST and looks at all the ten triplets in the 150-number space alloted to each node. In the middle of each triplet there is either the basis value or the number of a function in FNACTLIST that is to be performed.

The functions in the FNACTLIST in turn, typically refer to THISFCMNET, unless they are of a particular kind that should directly address the node network matrix for instance so as to change THISFCMNET. THISFCMNET is a variable that, with the GPS, is set to the value of FUNDNET, which typically is the first matrix of nodes in any use of FCM as defined in the Third Foundation G15 PMN. In the case of the GPS, it sets aside five nodes for each field in the spreadsheet, and extras also, to get the whole shebang going.

So to make a node network that interfaces beautifully and seamlessly with the G15 PMN Spreadsheet, we now know which variables and functions to pay attention to.

GETTING THE G15 PMN FCM TRANSLUCENT LOOP TO DO TWO NETS

We're driving on with 'hardcore programming' now in this volume and when you have time to read it while also doing your own bit of G15 PMN programming, I bet that you will get questions that you wish you can put to me and get instant replies on. Remember that when you phrase questions clearly to yourself--and having typewriter skills so you type fluidly and effortlessly on a full keyboard with text editor on screen in front of you helps enormously here--you may find that you get ideas as to how to answer the question, perhaps even at once. If you are the one-finger or two-finger type when it comes to keyboard work, I suggest you plan to upgrade your skills to two-handed touch keyboard typing expertise ASAP. A human brain is masterfully complex, and it needs the katharsis of clear expressions of a logical, coherent and intuitive kind that it can reflect further on through the majestic input of a beautiful computer screen. Your intuitions flow in and assist the brain in regaining coherence again and again when you make yourself humble to this process and indeed also to 'truth' in the larger sense, and allow time for such meditative writing sessions which have one function: to give yourself clarity.

Another person may have a need for a different series of expressions and so dialogue is a wonderful thing but often the real clarity, and foundation for good dialogue, comes from the foundation of doing one's homework at the keyboard in the form of clarifying writing. Mere verbal expressions through sound do not give the same full experience of the expression as typing it in. And it helps that the screen is bright spring green and black, as this stimulates but in a very harmonious way.

The Translucent loop is so simple that it both can and will be used in all sorts of ways also as regards enabling more than one node network. Here we will look at the simplest way to do it, which is to switch from one net to the other and back--ie, toggle net--in one of the first nodes in each network. Let us see how.

First, what is--cut to essentials--a loop? It is an algorithm that performs the same lines over and over again until some or other condition take place. A bit like this:

```
*START
  *DO SOMETHING
  *CHECK WHETHER EXIT
  *DO SOMETHING MORE
  *JUMP UP TO START AGAIN
```

In the philosophical understanding that we should not necessarily cater to infinity when we can bring in a definite number, we often (confer earlier volumes) make a loop that is going on 'indefinitely' like this--the first number being 2 with 9 zeroes, ie, two billion--which is not far from the 32-bit upper limit of number sizes:

```
LL:2000000000
GOODACTIONSHERE
ISEVERYTHINGDONE
N?
SE
EX
LO
```

One can also push 2,000,000,000 into the loop counter, which is i1 for the first loop, i2 for any inner loop, next to it, i3 for inner loop inside that again,-in order to make it exit without leaving the whole function with EX. So TRANSLUCENT has two loops, the inmost does the net or nets by looping over all the nodes, the outer repeats until a variable says that no further 'cycling through the nodes' is required. The outer loop uses the EX method. The inner loop uses the pushing top number to loop counter method. The outer loop don't use the two billion approach, it starts by a simple LL:1 and does something like this:

```
LL:1
Q1
GOODACTIONSHERE
ISEVERYTHINGDONE
SE
EX
LO
```

The Q1 reduces the quantity of the index by 1, 'quiet' the number by 1 (and confer M1 with M as in Moderates)--so it only can exit at the SE .. EX point. The variable that in TRANSLUCENT marks the exit is understood to be set by the node network. As the 'LO' is reached, the inner workings of the programming language hitches the value up by 1 again, but as it doesn't reach 2 this way, it won't exit through its LL..LO mechanics. It will keep on looping until the SE gets a DANCE flag, ie, the number 1, and thus allows the command on the next line, the EX for 'exit' to perform.

There are two conditions for a node in TRANSLUCENT to be considered as for performance of the possible warps in its triplets: one, that it has any warps in it--the flag at position 9 in the node tells this--two, that the node is market as active, and the flag at position 49 tells this. Note that a comment inside TRANSLUCENT says, incorrectly, that this flag is at position# 40. So read the code first and take hints from comments, knowing that some program comments may be incorrect--even as they may also point out something. It is honoring the creative process to check code more than comments when new programming is being made and it is honoring future programming projects to have the nerve to declare a program as finished when both the results of testing, and your intuition as a programmer, tells you that it is finished. And after that point, one

doesn't go back to fix up such what is later found to be incorrect comments inside the program. That is how it comes to be incorrect comments inside the Third Foundation and it is perfectly alright. Rooting out every weed of incorrectness tend to come along with new forms of incorrectnesses, at least in this context. In a translation of a point made by C.G.Jung, author of the 'synchronicity' concept of meaningful coincidences, what is perfect may not be complete and what is complete may not be perfect. If by 'perfect' we emphasize a kind of absolutely cleared-up state, while by 'complete' we more think of a process of unfolding organically in its fullest sense, beautifully, then in many contexts we should seek completeness, rather than perfection. However for high-security programs, when it comes to program correction--through superb testing--it is okay to be perfect as well, and maybe even necessary. This perfection doesn't not have to extend to comments inside the program. If a perfect program has some incorrect comments, then as a whole it is a complete program :)

I believe that prior to the technology of electronics which led to the development of the digital computer, the quest for expressions of the most beautiful and orderly kind, and with a sense of grand abstraction about them, led to various 'fetishes' in human culture. One of these 'fetishes' was the geometry of circles and triangles and lines and axioms around them. Another one, at least in Europe, was that of the Latin language. A third fetish was that of studying the texts attributed to Aristoteles in ancient Greek, texts that survived Christian suppression thanks in part to Arab scholars (and which Christian thinkers later seized upon with an attitude to merge it with bible-inspired thinking). Yet another such fetish was that of strictly rule-bound forms of poetry.

With the advent of computer programming, something new and potent came into human cultural development, and I believe this was an essential aspect of the best parts of the hippie cultural revolutions of the late 1960s and 1970s. Thanks to big tech companies, which at present try to dissolve programming in favour of their snakey AI products, and who portray copy-and-paste of the outputs, visual or verbal, from their algorithms as almost equating that of being a creator yourself, again the restlessness of a humanity not having an abstract pure expression for thinking and for personal mind-katharsis seems to be coming back. In 2024, at the time of writing, fewer people than before master keyboard, and those that do master keyboard, should be warned that the profession of writing is waning, statistically--leading to an Internet that feeds upon itself in a recycling loop without the human mind whether as creator or controller, and without programming being advised to be of importance to new young citizens anymore. This recycling loop of mindless material may have in it hypnotic features to move masses to certain emotions, engage in certain 'group conflicts' on the Internet, and at the same time may drive weapons and security systems and id systems so that there is less and less room for the human being--except possibly if that human being happens to be a billionaire who is sole owner of a tech company. If this billionaire happens to be off the hook, there's no pathway ahead unless politicians put into the form of law to switch off all dependency on so-called "AI" in absolutely all areas of life, and to forbid certain types of particularly

harmful categories of so-called "AI" expressions. In the long run, this dissolution of "AI" is a historical necessity--as I see it; it is bound to happen and I'm doing everything in my power to ensure that it happens.

Let us return to the main topic of this chapter: how to get TRANSLUCENT, the main FCM loop in G15 PMN, to handle more than one networks. It is meant to. But how? The foundation has been laid by it and its subroutines (or, more precisely, the functions warped to from within the nodes that it is performing) generally handling all references to the present node network by means of variables rather than more directly. In particular, around the start of the definition of the spreadsheet, we find:

```
FUNDNET
LK
THISFCMNET
KL
```

In other words, fetch the matrix referred to by FUNDNET, which is the GPS spreadsheet, and store that reference into the THISFCMNET variable. This latter variable is used by TRANSLUCENT, and it is used over and over again, so that any change will instantly affect how the TRANSLUCENT loop works. Also, the quantity of nodes in the net is also stored in a variable that is fetched again and again.

What we're interested in is to get something like this to take place after each successful run of the GPS nodes:

```
NETRBT
LK
THISFCMNET
KL
```

And also update the quantity. And then we wish one or more loop cycles with this NETRBT, this NETwork for RoBoTs, and after that, at a suitable point, it will switch back again to

```
FUNDNET
LK
THISFCMNET
KL
```

One of the perhaps mildly fun points about level numbers is that they permit a bit of 'anarchistic' programming--this is something I learned from the line numbers in an early 1970s language called BASIC, which preceeded each line with a number like 10, 20, 100 or whatever, within a range, so that you could type it into the machine in any sequence you liked, as long as you got the line numbers right. And by keeping intervals there, say, between 10 and 20, you could later on add a line quickly between the line numbered 10 and the line numbered 20 simply by choosing a number in between, like 15. Let's just remember to get that sort of the nodes done when we add extra stuff in the beginning. BS means bubble sort, and it is eminently a first-hand way to sort.

The level numbers are in this regard exactly the same. The first level number in use for the GPS is really high, and so we can perfectly well add nodes to the GPS in the beginning, perhaps with level numbers like 10, 20 and 30. Here, we can have a node that performs some kind of switching, and we'll now figure out how.

But first, to make this seem much more real to you, you can do some explorations on the G15 PMN terminal. Start up a form of the spreadsheet that requires you to type such as 'FCM' to actually perform it, so that you have all the

spreadsheet loaded in while you use G15 PMN interactively. Here, type, to get the warp number to visibility:

```
FUNDNET
LK
NN
```

Whatever that number is, just glance swiftly across it, and proceed to type:

```
THISFCMNET
LK
NN
```

And here, the warp number should be exactly the same. Usually, this is a pretty huge number. And as said, these warp numbers need not be the same from one program run to the next. It refers direct to RAM. More pompeous and, we can say, poorly designed programming languages, like those called 'object oriented' make fancy concepts to avoid showing warps as if they are afraid of transparent caging around their machinery; and then comes along other programming languages made by people who are frustrated with all the indirectness of the hierarchical classes of objects and which provide such as a possibility of 'storing a function in an array' and 'passing a function along to another function as parameter'. These big, fancy words are just a sugar coating on a cake that has rusty design. Object-orientation is taking an _approach_ to programming too seriously, and in that confusion, trying to pretend that the computer isn't all about algorithms and data even though it is. It is an amplification of the digital computer programming language in a meaningless direction. Historically, the language that had to be amplified was ALGOL. Existing at the same time was the rather ignored language called FORTH. ALGOL was amplified into a hierarhical nonsense that came to dominate the planet; meanwhile, FORTH, requiring a simplification of form to be less cluttered and more readable, instead was given an even more cluttered format in the name of 'generality' and 'independence' of the particular quantity of bits in the computer. So FORTH was given exactly the wrong treatment; ALGOL was given a meaningless treatment.

At the moment of speaking, few are using G15 PMN but I permit myself to describe its entrypoint with something of pride, because the advent of something right in a deep sense is not necessarily statistically easy to see when it arises, although it will be evident later on, also in statistical form.

How did I begin making the G15 assembly language? I did not at the time visualize PMN. I was watching waves, staying at a beach area in South-East Norway, looking at smooth rocks, asking myself: what is the minimum set of instructions for an ideal first-hand programming friendly CPU? And how does it core font look? For I wanted the CPU to come along with a font. Having got the fundamentals of this going in 2012, I was at another beach, Huk, at Oslo--not perhaps as grand as the Hvaler beaches, but with its own charms, and I sketched a way that I could imagine programming at a higher level G15. The Sun was intense, and I had a computer screen that could just barely show anything with all the blazing blue sky around, and the many bronzed bodies going in and out of the waves and kids splashing about and crying out loud. So I sketched something like this:

```
HINEWFUNCTION=
LL:10
^Hi!
PP
LO.
HINEWFUNCTION
```

This I wanted to be a loop that prints Hi! out ten times on the screen. PMN was born. Once PMN was done--and a couple of simplifying instructions was added to G15 to accomodate smooth fast running of PMN--and, in the process a new font also, the B9font, made while again at Hvaler beaches--the task was to get the B9edit, this editor that I have used since for all long texts incl. this volume--up and working. Next, GEM, the image editor, and again making sure the G15 had some suitable extra instructions to help swift showing of its most standard type of images. After a couple of dozens programs, the Third Foundation was born and in it the summing up of the physics insights I had been working on since the 2005 publication on physics (then trying to use something like a bit of Java, an object-oriented language to reflect some of the ideas) was possible, and that is the Super Model Theory in its mature 2017 form as included with every Third Foundation package since then and also published as part of a larger art book, registered at National Library of Norway, together with an art exhibition at Handverkeren, Oslo.

Let's again return to the theme of this chapter, how to get the TRANSLUCENT loop to handle toggling between nets.

To see even more as to how FUNDNET is defined, and where the assignment of the value of FUNDNET to THISFCMNET takes place, you can do two scans. Type SCAN and type in FUNDNET and then give the starting cardid for the spreadsheet, and any large enough number like 9999. As it locates FUNDNET, you type CAR and it will show how it gives value to FUNDNET. Type MMM and again CAR and it will show a card which also has something like this in it:

```
FUNDNET
LK
THISFCMNET
KL
```

The 'KL' puts the value in THISFCMNET which 'LK' fetches. The LK is, as mnemonics, a kind of 'lucky look' into the value of a variable. The reverse sequence of letter suggests a kind of 'killing' of the existing value of a variable followed by the putting in of a new value.

So we wish to set up a new set of nodes, for our robotic software groundwork, which performs neatly alongside the spreadsheet nodes. The shifting involves three variables, at least--in order to switch between nets:

```
THISFCMNET
FCMINDEX
FCMINDQTY
```

Come to think of it, in case a function in a net calls on something like FNAM or FNAMW--where the name of a node can be converted to its warp, at runtime--this should be updated as well:

```
NEXTFUND
```

The NEXTFUND gives the quantity of the funds in this net.

Hm, let's make two mini-arrays to store these--what is it called--quintuplet--that would be five, 'quadruplet' is the word I want, it means 'four of a kind'.

```
GPSNETVARS=
^abcdefghij.
RBOTNETVARS=
^abcdefghij.
```

When you build one net, wants to save the present state of the quantity of the net and such, in order to get on with building another net--or just to store it, so you can start the application which involves net-switching, then it is good to have an easy-to-use 'save' routine:

```
SAVENETVARS=
|IN:MINIARRAY
TX
THISFCMNET
LK
1
JX
YA
FCMINDEX
LK
2
JX
YA
FCMINDQTY
LK
3
JX
YA
NEXTFUND
LK
4
JX
YA.
```

So when the GPS is defined, and we're about to define the robotic net for our new application, we run

```
GPSNETVARS
SAVENETVARS
```

There is something so neat and sweet about mini-arrays, the ease and first-handedness shows the poetry of PMN.

Let's make the complementary structure, LOADNETVARS:

```
LOADNETVARS=
|IN:MINIARRAY
TX
JX
N?
SE
EX
```

This means that it only does a job with a nonbasis input.

```
1
JX
AY
THISFCMNET
KL
2
JX
AY
FCMINDEX
KL
3
JX
AY
FCMINDQTY
KL
4
JX
AY
NEXTFUND
KL.
```

The name of the first net, as used by GPS here, is simply FUNDNET.

Anyway, when the robotic net is defined, and the two nets are about to perform and sometimes interchange, we must get in this statement, --saving the RBOTNET and getting it back to GPS spreadsheet:

```
RBOTNETVARS
SAVENETVARS
GPSNETVARS
LOADNETVARS
```

This sort of stuff as you see is pretty easy to expand to 3, 4, 5 or some meaningful number of nets.

FUNDNET will get a companion called NETRBT (and in other software projects we can have more NETRBTs), and, for simplicity, we assert now that we want one cycle of the spreadsheet FCM nodes and one cycle of the netrbt nodes to perform alternately. Let us first consider a switch after one cycle with each; but we may need quite a few cycles of NETRBT for one cycle of spreadsheet if we are 'spreading the actions thinly' over many nodes in many cycles.

To do this toggling, we put in a node somewhere in the GPS net to get the switch, and we also put in a node somewhere in the NETRBT to get the switch. It could in some cases seem a bit logical to put that switch in after a cycle has performed, ie, in the highest node. While this is possible, it is not the obvious choice, because as soon as you shift the value of FCMINDQTY, that will affect TRANSLUCENT and since any two node networks are unlikely to have the exact same quantity of nodes in them, we must in case pay close attention that TRANSLUCENT doesn't under-loop or over-loop if we toggle the nets at the highest level of the net. The more obvious choice is to use one of the first nodes, and here we might as well say: the first node of GPS will handle toggling of net over to the other net, and the first node of NETRBT will handle toggling of net over to the first net again. It's like two babies feeding each other :)

The GPS node, when it has started, must allow the full set of GPS nodes to perform before it switches. But it can ready itself for toggling to the other net by setting a new variable that we create here, called for instance NEXTNET.

Just now, I wished to check--is it certain that no such thing as NEXTNET has been defined already? Here is what I typed in:

```
^NEXTNET
EXISTS
```

And it replied, courteously as G15 PMN often does, "Just make it!" Try it yourself when you have the opportunity.

Just remember that G15 PMN is not babysitting every action you make. If you make two functions called the same and you don't figure it out at first, you might get a lot of confusing results. It won't perform both when you type in the function name, and it won't argue against this double definition. You must check yourself, when you make a new function and you suspect it could be defined already whether it is defined. EXISTS can be used, and you can also use the SCAN function and scan, as for three-or-more letter functions, for FUNCTIONNAME= over a vast range of cards, and, as for two-letter core PMN functions, by a search on something like " BS:" Be sure to put in a space first, then you give the two letters--bubble sort BS in this case--and a colon after. That, as policy, is how the header of each two-letter function in G15 PMN is defined.

So let's imagine that NEXTNET is not the same as THISFCMNET, and we're in the beginning of the GPS net. Then in what could be the first node in GPS, we could have this sort of activity:

```
THISFCMNET
LK
NEXTNET
LK
EQ
| when not eq,
| set thisfcmnet
| to the other net
| and so also with
| fcmindex and such
```

In the next node, right after this, we could have the GPS being "courteous" to the other net and affirm:

```
NETRBT
LK
NEXTNET
KL
```

The first node of the NETRBT could have exactly the same algorithm as we had for GPS. The second node of NETRBT could be "corteous" to GPS this way:

```
FUNDNET
LK
NEXTNET
KL
```

Before we start up TRANSLUCENT, we should give the value of FUNDNET to NEXTNET, so that the GPS gets up smoothly.

This we can call CToggle, or "Courteous Toggle" between nets. The following interactions with the TRANSLUCENT loop would take place:

The TRANSLUCENT loop would call the first GPS node.

It would match over NEXTNET and THISFCMNET to be identical, and so let TRANSLUCENT proceed to the next node in the same net.

In this next node, the GPS will set NEXTNET to be the robotic node network, NETRBT. And then all the nodes in GPS will cycle through one time through the inmost loop of TRANSLUCENT.

When the outer loop of TRANSLUCENT restarts the inner loop of TRANSLUCENT, next time the GPS node 1 will find that NEXTNET and THISFCMNET are no longer equal, and it will adjust the values of THISFCMNET, FCMINDEX and such. This will instantly lead the inner loop of TRANSLUCENT to open up and give good firing-energy to NETRBT, its loop counter now no longer on the first node, but on the second node. In the second node, a CToggle--a "courterous" toggle will take place, in that NEXTNET is set back to GPS. But this toggle won't take place before the whole cycle of NETRBT robotic network nodes, have completed, and it is back to node 1 again.

The more I have looked at the upcoming program example the more I'm convinced that we should have more cycles for the NETRBT for each cycle of GPS. So instead of a plain switching back-and-forth between the two, I suggest a modification of the above: let it not be node 2 that sets NEXTNET when the NETRBT nodes are running--let it rather be a task node that does it when the task has been completely performed, so that we get a visual update at a point where the image has changed instead of a repeat of the same image again and again.

Quite often we wish nodes, especially the nodes that have a 'subtask' role (ie, lower priority) to be switch on and off by other nodes. While technically this could be done via the #49 'Active?' field in each node, it is a more gentle, and in some senses, a more longsighted approach to let an ordinary triplet value somewhere in the node have this role. That leaves room open later on for blending several networks by regulating their Active? flag whole series at a time.

Time for caffeine and a break for this writer. You know, one thing about certain types of long texts is that they can only be written by getting at it daily for hours with only at most a couple of days break between each streak. For otherwise the drive and tune of the thinking may dampen. The car works best when the engine is warm. And as the car needs oil, the human brain can do with a bit of fish every now and then. Good too as aphrodisiac.

SLIGHTLY MORE CONCRETE ABOUT OUR UPCOMING ROBOTIC APP

You will be able to find the completed program in an open and freely available G15 PMN app list so if in doubt, you can check against the program code there. It may even be radically changed, since sometimes the initial thoughts about a program design sometimes have to be changed deeply because new, unthought-of factors emerge in the programming process. This is part of the "dialogue with the computer" or with yourself through the computer and its formal computational language, and an impetus to the creative process.

I think we are ready to begin to set up nodes but first, let us figure out something worthwhile to do with them that is educational relative to how many more nodes can be set up, and which gives us, in software, a good idea about how certain "goals" set to some task nodes is able to fulfill themselves through many small "steps" by nodes--which here turn to a matrix of 160x112 size with either 255 (bright green) or basis (black) to implement these changes.

I need to see how feasible it is to set up--it mustn't be pointlessly complex, nor take an inordinate amount of net cycles to begin to see results, and the results should also appeal to direct experience of what goes on--so that the coherence of the node network is evident.

My initial idea, before the upcoming programming takes place is this:

*Inside the GEM Image editor, we shape a 160x116 image with the bright and black pixels probably by means of rather broad and simple rectangles, and save it to C9000. It should neither be all black nor all bright nor look identical to any of the fourteen match images. It is very okay if it looks a bit disorganized. The software "robot" is going to clear it up.

*The spreadsheet, part of this example educational program with a software-only robotic node network that we now make as app, will of course show the 160x116 image. We imagine this is, in some vague sense, an 'input' by a robotic camera.

*The spreadsheet will, as in the app #1005768, also show the fourteen match numbers.

*Let's call this new app #1005769. We create here a set of nodes. The most elementary actions are here oriented towards a 'fixing up' of the image within a tight set of rules: dividing the image into a set of imaginary 'lines' composed of same-sized rectangles, and not all that many--we let one elementary action be a swapping of any rectangle with any other on the same line. Let us imagine that this is a sort of virtual version of some sort of 'picking up and placing somewhere else' type of action by robotic motors.

*At the input side, let us have fourteen nodes, each storing the fourteen match numbers, keeping in mind that the maximum value of any such match number is 17,920.

*At a somewhat higher level (ie, with a higher level number that the input nodes which fetches the match numbers from somewhere--and why not from the spreadsheet since we're interested in seeing what goes on and so updating the spreadsheet quite often) we can have a node that tells us which of the fourteen match numbers are the highest. If, for instance, the input image is nearly a sine wave like match image #14, you would expect the fourteenth match number to be maybe 17,000 or so, and all the other match numbers to be considerably smaller. But with a more shall we say 'stochastic' image, you might get average match scores on one or two handfuls of match numbers.

*Any sophisticated real-life robotic G15 PMN FCM program must have top priority 'ethics' nodes but this is in principle easy to set up given enough sophisticated match nodes; this is work that belong to the implementation of physical robotic programs and in a software educational playground we do not need that ethics type of node.

*As top-level task node we wish to implement, one elementary action at a time, such simple moves on the main image that, if possible, increases the highest match value the most, given the range of possible 'horisontal'

actions along each 'line' in the image as for the swapping of rectangles there. If the highest match value can only be increased by a very small degree, like some hundreds, the robot shall declare the process as 'completed' and 'done'. Let us be aware that, while a simple program can meaingfully 'fortify' just one match image, then in the real world we may have ambiguities that we can best help bringing forth by elucidating more than one match image possibility. This could be done by having more than one net, and they could also carry on 'simulating' actions--ie without updating the visible image on the spreadsheet--before they have tried to raise two or more match numbers to the highest; and after all that, the 'winning' match number, or even the winning pair of match numbers, or winning triad, is given to further processing.
*In order to compare the various effects of the elementary actions on the image, we need to have a copy of the image and let the match go on relative to this copy.

In order now for the nodes to do the jobs one could otherwise compress into loops of loops inside an algorithm direct, we should have a layout of nodes big enough that we learn how to let these nodes interact in a suitable manner while the NETRBT loops again and again between each 'elementary action'. We could have a variable in each of the elementary actions--in other words, a number position in one of the triplets--that is a flag: 'is this actual mode?'--DANCE--or 'is this simulation mode'--BASIS. Each line could have a main task node and each task can call on all possible elementary actions for this line, by setting parameters in the 'motoric' nodes--the nodes with, in general, the highest level numbers. If you recall from earlier volumes, we can imagine a sort of mountain view of nodes where the lowest level numbers are the left side of the foot of the mountain, the highest level numbers are at the right side of the foot of the mountain, while in the middle--near the mountain top--we find that which is conceptually at the highest level, but which, with a certain well-defined language, we can call the high priority nodes including the top priority nodes.

A high priority task node will then set the subordinate tasks in the Simulation Mode, where changes do not go to the main image but to the copy of it, and gather the data from the incoming match numbers on this. We can imagine that the input or 'sensory' nodes also have a Simulation mode and an Actual mode, which determines which image they will match over when it comes to match number calculations. This is perhaps most elegantly programmed by ignoring the spreadsheet's own match numbers, since they are quickly calculated anyhow.

The higher-level task will, in each cycle, call on yet another elementary action, one after another on each line, then all again on one line after another--until all the lines, or all the lines we bother to include (to simplify we could decide eg to only take one part of the image), have contributed with match number increase options. The highest result alongside which line and just what action on this line is each time stored by this high-on-the-mountain task. When enough cycles have performed and the task sees that all options have been explored, the 'winning' elementary action gets the task of actually updating the main image--and the NEXTNET is set to GPS to show it. Let's complete this musing over the coming app by

asserting that the image is written back to C9000. That's as far as this software-only robot goes in 'realizing' its actions in the world!

We have now used rather broad terms to describe the activity and interactions of the nodes. It would seem that the next step in this process, before we do the programming, is to be considerably more specific. Just where do we store what flag, just how is the maximum match number stored and how does a main task check whether the completing elementary task has been performed, and in what cycle--maybe the same or maybe the next--will the writing of the simulation image to the actual image take place, and more such. When we have been specific about this, we will write, in capital letters, some of the key portions of the code here and I will take up the G15 PMN programming terminal and type in after what's written in this book and 'report back' on how it works and we'll keep on this until app #7005769 is complete and the book radiates, as it should, a good first-hand FCM node network understanding in the young and agile minds of my readers.

INTERMEZZO: PREPARING TO MAKE A SYSTEMATIC LIST OF NODES

In generating mindful expressions in huge quantities, the body/mind/brain needs more than a bit of food, exercise, rest and pleasant music. It must have a direct replenishment of the undercurrents of perceptive intelligence, where the abstract forms are directly experienced in a kind of state of mind which is both transcendental and, in an open sense of the word, tantric. And we're not talking mere minutes; not talking mere 'fascinated watching'; the most healthy, fresh, young human anatomy must be experienced from within and with the body active, the mind getting a sexual mode coupled with the artistic mode so that the deeper recesses of the perception of pure esthetic form is renewed: and that perception is the natural rocket fuel for human intuitive intelligence, whatever area it expresses itself in.

After dipping into a tantric resirculation of my energies, I can continue. Let's explore further towards the concrete implementation of our robotic node network with a software-only 'virtual robot' doing the job here.

First of all, in last chapter I went a bit back and forth as regards just how the robotic network is going to, as it were, have a railroad switch that switches the tracks for the railway train from the robotic network and back to the spreadsheet GPS network. There is a middle-ground between letting node that is amongst the first do all of that job, and letting a node that is nearer the top priority task nodes do it:

A top priority task node can switch on a flag in a triplet in one of the first nodes as a 'request' to switch to GPS. In that way, we retain a clear, first-hand good overview over just where that switch takes place, while any node in the robotic network can affirm that the time has come for that shift.

Another point: we have set aside desiseconds time since bootup or since startup of program to be put into several fields, especially associated with tasks. In some robotic frameworks, the G15 PMN operates in a more text and less graphical/interactive manner (to save computational time and energy and heat generation in the steering PC, and with the fullest focus on getting the robot to do its task properly), and in these frameworks, how to get a desiseconds-since-startup count may involve a command that goes to external electronics rather than in to the CPU of the PC. So when we make now a software realization of the robotic network to learn about nodes so as next to work with them alongside the physical robotic machinery, we should feel free to skip timing information, at least when it's not important to get the program made. In that way, this app we're now making, the app #1005769, can run on all G15 PMN platforms (or at least without any much adjustments).

Usually, when we visualize a FCM node network that we're about to make, we include in that visualisation a sense of just how fast it might be. It makes for a very different program if we visualize that each cycle will take, say, ten seconds or half a minute, rather than, in the typical case, a split second. But when it does take a split second and part of the robotic network is to signal something to the robotic motors, which may take several seconds to complete the elementary action implied by that electronic signals, we must also implement a sort of 'patience' in the network. It must not assume that just because a task has been initiated in, say, cycle #50 in the loop, that it will be complete at cycle #51. It may not be finished before cycle #250. And it may vary from time to time, from PC to PC, from robotic motor to robotic motor, from elementary action to elementary action, from one FCM network to another FCM network.
  So each task, when called on to perform, has, as we've already clarified earlier, triplet variable fields that inform other tasks as to whether the task is still going on or has been completed. These fields we might as well have also in a 'virtual' robotic network with a sort of imagined robot, because a task can consists of some or indeed many subtasks and as a whole, that would probably take many cycles to complete.

Put in other words--and this is to clarify my mind as a programmer in this project as well--in this sort of "dual-purpose" writing that is suitable in an Art of Thinking exploration--a node doesn't enroll another node in its own loop when there are other, more 'network-friendly' solutions. Rather, a node sends a signal to another node to do such-and-such, and let the network cycles and the TRANSLUCENT loop call that other node to respond on this signal; and the node checks--perhaps repeatedly--as to whether that subtask has been done and then it may send more signals, to the same or to another subtask.

The "high priority" nodes are those that organize much activity here--whether as a summing up of input features or as a ethical necessity node or as an organization task or 'supertask' calling on subtasks that, sooner or later, involve elementary tasks. So the 'high priority' nodes is that group 'on top of the Olympic mountain' of the network which have an organizing role, generally more than other nodes. If the FCM network has ethical necessity nodes

setting the constraints for all other nodes to follow, these have the highest priority of all the high priority nodes and (from this volume onwards) we say of them that they have "top priority". Distinguish this from their level number: think of it as a left-to-right 'position' starting with such as camera input nodes and going all the way through the high priority nodes and further and further to the right where we have the elementary actions. Got the picture? Level is left-right but priority height.

I think we're ready to assign concrete roles of definite positions to nodes, and give them definite level numbers, and begin to sketch the exact code to get this up and running. The least interesting part as far as educational FCM robotics programming goes is that little function that moves a block on a little image left or right so as to swap a bit with another. That's a very typical little algorithm and the only reason we are going to have that as part of this app #1005769 educational app is that we need something ultra-simple that at least as a modicum of association with how the action of a robotic hand may affect camera input. So I will spare the reader for those more conventional programming topics especially as earlier volumes have already have a share of such. Here we focus on robotic FCM essentials, and then you go and fetch the app #1005769 that has that algorithm. I will however get it to work while these chapters are getting written so I can give an impression of how it works. It is always a little bit of a thrill when the design you've had in mind for a long while suddenly is an imaginary 'engine' inside the computer that makes it act as if it has a personality associated with that design. There is always something new to that, if the programming project has any complexity at all. And this certainly does.

Next, I will try and compose the programming notes as lists--this has, when it comes to a certain stage of programming, a couple of advantages. One, it takes away the story-telling component around the ideas and instead gets the ideas succinctly presented and in perhaps a somewhat free sequence. Two, the list is compact and so easy to look at in the further working process as a guide and map--especially if it can be taken in at one glance. Three, the list allows attention to go separately to each idea or pointed listed, and so it is possible to spot whether a further clarification is needed at some points.

LISTS TO SUMMARIZE THIS FCM ROBOTICS PROGRAM

Here, as a programming framework, we summarize what we've already have decided earlier in this volume and also add new information which should be mapped before the app #1005769, a first robotic FCM nodenetwork performing only in software, is programmed--and this is the actual sequence for this author, with one exception: I try and go back to earlier chapter when writing on and correct when writing the later chapters, when I realize something should be changed. It's part of the honesty of the process that I haven't done such correction very intensely; also,

at some stages, I imagined completing the program before completing the book but felt that the later changes, in order to get the book to stick totally to the code, would perhaps make the text less lively and also there's, for the advanced FCM programmer, an energy of exploration in knowing that there are two approaches to the same program idea--this book is one, and the actual open source code for the program is the other.

The max 100 links are at node positions #50-149, and the ten triplets, with the middle value reserved for the number of an algorithm, and the first and third value free to be assigned, are here: 10,11,12 tp1 13,14,15 tp2
16,17,18 tp3 19,20,21 tp4 :mostly for match things
And mostly for task things: 22,23,24 tp5 25,26,27 tp6
28,29,30 tp7 31,32,33 tp8 34,35,36 tp9 37,38,39 tp10
In addition the Luxury Values are at pos# 40 to 46 and these can be used for match and/or task aspects; if still more is needed, see FNAYA etc for arrays involving passive data nodes which can be created as needed to accompany a normal active node. The links are at pos# 50-149, with the quantity of links at pos# 47, while #48 and #49 are flags: respecively, the high priority flag and activeflag. As for this brand of FCM networks we are initiating, we distinguish the triplets the way we just said, and also: The pos #47 has quantity links, and when any match links are used, these come first, and task links after these.

Note that eg a high-priority task can, technically, set activation of lower-priority tasks incl elementary actions directly without consulting the links; but doing it by means of the links is, as policy, considered 'cleaner' in that the links provide a consistent overview over what the functions inside the node are referring to and thus more easily can be changed.

*All nodes in this FCM network which aren't of the high priority types, whether 'match-oriented' or 'task-oriented', has a soft activation field (in contrast to the hard activation field #49) and itself will check on this field being DANCE before doing anything and will neatly reset it to BASIS when done. This we assert is position# 21, which is the third value of tp (triplet) 4. In other words, when adding an algorithm to any of these nodes, be sure that more or less the first thing it does is exit when pos# 21 is zero. When pos# 21 is other than this, some match-nodes will use this value, 1 or 5, to determine which of the images--the main image, or the 'simulated action' or copy image--to match over.

*Match-oriented nodes: main value is PERMILLE, ie, just how great, 1000=max, 0=not, does the input match up this particular criterion/feature. In the case of match numbers 17,920 is max so that will be calculated into 1000 here. Second value is function to calculate this on condition that the node has been activated in position# 21. Third value of tp1 is the probability, again PERMILLE, that this is right--which is here a redundant field, but we set it to 1000 just to comply with the general point, for this will certainly be an interesting field in physical robotic FCM networks. Interlude question: how do you, in general, work out w/some certainty just how probable something is? That's something we often do in daily life, but how?

*Second triplet, first value: this tells, for match-oriented nodes, more about this particular match--eg its

type (or if it were matching over a sound wave, its main frequency; or if you had color-specific cameras, its color type, like Red=10, Green=15, Blue=20). Here, we might as well have the match image number here, 1..14, and it saves us from 'hard-coding' the match number into the function of the match node. In other words, we can have the same function in all the 14 main match nodes and it fetches the match image number from this field. This is pos#13.
*Second triplet, third value: allocated (but not used here in this virtual robot FCM software) to deciseond (ie, the quantity of tenth of seconds since reboot of G15 PMN or at least startup of this program). This is pos# 15.
*Third triplet, first value: 1=match over the main input image, 5=match over simulation input image. In this app, the main input image is that which shows on the GPS screen to the interactor with the program. The simulation input image is a copy in RAM, not yet written to C9000, which is being 'experimented with' by the task nodes and matched over again and again so that the most suitable elementary action (with the highest level numbers) can be picked out. After each experiment, the original input image is copied again and again back over the simulation image.
When the best-performing elementary action has been identified, the elementary action is done at the actual input image and it is written to C9000.
*For all nodes, as said, we let the value at pos# 21 be greater than BASIS in order to signal that its functions actually shall do something. For the match image nodes, the value 5 is used to indicate that simulation copy should be used; DANCE means that the main image is used.

*Task-oriented nodes:
First value of fifth triplet: the next step--the first being 1--to be performed, or which has recently been performed. This is position# 22.
Third value of fifth triplet: DANCE when the step has been completed; set to BASIS by the higher-priority task calling it when this subtask node is activated. This is pos# 24. This allows the higher-priority task to call on matchings over a simulated elementary action and fetch the results before the next step is called on.
Sixth and seventh triplet: first and third values in these two triplets are set aside to more timing things, again in deciseconds--something which advanced physical robotic contexts have to have ample supply of.
The other triplets are used as seem fit, and both match and task oriented nodes can use all the luxury values. In case these are not enough, data-oriented supplementary nodes can be used, see such as FNAY, FNYAY and those sorts of functions for how to structure these; most of these are well-tested by virtue of the GPS which constantly uses them.

Here are the nodes:
Level number 100:
A node that, when activated, sets NEXTNET to GPS.
We use the 'soft activation' field# 21 as flag.

This is typically set by a high-priority task node that has determined that enough has been done that it's time for an update on the GPS screen again. This node also, neatly, resets its activation field. For more complex FCM robotic network approaches, there are many nets in a list.

Level number 500:
A node that always checks whether NEXTNET is different from THISFCMNET, and, when this is the case, adjusts THISFCMNET. This is always active.

Level number 1000:
14 nodes that checks either the main image, or the simulation image (ie, the copy of the main image with a proposed elementary action performed on it), for match numbers. These can be activated to do their matching job all at once, ie, in one cycle. Since they are algorithmic, and since they will only be activated by a higher priority task node when the image is ready, they can be assumed to have done their matching job after just one cycle.
The first triplets are updated; described above in this chapter. Soft activation field is at pos# 21 and in this case, value 5 indicates use of simulation image.

So pos# 21 also tells which image to match over, one=the main, 5=the simulation. As a courtesy to the next cycle, it also wipes the field to basis.

Level number 10,000,000:
The high priority task node in this case, "let's improvise this by elementary actions in the direction of one of the fourteen match images". This being in software and also being amongst the most simple yet full robotic FCM node networks possible, we do not set a node here to be marked by its #48 as an 'ethical node, top priority'.
We do however increase the level number much so as to give room for higher-level match nodes in the FCM networks you make yourself derived from this.
In this app, we do it somewhat simplistically and with a lot of iteration through the TRANSLUCENT loop: every possible elementary action is tried each time on a fresh copy of the actual input image over to the simulation image, and every match image is matched against each such possible step, and the winning elementary action is called on to do its job on the actual image and it is stored back to C9000--only at that point will the GPS show the result of the cycles. By a click on a button on the GPS, the next update of the image occurs. And by that, the interactor can see, directly, first-hand, how the computational node network does its job of approximating some of the fourteen ideals by a well-defined if rigid set of elementary actions.
When the high priority task at level 10,000,000 has completed its calling on the best-performing subtask on the actual image, it calls the node at level 100 to set GPS as NEXTNET. The node at level 1000 will then ensure that TRANSLUCENT switches 'computational attention' to the G15 PMN FCM Spreadsheet.

Level number 1,000,000,000:
The elementary actions. We divide the main image up, conceptually, in a set of horisontal lines; the height being 112 so this can be a couple of handful lines. The same or similar number of pixels are used in the horisontal direction so an elementary action is then all about, at a particular 'line', to swap one rectangle or square of pixels with another at two positions on the same line. Which swap it is about is organized through 'step numbers'. The step numbers may refer to all possible swaps on a line, in steps given by this decided set of pixels, or a subsection of these. We'll see when we program what seems feasible and elegant enough and which

does work out well exactly how many pixels and how many steps and how to organize them.
The use of the triplets is described above in this chapter; and that includes a field value that determines whether update is on the actual image or the simulation image.

That should be it!

ENTRAINMENT AND SOPHISTICATED GOAL-SORTING

We're now ready to begin to program. When you have in mind a design where many components depend on other components, also of your design, it is usually essential to get it to written or at least visually sketched form before beginning the actual construction. In the case of programming, construction involves paying attention to syntax and the ins and outs of warps, arrays, matrices and what not. When you are in the midst of all that, you want to get this right while just having to gaze on a 'commonly agreed' sort of construction map for the general direction of what you're doing. It is a very technical mode of action, to concretely implement something requiring a very high degree of precision of a multitude of factors: and this technicality sort of overtakes much the same aspect of the intellect that deals with the overarching design: and so the design should have been completed. In terms of feeling, getting the design expressed and sensing that it has a completeness and an adequacy about it, or even something superb, involves a great relief--practically (or even biologically) similar to getting out of a severe headache. So it is with this relief that I will now sketch the concrete programming steps for this app. I will sketch it in all essential parts but only as far as it seems meaningful to do, not filling up with technical trivalities of a kind that we by now should have handled well enough by the earlier volumes, and which in any case are all fully spelled out in the app #1005768 that you can and should look at while looking at the forthcoming program sketches.
Before we begin, let's consider briefly how the above sketched FCM computational node network for robotics could be expanded to entrainment, and to more sophisticated forms of goal sorting:
*Entrainment. While we do not want a program to in any way 'change itself' it is of value to create helpful auxillary programs that, say, in a database (ie, a list of numbers and/or characters stored on disk in a way that can be systematically retrieved and/or changed by an algorithm) stores such as a range of match numbers for certain objects eg shown to the robotic cameras in various light intensities and from various angles. Such an "entrainment program" may even propose certain ranges of match numbers that is indicative of such-and-such object; these ranges may be stored and opened up and quicly inserted into the respective node triplet values of match-oriented nodes during startup of the robotic program. The programmer will sit through such an entrainment session and optionally modify and perhaps confirm certain data sets; and then proceed to test how they work with the concrete FCM program; and after such an entrainment, give the program the 'checked and ready to

produce' stamp. It is his or her own personal stamp; it is the programmer himself or herself who feels that the program adequately expresses a mindful, ethical intent through the software and its associated machinery. Such entrainment must never be considered a tool for (the self-contradictory notion of) "machine learning". It is just a pleasant way of configuring a human-made program with proper settings, that's all.

To do entrainment as an extension of the above-sketched FCM node network would then typically happen by there being more match-oriented nodes in between the fourteen which are there now, and the high-priority task node. In that layer, one would label nodes eg to what sort of phenomena, human beings, other living beings, things and processes, that one would expect the camera to get a glimpse of in a typical run in this sort of context that the FCM robotic program is supposed to perform in. The names of these nodes could reflect these objects. To add more sophistication to these matches, there would be more levels of nodes between the objects and the match numbers, allowing links to do such as suggest 'mutually exclusive' phenomena. For instance, if one object is named something suggesting 'hot cup of coffee', and one medium-level feature is named something like 'cold', that feature would act to negate the likelihood of there being a hot cup of coffee at the same location, at least if the feature 'cold' was associated with a high degree of probability while the object 'hot cup of coffee' was matched on with less probability.

Entrainment programs would save time; each match number combination and probability can be typed in manually in the general case. In some cases, such as using a computer to decode some form of language it may make sense to build up a considerable database of numbers--and the programmer who is wedded to the first-hand approach will then, in such cases, ask himself or herself: is this data set obvious enough, human readable enough, or have we gone over the rails into a statistical handling of material that deserves a more first-hand approach? So, for instance, matching over typewritten letters in books in order to get a paper version over and into such as a B9edit text editor readable form is a case where a rather large database of character-matching data would make sense and can still be argued to be part of a first-hand approach. But a matching over a large number of human movements on a street in order to program a street-washer robot is a second-hand approach, for it would be taking an extremely diverse and also extremely ethically sensitive domain and try and compress the situation into something that one can let an algorithm 'solve'. This is uncalled for: if one doesn't have the time or aptitude to manually, in a first-hand way, program a robot to wash a street without crashing into people, one should not put a robot onto the street in the first place. A bit here and there in that programming can be helped by some sort of entrainment program, but the overall FCM programming must be done in a first-hand way when we talk about lively contexts.

*Sophisticated goal sorting: The computational FCM node network for robots that we have here has in it the exact type of nodes, and levels, and network-switching capacities, that allows its present form of elementary goal sorting to be rather effortlessly expanded, especially when more levels are added to the match nodes. The present FCM program sorts and locates the most relevant task or subgoal by means of comparison of the

present state of the input matrix with fourteen match matrices, that in that sense represent fourteen possible goals that can be (usually only approximately) reached. Adding more match nodes at a higher level, between the present match-oriented nodes and the higher priority nodes, automatically opens up for new and more sophisticated goals. On the other hand, grouping subtasks into gradually larger tasks, and allowing the simulation to branch into several subtasks performed--with new matches done on the result in each case--allows for such as second, third, fourth and even fifth and sixth level evaluation of task alternatives in a simulated fashion. This is all a matter of adding nodes, adding algorithms, adding links, and, above all, sketching how you want the triplet fields and such to store whatever it is to be stored to have the sorting material ready.

In some cases, it may be of value to entertain more than one 'master path' of action, so that the sorting could involve protecting data for the winning range of possible actions. For that, such as the fast two-letter function, implemented directly in G15 assembly, called BS, could show which of the possible actions have the best relevant match numbers. In some cases this could lead to a shifting of level numbers; and the BS would also be used to sort that particular FCM node network to get updated execution sequence.

The sense I have of the program now is that it is becoming more and more real: the sense of its 'substance' is there; and while this volume perhaps have become more technical than I had originally in mind, it is perhaps also reaching out to those who wish to excel in FCM programming with G15 PMN as never before.

Next chapter is all about coding, using the map of the app as given in a somewhat list-like form in the previous chapter.

The resulting program, when it answers a challenge in an elegant enough way, and turns out to work really well, and also has orderly, meaningful, first-hand understandable cards, is a gem of order; one can rightfully feel pride and engage in a sense of celebration when a program is, in that sense, complete. The pathway to making the program may involve a lot of back-and-forth and checking for this and that definition here and there and syntactical or other errors can be in the first version and may appear at first to be needle-in-the-haystack complicated and yet, almost always, supersimple once seen. A robotic FCM program is seriously "high level": PMN stands on top of G15, the definitions of FCM come about last in the 3rd Foundation, upon which GPS, the spreadsheet is built; and this program is on top of that again. It is all in a sequence so that there never is any doubt where a definition might be: it is always prior to its first use; and can be, in G15 PMN, found by the SCAN search in a blink. Building each card requires both a sense of the concept of the whole program, the exactitude of the G15 PMN syntax, and a sense of a kind of poetic interplay between the two columns of code, in black with its superb coding font, the G15 native Robotfont.

We need to define the matrix NETRBT. When we have a look at the definitions of FCM we see that some of the words used to construct a net involves THISFUND and NEXTFUND--and these requires reset as they are counters. These little things--but significant to get right--involve a bunch of cards in the beginning. I'll give some excerpts

of it, some comments at some points, and when we get on with the code, I'll write inside the function (or whatever) what it is it shall do rather than doing it when I sense it can be too tedious detail to spell it out.

MAKING ROBOTICS FCM PROGRAM:PREPARING MATRIX

In a previous chapter, where we first talked of the 'courteous toggle' that the networks should do in their first (ie, most low-level nodes), we said that this toggle is going to go in at the first nodes of the GPS. This, alongside some modifications of the GPS so it better works with the Batch Graphics version of G15, I will put into the GPS when the time has come to make the cards--see the resulting form of the GPS in the App# 1005769 and compare to its original form in the App# 3555558 to see the differences--it is just some cards. And the GPS may be, of course, further modified in other robotic apps in the future.

In a sense, the 'first' nodes of a FCM network are those with level numbers nearest 1, the DANCE.

How the GPS gets new node inserted 'first': This can be done by going into the GPS source itself, or we can simply make the nodes after all the other nodes in GPS have been made, with really low level numbers--before any of the other used level numbers in GPS. The one thing we must remember to do also is to resort the FCMINDEX. I will not bother this text with that little technicality, because you'll see it easily enough in App# 1005769 soon after the start of its code on disk F, and because the content of these nodes is largely as the content of the first nodes in the robotic node network we're about to set up.

NEXTFUND & THISFUND are used when making new node nets, and need reset. When making the progarm cards I will look at these notes, and at the moment of typing figure out how to do the layout over the two columns.

First, we run the 'SAVENETVARS'--save net variables to the miniarray to keep these as we described above.

```
NEXTFUND
BASISTHIS
THISFUND
BASISTHIS
```

We're going to make the matrix itself, and the rule of thumb is to give it extra bytes; something like this--inspired by the initialisation of the GPS--should work:

```
NETRBT=
^.
MAXNETRBT=
1000.
```

This constant, MAXNETRBT, says a thousand funds would be nice but we're only considered a few dozen so this is a rich world that allows such numbers without a thought.

```
MAXNETRBT
150
MM
200
AD
SZ
```

This should mean that the next && quote gets enough for a matrix. The 150 is as defined in Third Foundation, TF.

```
  &&
  NETRBT
  KL
```

That's it! When making new funds or foundries, we use such as FNEASYACT which sooner or later calls something that calls MAKEFOUNDRY and which in turn initializes all the 150 number positions of each new triplet properly.
Next, let's configure the matrix for speedy calculations:

```
  150
  MAXNETRBT
  NETRBT
  WWYYMATRIX
```

Now FUNDNET is a variable that has the GPS inside it. We have planned to let that be the case; it is THISFCMNET that's going to alternate. And THISFCMNET is also used when constructing the net:

```
  NETRBT
  LK
  THISFCMNET
  KL
```

We're going to need a sorted index and the quantity in that index, and here we follow, doggedly, how the GPS does it, just with other names than FCMINDEX and FCMINDQTY. We might compress 'INDEX FCM NET ROBOT' into something like this:

```
  INDEXNETR=
  ^.
  QTYNETR=
  ^.
  QTYNETR
  BASISTHIS
```

The QTYNETR we'll set to the exact quantity once the FCM nodes have all gotten their level numbers etc and we know how many there are. We give the INDEXNETR the warps to the matrix exactly as how FCMINDEX got its stuff:

```
  MAXNETRBT
  50
  AD
  SZ
  &&
  INDEXNETR
  KL
```

And:

```
  MAXNETRBT
  THISFCMNET
  LK
  INDEXNETR
  LK
  INITWARPINDEX
```

Also we need a helper variable to do the toggling between the nets. We have before called this 'NEXTNET'. It could have as initial value the GPS, which is same as FUNDNET is set to from the start here.

```
  NEXTNET=
  ^.
  FUNDNET
  LK
  NEXTNET
  KL
```

That's about it, to launch a second FCM node network when we already have one good going! Now let's give it flesh.
In the next chapter, we start with SETFUNDLEVEL and indicate what sort of functions we should equip the nodes with, as we proceed up to higher and higher levels.

MAKING ROBOTICS FCM PROGRAM:THE FIRST NODES

When you look at any FCM node network, including the GPS, you'll often see that functions for these nodes have the slightly cryptic articulation IN:TR#,FNWARP as the first comment line. This is more spelled out in the docs for the Third Foundation, which of course is App# 3333333. It means, input to this function--the function being placed in one of the ten triplets of one or more nodes--is TR#, ie, triplet number, and FNWARP, ie, warp to this very node or fund that now is being performed by the TRANSLUCENT loop. A function may use one or both or none of these inputs. If neither are used, you likely find two calls to SH, to 'shuck' these two inputs. But if a function is going to address a triplet value or links or something in a node, it is generally the right approach to use the FNWARP input rather than 'hard-coding' the warp for the node inside the function--it gets more flexible that way.

Let's do the toggle. A couple of chapter earlier on, we talked of level number 100. Here is the same stuff, translated in the direction of honorable G15 PMN language:

```
  100
  SETFUNDLEVEL
```

We're going to need a tailormade algorithm for this node, and these are called FNACTs and registered via a number--I will have to check with the GPS that the number isn't already in use in each case (just some dozens are in use), and the word for registering them is FNACTCHERISH. The 3rd foundation suggests 5000 is enough and when it is, we don't have to change the definition of FNACTLIST, the array that has the number-to-warp conversion used by the essential TRANSLUCENT loop. Here's a sketch of the function, where 'soft activation' flag is at #21:

```
  TOGGLETOGPS=
  |IN:TR#,FNWARP
  TX
  SH
  21
  JX
  WK
  N?
  SE
  EX
```

And now it is clear the 'soft activation' #21 is DANCE; which means we are not only going to let it do its job but also, generously, turn itself off so it only does the job once each time (if we use this series of program statements often we can make a function out of them):

```
  BASIS
  JX
  21
  KW
```

And now for the real action:

```
  FUNDNET
  LK
  NEXTNET
  KL.
```

And let's give it a robotic-sounding FNACT number--I just checked with the spreadsheet program that it doesn't use any FNACT number above 3999:

```
  &TOGGLETOGPS&
  4700
  FNACTCHERISH
```
That should take care of that. The 'WK' adds the two numbers, and then does a LK of the value there; when this isn't a DANCE flag, the SE calls the exit by EX. Otherwise, FUNDNET--which is the spreadsheet of course--is set to become the next net to be performed, NEXTNET. We'll soon set up the next node to do the actual transition. This is identical in the two nets, so it may be, when all comes to all, you find that definition in the finished app a quite an earlier stage in it--nearer the beginning of the GPS nodes--and so we just use the same FNACT numbers; and these numbers are typically a common ground between the set of nets. Let's use one of the helpful construction words, FNEASYACT, to set up the node so it's ready to be active; and then set the soft activiation field #21 to BASIS so it doesn't switch back to GPS before it has had the number of runs it should have. Note that if the net is to be restarted after exit from it, without loading the program afresh, one must look at each and every init value and be sure they are indeed have the right values for a restart or else make an init routine to do this which is eg auto-performed when exiting the program, before it is restarted. The usual start of the TRANSLUCENT loop is via the word FCM, but you can also make a separate word here for starting a particular set of nets.

The FNEASYACT ought to get the three values of the first triplet, the second being a registered FNACT, and also the name of the node, which doesn't have to correspond to any function name. Let's have a go at it:

```
  BASIS
  4700
  BASIS
  &OVERTOGPS&
  FNEASYACT
```
Note that any field we don't specifically give a value to after this call, is automatically set to BASIS; and we're fine with having 'soft activation' field set to BASIS here--it should be set to DANCE by high priority node each time.

And we're on to the next node, that carries out the switch-job properly--and I'm looking at our earlier chapter for level numbers and so on while programming--this FNACT I will put into the updated GPS code for it is used also there, as we know--and we'll just refer to its number, 4701, instead of this exact definition here. By the way, we'll use the 'Soft activation' position 21 throughout--also here where it is not obviously needed--just for consistency--and here it is set to DANCE:

```
  500
  SETFUNDLEVEL
  COURTEOUSLY=
  |IN:TR#,FNWARP
  TX
  SH
  21
  JX
  WK
  N?
  SE
  EX
```

And the essential 'courteous toggle' check:

```
THISFCMNET
LK
NEXTNET
LK
EQ
SE
EX
```

At this point, it will perform this in case they're differ

```
NEXTNET
LK
THISFCMNET
KL
SMARTTOGGLE.
```

This new function is about the other variables, mentioned when we first talked about switching between nets--the FCMINDQTY, FCMINDEX, and NEXTFUND. Let's gather the switch between the two sets--which in another FCM application certainly will be more than just two sets--into a function that does this with a bit of error check and which easily can be rewritten; we'll look at how to make SMARTTOGGLE after we have registered this function is a proper FCM net function, with a very readable number:

```
&COURTEOUSLY&
4701
FNACTCHERISH
```

And, --I use much copy and paste now and modify as we go along--the node, the fund itself, is easily set up:

```
BASIS
4701
BASIS
&MAYBETOGGLE&
FNEASYACT
```

And we ensure the 'soft activation' flag is DANCE:

```
DANCE
21
ADJUSTFUND
```

SMARTTOGGLE should have something like the following definition, considering the variables we defined earlier on when we first discussed switching nets, together with SAVENETVARS. In order to facilitate a little bit extra controls in case programming goes a bit fast when one tries new things in the future, and since this net toggle is used sparingly--ie, it won't slow down to put in an check in it and it will give more feedback in our future programming in case there's NEXTFUND has had a funny value--we put in some 'quality control' in this function:

```
SMARTTOGGLE=
|ACTION:WHEN
|THISFCMNET
|IS UPDATED,
|UPDATE THE
|OTHER THREE
|NETVARS
THISFCMNET
LK
SX
BASIS
T8
GPSNETVARS
1
AY
S3
RBOTNETVARS
1
AY
S5
```

Okay, that was a lot of 'preps': putting stuff into simple local variables in order to have snappier expressions in the following parts of the function: we have THISFMNET avaiable in IX, will set J8 to the right miniarray; have i3 with one miniarray and i5 with the other. If not, let's have it print one of the world's shortest error messages, &??&. I have long avoided what I will now do: to count just how many positions in a functions such a two-letter size quote occupy; I remember I decided, making PMN, that at least some quote-types should have more rather than less around them so a program might work even with a poke into a quote by such as YA, KL or KW that is a bit out of bounds. So how long could it be? Let me see: we have got to have what we can call a QUOTEWARP. We want the length --we want the quote itself, two ?'s, and we want a NILCHAR --adding that up we have 1+1+2+1=5. It is at least that. By a clever combination of FF and AY on a few example functions I type in and 'hack' into at the terminal, I will decode my own code (not for the first time):

The answer is 6. For speed of performance, the PMN code has a quantity inserted before even the length number that tells PMN how many positions to jump over to get across to next instruction, and that itself occupies one more, so 6. That means that if you wish to do &??& followed by PP on a condition, you could use D7 or D8 with a TN, the TN being a useful, pleasant fast way of putting in an extra code that does nothing except making a column more readable, which, in terms of human meaning, is of course doing something. That means, to put a two-size quote inside a D-like 'deliberate a jump' call like D7, you could, for ease of reading, make a |QUOTE comment where you also tell the length, plus four, of the upcoming direct quote in a function. You might do it like this: |QUOTE:6 POS

This concerns any &..& quote, or any ^.. quote, that appears inside a segment jumped over by something like D8. These quotes we can call 'direct' quotes, unlike the quotes whose warps are stored in some variable, and which are treated straightforward in this context.

SO ADD 4 TO THE QUOTE'S LENGTH--eg, for a quote like ^XYZ of length 3, we have 3+4, so eg: |QUOTE:7 POS

And that's how to use any of the two-letter functions that start with 'D' and that 'decides' to jump over a certain quantity of positions, up to 16 in the TF, together with one or more quotes. Another way is to put the warp of the quote into a local variable and just use it. When you have the warp to a quote, short or long, eg set to TX/JX, all

you need is to put JX on a line and there is no need of any calculation around it.

The connection to how the machinery of the programming language works in the background, and the assertion of the mind-stimulating value of doing arithmetic with whole, natural, simple numbers while programming, all enhances the first-handedness of your work when you do G15 PMN.

Let's get on with the function before I chat too much; remember that calls like D7 or D8 'decides' to jump given a DANCE input. J8, through T8, must become one of the two nets in this two-net switching situation:

```
J8
i3
EQ
N?
D2
GPSNETVARS
T8
```

That takes care of one alternative. We go slow because this routine does a bit of 'quality check' as said.

```
J8
i5
EQ
N?
D2
RBOTNETVARS
T8
```

And now for my first-time ever use of a quote together with something like D7 or D8; it has size 2 and so we add 4 and write 6 POS as shorthand for 'this quote, while it is written on one line, takes the equivalent of six positions or lines':

```
J8
YE
D8
|QUOTE:6 POS
&??&
PP
KK
```

This should do. It means that the program won't go on without eg an ENTER-click if this doesn't get right. This means that the programmer will likely get this message as soon as there is a mix-up and long long before app is put to production; it may mean the guy has to turn off the PC & back on but then the constant advice is: do a reboot of G15 PMN so that all disk-writing is complete, before starting a new program that might lead to a PC stop. In that way, disk integrity is protected.

The rest is, by comparison, excruciatingly easy:

```
J8
LOADNETVARS.
```

The LOADNETVARS only does something given a meaningful input.

According to the list of level numbers in an earlier chapter, we're on to level 1000, ie, the veritable match numbers themselves, as the next coding bit. We're getting on!

MAKING ROBOTICS FCM PROGRAM:SIMULATION MATRIX PREP
In the next chapter, we create the fourteen match nodes. Here, we blaze an orderly trail for just that!

The App# 1005768, made just prior to our work now on #1005769, has a function which is useful when the GPS is to show all the fourteen match numbers relative to an image which is already loaded. Function PATMATCOMP loads specified match image and compares. This comparison is done between MATCH1IMG, which we can call the 'main image' --of which we're about to make a copy so that we can 'simulate' various 'robotic actions'; and MATCH2IMG, which is loaded from disk. To save disk time, a future and rather obvious optimalisation of any repeated matching over the 14 match images is to store them in minimalized form, ie, as 112x160 matrices, rather than load them from disk through their conventional GEM image form. The key to the matching itself is the two-letter word RO, which was added to the Third Foundation set of standard two-letter functions in the course of writing the earlier volumes. It can be thought of as a mnemonic for, namely, RObotic matching over matrices, and it is superbly fast, written in G15 assembly, and just what we need for exact comparison count.
To get on with our stuff, we use PATMATCOMP here, but we are aware that this can be made faster in case a later production use of this node network indicates that the disks are buzzing needlessly much when accessing match images. (Whether or not that 'buzzing' is meant literally depends on the style of disks--some are, of course, not magnetic but rather like RAM only handling power-off/on.) Here's how to make it faster: make fourteen matrices, each of the WWYYMATRIX type, each similar to MATCH2IMG in size, and line them all up in an array--an 'array of matrices' is an elegant concept, and--since we all here in the G15 PMN 'league of nations' know ultra-well the warp concept, we know that this is a matter of putting the warps of the matrices into the array. Then 'clone' the PATMATCOMP into eg PATMATCOMPF, with 'F' added for faster, and input the position in the array instead of the position on the disk.

The one tension I have when writing this--and commenting on the process of thinking is presumably part of the art of exploring the process of thinking--is that I wish to do one card after another, and test that it compiles well step by step, and experience the sense of full order--whereas with each chapter now, prior to the programming, we're building up 'tasks' for me to implement in terms of programming and with only partial description of how to implement them. Nevertheless, knowing that the sense of order--that feeling of prevailing overview, oversight and harmony that comes with a really well-working program, well-written and performing smoothly like an infinitely enduring machine--all that will come, soon, makes the writing peaceful enough.
At any rate, let's make MATCHCOPY, a new matrix with identical structure as MATCH1IMG, and a storing-place for the original MATCH1IMG, MATCH1ORIG, so that it is easy to return to after the FCM node network has done its "simulation work" on the copy of the main image, which in our 'virtual robot world' represents the status of affairs faintly as could be inputted via a robotic camera.

```
MATCH1ORIG=
^.
MATCH1IMG
LK
MATCH1ORIG
KL
```
That takes care of that. We can now switch MATCH1IMG to whatever we want, such as the MATCHCOPY we'll make next, and switch it back when we're done.
```
MATCHCOPY=
^.
160
115
MM
SZ
```
This is, as you see, 3x160 larger, in terms of positions, than 160x112, and so plenty to allow WWYYMATRIX to do its normal structuring. Onwards, the SZ has now configured how much of RAM the && next will allocate neatly:
```
&&
MATCHCOPY
KL
160
112
MATCHCOPY
WWYYMATRIX
```
Done! Hm, we're ready to code the nodes now I think: these should have two values in one or another of the first triplets not already assigned for something: the first value can be a flag that says whether to use the main image, DANCE, or the copy image, BASIS. The other value we should store is the number 1..14 that says which match image it should compare with. In one of the first triplet values we'll output the result, recalculated from the match value result, which goes from 0..17,920, and over to our beloved PERMILLE, ie, from 0..1000.

Why PERMILLE? And why has G15PMN nothing of the percent-symbol that is typically in classical Ascii? Answer: to line up everything on a scale 0..100 is a bit stupifying, in my arrogant opinion, because, as a start, we need at least an additional digit in resolution for more refined thinking. And I mean to see that much of humanity has got stuck in statistical thinking around percentages for no other reason than a meaningless, never-planned convention that better be altered--quick--so we get more wisdom in thinking, planning--and in statistics. That's why PERMILLE --besides, I wanted to get rid of any symbol that appears to 'divide on zero' because it reminds of the unsmart way infinities have been handled in the past. So the new symbol in the Ascii 7-bit character set looks far more like a growing tree-branch or even flower and is used profoundly in the CAR-menu system of G15 PMN.

An abbreviation for permille, suitable for writing, is 'pmille' which can be pronounced 'permille'.

Now the fourteen matching nodes themselves. Sequence of performance in between them does not matter, as long as they are all done before we get on to the next level. It's perhaps worth nothing that a sort routine, such as in particular our favourite BS, can reorganize sequence between items that have identical sort-key quite freely. So with 14 nodes all having the same level number, after BS has performed--and it or something similar must be performed each time the level numbers aren't sorted in FCMINDEX--their sequence may be anything at all--and not necessarily the order which we typed them in.

MAKING ROBOTICS FCM PROGRAM:MUSING OVER MATCHES
As the saying goes, there are always many ways of doing anything in programming. We have a 'main image' and a 'copy image' and the first time the 'copy image' is being matched over, a loop should copy the main image over to the copy image, number by number. Where should we place this copy-loop? Let's think.

There are two places that appear fairly obvious, one is at the match nodes themselves. They get a flag (in a triplet, we must decide which) and when it is something matching goes over the main image, and when it is some other thing it goes over the copy image.

So one way could be, let's check whether the present image, MATCH1IMG, is set to the copy image when the flag suggests so. In case it is not, the match node function--which presumably can be the same for all fourteen match nodes--can take that as a hint that a copying from main image, safely stored at MATCH1ORIG, to the MATCHCOPY should next happen--and the MATCH1IMG is set to MATCHCOPY. I feel like this may be right when I write it but it also seems right to have given a thought to doing copy-loop somewhere else, namely at the higher priority task. It is the one which is going to locate the best pathway of action; it will start making arrangements for generating match numbers of various 'simulated actions'.

Each time a subtask has performed its job on the MATCHCOPY image, the task nodes are going to present the results of this. So, it would seem like the copy of the main image should be made before each subtask--in this case, before each "elementary action"--and it seems very clear that this copying cannot wait until match-time.

Often, when something feels right yet other approaches seems more right, I find that that which feels right may also seem right but only after that which seems more right is understood so well that the aspects best taken care of in that which seems right are also handled well through doing that which feels right, --or, as an alternative, that looking at all these aspects together, the feeling will shift and that which seems more right will also feel right.

Let that be Proposition# 1 of 1 in this volume about the Art of Thinking as such :)

Looking at where the copy loop should be placed, whether at the match node level or nearer the task levels, the feeling has now shifted, albeit slightly, in favour of the task level or levels. My sense of it is that we pursue this path but are ready to change if something appears, during the programming, to indicate that it's better to do it, somehow, at the match level after all.

By this assumption, the match nodes are implemented with the sense that the matrices are already all right--as far as their contents go. But the check as to whether MATCH1IMG is set to the original main image or to the copy should be done, and the setting adjusted if need be to perform right relative to the flag as to whether it is the main image or the simulation copy the match is going to work on.

And it is indeed meaningful for the match nodes also to work, sometimes, on the main image--not just the simulation copy; and that is, of course, because the task node is concerned with _raising_ the match values by a suitable smart subtask and so it has to have the original value, not just the simulation copy value. Technically, some minute computational time could be saved by fetching the main image match values from the spreadsheet instead of calculating them afresh, but it is snappier to program a fresh calculation, and the computational time is mainly in the many cycles where the possible actions are tested. So, when coding is more elegant, perhaps much more elegant, and the computational time involved is just slightly higher in the overall context, usually the elegant way should be chosen. Programming is about beautiful action.

About beautiful action--good programming--let it be clear that a good programmer has a lot to think about, simultaneously. The formal language should be, if possible, ruthlessly clear about what is being expressed but also fairly effortlessly to read in sequence. Like a good novel, a well-written algorithm in a good programming language can be read, and read in sequence. Now however the more there is of such things as 'conditional performance' of segments inside an algorithm--and this is accentuated when there are nested conditions (segments within segments with a conditional performance)--the more there are of 'jump-points' inside the algorithm. And these jump-points cannot that easily be 'read' in sequence; they have to be visualized, while reading typically must go up and down and re-reading of earlier parts must take place, something which is not eradicated by a hierarchical ordering of indents in code (something typical in earlier programming languages, before G15 PMN0.

For such reasons, G15 PMN is all about small algorithms in which there is relatively little of such jump-points; and where they are, they are meant to be laid out neatly in the two columns of cards by such tools as D2, D3 and so on; or the single-line form (that doesn't jump over a line when DANCE, but only at BASIS, unlike D2, D3..). And instead of the non-beautiful 'grouping of statements' in languages such as C, which uses { and }, or Pascal, which uses BEGIN and END, a small positive natural number is used instead--because such numbers are very much first-hand in every sense.

A detail note on programming style using D2, D3 and so forth is that, after TF (Third Foundation) was made, it repeatedly made sense to be aware of exactly how many 32-bit number positions are used when PMN code is compiled by the underlaying G15 assembly. And for two-letter words, or three-or-more letter function words inside a G15 PMN function--in all cases where these are neither loop jump-points like LO nor conditional performance jump-points like D2--the quantity of number positions is one--a single warp. But when a number is inserted, the quantity of positions are two: one for the warp that handles numbers, and one for the number itself. For quotes, it is four plus the quotelength itself; see our discussion of this near definition of LOADNETVARS.

The consistency of how numbers are treated allows the quantity of positions--which D2, D3 and so on are about--to be calculated. For quotes, see what we wrote about the |QUOTE: comment above, we add 4 positions to its length. For the far more native structure of a 32-bit number, all we need is one more position--the 'numberwarp'--and

this warps goes to a G15 function that puts the following number to the stack, and updates the present address of what is being performed by one extra position (ie, as a 'jump' over the number that is going to stack):

```
|NUMBERWARP:
314000000
```

So, if inside a function you use D2 to 'deliberate a jump given such-and-such condition' over two positions, and after this follows the number 314 million, you could write a comment line |NUMBERWARP to remind yourself that not only is 314 million occupying a single number position in the function, but there is also a warp to handle this number, ie, get it to the main stack, just preceeding it; and D2 should count this as equal in size to two normal function words as far as positions go.

So, in the opinion of me as creator of G15 and G15 PMN, the constant work on counting is refreshing, delicious and healthy, and indeed reinvigorating in a meditative sense, as well as a boost to intelligence, for a programmer. But while counting using integers is a virtue, the visusalations of hierarchies of such as nested conditions should not be enforced upon the programmer who after all have so much to think about in order to get any adequately complex program up and running. Hierarchies are not wrong but they are not novels. A program is, in a way, a novel in a secret language.

MAKING ROBOTICS FCM PROGRAM:MATCH NODES & INFINITY

I looked up the level description several chapters earlier and saw that, entirely in contrast to how we describe it in the previous chapter, it speaks of the soft activation field, #21, also to be used for specifying whether the image to be matched over is the main image, 1 meaning main and 5 meaning copy image. It's part of thinking to allow plans to fruitfully fluctuate a bit. I can see that this has the merit of saving up more triplet fields for future uses of similar match/task nodes, so let's go for this, after all. These are all at level 1000. Let's see what we can do here, which fits with the level description several chapters earlier, and also with the variables we have set up including MATCHCOPY and MATCH1ORIG and such:

```
1000
SETFUNDLEVEL
```

To pick a number above 4700--we've used 4700 and 4701 as robotic FNACT numbers in the previous levels--we can chose 4710 for next FNACT. This will interact a bit with the node triplet values. In an earlier chapter, same as where we dicussed levels, we discussed triplet uses for match-oriented nodes. What's relevant here is that main value at pos #10 has PERMILLE, 0..1000, and in this case a 1000 reflects a 17,920 complete match. The extra value, at pos #12 is here set to 1000 for 'full probability', but not read in by the function. At pos #13 we have the type#, which here is 1..14. This is ready by the function. And we have of course 'soft activation' field #21 which is 1, for main, 5, for copy image, and BASIS when the node should wait before performance. Let's set up these values in an initial state before making the function, for the first match node, where type# is 1.

```
  BASIS
  4710
  1000
  &MATCHIMG1&
  FNEASYACT
  |IMAGENUM:
  1
  13
  ADJUSTFUND
  |SOFTACTIVE:
  |0, 1 or 5
  |FIELD# 21
```
Since newly made nodes have fields not adjusted already to BASIS, that applies for field# or pos# 21 as well. Let's make a function:
```
  MTIMGNODE=
  |IN:TR#,FNWARP
  SX
  SH
  21
  IX
  WK
  T1
  J1
  N?
  SE
  EX
```
At this point, supposing the value wasn't BASIS, the J1 has either 1 or 5 in it. This will determine whether MATCH1IMG should be set to MATCH1ORIG or MATCHCOPY. Let's define in a function--before this function, when it comes to cards, but after here in this text, since we're sketching now, called SETMTIMG, Set Match Image. It can have the value 1 or 5 as input:
```
  J1
  SETMTIMG
```
So far, so good. Let's get the type#, ie, the match image number and at once convert it into the location of the match image, as expected by PATMATCOMP. The disk# is 6:
```
  6
  IX
  13
  WK
  DC
  220
  10000
  AD
  PATMATCOMP
  S4
```
Now i4 (cfr the node in the beginning where we say that we try to consistently use lowercase 'i' in the much-used variable name to avoid the confusion of digit 1)--it has the match number result. 17,920 means 1000 pmille. To figure out how to do this conversion with just whole numbers, I'll play for some minutes direct at the G15 PMN Third Foundation terminal, interactively with some calculations. Alright, we multiply by 1000 and then divide by 17,920, using rounded division RD. I checked this with half the value of 17,920, namely 8,960, and it gave precisely 500 as answer; and checked also with BASIS, and that gave BASIS as answer. So here we go:
```
  i4
  1000
  MM
  17920
  RD
```

And this is going to be put into the main value.

```
10
IX
KW
```

In addition, we wish nodes that do a task at soft activation to turn themselves courteously off when it's done, and why not make a little helpful extra function here:

```
21
IX
RESETNODEFLAG.
```

That little dot is, as you know well, a huge dot--more a square, really--when seen in the native ROBOTFNT of G15. Here, it marks the completion of the definition of the main function for the match image nodes, MTIMGNODE. Let's register it and make it ready to use for the nodes:

```
&MTIMGNODE&
4710
FNACTCHERISH
```

The utility function we should put in some earlier card:

```
RESETNODEFLAG=
|IN:POS,FNWARP
AD
0
W
KL.
```

Now let's sketch that SETMTIMG which we referred to in name in this function, and so will be placed just before it in the cards:

```
SETMTIMG=
|IN: A
|ACTION: SETS
|MATCH IMAGE
|TO 1 MAIN OR
|5 THE COPY
1
EQ
D5
MATCHCOPY
LK
MATCH1IMG
KL
EX
```

When it's the original, ie, the main image and value is 1:

```
MATCH1ORIG
LK
MATCH1IMG
KL.
```

We're going to set up 14 nodes at this level, and they're identical enough that this could be done within a loop. But here we'll just write out the first ones, and we'll type them into cards either as a loop or by copy-paste of the cards with small modifications:

```
BASIS
4710
1000
&MATCHIMG2&
FNEASYACT
2
13
ADJUSTFUND
```

And the next:

```
BASIS
4710
1000
&MATCHIMG3&
FNEASYACT
3
13
ADJUSTFUND
```

And so on up to 14. That's the match nodes, all implented.

Let's use the opportunity, since this is within the larger context of exploring thinking as such, not just the very particular yet educational and instructive form of thinking called 'programming', to see how easy it is in natural language to go from a sentence having a numerical boundary--".and so on up to 14" to a sentence that involves our human, in-born, God-given capacity for deeper intuitive visions and visualizations: ".and so on" or, more pointedly, ".and so on ad infinitum". When we chose the latter path in language, to call in the infinity concept, we are no longer doing 32-bit programming, of course. But supposing we took something involving numbers and dwelled and pondered on this question: well, then suppose we let something involving counting _really_ go to infinity, what then with the finite numbers? We may see the infinite, like the human brain (at the subtler level) as an orchestra, jam-session like--Bob Marley reggae rather than a symphonic orchestra perhaps--but what, in this larger vision, when we think about it, is the essence or substance of numbers like 1, 2 and 3?

A recurrent vision I have when I ask this sort of question is that they represent a form of 'dancing interaction' between what we vaguely can call 'infinities' --in other words, that 1, 2, and 3 talk about how moving, mindful, living structures (of an 'immaterial' yet real kind) are relating to one another; or, if you prefer, how they are 'rubbing against each other'. The sexual connotation is intended--no, not as a joke. This is to some hilarious, to others--who have any notion whatsoever of the world's mythologies connected to structures of reality at a deeper level--entirely elementary, trivial-- the phrase 'no-brainer' comes to mind. It is a no-brainer that the nature of infinity is sexual--if you have any sense for instance of a not insignificant portion of the pluralistic creation myths in hinduism, and that's only one of several examples.

What does this mean when on the human level of living? As we have touched on in the previous volumes, but which perhaps can be said with somewhat other words here, is this: the highly interesting form of brain/mind/feeling/ intuition/empathy/intelligence activation that takes place when both the mind and the body in the sexual sense, through pulsating action connected to such as the nerves in--let's be specific--the clit/dick, and other erotically sensitive areas (which in some states of consciousness is all over the place)--is a reconnecting with both of what we most deeply are, as infinite beings, and with what reality is.

It is in this state of what we can call 'enlightened sexual practice' which may mean sex as self-love or masturbation and/or sex as love of other bodies in a sensual/sexual way, and with the mind engaged in perceiving, with fascination, a smorgasbord of intensely healthy, elevated human anatomy forms, that we renew our brains and bodies, and it is in connection with this state

that a human may also 'reach out to cosmos' and to its source, God and his angels or muses, with prayer. This prayer cannot be a command, of course. What has been created is not entitled to command that which is creating. At most, it can be a request; but the complementary feature, as important as prayer if not more important in many senses, is intuition in the sense of sensing what is right beyond one's own ego and the coherence of brain to stick, diligently and robustly, to the crystallized result of such genuine intuition regardless of pains and inconveniences in daily action.

Since we're approaching--both in sheer number of pages, and relative to the aim, viz., to finish the app# 1005769 with robotic FCM node network in a well-explained way--the completion of this volume, and since a number of tasks are ahead before the completing volume in this series, including steel-and-plastic-shaping work for getting not just one but twin robots going and with plenty of apps for getting them to do useful things for us; and also, to get an EEG apparatus going for real in which some of the propositions I have about the activation of brain during certain types of sexual activity can begin to be more deeply researched, I will use the completion of this chapter to go into two points which, at the moment of writing, is of huge concern to young people in USA and elsewhere, that of abortion and that of whether there is any point trying to save the planet which appears to be going into a spiral of extreme weathers, torrents of rain, floods, and extreme heat, ice pole melting, pollution, deforestation and what not, and wars on top of that.

For those hugely interested in the history of ideas of the 19th and 20th century, they will know that Bertrand Russell--who, in various and enduring ways, took to believe not in the 'science' or 'metaphysics' of Karl Marx but in various toned-down political ways inspired by marxism such as socialism--was also a core influence in the evolution of so-called "atheist thought" in the 20th century.

In a series of essays and books, and with his often fearless and slightly witty logic, and with a razor-sharp sense of the English language, he 'dissected' all major philosophies he came across. However, works by Kurt Goedel disseminated the standing Russell had in pure logic, and the fumbling Russell showed in trying to relate to and have a discourse with the younger charismatic thinker Ludwig Wittgenstein left one with a feeling that Russell had got his essential tenets somehow wrong. Wittgenstein, in my interpretation, tried to outdo Goedel's second Incompleteness Theorem (which hinted on my understanding of infinity as beyond reach of human formalism) by going 'around' Kurt Goedel. But for all his efforts, he didn't seem to erect anything such as a new grand vision of the universe, which, prior to the 20th century, and, in my opinion, still and more than ever, is the duty of the philosopher (more than the physicist, though some were both, such as David Bohm).

Those who carefully read the biographical notes of Bertrand Russell will be able to trace this passage, which he wrote when he commented on a short book or pamphlet he had written on the sense of infinity: that he would not have written it had it not been for the influence of a certain lady. In other words, the atheist Russell had a phase in his life involving what he termed "mysticism", and he ascribed his fascination for this non-atheist topic, in which he breaks with typical statements on the finite and the infinite, to a phase where he had what I

can imagine to be fantastic sex with a beautiful young woman whose mind was very meditative and compelling.

My interpretation of all this is that most of his life, Bertrand Russell wasn't sexually activated in a full sense and that was part of his typical flat denial of the significance of the infinite.

Turning now to Russell's flirting marxism, we note that that Marx talked of what he took to be the future, in political terms, and that Marx spoke of two phases: a first phase in which "people take things back" from the superrich and their wolves, in which there is hierarchy, and a second phase in which there is a deeper communion and less hierarchy--thereby "communism". If we for the moment let all the thousands of rediculous assumptions that Marx added to this be dissolved, and remember also how drastically ugly the leaderships eg in Russia and China have been insofar as they have tried to implement any form of "communism", then these two phases--don't they make some abstract sense? As I thought Russell had written but, in now checking it: it turns out it was Niels Bohr, the danish physicist, who said this (possibly in German, but he also wrote and lectured in English if I'm not mistaken): the opposite of a fact is falsehood, but the opposite of one profound truth may well be another profound truth. That, or the corresponding thing in German or Danish, is a quote attributed to Niels Bohr. This ties in with the concept of Complementarity which Bohr was much focussed on--indeed probably well too much--but, once we dissolve the false assumptions Bohr had as to the impossibilities of going beyond the present quantum theory in terms of visualisation the background reality, the concept of Complementarity once more appears appealing (and my father constantly emphasized this point in our discussions, and I'm grateful for that.) So in a future book I can imagine giving this concept some more time together with my own concept of quantum fields. Q.f. again ties in with my Super Model theory. S.M.Theory is a set of ideas with a hint of a computational aspect but a fundamentally organic way of bridging quantum theory with general relativity theory and extending it towards biology, involving a universe that can be visualized as part of an organic multiverse with a creator and his angels or muses. This has not got the mathematical inconsistencies of present physics for it is written with an eye to illustrating formalisms done in G15 PMN as an alternative to mathematics, and as illustration of theory, not as core of theory--meaning, the theory isn't showing the universe as a machine.

Anyway, politically, a hope of some sort of 'historical necessity'--including but not limited to the hope that rigid dogmatic execution of limited interpretations of ancient scripts by religious hierarchies will go away--a hope that history will 'sort out' and a good future for all will come has been a driving-force for minds of the 20th century. And so there could be elements of a profound truth in something of that perspective if we filter away all its nonsense--of which there is plenty--and at the same time consider that the opposite of that perspective may be "another profound truth".

And so the little scripts used by fat, rich churches, mosques, temples all across the world to justify the persecution of those who practice love relative to those whom they love have made people, and rightly so, furious about the seemingly pointless limitation of human enterprise; and at the same time humanity, with billions

deflating the resources of a single planet and in all overt technology far, far away from a planetary nomadism (which I for years, and in writing, has regarded as the only pathway ahead for humanity), have a desperation at not seeing, with ease, light ahead. Unless they are very 'faithful' to their little books, in which case they usually interpret their books in the direction that men are best and it's up to God which children shall grow up and so abortion rights for women who don't have the economy to raise children should be removed. This is a giant, hefty conflict of opinion that is leading to hints of chaos at elections.

So let's imagine here, two profound truths: yes, phases; that's truth# 1. Yes, something holy is real and there's a future and there's something to letting this take over. That's truth# 2. When we combine them, we can see something like this as a perspective: right now, humanity is in a state where lack of invasive medicine of many sorts would rip societies apart in desperation for there are few joys and much toil and little resources for each. In this phase, let's "take things back" from the old corrupt religious leaders who so hate women's freedoms; let's vote in favour of full rights for all invasive medicine if that can lift up the lives of young women so they are not burdened with carrying children before they have built strength enough, also financially, to help the kids grow up--in other words, it must be wanted; and a woman's desires are more holy that the texts read by bearded old zealots in that regard. Most of those old texts were written in languages that could not even describe a girl, a woman, except as a peculiar add-on to The Men. I'm not saying there isn't a God; I'm saying that doubt in the old texts is the only way forward to Faith. In short, the old so-called holy texts are, in large parts, written by women-haters--and the Christian (and Jewish) traditions are not excempt from this. The same is so for Arabic in Islam except that in their description, heaven is abundant in beautiful young women and so there is a glimpse of true adoration for the woman, shown also in the female genitalia lookalike architectures for their mosques.

So a God-believer should believe that this phase is different from the future-phase: this is how to combine the 'destilled' profound truth of one perspective, the socialist one, with the 'destilled' profound truth of another perspective, the religious one. The color of this attitude is perhaps, somewhat coincidentally in modern politically terms, more 'violet' than red or blue; and one could argue that a complement to some shades of this color is bright spring green--which again is complementary, in a sense, to pink.

The natural Personal Computer monitor color of choice is in alignment with the research done also by IBM prior to their launching of the IBM Personal Computer in 1981. This thing about colors go in phases, and this is, I believe, the first year--this year in which this is written--since then that a similar (somewhwat dampened) green have been something that the younger generation is rooting for, with slogans like 'brat green' I expect the present focus on this color will be brief but in future trends, it'll come back and, I predict, get gradually more established. It will be realized how immensely fruitful it is to work with PC green; how energized and coherent the brain gets when it's had hours in which green monitors are used to view also photographs of the most beautiful young women on the planet. The afterglow of such a bath in such green is one

in which the brain is more alive with colors than when color monitors are used--in my opinion. That depends, of course, on the person also having a life and not just sits in front of the desk all day long. Green monitors involve, in other words, a way to make also 'real-line' seem more important--for it caps 'on-line' experiences before they dullen the senses by overburdening them.

Similarly, I do not find any further developments of the computer equipment along the lines called 'metaverse' or 'augmented reality' or 'virtual reality' anything but commercial gimmicks with an appeal mostly to those who haven't got a life and/or have given up mostly everything.

Also, it is clear that the future does not belong to those who seek to try to put quantum features under human control to a larger extent than that done already in digital computer chips which runs normal 32-bit (or slightly higher, eg 64-bit) Personal Computers today: there is no 'quantum computing' and never will be, except that people can make-pretend that they are programming directly into quantum theoretical possibility fields and make-pretend that they are more aligned to the future by doing so whereas in fact they are wasting their time on writing over-complicated and still digital algorithms that has no promise of any significant improvement over a digital computer. It is already harnessing quantum powers because that's what silicon transistors, big or micro, are all about.

So back to the two points of desperation and conflict in the minds of young people: one, is it against God to be positive to invasive treatments such as abortion? In this phase, here, where billions compete for few resources on a steadily more polluted planet, it would be grotesque to try to say that the wish of the creator is to go against the wishes of his most beautiful creations in this bodily trivial regard that is so easily handled by modern medicine. One doesn't have to be a socialist to approve of being rational, simple, wise, and to be pro-abortion is entirely and in all ways perfectly compatible with being pro-God; and this does not preclude the understanding that there can be a future, a different phase, a planetary nomadism, in which the proudest creation of God, has a more directly God-experienced situation in which there is such an abundance of joys, also sexual joys, that all and everything political becomes very different and the past reasonings fairly much gets dissolved. Which is to rescue a bit of the profound truth of the spiritual traditions. One can think both thoughts in one's head.

The other topic of huge concern is: does anything have any sense anymore, in terms of how all statistics shows that planet Earth appears to be, seen in terms of cycles of hundreds of years, or thousands of years, to be rather fragile and, if not smashed by a large unseen asteroid, soon undergo too many challenges that there will be any humanity left on it? That is the real perspective given to school-children all across the planet where children are as fortunate to go to a school that teaches anything at all except route repetition of scripts as determined by shoddy leaders.

I will answer in two ways. One is that with all my (grand or small) intellectual powers, and with all my (right or not) faith, and with the entirety of my instincts--and intuition--and my intuition, I know, is formidable and I will not excuse that statement--yes, absolutely, there is no doubt that humanity always have a future that everything we do in the present matters. There will be, miraculously or not, a transition--not to

rediculous hostile planets like Mars or Venus--but a real transition, and in time, so humanity as a core and essence will not just survive, but super-fantastically so. I'm sure of it. And I'm not talking a rediculous transition into a 'digital form' of personalities, as if a personality were merely an algorithm plus some data. I'm talking real living human beings and cars and computers and sex and fun and fashion and music and beaches. What exists now will, by warps, exist even more beautifully in a future of humanity, I'm sure of it. So that is one form of answer. These warps must take, I think, the super-model theory seriously; the algorithmic approach to quantum science won't do.

The other is more 'zen'. It goes like this: if you don't believe in God or soul or such you believe that all is a machine and well, then, just live in the present and don't care one whit about the future for a machine is just a machine and it doesn't matter in the least whether it continues or not. So if you're an atheist, in this phase of humanity, live up to it and learn to the art of smiling in the present and for God's sake don't go around depressed because of statistics. Who cares.

And if you do believe in God, well then, do believe, and don't doubt it and for that reason don't worry and learn the art of smiling in the present and for Christ's sake don't go around depressed because of statistics. But then don't disallow God of playing around with half-revelations here and half-revelations there and of wanting phases and allowing medicine to come to the point it has for a reason --the reason of being humane. Don't--as the USA policician Kamala Harris likes to say--let the government come "in between" the woman and the doctor. Totally the humane thing to say, whether for the atheist or the believer. The men who honors themselves and their texts as more important than women have misunderstood God's creation entirely: God made women first, and men as an after-thought, in order to help women make more women. You can say that without saying, like the singer Arianna Grande, that 'God is a Woman'.

That there can be 'another profound truth' in the opposite of one 'profound truth' is something I also think we could consider connected to what we very loosely can call 'capitalism'--of which the type I support is a capitalism of a small-business type--and what we can call a 'compassionate approach to collaboration'. I have touched on this sort of thing other places, and will but quickly refresh the themes here: it is not automatically so that an emphasis on money leads to a decline of 'compassion'. Nor is it the opposite thesis of Dialogue. A market where handcrafted products and lovely services are offered can also be a compassionate market. The fact of setting a price can create an interesting dialogue. There can be a direct personal motivation towards quality, not just quantity, once a product or service has got a price. There can be a reduction of the potential for quarrels when one is clear about a leadership process in which contributions are paid for in a pre-arranged way, with prices people have agreed on. A price can contribute to disentangle a too convoluted collaboration process and contribute to sorting out individual preferences and priorities for the businesses involved. Formal agreements, possibly with prices, can be excellent as groundwork for top dialogue.

And in order to facilitate a market, there is a meaning to building up brands, so-called 'brand awareness', in the sociology of a market. This can go together with launching

services and products that have neat names, perhaps code names of one or some letters and a number.

Indeed, finding a captivating sort of code name or number for a product may vastly assist its introduction in a market. In that sense, there can be a certain magic in setting a code name or number to a product. Perhaps the product or service is simply named by a single letter, or two letters; or by a number. While this may give a great initial boost as for launching this product, there is a caveat: nobody has the right to trademark or copyright the letters of the alphabet or any number. These are for all humanity to enjoy, be aware of, play around with, express themselves in, and in an important enough sense, _own_. That means that once a service or a product, or even a company name, has been launched given one or two letters, or a number, it must change this to a more particular and less universal name to avoid 'upsetting the magic' in the letters and numbers per se. For instance, if a product was launched under the name "737", it must of course change this name--which is but a number--because it is inherently meaningless to suggest that a company can in any way possess or own something universal like a number. And while amazing effects and visualizations may be associated with some numbers--just look at 37, for instance, which is astonishing in its radiance; it's a prime; it radiates up from 3, one of the most organzing numbers we have, to the natural number of structure and hierarchy, 7; it is a prime number; when multiplied by three it becomes three one's, 111, again signifying something great; and when multiplied by its component digits in the ten-digit number system, it becomes the number so talked about in Christian texts, as 3 times 7 times 37 is 777. When a company adds a digit to 37 and seeks to possess it, the number ceases to help the company and instead becomes a burden to it. Add a letter to the number, and it's a different story. The same when a company tries to become identical to one letter, or two letters: at first it is fun; but then it becomes like a joke one has heard before.

At any rate, both for enjoyment and to spot outrageous mistakes, and to refresh memory of just where the book text has come before it's extended, I read the earlier passages. And I wince at grammatical mistakes, at peculiar uses--or lack of uses--of such as commas, misspellings of names, and ponder sometimes over whether a "not" is lacking in a sentence for it to sort out correctly ;)

I think the following is the case: those who are the most concerned about punctuation and spelling in their natural language essays are those who are the least competent in programming. For these people appear to me 'afraid of disorder' and so seek to impose, on the flow of writing, a rigid scheme of correctness because they do not regularly experience the joy of perfect syntactical order in formal expressions.

I do think that there are many concrete instances where top spelling accuracy in natural language makes sense: one is where a paragraph is high-lighted and placed in a hard-to-avoid location like up front on the panel of a machine. To read the same paragraph with the same uncorrected spelling errors daily or each time one encounters that machine would be annoying. Another instance is when, for the sake of generating a genuine laugh-out-load experience we have comical writing where elegance and effortless quick comprehension, drinking in the language in a rhythmic way, requires a sentence construction one doesn't look twice at and blinks--though of course, there may be

elements of that comics which works even better with "just the right mistake".

But in somewhat more explorative, more philosophical texts, what is the point of perfect punctuation? In spoken language we regard it is as part of personality and of body language to stumble in a sentence and begin, half-way afresh and start off in another direction, or to cut off a sentence after just some words because things are tacitly understood--and so on. Why cannot this personality carry into written language? I saw, for instance, I had written a word I cannot remember having seen before, but which in the context above was nevertheless perfectly comprehensible--when I said, such-and-such is now all "implented". I have used the word 'implemented' so often above--perhaps too often--that it is hard to avoid the understanding that the intended word is 'implemented' and not 'implented'. So why correct it? Is this not about the process of thinking? Is it a reality that thinking gets clearer when the official schoolbook rules of natural written language is imposed on the flow of explorative writing? Or is it rather a stultifying of process which may obscure certain hints about the actual flow of thinking--and if there is anything we don't need in philosophical exploration, it is conscious obfuscation.

On the other hand, things get naturally obscure, sometimes, without perhaps intending to, when spelling and punctuation and/or grammar isn't correct, and in a philosophical text--unlike a comedy--to pause and ponder over what on Earth could such-and-such mean--may be a very philosophical 'effect'--in other words, quite fruitful, even if perhaps not consciously intended at the moment of writing. The love, philos, of wisdom or truth or knowledge --sophia--as philosophy, requires a humility of person relative to something infinite and beyond. One thing is what the person may have intended; quite another, and often far more significant, is what the text, in a philosophical context, may lead to of contemplations, questions, creative ideas, intuitions, insights and perspectives. My proposition is, as it has, I think, always been as regards this theme, that philosohpical texts should have a body language that goes beyond conventional grammar, punctuation, spelling and sentence construction. I'm grateful for many conversation with the Norwegian philosopher Arne Naess also on his view on Baruch de Spinoza's ethics philosophy. Spinoza wrote in Latin in an almost geometrical manner. Naess stressed a similar point--namely, that here it's more interesting what _we_ think such-and-such ancient text can mean--and the exact meaning of the author possibly less interesting.

As to the notion that the opposite of a profound truth might be another profound truth, it occurs to me we should also consider the opposites that already, in ancient Greece, Plato framed so well--that of democracy versus the autocratic rule or dictatorship. To my mind comes the image of a dancer--a well-trained, strong, flexible, agile dancer, who can do magic with her body. At one point in her mind, she has--if she's truly expert--a choreography. And because of the elegance and flexibility and strength and well-tonedness of her body, the choreography can unfold itself, and if it is a good choreography, the dance is good. Let the dancer be a metaphor for a society. If there is one point that controls the movement, and the movements are expertly trained and harmonious, that is akin to what Plato called an 'enlightened dictatorship'. The autocrat, the ruler is that one point. If that one

point is corrupt or self-centered or destructive or too greedy, then all the expert elegant force of the whole society is to no avail--that is a 'bad dictatorship'. In our metaphor, it means that the ballerina, however smooth and supple in her limbs, however flexible and strong and musical, makes a mockery of the dance because the choreography is bad.

An argument often made is that since one cannot guarantee that the autocrat is good, a democracy is a safer bet. There are usually at least three ways in which a democracy is different than a dictatorship. The first is that the leader, chosen by votes, are typically changed a couple of times each decade. The second is that this leader usually doesn't make decisions alone, but in a process of constant consulation with possibly a vast swath of assistants, institutions and allies. The third is that the decisions, when made, aren't carried out by a smoothly tuned, obedient regime, but is filtered down through yet more democratic-like institutions before they become implemented. In a democracy of a socalled "socialist" bent, people are principled in respecting people at large, for the care of the largest quantity of people is considered a value in itself; and this care has a priority over executing the decision of a leader or leadership.

In the metaphor of the dancer, a democracy will, I think, more typically be the untrained dancer, indeed one who cannot make up her mind how to dance, except so slowly that it can hardly be called a dance anymore. There is neither the focus on training nor on having a clear and good choreography nor on unfolding it as timely dance. The immediacy exists in the democracy between any two or more citizens in the democracy who happen to meet well, but the immediacy one can sense in the image of the first type of dancer--who with graceful, strong, slender limbs unfold delicately a wondeful cheography--has simply dissolved. What rescues democracy is that its inertia, its 'absent-mindedness' and so on, may be much better than an autocracy gone wrong. The strong, powerful dancer is all the more absurd if the choreography is hopefully missing the mark; how much better it is with the fat indecision and mediocrity of the democracy then!

Human beings, including autocrats, are not only mortal but their brains decay each decade after adulthood. While a dictator may at his or her peak be wise and intelligent, he or she will not remain that way; and when that dictator dies, the children of this dictator or those who, by appointment or revolt, take over, may be stupid from day 1. This means that dictatorships, despite possible initial advantages, typically lead to self-destructive societies.

The choice of political system cannot be independent from worldview. In the atheist worldview, there is no higher intelligence and no absolute values apart from what humans adopt; and so, generally, that tends to suggest democracy. In a monotheistic worldview, some may be better 'instruments' in listening to the Deus than others and if the society is lucky enough to have such a one as autocrat, and in addition the rest of society is streamlined to implement--as with the image of the well-trained graceful dancer--that could make of course the best sense, but only as long as the instrument is a worthy one; and for human beings, that's always a limited time.

To return to the far more mundane topic at hand:
In the next chapters, let's define level 10,000,000, with a high-priority task in this robotic FCM network. This is

incomparatively more complex than all the other levels, because it acts to organize everything else, the way we've sketched this FCM computational node network. To make sense of the level ten million program, go back to the upcoming chapter repeatedly, with 'about level ten million' in its title.

MAKING ROBOTIC FCM PROGRAM:ABOUT LEVEL TEN MILLION
Yes! Consulting the list of triplet values we made in the earlier chapter that also had level number descriptions, the first value of triplet# 5 is the most important for the upcoming node. This is the number that, when one is added to it, will be the next step to be performed. Triplet number 5 begins at position# 22, so this has step number. Whether or not the main function for the node is put into the first triplet, or in another triplet, such as this, doesn't make any difference in practice for this particular FCM network.
And the single high-priority node we have sketched ideas for has indeed many steps; when it is complete, it is the task of this node to reset itself (and earlier on, each elementary action node) and, via 'soft activation' field in the level 100 node, to allow the network switch go over to GPS and show the glorious result of the work this task node has undertaken.

In the chapter describing the triplet uses and links, and also the levels, it's mentioned that links to match-oriented nodes come first, and links to subtasks and such, second--of the hundred links a node can have. The node we're about to make here at level ten million is full of links. It should have links to all the fourteen match number nodes we've already set up, as well as links to the 'elementary actions' we're going to set up.
It appears to me that we should have some neatly named functions to make the coding of this function easier. We can have one called SOFTACTIVE, which has link number as input, reaches out to the node linked to by this number, and sets the 'soft activation field' #21, to DANCE. We can also have functions called something like INSOFTMAIN, which we can, as intended meaning, think of as 'get in the (previously soft-activated) node's main value'. Its input is the link number, and it gives the first value of the first triplet of that linked-to node. These should also have the warp to the present node as input. By avoiding a 'global variable' as to which node is present, all the 'computational attention' is through the TRANSLUCENT loop to what it calls of node functions with the warp to what it has 'decided' is the present node as input. When we have several nets being performed alternately, global variables to refer to which node is presently being can sometimes require extra explanations.
And we can make more than one, eg FTSOFTACT, which we can think of as 'from-to soft activation of nodes which are linked to, in the range specified'. When the fourteen match nodes are going to be activated, we want something like this call--here assuming TX has been used to store warp to the present node:

```
1
14
JX
FTSOFTACT
```

We can also have a FTSOFTACT5 that puts value 5 there instead of 1, since these are the two types of activations for the match image nodes here.

And maybe we could make an input-version of FTSOFTACT, --one that compares and picks the highest-numbered pmille in the main field. It could be called SOFTACTWIN. It would have a range as input, like FTSOFTACT, and would scan for the topmost pmille match number--ie, the winning node in this range.

When we make such a function, keep in mind that this would pick any one of the winners when more than one share the winning value. Here, that's perfectly alright. In another context, generating a list over those sharing the highest main value might make more sense. What is the key here is not which exact match node that gives the winning pmille, but which 'elementary action' that is leading one or another match node to give a winning pmille.

How many 'elementary actions' do we have here? It's up to us, as long as it makes visual sense. When we start up the app made just before the one we're working on now, the app# 1005768, and we feed it with a 'binary image'--ie, one that is in each pixel either black or bright green-- it quickly shows the match numbers; and we can also use it to show the match images. Each elementary action we make should have as a potential, when used right, to increase the match number of, say, an image we compose in GEM by means of a modification of the inbuilt match images-- slightly messing it up and letting this app we now build, app# 1005769, improve it again.

By instinct, I suggest ten 'elementary actions' here, each relating to a different 'line' (broadly defined) of the 116x160 binary matrix image.

That means that, on the task side of the link section, we're going to have ten links there. What more do we need as for links? We are going to activate GPS FCM net to show the result when an elementary action has been determined after trying out the various actions and getting the match numbers for each.

So, to summarize the idea of how this net does this, as organized by the higher-priority task node here at level ten million, each elementary action will lead to an activation of all fourteen match nodes, and in a field in the task node we store the number of the link of the best-performing elementary action so far recorded relative to the highest increase value of the best-matching match image. So before any action is done over the copy image, there is soft activation of the match images on the main image. That's step 1. In a field in the task node, the number of link to the best-performing match node, alongside its value, is stored. If the value is already 1000 pmille for a match node, it means the image is identical or practically identical with a match image, and nothing more remains to be done, and computational time goes over to GPS again.

We don't have to only use the uppermost triplets for a task if it needs extra places to store info; but we also have the luxury values and my sense of it is that we're going to use some of these now for task counters and such.

Remember that a simple SCAN on the word TRIPLETS should give you the short-hand overview over a node when you have the Third Foundation or an extension of it at hand. And this foundation has proven to be extremely versatile; the extensions of it so far has been entirely in alignment with it and without any changes of core code. So it is a permanent, stable, robust, trustworthy aspect of programming. While ideas of a fourth foundation, and such,

occassionally will be explored and perhaps realized, this is meant to function alongside all the existing foundations and not to replace anything and to be coherent with all the core tenets of G15 PMN in its Third Foundation fulfillment.

We're going to create a lot of links for the higher priority node and for this, we'll make some auxillary functions using in particular FNAMW, which gives us the warp of a node once we input its name. That presumes, of course, the node has already been created. So we'll add the links to the match nodes as soon as we've made the higher-priority node on this level, ten million, and we'll use a variant auxillary help function when we are at the elementary actions level and wish to add these also to the higher-priority node. Of course, it is possible to create the elementary action nodes before we create the this task node, and sort the levels afterwards; but it is good to concentrate on completing the higher-priority task, also because it sets the constraints for how the elementary actions should be shaped as nodes.

As we said, step 1 is activation of the match nodes on the main image, and step 2 fetches best match number. We don't strictly have to store which match image this refers to; what is clear is that there only so much 'clearing up' the elementary actions can do and when there is little or no increase of the highest match number relative to how it was before the elementary actions were tried out, then the higher priority task 'decides' that its action is fulfilled and it soft-activates the node that sets GPS on. It's enough that the previous highest match number pmille is stored--whether that came from matchings over the main image or from a recent change of it through elementary actions. By the way, step 2 stores best match number in one of the luxury fields, pos# 43, and it's read in by step 4.

We define step 3 as sweeping over many cycles, which means it has 'substeps' or sweeps: each elementary action on the simulated image will lead to a cycle where the match numbers are generated, the highest pmille stored when a particular elementary action got it better than all the previous ones, alongside the number of the link to this elementary action. In our app# 1005769 each elementary action will work on a fresh copy of the main image; we must remember to program in an oft-repeated copy action there. We could designate a fairly high-level subtask to do this, as a separate node, but we'll keep it one node here.

Step 4 will be to let the elementary action that the task node 'discovered' as the best one, act on main image and store it back to disk. This on condition that the found increase of match pmille was a significant one. We can experiment with settings here for what this threshold should be, perhaps 15 pmille or so as a minimum. Luxury value at pos# 43 holds the previous best match number for the main image. In this experimentative FCM app as we plan it, the interactor with the program will click repeatedly to get the FCM network to "do" something with the main image. At some stage it has got nothing more to do--which is when the improvements on the best match number is marginal; at which point the interactor may have a go with a different main image, perhaps as prepared in GEM. Had this been an FCM network running a live robot, we would

have made a triplet or luxury field have a flag that says, like, "This goal has been adequately achieved". Here, it will be obvious to the interactor who clicks repeatedly in the app to see the modifications done, one after another.

Step 5 is that the higher-priority task node resets itself and is ready for more cycles should the interactor with the program want such, as indicated, typically, through a letter-warp in the GPS. The interactor will click repeatedly on the letter, each time the 'winning action' is performed until, given enough clicks, as much has been done as can be done given the definitions of these actions.

This book has more than achieved its programmatic goal when step 3, with all its sweeps, have been programmed. The remaining steps and the levels with higher level numbers--here, essentially the elementary actions, are trivially easy in comparison.

So we use the term 'sweep#' for the substeps under a step in this sort of node. The steps can be functions called on from an array like this, where we use the phrase 'hptask' as abbreviation for Higher-Priority Task--this we perform after we have defined functions for each of the 5 steps:

```
 HPTASKFUNCS=
 ^123456789.
 ^HPTASKSTEP1
 FF
 1
 HPTASKFUNCS
 YA
```

And so on, up to four. Here's number 2:

```
 ^HPTASKSTEP2
 FF
 2
 HPTASKFUNCS
 YA
```

Tiny arrays can be made as constants, rather than via variables, when the quote fit on a single column, thus simplifying expressions and good to use with AY and YA.

Let's assert that each function of the HPTASKSTEP type itself will increase the triplet value that holds the present step number for the task. As we said, in the case of step 3, it has many 'sweeps', ie, cycles of the TRANSLUCENT FCM loop, before it completes. Let us look at what these are in a list-form, so that we can figure out how to organize them by one or more numbers. These sweeps can also be organized through numbers similar to step numbers, and each sweep might as well also be in an array of functions as we have already planned for the steps.

We can use the Luxury Values to hold, in updated form,

*the current sweep number
*link# for the recently recorded best-performing subtask
*the pmille it got from this subtask
*the pmille the main image got from the highest-pmille-yielding match prior to the tentative elementary action
*the current link# for the subtask next to be called on --it needs to try out all 10.

So there is a Luxury Value that stores the link number for the present subtask under 'evaulation'. Next, the sweeps!

SWEEPS for step 3 of the high priority task-oriented node; as with the steps, after a meaningful amount of "work" by sweep has completed, it lets the computational force go on

to other nodes--and updates the sweep number only when it has had enough cycles to complete its task.
1. Reset the values for a fresh new series of sweeps. Importantly, that involves setting the elementary action to be considered to the _first one_. The last sweep # will go up again and again to the next sweep # in order to have 'considered' all the elementary actions and picked a winner.
2. Copy the main image over to the copy-image and soft-activate the present elementary task under 'evaluation'. In a real robotic context, the completion of that task is out there in the environment involving servo motors and such, and it can take seconds, even very many seconds, to complete. Here, we can confidently update the sweep number to the next step at once, since we're doing a virtual physical action.
3. Soft-activate the whole range of 14 match images, and with the setting that they work on the copy image. Reminder: the main image represents the 'current state of affairs', while the copy image represents what we may (metaphorically) call the "thought of the action". When you're about to do something, then in your conscious mind you'll have a sense of the likely effects--this is the algorithmic equivalent in the FCM context.
4. Fetch (one of the) best-performing match pmille's and in case this is better than the already-stored best match pmilles, update the pmille and also record which elementary action is so far the winner candidate.
5. Update subtask link# by one. If it exceeds the quantity of elementary tasks, which is here ten, the sweeps are complete; otherwise, it continues with more sweeps here beginning with a fresh copy-over of main image to copy image--ie, reset the sweep count back to sweep# 2. Note again that the key loop here is the TRANSLUCENT loop as defined in the core G15 PMN, so here we merely update the sweep number rather than bundling the sweep within the typical LL .. LO type of loop.

And in the high-priority task node, the _steps_ it has it goes through sequentially, as the TRANLUCENT loop calls it in fresh cycles. In this node, then, the step counter isn't set back to an earlier point except when the task is complete. But it is perfectly okay to use these step numbers also for that which in practice amounts to 'loops'. In other words, not just the sweep numbers but also the step numbers can be made to be loop-alike.

The functions representing each step in the high-priority task's sweeps under its step 3 we can put in an array called HPTSWEEPS. These functions, as the HPTASKSTEPS, take one input, the FNWARP to the node itself. Their activity reflects in how the triplet and luxury values of the node change, rather than an output from them on stack.

So, as a start--we have to go slowly here so that all the details of the FCM node network gets talked about enough--let's set the level# correctly:

```
10000000
SETFUNDLEVEL
```

The node here is high (not top) priority, the highest we have in this example robotics FCM program, and it is a hugely organizing node. As sketched in the previous chapter, and in the overview chapter over the triplet values earlier on in this book, these are the fields, and their positions, of concrete use in this programming for this node now:

21 'Soft activation field' For this node, it's DANCE.
22, the first value of the fifth triplet: Step#
This is reset by the node to BASIS after it has completed a coordination with other nodes to 'tidy up' the main image; it is increased by this node itself by one as it goes through the steps. The step 3 is the only one with substeps, ie, 'sweeps'.
24, the third value of the fifth triplet is used as a flag for subtasks and elementary actions to indicate completion of task, but not necessary to use here, since even the elementary actions will for sure complete their job after only one cycle.
Luxury values in the node:
40 Sweep number. Step 3 uses this, and these:
41 link# for the recently recorded best-performing subtask, which may increase as it cycles through the various tentatative subtasks performed on the simulation
42 the pmille it got from this subtask--and in this FCM node network, the subtasks are exactly the elementary actions; this pmille and the next refers to the best match number
43 the pmille the MAIN image got from the best match --this is in order to compare whether the gain is significant enough--stored here by Step 2
44 the current link# for the subtask next to be called on. During the init of the sweeps, this is set to 15 since the task-link comes right after the 14 match-oriented links.
Now, let's define more along the lines we've started to indicate. We have, in previous chapter, indicated that we want functions SOFTACTIVE, FTSOFTACT, SOFTACTWIN; for each step we want HPTASKSTEP1..HPTASKSTEP4, which go to array HPTASKFUNCS; for step 3 we make an array called HPTSWEEPS, and make HPTSWEEP1..HPTSWEEP8 functions and give to this array. The main function for the higher-priority task we can, by FNACTCHERISH give number 4750 (in the standard definition of the FNACT function array, we can use any so-far unused number up to 5000; the 47-series has, as far as 'number significance' goes, an association with the robotic, and the high-priority node is midway in the level numbering, thereby 4750). We can call the main function for the higher-priority task node for HPTASKNODE. Now let's go to the work of completing the programming of the node on this level--and then the remaining work to complete this FCM robotics app is certainly easier.
When we make HPTASKNODE, we might as well set the first triplet to BASIS, as it is task-oriented, not match-oriented (though it has a match-like aspect, of course). We shall also make helpful functions, one or two auxillary functions, to add links quickly. The first 14 links are to MATCHIMG1..MATCHIMG14. The next ten links we add once we've come to the more elementary actions--there are many ways of doing this of course, we could have made the elementary actions first and flipped the level numbers back and forth and sorted with BS afterwards. But here we stick to the gradual increase of level numbers, there's a "node-near" first-handedness about that. And if we want to add all the elementary actions after we've made them, we can have two node names as input to this auxillary function--that which the node is going to link to, and that which is the name of the node to get that link; the latter being, in this case, HPTASKNODE.
The word 'link' is easy to say and perhaps said too often in this context; a little 'word-magic' is sometimes making programming more full of meaning. In the human brain, there are myriad ways in which some form of

connection or resonance can be established between two or more regions of brain/body. Let's pick a word inspired by our sense of the jam session symphony of the coherent, resonant human nervous system, without implying that we are encapsulating it. It is merely an inspiration; FCM is, after all, going to be an expression of mentality; and so to seek inspirations are to the point.

Suppose EATASK1NODE is Elementary Action Task 1 Node, and we wish to put this into link position# 15 of the HPTASKNODE. How about this syntax:

```
15
^EATASK1NODE
^HPTASKNODE
RESONATE
```

And the next:

```
16
^EATASK2NODE
^HPTASKNODE
RESONATE
```

Like 'entrainment' the concept of 'resonance' is more physical and much less psychological/mental than words like learning/training and reasoning/cognating. So words like learn, train and reason should be avoided in all scientifically well-grounded, high-integrity discussions of the features of pattern matching node networks and their capacity to enact a meaningful goalsorting in terms of action, such as robotic action. It may _appear_ that the node network "learns" but if the quotes are taken away we are over to sales tricks, hypnosis-attempts and a kind of 'gobelian propaganda'. The lowest kind of politics known to humanity uses gobelian propaganda--a repetition of lies with the intent to make gullible people come to take them for granted more or less like truths.

So RESONATE is, psychologically, not a presumptious term. It has in it a humility, psychologically speaking, and this is a token of good work in the realm of FCM robotics. We should have a variation of this function when the node linked to is the present one, just defined. When we have just defined HPTASKNODE and wish to refer to MATCHIMG1 we will have a similarly-named auxillary function, that we'll define in a moment. It will look like this, for the two first links:

```
1
^MATCHIMG1
RESONATEWITH
2
^MATCHIMG2
RESONATEWITH
```

The RESONATE and the RESONATEWITH functions have in common
that they add 49 to the link# given, and check at pos# 47 that that number is already higher than the link number, and if not updates the quantity of links in that position.

Let's do the rest of level 10,000,000 in upcoming chapter.

MAKING ROBOTIC FCM PROGRAM:LEVEL TEN MILLION FUNCS

Let's first make some of the functions we need to complete this level--and we can begin with RESONATEWITH. It has two

inputs on stack, the link# and the node-name. It will use our friend THISFUND similar to how ADJUSTFUND does it--in fact, I will scan up that definition now and get ideas in order to make the functiion. It adjusts #47, the quantity links in this robotic node, only when the present link# exceeds its existing quantity. Link# are in range 1..100.

```
Right, this is it:
RESONATEWITH=
|IN:L#,FNNAME
FNAM
SX
S3
49
WHEREISFUND
S4
```

At this point adding one to i4 gives us warp to link# 1, while substracting two from JX gives us warp to amount. Also iX has the node# for the node to which we will make a connection, and i3 link#. Let's make it:

```
iX
i3
i4
KW
```

And let's get the quantity updated when necessary:

```
Q4
M4
LK
i3
MAXOFTHIS
M4
KL.
```

The function RESONATE is similar but doesn't lean on THISFUND, so we must mimick the effect of WHEREISFUND to get it. This should work--here I copied and pasted the RESONATEWITH functions and modified until I got it:

```
RESONATE=
|IN:L#,TONAME,
|FROMNAME
|ACTION:LINK#
|IN NODE NAMED
|FROMNAME TO
|NODE TONAME
FNAM
SX
FNAMW
TX
S3
49
JX
AD
S4
IX
i3
i4
KW
|QUANTITY:
Q4
M4
LK
i3
MAXOFTHIS
M4
KL.
```

We have talked of wanting functions SOFTACTIVE and INSOFTMAIN above, time to construct them. Link number is from 1..100 and should be added to 49 to get the position

in the node; add this to the nodewarp and we can fetch the node number stored at that link. This is to have its soft activation field, pos# 21, set to 1.

```
  SOFTACTIVE=
  |IN:LINK#,
  |FNWARP
  49
  AD
  WK
  FNWARP
  DANCE
  W
  21
  KW.
  SOFTACTIVE5=
  |IN:LINK#,
  |FNWARP
  49
  AD
  WK
  FNWARP
  5
  W
  21
  KW.
```

And here is INSOFTMAIN:

```
  INSOFTMAIN=
  |IN:LINK#,
  |FNWARP
  49
  AD
  WK
  FNMAINVAL.
```

Time to get FTSOFTACT and SOFTACTWIN constructed, as we have earlier planned:

```
  FTSOFTACT=
  |IN:FROMLINK#,
  |TOLINK#,
  |FNWARP
  TX
  S4
  S2
  LL:1
  i2
  JX
  SOFTACTIVE
  i2
  i4
  LT
  SE
  Q1
  H2
  LO.
```

The FTSOFTACT5 is the same but that SOFTACTIVE5 substitutes SOFTACTIVE.

Here, H2 'hitches' up i2, link#, until it reaches i4.

Amazingly, SOFTACTWIN is very easy:

```
  SOFTACTWIN=
  |IN:FROMLINK#,
  |TOLINK#,
  |FNWARP
  |GIVES:WINVAL
  TX
  S4
  S2
  BASIS
```

```
LL:2000000000
i2
JX
INSOFTMAIN
MAXOFTHIS
i2
i4
GE
SE
EX
H2
LO.
```
These routines assume meaningful inputs in terms of from link# and to link#. Be sure to notice that SOFTACTWIN has an output, a result, on the stack, and that that has to be handled by the node function that makes use of it.

MAKING ROBOTIC FCM PROGRAM:MORE FUNCS AT LEVEL 10 MILL

The SETFUNDLEVEL has been set to 10,000,000 and I'm looking up here and there in the recent chapters to find what we have to define to get through with this level--clearly the most advanced, as far as definitions go, in this FCM network. We have defined the array HPTASKFUNCS, and will put it to good use in a moment. We need an array for the coordination of the processes under step 3, which we called 'sweeps':

```
HPTSWEEPS=
^123456789.
```

When we define the individual sweeps, we put them into the warp by this sort of statement:

```
^HPTSWEEP1
FF
1
HPTSWEEPS
YA
```

To say that we put a function into an array is, of course, a shorthand for saying that we put the warp of the function into the array. The power of this action is greatest when it is transparent to the programmer what the computer is doing. One can make various languages that mimick some aspects of warps but conceal some features in a fancy syntax but this will at some point suffer because of the lack of transparency in what is being done: a case in point is when a language is designed to have a syntax that 'allows a function to be passed as a parameter'. This is not what is the case, just as the fact that within a single computer, hiearchy is, generally speaking, not how RAM is organized, rather it is a sequence. And it is never really a 'function' that is passed as parameter (unless one means the quote of the program definition for the function), but rather the warp to it. Nor is RAM organized as 'lists' that can come and go. Nor is program performance a result of a computer 'seeking' a best logical match given premises: all such syntactical structures have to be hammered into sequential algorithms doing things with a sequential RAM that consists of nothing but numbers and some of these numbers are warps, while some represent data including more readable numbers and text. And the only natural way algorithms can call several other algorithms over several levels is by means of a stacking of parameters in an array of the 'last in,

first out' type, which, together with global variables, is the ground-approach to all computing for its sheer simplicity--a simplicity so strong it can effortlessly be implemented at the electronic level. What you experience with G15 PMN is a direct connection to how the computer is organized--or can be organized, in such case as when you run a PVI, a 'practical virtual implementation' of it. And this is a training of mind, a training of the human mind--for it is having an additional language, the most natural of number-oriented formal languages, as added tool in its toolbox for self-attention.

Anyway, we have enough definitions that we can get the main foundry definition at this level going:

```
BASIS
4750
BASIS
&HPTASKNODE&
FNEASYACT
```

As FNEASYACT is defined, the function number, here 4750, is put into the centre of the first triplet. While in many cases it may be of value to 'tune' the triplet values to the function in them, as we set it up here, every node makes use of the first triplet whether it's match or task oriented--as regards the main node function.

The 'soft activation' flag:

```
DANCE
21
ADJUSTFUND
```

Earlier on, we have clarified the RESONATE and RESONATEWITH approaches to set up links, so we won't repeat it again here.

So far, so good. The giant 4750 function is giant only in how much it controls, while it is composed of a number of rather tiny subfunctions. This is the toplevel--and notice that when we do things through a FCM network, we allow TRANSLUCENT to do the looping--the node merely handles the counting and the sorting out of what step is next. The steps will update the step number, at pos# 22, but since it starts out at BASIS, we add one before the step is called, every time. This drives the steps:

```
HPTASKDRIVE=
|IN:TR#,FNWARP
SX
SH
|FNWARP TO STEP:
iX
22
WK
|ADD 1 TO STEP:
UP
i1
HPTASKFUNCS
AY
PF
&HPTASKDRIVE&
4750
FNACTCHERISH
```

This is the number, 4750, that we put into the main node we just defined, a couple of paragraphs earlier, namely HPTASKNODE. You are free to make a node with an as-yet undefined function number as long as you get it defined before you switch on the robot! ;)

Also, you are of course free to call on the functions in an array such as HPTASKFUNCS in a definition before you

have given this array any content, as long as you get around to give it content before you start up the program. Let's give it content. That's our steps and the sweeps.

In this architecture, it is the job of the HPTASKFUNCS to do such as reset the counter and--in our app--also let GPS be called on after each elementary action by a soft-activation of one of the first nodes. This soft-activation is also going to happen when there isn't any elementary action that provides salient improvement of the main matrix. This way of doing it adds 1 to step number before calling the function, meaning it's kosher to let the step number be at BASIS at start of performance of FCM node network.

We have five HPTASKSTEPs to make. The first activates 14 match nodes over the main image and the next gets the winning value. Here's step 1; step 2 will get them; step 3 is the 'sweeps' figuring out the correct next action; while step 4 does it and step 5 resets. Each step will have an action in it on the step number. Let's have a go at them--and here FTSOFTACT rather than FTSOFTACT5 is used since we're working first on the main image:

```
HPTASKSTEP1=
|IN:FNWARP
SX
1
14
iX
FTSOFTACT
iX
NODENEXTSTEP.
```

See an earlier chapter, 'ABOUT LEVEL TEN MILLION.' as to how to insert such as HPTASKSTEP1 into the HPTASKSTEPS.

The function NODENEXTSTEP increases the step number field at pos# 22:

```
NODENEXTSTEP=
|IN:FNWARP
22
AD
1
W
KU.
```

And why don't we do a 'NODENEXTSWEEP' as well--the sweeps have their counter at pos# 40:

```
NODENEXTSWEEP=
|IN:FNWARP
40
AD
1
W
KU.
```

The HPTASKSTEP1 presumes we've got the links lined up: the first fourteen are to the match nodes, the following ten are to each of the possible elementary actions. In the app# 1005769, and these apps come with source, ie, full program code, there is a bunch of RESONATEWITH that links this node to the previously defined match-image nodes, and after the elementary action nodes are made, a similar bunch of RESONATE connects them to this node. We can't link them up all at once because a link is something you do to something that has been defined, and, as you might recall, we move through the levels:

```
input-level => matchings => higher tasks => elem.tasks
```

And in this our app we have just one higher task, the one that has this very elaborate algorithm with many steps and also many sweeps under one of the steps in it.

So, my dear FCM programmer, let's bear in mind that when we say of a more usual function that it has many steps, we typically mean that it starts on the first and goes through them and then exits. But the style of function we choose to implement in our computational node networks to drive robots, is so that it must allow other functions some computational time each time it has made even the slightest step (or 'sweep'). The way I use the word 'polling' that's what it's about: the FCM network is polling its various nodes, or the functions in the nodes; and this gives a sense of some degree of "simultaniety" of the digital processing.

Therefore, with our nodes having triplets, and triplets --or at least one of them--having functions, it make good sense that at least one of the values in the triplets has a "step number". In that way, the function can 'see' how far it got the last time it had computational focus--ie, the last time the TRANSLUCENT loop called on it--and carry on from there, and update its step number. And when a step has substeps under it, we can call that 'sweeps' and by analogy, there is a sweep-number stored somewhere in the node (whether in the triplets or in the luxury numbers).

Onwards to step 2. We're going to bring in SOFTACTWIN and it fetches the best match number; in the overview chapter ABOUT LEVEL TEN MILLION we declared that the lucky luxury value to receive the winning lottery ticket number is # 43. We're talking here the main image. Thereby,

```
HPTASKSTEP2=
|IN:FNWARP
SX
1
14
iX
SOFTACTWIN
43
iX
KW
|RESET SWEEP#:
40
iX
RESETNODEFLAG
iX
NODENEXTSTEP.
```

This feels to me like light jogging down a hillside in the Sunlight after a light lunch including coffee on the hilltop, compared to earlier chapters. We'll get through all the steps and sweeps by cruising through the list of what is to be done and translating it to our lovely formalism G15 PMN.

Time to begin on our sweet sweeps. Through many cycles, these 'try out' the various elementary actions on the copy image--ie, they do a sort of 'simulated action'--and the best pmille match is stored in luxury field# 42. As with steps, the high level is very simple also because it doesn't bother to update the count number--yes it adds 1 to it, but it doesn't store it back. It adds one because the first value of a typical short array is--unless we define it otherwise--the position# 1. The sweep number, at position# 40, was just reset.

Here, you'll see that the warp to the foundry is copied by one of the three one-letter functions I made in the core PMN, F. We can think of F as 'forge a copy', W as symbolic in letter-form to a sort of switch, while D is symbolic for a sort of bridge and rhymes a bit with with that word. These three have their roots in that which inspired a key aspect of G15 PMN, namely Forth. Their

naming here so that they stand out is because of their essentiality and, we might say, "truth" (in a stack-oriented context). There are three variables in G15 CPU code that I shaped to stand out in a vaguely analogous way--a group of G15 functions are connected to them--the THIN A, THIN B, and THIN C, sometimes abbreviated to THA etc. You can see a list of G15 function by, in the edit mode of the CAR menu for G15 a click on CTR-Y and browse with PgDn there.

This is a design intuition I always have used: that the highest design and beauty involves blending aspects of three higher principles or personalities, which is of course the three muses. Here, in PMN, the somewhat 'coquette' shapely function and form of W suggests the muse Lisa, known for her (righteous and well-deserved) vanity. The leap of D is resonant with the kick of the pretty-footed warrior muse Athina. The ever-abiding flawless presence of Helena resonates with the ease of F.

```
HPTASKSTEP3=
|IN:FNWARP
F
SX
|GET SWEEP#:
43
iX
WK
UP
HPTSWEEPS
AY
PF
```

The first sweep--here I follow the list of the sweeps as discussed deep into the chapter 'ABOUT LEVEL TEN MILLION', resets the values used by sweeps; the second of the sweeps is part of the repeated sweeps, a sort of loop-through-the-cycles, to locate the best-performing subtask. But before we go on, a little pause in the coding makes for a more holistic book on thinking ;)

INTERMEZZO: LETTING THE SUBCONSCIOUS BE PART OF THE WORK

One of the many objections I have to too-strict 'measurements' over how we humans perform in a work situation--a issue of much discontent in the (over-) technologized societies we have at present, where employers appear to increasingly demand efficiency in 'every minute' and use machines and unethically made program to enforce this--is that the most brilliant forms of human natural intelligence emerge out of the subconcious and this takes time; a time that may appear to be leisure-time but which is everything-but.

So, after dabbling into an intellectual question, the person may go on to having sex, resting, doing some yoga, browsing some news--everything except working--and upon returning to the work at hand, a solution may present itself as if by magic. Yet that which we--though the term isn't exact--can call the 'subconscious' was not only hard at work all the time, but could not have been as hard on the work if it had not been for the sex/rest/yoga/whatever --for it uses as fuel for its work other bodily and mental rhythms than that which appear to be 'efficient' in an external sense relative to the task at hand.

And in this consideration, sleep is not just sleep: it

is also work, indeed possibly exceedingly hard work--and what we may carelessly call 'dreams' may, sometimes, be at the core of this very real and hard work. Along the same lines, a headache may not just be a headache but a way the body and its brain organizes some of the subconscious work --and indeed the cure of the headache, properly, may be to divert from the most obvious task at hand--indeed also deviate from the task the employer has given to the employee--and unfold a different type of activity, perhaps an intellectual one but without direct connection to the designed task ahead.

So it is not that 'we must let in the irrational' in order to have a better society and better lives, but more that the rational is not just what it appears and what may at first hand appear irrational may not just be irrational --just as it may not be simple always to tell what is simple, nor may it be obvious at first to say that which appears obvious later on.

MAKING ROBOTIC FCM PROGRAM:LEVEL TEN MILLION, SWEEP# 1

In a funny way, programming requires that you have your life in order. For doing good programming requires a great presence of mind. Lots of things must be kept lingering on the threshold of conscious thought, and swiftly recalled as needed, and this presupposes a high degree of coherence of mind in those hours you program. To get that coherence of mind, deep uncertainties about how your life and your projects are unfolding must be met successfully and transcended.

And for any programming project of magnitude, while you're in the midst of it, we're not talking of "Let's give it five or ten minutes now and get some progress!" Even given a pause of no more than, say, fourteen days, it can easily take an hour, or two, or three, just to get into touch with all the facts of the present programming so as to move it forward. It's like a giant engine with lots of dials and levers and knobs and they must all be adjusted carefully, and oil applied, and checked and double-checked. Only then the ignition switch is set to ON and the roar of the engine starts and liquid gold is generated, the liquid gold of thinking.

As for the programming of the high priority task in its step 3, which has substeps or 'sweeps' as we call them, this is about finding out which, if any, 'elementary action' on the matrix pushes it at least a little bit in the direction of one of the fourteen match images used in the pattern matching in the approach to FCM we take in this volume.

The first sweep could have been done by enlarging a step --since it's only about initializing and we really do not need the whole FCM TRANSLUCENT loop to go through all the levels once more before the next sweep. But while it's okay to be aware such possible improvements in speed, this isn't a significant delay and it's OK also to not squeeze the last drop out of the lemon that the computer is in your programming. Sweep number 1 sets the number of the elementary action to the first one; it is stored in luxury value# 44. Note that this number is 15, not 1, because it refers to the number of the link, and just

before the 10 elementary actions comes the links to the 14 match nodes. Luxury value# 41 is the winning elementary action so far; we can initially just set that, too, to 15. Its pmille match value is stored in # 42 and this we set to BASIS. And we already have # 43 with the best match pmille for the main image. The repeated copying-over of main image to copy image prior to each elementary action having a go on it, is done in sweep# 2.

We can call the functions doing the sweeps for anything we like as long as we fit their warps into the slots of the HPTSWEEPS. We have already suggested 'HPTSWEEP1' So:

```
HPTSWEEP1=
|IN:FNWARP
SX
15
44
iX
KW
15
41
iX
KW
BASIS
42
iX
KW
iX
NODENEXTSWEEP.
```

How to put the function, via FF, into the array of sweeps we have already outlined before, so we won't repeat it here. [See chapter ..MORE FUNCS AT LEVEL 10 MILL.]

For the sake of being 'honest to the process' in this explorative work on thinking as such, let me add that prior to writing this chapter I had some vague senses of there being some possible confusions in the code in the earlier chapters, so it took time building the sense of overview before I actually did some new coding. And in building this sense--by reading back and forth and using much the search function in the B9edit text editor--I did find indeed a couple of things to improve and indeed one thing to correct in the code. This is to re-iterate the point that feeling and thinking goes together; that the quest for clarity of feeling may involve using some features of feeling as a 'radar' to pick out something that ought to be fixed; that feeling may emerge from a thinking that, even if the conscious level of thought for days have had plenty of other things to focus on, in fact has algorithmic clarity and exact relationship to detail.

MAKING ROBOTIC FCM PROGRAM:LEVEL TEN MILLION SWEEP# 2

For the nth time, I begin a chapter level 10 mill "DONE"--only to rename it soon enough--because the complexity of that level--let's say the "most simple complexity that can do the job"--or "adequate complexity"--has in it a bunch of necessary nuances that weren't that easy to pick out 'from a distance' Writing a program while writing a book means that such an under-estimation of complexity affects the layout of the book--it keeps on getting larger than the earlier parts of the book seem to presuppose. It's

tempting to just 'get the program done' and then write the book to its completion but then the thinking behind the program lines would get inadequately documented. Yet I see that the character count of this volume vastly exceeds the previous volumes. Be it as it may, it has been a lot of novel works on my part between this volume and all the previous ones; so much so that the book is almost written in a new life context and part of the process of thinking is to bring what is relevant of the context onboard the ship, so to say.

Of some interest here, in the philosophy of thinking--"pure thinking"?-we might say, is that the experience of making a program when one knows a programming language structure like G15 PMN, and one has a visualized, meaningful goal, means that a landscape almost by a form of 'fractal magic' unfolds before oneself, with hillsides and mountainsides and sleek mountain rivers and some beaches, and, by the rules and tokens of the language, one must take one step at a time. One can glimpse the distant mountain but to get there one must have a good backpack and go through these other places. And in going through these other places, we may find, as Gandalf, Aragorn, Bilbo, Frodo and the others in J.R.R.Tolkien's tales, that there is much to be experienced and errands to be successfully managed. John R. Reuel Tolkien, in that sense, was a master programmer, though born in the 19th century, half a century before the first intimations of computer programming in humanity--first perhaps through the works of Kurt Goedel, then most obviously by the works of Alan Turing.

And, by the same token, can we not imagine that the process of creation--Alfred North Whitehead spoke of God as "the Creative"--takes place, in the sense of George Berkeley's idea that the world exists by virtue of it existing in the mind of God--in exactly such a meeting-point between a goal and a formal language? Attention flows in towards the goal, through the formal language, and a landscape of great beauty unfolds. In going into it, even as one has oneself shaped the goal, and knows the language intimately, there are surprises, there are tasks, it is engaging. Feelings come in and--as we have touched on some places in this and in the previous volumes in this series--these feelings are not merely by-products of the thinking but can be informative essences as well as forces in the thinking.

And let there be no doubt that I know G15 PMN well: while I have to constantely look up definitions--and sometimes forget for a while that a definition does exist, such as for permille, so I code it afresh--the whole shebang is my doing. And yet the shaping of a program is a giant exploration process. In programming, the old adage--which I believe I made myself--and which can be misused so as to justify laziness--seem to apply a lot: 'walk where it's effortless to walk from where you're standing, and which is relevant to where you're headed!

As for walking on with our program, we have some sweeps to do. Sweep 2 and onwards are performed on the principle that there is still more to do in this 'loop' [I quote the word for the real loop is TRANSLUCENT; here we merely let the sweeps be updated and the counter be reset until done with all the elementary actions.] The last of the sweeps checks whether there is still more to be done, and, in such a case, puts the sweep counter back to this, sweep# 2 again.

Here, the main image is copied using the typical copy-routine in G15 PMN called *FW*, over to the copy image, and the present link number to the elementary action is offered to function *SOFTACTIVE* defined in our chapter *'LEVEL TEN MILLION FUNCS'*. *SOFTACTIVE* expects the warp to the present foundry on top of the stack. Luxury value# 44 has the link#.

```
HPTSWEEP2=
|IN:FNWARP
SX
44
iX
WK
iX
SOFTACTIVE
```

Alright, let's do the *FW* part also. It has 'from warp', 'to warp', and count. As a comfortable and rich rule, we tend to define matrices so that they have at least one row extra, in addition to a bunch of extra bytes right before it when we do the *WWYYMATRIX*. The *FW* is superbly fast and so, in order not to focus too intently on whether the matrix starts at row 0 or 1, one might as well include an extra row in the copying-over process.

```
MATCH1IMG
LK
COPYIMAGE
LK
112
161
MM
FW
iX
NODENEXTSWEEP.
```

It takes a flicker of a flicker of a second to do the MM otherwise we would have 'hard-coded' in the result of that multiplication--somewhich which makes sense if the MM is is inside a loop going many hundreds of thousands of times but not when it's about to be performed fifteen times. The number '161' is simply 160, the number of rows, plus one.

MAKING ROBOTIC FCM PROGRAM:LEVEL 10 MILLION DONE!

As said in the intro--whether or not contradicted in some of the writing process between it and here--by this book, we complete most of the app--at least 800 permille. Since the app should be available at g15pmn.com, ie, the app# 1005769, you can use this book as documentation to get to grips with the program and whenever the program has some of its lines changed compared to here, this writing-thinking that created the program, it is because in the 'dialogue' with the noble G15 PMN PC it was apparent that a change had to be done to get the program to be done. But as you notice, the mere fact of knowing a formal language well is enough that this dialogue goes on inside you as you sketch the program. Getting it to run on the computer is almost trivial; and yet the computer is essential in driving forth this form of consciousness in humanity that the programmers represent. And so it is my intuition, and belief, that the intuitively-designed G15 PMN can be the core of the formal education for all kids and up, which, together with the mind-dance of brilliant English as in P.G.Wodehouse writings, enlivens the mind for a joyous life.

The app won't have many changes from this book. Elementary actions will have an algorithm that does something like this: divide the image matrix up in equal portions. Let each elementary action be swapping one part of that portion with another--rather like a jigsaw puzzle with large parts, in which one part is taken out and inserted at another place.

If the construted image have been made with a knowing that this is about the type of action that will be applied in order to create more order in it, it should be possible for the human interactor with app# 1005769 to see order increase with each "Go, Improve It!" action through a click on the keyboard. The FCM network will do as much as it can with each image. And all the features of a full robotic network have been given a vast type of 'Hi world!' go-through for the budding advanced G15 PMN programmer.

Onwards to sweep# 3. This is the soft-activation of the whole range of fourteen match images, and with the setting that they work on the copy image.

Before that, let's make a small comparison between the notion of doing *NODENEXTSTEP* and *NODENEXTSWEEP*, in a function that is called from a node which is to do several things in succession, and possibly with repetition--and the notion of doing such as LL .. LO loops -- or just calling many functions after one another.

The whole mesmerizing difference between robotics programming in a node network and more in-computer programs with a normal, 'classical' style of loops is that of the dimension of duration--put simply, time. The algorithms that runs robotic hardware has an 'obligation' to stay in touch with that hardware and not just rush on. So time comes in, we might say, and sweetly messes up the prestine order of algorithmic computer programming. Not only is there a waiting action before such-and-such move of the robot arm or whatever, but also there is the constant update of other functions 'here and there' in RAM --functions which add up to the sense that, while in principle there is one sequential action sequence, on the human experience level and for all practical purposes from the 'perspective' of the robotic hardware--there is a kind of "parallelism". We gain next to nothing having an actual parallelism at the CPU level, ie by having more than one CPU, but the perspective of parallelism makes sense when analyzed in terms of (desi)seconds rather than, say, microseconds.

In algorithmic terms, this is what I like--whether it's precise or not relative to classical computer jargon--to call 'polling'. We might say, here in FCM, in the 3rd Foundation G15 PMN, the *TRANSLUCENT* loop is 'polling' over a number of functions, organized by level numbers.

And so which functions that do something and which are waiting on others are decided by a flag they have here, which we call 'soft activation'; and those that do something, may turn on and off the soft activation of other functions (ie, functions belonging to other nodes) in our FCM computational node network.

The *NODENEXTSTEP* is then a way to organize the doing of several things, one at a time--and letting the *TRANSLUCENT* go around polling all sorts of other activities that may go on in the network--and so also for *NODENEXTSWEEP*. When, in the case of sweeps--considered here as 'sub-steps' under one step--are complete, the last sweep should call '*NODENEXTSTEP*' in order to signal that the sweeps are complete.

In an LL .. LO style of loop, the counting as for cycles happens in the background--that is to say, the G15 code underlaying PMN does the update of the i1 counter and the check of whether enough steps have been performed, at the LL point and the LO point. The LL .. LO style is snappier, but the *NODENEXTSTEP* and *NODENEXTSWEEP* incorporates a robotic sense of 'there being possibly many motors and several cameras and so on being active rather at once' and so is psychologically experienced as more 'organic' and more resonant with the 'sluggishness of the material world' in which things are not just pure abstract algorithms but also motors which occasionally need oil and which may or may not squeek and which may or may not have enough horse power to lift the darn cup or whatever it is ;)

So for instance, when the robotic has successfully moved its arm or something, there is a *NODENEXTSWEEP* or the like that then calls on an analysis of what is coming of data from a relevant camera to see how well the arm is holding on to that cup. These little collaborations are all about having a good resonance between, on the one hand, the formal abstract computational node network, and, on the other hand, the robotic machinery but there!

Let me also add, it's something about the explicit goal-setting process that must be in negotation to what we can call the deeper and more encompassing goals associated with the activity--in this case this book, and the book series. A goal--'to get the program done'--may exist in the mind simultaneously with the feeling that something about the programming, and thereby thinking, process, has been left unsaid. And this sense may prevent the degree to which the thinker, the programmer, has an ease with which to engage with the programming, because the sense has in it core information that can be unfolded given that one gives it time and space and energy.

This is one of the chief reasons why I don't use algorithms to structure my days and months, but have a notebook-approach to it. There are actions, such as car-driving, that requires full attention to the task at hand with no wavering of mind in order for it to be done with the safety that it deserves and requires. But when we are working out essays on the art of thinking as such, it is 'against the law' to become of an instrumental mind-set' and ignore quiet perceptive feelings of unsaid things in order to reach a concrete milestone.

And indeed in much of life, this is required, and in all of art, and much of life is art.

There! The next sweep uses 'From/To Soft Activation 5', or FTSOFTACT5, the number 5 indicating the the method of activation is to set the activation flag to 5, rather than 1, to indicate to the 14 match image nodes that the copy image is in focus. *FTSOFTACT*, rather than *FTSOFTACT5*, was used in *HPTASKSTEP1*. Here we will have repeated use of *FTSOFTACT5* until all the elementary actions have been tested and the best match permille number recorded, alongside which elementary action that gave this number.

```
HPTSWEEP3=
|IN:FNWARP
SX
1
14
iX
FTSOFTACT5
iX
NODENEXTSWEEP.
```

Next: Here, [1] get the best-performing pmille--we already have a routine for this, SOFTACTWIN; [2] compare this number with the number already recorded--that's in # 42; [3] in case it's better, store the new number back there and also note which elementary action that delivered this improvement in # 41.

```
HPSWEEP4=
|IN:FNWARP
TX
1
14
JX
SOFTACTWIN
S9
```

That is to say, the i9 contains the match number for this elementary action. Let's see if it is better. [Note that we do not of course say that the computer should 'see' whether it is better, nor did we instruct the computer to 'note' the value. Rather, we as humans, real thinkers, real programmers, we are the ones doing the noting and the thinking and while doing this, we also program the computer to do something that we might say illustrates a formal aspect of this--not transferring psychological terms to the machine.]

```
JX
42
WK
i9
GE
SE
EX
```

Note that in typing it into my specially designed 'native' G15 programming font--also called RBOTFNT or RobotFont -- using the two-column approach [which contributes to a sense of 'conversation within the card' in how it is psychologically read], the use of blank lines is a recommended programming policy around 'jump points' such as SE and D2.

The routine goes on in case there's reason to update. So let's update:

```
i9
42
JX
KW
44
JX
WK
41
JX
KW
JX
NODENEXTSWEEP.
```

The following sweep could, as I believe I've already hinted on somewhere, be written as an extension of the previous sweep--for there is no need to wait to do the

forthcoming update of the elementary action number. However the sense of leisure and good luxury that pervades G15 PMN programming must also be part of its essential FCM programming. Why pack two sweeps into one, when it looks good and has an ease to let it be two of them? True, the TRANSLUCENT loop will work on polling all the other nodes between sweep# 4 and the next, # 5, but what of that? Polling has a nice feel to it even if it's unnecessary between these two sweeps as this is the controlling node in this case and we know the score. However, in another, expanded case where the first step of the programming process may be much a copy-paste process of this program, it may be a delight to see the sweeps each being small, as some extensions may be on the to-do list to implement here and there in the sweeps--possibly some extensions that require a polling, not just as an option, but as a necessity. Anyway, here's the completing sweep, # 5. This is what we want it to do:

It increases the elementary action number by 1, and as it goes in our case from 15 to 25, there being 10 such actions or subtasks, we check whether it it exceeds 25. In such a case, it's about time to do *NODENEXTSTEP*. The next *STEP* does the 'winning' subtask directly on the main image and stores it back to disk--or does nothing in case the improvement is too marginal. And there is a step that gives control back to the G15 PMN FCM Spreadsheet, or GPS as we also call it, where the interactor--which may be you!--watch the results, the image gradually gets more and more orderly in the sense of matching up to one of the 14 images until, by the 10 elementary actions as here defined, there isn't any more these can done given the input that the interactor, you, provided. The input you provided could be something like a light messing up in GEM of one of the 14 elementary images.

Whether or not we have described clearly in such as step 1 how to get the main image matrix properly updated by copying in that which is on disk, as given to GPS, I will have to look at when I program the app itself. The app contains the corrections and additions necessary to get this program to run on the computer.

```
HPSWEEP5=
|IN:FNWARP
TX
44
JX
WK
25
LT
D2
NODENEXTSTEP
EX
```

At this point, the routine continues given that not all in the series 10 .. 25 have been checked as yet. [The reason why a sweep can call 'NODENEXTSTEP' without further ado--ie, without calling for an exit of the sweeps first--is that the routine that 'pulsates' the sweep are inside of one step, which is also why we call the sweeps for 'substeps! And as soon as the node exits, with the updated step number, there will be no more calling on the sweeps before again, in this case, step 3 is reached with the resetting of the variables used by the sweeps. It will go straight and without a problem to the next step.]

The sweep, having more elementary actions to call on, updates the link number:

```
 1
 44
 JX
 KU
```
And changes sweep number to # 2 and in effect creating a loop:
```
 2
 40
 JX
 KW
 JX
 NODENEXTSWEEP.
```
Hereby, we have the sweeps for step 3 done!

And that also, of course, completes step 3. We have earlier described five steps. The fourth step compares the pmille the main image got earlier on, at luxury value # 43 with the luxury value # 42, the pmille produced after the "probing-action" found most successful by the sweeps, had done its job. In case the # 42 is significantly better than # 43--just how much is something to be tuned while doing the program--the elementary action is soft activated again--but to do the job on the main image.

One more cycle, and we're at step number 5, which of course is to set focus over to the GPS aspect of the G15 PMN FCM computational network, or networks, so that the interactor can view the result.

Now it should be fairly straight-forward to program the app. The completing steps, and the elementary actions at the highest level numbers, are after all very easy.

YOUR OWN DANCE, YOUR OWN EXERCISES, YOUR OWN SKILLS

We must all fight stupification of humanity. Robots are here to help us do things that are too stupid for us to do and computers are not here to outwit us. So computer development must be kept in check, while mind-powers of us all should constantly be honed and fine-tuned. Just as you must improvise in your own dance skills, and bodily exercises, every day, so also must you practise two-hand typing on big keyboards in front of PCs and learn to dance in thought, be it just some minutes, every day, using the calm meditative focus of the green text editor and programming editor in front of you to assist you to sharpen thought, and to help you be aware of your own thoughts by expressing them. This awareness means that you are upholding a dialogue with yourself.

From thereon you can be social in truly meaningful ways, and build affluence and experience in sex, learning of the importance of beauty, also spiritually--always more learning--and all this, including the wealth, will have meaning.