to imagine us making the cat

1. we have vision input

2. we have independent modules

(keep map, keep objtype)

3. we have drivers (Bev cyc)

4. we have simulations (planned from table)

5. we have beacons (another table for obj response)

6. we need to combine them

--

we end up with two optimizer / agent to handle the output

starting from the endpoint of triggers (modstream) onto simulations

ontop of improving how the modstream grade and prioritize trigger

we also need to improve CYC modulator on judging and continuation

among with some later improvements relating to

checking , independent limits, etc

Page added to explain content

QNA

" So what exactly drives the whole agent cat? "

- Two systems; seemingly at different level (not actually independent)

  A chem. to trigger to modstream to sim

  imagine you have independent modules that process vision input ; turns them onto triggers along with all the other necessary map details; such as objecttypes (confirmation); actual map data (actual map that the agent is at); routes; goals; of such map

  So we have one line of processes down below that ends up making Triggers;
  And rely on chems to talk to each other (also built up)
  Triggers by default is processed by modStream; to become latches; and then simulations

  B cyc. To cyc breakdown; to n; to fill with sim; to refold; to judge
  (on which judging element is missing*)

  Then imagine you have an entirely separate system; called the CYC driver

  Which is a (process reward model); or decomposition of "two line of policy" ;

  policy on which starts from a inherited baseline; before being "run" and then realizing which part contains more (error chems; at the bottom level);
  and then "after being run" gets judged
  Between two (at the start; this would be the inherited vs the real run; real first run)

  So you have a CYC system running on its own; initially with baseline

  Composition of steps; that are broken down; as | n | 's n's are made out of triggers or sims that are deemed most important at that time (these happen at the chem. layer)

  But once they are done , these |n n n n n |'s get refolded onto one |cyc|

  And that cyc is (somehow) judged along the tree-of-decision; as to which or how to proceed

  Given which one of the route (this vs the baseline); has which component of the steps

  That are the better one; ontop of updating the composition steps of those;

These also judge for whether the step of the current |Cyc|cyc|cyc|

Three step to satisfy the cycle; was done correctly;

For example; if a WAKE – HAPPY – WAKE ; ongoing cycle

But the HAPPY parts throws a lot of errors

Then above a certain level of error; the cat continues to | JUDGE |

That the next |cyc| that HAPPY breaks down onto;

Needs to account for that "Below threshold" (failure) tree continuation

Resulting the next Cyc's that also contributes to the

"highest context" that becomes a mod (to join the Chem.part of the system)

(all of which is vying for the position to publish the "most fitting trigger"

For modstream (of the lower layer); to accomplish and make a sim out of

This assumes that cyc-modulator magickally knows how to proceed

With the steps and how to judge << this is true; the whole system doesn't explain how

This would be done (and simply points to things such as Process Reward Model,

Or any such models that lead to decomposition of steps; and choose the better step,

Whilst; executing the context of the higher chain; and deciding whether the threshold;

Meets the requirement for continuation of such chain)

This also assumes that modStream and simulation types are all hard-coded and table-given

And their filling and their satisfaction is "presumably handled elsewhere" or unmentioned

Which is not yet discussed IF discussed; it would look like

"THIS CYC ; would contain THESE SIMULATIONS ; having these SATISFACTION TABLES"

So anyhow; we have two competing level of system

On which the satisfaction of

Each is actually linked;

But the bottom layer would go on UNPERTURBED irregardless of the top layer

Just producing the triggers that are considered by the modstream to be

Of most expedite of the current “millisecond” or such scale of time

Meanwhile; any coherent action is actually more dependent on the TOP LAYER

Which the CYC is broken down by the CYC modulator; carries with it

A SIM Type table; and a highest-context to become the (special module)

(modRoam in the park case); but these breaking down

And getting judged has to be done together with the bottom layer

This is done by breaking down the chain of steps (of one cyc)

Onto |n n n n n| like bracket;

Where it is assumed that the filling of these | n | ‘s

With the “somehow relevant” SIM of choice (highest trigger);

Will eventually always result in a one |Cyc| bracket

That always knows how to decide which |n| and whether the |sim| it contains is correct

This could be doable* (but poorly described in terms of detail)

If the CYC in question; when being broken down;

Carries with it a set of anchor (for every different CYC ; i.e. YOWL / ROAM / EAT/ CHEW)

On which the anchor would set itself a “table of simulations”

On which the context of these “table of simulation” is perceived properly by the

modStream; when deciding which Trigger matters most.

So; in summary;

We have two layer of the system on which the completion of each

Competes for the attention of the modStream (a trigger grader)

To compose a simulation which is defined by the sim-table

Which is defined by the |cyc| that is currently being run

Which is defined by the |cyc modulator| that judges the previous |cyc bracket|

On which whether

That |cyc bracket| broke down onto a series of |n n n n n n|'s

That got satisfied properly or improperly; by the simulations; that was decided

This is the gist of the operation of the system;

The whole writeup was to depict the details on how each component should be built

Or run on its own; admittedly; it was difficult to convince how this is

Properly buildable; without mentioning or including

1. The SIM Table
2. The CYC Table
3. Nor
4. The CHEM table

(for each relevant scenario)

Alas. The prediction of these Tables require going from a Baseline SET

(say . 11 CYC's 300 downstream CYC's , 100 sim types, and 100 chem types)

Spread across those modules mentioned;

All of which go onto the same flow

The purpose of the paper is actually only to justify the Flow

On which itself is, :/ actually just a posit; that would only be test-able

Given that these components already exist;

So to summarise (actually already written in the Chempool bottom part)

Currently missing parts are:

1. **Sim table**
2. **Cyc table**
3. **Chem table**
4. **Cyc modulator** (PRM between two line of compositional resolutions)
5. **Trigger table** (for modstream to know which Trigger is highest priority;
   Half of which is determined by CHEM^^^ stacking
   (so half of this is already predetermined by
   Each module running hard; and producing Error that stacks

6. **Sim resolver** (a game world to resolve)
7. Supposedly
8. These table will tell how to
9. **Decompose cyc; judge cyc;**
10. **Decide chems** of each module
11. **Decide triggers** and subsequent SIM that comes

Some of these should be “experimentable” using handwritten components / substitute

But the PRM is critical / process reward model (alphago like break-down-of-the-steps?)

On which without such knowledge or implementation; it would be impossible

To check which route was done better (which CYC was completed what way;)

Although… if without such a PRM method; we could possibly

(as with DEMO 0.01 suggest)

1. Simply write the 11 CYCs
2. Simply write what they break down onto
3. Handwrite / hardcode a simple simulation (any sort of bubble on a map would do)
4. Steer the resolution to fit the actual behavior (by giving it simple conditions)

5. Then cheat the PRM step; simply show a Tree; on what was decided

6. During the run on a given scenario (so no learning was done;)

7. And then we only show how the CHEMS were accumulated

8. And then only show how the highest TRIGGER was selected

   (both by the Chem and a multiplier weight table)

9. And hopefully also how a learning step was done
   (in this case; about modstream deciding
   Whether a value of a CHEM (stacked)
   Or a value of a Trigger vs Context (CYC)
   Is as high as it should be,
   And lowering it if not;
   Prioritizing other table next time

10. This is all assumed to be run on a slightly slow | n n n n n n | block of execution

11. On which instead of some 10 milsec for one |cyc| (nnnn) block; instead we run it in 10 seconds

12. And simply depict how that 10 milisecond would've concluded itself; (or a chain of 10 step of these) would've concluded itself; by handwritten trigger-to-sim (and chem. stacked)

13. And this is the essence of the DEMO 0.01 (before even adding Checking) (BM1&2)

14. Checking and Limit (BM1 & BM2);

Should actually be critical / contributive; producing their own Chems & trigger
That would be highly graded by the modStream

Given a scenario was run properly or not; but demo0.01
Only looks to run a
Simple run where a cat decides "what to give attention to"
And how does the "chem. stacks when giving or noticing such attention"

And subsequently "how modstream then decides"
Filling & concluding the | n n nn n | block with Sim; getting combined
And thus judged (manually; cheated without any learning)

Apologies if the paper was confusing to mention all of the steps above
Without mentioning the missing components

(it was actually mentioned in BIG NOTE)
and
(it was actually mentioned in Chempool page)

But a overall overview of the entire system on what happens
When these things are missing
Should've been given

Anyhow, back to resume the original paper
(with the skim images to depict parts of the process)

P1P2

Vision Oriented Hippocampus

.For cat.

Building vision processing nodes

that could integrate with simulation

and priority decisions

.

P3	*skim*	*P26*	*big note*
P11	*qna*	*P30*	*paper*
P16	*example*		

1. Skim over some pics
2. Ponder QNA
3. Example
4. Big Note* (weakness)
5. Paper

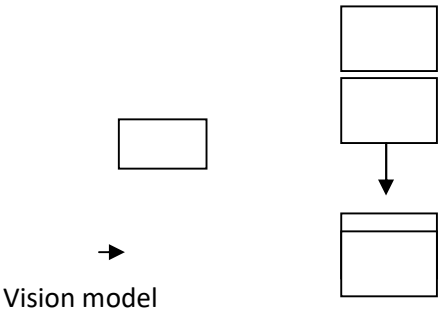
Or skip to page 30 / page 60 for modules and their hardware specifics

Mayhap got instant intuition why its built this way

(with the independent outputs)

Some pics to skim content

Pic 1



Scanning Trio (map,points,routes)

Modscan modloc modnav

Tracking Trio (shading, attention, tests)

Modwatch modlook modtude

Modstream and modlatch

For simulations

Pic1 intuition:

Modules that process vision input; with a set Output

How to make them both "independent?"

yet "goal connected?"

(with a driver system / satiety system)

Thus we do this part

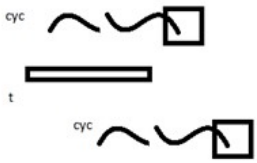
(:: we add chems.)

P1P2 seeks to posit these component

(if they botch; they still produce; just an #ERROR chem.)

(they end up in modStream (pre-simulation) chem. Bucket)

Pic 2



11 core epigene drivers

Of a presumed cat

(to be tested and improved as we work)

11 of these going thru a modulator

Getting decomposed onto a

| n n n n n | n n n n | n n n |

Blocks; where | n |

Is a slot for interjection

From simulation;

Pic2 intuition: *to be composed and tested and altered

OK; so we have 11 of these

Things like; CAT WANTS TO BE HAPPY

WAKE-HAPPY-WAKE*

; CAT WANTS TO BE ADULT

WAKE-ADULTS-WAKE*

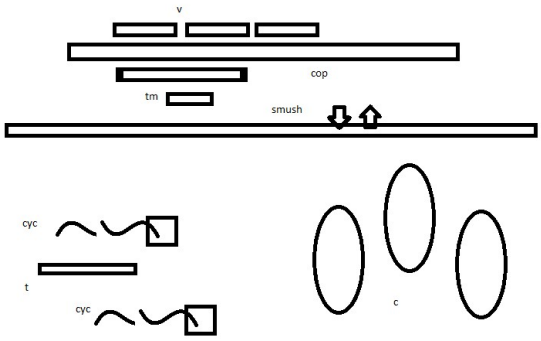
; CAT SATIATES HORMONE

ADULT(stashed) – ADULT!! (trigger) – ADULT.

And somehow; these has to play with current awareness

(map n object context; to initiate their satiation)

Pic3



Our intended first demo.. with cycs going forward

Creating themes,

todefine a |cyc| block to decompose

To |n n n | slots

N are interjections / actions / checking

Fill from modules

Each | cyc | is then re-folded; and judged*

C

there is just a container for objT objP , map, etc

v & t both work with these; they are staple Formats

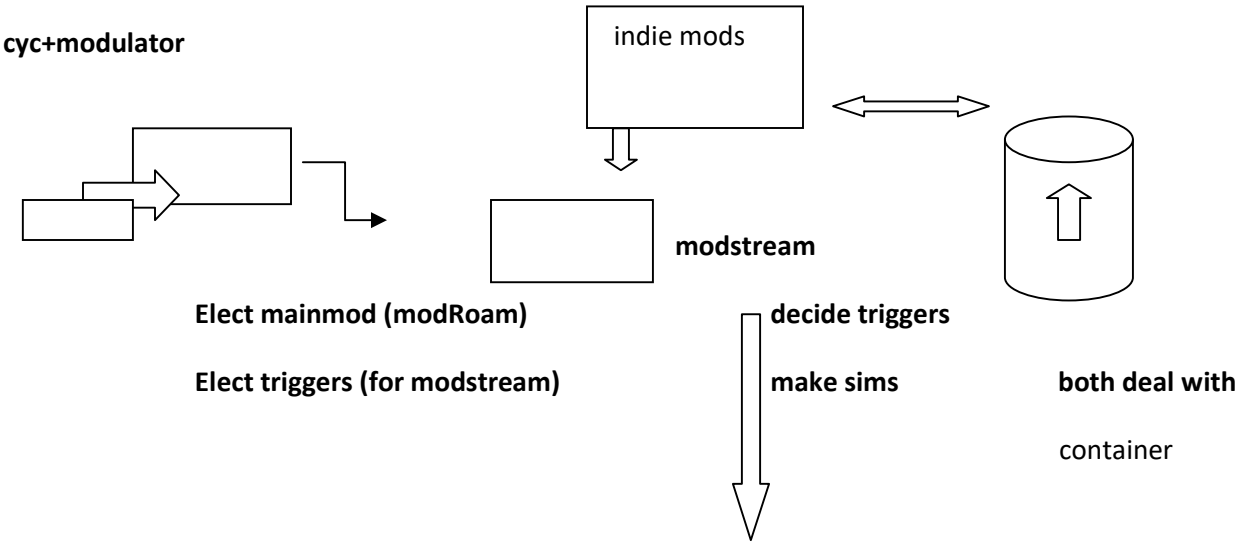
We described the circles on the right; the container for objt / memory

We described the bottom left; a driver system thru a modulator (morphospace)

Top part is about this p1p2, where independent modules produce trigger

In accordance to awareness of the other two parts

Pic 4



*

where CYCs & CYC to fulfill come first.

Indie mods operate on the side.

Both of them publish triggers. Especially the

CYC related to map

(the highest context for goal direction).

CYC-map publish modRoam instead

which will join the indie mods, in publishing triggers

all of which is graded by modstream for publishing simulations

***special rule** (for modRoam creation, highest context)

We assign CYC-map among 10 other CYCs to agent cat

CYC's are decomposable cycle needing to always be fulfilled
CYCmap is special because modulator is aware that the highest context (atleast in this demo; would be the MAP, this is mostly handwritten)
CYC-map break down onto |MAP?| (cyc cycle) to become (|n|)

(WAKE-MAP?-WAKE) (name and composition pending),

which triggers the creation of modROAM associated with the map

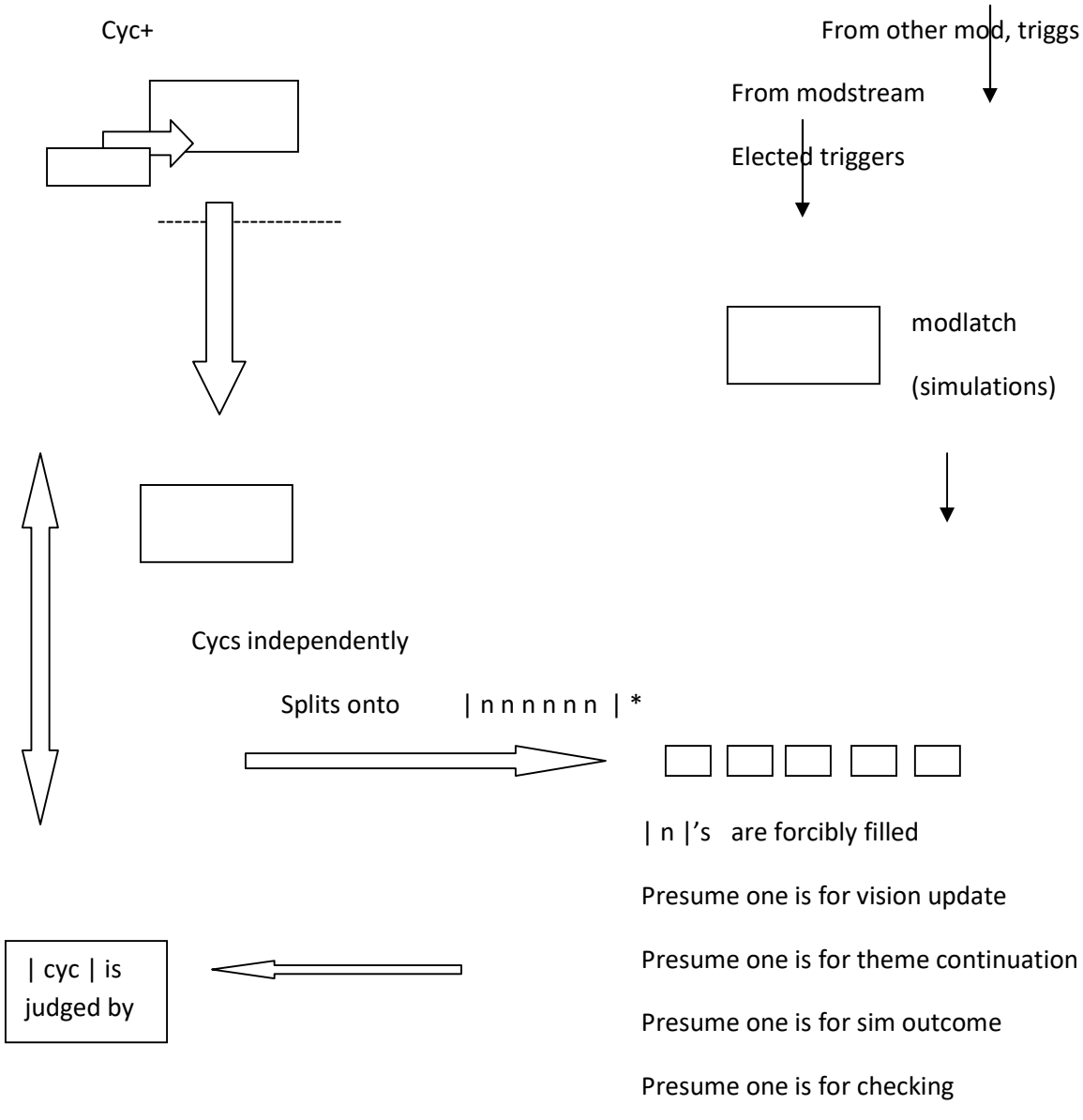
modROAM joins other mods, as the publisher of triggers

triggers get tabled and elected by modStream,

publishing the simulations.

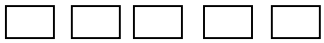
Other CYC's (as CYC map do) also break down onto

| n n n n| > continuation chains; which | n | is to be filled with interjection from those sims



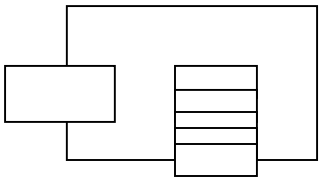
What ends up happening

| 6 (?) blocks |



*

| becomes **one** | cyc | | (of the 3 chain)



x 11 + modulator

Cyc with its modulator

Still wants to go forward

(tree search , or)

Judge; and make tidier

Produce triggers again

One of the 11 CYCs

Decided by modulator (step search)

Become next |cyc| to decompose

easier reasoning step

modstream with triggerpool

Deciding..

Modlatch with sims

eventually talking back

to |n| ‘s

Pic 5

Later on…

Screens are..

Lingering theme that lingers too much that they are just

Made onto a static depiction (cat-overhead; etc)

For quicker referencing

Chempools are..

Chemical emulation system.. for attaching to module

To time/ combine / trigger intiations

Modules are..

Independent organ-like system.. divided by their critical output

They usually carry chempools & screens with them..

Stress^
Stress^^
Stress^^^

Chems aggregate, compounds, each with different reaction table (see table)

Used as trigger

module described

module described

Produce trigger

They end up in modstream grades

Theme makes [latch]

modLatch keeps the [sim]

the outcome becomes [update]

*one of the more important :. map-related-CYC becomes a main module

(i.e. modroam tracking #roamDONE (chem.))

>

onto detail

A Map of Core Depictions

Core challenge: depict a

Postprocessing unit of

Vision Model; and combine them with Goals

And sequential queued goals

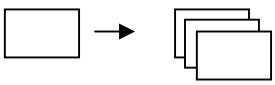
Core queries: depict the important

posits

1. How to process Vision Input by Independent Modules

- A number of Output-dependent Organs

  Called Modules; having their own chempool



produce something

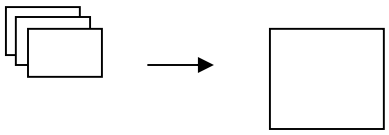
regardless of condition

*

Work with output

2. How those Output are first transformed onto Simulations

- Modules accumulate Triggers
- modStream grades the triggers
- produces Themes
- Themes produce Latches
- Latches are a bundle of Simulations
- Handled by
- modLatch; to resolve; and update



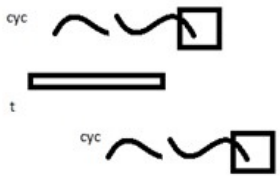
accumulate trigger

trigger graded

become themes

.

3. How CYC's come to play



Everything starts with CYC's that needs filling

We do things to fulfill CYC's

CYC's are decomposable 3- (ROS-like) decomposable

action sequence; representing

states that are to be broken down until it become

actions (muscle % tension)

CYC's are tested and learned as it happen

The way of doing this is to break it down to

*see big note; author is not sure

*how valid is the protocol of seperation

*e.g. between (many n's) for each;

*or a decreasing amount representing

*narrowing window for interjection

*but we depict one version now

Break it down onto

| n n n n n | | nnn | = one |cyc|

| n n n n | where

| n n n | |n| is a

processing or queue node

for a thing to be focused

and call for an update

or be refreshed

n 's are usually filled with v .or. t

v (vision nodes) – as with scantrio & tracktrio

telling or putting in their highest Trigger

(and the subsequent sim to solve from)

T (themes) - as with modstream's theme, the elected current
trigger to think about,
to solve for their latch & simulation
and to report what the decision is

or other nodes to come; such as CHECKING
or perhaps; a search

These are described this way; because they are yet to be tested* (!)

And this is a proposal to make it in this example:

EXAMPLE:

Say. Yowling.

(say we go straight to YOWL

(presuming we don't write the prior (something like

v WAKE-HAPPY-WAKE*)
v HAPPY-PEACE-HAPPY*)
OTHER-PEACE-OTHER*)
PEACE-BOGUS*?-PEACE*)
BOGUS-YOWL-BOGUS*)
YOWL-YOWL?-YOWL*)

again, these are arbitrary; and they have top and down nodes that come from a certain baseline cycs satisfaction, we need to decompose them and test a lot; to figure out which is the morphospace

(and how to write later)

*notice how these are Homeostasis oriented satisfaction quotas (satiate, stable)

- Pretty much a CYC type that demands the agent
- Try to be happy; otherwise the CYC goes around
- Producing THEMES that end up being SIMS
- That result in #ERROR or unfulfilled Chems
- (when it goes onto the simulation node)

Problem is; we are currently trying to approach the Agent by, missing major components (reason why we doing the current indie mod emulation; despite pending on the PRM)

Constructing an agent that can

RUN CYC's

Thru SCENARIO (arena)

And See if good..

(apply PRM ; process reward model)

And thus improving as we go along

..

(see big note about our pending PRM)

Also that, trying to construct

a good CYC table, would require

..alot of testing ..

thus.

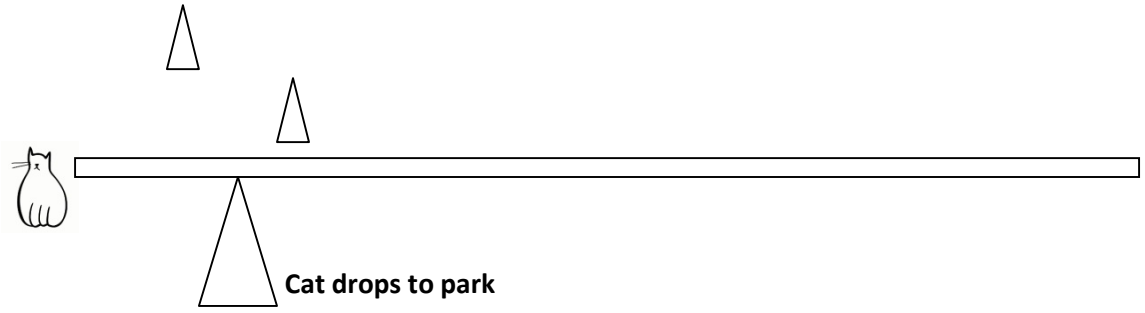
(thus doubles as an appeal to test!)

Cat example

dropped onto a park

Hippo nodes

(becomes a | n |)



Cat drops to park

ScanTrio (makemap) . VisTrio (track obj)

Initiate:

- **drops to a park scen 1**
- **panics! (scan&track trio) both recognize everything as novelty**
- **but. CYC was already running. (several)**
- **one of them, say CYC-Map (namepend)**
- **Knows whats what, sends a TRIGGER(THEME) to modStream**
- **picturing the instance where that particular THEME becomes MAP**
- **and that MAP recalls the relevant objTypes and confirmations**
- **mechanism of CYC-MAP electing highest mod is hardcoded***

(map = actually context, highest context wrap always exist)

(THEMES are mostly hardcoded with alterable nodes for learning)

- So thus we have a MAP and a MAP update,
  - This causes the highest (map relevant) THEME
  - to become mod

- In this case, a modROAM*(park1),

- Other cases on which this does not happen / differently are; for example
- If the Highest CYC is danger recovery or something else like
- Bodily harm / recognizing inability to (*core future)

- Biology may infer that actually some of the core processes

- Are dealt in the body's level of organ, (hippocampus being just one of)
- Letssay a modCrisis
- But ; since we are just trying to have a DEMO 0.01
- Which is a ROS agent traversing the land
- Lets not worry about that yet
- (we haven't port any biology / do such tree-ordering yet)

with modRoam now; a independent; unrelated to other cycs

lets depict

For mapping

At this point the cat has

CYC-MAP > KNOW – MAP – KNOW

CYC-ROAM > KNOW – (ROAM -) – KNOW
CYC-ROAM >

If don't know

CYC-MAP > KNOW- MAP – KNOW

CYC-MAP? > KNOW- (MAP?)(theme for min. mapping) – KNOW
CYC-MAP?% > KNOW- MAP(%) - KNOW

CYC-ROAM > KNOW –(ROAM) – KNOW

(roam = modroam)* (ongoing)

List of all **(some of the pondered)**

- CYC MAP
  CYC MAP?
  CYC ROAM

  -
  (from modRoam)

  CYC ROAM >
  CYC FOODPLAU
  CYC NAVS
  CYC TALLY

    - Foodplau could be decomposed onto
    - CYC STALK- CYC HUNT or
    - whatever we decide to make for the Tree of
    - Hunting that happens

CYC PORT - actively run to double up creation of subCYC or decide such focus

(ensuring a prioritization of changing cyc / smoothing transition)

CYC ACTIVE - actively run to double up task-related trigger

CYC REFRESH - actively run to ensure backlog of triggers are queried / asked-to-check

CYC BODY - actively run to maintain body health – haven't made

Say we keep these 10/11 distinct type
To always run
Thru a modulator;

Some more active than others

Resuming on these breakdown of 11 CYCs;

We now have a elected mod from the map-specific CYC

Resuming: - **drops to a park scen 1**

- **panics! (scan&track trio) both recognize everything as novelty**
- **but. CYC was already running. (several)**

Resuming - **them | cyc | cyc | cyc |**

- **becomes | n n n n n |**

- **nodes in the HIPPO ; run independently**
- **if any is triggering (modStream) -> (theme)**
- **fill one |n| with it**
- **fill another one |n| with an existing (theme)**
- **list all the action cues of those**
- **calculate the next action for the | n n n n n | (as one)**
- **that one cyc |, goes back to the modulator (an agent)**
- **decide to decompose , refresh, or pause***
- **pause is a yet-to-introduce fallback / checkback mechanism**
- **that we haven't decided yet on how to play well with BM1**

.

(refer to DEMO 0.02)

(tbd; likely a case of porting BM1 to no longer be a HIPPO module; but instead ; a independent process that checks a string of n |cycs| (not n)

Or a string of n|cyc->|cyc| decomposed

Or any arb time period; (set by Identity)

Then produce a special CYC-PAUSE for thus

- **Welp, we resumed the cat scenario to the part where we suddenly went on a tangent about "moving the pause mechanism to DEMO 0.02"**
- **Resuming "Decide to; decompose; refresh; or pause.**

By modulator

Anyhow

Resuming - them | cyc | cyc | cyc |

- becomes | n n n n n |*

- mods run independently

- storing all their outputs in the CONTAINER

- with chempools regulating their

trigger and their timing

(by virtue of waiting for a combination / excess)

modstream with modlatch becomes sims

n gets filled

- a | n n n n n | becomes back to a |cyc|

- gets decided whats next by the CYC modulator

- **if. For example.**

Cat DROPPED to a HOUSE

- **repeat the process again from step 1;**
- **where CYC-MAP**

goes WOT!? > (MAP?)

Within all these CYCs , every SIM (inside Latches)

Already represent an update

There is also learning episodes in:

LATCH resolution, THEME resolution,

| CYC | resolution, and object parameters that gets updated

In every one of those,
so theres a lot of room for randomness
Before even counting for modSocials / SocialCortex

Or other mechanism yet planned

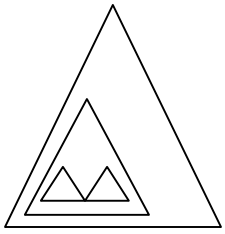
(on further work yet made)

Ontop of that; there's also a later part, where SIMs & SIM resolution are ported onto

Something called Quickcalc (labeled in the pics as a Triangle)

These are FE range %-% that can be nested

(methods akin to multivariate Bayesian linear regression)



Hoping to optimize the sim process

x

BIG NOTE 1

& 2:

On PRM

Problem 1 worries that we aren't sure about the

CYC modulator; (whether they are made of two parts; solver? Or input/output)

CYC modulator; (on currently we presume there would be some agent

That can compare the composition graph of

A sim; would be better than another

(or a policy of a sim)

thus knowing which |cyc| to follow / decompose

this is partially manually doable

thru obvious completion nodes *that didn't happen*

i.e for YOWL

if any of the HAPPY/PECE/BOGUS/YOWL

didn't get homeostaticly resolved

then it would be obvious what the continuation is

Trouble is, we aren't sure, and we (think)

that implementing some formal method of comparison

Is probably (definitely) needed

*prob1; modulator?

*use concept selector? Paper?

Z-score approach by baseline sample?

Problem 2 worries that the protocol of breaking down cyc

Is untested for which is which

And why it has to be so

OR whether these are simply molecular length

(With varying backbone for attachments)

And therefore it has always been variative between individuals

Or health condition or such

We presume it's a decreasing amount of

| n n n n n |

| n n n n |

| n n n |

To currently test.

Because this would imply that;

As the three state of a CYC-chain is nearing completion; or being run

The first node; has a lot of interjection points

Where; the action is

observed for "should we?"

The second node; has less interjection points

Where; the action is

observed for "does this fit?"

The third node; has one (least?) interjection points

Where; the action is

observed for "WELL THAT'S THAT(!?)"

"florp?"

(refreshing confirmation / whole chain cut)

So problem 2 being; is this even smooth upon running?

We figure that just providing a

TREE of 11 CYCs

Some 20 length decomposition for each

Some tree-search table protocol

Some object-types templates for which Sim

Some few theme-solving-table

(all of which is (?)doable thru ROS)

We might be able to come up with an agent that;

Upon a short enough smushing and combining

(the | n n n | n n | n | 3 cyc cycle)

Is done within few dozen miliseconds

Supposedly; if such is possible;

Then its probably gonna be …

Pretty doable*

To make it lifelike

(big shrug)

.

(and by lifelike;

Before adding social cortex & its intricacies

(by lifelike; we probably

Get a gecko.. despite aiming for a cat

~_(._.)_~

(no specista! Vs gecko; simply that

they are … more predictable)

Problem 2… | n n n n n | -> protocol itself

Is not so polished /

perhaps theres a paper (existing)

#something to compare between decomposition nodes

#something to time actions and grade the contents quickly

(some kind of transformer / FE decision calculator

That can interchange some components)

.

That leads to this being practiced well

But we're not quite sure!

So thus we described the limitations. Back to CYC. Then to P1P2 + CYC

Resuming from:

1. How process vision input
2. How input becomes simulation
3. How cyc' come into play

v

Ok.. now we

4. How to combine it all…

- OK so we have CYCs getting broken down
- map-related-CYC becomes mainmod
  (as the highest context)
- mods process vision input; and run their independent process
- mods produce triggers
- modStream has a trigger grader
- also from cyc
- trigger grader grade trigs
- become Theme

Themes. (also) Sims. Fills | n | 's

| n n n n | 's get recombined (in a snap)

| cyc | get judged on what next

(on the tree search)

(sometimes… these are simple things..

(which is why we think its pretty good to suggest to make!)

(these manual chem. ports; and BM1 & BM2 mechanism)

such as completion checks

(check sim, sim done, didn't die, this ok)

| cyc | continues. And so the cat traverses around.

Fulfilling them

… **some bio stuff to silly ponder** ..

- are these brain waves? *(syke analogy)
  - (cycs -> wave
  - (themes -> wave v (less)
  - (sims -> wave vv (less)
  - (vision -> wave vvv (less)

*.. probably the inverse.. the cycs are the deepest / most invisible..

(means brainscans are probably (vision?) + sims painting a big picture over a landscape)

- what about audio?

"(maybe) paint the same sim world; put beacon on self
(or selves) " (as described in beacons.doc)

"(have them pulse) (w/ audio)
"(everything else happens per usual)

onward

List of content

1. abstract
2. components
3. component details
4. examples
5. summary

Abstract

Hi, updating on the original P1. Cat Hippocampus Emulation, we restate our original intent

Of creating a Cat Agent with minimum capability of labeling & combining the labeled objects

With simulation types; that are driven by CYC (epigenetic emulation) drivers

And then onto SIM resolution to then become updates

p

p

Ontop of referring to our Pilot .doc for the whole description of a project

And the sample of how its done

We'd like to display if this component would run on its own

This part deals with the : LABELING & TAGGING of objects (ScanTrio ; TrackTrio)

p

This part deals with the : Taking CYCs and Triggers (from Trios) onto THEMES & LATCHES (sims)

p

This part then : Creates a protocol on putting those THEME and their filling

Once those THEME and VisionNodes independent processes are drawn

We will then draw a CYC-CYC-CYC CYC-CYC-CYC queue bar

p

Which is the AC (action coordinator) (the smushed form is called AC)

Our current method on operating the Agent lies in

Dividing each CYC block onto | n n n n n n | six smaller blocks insisting on updates / force filling from the priority tables (decided by CYC-modulator); and then resolving each block

#p

#p PRM. et

For how the CYC should progress (decompose / stay / other)

.

Finally; between the PROCESS + the CYC + the SMUSHED (AC) outcome

We decide the next.

[combine them] and have the CYC decide (depending on stage) (refer to $$$ - $$ -$ decreasing tempo states), atleast one segment is likely hardcoded

.

We compress (depending on novelty%) onto quickcalc ; using FE% range-range nested atop each other. and make a demo

Later part hope to emulate #p ainf

For result keeping

On the process itself; of deciding which path might be better:

Ontop of confirmation using the hardcoded (and minimally variable) SIM condition

we ponder to gain some help using:

#p

Components

Screens

Chempool

Modules

- discussion

BM

Sim & Latch

AC.

Summary

 vs

Make cat. Traverse scenario

Label things.

Get CYC driver

Solve. Update.

Some theme go upward

Some chems to tally

Screens

Screens are permanent vision transposes

Screens used to be recurring themes & cyc

Always having a distributed reference in the brain

But since they are likely to linger all the time

We simply put them as fixtures

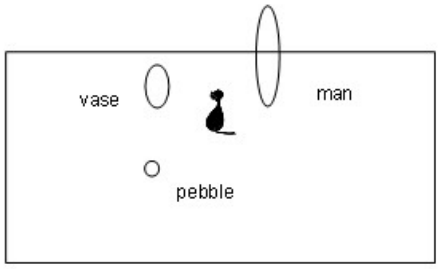
Screens only work this way

Input	Linger	<- Requests -> Output

Notable screen types are:

- cat overhead
- cat shading
- cat currentmap
- cat foodplau

Other



a sample overhead

Linger.

Screens are mostly abbreviations of recurring themes ; they linger and handle requests on the fly (source of inference & beacons)

Chempool

Chempools are a depiction / special node to tally

Accumulated affectors / chemicals / hormones

That emulate how a organ

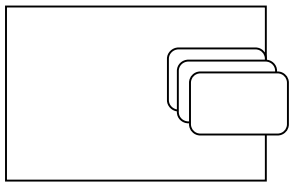
depend on this triggers / combo

we describe module

we assign chempool

some to watch individual chems

some to combine chems



One for each,

One for combines,

Most chems behave with an annotation that gets an increased ^ behind it

To denote how much got combined; (to then change onto) –

For example:

Tallying

##-STRESS for modRoam

##-Satiety for modRoam

increasing

STRESS^ ->

STRESS^^ -> accumulates three steps (1 step each 6 seconds unprocessed)

STRESS^^^ -> (arbitrary hardware grade parameter)

(in reality; the timing of these ; could actually

Refer back to the identity* (thus somewhat variable))

(this is likely done thru *blotting*;

we haven't gone there yet!)

STRESSU – # -> becomes a popped timed affector

-

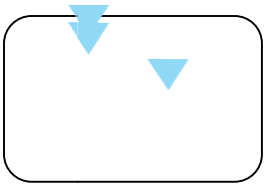
These are arbitrary notations

meant to mimic biological tallying

Depiction

add 1X every FE% unexpected behavior of ObjA.

also;

Pool 

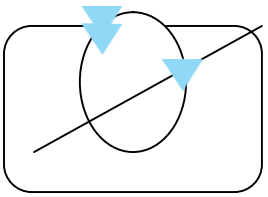
merge ,say, 3x + 2y = 4n capacity at 8n / second.

(ex:)

take clump, merge

merge

swipe



=

these inv.tringles are either tallied / trigger

Some examples

Onto a effect / trigger

- Tallying #focus at visiontrio
- Tallying #pairwise at visiontrio
- Tallying #queued at modStream
- Tallying #queued2 at modLatch
- Tallying #stress at every node

Say. modWatch.

Having chempool for

#QUEUED - amoung of chems , might pass on to BM2* (later part)

And #STRESS - amount of failures , chemical failures of any kind

And #ERROR - amount of uncertainty in the whole scene

#PING - ping^^^

for “why haven’t you properly shade this pair yet?”

forcing the shading process to happen by modLook’s

neglected request

All of which probably exist in most organs & modules; and share-able

(thus implying that whole-body deterioration could alter mental states)

>

Anyhow , that's that.

We are not quite sure yet!

On the whole table

.

Just to recap; the unsure parts of this work (until a whole cat)

LIST OF PENDING COMPONENTS

- A model to select method / approach (might use #p?)
  On cyc modulator

- A table to grade triggers
- A cyc table for initial agent driver
- A cyc decomposition table for the same
- A cyc learning mechanism by taking a clump of "themes" associated to the cycnode / goal
  And then figuring which is which;
  (or if unable to do instantly; then produce #PON chem.; for major thinking theme)

- A chemical table for smooth organ function
- A sim table for modlatch
- A theme / streme table for how many sims per each
- A biological porting for some of these;
- For variance / other mechanism

- BM1 & BM2 is sidelined as the next process
- (involving LIMIT & CHECK)

Modules

List the modules first; the things that we attempt to discuss in this paper

- **(1) Scanning Trio**

Responsible for the making and maintenance of MAPS

*maps are actually contexts (in biology)

These are merged with vision, but perhaps in the actual brain

This is likely to be the endpoint / endstream of vision cortex

Where it merges with hippocampus / or the region that responsible for a theatre

(a screen). And also merge with audio / other sensory input

At the same point (in our case; we could represent those with the

Same SCREEN + objTypeSocial Beacons (with only audio))

Scanning Trio includes:

modScan - scans & make map

modLoc - put points on map

modNav - put routes on points

- **(2) Track Trio**

Responsible for keeping track and associating objects with their labels

Beacons, Themes, Latches; Sims; and goals

Produces, shadings. attention. and tests*

Track Trio includes:

modWatch - gather other affectors , put shades overlay

modLook - gather other affectors, direct attention

modTude - gather beacons, publish tests, publish triggers

modTude is particularly responsible for talking with SocialCortex

sifting thru the beacons; and publishing tests (not yet triggers)

which is a range of FE% (false prediction)

when violated; becomes trigger; or pre-trigger chem.

Other prominent module includes

- **(3) modElect (modRoam / other)**

Responsible for electing the highest brain wave context as the core driver of hippocampus

Most cases this relates to the AREA / MAP of where the agent is at; responding to CYC-MAP

(refer to Example collection 5. at appendix)

modRoam behaves as a tracker of several core goals

Of the highest context, imagine you have a bunch of

THEMES. LATCHES. SIMS

And somehow this magickal crab (the modulator)

Is able to tell you that " hey ; so we are here to attain these "

"these" became the core driver of the module

And they operate independently just like modwatch / others

To ensure that the output a chem. Of

#GOAL1 satiety

#GOAL2 satiety

And their play-by-play rules

For example. modRoam with

FoodPlausibility producing chems of #

RoamSafety #

RoamConfirm #

RoamExcess #

With each a modulator to tally their compounding / the errors within

This will make the cat instantly realize whats wrong

When the vision node is tallied onto a | cyc |

(force filling a | n |)

This works just like modRoam of yonder (the previous version)

They just insist on the proper production (over time)

Of the core goals of roaming

And if not satiated; they behave like other organs

Producing faulty chems that either

Put stress

Put trigger

Put #PON (by downstream)

Or worse!

(wat worse?)

Other prominent module includes

- **(4) modStream modLatch**

modstream and modlatch are endpoints,

as part of the hippocampus (?)

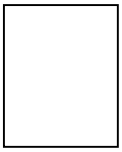
They operate to list a number of chem.triggers.

From all sources*

(or perhaps if later we add modBody for example)

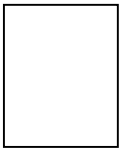
(we simply put it as a trigger publisher too)

We then conjure a trigger table



*dumb low priority …

We then assign them minimal variability



*dumb^^^^ alas..

(slightly more important (!))

. later updates might include a quick simulator or a layer for quickcalc / #pon chem

Modules come with Chempool attached (as with described)

Some of the build to be attempted is written at appendix below

(slightly edited) they contain first intention of engineered

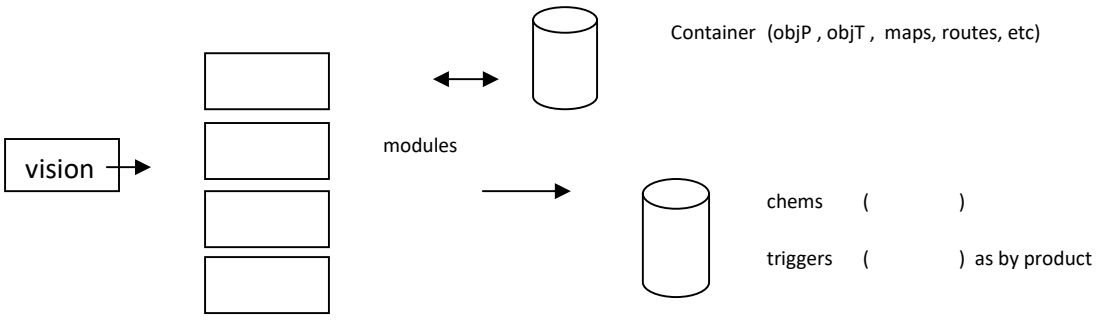
Module nodes

These are all hopefully independent, with Fallback chems

And fallback inputs; all of them producing ERROR chems

That denotes something is seriously wrong (with origin)

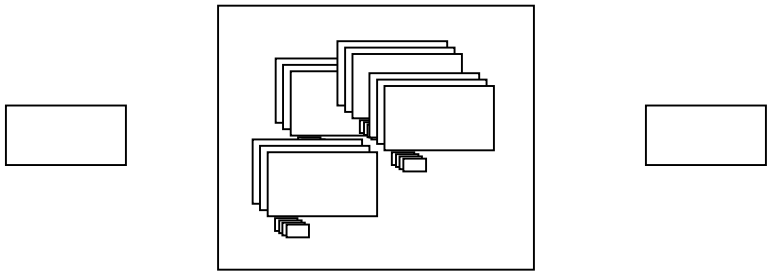
Readily as a goal is composed in real time



works anywy

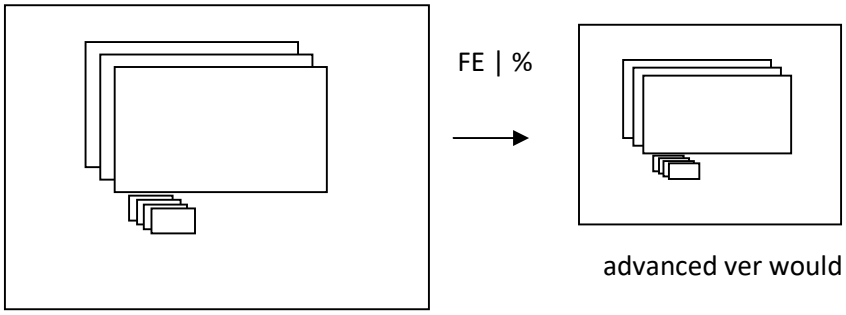
Pics.

Stream:



stash track update

Latch:



advanced ver would see

latch try to minimize FE

stash nest FE# %

See appendix below

For addition of module details

(pic version of trios, bm, n other)

From previous version

(with some edits)

AC

AC is supposed the QUEUE system for an active agent to fulfill or mark fulfillment

But recent attempt (the smushing attempt)

Makes us wonder? Whats the tidiest?

Do we depict queue for CYC's?

Do we depict queue for SIM's? THEME's ?

Do we depict queue for all pending stuff?

(these are pretty much the UI for the processes under works)

Wait leme think..

- *should we just draw them as brain waves !?
- Lets think of this for abit

HUH. problem is BM1 is probably one of the bigger of those

Also #PON ..

(checking and *daydreaming* (or just dreaming))

Both of which we haven't got to yet;

So I guess we just separate the AC to two kinds; currently there

And simply imply that theres gonna be one

For the self-other copying simulation. (PON)

And the default-subconscious simulation? (BM1)

Current proposal for AC

Onion design.

Uhhh.. not yet I think

So we thought about which one is best to depict

(to keep track the big part of the model)

And we thought! Maybe best is to display the part where

The PRM (Process Reward Model ?) (Process Preference Model?)

of choice; that is used to :

(and thus displayed)

AC 1

Theme resolving / sim resolving

Selecting the proper sim

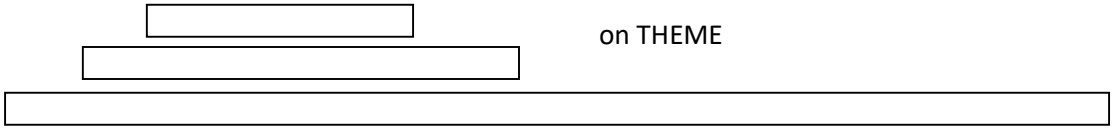
And what sims are chosen and how does it lead to the next

(applying process Decomposition to a annotated chunk*)

.

(then giving several recommendations based on stand-out features)

- (anyhow; method is beside the point; we want to depict something to track process)



on THEME

Chunk to chunk (based on chunk type!) and its steps

i.e. depicting the part where the agent

Thinks about:

- Hey I need to reach this goal
- This goal is “this type”
- “this type” has these “decomposition”
- “these decomposition” has these “chunks” (lots)
- Among these “chunks” there are these “chunk types”
- Among chunk types “a b c” , we diffuse abit for
- Then we test for result
- Then we compare on that testing which part
- Was used most or result in what sort of effects
- Which seems difficult! (true)

But could it be uhhh…

1. That there already exist a method for doing this
2. As long as the SIM / game type is generalized?
3. Thus is a matter of testing
4. How robust the generalization and
5. Decomposition of the chunks?

Anyhow! It be nice to have a bar to depict these

THEME stacking

Using the process above

Which in formalized guide; would mean that

AC1 should depict:

Step1. What type is THEME

Step2. What theme DECOMPOSE TO

Step3. What those are CHUNKS

Step4. What chunks ----

Step5. Would it be testable..

Step6. How to test..

Step7. Test result.. etc (standard PRM! Steps)

To represent these PRM (process reward model) steps; in chunks of

THEME stacking/ THEME decomposition

(to about less than a dozen layers preferably!)

(or just select the better ones!)

Would be really nice! (and hopefully doable)

AC2 should depict:

Step1. SAME THING

Step2. But on CYC scale

Step3. Which should be simpler?

Step4. Because CYC are denoted by known THEMES

Step5. And themes starts hardcoded;

Step6. Before they get associated with a type

Anyhow! Similar-ish PRM – related depiction

And decomposition for the second one;

Only for the CYC modulator this time

Yep. To conlude, We think having these

PRM STEPS

(on what model that we get the chance to use hopefully!)

Would be pretty nice to display as AC

(They also represent which process is currently on the works

And why!)

Ontop of that; some later works include:

(yet to make, for 0.02)

AC For:

1. Put BM1 (a independent display of BM1 , we could just use BM1's display ??)

   (being a staple that (should always) override or publish pause) attached to

   (but we could also tag the SIM's (or related theme)

2. Put #PON (a duplicate AC series with an extra difference; representing a daydream)

   If a daydream lasts too long with a certain long context,

   This could be similar to a baseline PRM depiction,

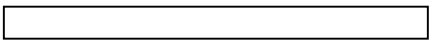
   So maybe not;

*On #2. Welp… come to think of; realizing that by saying these!

Likely means that we haven't got a firm grasp on what to model and display

(Tho it is likely much helpe-able by doing the other parts!) (untested prm.. welp)

Summary



Hi! We'd like help! In testing the PRM methods or plugging them

From recent papers

Hi! Cats can be cool :D

Also we hope that the current design is doable

(somewhat)

Using available ROS-related tools and framework

It would require long hours

On handcrafting the table and the CYCs

(and testing them too!)

Designing CYCs.. *speculative* Epigenetic Basis

Much like Dr Levin's start-to-end bioelectricity innate patterns*?

(?)

But! surely its worth it!

Think of it like we're trying to

Compose the early days of

protein sequencing (but much less difficult) (extra romance cuz cat)

._.)

Yupp! thanks

X

Pasted content at below

(includes BM 1 & BM 2 depiction)

References

-

- Same as paper 1
- Plus recent paper
- (moved to end part)

-

- notable new ones
- on approaches of PRM / LSTM /
- on morphospaces / goal-directness cycles
- on youtube content
- regarding consciousness
- epigenes and many
- neuro podcasts such as
- Cognitive Rev / MLST

-

-

plans to incorporate bm

DEMO 0.01

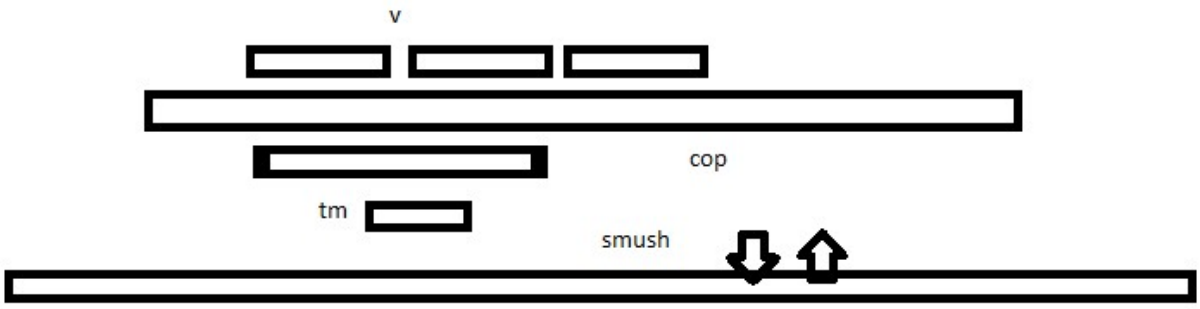
DEMO 0.02 proposal

"meanwhile; after we update from the smushing version, we could consider

adding a checking block

To include [checking] & [limits]

we imagine might look this: (BM 1)



\+ ☐ check block , policy as an | n |

Take this part;

add a Block that Checks a [variative Length]

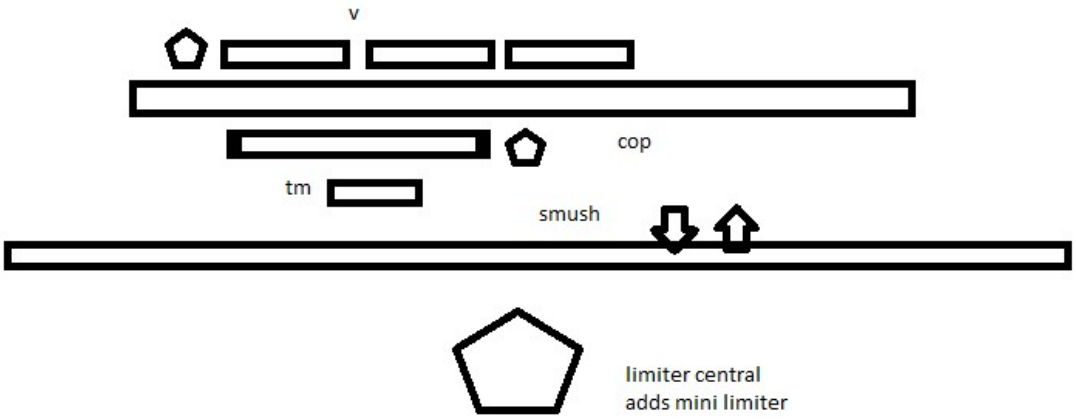
Of the Smushed block (BM1)

Then add back a Block; to always account for that checking

To the 6 node / breath mode

(details below) p 55

Version 3; adds a Limiter / % modulator to any input (BM2)



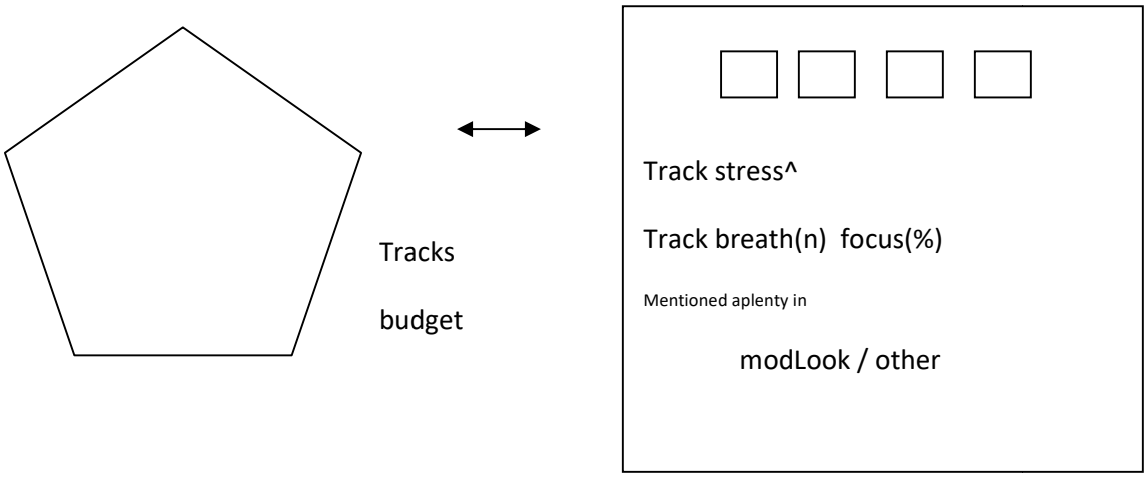
a planned mechanism

to combine & publish budgets

check p59

On BM2, we add a node to gauge & track different processes

We list, collect, and assign



BM 1 BM 2

.

BarsMod1

BarsMod2

BM1 is a checking version attached independently to the Hippocampus

BM2 is a limiting version attached independently to hippocampus

-

BM1 makes sure that a person knows if something didn't happen as it should

(instantly; without observing again)

BM2 makes sure a person know the overall limit of his budget and doesn't go wonk

(instantly; without attached goal)

-

BM1 is needed for quickly realizing a pre-trigger pre-simulation analysis

BM2 is needed for quickly keeping a budget of threshold

(also communicates with other parts)

-

Drawing them:

Depiction

.

.

BM1 takes a independent time-checking period

(not theme dependent; theme or correction is applied afterwards)

(should also set parameters for bodily tempo;)

For example

(100 – 1000 minutes)	-	a regenerative long term	(affected by concentration etc)
(10 seconds)	-	a plan long term	(affected by attached plan information)
(1 second)	-	a execution check	(affected by focus level)

Then we draw each with some variance;

And fill them; (quickfill with quickcalc's version of SIMS)

. It is also likely that we can process the quickcalc triangles here.. but not sure

Then we draw the bars :

BM1 taking a quickly rendered (either this or that would be likely)*

(not a single outcome; several outcomes diffused with empty nodes)

(diffuse =with empty nodes as in; the chemical doesn't take exact inputs)

(simply allow for ranges; (the initial reconstruction itself is not exact))

(adding the obligatory diffusion would allow this process to be reliable)

(i.e. by always assuming for "this range" of "this leeway" and

"that range of "that leeway", then counting FE for each,

deciding afterwards what direction

did the instant; went wrong)

Compare them with

modstream's known composition and

modlatch's current best guess on what the FE (?)

Seem highly speculative; and yet to be worked on;

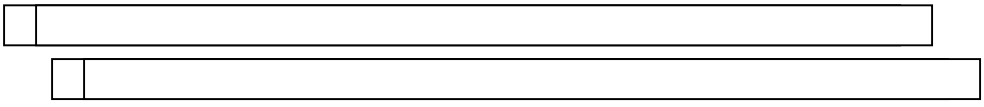
((that's true))

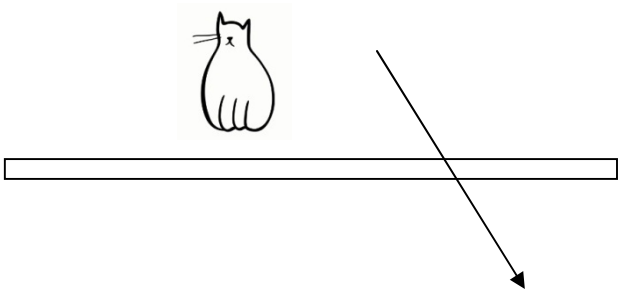
(shrug)

for each Bar; you both fill:

" Whats supposed to happen (learned)(copied)(composed (modstrem)

" What happens (FE% calc separate from latch)





(shallow) bm1

Low affordance

at walk… (!?) *checks back

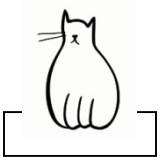
rat zips

modwatch & bm1 separate trigg

*trigg

*trigg

rest area



“should #pon?”

(many resource bm1)

on another time tho

cat chillax.. many small triggers

(reviewing what happened)

(reliving 2 / 3 minutes letssay)

(producing trigg for #pon)

Triggers are not

directly accepted tho

Cat would get busy as per usual

and any free period it could devote extra resources to this diffusing

And the output becomes clear ; (with directionality)

(subconscious realization becomes clearer as we relax)

This can perhaps (?) double as the #PON basis

Or such composer

meanwhile for BM2

We.

Pondering different type of limits

We could do:

- Introduce by each module
- Infer from bodily function , then fit in

- Register all those Budgeting
- And Chems
- Make a special tracker for them
- BM2.

Say for example

> chems stacking- track on the side for total on whole

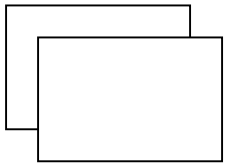
> bodily hardlimit - total progressive alarm to system failure

> embedded limits - depicting theme size; perhaps talk to identity
(in future) (identity = policy keeping)

> cycle limit - could also log each cycs for a modified limit
since each cyc would have its own modulator
maybe this is for some experimental
bodily affector, like, emotional chemicals of doubt
or such like; that are seperable from

Step1)

Make a top controller and a copy



(nodes, talk on the fly)

(regarding copies; copies are used for emulating biological process like;

Copying entire AC's / identities / ;

.

In this case; the just means we have another node ready

For processing any duplicable process

(suppose the agent wants to #PONDER and copy itself on a simulation;)

(then to quickly have a space to emulate "ouch that was painful")

(since in the body; these copies are also made out of chemical*

(actual chemical actually existing), thus, this "keeping a spare node"

and "when spare node gets cleaned; residue pops out"

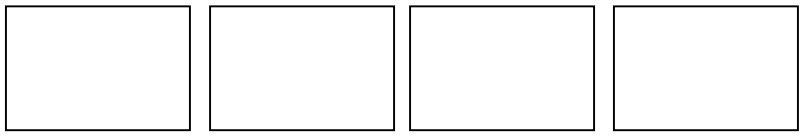
likely makes sense

Step2)

Track each

For whichever processes we want to budget

Chems table



One type simply accumulates

PANIC^ -> PANIC^^ -> PANIC^3 -> et

One type has open ports to sustain;

Before producing ;

GENERALPATIENCE < < < < <)

-> produce PATIENCEBREAK^

Why not just Patiencebreak^ ^^ ^^ ^^^ ?

.

Haha this is just crappy engineering

~

Simply put a wishy-washy temporary component

Until we have decided whether

It should be a module or not

Hey if it gotta work; it gotta work

(since the whole point is to track)

We end up with many screens tracking many chems & their compounds!

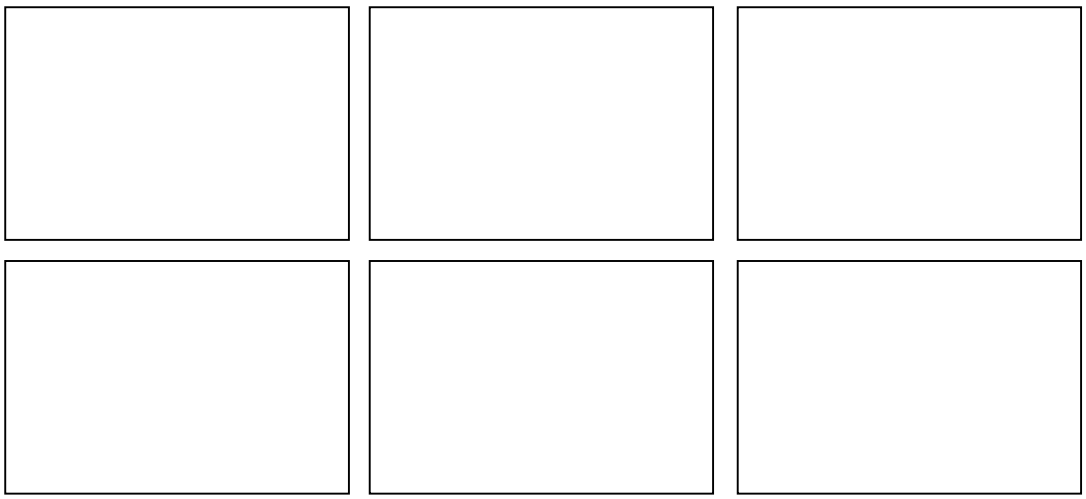
Easily introduce a 

stopper of sort

yes! This is a tidy way that the body probably also do

especially when it comes to “blotting”

and bio ported 3d space for the components



One for every Token? One for every LimitBucket? (as above) + make rule

BM Summary

BM 1

& BM 2

Probably contributes alot

To lifelikeness

But its kindof too complicated

Just to suggest all the CYC-modulator (and the tree search)

Before even combining them with sims

Besides that; it is possible that the clustering are

Done slightly different; pending

Social Cortex & other porting

Moving to

Modules

Updated

24 / 01

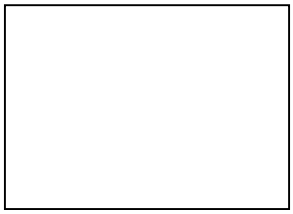
TrackTrio

modWatch - keeps track of objects ; produces shadings;

- denotes for lack of data ; denotes for sufficient data; denotes for ongoing

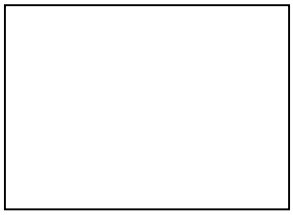
modWatch is a busy node

many screens:



one main -the one the agent uses for decisions

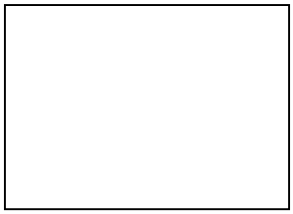
[for the current builds; we only care about this; and the shadings]



xx imaginary - we have a stashing system

Where noise/free inference based subconscious

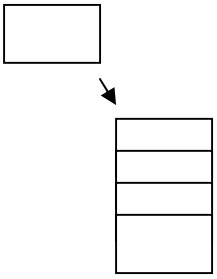
can be explored, ready to make spontaneous guesswork



xx alt - the way it works is; modwatch talks to many modules;

those modules provide quickly disappearing alternatives, constantly regenerated; but most times; one or two become a ; 2-3 seconds lasting predictive overlay over a vision; in essence; we start with everything having 50-80% dosed FE uncertainty (if we want to venture onto there); tiered upwards once; and we keep things with PendingPairs / PendingFeatures; ready for highlight; these, along with the main; are rotated as the current pure focus for modLook / modTude to infer. (biologically; this becomes the source of realizing that "OH im actually looking at THIS HUGE PATTERN; then that pendingPair becomes the real thing to source Tude (object feature assigning) and Look (directional) onto)

This is best illustrated / tethered with a Theme (if we happen to make a separate AC simply for feature identification). (otherwise; just omit the entire thing and make a makeshift that is always the main)

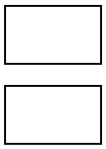


What is the above writing about?

1. Listen to other modules
2. Have many screens / has to be ..
3. Between 3-6 lessay
4. For staple PAIR CONFIRMATION
5. And for staple FEATURE DETECTION
6. And for staple OBJECT PARAMETER CONFIRMATION

(staple processes to attach quality to objects; as with prior to beacons)

For these; imagine the 3-6 screen



Gets filled with shadings, that is mostly composed of Quickcalc

(a abbreviated SIM that gets quickly solved; to check for whether this predicted quality or feature is within FE % - %)

Empty?

These will be most of modWatch's main duty;

These screens are SHARED with the simulations from THEMES

Therefore; their amount; their free amount

And their; un-utilized amount

Becomes the baseline for mechanisms such as modStream FALLBACK

Or other daydreaming / BM1/ recall by focus capacities

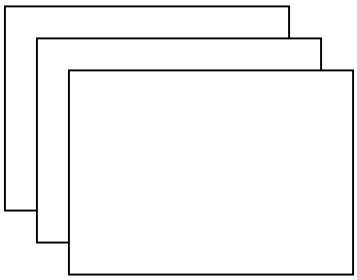
7. (FOR EXAMPLE, un bio ported)
8. THEMEs copy their SIMULATION
9. Onto these Screens
10. They pop in n out

Still does what this does;

SHADE for Nested FE confirmation

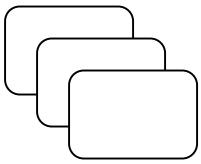
(shade until threshold)

Then; a separate quick converter for:



many shadings ; they are always shaded for other modules

each of this screen has a chempool counterpart; with one chempool to aggregate for the main module itself! (why? Because each of these need timing; and the main mod also does; for backlogs)



one chempool for each ; one for the module it self

currently; only processing token inflow / backlog of process

suchlike SHADEFOR^ SHADEFOR^^^^

suchlike WTFISTHIS!?^ WTFISTHIS^^^^? (after many backlogs)

when it gets for ^^ - ^^^^ its very likely to have its own big latch

these CHEMPOOLS that connect to these Shadings

accumulating violated FE range

aggregate the violations onto a trigger

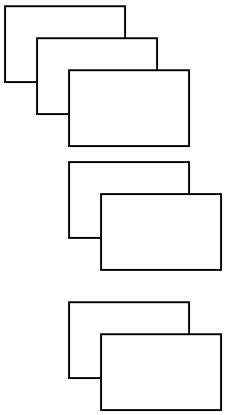
modLOOK -

governs attention & eyes try its best to let this signal through

- due to how modlook refers a lot to movement coordinators ; habit;
- id intervention; and other nodes of process; we currently put this aside

and use existing modules / codeframework for robotics-head-direction

but in all likelihood; modlook is gonna have



small potential screens – for prediction of what happens when its “looked at”

small coordinative screens – for talking to body; n goal

small ongoing TRY’s - robotic protocol for increasingly valid pushes (headdir)

Why are these called SCREENS? (instead of nodes)

Decomposing these snap subconscious decision onto

themes & sims would be too much, also,

in all likelihood; there exist an organ in the brain

That serves as a empty node (that hungers for filling)

So its more appropriate to design it this way?

(it’s a empty node; but it behaves like screen; it just copies and transpose slightly)

(to then talk back on the fly)

.

Same mechanism as stashing for modWatch; tiered FE resolution with the highest

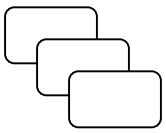
Becoming the main; and the second highest(s) become chained with a Condition / quickclue

::

- in essence; these are blips of consideration that LINGERS
- and filled with quick resolution latches (already at formulaic)

if any small timing component; it should be one for consideration; one for blip<->latch

and one for the main module just to log overall stress

3 chempools 

blip/latch-worthy? Consideration^

blip/latch-worthy? Consideration^^

Overall stress logger

.anyway. SYKE! Lets just admit modlook is wip~ (for p4)

modTude

modTude takes all the data; and assign a label

(especially BEACONS)

For an assigned BEACON origin; or BEACON type

(Two or far more can definitely be attached to a person; given a signal behavior)

modTude then assign a range of FE%-FE% of allowable

violating the FE range would cause a trigger to be formed

to be queued , compounded (a chem.)

or just kinda stay there to

burden the chem. system

modtude is busy trying to confirm calls from modWatch

or have its own supportive identification of WatchScreens

all it does;

is identify everything; possibly pair everything;

possibly featurize everything;

modTude is divided onto 3 major categories:

(they trigger by available stamina)

ONE ON INTRO	-	new things; log and process
ONE ON SUPPORT	-	always copy; always reprocess
		(redundancy)
ONE ON FOCUS	-	always ready for a Latch
		(being connected to latch means that it also connects to LatchFallback)
		(thus doesn't need its own fallback)

FOCUS gets the most budget; despite not always used;

.

INTRO gets secondmost budget; actually a latch; but formulized per typical / prior to upgrade to latch

.

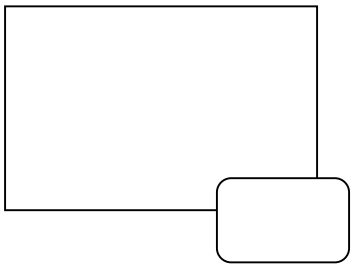
SUPPORT is ongoing; sometimes painted by ID (identity) and habits.

(subconscious preference)

Make a screen to tally

Overall density

Make a chempool to go along with it



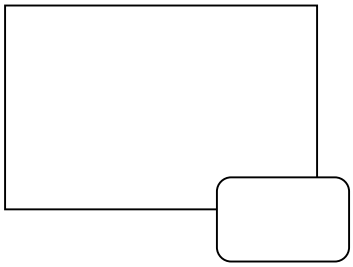
Make a screen for

SUPPORT – subconscious expectation

(stashed) (we probably cant simulate this yet)

(too much bandwith / hardware costs)

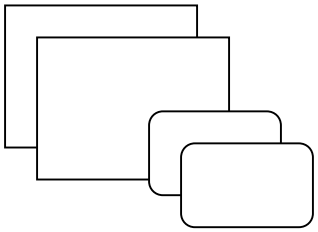
A chempool too whynot (not using anywy)



Make TWO screens

For each OBJECT in INTRO & FOCUS

(two for every in both)



quickresolve latches

unless it doesn't resolve

Then its ongoing MBLR

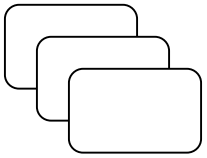
Which makes it a latch + tude;

And processed as both

(tracked by tude; filled by latch)

Make a overall chempool for

Overall logger for each



support & focus & intro chems

(agent would say “im busy understanding what im looking at, due to ##)

Overall for main (stress related)

Best way to intuit this is; modTude / object feature recognition is always supposed to have its own AC; and overlapping THEMES that help them tier up their recognition; but we omit all this for current agent.

Frankly.. ._.) (just a hunch) biologically; if you see a Brainscan that goes like:

[Clusters of electronic Activity] x 3 in a brain letssay. Well; those ElectricClock are all supposed to be AC’s with tiered Themes. But we just omit this for now.

Now we move on to MAPPING TRIO

Guh, for Scanning trio, we aren’t sure the best policy

Of separation / for best output

Whether modscan should do the tagging or

Whether modscan just makes a map out of patches

And modLoc would trace where the agent is at

And perhaps also trace the goals

Anyhow, we aren’t sure, there are completely different version

of these trio .. lets just gist it for now

modScan

modscan takes vision input

modscan makes patches (From known pattern & duplicate patterns)

modscan makes map from that, map could just be a data-format/ molecule size

that is now big enough to afford being a context for

upper level chemical process

current proposal

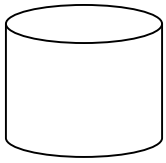
arbitrary example on how things go

⇨	⇨	⇨	
Make patches.	Make map.	Take Request.	Make more Map.
##pattern	map-TAGtoconcepts	x	scale bigger?
##pattern	map-TAGtoconcepts	x	scale smaller?
##duplicates	smaller map? wall	modwatch	scale concept?
##recognized	smallermap indeed	modwatch	search patt?
Epitemplate	smallermapConcept	modtude	attach patt?
##pattern	scale up on force	x	x
##pattern	becomes as big as	sims	scale as big?
	Focus afford.	X	enroutemap?
	Actual map.	x	
	Actual context.		

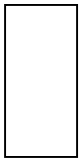
etc

Take Scannables take request Make Tags

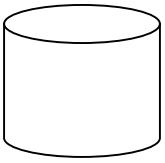
Take Tasks transform Make Output (Map?)



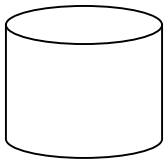
Scannables



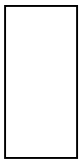
tasklist



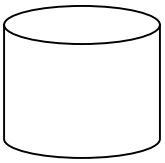
tags



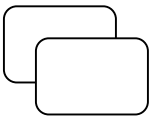
Tags



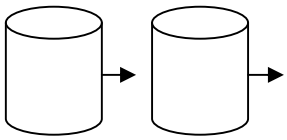
task



maps



Chempool to document fulfillment, and table to signify what happens if not



Weights / requests
For modLook / lowlevel themes
To recognize / query
Certain input sources

biologically

it works like this: tags imply

a Bucket of Tags needing filling

a Bucket of Sensory requests … a bucket of output

modLOC

modmap takes the colorlogged snippets of surfaces and clues; and convert them onto a map

GOTTA HAVE MAP

GOTTA MAKE SURE EVERYTHING HAS CONTEXT

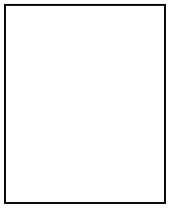
Raw concepts

concepts + map

In reality; perhaps there is some part of this that is simultaneously used by

modLatch or some sort of sim-related uses? NOT SURE!

we are currently just concerned about making triggers that fit



rawconcepts

conc+map

things that help

what things help? For example, perhaps this mod necessitates the forming of

CatOverhead

(a staple screen)

Who knows which organ actually produces or contains (!)

Which actual staple (in terms of biology)

But we posit that;

We have a staple structure that needs to attach

Maps to Concepts

Sometimes; the concept demands the map

(this makes sense; because

Remember that it came from CYC

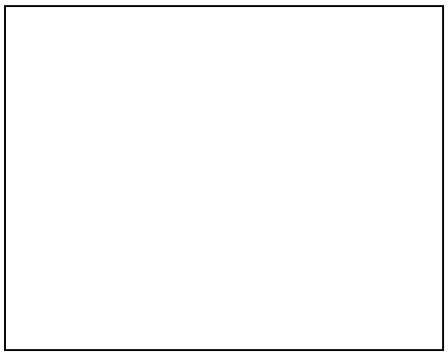
And CYC breakdown; before it goes to these structures

CYCs are triggers therefore they come from concepts first

Welp! Not sure how that impacts in detail tho

modMap deals with converting a CONCEPT to a CONCEPT + MAP

and therefore, a goal is always attached* (being a concept n all)



make map. have goal.

Put self?* or otherpoints

modNAV

modNav firstly collects the necessary links

- Maps
- Goals
- Objects of threat
- Objects of uncertainty
- Objects of note (themewise)

modNav colors the map with objectives and the relevant parts

to an actionplan / sequence data that relates to such map

it talks with other mods; (this is usually modStream ;; page XX)

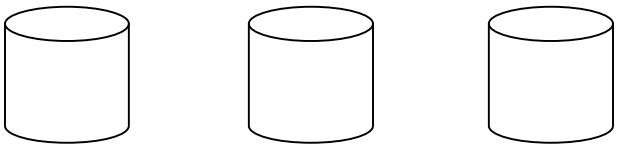
Each one of them is attached to a maptype

And when a few of them are located together in a row;

We cluster them together and make a Big Screen (a new Map)

Make big screen; pepper the things inside of it; with smaller map fragments

(as context)



Collect combine offer & prioritize in pools

(the pools optimize by priority themselves)

(this part is likely to be optimized in future)

overall, modNAV deals with producing

ROUTES

(and shadings too?)

(not sure, maybe yes)

(both modLoc & modNav shades)

(? or modnav puts the route

(? and modloc shades the route

(not sure! Looking for best practices

Before (posit) (hoping)

confirming biologically

Mayhaps, later mechanisms involving algorithm learning

Or some sort of simulation update

Would update the objP inside the MAPS

Used for the sim, therefore,

Eventually becoming a sort of policy data for Route?

Thus, maybe modNav

deals primarily with adapting this

(prev POLICY) + (new MAP*?) = new Route

Or maybe even straight up conjoined purpose with

Policy Gradient / maybe this is PRM? Who knows.

Epilogue

Endnotes

Hur hur! paper attempts to depict some conjoin-able

That eventually function to produce simulations

From inputs (The relevant ones!)

But! all too many missing elements

(tho already hypothesized for purpose)

MUCH THANKS AND >:D

CIAO

at

Page 77!

References#2

-

- YT Sources (mlst / cognitivrev / etc)

- HOPING TO GET PRM HELP

.

- Kunihiko Fukushima Neocognitron (1979) - First CNN Architecture with Convolutional and Downsampling Layers The foundational CNN architecture introducing convolution + downsampling. https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf

Alex Waibel Phoneme Recognition Using Time-Delay Neural Networks Method using time-delay neural networks for speech/phoneme recognition. https://isl.anthropomatik.kit.edu/downloads/Pheome_Recognition_Using_Time-Delay_Neural_Networks_SP87-100_6.pdf

Jürgen Schmidhuber Linnainmaa (1970) and the First Publication of Modern Backpropagation Discusses Seppo Linnainmaa's contribution to backpropagation. https://www.idsia.ch/~juergen/who-invented-backpropagation.html

W. Zhang Proceedings of Annual Conference of the Japan Society of Applied Physics Sept. 1988 Conference reference from September 1988 in Japan. https://drive.google.com/file/d/1nN_5odSG_QVae54EsQN_qSz-0ZsX6wA0/view

Sepp Hochreiter Long Short-Term Memory (LSTM) Neural Network Architecture (1997) Seminal paper introducing the LSTM architecture (Hochreiter & Schmidhuber). https://www.bioinf.jku.at/publications/older/2604.pdf

Jürgen Schmidhuber Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks 1991 paper proposing an alternative to dynamic recurrent networks, today called "unnormalized linear Transformer." https://people.idsia.ch/~juergen/FKI-147-91ocr.pdf

Ashish Vaswani Attention Is All You Need Seminal 2017 paper introducing the modern "quadratic" Transformer architecture, which scales quadratically. https://arxiv.org/abs/1706.03762

- On biology / et

Spread across many YT sources ; iai . MLevin'sacademicchannel

ainf. Channel / et

From paper1

1. LeCun, Y., Kavukcuoglu, K. & Farabet, C. Convolutional networks and applications in vision. in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* 253–256 (IEEE, 2010).
2. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
3. Lai, K.-T., Lin, C.-C., Kang, C.-Y., Liao, M.-E. & Chen, M.-S. VIVID: Virtual Environment for Visual Deep Learning. in *Proceedings of the 26th ACM international conference on Multimedia* 1356–1359 (Association for Computing Machinery, New York, NY, USA, 2018).
4. Hamrick, J. B. Analogues of mental simulation and imagination in deep learning. *Current Opinion in Behavioral Sciences* **29**, 8–16 (2019).
5. Li, W., Wang, L., Li, W., Agustsson, E. & Van Gool, L. WebVision Database: Visual Learning and Understanding from Web Data. *arXiv [cs.CV]* (2017).
6. Baars, B. J., Geld, N. & Kozma, R. Global Workspace Theory (GWT) and Prefrontal Cortex: Recent Developments. *Front. Psychol.* **12**, 749868 (2021).
7. VanRullen, R. & Kanai, R. Deep learning and the Global Workspace Theory. *Trends Neurosci.* (2021) doi:10.1016/j.tins.2021.04.005.
8. Creators Nelson, Benjamin1 Show affiliations 1. Active Inference Institute. *Exploring the Development and Integration of Cognitive Mechanisms in Search of a Unified Cognitive Computing Framework*. doi:10.5281/zenodo.13125068.
9. Worden, R. The Requirement for Cognition, in an Equation. (2024).
10. Creators Nelson, Benjamin1 Show affiliations 1. Active Inference Institute. *Prioritization, Iteration, and Convergence in Cognitive Systems _ Bayesian Inference, Perceptual Gating, and the Requirement Equation*. doi:10.5281/zenodo.13363380.

11. Bronstein, M. M., Bruna, J., Cohen, T. & Veličković, P. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv [cs.LG]* (2021).

12. Hasani, R. *et al.* Closed-form continuous-time neural networks. *Nature Machine Intelligence* **4**, 992–1003 (2022).

13. Lerner, I. V., Yurkevich, I. V., Stepanenko, A. S. & Constantinou, C. C. Loss Fluctuations and Temporal Correlations in Network Queues. in (IEEE, 8 2008). doi:10.4108/ICST.WIOPT2008.3195.

14. Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M. & Su, Y. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *arXiv [cs.CL]* (2024).

15. Hoffman, D. D., Prakash, C. & Prentner, R. Fusions of Consciousness. *Entropy* **25**, (2023).

16. Kannengiesser, U. & Gero, J. S. Design thinking, fast and slow: A framework for Kahneman's dual-system theory in design. *Design Science* **5**, e10 (2019).

17. Krämer, W. Kahneman, D. (2011): Thinking, Fast and Slow. *Statist. Papers* **55**, 915–915 (2014).

18. Ward, T. & Garety, P. A. Fast and slow thinking in distressing delusions: A review of the literature and implications for targeted therapy. *Schizophr. Res.* **203**, 80–87 (2019).

19. Kahneman, D. *Thinking, Fast and Slow*. (Macmillan, 2011).

20. Chen, D. *et al.* Deep Reasoning Networks: Thinking Fast and Slow. *arXiv [cs.LG]* (2019).

21. Safron, A., Sheikhbahaee, Z., Hay, N., Orchard, J. & Hoey, J. Value Cores for Inner and Outer Alignment: Simulating Personality Formation via Iterated Policy Selection and Preference Learning with Self-World Modeling Active Inference Agents. (2022) doi:10.31234/osf.io/k4cas.

22. Heyes, C. Précis of Cognitive Gadgets: The Cultural Evolution of Thinking. *Behav. Brain Sci.* **42**, (2019).

23. Bolis, D. & Schilbach, L. 'Through others we become ourselves': The dialectics of predictive coding and active inference. (2019) doi:10.31234/osf.io/6uwyn.

24. Barbulescu, R., Mestre, G., Oliveira, A. L. & Silveira, L. M. Learning the dynamics of realistic models of C. elegans nervous system with recurrent neural networks. *Sci. Rep.* **13**, 467 (2023).

25. SIMA Team *et al.* Scaling Instructable Agents Across Many Simulated Worlds. *arXiv [cs.RO]* (2024).

26. Friedman, D. A. & Smékal, J. Generative Research Teams: Active Inference Compositions For Research and Meta-Science. (2023) doi:10.5281/zenodo.8164667.

27. Karl J Friston, Thomas Parr, Conor Heins, Axel Constant, Daniel Friedman, Takuya Isomura, Chris Fields, Tim Verbelen, Maxwell Ramstead, John Clippinger, Christopher D Frith. Federated inference and belief sharing. *Neuroscience & Biobehavioral Reviews* (2023) doi:10.1016/j.neubiorev.2023.105500.

28. Creators Wilkinson, Thomas M. 1 Cordes, RJ2, 3, 4 David, Scott5, 3 Friedman, Daniel Ari2, 3, 6 Show affiliations 1. Office of Health Security, U.S. Department of Homeland Security 2. COGSEC 3. Active Inference Institute 4. Pivot for Humanity 5. Information Risk and Synthetic Intelligence Research Initiative (IRSIRI), University of Washington, Applied Physics Laboratory 6. University of California, Davis, Department of Entomology and Nematology. *The Properties, Processes, and Perspectives Inter-Framework (P3IF): Multiplexing Interdisciplinary Requirements Frameworks to Manage Information Risk and Foster Cognitive Security*. doi:10.5281/zenodo.10034512.

29. Creators Cordes, R. J. 1, 2, 3 David, Scott2, 4, 5 Friedman, Daniel1, 2 Mikhailova, Alexandra6 Penland, Andrew7 Young, Sam Zacharias, Colten1 Show affiliations 1. COGSEC 2. Active Inference Institute 3. Pivot for Humanity 4. University of Washington-Applied Physics Laboratory 5. Information Risk and Synthetic Intelligence Research Initiative (IRSIRI) 6. University of California, Davis (UCD), Center for Neuroscience 7. Eight Arms Nine Brains. *ATLAS: A Question Oriented Approach to the Use of Pattern Languages in Knowledge Management*. (2023). doi:10.5281/zenodo.10296602.

30. Bengio, E., Jain, M., Korablyov, M., Precup, D. & Bengio, Y. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. *arXiv [cs.LG]* (2021).

31. Parr, T. & Friston, K. J. Generalised free energy and active inference. *Biol. Cybern.* **113**, 495–513 (2019).

32. Pezzulo, G., Parr, T. & Friston, K. Active inference as a theory of sentient behavior. *Biol. Psychol.* **186**, 108741 (2024).

33. Creators Nelson, Benjamin1 Show affiliations 1. Active Inference Institute. *Comparative Analysis of Active Inference in Hebbian Networks and Cognitive Computing Frameworks*. doi:10.5281/zenodo.12562484.

34. Smékal, J. & Friedman, D. A. Generalized Notation Notation for Active Inference Models. (2023) doi:10.5281/zenodo.7803328.

35. Noble, R. & Noble, D. Physiology restores purpose to evolutionary biology. *Biol. J. Linn. Soc. Lond.* **139**, 357–369 (2023).

36. Friston, K. *et al.* A Variational Synthesis of Evolutionary and Developmental Dynamics. *Entropy* **25**, (2023).

37. Hamilton, J. P. Epigenetics: principles and practice. *Dig. Dis.* **29**, 130–135 (2011).

38. Jablonka, E. & Lamb, M. J. Evolution in Four Dimensions: Genetic, Epigenetic, Behavioral, and Symbolic Variation in the History of Life. *Life and mind.* **462**, (2005).