ARCHITECTURE THEORY WORKPLAN LOGIC ALL GOOGLE DOCS

- ❖ **MAKE FURTHER SPECIFICATION/ TODO LISTS FOR THESE**
- ❖ **(and indicate importance)**
  - ➢ Organization Docs
    - Project-list (with completion levels)
    - Links to SWISH Interface to Try
  - ➢ Establish Bina48 Todo
  - ➢ Establish LOGICMOO Todo
  - ➢ Establish AGI Todo

  - ➢ Make PDDL planner system into pluggable library
  - ➢ Make a web-ui based PACK installer/browser
  - ➢ Convert the Logicmoo/PrologMUD system to CPACK(S)!
    - the only reason for delay is that *parts* may still yet separate
  - ➢ Web based process controller (Perhaps ClioPatria cpack)
    - **(NEEDS AMBITION) Due to it actually needing to decide what a process will be**
  - ➢ YAP port of SWICLI **(TODO - cleanup (minor bugs?) and package)**)
  - ➢ Work on SWICLI docs and mono issues 700$ **(TODO - cleanup (minor bugs?) and package)**)
  - ➢ Write install docs **(NEEDS REVERSE ENGINEERING)**
  - ➢ Sync with FLP **(NEEDS)**
  - ➢ CIAO-Assertions system for PlDoc and analysis **(NEEDS AMBITION)**
  - ➢ Release of slack_prolog **(NEEDS AMBITION)**
  - ➢ multivar's C extension to SWI-Prolog
    - The C code and Prolog API **(TODO - cleanup (minor bugs?) and package)**)
      - Decide just how pervasive the change will be (like re-#define deRef?)
    - Building multivar's SWI-Prolog on windows **(TODO - cleanup (minor bugs?) and package)**)
    - ….
  - ➢ 3) 8-40 hours - use my multivars to really make the lisp code work (**25% done+**).. **(NEEDS AMBITION)**
  - ➢ 4) force myself to (or not) to update to the new JPL **(done)**
  - ➢ Set up TRAVIS-CI **(NEEDS)**
  - ➢ Get Lenat permission on KB 7133 **(Decide to skip?)**
  - ➢ do a release of prologmud since three people have asked me to update **(NEEDS)**
  - ➢ 2) release of slack_prolog **(NEEDS done as PoC for 'dictoo')**

  - ➢ Setup Appdapter to use these visualizers:
    - SWOOP
    - OpenCyclograph https://youtu.be/zr9uT1y5aj0?t=16
    - (Unrelated Make OpenCyclograph work with OpenCyc 4.0)
    - Repast?
  - ➢ Upgrade these projects JPL to JPL7

  - ➢ Improve WAM-CL
    - Get DAYDREAMER running HAS-BOUNTY $2000
    - Get Knowledge-Machine running HAS-BOUNTY $2000
    - Pass 85% of CL-ANSI HAS-BOUNTY $2000
    - Find contributors

    - ■ Online IDE Tracks: For multiple people trying/editing LOGICMOO

      - ➢ **SWISH/VSCODE/THELIA-IDE**
        - Reuse and Pool SWI-Prolog instances **HAS-BOUNTY $300** (Lang:C/Prolog) 6 hours
        - Shared MUD Project Config **(Needs VSCODE-REmote)**
          - As SWISH Shared MUD Notebook ?
          - Fork/Create New MUD
        - Live editing Prolog Files **(NOT DONE)**
        - Telnet console **(NOT DONE)** (RESEARCH IF HTML DISPLAY IS DONE)

- PDDL Files (**NOT DONE**)
- CLIF Files (reuse VSC-PDDL plugin code) **HAS-BOUNTY $1000** (Lang:JScript/TypeScript)
- TogtherJS Like sharing based on File (**DONE**)
- Continuous demo integration over Telnet (LATER)
- SWISH that has browse file system **HAS-BOUNTY $300**
- Live editing **SHARED REAL** Prolog Files **HAS-BOUNTY $200**
- Web browser of "Prolog Source State" **HAS-BOUNTY $200**
  - Maybe improve xlisting_web to allow batch editing of Prolog HAS-BOUNTY $500
  - Improve xlisting_web to allow full https://en.wikipedia.org/wiki/Knowledge_Engineering_Environment HAS-BOUNTY $1000

➢ **logicmoo_base**

- CLIF/KIF to PFC-Prolog library (This is **the main thing logicmoo_base** is supposed to supply)
- IKL Test Suite / CLIF Test Suite

- 5) working on logicmoo_base to make it better at inference and proof generation (2 months)
  5a) PUZ and ANA
-
- 6) working on E2C (english to CycL) but counts on 5 being better.. but makes the Nomic properties of PrologMUD more accessible (3 months Works for me version but needs #2)
-
-
- 7) Integration of a planner to work with the logical state of assertions to direct agents in the PrologMUD (3 months)
-
- 8) Curate our CycL/KIF/PDDL content someplace

➢ PFC - Prolog Forward chaining **(done)**

- make Justifications cache smaller!
- PFC is basically a term expansion system that allows one to manage undoing terms expansions as things change.. also at any point when you are looking at the expanded term.. you can get a "why it is expanded that way?".. these structures are large and some expansions can be combinatorial (but they loop terminate at least) **(done)**
- thinks("i like whipped cream"). ⇒ thinks_words([i,like,whipped,cream]) ⇒ knows_about_text([whipped,cream]) ⇒ knows_about_object('tWhippingCream').
- If i unassert knows_about_object('tWhippingCream'). the original thinks("i like whipped cream"). goes away.
-

❖ LOGICMOO_UTILS

- with_loop_check/2 **(done)**
- with_thread_local(Temp,Goal) - temporary assertions while running goal **(done)**
- with_predicate_IO/3 **(done)**
- rtrace/1 must/1 sanity/1 **(done)**
- setup_call_cleanup_each(EachSetup,Goal,EachCleanup) **(done)**
- NONDET calling of prolog engines (perhaps use new engine interface) **(TODO)**
- idiomatic clause_expansion/3s **90% done (TODO - cleanup (minor bugs?) and package)**
- no_repeats(Vars,Goal) - **(done)**
- filematch/2 (also this needs performance improvement) **(done - but needs new eyes)**
- Listing vars **80% done (FIX bitrot)**
- xlisting/1 **(done)**
- Hookable Clausedb **60% done** (overrides existing retract[all]/asserta/clause/erase) **(TODO - cleanup (minor bugs?) and package)**)
- use above to make attvars usable from the above Hookable Extensions
- Hilog Reader **(FIX bitrot)**
- must/1, must_det/1, rtrace special semi-interactive debugging **(done)**

➢ UABCL/LARKC

- Swipl starts SWICLI (Prolog)

- SWICLI loads IKVM (C#) (Skipped changing IKVM to allow overload of finals)
- Scans classpath and load Java classes (Java Cyc)
- SWICLI loads UABCL (Lisp)
- UABCL calls cyc.lisp
- Cyc.lisp start BeanShell Scripting and other things
- Load ASDF packaging and SWANK/SLIME

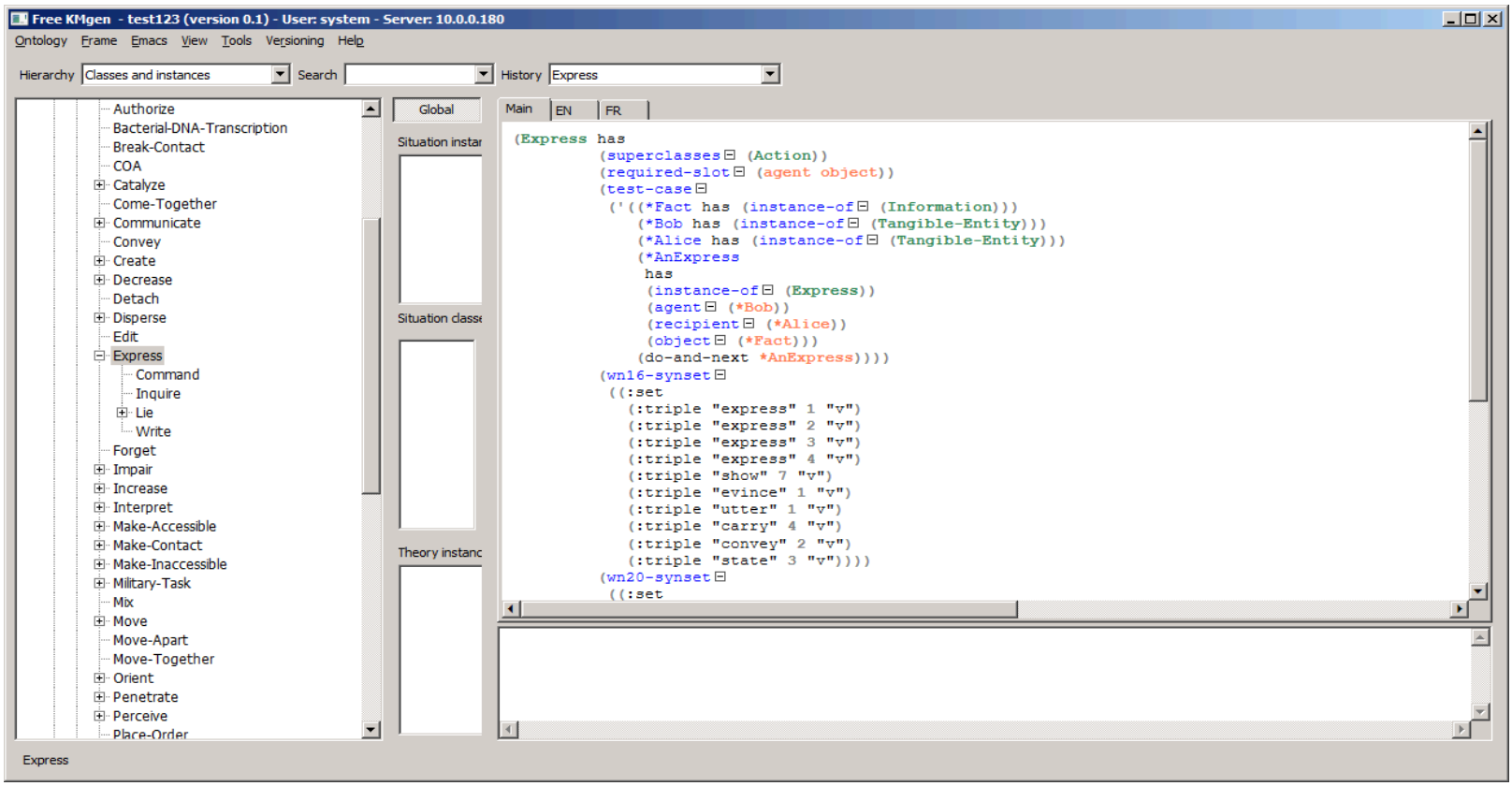
❖ Ensure SWIPL loads at the same time - STARTED
❖
❖ Register Prolog predicates into Cyc KB -
❖ Make .KE file (Mark Some preds and some Mts for being CalledFromProlog or CalledFromCycL)
❖
❖ Whenever CYC has an assertion (ist someMt (loves A ?B)) Prolog will have user:ist(someMT,loves('A',B)).
❖ Whenever prolog has assertion user:foo(bar) cyc will have (ist prologUserMt (foo bar)) effectively present
❖ .
❖ Ensure that CommonLisp Package system is not separated from CYC's (in-package) = BAD IDEA
❖ Ensure FLet temporarily shadows SubLSymbol.this.getFunction() - DONE
❖ SubLEnv + LispEnv should push/pop simultaneously - DELAYED

2) developer setup/install plus continuous integration (8 days)

3) working on PrologMUD to make it more playable (1 month)

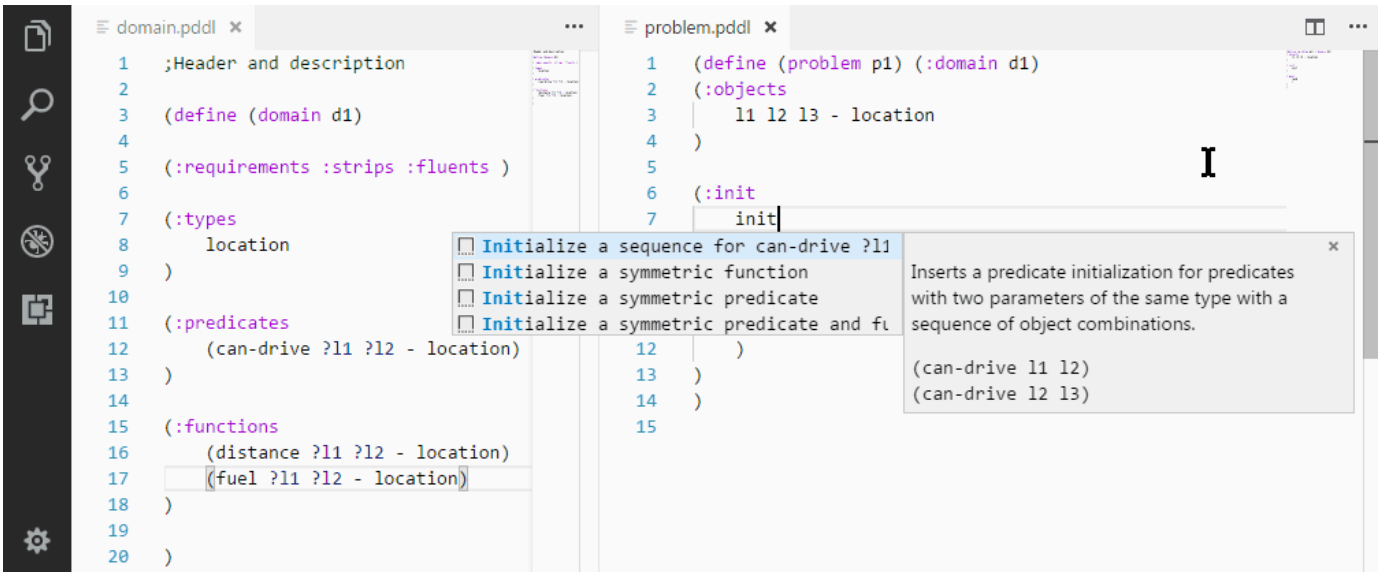
4) Prolog CYC-like Proof browser/editor (1 month)

This is a content browser like



That would let you find a term .. then focus on it.. and see and edit content that has to do with the term

While editing, it might be nice to see what is going on.. good example. is..



Live link = https://raw.githubusercontent.com/wiki/jan-dolejsi/vscode-pddl/img/PDDL_symmetric_init.gif

Objectives: Allow

TODO Week of 4-24-2017 (it's been a year today .. and nothingings gotten done!)

Making test questions to ensure theorem prover is working in https://github.com/TeamSPoon/logicmoo_base/tree/master/t/examples/fol

Questions: Vaadin too much a rabbit hole? (too much Jetty?) AppdaptorGUI? https://www.youtube.com/watch?v=RIW8eCZ1T7o
(Too much Scala with Generics wank?)
Write it again in java? (Too much time?
Start BeanShell

it runs sdesktop();
BeanBowlGUI.addObject(this);
BeanBowlGUI.addClass(SubMain.class);

Start startEmbeded() - Loads the JRTL fromSubMain.startEmbeded()
Start startDaydreamer() - Loads the DAYDREAMER
Start SWI-Prolog

:cd ../cl-bench
(load "do-interpret-script.lisp")

:cd ../cl-bench
(load "do-compiled-script.lisp")
)

Attempt to see if ResearchCYC is viable for prototyping impl of LM489 / PrologMUD

FOR RELEASE

- Fix movement code that seems to be broken - FIXED
- Stop generating apathFn(s) when unneeded - FIXED
- Cardinal directions are avoided still - DONE?? **(NEEDS) (50% done)**
- .

- Ensure Hungarian PrologMUD cases are used in CYC - IN-PROGRESS
- Only call fully_expand when nesc - VERIFYING
- allow isa(I,C) ⇒ antecedents - CURRENTLY
- .
- Get the thing to load without violating logicmoo_debug==true rules CURRENTLY
- Figure out why infSupertypeName is being called more than once - FIXED/DONE
- Allow getting past break in debugger - DONE
-
- transform isa(-,+) triggers to +(-).
- one shot trigger =*=>
-

Medium goals:

Translate Bina48 transcripts to DRS
All prolog user declared code/data to be persisted into both CYC and Prolog
All CYC data to be dynamically allocated in CYC KB for browsing and editing
DAYDREAMER operates on CYC-KB instead of builtin GATE
DAYDREAMER synchronizes "CX" states to Prolog

Create a StandardObject<->Prolog SYNC
(StandandObject includes ASSERTION / CYC-CONSTANT)

FOR BOTS

- Check on STRIPS and decide if should use HyHTN instead
-
- Profile startup sequence
-
- baseKB:mpred_eval_lhs(pt(tHominid(iExplorer2),rhs([tAgent(iExplorer2)])),((tHominid(_G1202)==>tAgent(_G1202)),mfl(baseKB,'/opt/PrologMUD/pack/prologmud/prolog/prologmud/vworld/world_2d.pl',260)))
- .
- ?- correctType0(cuz,[_],ftList(ftVar),O). % needs to succeed
- ?- correctType0(cuz,[1],ftList(ftVar),O). % fails
-
-
- Fix macro/term-expansion to not need loops checks
-
- Fix the fact the assertion check is actually doing a query instead of assertion check (unfixed causes a loop)
-
- Port new sub lisp features to Common_Lisp_Extesniosn (1 day out of 3 complete)
-
- add e2c.ke improvements to 4Q
-
-
- ensure new strings are represented like.. ['"this"','"is"','"a"','"string"'] '"the"'
- using convert_to_1cycString/2
-
- build TinyKB and use the "renames.pl" file for the next stages
-
- rebuilding kb7166.qlf (**qcompile kb_7166)**
-
- generate renames.lisp (and run in kb)
-
- keep compile time below X00secs CPU time
- - system_base 30secs->5secs again
- finish e2c restringification

- Port LarKC engine to the new CYC release (Adds ANSI Common Lisp language)
- Extend CYC's Natural Language Understanding for E2C
- Pred PrologMUD
-

1. Decide about use of Cycorp`s CYC in PrologMUD
2. Do I extend subl.jar? (8/10)
3. Do I load my common.lisp? (28/30)

4. But prolog and E2C itself does not need to use CYC at all..
5. But it can be written totally in my extended version of CYC above
6. If not use CYC at all then I may be able to distribute it
7.
8. A) PrologMUD in Prolog? (177 / 180)
9. B) PrologMUD in Lisp/CYC? (0 / 40)
10.
11. E2C in Prolog? (30/60)
12. E2C in Lisp/CYC? (30/60)
13. E2C in Lisp/Prolog Hybrid? (option 1 + 30/60)
14.
15. LM489 uses Prolog and optionally PrologMUD (3/180)
16. LM489 uses CYC and optionally PrologMUD (3/90)

```
% accessible from any microtheory the REAL impl (and from prolog from anywhere)
:- kb_dynamic(telnet_session:player_address/2).    % DEPRECATED
% accessible based on the rules of the microtheory (and from prolog but needs to look up defaultContext)
:- kb_dynamic(player_favfood/2).

% accessible from any microtheory the REAL impl  (and from prolog from anywhere)
:- decl_shared(telnet_session:player_address/2).
% accessible based on the rules of the microtheory (and from prolog but needs to look up defaultContext)
:- decl_shared(player_favfood/2).   % DEPRECATED
```

-
- cycQueryCheap(Sent,Mt)
- confirm mpred_prop/3 is now standard in code
- do not safe_wrap/1 library code anymore
-

THIS WEEKS TODO LIST

- decide between MPREDS: lots-of-modules-autoloaded, lots-of-modules-noautoloading, one-module-mpred, one-module-user
-
- Export.. deductions.pl
- Export.. assertions.pl
- Export.. renames.pl
- write the "renamer synchronizer"
- basic rudimentary query
- write PFC based destringifier rules
- dump destringifications
- write PFC based DCG Creator
- test suite for small set of parsables
- term index
- cycQuery(Sent,Mt)
- strings "hi" stored in '"hi'
- strings "hi there" ['"hi','"there'].
- C2E ??
-
- clarify the ***when*** certain term expansions and libraries are needed

- define the difference between kb_dynamic/1 and decl_shared/1 (or merge them)
-
- stop calling fully_expand so much! 80%
-
- call mpred_expansion only during clause compilation 75%
-
- write up the structural ontology 10%
- Fix predicate property ontology!
- .
- Finish this YEARs list below

Allow Base Ontology to dictate disjointedNess of Collections…

	Types	Instances	TypeTypes
Actions			
Agents			PersonTypeByActivity
Artifacts			
Capabilities/Affordances			

Actions
Agents (and PersonTypeByActivity)
Artifacts vs PhysicalPartOfObjects
Capability
Events
Scripts
FormulaTemplates
Groups
LinguisticObjects (words but not strings)
Microtheory (Kripke frames, Belief sets)
Place
Quantity (Includes Colors)
Relation (includes Collections)
ScalarIntervals
Situation (includes Goals)
ExpressionType
TimeParameter
Topic

This MONTH'S TODO List

- Export Previous content of Past LM489 project
- Re-Read that content and convert to DRS
- How far apart is DRS from XPs?
- convert "wallace format"to DRS
- Is "wallace format" format still usable? I mean how many XPs are deductible from it?
- Load/Saved LM "worlds" (containing XPs)

❖ Follow through on formulating research plan 8cD

This YEAR'S TODO List

- …
- Create a list of doc that describe how logicmoo is used (cyc.com ieee.org/ ISO/IEC 24707 / http://www.ifsowa.com/talks/clintro.pdf etc
- ….

- Learn KQML Communication

- Metaepistemology
- Cathoristic Logic
-

EXTRA TODO LIST ITEMS

- Supplement isa/genls system with MOTEL's elementOf/assert_ind 99%
- Try to disable the pre-existing and confirm there are nto missing results ???
- genlPreds/t subrole/fact
- Ensure MOTEL is not regressed
-
-
- ensure levels of bootup of prologmud
    - ensure_loaded(library(logicmoo_utils)).
    - patching SWI's remembrance of current $term_of_file/$term/$term_of_reader
    - ensure_loaded(library(logicmoo_base)).
    - ensure_loaded(library(logicmoo_user)).
    - ensure_loaded((logicmoo_repl)).

1. ensure works while unsafe_speedups=true (run's sanity)
2. ?- coerce(I,vtDirection,O). % must take less than 3 seconds
3. Don't regress the MUD startup! [x]
4. http://logicmoo.org:3020/hmud/ <- make work
5.
6. Head expansion has a loop in it (todo)
7. Figure out if singleValueInArg/2 should be based on assert_sv or some other trigger!
8. prolog_repl needs to **not** hide output stream data! (mostly fixed i think)
9. Get "help" command to work (fixed - but double spaced)
10. When moving ensure mudAtLoc/2 updates (broken)
11. Fix bumping into walls
12. EXTRA text(localityOfObject(apathFn(iLivingRoom7,vDown),iLivingRoom7)).
    a. Don't create non existent paths (Fixed?)
13. fix singleValueInArg/2 (fixed!)
14. Fix downward movement
15. Edit Theory Page

https://docs.google.com/document/d/1jQTXPfTi3OfkxJKSdbaskk0BZlQb5nYFlO-hJqdkvzA/edit

Logicmoo implements a **paraconsistent openworld defeasible modal temporal epistemological deontic logic.** A logical system that attempts to deal with contradictions in a discriminating way. That is concerned with studying and developing paraconsistent (or "inconsistency-tolerant") systems of logic. It deems *fewer* propositional inferences valid. Classical logic may be considered a smaller subset of Logicmoo. The point is that classical logic must propositionally validate everything. Logicmoo does not, so in some sense, then, Logicmoo is more conservative or cautious than classical logic. It is due to such conservativeness that paraconsistent languages can be more *expressive* than their classical counterpart.

How To Prepare a release:

mkdir /opt/RELEASE/
git clone https://github.com/TeamSPoon/PrologMUD/
git pull origin development
git push

Rules of what is "already true" can violate "what can be true" (you find out you have to divorce your sister, since you just moved away from utah)

the secret to what i am trying to do is.. the planner has a set of rules called "IDR" about what is legal and not dictates what the Internal Dialog (ID) is allowed to say to itself.. However The action planner has a set of operations that need to pump thru the ID which will probably break them.. Sp whenever the IDRs are broken it plays that game we've all seen called "Guess what animal i am thinking of?" So the ID legal Rules is the "animal guess tree" when the tree hits the bottom instead of asking "Ok, what question should i have asked to know it wasn't a panda bear?" .. its says "what internal dialog rule should be altered to allow me to think of that action?" IDR55 "What could i have inserted (new dialog) to make IDR55 happy?" .. then it inserts that new phrase into the planner Rule .. initial ID tree (which i had not described well) is a tree that it can fall back on to decide what to do when the planner has no ideas to pump thru

[23:40] <dmiles> thus the ID must always be generating even when the environment give it nothing interesting to do
[23:41] <dmiles> this is a contrast from writing this system completely as a planning system.. which i felt of course inclined to do
[23:42] <dmiles> this doesn't stop me from trying to not care about the ID .. as it might still without the ID "meatgrinder" which makes doing everything a PITA
[23:44] <dmiles> but having it there and retrainable (Guess what animal i should be thinking of (and losing the game constantly)) to meet the conditions of planning based impl imposes .. i think it will conformed each step of the way
[23:46] <dmiles> it will get forced into either a set of conforming rules, or the tree will show it how to repair plans in a new way
[23:47] <dmiles> having the ID as a secondary environmental factor is there since ...
[23:48] <dmiles> Consciousness erupts from the in-ability to separate our skin-in-the-game from that semi-auditory process of hearing ourselves think
[23:48] <dmiles> We struggle to have that mind’s ear “hear” phrases that are connected in such a way that the least amount of cognitive discord takes place.
[23:49] <dmiles> Intelligence comes from our innate ability and requirement to create narrative sounding language around mental events.
[23:50] <dmiles> Personally, I am afraid of the consequences of my internal voice (ID) ceasing. I think that would mean I ceased to exist!.. Or the horror if that voice ever said something *I* disagreed with. Somehow I commandeered (and personified) that voice and made it "me".
[23:51] <dmiles> so my planner is first priority begins number 1 preserving my narrative style i have for myself
[23:52] <dmiles> well the planner takes into account avoiding breaking those IDRs when it possibly can
[23:52] <dmiles> (that is how it preserves that voice)
[23:54] <dmiles> i am hoping the generation of ID (without the planner) at times will force the planner into new actions

So here is an example:

[23:56] <dmiles> that is if my internal voice says is only comfy saying "a b c d e" yet the tree wants me to say internally "a b d e" .. i need to find a plan that will all "c" to take place
[23:56] <dmiles> then try to start the plan at after saying "a b" .. however the plan is not that good it decided to say "c q"
[23:57] <dmiles> However won't like hearing "a b c q d e"
[23:57] <dmiles> So instead hack the plan to lie to me saying "c d e q" instead
[23:57] <dmiles> the plan still works but now gives me what i wish to hear!
[23:59] <dmiles> IDR now is hacked as well to allow "q" to follow the "a b c d e"

Name	set_prolog_flag/2	Meaning	System Call	Usercall	side-effects
speed	unsafe_speedups=true	speed of the runtime code	clause_b	cheaply_u	
safety	unsafe_speedups=true	run-time error checking	sanity	sanity	
tolerance	logicmoo_debug=true	run-time error correction	fix_mp/1, find_and_call	call_u/lookup_u	
debug	logicmoo_debug=true	rtracable	rtrace notrace		
startime	pfc_booted=false	startup and incremental compiling			

/*
cheaply_u(G):- quickly(quietly(Goal)).

*/

speed
safety
tolerant

debug
compiletime

5) Make quicksave files that allow incremental debugging (mostly broken)

6) lazy/1 when used in isa/2 doesn't have to alway "happen" (next version needs tested)

(solved?) unary type preds must FWC as we should no longer BWC

Simplify isa/genls "Typesystem" currently the type system is implemented using pretty old code (BWC) which seems to be a current slowdown and preventing Sane performance. (slowdown was unrelated))

Predicates to simplify (and retain) are: assert_isa/2 isa_asserted/2

Preds to remove: assert_hasInstance/2

REMEMBER: In general, intuitionists allow the use of the law of excluded middle when it is confined to discourse over finite collections (sets), but not when it is used in discourse over infinite sets (e.g. the natural numbers). Thus intuitionists absolutely disallow the blanket assertion: "For all propositions *P* concerning infinite sets *D*: *P* or ~*P*" (Kleene 1952:48)."

~~Fix Web Interface bug that it gets "stuck"~~

1) Ensure modules are asserting information into the correct MTs
2) Pass ALL (easy?) tests
   Run with: script README.MD "*01.p*"

Delayed until later:

1) Begin converting over to specialized handers: like:
   clause_expansion/2/4, body_expansion/2/4 sub_body_expansion/2/4 call_expansion/2/4 and sub_call_expansion
2) Confirm internal modules are registering themselves to assert to baseKB (done)
3)
4) Ensure rule macro predicates are being used checked just before assert/query time
5) ~~Document what module imports need to look like (dmiles keeps changing on himself and having a hard time keeping up)~~

   a) Prolog's "import_module/2" resolve "code inheritance" (not content inheritance like genlMt/2s)
   b) **genlMt/2s wont work while import_module/2** are used
   c) baseKB (and other TBoxes) imports only: system (isa: mtCycL)
   d) Aboxes such as myMt inherit only: system
   e) user module may **NEVER** import mtCycL **(AND never the other way around!) As it ends up enforcing useless restrictions on inheritance**
   f) Code modules may not inherit from mtCycL/1s

```
?- abolish(scce0/0),assert(scce0),retract(scce0),
   setup_call_cleanup_each(
     asserta(scce0,REF),  % r1
     (between(1,3,X),w(a(REF:X))),    %o2
```

```
    (erase(REF),w(leaving(X)))),  % r3, o4
   \+ scce0,  % r5
   w(b(dead(REF): X)),fail. % o6,  r7
```

- r1 = S is freshly called each time
- o2 = S and G share variables
- r3 = S and C share variables
- o4 = G and C share variables
- r5 = Side effect of C is true
- o6 = X shared outside of scce/3 and still bound when leaving
- r7 = Optionally, REF shared outside of scce/3

(r= required, o = optional)

A KB has proven that a proposition P is true if and only if:

1. P is true, and
2. KB can assert that P is true, (that no evidence contradicts the claim) and
3. KB has support justified in asserting that P is true

JTB-Negation Impl Type means that for the the predicate to be true.. It must fail to be justifiably false as well as

Negation by Failure Comparison between PROLOG and LogicMOO

1. The PROLOG language only allows positive literals in the states, while LogicMoo can support both positive and negative literals. For example, a valid sentence in PROLOG could be Rich ^ Beautiful. The same sentence could be expressed in LogicMoo as ¬Poor ∧ ¬Ugly
2. In PROLOG the unmentioned literals are false. This is called the closed-world assumption. In LogicMoo the unmentioned literals are unknown. This is known as the Open World Assumption.
3. In PROLOG we only can find ground literals in goals. For instance, Rich ∧ Beautiful. In LogicMoo we can find quantified variables in goals. For example, ∃x At (P1, x) ∧ At(P2, x) is the goal of having P1 and P2 in the same place in the example of the blocks
4. In PROLOG the goals are conjunctions, e.g., (Rich ∧ Beautiful). In LogicMoo, goals may involve conjunctions and disjunctions (Rich ∧ (Beautiful ∨ Smart)).
5. In PROLOG the effects are conjunctions, but in LogicMoo conditional effects are allowed: when P:E means E is an effect only if P is satisfied
6. The PROLOG language does not support equality. In LogicMoo , the equality predicate (x = y) is built in.
7. PROLOG does not have support for types, while in LogicMoo it is supported (for example, the variable p : Person).

Jeff Orkin https://www.youtube.com/watch?v=iNUEfB6yc34

IF AI: https://www.youtube.com/watch?v=oyGwBfaEPeY

Finin
http://vs1.textlab.io/store/data/000984593.pdf?key=7f3426783dda21612db89f38bf27f0e6&r=1&fn=984593.pdf&t=1469244529154&p=86400

http://www.unipi.gr/faculty/mvirvou/Jonesetal-X000.pdf
http://metadata-standards.org/24707/index.html
https://developer.logicblox.com/wp-content/uploads/2013/10/constrainthandling.pdf

18:19aindilis http://www.inductive-programming.org/ip-systems.html
18:20aindilis http://www.grammarlearning.org/implementations
18:20aindilis https://github.com/aindilis/iaec-notes

Wide Theory Resolution http://www.cs.toronto.edu/~sheila/2542/w06/readings/christian_slides.pdf

Asher:
https://www.dropbox.com/s/9n2910vbcdikuac/Term.png?dl=0+and+https%3A%2F%2Fwww.dropbox.com%2Fs%2F6ihf8v4l235zdyh%2FSign.png%3Fdl%3D0

https://www.dropbox.com/home?select=Turing+Diagrams+-+Systems+of+Calculus+Based+on+Ordinal+Logics.pdf

E. Short:

https://c.ymcdn.com/sites/www.igda.org/resource/collection/250243EB-2D61-473C-A0F2-BAEEFED1CCCD/IGDA_Webinar_-_InteractiveFiction-AIOverview.pdf?hhSearchTerms=%22emily+and+short%22

Contrasting Logics: https://www.youtube.com/watch?v=sQ-M4zJ7574

Game Languages: https://www.youtube.com/watch?v=TH9VCN6UkyQ&list=PLmV5I2fxaiCKfxMBrNsU1kgKJXD3PkyxO

http://www.swi-prolog.org/pack/list?p=logicmoo_base

Language	# Syntaxes	# Base Buitins	Ontology Size	Examples	Where Used/Software		More Info
GDL		role(*a*) means that *a* is a role in the game. base(*p*) means that *p* is a base proposition in the game. input(*r*,*a*) means that *a* is an action for role *r*. init(*p*) means that the proposition *p* is true in the initial state. true(*p*) means that the proposition *p* is true in the current state. does(*r*,*a*) means that player *r* performs action *a* in the current state. next(*p*) means that the proposition *p* is true in the next state. legal(*r*,*a*) means it is legal for role *r* to play action *a* in the current state. goal(*r*,*n*) means that player the current state has utility *n* for player *r*.		https://github.com/SoarGroup/Domains-General-Game-Playing/blob/master/kifs/mummymaze-modified.kif	GGP		

		terminal means that the current state is a terminal state.					
SUMO SUO-KIF					SigmaKee		
MELD KIF					OntologyWorks, SNARK		
CGIF	XML/S-Exprs				ACE/Attempto		
CLIF					ACE		
OWL / RDF					Protege'		
CycL	KE Text, CycML	712 (X007)			CYC/OpenCYC/ResearchCYC		http://www.cyc.com/documentation/ontologists-handbook/tools-knowledge-editing-and-ontology-development/implicit-cyc-functionality/

1) GDL - http://logic.stanford.edu/classes/cs227/2013/readings/gdl_spec.pdf
9 - bultins role/start-game/
0 - Ontology

2) SUMO/SUO-KIF - http://www.adampease.org/OP/
5 - Sentential/Syntactic Operators Builtins exists/all/implies/equiv
500 - Ontology

2) KIF - http://logic.stanford.edu/kif/dpans.html#6.2
? - Glue Builtins from MELD - defmethod
0 - Ontology

4) CycL - https://en.wikipedia.org/wiki/CycL
- Sentential/Syntactic Operators Built-Ins thereExists/forAll/implies/equiv
8000 - base assertions

5) Common Logic
CGIF - http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_X007%28E%29.zip
CLIF -

6) LogicMOO
? - 10 Ontology builtins: implies/forAll/thereExists/+ rewriteOf + holds/N

Vocabulary:

DRA_META
In pl-vmi.c, specially marked predicates get a wrapper that implements something written on prolog

```
ffooo:-writeln('foo').
barrr(G):- writen('Calling':G),G.


?-  '$set_pattr'(ffooo, pred, (dra_meta)=barrr),ffooo.
Calling:ffooo
foo
```

HT_BLOB

Finish OV_CALL in VM
Finish HT_BLOBs
Write attvar tests that accepts the challenges of HT_BLOBs
Build a TRIEs system leveraging HT_BLOBs
Make DRA use those TRIEs
Make DRA work using

http://stackoverflow.com/questions/9064492/limits-to-expressibility-in-cyc-or-similar-knowledge-base-projects

What I needed is software that I can put multiple programs/algorithms into and see how they do against some shared problems.
When I get the results, I can modify techniques. CYC was that program I used for FOL. Gym for NNs? NARS one day a hybrid of these two programs?

LogicMOO

LogicMOO is a development environment for constructing complex knowledge-based systems. It provides a flexible multi-paradigm toolkit that enables developers to combine rule-based, object-oriented, logical, functional and database programming using accepted software standards: OPS, Prolog, and Common Logic.

Features

- OPS-compatible forward chainer
- Prolog-compatible backward chainer
- CycL-Like knowledge representation using Rule Macro Predicates
- Complete online hypertext documentation
- Full access to MPRED_PFC IDE
- Multi-platform delivery

Learn more about the features and benefits of LogicMOO:

- Rule types:
- Matching:
- PFC Backward chaining:
- Predicate impl types:
- Microtherory's interactions with Predicates

Programming Environment

- Logic listener
- Rule-context browser
- Object browser
- Class browser
- Forward chaining history
- Backward chaining subgoal trees
- Enhanced system builder

Objects

LogicMOO uses Prolog for its knowledge representation. Logtalk provides an exceptionally flexible environment for object-oriented programming.
LogicMOO classes are conventional CLOS classes with the addition of mixin class to enable rule-based pattern matching. Therefore, LogicMOO objects can be used by other programs, and it is simple to port existing CLOS code to LogicMOO.

Object-Oriented Knowledge Representation (MOO!)

https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=krol%20prolog
http://eprints.utm.my/32714/1/PhuachiuKiang1992_KOMBASEaKnowledgeRepresentationSystem.pdf

Features:

- Multiple inheritance
- Before and after methods (daemons)
- Multi-methods
- User-definable method combination
- Instances that can change class

- Metaobject protocol for customizing the object system

Rules

LogicMOO provides both forward and backward inferencing through a uniform rule syntax. Each inference system can call the other, and both support pattern matching, object modification and procedure calls.

LogicMOO rules perform pattern-matching directly over the object base. Forward chaining rules use this pattern-matching to perform actions, while backward chaining rules use it to deduce goals.

Contexts and Conflict Resolution

Forward chaining rules may be grouped into contexts to increase the modularity of the rule-base. Contexts acts as the organizational unit for agenda-based invocation, conflict resolution, and meta-interpretation.
Conflict resolution strategies are each composed of a number of tactics. A range of built-in tactics are provided, but users can also define their own.
User defined domain-specific tactics simplify rule-based programming enormously by supporting a clear separation of the declarative and procedural knowledge.

Metarule Protocol

Complete control over the forward chaining cycle is available through the Metarule Protocol (MRP). The user can override the default forwarding chaining cycle for any context by providing an associated backward chaining meta-interpreter. In this way, facilities such as rule explanation or uncertainty management can be customized by the user.

Forward Inferencing

Features:

- OPS5-like syntax
- Modularity and control through rule contexts
- User-definable conflict resolution with built-in OPS5 strategies
- Meta Rule protocol for customizing rule execution
- Logical dependencies for truth maintenance
- Multiple independent inferencing states

Backward Inferencing

Features:

- Prolog-compatible
- Lisp or Edinburgh syntax
- 4-Port debugging
- Mode declarations
- User-selectable clause indexing
- Graphical debugging/tracing

Implementation

The LogicMOO chainers are an extended Prolog based on the Warren Abstract Machine. High performance is achieved by compiling each Prolog clause into a function and handling Prolog control flow by continuation passing.

On X006-03-01, binnetcorp@gmail.com <binnetcorp@gmail.com> wrote: > An elegant way to handle this is to have an answer iterator as part of > the language. Findall/3 and its controlled versions can then be derived > naturally (see http://www.binnetcorp.com/kprolog/fluents.html, also a > paper in CLX000). > > BinProlog and Jinni Prolog (see http://www.binnetcorp.com) implement > such iterators as part of the "multiple Prolog engines" API. > > The most efficient way to provide such an iterator (an easy addition to > any Prolog!) is a weak variant of setarg (BinProlog and Jinni Prolog > have it as change_arg/3), that simply updates in place a _constant_ > argument with a new _constant_ (no trailing or value trailing needed). > > Based on that, one can build a complete API controlling sequences of > answers, as follows: Nice! B.t.w. SWI-Prolog offers change_arg/3 as nb_setarg/3. It shouldn't be exposed to much to the normal Prolog programmer, but I find it an increasingly handy building tool for things that simply _need_ to survive backtracking and where assert is the only traditional way to realise your goals. Mostly because it avoids all the pain to make code reentrant, cleanup after exceptions and thread-safe that is associated using permanent shared resources such as dynamic predicates. GNU-Prolog also offers somethings similar. It would be really useful if someone (or maybe as a joined effort) created a document (Wiki? We could attach it to the Prolog pages on Wikipedia?) describing all these features over various Prolog systems. Useful for people aiming at portability, for Prolog developers to make it easier to make a choice that is more compatible to others and useful to get to a standard ... Cheers --- Jan

```
:- kb_dynamic(adjectives/1)

adjectives(“good”).



% prolog_load_context(module,Mt).
% defaultAssertMT(Mt).

adjectives(String)/string(String)==>{retract(adjectives(String)),string_to_atom(String,Atom)},
  istAsserted(atomic_adj,adjectives(Atom)).
```

REFERENCES

BACK SOON

(10 minutes)!

[04:14] <dmiles> I wonder if Epistemic Logic for Event Calculus can be turned to Natural deduction or intuitional logic. here is such a system... http://www.ucl.ac.uk/infostudies/efec/EFEC_LPAR_19.pdf

[04:22] <dmiles> Executability conditions denote what data the program must have present in order to execute an action.
[04:31] <dmiles> It was later realized that the application of dynamic logic goes beyond program verification or reasoning about programs. In fact, it constitutes a logic of general action. In a number of other applications of dynamic logic are given including deontic logic , reasoning about database updates, the semantics of reasoning systems such as reflective architectures. As an aside we note here that the use of dynamic logic for deontic logic as proposed in needed an extension of the action language.

[04:32] <dmiles> The logics thus far are adequate for reasoning about programs that are supposed to terminate and display a certain input/output behavior. However, In the late seventies one came to realize that there are also programs that are not of this kind. Reactive programs are designed to react to input streams that in theory may be infinite, and thus show ideally nonterminating behavior. Not so much input-output behavior is relevant here but rather the behavior of programs over time.

-
[04:32] <dmiles> a different way of reasoning about programs for this style of programming based on the idea of a logic of time, viz. (linear-time) temporal logic. (Since reactivity often involves concurrent or parallel programming, temporal logic is often associated with this style of programming.
[04:33] <dmiles> a line of research continued to extend the use of Hoare logic to concurrent programs
[04:34] <dmiles> . A logic is exogenous if programs are explicit in the logical language, while for endogenous logics this is not the case. In an endogenous logic such as temporal logic the program is assumed to be fixed, and is considered part of the structure over which the logic is interpreted.
[04:36] * athousandwordss (~ekansh@139.167.189.33) Quit (Ping timeout: 258 seconds)
[04:39] <dmiles> Even first order statements about natural numbers, may be undecidable, but an extra abstraction step makes the logic decidable again. The basic building blocks for programs in Hoare logic
[04:42] <dmiles> Proof systems for dynamic epistemic logics are used to create programs that can be block chained safefly
[04:42] <dmiles> these are situations that constructive logic cannot be used
[04:44] <dmiles> a simple way to understand how this works is immagine you have a toolkit of constrcutive logics that are suffiecient for handling program situations
[04:44] <dmiles> yet you have no idea what kind of program you will ned to construct at first
[04:46] <dmiles> though pretend you do know at least enough to begin the initial code constructions
[04:47] <dmiles> if you are able to begin to deduce what the program will need to do.. you can begin selection out of your toolkit
[04:48] <dmiles> as time goes on, the program builds up data states that will either limit or expand the operator functions
[04:48] <dmiles> (what possible operator functions will be used)
[04:49] <dmiles> however being careful to not generate the block chain too far out.. since there are several exceptional states the

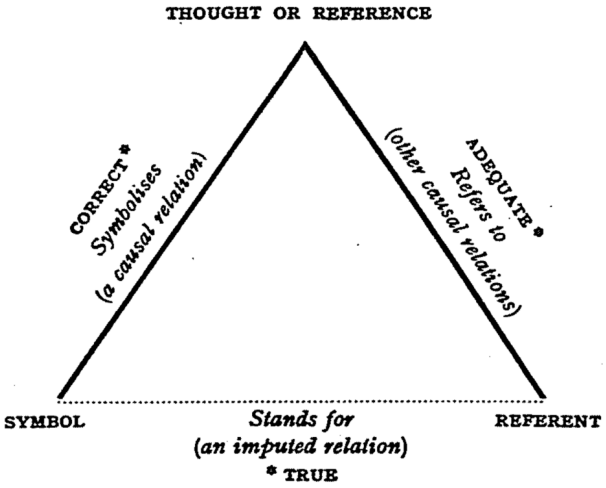
program can take on
[04:50] <dmiles> the program at every point may accept that the next states may be entered... but the program must also know how to backtrack to better more decidable code
[04:51] <dmiles> the easiest way we implement that is taking a rule based approached
[04:55] <dmiles> a name for this is called coherentism.. the program has a past and a current and a future.. Since every element is a choice best on a set supported elements in the past or a set of possible next-ward elements.
[04:58] <dmiles> the secret is these are based on a set of coherence.. even when no coherents exist capable of explaining the next state, you still are capable of getting a proof justification of current position
[04:59] <dmiles> sometimes a program chain has to pauses and await more actionable intelligence.. or the programming needs called in to figure out what contingency plans he forgot
[05:00] <dmiles> this is analogous to "test driven development" hehe
[05:02] <dmiles> here is a paper on measuring that coherence
https://www.illc.uva.nl/Research/Publications/Reports/MoL-2015-30.text.pdf

[05:04] <dmiles> i don't like that it "polishes turds" (this is what i call sidestepping complexity by using "float"s) In other words probabilistic measures
[05:05] <dmiles> but this is OK because these floats create an ordering heuristic
[05:05] <dmiles> ordering
[05:07] <dmiles> one should mentally be rewriting that paper as it is read to were the quantitative measures are used to admit to a set of "Quantities"
[05:07] <dmiles> Qualities*
[05:08] * athousandwordss (~ekansh@139.167.198.196) has joined ##logic
[06:33] <dmiles> because part of the MsgAllert contract says that the user "sees the alert"

[05:08] <dmiles> because the moment you can measure .. the moment you have identified a quality
[05:10] <dmiles> (whenever people measure, they initially create a theory of the thing they are measuring to be fully defined ... .. for instance we can measure "ones" and create numerical values this way.. just like we can measure the second place of a complex number)
[05:12] <dmiles> That is, if the non-quantitative aspect could be taken as a factor while measuring coherence, it would be possible to construct a coherence theory in which no longer needs measure
[05:13] <dmiles> (this is how CYC works this out)
[05:16] <dmiles> Epistemic Logic is sometimes considered the logic of Belief.. but that is too soft or weird.. instead it should be the Logic of "what has be logged so far"
[05:18] <dmiles> in other words we can leave the "belief" domain and allow the logic of "how much of the world has been revealed "so far""
[05:19] <dmiles> also "how much of the world has been been properly distributed over having the N - number of relations currently defined"
[05:20] <dmiles> thus i give formal proof my elephant is much like your elephant.. Yet i have not proved either one of us have the same animal
[05:21] <dmiles> we have in that instance the ideal of potentially of creating two new symbols for our respective animals
[05:23] <dmiles> the only sure thing is that we can not proof that are part of the same composite sets until we use someone's animals to create more symbols
[05:23] <dmiles> the only sure thing is that we "can create a new composite set"
[05:24] <dmiles> "can create a new composite set" this is a fluent action that is a operation about defining probability
[05:25] <dmiles> in other words we are properly grounded in the laws of epistemic definitions of logic
[05:26] <dmiles> (the trick used in EURISKO and CYC was that at every point in the creation of programing block chains.. every operation is an epistemic operation)
[05:27] <dmiles> this is way FIPA and CoABS systems looked ideal
[05:28] <dmiles> (they are languages for manipulation of logic on an epistemic level)
[05:30] <dmiles> however whenever I am generating FIPA programs/operations.. I feel a little untethered .. and fall back on PDDL language to keep the FIPA program honest
[05:32] <dmiles> (to understand the system design, remove FIPA from this description and code your deduction operational routines in PDDL!)
[05:33] <dmiles> (though you'll want to add FIPA back in so at least you have a operational language)
[05:37] <dmiles> (oh also the reason FIPA is chosen, is just for the P being "Physical" .. it allows the mind to grasp how to encapsulate things)
[05:37] <dmiles> (yet there is nothing inherently useful that these are even Agents)

[05:43] <dmiles> anyhow i just wanted to say that constructive logic is useful i think for some applications like single use tools
[05:43] <dmiles> but unable to be used in knowledge based applications
[05:44] <dmiles> but unable to be used to pre guess the application of knowledge in knowledge based applications
[05:46] <dmiles> for instance constructive logic might specify how a SQL INSERT takes place.. but it can't specify the "why"
[05:46] <dmiles> and the "why" is even simpler
[05:47] <dmiles> (and example of "the why" is the customer just got 30$ in credit of their last purchase .. and instantly have store credits they are spending)
[05:48] <dmiles> the "why" is the program had correctly generated the proper coupon codes
[05:49] <dmiles> (they added coupon codes yesterday to the system.. they could do that without having to hire programmers to implement it)
[05:51] <dmiles> because things that have both a data representation and also may contain concepts that don't have data representation are equally processable by a system (why should a programmer explain what a zipcode is?)
[05:51] * groovy2shoes (~groovy2sh@unaffiliated/groovebot) Quit (Quit: Leaving)
[05:51] <dmiles> (why that is classified as a "knowledge based application")
[05:53] <dmiles> i do think that constructive logic can explain things that are not represented directly from data
[05:54] <dmiles> but before people will feel comfortable using this extra level of abstraction ... they need to have a systematic representation space
[05:55] <dmiles> a systematic representation space that does not *always* result in code generation
[05:58] * athousandwordss (~ekansh@139.167.198.196) Quit (Remote host closed the connection)
[05:59] * athousandwordss (~ekansh@139.167.198.196) has joined ##logic
[06:05] <dmiles> also I should mention this is not vapourware .. this was created at Cycorp. It is the Cyc Behaviour Language (CBL) was developed by Stephen Reed over ten years ago.
https://groups.google.com/forum/#!msg/ontolog-forum/ux1EFGQdfNE/GrfDWQo6AgAJ
[06:05] <dmiles> (that has the type classes defined for it)
[06:07] <dmiles> the first program it wrote was one that sent email to people whom left their food in the fridge
[06:07] <dmiles> because it deduced that was usefull code to generate
[06:08] <dmiles> nothing AI-ish about it.. its purely Logic
[06:11] <dmiles> i am afraid perhaps a constructive logic system would not have been given high level goals
[06:11] <dmiles> that would make it start figuring out what programs are missing in order to generate them
[06:13] <dmiles> i mean i don't think i've seen much yet were category theory and intuitionistic logic has been used to teach computers that sort of thing
[06:14] <dmiles> i think that is just a lack of creativity.. it might not be the intuitionistic logic's fault
[06:17] <dmiles> most all the computational logic systems i studied over the years are made to do tricky things.. just the logical systems i see that come out of intuitionistic logic is still wanking on if they can define a sieve yet
[06:17] <dmiles> rather than trying to logic
[06:19] <dmiles> or trying to prove a 4-color theorem
[06:20] <dmiles> when logic instead, could be used to define a 3-fold + four-fold Arthur Young theory correctly
[06:21] <dmiles> which would be actually usable because it has helped fix many syllogisms
[06:21] <dmiles> this 3-Fold is what Aristotle calls "The Triangle of Meaning" ..

Think of the threefold as a "Representation modality"



Symbol "DOG" (Logic) is not the same as referents "MY DOG" (Discourse)
Nor are either pure thought

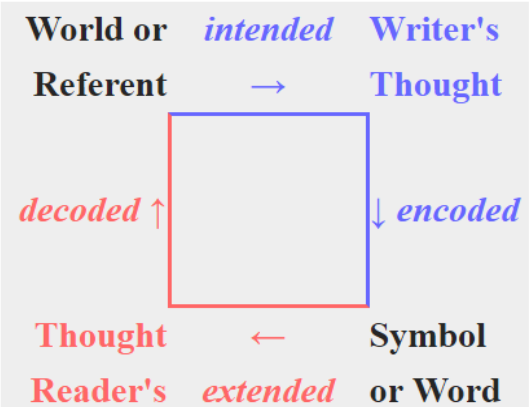
Writer's THOUGHT retrieves SYMBOL suited to REFERENT, *Word* suited to *World*.

Reader's THOUGHT retrieves REFERENT suited to SYMBOL, *World* suited to *Word*.

1. The matter *evokes* the writer's thought.
2. The writer *refers* the matter to the symbol.
3. The symbol *evokes* the reader's thought.
4. The reader *refers* the symbol back to the matter.

(So we reason we must accept that we are coming at it from one corner to another very often)

4-Fold logic teaches us we can still operate in any of the 4 conditions below



Eventually Arthur Young realizes there are at least 12 points of view (3x4) each that could transition in 11 other views
So he created the "Geometry of Meaning" .. The goal was to decide if there was a logically sane way to
Argue from **View 7** and switch to **View 8 (**without making a logical fallacy)

12x11 was a bit overwhelming so he created 3 moves per the 12 positions .. this way he could move from and of th 12 views in stages of 4 , 3, or 7 …

Logical modal operators are defined for such navigation.

"i knew about her not coming to visit" → "I didn't see her come by"

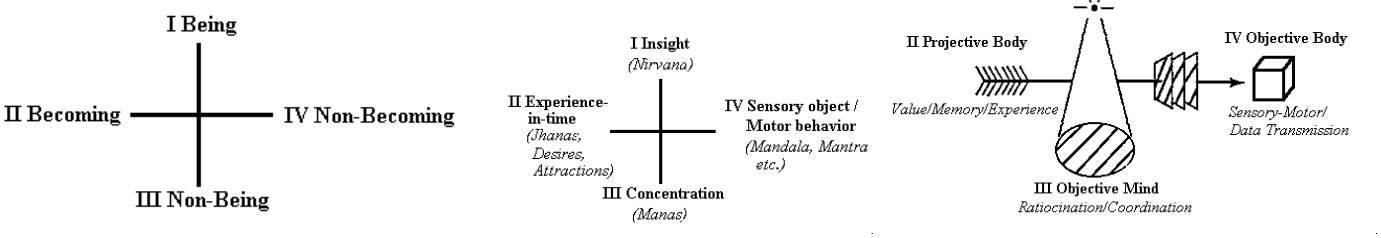
This method it is also capable of obscuring fallacies.. An when it does, They look very very much like things that we sometimes realize not always immediately.

First recognise the idea of the quantity of "X" we may say that when we add "X" more we move the same distance each time away from "0" .. Next we recognise after the idea of "increase by X" should we not also be able to "decrease by X" ? Surely!
In order to recognise decrease we create language for it called "-X" If we can make X number of Xs than we should be able to X^2 .. and Square root X .. (oops did we lose the +/-) "data loss" .. . Can't we divide by Zero?

Representational comes in 4s:

X/0 = "infinity"
Square root of -1 = "imaginary numbers"
Square root of a squared X = "data loss"
X is some arbitrary distance from Zero = "unknown"

Here it get a little silly and kooky.. but here are examples of 4-folds



- "violating the rule of logical types." I know of no such rule. In fact, since Bateson had likened logical types to the derivatives (which also deal with ratios) and there is a rule for derivatives (successive divisions by time, i.e., *ds, ds/dt, ds/dt²,* etc.), he had in effect provided just the rule needed to support more logical types.

Another giant of science, Eddington, placed great stress on the fact that science deals with the relationship structure of the universe, and distinguished between things and relationships between them. Thus a map might show the geographic position of towns and cities, and this would display one kind of relationship. A list of the populations of cities would also show relationship.

So far so good. But do these two logical types -- **elephants**, and the **class of elephants** -- have anything in common? Why should I ask this question? Because I need to show that since they do share one property there must be other types which do not have the property shared by a class and its members.

What then is this property? Both are *objective.* The animals that roam the jungle or inhabit the zoo are physical objects. The class of elephants, or the concept elephant, is a mental object in the sense that it can be communicated. This suggests that since there might be some aspect of an elephant that was not objective -- in fact since we have both a physical and a non-physical aspect of the elephant -- there might be a non-objective, physical and non-physical aspect.

Such would be the *need* for the elephant, which would be physical and something else that is non-physical *and* not objective. This I will venture to say (without proof) is the *purpose* of the elephant. It might be thought that this is physical, but I have to begin somewhere on what I propose to show, namely, that there are four and only four major aspects to the elephant (or to any object) and we can define the two that remain as opposites to the two we have already mentioned, the physical object and the mental object. While physical describes the animal itself, there is another term that is more appropriate to the description of logical types. This is the term particular, which stands as distinct from the class of elephants, which is *general.* I can now define four logical types:

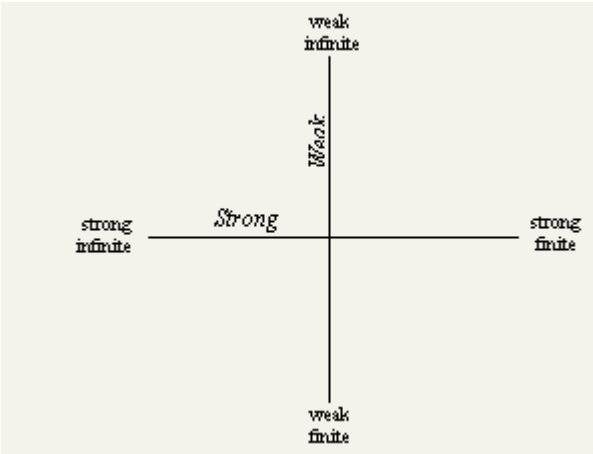
Particular objects correlate to physical things.

Classes are general, correlating to concepts.

Value is general, based on need.

Purpose is particular.

The last two I will call *projective,* as distinct from objective. Here we must be careful because the mind would have no difficulty describing the value of an elephant, but this is not what I mean by value. Perhaps we could shift to eggs. The price of eggs is objective; it can be communicated. The value is something that will vary with persons; if you were dying of starvation an egg would have great value.
This gross disparity is not reconciled by any account I've seen. While I'm given to mixing categories in the sense that I can find a method by which four categories can be shown to be interrelated, this method depends on the recognition that the categories can be defined as the permutation of two dichotomies, a and non-a, b and non-b, a and b being independent. The method also requires something in common between opposite ends of the diameters. This is not evident in the case of the forces. There should be something in common at upper right and at lower left.



And purpose is different again. Your purpose might be to eat the egg, to raise chickens, to make Easter eggs.

1. Particular objective	Solid	Object
2. General objective	Plane	Form
3. General projective	Line	Value, scale
4. Particular projective	Point	Purpose, direction

But permit me to reverse the order and begin with purpose:

Point	Purpose, direction	Particular projective	Origin, first cause

Line	Scale, time, value	General projective	Motivation
Plane	Form, concept	General objective	Identity
Solid	Physical object	Particular objective	Practice

These four can also be correlated to other fours:

	ARISTOTLE'S CAUSES	**JUNG'S FUNCTIONS**		**TRANSFOR-MATIONS**	
Purpose	Final	Intuition		Rotation	
Value	Material	Emotion		Scale	
Form	Formal	Intellect		Inversion	
Object	Efficient	Sensation		Translation	

Furthermore, one of the major tools of religion, meditation (Goleman, 1977), may also be viewed as a "cycle of opposites", a process which cycles through the four levels of reality.

How to correlate this with the breakdown of the law of association?

To do this, find an interpretation of a, b, and c for which (ab)c (not=) a(bc).

Let a = brother

Let b = John

Let c = mother

Then (ab)c = the (brother of John)'s mother = John's mother, whereas a(bc) = the brother of (John's mother) = John's uncle. What is it about this example that is responsible for the breakdown of the law (ab)c = a(bc)?

It is obvious that we are dealing with two kinds of relationship, parental and fraternal. These are two dimensions of relationship, the vertical relationship of parentage and the horizontal relationship of siblings. Hence, where two-dimensional choice (minimally necessary for true mobility) is involved, the law of association breaks down.

https://www.google.com/search?q=%22obvious+that+we+are+dealing+with+two+kinds+of+relationship,+parental+and+fraternal.%22&num=100&rlz=1C1CHBD_enUS725US725&filter=0&biw=960&bih=479

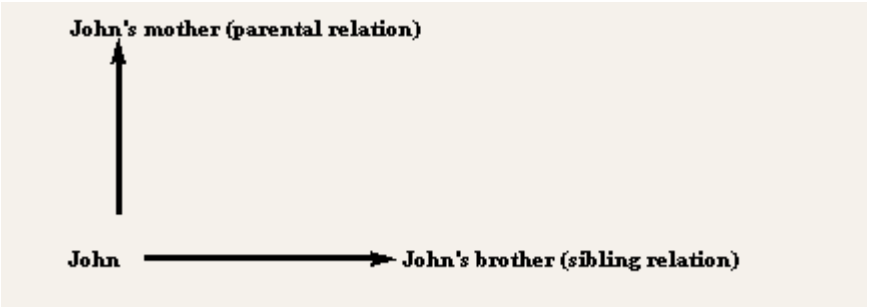


Fig. 37

The breakdown of the law of distribution correlates with my seventh state of process. In my system, this stage is subject to but one law, the law of hierarchy. How does this correlate with the breakdown of the law of distribution?

Choose an interpretation of a, b, and c which will cause the law of distribution to break down.

Let:

a = commander of

b = company b
c = company c

Then ab + ac means the commander of company b plus the commander of company c. But a(b + c), the commander of companies b and c together, is an officer superior to the captains of b and c individually--a colonel, at least--and two captains do not equal a colonel. This is the very essence of hierarchy.

I admit these are analogies. They are not what is customarily sufficient in mathematical proof. They do, however, have the merit of revealing the correlation between two distinct approaches-my theory of process and Musès' theory of dimensionality. I offer them in the hope that the non-mathematical reader may be aided in grasping sometimes unfamiliar conceptual levels in hyper number theory.

I was moved to look for further correspondences.

Example:
If you are holding "2" as a symbol.

TODO - finish explaining Arthur's GOOD IDEAS

when a customer got 30$ in credit of their purchase .. it might be smart for the program to let the customer shop rather than logging them out (which would have been the default behaviour)
[06:31] * rgrinberg (~rgrinberg@24-246-56-85.cable.teksavvy.com) has joined ##logic
[06:32] <dmiles> another example of Logical programing behaviour.. was when CYC Behaviour language sends an email to tell you your yogurt expired.. it one day had to start waiting for the automatic mailer daemon to see if the the message was undeliverable

http://svn.traclabs.com/svn/3t/

Rule I: No more causes of natural things should be admitted than are both true and sufficient to explain their phenomena.

Rule II: Therefore, the causes assigned to natural effects of the same kind must be, so far as possible, the same.

Rule III: Those qualities of bodies that cannot be intended and remitted and that belong to all bodies on which experiments can be made should be taken as qualities of all bodies universally.

Rule IV: In experimental philosophy, propositions gathered from phenomena by induction should be considered either exactly or very nearly true notwithstanding any contrary hypotheses, until yet other phenomena make such propositions either more exact or liable to exceptions.
I think i'll write this in the next couple days as HOL axioms
HOL axioms translate to code automatically in logicmoo
for example (thereExists ?X (and (isa ?X Type) (subRelation ?X causes)
(arg1Isa ?X State) (arg2Isa ?X State) (arg2Genls ?X Phenomena))) (isa (?X ?State1 ?State2) Observable) etc

Douglas R. Miles

@logicmoo

02:34

bottem of page 262 of http://marte.aslab.upm.es/redmine/files/dmsf/p_oasys/160623124554_195_Poli_-_Theory_and_Applications_of_Ontology.pdf explains the lanuage i am using
The binary predicate #$arg1Isa restricts the first argument of a relation to being an instance of the collection which is the second argument of #$arg1Isa. Assertions using the relation are rejected from being put into the knowledge base (KB) if the first argument is not known to be such an instance and such a conclusion will not be derived from rules. The predicate #$arg2Genls similarly restricts the second argument to being a subcollection of the specified collection and #$resultIsa / #$resultGenl makes any non-atomic term generated by a function to be an instance/subclass of a specified collection (Collections are Intensional Types.. Sets are Extensional Types as well) we have 1st Order,5th order etc.. the system of logic invents types as it runs
This message was deleted

Răzvan Flavius Panda

@razvan-panda

02:42

@logicmoo
Rule I: No more causes of natural things should be admitted than are both true and sufficient to explain their phenomena.
Rule II: Therefore, the causes assigned to natural effects of the same kind must be, so far as possible, the same.
Rule III: Those qualities of bodies that cannot be intended and remitted and that belong to all bodies on which experiments can be made should be taken as qualities of all bodies universally.
Rule IV: In experimental philosophy, propositions gathered from phenomena by induction should be considered either exactly or very nearly true notwithstanding any contrary hypotheses, until yet other phenomena make such propositions either more exact or liable to exceptions.
I think i'll write this in the next couple days as HOL axioms

^ is that related to how the Scientific Method works?

Douglas R. Miles

@logicmoo

02:42

yes

Răzvan Flavius Panda

@razvan-panda

02:42

Rule 1 sounds very dodgy...
Any idea where we could find the best definition of Scientific Method?

Douglas R. Miles

@logicmoo

02:44

An example of that is "when dropped,things go toward the floor" , "i will claim it is a force called gravity" "gravity makes things magnetically cling to the earth" but not allowed to say it "makes milk taste good"

Răzvan Flavius Panda

@razvan-panda

02:46

I sense some cyclicity in this: No more causes of natural things should be admitted than are both true and sufficient to explain their phenomena.
No more causes and sufficient to explain their phenomena
that is probably why I felt it is dodgy
it is self-true
basically - the sentence can be simplified without any information loss

Douglas R. Miles

@logicmoo

02:47

i think its saying we should not create two forces if we dont have to

Răzvan Flavius Panda

@razvan-panda

02:48

where did you get those definitions from?

Douglas R. Miles

@logicmoo

02:48

even though it really may be two forces at play
https://plato.stanford.edu/entries/scientific-method/

Răzvan Flavius Panda

@razvan-panda

02:49

becaue of cyclicity:

No more causes of natural things should be admitted than are both true and sufficient to explain their phenomena.

has he same information content as

if some model is always accurate then don't bother considering that it might need other things
I would say the 2nd is a better re-formulation of the first, because of lack of unneeded cycle
that is subsceptible to how many samples would be considered sufficient proof and how good the proof would be etc.

Douglas R. Miles

@logicmoo

02:52

good examle of a Type .. the "Sample"
the program creates a definition of what is allowed to instantiate a Sample well enough to count sample instances
one can later create create a parametric type called Experiment<SampleByMeasuring>
though this is not C# just saying it is good to parameterize what types will vary as the need varries

Răzvan Flavius Panda

@razvan-panda

03:12

yeah
talking about generics
AGI interface will have stuff like Solve<T>
generic over T since T can be any task / goal type
and generic because an AGI is generic / general by definition

Răzvan Flavius Panda

@razvan-panda

03:36

I got the AGI company name reserved
@logicmoo @Scotchmann Fairy Tale - Artificial General Intelligence Solutions
for short just Fairy Tale :sparkles:

Scotchmann

@Scotchmann

03:48

@razvan-panda well, sounds good)

Răzvan Flavius Panda

@razvan-panda

03:52

if anyone says the name does not sound good I will punch their face in the face haha
https://www.youtube.com/watch?v=VwWqFRsOcaY
pff, will play externally in browser

Răzvan Flavius Panda

@razvan-panda

03:58

aaaaaaaaaaaaa
I was certain I will do something very stupid
I for got to put Safe in the name :/
oh well, will have to change it later

Scotchmann

@Scotchmann

03:59

I will punch their face in the face haharecurrence detected.. your actions are not safe :D

Răzvan Flavius Panda

@razvan-panda

03:59

speaking about safe, a very good Haskell programmer that also does Agda proofs and stuff confirmed the monads thing I mentioned earlier
you could use monads to guarante properties about an AGI for example
I will punch their face in the face haharecurrence detected.. your actions are not safe :D

I am RSIing myself all the time - it is perfectly safe and recommended
RSI is one of the core properties that emerge in AGIs
or maybe even the basis of their definition
we humans are biological RSIs
all life is RSI

Scotchmann

@Scotchmann

04:01

RSI is evolution in my terms :D not my only though
evolution is the essense of any process

Răzvan Flavius Panda

@razvan-panda

04:02

anyway, time for us to RSI the shit out of humanity until we can solve the bug of death by aging and other diseases
have to go now, ttyl

Douglas R. Miles

@logicmoo

14:05

i have to say this is the best AI book ...
https://www.amazon.com/Daydreaming-Humans-Machines-Computer-Thought/dp/1478137266

Răzvan Flavius Panda

@razvan-panda

14:07

@logicmoo thank you for sharing

Răzvan Flavius Panda

@razvan-panda

14:55

@logicmoo I am still reading this AGI related book:
https://www.amazon.com/G%C3%B6del-Escher-Bach-Eternal-Golden/dp/0465026567

Douglas R. Miles

@logicmoo

15:07

Neat, I remember that book (my HS library had the 1979 edition)

Răzvan Flavius Panda

@razvan-panda

15:08

@logicmoo what is an HS library?

Douglas R. Miles

@logicmoo

15:08

High School
I remember it was disagreeing with the other AI books of its time

Răzvan Flavius Panda

@razvan-panda

15:09

do you remember what it was disagreeing on?
I barely began reading it only a few pages in

Douglas R. Miles

@logicmoo

15:10

it was cool it offered a different perspective.. Closer to Searle than to Schank

@logicmoo looks to see the particulars

Răzvan Flavius Panda

@razvan-panda

15:10

I am not aware of their different perspectives - but perspectives are just perspectives
N perspectives can all be correct
they are just different projections on reality

Douglas R. Miles

@logicmoo

15:19

I think my perspective was slightly skewed in I saw the book more as adversarial than it was .. What the book teaches is a more well rounded view of the problems than other books.. I saw it at the time as not addressing the problems but just complaining about them
And leaving them generally unsolved from my point of view.. if not worse sometimes declaring them as unsolvable

@logicmoo owes giving at least particulars.. so looking at the book as to not explain them incorrectly

Douglas R. Miles

@logicmoo

15:23

I was going to say the "Strange loop" was a problem but now i see it as a solution :

I even have test in logicmoo that give first class standing to strange loops

Răzvan Flavius Panda

@razvan-panda

15:24

I haven't read the full book yet, but the strange loops idea seems to be the center of it
as an automatic regulation systems engineer that makes perfect sense to me

Douglas R. Miles

@logicmoo

15:26

https://github.com/TeamSPoon/tabling_dra/search?p=2&q=coinduction&type=&utf8=%E2%9C%93
if i understand "Strange Loops" to be a recognition that Co-inductive problems exist
oh i remember what was adversarial it was that the "strange loop" concept seemed to try to permeate things everywhere.. yet did not explain how to represent them
instead claimed them to be unrepresentable
but i hope i am wrong in that last statement
i probably just didn't know enough at the time
it seemed like it was going to eventually claim only strange loop system (which seemed to only be biological) can deal with the fact the world is made up of strange loops

Douglas R. Miles

@logicmoo

15:34

so i saw Hofstadter as a strong adversary against the possibility of creating AGI. However if i was to re-read without that belief i could probably come away wit that he was attempting to make sure i understood what traps i might fall into
Someone else said this:

An approach to infinitely repeated games using coinduction was introduced in [LP12] and continued in [AW15] and the working paper10 [BW13]. Infinite
and coalgebraic games are mentioned only briefly in this thesis, in §2.2.1 and the conclusion. However, given that coinduction and bisimulation are the correct techniques for reasoning about infinite processes, it is likely that they will continue to be important in game theory. In particular, if trying naively to verify

that some strategy of an infinite game is an equilibrium, then infinitely many properties must be checked; however a finitary proof technique based on bisimulation should be expected to work.

Douglas R. Miles

@logicmoo

15:39

summary: A finitary proof technique based on bisimulation and coinduction should be expected to work on Strange Loops
just for refenrce wher ei got that was http://www.cs.ox.ac.uk/people/julian.hedges/papers/Thesis.pdf " logic for social behaviour"

Douglas R. Miles

@logicmoo

15:49

Heh https://en.wikipedia.org/wiki/Hofstadter%27s_law is the egg cracking problem "The recursive nature of the law is a reflection of the widely experienced difficulty of estimating complex tasks despite all best efforts, including knowing that the task is complex." -- specifically the point of the "knowing that the task is complex." .. what i mean is a formal system can express a task so well there is not more ways to break it down
since intuition tells us that we will always stop short of decomposing a task perfectly well (humans and AGI will always fail at this (humans more often)), .. can we ever trust that we really did fully decompose a task or concept?
Thus the workaround has been invented later as a rule of thumb with humans.. "estimate how long it will take you... now double that"

Douglas R. Miles

@logicmoo

15:56

The most important rule of thumb though "if it sounds good to ourselves.. it is good!"
example: "i like this moosehead beer because it has a minimal Caribou footprint" (pun on Carbon Footprint)
Sometimes just the cadence of the inner voices thoughts are good enough to pass our smell test: "...and that is what she said "+<imagine sound of cymbol>

Douglas R. Miles

@logicmoo

16:04

This blatant disregard for using sound judgment (that is in fact what i am proposing!) is how humans coexist with strange loops
The fact we are not drooling imbeciles is only due to the fact we have gotten so darn good at composing mental limericks
We build little caged frameworks that keep us falling off cliffs (mentally)
Whenever we can voice those frameworks outwardly .. we get excited and call this "logic"

Douglas R. Miles

@logicmoo

16:09

However, truth be told, we do not use that logic to think.. 😀 we only at best use it to "prevent" our absurd manner of thinking (whom we really are) from being used outwardly
Why we enjoy music and poetry, is to us we see there was a bit of "order" to something that we did not have to work hard imposing our ever annoying filter we use on ourselves to create "logical order"
Here is an example where that filter was a bit "too good"
https://www.space.com/13197-mars-canals-water-history-lowell.html

Douglas R. Miles

@logicmoo

16:20

Lowell only saw the "canal" lines on Mars
since all the other random lines (which were more prominent) had no sensical reason to be there in his drawing
This facet of the mind is probably the single most important thing that makes us able to be inteligent that we can compartmentalize things so darn subjectively (non-objectively) .. we start to make sense

Douglas R. Miles

@logicmoo

16:29

we are motivated to re-enforce such order (so when our minds are too tuckered out to actively *subjectify everything*) we can still look around ourselves and see perfectly aesthetically round and square and triangular objects in our environment.
we can look around us and everything has a name and a purpose as that makes things at least on some level be less nonsensical than what is really there
i believe even sea slugs do this

Douglas R. Miles

@logicmoo

16:36

This methodology compartmentalizes strange loops
it was wild to discover (for me) that such bass-ackward illogical system of reasoning is the system that makes the most intelligent beings
so far this is the only theory I have heard that explains everything humans and animals do that is logical or illogical .. and provides the exact how and why

Douglas R. Miles

@logicmoo

16:43

other theories and ideas seem to "hope" that some brute force approach will emulate or even discover what i described
what i described though is such a seemingly horribly ineffective design that i am pretty certain programmers would write their programs specifically to avoid discovering it
the NN crowd since they have no idea what their programs do might not prevent it.. haha
But definitely there are more advantageous designs beyond how the human and animal minds work.. that I am sure the NN will adopt something else

Douglas R. Miles

@logicmoo

16:52

What is interesting about the illogical design I proposed, is it's il begotten relationship to symbolic logic .
And it propensity to embrace the most relaxed views of "well formed formula"
(like in the Carabough example above)
all the mind has to do is create well formed nonsense (in that case along a homophone)
If i had decided that was a joke instead of sound reasoning I would drop a E.M.P to reset the system
(called internal laughter)

Douglas R. Miles

@logicmoo

16:59

on the other hand if i decided the thought made sense. I would have to start the "journey" of what it takes to belive my thought
one time i sold a program that was based on that journey.. since hte user didnt want to learn programming they would say the things in english from the voice of the program as to what the program was supposed to be thinking
the well known (almost the same implementation of it) is the game of "Guess what animal i am thinking of"
where the user tells the program (once it fails) what it should have printed instead

Douglas R. Miles

@logicmoo

17:05

ok.. so now how this relates.. my thought originalkyl was "i like this moosehead beer because it has a minimal Caribou footprint"
understand before i said that.. i had no clue why i like the beer
probly i just like all bears
but now i have unlocked the deep mysteries (made up some rubbish) why i like the beer
i probably have to spend the next few thoughts justifying such a lie to myself
eventually I'll make up enough logical evidence (but not until someone twists my arm)

Douglas R. Miles

@logicmoo

17:10

One supporting piece of evidence is I might make up the fact that for now on when given the choice will choose that beer
(right how is what i will do in the future eveidence.. no matter)
in the past had i ever heard of such a beer i promise i would have prefered it!
once i tasted the beer i might claim itws was the slight taste of salmon
in fact i will go onto construct that my mother made salmon every time my favorite uncle came over
had all this been the truth that would make my narative even more belivable to me

Douglas R. Miles

@logicmoo

17:16

i wish this had the name like "the theory of conscription"
Malcome Gladwell speaks of this aspect of the mind in "why you shjould not ask people why they like something"
(no joke!)

Douglas R. Miles

@logicmoo

17:29

What this is all about is constructing a story of whom we are

To be compatible with the most amount of Case Based software one can use SitCalc as it

was designed to allow " ELABORATION TOLERANCE" explained here:
http://www-formal.stanford.edu/jmc/elaboration/node4.html

Douglas R. Miles

@logicmoo

17:34

SitCalc is a formal method to encode experiences well enough to relive them upon remembrance
Elaboration tolerance and belief revision have much in common, but we are looking at the problem from

the opposite direction
elaboration tolerance concerns what is added or changed in memory to get the effect you now want