7/31/00 Bulg.

.01: 166.40 One impt idea is that I have (via MCT) many modes of lrng possl.,
So I can take an advanced part of a TSQ, — decide what
info it needs to precede it, Then get that info into TM —
in any of a variety of ways.

Re: "T. Discovery of t. laws of Algebra" This turned out to be diffclt.!
Perhaps it is because t. laws of Alg. were invented rather than "Discovered"?
No! T. operations of Alg. were defined befor t. Laws were devd. I think t.
discovery was ~~that~~ (perhaps) that t. laws could be extrapolated to Literals.

A nice way to do a TSQ: First write an orderd list of
problems & concs I want TM to know. Then devise a machine that
can "do" that list. Each part of t. list can be done in t. most convenient
way" — w. attempts to do it via lrng. Then I begin to
replace "expert-system-like" parts w. true lrng. As part of t. list", I
do include impt. concs. So even after TM has acquired t. info in a
rather Ad H. way, it can learn rather well

One TSQ direction ANL, conc. of "Quantity" → simple linear eq. soln.
Then more diffy in untangling equs in 1 unk.
2 linear equs in 2 unks, 3 then → n equs in n unks.
also any quadratic, cubic, quartic sequence w. the single heuristic.
Also, very complicated equs in 1 or 2 unks. that are solvable
~~as Eqs~~ linear or quadratics or: . . . .

.24 So: Get Qa ← (Quantity?) meaning clear: lots of disparate examples: → | Also Note: 173.10–.16
$Qa(3) = ~~3~~?$ answer 3. : $Qa(7+5)=12$
.26 If $x=3$ Then $Qa(x)=3$. Note 2 meanings of ff, Then:
1) .26c 2) If $x=3$ Then $y \leftarrow y+1$: Give them difrnt notations
$Qa(3\times(7+3)) = 33$: Learn about paren: Also how related to
stack.
Then: If $x+3=5$ Then $x = 5-3=2$. (soln of simple eqs).
If $(x+3)\times 5 = 50$ Then $x = \frac{50}{5} - 3 = 7$ (more complex equs)
Laws of Alg? If $x=y$ AND $y=z$ Then $x=z$
How to teach this. Meanings of AND, OR, NOT. ?
~~.34~~ If $x=3$ Then Not $x=6$; If $x=3$ AND $y=7$ Then $x+y = 3+7=10$ → 168.21
→ .24 – .34 should keep me busy for quite a while!
[common notation = is $\overset{\text{assignment}}{\underset{\text{Assignment}}{\leftarrow}}$ and == is $\overset{\text{equality}}{=}$]

So first write t. TSQ, (perhaps telling how many examples of
each kind of input.) What I want here is an intuitive idea of
what t. TSQ is supposed to convey.

8/1/00 Bulgaria

Next, do a detaile dcrn. of what each concept "means". Its properties,
— The meaning of any expressn. containing it.

.02 To deal w. roundoff error: for ~~system~~ linear equs only, use the rational fractions instead of floating pt. (ratios of integers). TM has builtin routine for removing common factors of Numerator ~~&~~ or denominator.

.05 — But this will not work for irrationals. — So may be use idea of approximate equality. → (.18)

.07 "Discovery of Laws of Alg." was diff/t to formulate for TM because arith w. large nos. had so little cc for it. We can, hwr, reproduce condition of Human discovering the laws, by ad-hoch making cc of arith >> cc of simple logic of the "laws of Alg."

see 169.37 for remark on this

.12 for Humans, large cc of arith of large nos., was a Heuristic! — we have
.13 to simulate this heuristic for TM! But May be no need! → (.33)

An analogous situation: In Polya an impt heuristic "state the problem exactly": For TM, this can't be done. The problem is stated exactly already! For TM the heur is to "state the problem in its vague, original form, in which ~~[illegible]~~ associations w. other ideas are expressed in the ~~[illegible]~~ form of the statement.

.18 (.05) Re: .02–.05: At an advanced stage we can let TM use floating pt. & "discover"
.19 that if 2 nos are very close they are "probably equal". → (.33)

.21: 167.34: An additional concept: "Truth": ~~[illegible]~~ T(3==33/11) = 1 (true)
.22 T(3==5) = 0 (false). → (.36)

.23 (.13) If we use floating pt. in 486 or Pentium, arith takes ~100 times as long as simple logical reasoning! Hwr, it mite be well to simulate Mult & Div taking much longer than Add, subtract: In floating pts, they take about same time: If we don't do floating pts, Add, subtract are as fast as ~~E~~ logical statements, but Mult/div takes longer.

If +, − takes as ~~long~~ long as logic, then ~~3+7 = 7+3 would~~ x + y = y + x would not be discovered.

Hwr, in SAARB, I found a way for TM to Discover laws of Alg, because
.31 they were useful in solving equations. This is a somewhat difrnt
.32 path from Human Discovery of the laws.

.33: (.19): Well, in most cases if we use floating pts and $|A-B| < \frac{10^{-5}}{A}$, then A=B.
. "will work" almost always: for even fewer errors use double precision & smaller threshold.
.35 (.33R) is probably good enuf for most early work.

← 168.21

.36 (.22) The 167.24–.34 would be an alt Alg. (perhaps w. True( ) added) for me to begin to teach English, it would be easy to ~~extend~~ extend that algebra
.38 in many ways: quadratic, cubic equations; e^x, ln x, complex nos, trig. and h trig., & trig identity proving. A/o several equs in several unks linear/non-linear; Matrices, Vectors, etc.

Anyway: for time being, assume that I can deal w. equality (168.02-.05, .18-.19, .33-.35) and assignments of arby cc. to various Arith. Operations (168.07-.13; .23-.32)

.03 [I want to avoid that much detail, at this pt in my Analysis] → 180.24

Ok. so start TSQ!

Def: "Q" mean "Q." mean "Quantity of" = "evaluation of". Normally it is a number: (real or complex): whether I want it to be able to be true, false or a string (that mite be the output of an algm) is not at present, clear.

So Q(4) = 4 ∴ 4 is correct reply to "Q(4)=?"

Initially TM knows ~~numbers~~ numbers v.s. non-numbers (= strings).

Defs { Q(4)=4 has 6 symbols; 2 are nos, 4 are non-nos. Henceforth "=" means logical equality "←" is an assignment statement

Q(4+5)=20, Q(5*8)=40, Q(9-3)=6, Q(7÷3)=2.3̇, etc.

We may want to teach one operation at a time or a mix.

With Q(4+5)=20 TM learns that ~~[illegible]~~ Q(4+5)=? has answer ~~#~~ 4, 5, add (op code). This is obtained by taking the 2 numbers in "Q(4+5)=[redacted]" and adding them. The rest of the symbols are irrelevant.

Then Q(8-2)=6 .... Q(4-7)=? TM says "11" since 4+7=11, This is, hvr, wrong. After Q(8-2)=6, hvr, it is clear that simply adding the nos. in "Q(8-2)=" will not always work. After a few examples of "-". or one example of hy precision, TM finds that Q(8-2)= gives 10 ~~[illegible]~~ a fraction of time and 6 another fraction. This is the best model, until we do more search and find whether its "8,2 sum" or "8,2 sub" depends on whether a "+" or "-" occurred in the data of the corpus.

(Th. detailed mechanics of this decision is as yet, (undecided unclear)).

After Q(8(+/-)2) is learned w. 100% accuracy (i. very short code)

we try Q(8*2)=16 & ~ data: at first TM looks at the codes representing "-", "+" "*": If "*" code is most similar to "-" codes, it will give hyper pc to ~~[illegible]~~ 8*2, sub than to 8,2, add. Hvr. as we give more examples of "*" (a/o more precision), TM looks for a better code and finds that the symbol * makes 8,2 mult, the correct response.

Similarly, it finally lrns ~~[illegible]~~ Q(8/2)=4.

.32 One thing is unclear: Just how much cc TM spends on these srches. After being give Q(8+2) act, Q(7-3) act., it could stay with 50% accuracy. Perhaps I can arbly give it a Threshold of p=.9 or .1; or p=.99 or .01 for its predictions.

.35 T. .99 .01 can be obtained [illegible] (bigger ssz. or more precision (no. of bits) → 171.31

.36 in examples of corpus. This seems to solve it for Algebra & much of Math! → 170.01

.37:168.12 SN Cheapness of Ram & disc storage make for a negative heuristic for psnrs w.r.t. compression techniques. Cheapness of hyper speed computers is neg heuristic for ~~[illegible]~~ learning efficient use of machines.

(.37) Negative Case Counts

(.06) TSQs & /possl. Extensions of them

.01: 172.37 "oneway" connection from A to B could be a ckt. w. A as input, that "drives" B in a way that a cap of size C would drive B from A. To simulate t. cap acting from B to A, we would need another ckt of this kind. The value, "C" of t. cap. is a voltage input of t.

.05 simulating ckt.

.06: 172.17 : 167.24 - .34 (173.10) is initial TSQ [redacted] Some other relevant notes, extensions additions:

167.34 - 168.05; .18 - .19, .33 - .34 || 168.07 || 168.21 - .22

.08 Extensions: 168.36 - 169.03, 180.24, 18[illegible].5

.10: 167.24 Another possl part of early TSQ:

~~[illegible]~~ 3=3 ; 3=? x=3 ; x=? 3=? [y=x, x=3, y=3; y=x, x=7, y=?

.12 Would it quickly learn x=y, z=y, w=z, x=? ; w=? i.e. a long string of equalitys. [Advanced Algebra | 2,3,n unknowns linear / Matrix algebra | quad cubic biquadratics | e^x ln x, complex nos, trig identity proving. → 180.34

Also I may want to introduce t. "If, Then" / logical notation. (not control If, Then)

What is another good name for it.? ~~[illegible]~~ Cond imp.

.16 Condition, implies. Cond, Imp. / cond means "for this problem only".

~~[illegible]~~ Hvr, Just ~~What~~ is "for this problem only."" How much can be transferd (learned) to other situations?

A real possy; That Ordinary Human induction ~~[illegible]~~ (even in a pure math setting) involves very many assumptions (≈ "Common Sense"). While it may be quite important & very pervasive, it may be possl. to List the assumptions made.

e.g. in Cond X=3, y=x imp y=3

we carry x=3, y=x into t. imp side of t. cond, imp. —

[~~But we do not carry~~ we then want TM to carry the transitivity of =, that was lrnd here, into future problems, but not carry x=3, y=x etc into future probs.

How serious a problem is this? It must be "clear" "By induction" i.e. (or inducable)

.28 ~~[illegible]~~ from t (corpus examples) alone. → 174.04

Each induction problem has to be defined either by CB or by compresn Expected. — Otherwise TM doesn't know when to stop searching.

On OSL: A very common use of OSL is in Heuristics — Someone successfully worked a problem a certain way in t. past. This is about t. same as "Case Based (Learning / Reasoning)". The practical aspect is indexing events / processes, so they can be retrieved for (CBR / OSL)

.37 SN On Negative ssz: using t. D.F. $p^A(1-p)^B$ as a prior. is equivt to ssz of A+B: This D.F. w. A &/o B < φ could be meaningful. It would mean concentration of pc at φ &/o at 1 (Not unusual: Particularly in MoR). → 174.01

8/9/00 Bulg

.01: 178.40 Another kind of "Hint": Hard or soft (= probabilistic) restrictions on set of concs a/o methods of combination of concs used to solve a problem. An extreme case would restrict TM to search for a known (to teacher) soln.

In general t. value of TM's srch for a soln. ~~to TM~~, is reduced (sometimes considerably) by t. giving of "Hints" — so when we give hints, we should try to minimize this Bad aspect of ~~[illegible]~~ t. hints.

Also possl. Soln. is MAD (Mutually Assured Destruction). Each opponent is unsure that he can completely destroy all all adversarys, so it he tries & fails, t. remains of t. Adversary(s) will try to destroy him. — In this case, best concentrate on defense & development of more powerful ~~[illegible]~~ weapons! (~~= Arms Race~~ deter) = (offense ~~Arms~~/defense) race (Arms offense/defense) race.

This written — Prior to CIAP = — copy to "CIAP".

A search is "Of Value" if ① It discovers/new useful heurs/concs. ② or it changes params of old concs. ③ Discovery of solns to probs that are different from those obtaind when Hint(s) were usd. (This is maybe a subclass of ①)

.17 "Am. Anfang" I don't expect ~~[illegible]~~ searches will be at all "interesting":

.18 TM will simply find concs that are compositions of other concs. Later, I expect that TM will watch (another person or itself) work a problem ~~[illegible]~~ & induce (by OSL or ~~[illegible]~~ Multishot lrng) a way to do a search or a

.20 set of rules or heurs for ~~[illegible]~~ solving probs of that type.

~~3~~ Big Complaints I had about t. TSQ's designed at SAARB:

.22 1) Hard to write (MCT is supposd to fix that!) ③ It was beginning to "scale" badly; pc's of concs were getting too small too small as TM found more of them; poor organization on Poor organization on TM's part?

2) It only found Cons that were solns to problems in t. TSQ.

This second complaint is (partly at least) dealt w. in .17. I need more complex probs befor I begin to notice useful sub-trees in solns.

— also I need more complex probs befor TM begins to discover stuff like .18 – .20.

A big thing wd be for TM to discover what a "Proof" is! Ramanujan's "education" George Shoobridge Carr top of 176

And, of course, if TM could usefully go thru Carr's book(s), or just part of one! it would be a Major Break thru.

.37 List of Recent elementary TSQ refs: 178.09 – .27 ; 175.25 – .40 "176.21 ff. is how TM works" 174.30 – .40 ; 173.10 – .16 includes advanced stuff ; 173.06 – .08 ← {gives refs from 167.24 to 169.03

characterize operator v.s. unordered set of finite objects

From .37 ff I should make a list of (conc, example sets) and ~~[illegible]~~ Try to order them in a reasonable way. I have been concerned w. t. exact. details — then DON'T BE. Just write t. TSQ & put TM to work on it. TM should be able to work it even if it's not exactly right! I may discover some very serious deficiencies

180.01

T.'

.01: 179.40 in t. TSQ's! But O.K.: Then maybe fix it. ← (But Perhaps Generalize it first — like .11 ff)

.02 Also, I want to Address t. "scaling" problem (#3 of 179.22R) ← (A very Serious Problem)

.02 Involves finding ways to categorize conc., ways to know when they are likely to be used. I should be able to get ideas about This by examining my own workings of t. ~~probs~~ probs. in t. TSQ.

.06 So make a list of actual problems in 179.37: Hvr, do it on ~~a~~ a text editor, so I can insert lines. In Raso notes, write

.08 Comments on t. set of Trg. Examples.

T. Q of Operator v.s. Unordered set of sets seems impt: Think about it.? ABcde

.11 Otherwise, Just write t. TSQ and worry about .02, when it occures.

But when it occures: Try to get a good perspective of t. Problem — ~~[illegible]~~ "IN ENGLISH"(at least) — But look at it in t. most GENERAL Possl. way.

T. way it looks Now! That t. details of t. TSQ are not very

.16 impt. — That it does, hvr, have to ~~have~~ t. needed info in it. If it

.17 does not, This will become apparent in t. mode of failure of TM's attempts to predict/"teach" it.

Hvr, the "scaling" problem of .02 can be serious.

Also note that ~~in t. former~~ if we run into .16 — .17 (inad not enuf info in corpus), t. MCT should give us several ways to put ~~[illegible]~~ t. needed info into t. corpus!

.24: 169.02 SN on Extending t. TSQ! ~~#~~ x^2, $x^{\frac{1}{2}}$, $x^{\frac{3}{10}}$, x^{real}

exp(x); ln x; hyperbolic trig funcs., (Laplace xfm?)

complex nos. " " " (" ") (ℱ xfm)

So befor I introduce complex nos, hyperbolic trig & Laplace xfm are definible.

(~~Also~~ for laplace, I need to define Integrals ←? — maybe not — use real analog of fourier series.)

→ Tho. for both Laplace & ℱ xfms There are "Σ" rather than "∫" versions

⟷ Also note! Solns of polynomials can be extended via complex nos. This does not need e^x or ln x. So maybe introduce complex nos after "polynom. soln". Then e^x, ln x etc.

.34: 173.12 → Q: Could I Get TM to invent Complex nos, etc. ? This is in T. direction of t. "creative" Mathematician (Conats' "AM") → 176.01 – .13 (I want Ramanujan)

Perhaps Think about How TM could invent Complex nos: What would be ⓐ Motivation ⓑ heuristics to direct it to ~~t~~ t. best known way to go complex nos.? Any other way(s) t. idea of a "soln" of a poly, can be extended?

8/10/00 Bulg. 181

→ one way I dealt w. my personal "scaling" problem: when I lrn a new, useful conc. (= idea a/o category), I try to think of all t. areas of inquiry in which it could be useful. This could be very narrow area or very broad problem area. This process of suggesting applica areas is solvable by Inductive Inference. → Infant TM has too small ssz for induction in this area — But infant TM doesn't yet face the "scaling" problem ↳ 183.07

.01: 180.40: Re: transcendental functs: There is a nice, fast computer way of calculating them all (≈ t. Cordic Method).

Another (or perhaps t. same) way, generates e^x by $e^{x+a} \approx e^x(1+a)$ logs are found by expansion of binary nos. Multiplication & division are done by a kind of successive approxn used in A to D converters.

.06 More on "Scaling": T. scaling problem is one aspect of "Too large CJS": In t. past, I'd dealt w. this by TM pc of Concs by factoring them, then making TSQ to discover t. factors & make them of hy pc. Hvr, even this technique will fail for "Scaling" reasons —

.10 i.e. if we don't find ways to categorize to narrow down t. search for a conc (t. "narrowing down" should be (stochastic) soft rather

.12 than Hard = deterministic)

.13 Hvr, it may be that for a Young TM, there is not enuf ssz for useful

.14 categories, to do this.

Note that this "narrowing Down t. Srch space" (of .10–.12) is one of t. commonest types of Heurs. — But for a young TM, we will prob'y have to insert t. needed heurs. — because (.13–.14) TM will not have enuf experience to derr t. relevant heurs. itself.

,19 — So t. "Scaling Problem" may be equivt to "Having Adequate Heurs"

→ An adequate soln. to t. scaling problem implies that t. CJS of most problems stays about t. same as t. corpus grows

So start of TSQ: file C:\ TSQ1 (on Zenith laptop) see 181.05

167.24–.34 I had idea that t. Conc. of "Quantity" would be useful. but I'm not so sure of this now! — It may be that TM could lrn these TSQ's "about as well" w.o. that "Conc." Part of my objection to it was that I don't know exactly how I want to define it.

$Qa(3)=3$: and $x=3$ imp $Qa(x)=3$ But also "$x=3$" So I'm not sure what $Qa(\cdot)$ is about! Usually, if not always, if "$Qa(\ )$" are removed, t. expression is still true (& vice versa): so it looks like it's redundant. — The redundancy can have heur value, this one seems vacuous! (tho I'm not really sure).

Anyway: l, m, n are say 32 random bit nos.

HVR: Many of t. properties of =, +, − discoverable by TM, empirically.

,33 T. way I'd considered doing this (most recently) was to have +, −, ×, ÷ be very expensive relative to logical reasoning.

In t. present tsq pgm "TSQ1" TM "learns" "laws of Alg" but the learning is of a "Skinnerian" kind (small cjs, very little cleverness)

.37 In .33 one way to do this is to have TM try to find ways to do arithmetic "fast". Probably, it would be a good idea to try Both (.33, .37) and t. method of "TSQ1" (181.5)

182.01

8.10.00 Bdlg.

181.5

TSQ1

This will be an ordered list of problems, tasks, definitions, ...
--- toward the construction of an initial TSQ for TM.

167.21-.31

l, m, n are 32 bit random numbers (or 80bit IEEE reals?) QBASIC 32 Bits for single prec. 64 " " double precision.

u, v, x, y, z are variables

":" separates examples. "," separates data within an example

cond means "conditions for this problem"

imp means "what is implied by these conditions"

n=n e.g. 3=3, 7=7

cond x=n imp n=x

"[" and "]" are "metasymbols" [3+5] means "8"

l+m=[l+m] e.g. 4+5=9 --- learning Addition 4+5=?

[l+m]=l+m e.g. 9=4+5 --- Equality Commutes ? = 4+5

cond x=y imp y=x Equality Commutes

l+m=[m+l] Addition Commutes

x+y=y+x Addition Commutes

x+m=m+x Addition Commutes

l+(m+n)=(l+m)+n Addition is Associative

cond x=m imp x+n=m+n --- if equals are added to equals the sums are equal.

cond x=y, u=v imp x+u=y+v --- as in previous example

l-m=[l-m] learning Subtraction

m-m=0 meaning of Zero

x-x=0

cond x=m imp x-n=m-n -- if equals subtracted from equals, remainders are equal

7/26/00 Bulg ALP (.01) Convergence of lnp' to lnp

Optzn Techniques: (08) 1205 : Jsch

.01 In S78 I showed ~ $E \sum (p_T - p)^2$ was < ~~maxima~~ $\ln \frac{p}{p_T}$

~~r is max value of ratio of~~

Also Note Method of srch for peaks.

.03 A Q was what about $E \sum \left(\ln \frac{p}{p_{True}}\right)^2$?

.04 Actually, [redacted] S78T3 proof showed $E \sum \ln \frac{p}{p_{True}} < \ln \frac{p}{p_T}$ which was stronger than .01. Tho .04 is not ~~directly~~ (apparently)

.06 comparable to .03, it does suggest that $\ln \frac{p}{p_T} \to 0$ rapidly!

155:20 ff on ordering of trials for $\overset{OT}{OZ}$ probs.

.08 152.24ff On finding ~~peaks~~ Max of a function: Assume a certain value

.09 for ϵ: $(f(x+\Delta) - f(x))/\Delta < \epsilon$. (any Δ)

Using this ϵ value, do a search w. mesh edge δ. Find f_{max} for that Mesh. ~~Then~~ Because of .09, certain regions of the mesh, we know can't have pts w. $f \geq f_{max}$

.12 Do search w. mesh edge = $\frac{\delta}{2}$. Over regions of net for which f could be $\geq f_{max}$. This $\frac{\delta}{2}$ search yields new f_{max} (maybe = old f_{max}) and gives (perhaps) new regions where f can't be $\geq f_{max}$.

To do the search: Make a "linked list" of regions in which $f \geq f_{max}$ can occur: Each such region has a Δ value and 2 end values f_1 & f_2; ~~Also~~ So, peak possl = $\max(f_1, f_2) + \frac{1}{2}\Delta\epsilon$: we elim regions from our linked list when that becomes < latest f_{max} found.

So, we do the whole space until all pts are covered that could have $f \geq f_{max}$.

Then we do $\epsilon \to 2\epsilon$, & we redo the whole search.

We re search for new ϵ then $\epsilon \to 2\epsilon$ & redo it again.

.23 This continues until we find no new peaks w. $\epsilon \to 2\epsilon$. (.32)

.24 MORE GENERALLY: R.E. OT's: An OT is impossl. over a continuous space unless one makes some constraints on G(x). Otherwise G(x) can be anything at any point and info about some pts. does not constrain the value at other pts., so no search method is possl. (other than random or exhaustive srch at hyperdimension region).

In Approximating Functions f(x) using linear sums of other Basis functions; It is well to know the constraints on f(x) & design the "Basis" functions for these constraints: — This is ~~true~~ for

.31 Neural Net approxns as well.

.32 : (.23) A weaker assumption than $f(x+\Delta) - f(x) < \Delta\epsilon$: Constraint on ≈ "second deriv" of f. So if ΔF has been large in an adjacent mesh region, we allow ΔF to be $\epsilon\Delta$ larger than that in the present Mesh.

"Second Deriv" may have to be "Second Difference": The details of how to use a "Linked list" (or some similar data Str.) would have to be worked out

7/28/00 Bul 6

PSG discy

.01: 149.40 This defines [redacted] R_s = t. symbol set that precedes S.

Next: Define the symbol set that follows R_s (R will be a member).

Call it T_F : So we can make t. concat product set $R_s{}^\frown T_F$.

So, we have folg. ways to create (tentative) nyunsts:

1) All prim symbols are nyunsts w. 1 member.

2) T. set of symbols (or nyunsts) that precede or follow a gn. nyunst.

3) The ~~AND~~ "OR" of 2 nyunsts.

Note! These "nyunsts" are really "BAGS": so its easy to "OR" them but "AND" ~~is not at~~ has no obvious Defn. (I wrote about this in t. past)

Maybe see notes on G. Wolff.

To define ~~x~~ α_F (t. set of symbols that follow symbol α):

we write αF (a special symbol.) before coding t. corpus: every time α occurs, we code t. following symbol as a separate Bernoulli seq. If this method of coding ~~x~~ ↑ pc of corpus, then α_F is a legit ~~x~~ nyunst.

How we define & evaluate R_s (t. ~~x~~ symbolset that ~~preced~~ precedes R, is unclear

---

(SN) I do want to find ways to effectively "Undo" part of a Gend. Grammar.

— Otherwise, I will simply accumulate errors. Could "Reparsing" be of help?

PSG rules are of 2 types only:

$$\boxed{\begin{array}{l} A = C{}^\frown D \\ \hline A \to C{}^\frown D?E \\ \quad \to F{}^\frown G \end{array}}$$

concatn of 2 ~~x~~ T = concat of 2 NT's.

NT = Boolean AND of several other NTS or of several Concats.

So: A Gend could be a [redacted] small (cogently) set of Grammar rules.

A Mut/recmd (crossover) could remove grammar rules as well as add or modify them.

A loop $A \to D \cdot A \cdot E$, $A \to CD$ is same as any other Gramm rule.

Hvr. every loop/NT should have at least one production that is not in that loop!

e.g. $A \to \overset{=}{\doteq}$ [redacted] $A \cdot D$, E But in general when there are many loops, t. nec'y constraints (to assure meaningfulness), are complex.

~~It may be that~~ I want to know t. rules so I can avoid making trial ~~x~~ modifns (Gends) that are meaningless.

~~IN GENERAL! Explore how ... ~~

LATER! Read 149.15 – .40, 161.01 ff. when starting on PSG discy

I think there are some Good ideas here: some need developing, hvr:

I vaguely ~~got~~ have t. idea, hvr, that PSG discy may not be critical to TM or TM's lrng. of language (since it will not lrn lang. by looking for regys in ~~x~~ a large corpus: It will learn by associating English Q's, A's w. Things ~~x~~ in a Domain it knows About.

7/29/00 Bulg REV

140
141
148
158
[illegible]
159
162

[illegible]
267
268
274

.01 : 159.40 : 153.01 ···· 157.19 Discusses ways to get these 2 kinds of approxns of G(X) w/o P(G,X).

~~[illegible]~~ 154.01 : On relation of Mutations/Crossover to the P(G,X) & G(X) approxns.

157.24
~~157~~.24 — ~~[illegible]~~ : That the Treatment of ~~Optim~~ "Path finding" of 152.13 – 156.05 Doesn't do "Experiments" : How to deal w. this (a bit).

157.28 – 158.06 Suggests ~~a~~ one possl. Genl approach to "EXPERIMENTS" in the treatment of 152.13 – 156.05

160.01:.06 ON Sol 78 T3 : How it can deal w. the $(\ln\frac{p'}{p})^2$ error criterion !

161.01:40 PSG discovery : (Continuat of 149.15 – .40)

Review of Reviews. The pp of reviews: 140, 141, 148 [158] 159, 162.

140 ~~re~~ details from 117.21 to ~139.40.

159,162 " " 142 to 161

141 Discusses several Models for G(X). (easy to find X ∋ G(X) is ≅ Max.)

148 Gives an allover view of Motivation : Why this stuff is impt.

158.13 : A repeat of the 148 : perhaps slitely difrnt.

164.01

72
52

.01 Now, I want to outline a ≈ final T.M., based on MCT. ā (probly perhaps) Lsrch. First a rough outline, Then progressively more detail.

T. main idea is to do a T.S.~~Q~~. starting w. Algebra. Using various ~~teq~~ techniques to reduce CJS. Mainly lrng. definitions by induction: Then giving problems in which those defns. are useful.

{ Also, we want TM to be able to devise complex (perhaps recursive) concepts to solve problems. { look at Koza's methods for some ideas on how to do this — (it isn't obvious!)]

After T.M. understands some Algebra, begin teaching English, by asking Q's in English ā expecting Answers (in TM's normal mode at first, — but later, in English). See how far we can go w. This — ā see if TM can work hard probs (eventually). Since Koza seems to be able to work Hard probs, using a rather unsophisticated srch system (i.e. it doesn't learn heurs to guide srch), I think that ∅ should be able to do far better.

.16 A VERY IMPT. fundamental idea: That I want TM to learn things t. way I think humans do. — So if there's some reasonable, good, way to solve a problem, perhaps using heurs based on experience of TM ā/o t. Scientific Community — I want TM to learn to use those heurs. — ideally it should learn them from its own experience — but if t. CJS is too large, I will give it "hints" to ↓ CJS.

— The default "hint" is to be ~~force~~ "opening" TM ā arranging for it to "discover" t. desired conc. I will probably be giving lots of "hints", but I expect that eventually TM will be able to discover some serious, large CJS concs, "on its own".

At first, TM will solve all inversion problems by simple Lsrch. Hvr, as TM matures, it should be able to do "proofs" ā "finding solns" that it does by inducing from examples. This induction is done by Lsrch, Then t. Algms (found by Lsrch) are used to solve t. problems.

— A simple case would be soln. of linear eqns. — or perhaps numerical ā/o literal evaln. of an alg. expressn.

⟷ Perhaps Look at latest version of t. "Paul-Solomonoff" paper for some useful ideas.

Re: Alg Notation Lang: A new idea was to teach t. idea of "Quantity" — which make ALN a lot easier, ā its a conc. that is very useful later.

Perhaps ~~Review~~ Search Notes.

ANOTHER impt idea: Be sure to RETAIN PERSPECTIVE. Don't get bogged down in picky details. Keep discussn. in BROAD, "English" terms as long as possl.

165.01

7/30/00 Bulg.

GA

Evolutionary Computing

.01:162.40 On Speeding up GA: Paper in Recent IE³ T on E.C. July 2000 p. 188
Uses "Momentum" concept from Back Prop in ANN. to speed up GA srch: In the cases they
report "the speedups enormous, — but of course they present only the cases
.04 where it worked. Still it would seem like a v.g. idea. → (.09)

They suggest that other ideas from Back prop. be used for GA. ~~P~~
Perhaps partial derivative idea?

Look at these

Pogio wrote a paper showing how ANN, radial Basis functions, fuzzy (logic controllers)
were all very similar. Also paper by Farmer(?) on similar theme.

.09: (.04) Re: "Momentum" This is essentially a derivative of G. They multiply it by a
constant, μ, to obtain next jump size. The constant μ is chosen by user.
If they had second derivatives they could do optimum jump. Hvr. second
derivs are very noisy. One has to pool data to ↓ noise: 3 kinds of
"pooling" ① Over time of same vector component of $\vec{x}$ ($\vec{x}$ is fn of t.
(and) ② over different components of $\vec{x}$. ③ Both. $\vec{x}$ components
in ~~data~~ near by or more distant regions of $\vec{x}$ space. (We are doing
a STEIN'ish analysis).

Also Two books on GA Theory: ~~Rev~~ One reviewed in recent
July 2000 p 191
IE> Trans on Evol. Comp. Other (by reviewer) was referred to. [redacted]

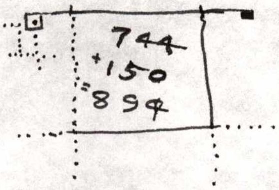
① Michael D. VOSE
~~M.Vose~~ OM.VOSE
② Hans-Georg
Beyer Springer V. 2000

Also look at a book that called "Searching w. Probabilities" Andrew J. PALAY 1985
↳ This uses Chess as Example Source of Problems.

172.19-173.05

(185.04)

7/30/00 Bulg.



164 is a "Speeding up GA" See earlier Bulg files

.01 : 163.40 E.G. : I want TM to learn meanings of various (Terms definitions) in Algebra. In many cases, exact definitions of what I want, are quite diff't. ("Set" is a good example: even "set of numbers") In these cases, I could just use ruff definitions: give some reasonable examples, w.o. my really understanding how t. complete defn. could be understood by T.M.

So: TM would at first have very narrow ideas of what a "set" was, & would later have to "Generalize" this concept. T. simplest way to Genz. is to remove Constraints; But this is certainly not t only way ☺.

. I mite just give a few examples of a concept, then see how

.11 T.M. has done on it by asking about new ~~[illegible]~~ cases. — Then I could see if it (TM.) made any errors in t. concept. This could go back and forth w. the errors by TM. Hvr., if it does go "back & forth" many times, I'm afraid of Getting close to "Skinnerian lrng" — w. TM.

.15 ending up w. a long defn. w. many "special cases".

What To do: I do t. "Bad" lrng of .11 – .15 : this teaches me how my examples are inadequate. I then mind-wipe TM of that part of t. TSQ (if this is poss!!) → Tho I could store TM's mind befor I do this section of t. TSQ. — so easy to erase subseq. [redacted] "maltnng". ←("Mal-training"?)

I Want to keep ~~[illegible]~~ my ~~defns~~ "in English" so they are as "~~Vague~~" (≡ general) as possl. In t. case of "Set", it's maybe hard to define it w.o. using t. concept itself! — So maybe it's ~~a~~ "Primitive Concept". So— if I run into trouble devising an "ENGLISH" definition, Consider t. possy that it may be a Primative concept!

Another Impt. idea is that I needn't be very careful about introducing concepts that may have to be discarded or grossly Modified, later. TM should be able to deal w. this, & not "get stuck" somewhere by "Painting itself into a Corner" ☺.

.32 So what are some prelimy concepts / defns? — What are some prelimy probs?

first! Quantity: ≡ ~~[illegible]~~ Qa ~~[illegible]~~(32) is 13. Qa(3) is 3
~~[illegible]~~ Qa(3×7) is 21.

$\underbrace{\text{If } x=3 \text{ Then}}$ Qa(x) is 3. The If, Then idea

The If gives a general universe. "Then" gives some Conclusions about it. The idea of Parenthesis. Also ideas of a "Stack" T.M. should lrn to (166.01

~~Stack~~ stack **STACK** THE PRIME HEURISTIC

↓ .21

.01:165.40 relate a stack to parenths.
If x = 3 Then 4 × x = 12
If 5 × x = 12 than x = 12 ÷ 5.
Perhaps just try teaching numbers, + (and perhaps −.) Then,
~~[illegible]~~ Q2(3) is 3. If x = 3 Then Qa(x) = 3
~~[illegible]~~ Then eventually: If x + 1 = 5 Then x = 5 − 1 = 4.

.07 Maybe use "=?" as way to ask Question. This is a reasonable induction Question form.
So list some ~~Com~~ Concs I'd want TM to recognize;
Some kinds of Behavior & kinds of lrnng.
Q2: The idea of "Quantity" or "evaluation" I'm not entirely sure what I mean!
The idea of Parenthesis & perhaps related stack. ← This could be very impt! Very useful in ordering tasks.
165.32 — 166.07.

▨ Ability to solve simple eqns, then more complex eqns. After eqns w. +, − are solved, ×, ÷ are introduced & more complex eqns are solved.

Another trick: TM learns to solve x + 3 = 7;
then we teach it mult, div. & see if it learns to solve ~~x+3=7~~ x · 18 = 13 faster. Then it lrns to solve x + 3 = 7. The Q is: does it recognize the similarity? (A variety of "(short lrng")

.21 The **PRIME HEURISTIC**: "All the info is in the p.d. ⟷ QUICK L srch is the best way to solve the problem"
■ Apparent Exceptions (like Quick sort) can be dealt w. by properly defining the problem. In this ↑ case, one redefines the problem to be ~~[illegible]~~ lo cc solns of problems (not necly max $\frac{pc}{cc}$).
MCT makes it relatively easy to put any kind of info into the P.D.

Hmm, Note that Most INV probs. are Solved as OZ probs.!

amount.

.27 Perhaps: First write fair ammount of ruff TSQ. Do this before working out any "details". Stuff like 165.32 ff, but more "complete"
Something close to what I expect to feed to TM.
For .27 ff: In later (or even perhaps earlier) sections of the TSQ, Tell just what Heurs are expected to be used (if any!).
Look at those 3 or 4 TSQ's I worked on in SAARB: To what extent can I use them or (parts of / extrapolns of them?
Promising Approach: Write TSQ's for an Advanced TM, first. — Then work backward to get to Primitive Concs. a/o use many "Hints" to ↓ CJs of ~~Advanced~~ Solns. to "Advanced" Problems.
Sol 89 had some reasonable ideas about TSQ's: perhaps reread it.
I had difrnt "Phases" of the expected TSQ — culminating in Ability to read/understand English text books.

167.01

.01: 169.36! So, next, $Q(3*(4+2))=18$.
The idea of introducing $Q(\ )$ notation was to ~~get~~ give TM a "feel" for "Quantity" — but my guess, is that t. examples up to this point, would not do that, & so t. concept of "Quantity" would not yet be used as a heuristic in (.01)

So try this: "If $X=3$ Then $Q(X)=3$."
If $X=Q(5)$ then $X=5$

Q: Do I want to give occasional negative cases? A neg case means that t. code should not give hi pc for those examples. — T. idea of neg cases may have arisen where we are fitting "concepts" (in Valiant's sense) to data. Winston used it in his "Arch" v.s. "not an Arch" paper. Also Grammars to [redacted] corpi.

A negative example is meaningful for pure Math. A neg. example is a

.12 positive case of "Not(t. concept)". Pos & Neg Cases have interesting property ~~with~~ → 172.03

Maybe getting too Complicated → well, $T(Q(7,7)=1)=$ ~~f~~ (Truth) Value of $(Q(7,7)=1)$ is "false".) ← Maybe getting too Complicated!

.15 $4*Q(4+5)=36$: So Actually "Q" acts like parentheses.
in (.15 L) → ommiting t. Q would give correct expression.

●

What I've been trying to do is get TM to understand what a "Quantity" is:

we: say $X=3+5$ what this "really" means is that $Q(X)=Q(3+5)$
so: X is a symbol: $Q(X)$ is its "value" "3+5" is a symbol. $Q(3+5)$, its value.
Would things be easier in RPN? T. system is certainly less a.H. than normal Alg. notation.

{ An advanced problem would be for TM to learn to x/t from RPN to normal Alg to Polish notations.

Remember: T. pt. of all of this was to reduce t. cjs $-\left(=\frac{cc}{pc}=L_{cost}^{-1}\right)$ of learning Alg-notation. I felt that Quantity was an impt conc. that was more generally useful: — say in explaining to TM what "solving an eq." means. — And this is why I want to "go over" a large part of t. TSQ, to ~~see~~ when certain concs are important in many places of t TSQ

.32 Perhaps "Built-in" concept: T. idea of "localization" ≡ $\boxed{\text{Subnet}}$ subnet
Another possy is that t. "idea of Quantity" is ~~to~~ inherently diffıt to teach.
~~Pre~~ So — (.32) — need to "build it in" somehow.

Or, I could just write out t. soln. that I want TM to acquire & go on from there. — In fact, it would be well for me to write a t.s.Q., write t. soln's I'd like TM to acquire, then look at t. CJS's involved, then see what I can do to reduce them.

What I much want is the CJS that seems small to a Human, is also Smaller to TM.

Thus it could be that t. reason certain CJS's are small for a Human is that ~~he has other~~ Ing instances that are not immediately apparent.

Re: ANL: Say I used RPN or / Consi notation w. parens. (like LISP): If TM learns *(3,4), *(7,5) etc, then t. ~~idea~~ idea that strings & parts of strings (subtrees) that have been much used in t. past, have hi pc for future use! This mite ~~be~~ an adequate heur for ANL. That t. idea of "Quantity" mite ~~be same~~ as t. "idea of" sub-tree. — which will (probably) be a built in heuristic.

SN If I use 1 example w. hi res. v.s. many examples at low res, ~~t. pc of t. resultant concept will end up at~~ t. same: ~~But~~ The pc of discovery of t. conc will be t. same ... but for combining w. other concs: — T. fact that a given conc. was used many times in t. past (even if w. low res examples) is important — it would seem to give it higher pc v.s. use w. 1 hi res exp. example in t. past.

Actually ANL can be expressed as a stack operator.
+ 4,3 → add 4,3. ; ~~first~~ (t. mechanics of this is unclear!)
Anyway, after it acquires t. operators for $\overset{*}{+}(4,3)$, it will be able t. acquire t. op for combinations by combining sub-operators. &

It may be that in my Sgrb approach, I used a common "Memory" for concs, & so I didn't really give them proper pc's — — In particular I didn't give them Conditional pcs (which functioned so as to narrow down their expected applicns & i.e. ↑ their pc's in a gn. applicn. (~ to idea of "directories")

Hvr., I'd like to avoid that much detail — but I do want to ~~be~~ sure I get t. PC's of t. concs rite, so they are indeed much more likely to be used as parts for candidate Concs

.31:169.35 1) On the ~~[illegible]~~ 10% or 1% pc of error at 169.32 – .36 : In general, our ability to get 1% certainty in predn. will depend on SSZ of examples used f. new Conc.

.34 2) In discovering new concs.: A (Human-type/-level) "soln" is not complete until I have some way to narrow down t. concs that were used in t. new cand conc. This can (perhaps usually will) involve categorizations of concs, and for each category, conditions in which its likely to be relevant. This (category/condition) set will be a heuristic & it will operate in a Bernoulli seq-like manner. (i.e. the way I code a Bern seq.) → The utility of a

8/3/00 Bulg

MAIN METHODOLOGY for TSQ WRITING .11 also see 171.34

.01: 171.40 Theory of this kind will depend on its efficacy in pc of past

.02 ——— Solns. of induction problems. → .11

.03: 170.12 : That ~~some~~ any thing must be one or the other but not both. These are "category" concs — the kind defined by Computational Lrng Theory ( ). ~~ftw~~ These concs are e.g. Number, v.s. not a number; even v.s. ~~not even~~ odd, etc.

[~~Tr~~ I use, hvr, normally, a more General defn. of a conc. It is a tree or sub tree. It is a part or whole of a computer pgm. [It is a string of op-codes.

Give special names to concs. That partition the universe into 2 parts like this: Call them P concs. (PARTITIONING concs) — They mite do > 2 partitions. [A "P conc." may correspond to an "OB"] cf. OB, OP Algebra.

Def

.11: .02 ← So: Re hours! ~~#~~ I want to first design the TSQ, & I have for each problem ~~tot~~ a Human heuristic soln. — So it seems reasonable that a Human could make the necy epistemological jump — so it seems like the CJS for Humans is acceptable/reasonable. Then I have the problem of converting (apparent) human hours into TM-type hours. If I have trouble w. this, it is clear that I don't completely understand the Human-hour!

.17 → 173.06

Put ref to this at end of work on GA

.19: 164.40 SN On GA: Design of filters/electronic Ckts could be much speeded up ~~by~~ by using RW. components ~~blah~~ (~~ex~~ RW ckts) for Cands. This is because (A) Spice is very slow (B) RW can implement things Spice wouldn't be able to think of. Puting components together could be done by relays at scaled-down speeds. For faster trials, we would have to find electronic ways to simulate these trials.

(NB) At present, there is research afoot to design ~~components~~ devices that can easily be switched from ~~being~~ being a computer say to a fm radio, to a GPS station, etc. — find out how this works — it would be great for GA work!

Another idea in this ~~thing~~ vein is use of Analog Signals on digital components ~ FGA'S w. analog output but digital switching.

Another possy is to use GA for Analog inputs as well as digital inputs to circuits.

It mite be expensive to have Variable L C R, ~~s~~ in ckts, But we can Make analog ckts that ~~convert~~ from voltage to L, C, R. ~~Appl~~ Connecting a C, say, of such a type betw. 2 pts, can be not so easy!

.37 ——— Maybe not so difclt! Say we want a cap betw. pts A & B. → 173.01

.01: 173.40: Hvr, if A < ϕ then A → A+1 will ↓ pc of $p^A(1-p)^B$ · which is wierd] [$\frac{p^{A+1}}{p^A} = p$]

Hvr, its true even if A > ϕ ! Perhaps its nogy for Norms.

.04: 173.28: 173.10→.28 seems to be a real difty — in t. sense that I will often have an induction problem in mind, so I think I know what it is, & I know what answer I want — But in fact, I really haven't thot out just what t. constraints on t. problem are.

This is Like t. "Robot's Dilemma" : T "FRAME Problem in A.I.". When Given a new problem in a sequence, t. Robot doesn't know what features have Changed & what ones have remained t. same.

In 173.10→.28 t. Q is: "Just what are t. constraints in each new problem?" It may be that this can be answered by a more **GLOBAL Induction** — But "in any way" There isn't much corpus to make "Global Induction" on. — So I have to be careful starting out:

Possl. troubles: ① I dont put in vital constraints so it fails: ② I inadvertently put in extra info, so it works, but it contains seeds of future failure.

.20 ANother Way to write (Human-ish) TSQ's: Make large set of Problems: For each Problem & separate

.21 write t. necy apr info needed as well as t. search procedure.

A prime difty (w.r. .20) is that we will not have any idea of how many & what assoc. pc's, there are for t. other (not "correct") concs., so we can't have estimate for cost of t. expected "soln". → On t. Other hand, we could make as part

.245 of t. "Problem soln" various hours to ↑ t. pc's, (for that particular problem) of t. concs used in t. soln. — This last is certainly an impt part of t. soln. of any largish problem.

→ Hvr, even w.r.(.245) .20-.21 could be a useful way to organize a tsq. 10:16 17 18

.31 Some small TNG Tasks:

1) Equality: 3 = 3 ; ~~cond 3 = 3 imp 3 = 3 ; cond 3 = 3 imp 3 = 4 low pc.~~

3 = 3 hype 3 = 4 low pc

2) introdn of cond, imp formalism: ↗ See 178.09 for discn. cond $\left(\begin{matrix}3=x\\ X=3\end{matrix}\right)$ imp $\left(\begin{matrix}3=x\\ x=3\end{matrix}\right)$ ← "cond, imp" defines Local v.s. Global ideas

cond X = 3, y = x imp y = $\boxed{3}$ {other problems involving transitivity of "="}

3) 3 + 4 = 7 || cond x = 3 imp x + 4 = $\boxed{7}$

4) 4 − 1 = 3 ||(cond X + 3 = 8 imp x = 8 − 3) ← Downward more TNG before this?

~~also cond X = 3 + 4 imp x~~ 4 − 4 = ϕ 3 − 3 = ϕ 3 + ϕ = 3 ; ϕ + 3 = 3 >>>

(175.01)

.01: 174.40 : Possl. Notation note) L, M, N are by processing random nos. ← 32 bits say

— So t. single example L=L is equive to t. examples [$L_i = L_i$] $i=1/2^{16}$ (for L_i's 8bit nos)

.02 For realization of a Lrng Algm; TM can take 2 general forms: ① In one it hunts for low cost operators → w. input string "N = ?", t. output will be "N."

② In t. second, It tries to find various Nos for "?" that will give entire corpus by pc.

— #2 is, perhaps, t. more general soln. to t. problem. Usually, #1, t. operator

.06 form, is insensitive to regularitys in t. "input" part of t. corpus.

For t. N=N corpus, t. N=? problem has one simple soln: Look at input string: find first number: this will be output.

For #2, we find regys in input: This could be useful for predicting long term trends in t. types of problems being Given. — it could be used to guess at types of future problems that will be Given — up to the future Limit of the "Horizon". This could help in t. long-term optimization of TM's Behavior.

What are t. MAIN IMMEDIATE PROBLEMS?

1) Choice betw. ① & ② Model types in .02 –.06

2) Choice of what parts of 174.31 –.40 to use, to start on

→ 3) Fear that Choice on ② would get me "stuck" in an "Non-English Level" Aspect of TM. — That I would be getting "Too Specific" for this phase of TSQ writing,

Perhaps for t. Probs of 174.31 –.401. For each problem, write out t. kind of behavior I'd like to be elicited from each ~~[illegible]~~ Material Concept. introduced.

In Some Cases, I don't know just which properties I want to introduce "at t. beginning": e.g. Commutativity of Addition? (probably)

.28 $X+X=2X$ (probably not: this involves [$X+X = 1\cdot X + 1\cdot X = (1+1)X = 2X$.] ← .28R

If TM learns X+X this way, will it really be able to "count" objects? Perhaps I could introduce t. counting process (later or ~~[illegible]~~ earlier or in a ~~[illegible]~~ "separate stream") & then have it somehow integrated w. ~~[illegible]~~ t. idea of (.28R).

For "counting" TM could do "j ← j+1" every time a new "X object" was found.

Also, I could introduce Multiplication via 3+3 = 2×3, 2×3 + 3 = 4×3, ~ to counting

I want T.M. to know all of these aspects of arithmetic operators.

i.e. to get sum of a bunch of 3's: count the no. of 3's, then 3× t. counter no.

For this trick to save any CC, "adding 1" must have << CC than "adding 3".

We can use "adding N" where N is a large binary no.

8/6/00 Bulg

(.21) HOW TM CODES NEW DATA
(Steady State ~~E~~ Learning)

Ramanujan: | Synopsis of Elementary Results in Pure and applied Mathematics. 2 vols (1880-1886) 6,000 forms
By George Shoobridge Carr (Maybe Camb. univ. Press) } up to 1860

.01 On t. Other hand Perhaps TM (as a "Curious" Scientist) ~~a~~ should get a lot of emotional Excitement from discovering these facts, rules, laws about Arithmetic. — A ~~Ramanujan~~ nujan - like Goal! Srinivasa Ramanujan 1887-1920

.04 Evaluating ~~Pla~~ t. "values" of these "discoveries" is dif'lt! Lenat's "AM" did it, hvr., in a not unreasonable way). Its evaln. rules { for "interestingness" would develop as it had hours for developing them.

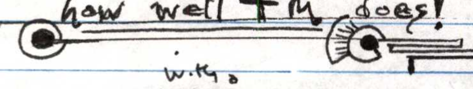
A TM w. ~~very~~ long TSQ would itself discover heuristic Rules for telling whether a conc. was "interesting" or not. For an infant TM, hvr, we will have to insert some of these ~~—~~ heuristic Rules

.10 "am anfang!"

Q: How far [illegible] can TM go w.o. ~~heuristic~~ Lenat's hours of .04 - .10? I'd sort of like to make a "Minimal TM" to

.13 Start off.

I could just put ~~in~~ a reasonable no. of concs int. corpus & see how well TM does!



[redacted] with a Suitable TSQ, I could use GA to find a population of low k cost. As new examples appeared, t. population would evolve to track t. new data (as well as ~~#~~ better its fit to t. old data). This mite be a good way to start TM.

.21 → Re: ↑ T. way I envision TM normally working: Say its looking for Operators that map input problem into output soln. We start, doing an Lsrch for an operator that solves t problems well. When we augment t. corpus, I expect that small / trial changes will be made in t. defn of t. operator (≡ "soln."). This is essentially pure mutation.

→ I think its about t. Same for Non "operator" inductions

.26 [Re: .21 In coding Sequencial Corpi, when t. corpus is Augmented,
.27 normally, t. code is ~~a~~ simply augmented w. minimal search needed.

.28 [Hvr, occasionally, when a new section of t. corpus comes in, a ~~large portion~~ part (sometimes large) of t. code must be rewritten. BACKTRACKING This can be done by using ~~another~~ parallel codes that were not t. shortest for t. "~~prev~~ post recent" corpus augmentation. (This is an aspect of "Theory Revision").
.31
.32 A ~~possibly~~ ~~re~~ real related technique is OSC. OSL is usually simpler than other kinds of "Theory revision"]

~~revision, de~~ deletion, revision, addition of ~~a~~ "Definitions"

In .26-.27 we add No New Definitions, but in .28-.32 (including OSC) we ~~add or modify~~ also add new defns a/o ~~d~~ (delete modify) old ones.

~~A~~ common type of Theory Revision, (of which OSC is an Extreme Case) occurs when t SSZ ~~in a corp~~ for a given ~~defn~~ definition in t existing ~~corpus~~ corpus is too small to warrant t. definition, but ~~augment~~ t. recent Augmentation of t. corpus ↑ t. SSZ for this defn, so it becomes Legit.

.39 Another kind of Simple "Theory Revision" Occures when we notice (177.01)

8/7/00 Bulg.

A/O we may break code down (or models) into smaller operators & try to reorganize it in a better (higher pc) way.

.01: 176.40 regularities in t. Code of t. corpus (so we write a shorter code of t. previous code).

.02 An indication that we need ~~[illegible]~~ "Serious Theory Revision" — that t. new corpus has many parts that cannot be predicted w. t. accuracy that we had obtained in t. previous corpus. ~~[illegible]~~ (— i.e. Our old theory doesn't seem to be working w. t. new data). In such a case, (if it is possl.) we should (try to) arrange to get more of t. aberrant-type data (= "experiment"), very very imp criterion!

This problem can occur in t. very beginning of t. TSQ! see (205.08)

.07 to ↑ its ssz — also try to get more diversity in that data. → (.20)

In 176.28 ff I talk about ~~[illegible]~~ (re)coding a "new section" (or ~~[illegible]~~ new ~~[illegible]~~ augmentation) of t. corpus. The corpus hvr, may not be so "sequencially" organized, So a "t. new section of t. corpus" could include "all problems involving triangles" rather than just "t. last 1000 bytes of t. corpus". Slightly more generally "All parts of t. corpus that used a particular defn. or set of defns". T. code for those parts

.15 has to be revised.

176.21 ff (& .26 ff in particular) Is a VERY HY LEVEL "ENGLISH" dscrn. of TM's operation! While its fairly good, it proby doesn't extend very far into t. varieties of TM behavior. Hvr 176.21 ff is very IMPT! — I do want to continue it.

Varieties of Coding techniques:

.20: (.07) Also note .02 – .07: Befor trying to get new data relevant to t. new phenomena: — we look at t. old corpus for "sections"/"parts" that are usefully similar to t. new intractable subcorpus. — This is usually not an easy thing to do.

Note: 176.21 ff is on Coding a Corpus ≡ [Induction; Prediction] (presumably sequencial, but I should think a corpus of unorderd finite objects would not change t. discussn. much (if any!).

.26 Hvr, TM's other usual probs are OZ probs & INV probs. INV problems can be solved via Lsrch (see [illegible] 166.21). Hvr, note that most INV probs are solved as OZ probs

T. MCT does link (sprobabilit Lsrch ~~to~~) Optimization (≈ OZ probs): Enables one (in theory) to give proby that a gn. O.T. will do best, w. a gn. OZ problem — So Lsrch can be done on OZ probs in a "practical" way.

Hvr., Note that t. MCT treatment of OZ is entirely Empirical: It looks at the G results of each OT on whatever corpus ~~[illegible]~~ was used and from this it ~~[illegible]~~ predicts t. likelyhood of any one of these OT's giving a max G of t. given OZ problem w. t. stated cc. limit. ~~[illegible]~~ MCT does not look at the structure of each OT to determine t. feasibility of ~~[illegible]~~ each OT for t. given OZ problem. So MCT's analysis would not suggest ways to design new, good O.T.'s Perhaps "TM₂" would do it?

178.01 to some extent But not much!

187.25 spec

8/7/00 Bugs:

$(1+i)^2 = 2i$ co ; ~~Imp~~

users.javanet.com/~ Bob mere co : imp
ElmoreElevators

Logical Negativism
Illogical Negativism

So, ritenow, [redacted] Major Problem areas:

1) Worry about the Learning Algm needed for General INV, OZ probs

177.26 –.40 discusses this abit. T. "General Process" of 176.21 seems Reasonable.

2) Concern about t. epist. details of t. tsq! What do I expect T.M. to assume about the examples in t. corpus. The "global" v.s. "local" variables, concs, defined by "cond, imp" for each problem! but there may be ambiguity. → .09 .06

3) Which models of induction: {175.02 or .06} to use.
(operator — unordered set of finite strings.)

.09 .06 In t. cond probs w. {cond, Imp}, we want certain parts of t. lrng. to be global, yet certain ~~[illegible]~~ info applies to one problem only. (= "local")

In mathical langs like Maple, Mathematica, Mathcad... there are [redacted] tricks they [redacted] use to make things unambiguous.

equality : 3=3 4=? | cond x=3 imp 3=x (sym) | cond x=y, y=z imp x=z (trans.)

Addition: 3+4 = 4+3; (Assoc?: (3+4)+5 = 3+(4+5)) Advanced?

Subtraction: 3+φ=3 ; x+φ = φ+x=x ; 3−3=φ ; x−x=0 cond x=3 imp x−3=φ 3−x=φ

3−5 ≠ 5−3 3−5 = −2 ; 5−3=2 ; 2 ≠ −2. This idea of 2, −2 is unclear. N ≠ −N unless N=φ.

Well: I can teach these relatively A.H. things about t. fund. concs of Algebra!

Will TM be able to induce/deduce certain other interesting things)

Perhaps t. main goal of t. Learning of Algebra is ~~to [illegible]~~ co. for TM to acquire a complex enuf domain of knowledge, so that I can [redacted] get it to ~~lrn~~ usefully lrn to understand English S's about that domain, & get it to genz. English to other domains.

.27 Also, to be able to deal w. "English" that has errors in it.

.28 Perhaps write down a large bunch of things about =, +, −, φ, {co, imp}, x that I want TM to Learn; so that I can order the examples so it can learn in a useful way.

Then, try to see if TM can "lrn." it. If there is trouble, return to .28

Don't get too deeply involved. At present, t. problem is mainly on t. adequacy of t. info in t. TSQ, and not on just how TM "lrns".

So try juggling around/back & forth. Trng Sets & modifns of t. lrng. Algm, & a priori ideas for lrng.

Re: T. "lrng Algm": One impt. part of it tells what parts of t. corpus to test when a Modifn. of t. Coding algm. is made. ~~But~~ One kind of "Hint" could ~~to~~ help TM with this problem. (A "Hint" is any kind of info that ↓ cts for t. "student"!

8/11/00 Bulg

.01: 181.40 : I should outline a bit more on 181.33, .37! Some sample problems.

cond $m \overset{\div}{+} n = [m \overset{\div}{+} n]$ imp ~~[illegible]~~ $m \overset{\div}{+} m = ?$

$m - m = \phi$; $m \div m = 1$!

$m + \phi = m$; $m - \phi = m$; $m * 1 = m$; $m \div 1 = m$

After these have been learned; TM should be able to quickly "learn" to work

$m + n - n = m$; $m * n /$ ~~m~~ $= m$ (in last, there is trouble if $\underline{m = 0}$ ε)!

Associative law could save lots of time (w. commutative law)!

.08 $(m+n) - m = (n+m) - m = n + (m-m) = n$.

.09 How, TM has to find ways to do this reasoning fast otherwise it doesn't "pay" —

.10 — (Say TM had to do a large srch to find these xforms in .08.) → (.20)

One more trick: If one has a seq. of "+" ~~oper~~ or "*" operations, omit the parens, because one can put them back in an arb. way.

Also: ~~[illegible]~~ a string of (÷a), (*b) type operations can be ~~[illegible]~~ rearranged in alpha by order because of commutation & assoc. [

Similarly w. a string of (+a), (−b) type operations.

Some rules, however: [It may be necy to insert a leading φ in an +, − string
" " " " " " " 1 " " *, ÷ string

$$3 + 4 - 7 - 3 = \phi - 7 + 4 - 3 + 3 = 0 - 7 + 4 = 4 - 7$$
$$3 * 4 / 7 / 3 = 1 / 7 * 4 / 3 * 3 = 1 / 7 * 4 = 4/7.$$
} "calculator Arithmetic".

.20: (.10) So this sort of thing motivates TM to find fast ways to do searches

.21 & any other logical processes.

.23 A Third way to get TM to discover "Laws of Alg" would be in the direction of what I did in SAARB on this problem: That T. laws are useful in solving (linear) equs. I may be able to do this better, now, using MCT!

.27 So 3 ways (1). T. method of c:\TSQ1 (2) 182.01 −.21 (speed in calcn) (3) 182.23 (use in solving linear equs.)

N.B. In solving equs, like X + 91 = 43, TM could ~~to~~ simply try integers to find x; But if we make addition expensive, it will be motivated to find faster ways to solve linear eqns

.32 **HEURISTICS** can be a ~~any~~ fact (usually problistic), technique devn, ... it can be computed locally (devised for a particular problem): it may involve much "logical" reasoning to derive a/o apply it in a particular situation. OSL could be important: since the solution-info that is relevant may have only occurred ~~Onc~~ Once!

For OSL (& other applicns) (indexing / Categorizn) is very impt. When an event occurs, we want to be able to recall it as soon as possl. when an associated (perhaps relevant) event occurs.

.01 This idea of "association indexing" is very impt., but there are probly very many difrnt methods used in Human "indexing" for (relevant) recall.
One very impt. way that TM "Matures" is in its learning of many new important, original, idiosyncratic methods to categorize situations events prob.solving methods for retrieval. situations

In General, every time one solves a problem, or ~~[illegible]~~ gets of impt. info from
.07 an experience (this can be during t. soln. of a problem), Also see top of p 181 one indexes that event in Many ways. "Many" is impt. because (a) I want it to be accessed in many difrnt "similarity dimensions" (b) If t. present New event is similar in enuf "dimensions" to an event in t. past, it will have enuf "code bits", "description bits" in common w. that event to "qualify" for OSL.

How to discover these "Categories": When I am working t. problems that TM is working on, observe t. associations that I make & try to ~~[illegible]~~ devise categories that would ~~[illegible]~~ help in that (as well as other) useful associations.

Its not clear that the foreg. stuff will be very useful for infant TM — & early TSQ's. I could just assume large IPC available: Say 10^{12} ops/sec. Later, when t. "scaling problem" becomes serious, I will certainly need "categories" to narrow down search.

.18 So: This is an impt. idea: Categories are hard to devise for early TM work, but will not be very necy. Later, hvr, there will be very many choices of concs to combine & categories (& other heurs) will be needed... but they should be available (from data a/o introspection) at that time..

Another Aspect of Categories & heuristics whenever they are used, they not only ↓ srch time, but they narrow the Proby distribn. for t. Answer. For MTM this may not be critical — but in NMTM it can be very useful

The Category/heur idea is Very Impt. in CREATIVE solns to Probs. A good set of Categories/heurs will enable rapid solns. of otherwise unsolvable probs. Hvr, it will ↓ diversity (unless it is a v.g. set of heurs). A novice researcher will have fewer heurs & could have more diversity in search. Search will take much longer (than w. "Good" heurs), but could yield a very "Creative", "Unexpected" soln. — could be much better than t. more rapid heur. srch.
The Methods of Conventional (not super GA) GA may be an example of
.33 a slow, hy diversity srch, that can find V.G. Solns.

176.21 ff is a reasonable way for TM to Operate for Induction problems. For a seq. of INU problems, we solve early probs. w. simple Lsrch, using our Apriori that we insert into our ~~[illegible]~~ Reference umc. As we solve more problems, we add constructing concs, to t. (UMC/pd) & we change pc. of older concs. ~~Occasionally we~~ New problems ~~[illegible]~~ usually involve relatively small incremental Modifns. of t. pd. Occasionally, like 184.01

8.12.00 Bldg.

[86]
[95] is on GA:
(Mutation & Recombination)
& "Creativity"

.01: 183.40 on 176.28, we will have to "backtrack" &, and redo t. recent string of concs. in a better way. ~~Just when~~ Under just what Conds, we decide to do this, & just how we do it, is unclear ~~[illegible]~~ 176.39 ff has some ideas on "Theory Revision". 177.02–.15 is of particular interest — it can deal w. "Non-Sequential" data.

.06 On 176.21 ff I have this sort of "System" by which TM works:

.07 Contrast this w. MCT analyses, in which I use at all times a "Conditional" [redacted] Pd — with each new problem/ deem being the "condition" on t pd to be used in Learning that problem.

Well, MCT ~~this~~ isn't really disturbed by/ .07: MCT simply Gives a way to ~~[illegible]~~ take a corpus that consists of several kinds of probs. (Induction, Inv, OZ) & obtain a pd from that corpus. that can be usd to get predns/pd's for any aspect of that corpus. .... I need to get t. Q of .06 resolvd!

I think MCT & .06 are really "orthogonal" They are on different parts of t. TM problem, & they don't really interact. MCT ~~[illegible]~~ tells what needs to be coded — what codes need to be minzd. .06 tells a heuristic way for minimizing that code. ||| MCT defines t. problem; .06 tells an approx. way to solve t. problem..
.15

.21 On t. Q of whether to model using a ① "Stoch operator" or use the ② "Finite unordered set" model. Actually, t. 2nd model has t. same final form as the first, but uses more info — it uses info about regg & in t. {input set to t. stoch operator}. It may be that in both cases we end up w. a stoch operator, but that t. Gen ~~[illegible]~~ over t. universe of
.26 such operators [illegible] differnt in t. 2 cases.

.01: 184.40 On t. "Learning" part of TM!

Look at TSQ1! (181.5)!

n=n (a set of numbers w "=" connecting them!

The problem is to make a "program" of primitive concs, so that

.05 input n = gives n as output.

[I will desc. what TM does in English; Then later isolate out the primitives needed.]

.08 ① In t. case of .05 a soln. must be: — Since output is a number, t. final operator must have a no. as output. (useful in "backward chaining")

The only way to get n^{output}, is from n^{input}, so t. soln is to take t. number in t. input & present it as output. So t. pgm picks t. no. on t. input & transfers it to output. (or renames it as output)

.13 ② try 3.7+4.1 = 7.8 ... etc. a long seq. of examples like this, or a few hyp precision examples.

Again output is no. so final operator must map to nos, +, −, ×, ÷ or transfers.

Transfers don't work: + − × ÷ all have 2 ~~inputs~~ num. inputs, 1 num output.

try all 8 possys: m+n = [m+n] & m+n = [n+m] both work o.k.

.17 ? ⇒ Hvr. .08 no longer works on t. new examples. d? ← (what is meaning of this remark?) → (204.24) ~~it means that this system can't do "n = n" correctly~~

— SN) QS: ① How long would it take to lrn these things w.o. t. Heurs? —

② Under what circumstances would it lrn ~~[illegible]~~ these heurs itself?

When I put a Heur into TM! Imagine a TSQ that would result in TM (learning ~~acquiring~~) that Heur, & put it into TM in t. form that it would have lrnd it.

I think this will keep TM's structure very simple! Easy to pgm, Easy to Understand, (Easy to Debug. ?)

.25: (176.21 177.40) A Kind of lrng Model I've had in mind (in linew. 176.21 – 177.40)

This "Concept net": No feed back in net; No loops in net, but "loops" are a type of **Functional** that is used to combine functions to obtain a new function.

A Net of this kind is probly limited to ~~Primitive Rec. functs.~~ — This is not ordinarily a serious limitation (I Think!)

L2005 — probly Total Rec. functs w.o. diffy

The language assigns pc's to subnets of functions. At any ptim time, the system is a Bernoulli seq. of t. ~~[illegible]~~ functions (≡ ~~primitive~~ primitives + subnets that have been defined) —

.32 — a Composition of these functions.

.33 We also have a lang. of this sort for not "operators" but "observers".

The combin of t. 2 is t. ob-op algebra. An ob is a function w. output of ~~[illegible]~~ T/F or a Pd on T/F (≡ Boolean).

ob: Boolean, numerical or string input; Boolean output.

op: " " " " : Boolean or numeric output (no Boolean output).

ob, ob → ob; op, op → op; ob, op → op

The Boolean output controls which ~~[illegible]~~ of several functs is to be used.

I think one of t. main functions of t. OP-OP Algebra was to get good Cond'l PC's — to deal w. t. "Scaling" problem! 228.36

8.16.00 Bulg.

.01 | 187.40: This "Selection" functional of t. obs. is mindful of certain primitive functions used in Rec. func. Theory. — So it may be possl. to combine t. ob, ops into something ~~[illegible]~~ near "Universal". — But more important — t. "Algebra" ~~[illegible]~~ may be adequate to express all of t. kinds of "regularities" in a corpus (including Hours) that I want to express.

T. way induction is done using t. func. nət: We normally try operators in pc order (w. o. using the loop functional). Occasionally we notice function that involves several repetitions of a function: We then try to find a ~~[illegible]~~ higher pc ~~version~~ version of it ~~[illegible]~~ using t. "Loop functional"

.10 The loop functional is a combination of a ob and an op. The op is used repeatedly; the ob looks at some result of t. op & tells it. repetition when to stop. The op is usually a vector function: w. 2 components.

.13 [redacted] One component is used for stop criterion. This is an "Until" loop. [Also, one need initialization of both vector components] One very common component of the vector argument ~~is~~ is ~~[illegible]~~ initialized with x=0; and updated with x←x+1. The threshold for termination of t. loop ~~is~~ is n & t. loop terminates when x = n (an ~~[illegible]~~ "ob" function).

.17 It is probably possl. to devise functionals that could define functions beyond Prim Rec. Functs, using ~~[illegible]~~ a formalism ~ to .10→13, but I will not go into this just now.

So, using t. 2.141 model for assigning pc's to new funcs; (the details of coding the [redacted] subtrees will have to be worked out) and t. ideas of 187.25 – 188.17: Is this enuf for a more or less, a complete model of "practical learning"? [It solves (or is supposed to solve) t. "Symbolic Regression" problem

What it does seem to do is give a model for learning (functions/operators / stochastic operators). This is a Very Large area of induction: it means that we show TM examples of desired I/O behavior & it extrapolates,

.27 probabilistically. → 189.01 spec

.28 ~~[illegible]~~ while its not really solving ~~[illegible]~~ INV or OZ problems: ~~[illegible]~~ It could "Learn" to do so. Say we label all Inv. problems; then gave "solns" (i.e. traces) for solving them. From many examples of this kind {"Building up": (starting w. simple inv. problems then building up toward more dif'lt INV probs.)}, TM builds up a good pd to use for t. standard ⌊such soln of INV problems.

Mixed Corpus Thrm.

Similarly, OZ prob. soln. could be taught.

On t. other hand, once TM can do "Induction" (i.e. 187.25 – 188.17) TM can use MCT to ~~[illegible]~~ update t. ~~[illegible]~~ cpd's needed to for ~~[illegible]~~ ⌊such as Inv & OZ probs.

(SN): Perhaps I/now have a "Critical Mass" of ideas for TM: That for any TM sub-problem that arrives, I have some ideas (that I've developed in some detail) that are good enuf to solve that TM sub-problem.

189.01

8.17.00 Bulg

4:32 pm after ~ start.

188.40

.01: 188.27 So; OK: Then — so ~~z~~ Z141 + 187.25—188.17 does give adequate model for learning stochastic Operators? (≡ functions). Consider also, lrng. of Neurs [redacted].

[Dropping, for t. Moment, t. Q of (or t. details of!) 184.21—.26]

One problem is t. Ordering of Function Trials for Lsrch: See 130.19 ff on how to count t. ~~[illegible]~~ no of operators using n functions of a different type, w. the i^{th} function type having $c(i)$ inputs (& 1 output)

136.21 tells how to construct them in BASIC

One diffy is that there will be many duplications of functions in the listing of 130.25ff! (see 131.11 ff for eqns). This is because the list of functions to be used, will contain compositions of some of the primitive functs. If we list them in pc order, the hyer pc versions (not come first) will contain t. composite form of t. primitives. — So we will get dups, but later in t. listing. — This is t. usual diffy of Lsrch being controlled by (≈ K cost) rather than ρ cost. My impression is that usually this doesn't ↑ cost of Lsrch by very much. (perhaps a tolerable factor of doubling!)

"Symbolic Regression".

In general, the trial operator will not contain more than 3 or 4 functions, because t. pc ↓ so fast as t. no. of functs considered ↑!

To order t. Macro functions: Assign to each funct a pc =

.17 $\frac{pc_i}{d^{c(i)}}$; pc_i is t. pc of drawing t. i^{th} function. d is t. no. of inputs to t. Macro function

$c(i)$ is the no of input of t. i^{th} (micro) function.

Actually, t. factor $(d+n)^{c(i)}$ should be used, n being the order in which the micro function is chosen — but assume that $d >> n$ (n will be 1 or 2 or 3) and usually d will be > that. (I Guess!)

Anyway, use .17 for the effective pc of t. i^{th} function: then order the listings of t. functions via Huffman (142.01ff)

Normally, there will be a ~~large~~ long list of micro functions to be combined. Also "Normally" there will be heuristics that Modify t. pc's of t. μ functions, in line w. t. "Nature of t. present problem".

So: T. induction process for "Symbolic Regression" consists of: trying to find the "Best" Macrofunction to fit t. data: We do this by first assigning pc's to t. component functs to be combined: we then order strings

.30 of them wrt. $\frac{pc_i}{d^{c(i)}}$ (.17) using t. Huffman Code technique. [The pc's of t. trial strings are first Modified by any available heuristics.]

We test t. ordered strings of .30 using Lsrch.

We usually ~~[illegible]~~ find a good fit:

The Function grows along w. t. corpus — just as t. defns. in t. "Procorpus" grow in Z141. Hvr, t. difference is, that in t. case of "Symbolic Regression" t. only Functions that ever get defined are ones that are, for a certain length of t. corpus, complete Macro functions for that chunk of corpus.

I ran into this trouble w. t. TSQ's I wrote in SAARB, but 190.01

8.18.00 Bldg

ABCDE

.01: 189.40 I thot that this was a fault of t. TSQ's! — But it was not (at least not entirely) because t. Induction model used Implied this "Fault"!

One diffy seems to be, that there is only one Macro function: So only 1 "conc. not." If a particular ~~[illegible]~~ primative or defined Funct. is used several times in t. not., I think it automatically get higher pc. Hvr, I may be possl. to examin t. Macro Funct & find subtrees that are used ≥ 1 time, so it pays to define them. This amounts to a Recoding of t. Macro funct (≡ conc. not)

I'm not sure t. Conc not & t. Macro funct. are really t. Same. They are not identical, but t. ordering of concs & of function definitions have (1 to 1) ⇔ correspondence.

.14 A vague idea: That somehow t. pc of a micro funct should be related to how often it is used in coding t. corpus.

Another idea: That t. pc's of micro functs should be related to t. frequency with which they were successful ~~in devsn~~ in helping to devise (temporarily) successful Macro Functs.

Another "idea": we have obs that switch ops on & off. perhaps t. fraction of t. time that a funct is switch on is impt.

I feel uneasy about the "statistics" of t. components of a small Macro funct. In a stoch PSG: the frequency with which various paths are used, determines their pc's. Hvr, in MTM, this may present a problem. None-t.-less, even in MTM, I will have a SSZ for each "decision" — even tho it's always ~~[illegible]~~ "like" $\binom{13/0}{0/10}$.

Also, in linear regressn, t. ~~[illegible]~~ values & precision of t. coefs are critically ~~dependent~~ on ~~SSZ~~ SSZ.

Hvr, the functional forms for linear (or nonlinear) regressn: ~~[illegible]~~ what is t. SSZ for those? Well one ~~[illegible]~~ choses a functional form for each time series predicted, So SSZ = no. of Time Series predicted

What is SSZ of "components" that help devb functional forms for regression? Say we used many difrnt functional forms for regressn. Then t. SSZ of t. components of these functional forms, would be clearly defined. Hvr, suppose we ended up with 5 difrnt forms for regression & we used them all t. time hence forth. Then t. max SSZ ~~of~~ for any component function would be ~5 only!

So: Consider LARGE, MCT-type, system, w. many → 191.17 spec difrnt kinds of problems / modes of behavior {Inv, OZ, induction of Time series / unordered objects, Heuristics

8.18.00 Bulg.

abcdefg
ABCDEFG abcdefg.

abcd

(size!)
kind

It may be that my "intuition" is more oriented toward a system of that kind. This is an impt problem: In particular, t. Q of SSZ for Heuristics is Impt

In Z141, (w. n to p defn for n ≥ 2), an ngm. can be usd either for prediction or for defn. of another ngm. (In general, t. costs of t. 2 applicns can be difrnt, but they have been t. same in t. systems I'v analysd.)

In t. SAARB TSQ work, each function defined was defined once, usd in predn once & could be used ~~as part~~ to help define another function, several times

In Z141; each ngm is defined once, can be usd several times to define other ngms; can be usd many times as a predicted object.

SAARB & Z141 differ in that in Z141 t. ngms were both ⓐ part of t. corpus ⓑ objects that could be usd to define more ngms. 69

In Saarb: t. functions were used to define other functions; which could be part of t. ~~[illegible]~~ One Macrofunction, that did t. prediction.

.17: (ospace 90.40) In MCT prediction, each subcorpus usually has its own prediction function. Sub corpi can be: difrnt time series, ~~difrnt~~ & in t. case of ~~unordered~~ objects predn, each set will have a difrnt stoch lang. : Each OZ problem will have its own wtd set of OT's. Hvr, t. total of all predictors are able to share concepts & subconcepts. We want to minimize t. total bcost of the

.23 Entire prediction "object". A later development will be the integration of t. various sub-systems, ~~by sha~~ — at first by sharing Functions — later by a unified system that modified itself in response to t. nature of t. problem(s) given to it.

Say we had a "Unified System" like (.23). How would we (modify/improve) it? Well, suppose t. system was a person, & was usd to run his life. Part of t. System would deal w. Physics. So we have these "laws of physics": we observe some new phenomena! "The error of prediction is much larger than we expected" (I don't exactly know what that means!) But anyway, we decide to try to revise "t. laws of physics" We do this in t. "standard ways", try back tracking as little as possl. & trying to modify t. part of "t. Laws of Physics" that seem most relevant. If we are successful, almost all of t. laws will be about t. same, & t. rest of t. "human response operator" will not change. (later we may try to integrate t. "modified laws of physics" into t. rest of the "human response function")

Perhaps Go into more detail on "Theory Revision". How it is done! How PC's are assigned to concs.

A common way to Learn: Each sub corpus has a particular

8-19-00 Bugs.

80×64 =
5120 = 5M

[redacted]

90M
10
40M''
~~64~~
16 = 600k'' = 70 days,

Method of solving its problems: Say a function (≡"op"). Also we have a way [redacted] (OB) to recognize that sub corpus, so we can apply this Op to it.
The main process, then, is developing obs & ops for various sub corpi.
After we have lots of sub-corpi in assoc obs, obs; we try to integrate all of this, by finding similaritys in obs, ops, for difrnt sub corpi
These similaritys can be due to "sameness" in much of t. functions involved or it could be due to "Analogy" xform mapping from one to t. other.
Think of t. development of Sci & Theory revision: Do this in English.
Get forms of heuristics. Get algms for assignment of pc.
Consider t. GRUE problem-type

[Hvr, I should be able to do same Analysis on t. Elementary Algebra TSQ directly! — Perhaps do both in ||.]

.14 One kind of activity in a (scientists) person's life is Mathematics! — So if he is working on a general theories for physics, he will use mathematical formulas & formalisms, ect, that he is interested in! Math & Physics can be synergistic! So t. idea of applying a xfm to physics is much hyer if a xfm has been much use in Math!

.19 In general cross fertilizations of difrnt fields of interest ("Domains") are very [redacted] impt in determining pc's, in guiding heuristic srch.
Perhaps this means that certain concs. that would ordinarily only appear in t. dom. of a sub corpus, could appear in t. [redacted] sub-corpus of a difrnt domain.

.24 Also when t. sub-Corpus is Algebra (& other Math), there will be definitions of objects occuring in t. sub-corpus, that are also useful in t. Macro Function for Math (as well as in other Domains)

.27 So, Math is a particularly good thing to do induction on, for this (.19, .24) reason → 200,32

.28 While .19 ff seems very true, none-the-less, it seems unreasonable that I shouldn't be able to get more (hyer) ssz (conformability) of ~~a~~ concs. that
.30 are mainly useful in t. Macro Function.

.28-.30 seems like t. present main Bottleneck!

I'm not at all clear on what objection .28-.30 is!

.33 Another pt. related to .28-.30, that seems strange: Say concs A & B are used to define C, & are not used much after that, but C is used alot.
A & B get no more ssz! — even tho they are part of C! — This may be ok.
A & B contribute to t. Apriori pc of C, but after C gets large ssz, that apriori becomes much less important → (But still A & B are "Part of" C!)

Re: .33 ff: if C was definable in several alternative ways would we want C's hy pc "fed back" in a "Bayesian" allocation to its "definers"? → 193.01

8·20·00 Bulb

Platinum less wts sieve.

$10/8.

= 1 pint

100h = 10kg = 20lbs.

[redacted]

.01: 192.40: On t. other hand, if we are mainly using these concs in definitions & they don't have a large ssz in ~~[illegible]~~ that capacity, then
.03 they get low pc ~~&~~ as definition components. (I may or may not want to ~~[illegible]~~ pool data from defns & from sub corpi — whether I do so, depends on ssz — I can do both & wt.. of which error is back, will depend on ssz
.06 in both "defns" section & "sub-corpus" section. → .30

Reasons for publishing.
1) Esteem ~~[illegible]~~ of Colleagues.
2) Need of money
3) Feedback
4) Entreaty by Colleagues!
5) Good to summarize ideas. Put them in [illegible] process form

In t. SAARB TSQ's: I did note that t. only concs defined were those ~~[illegible]~~ that were solns. to problems. I ~~[illegible]~~ more recently "decided" that this was not a fault of t. TSQ, but t. method I used for induction implied that this would occur. ≠ — Hrr, it may not be necy!

① Often t. Macro Op (Funct) is in sections so t. soln. of a problem could, essentially, just add a new "section" on to t. Macro Funct.

.14 ② Secondly: After we have this Macro-Funct; we could partly or completely decompose it & re code it in a difrnt way to ↑ its PC in terms of functs that have been used in other parts of t. "Universe" — ~~&~~ as parts of Macro Funct a/o Parts of ~~[illegible]~~ sub-corpi.

~~[illegible]~~ One point of t. re coding in .14: Mainly to ~~[illegible]~~ find ~~[illegible]~~ concs. of hy pc that are used frequently/effectively. ← (?) ... (.19 is more to t. point!)

.19 Also: / Main point. Re coding gives us shorter codes & ~~[illegible]~~ less expected future error in predictions.

I think I've lost sight of .19 as a Major Goal (or sub-Goal).

I had been focusing on t. idea of inventing new Concs & verifying their legitimacies. — — But this is always secondary to t. goal of .19. — Getting Max total pc of codes [redacted] w. t. available cc.

So: what about t. difty of 192.28-.30? Actually, look at 192.33-.40

If A & B were only used once to define a useful conc, C, then they are, indeed, not very useful for defining new concs, & they should get low pc for that applicn. — 193.01-.06 is certainly relevant (& correct). Note .03-.06
.29 on pooling of data from 2 difrnt applicns. of t. defn. of a conc. →

.30: (.06) A Possl example of such "Pooling": we teach TM a certain conc, by giving it ~~examples~~ examples so it can tell if an instance given is a case of t. conc, or not. (This uses "conc$_1$" in t. sense of Computational lrng Theory ("Valiant")) This defn. then is in t. sub corpus. We then get TM to use this data ~~[illegible]~~ either as a predictive conc &/o as part of t. defn of a new predictive ~~conc~~, conc. I had planned to use this method of lrning very often in early (at least) TSQ's. T. concs lrnd, would be "Terminology" Terms of ~~[illegible]~~ apparent importance in Math, Algebra (& perhaps Physics)

Def. Conc1 is
All conc1
are concs.
All concs need
not be conc1's, how

So T. Q. is, in view of T. "apparent" soln. of 192.28-.30: (on 193.19 — .40), can I start to write a complete dscn of (t. initial) "Lrng Algm"?
It will be pretty much like ~~a~~ T. SAARB TSQ, except that it will be vaguer (In English) — i.e. More "General"! Also, I will do much training on impt "$concs_1$" of Math, Alg.
(Also perhaps simply concs of Math, Alg.).

Try to give Examples of the Varieties of Lrng that can be done w. this method — specifically try to go beyond the Saarb TSQ examples. (tho some of t. examples not "fleshed-in" much at Saarb, might be worth going into now. E.g. solns of linear → quad → cubic
.10 (& maybe 4th °) — & maybe higher, using properties of Eliptic functs)

Write a "review" of [redacted] 163.01ff.
−2 = 16½ pp 2? pp − 1½ = 13½ pp 7 pp + 1½ = 8½ pp +2 = 10½ pp
2 parts: ① 163.01 — 184.40 ; ② 187.01 — 193.40 ② includes 176.21 — 177.01 — .40
Part ① is on TSQ in General part 2 is on t. learning Algm.
While Part ② would seem to be a self consistent "unit" apart from Part 1: this is not so: There is much in part 1, that is impt rel t. lrng Algm: e.g. 182.32 — 184.19 (~impt); on "Heuristics" "Categorization"
.17 so section ② is 176.21 — 177.40 + 182.32 — 184.19 ; + 187.01 — 193.40

Section ① is 1½ of section 2 in pp. size ① = 16½ pp ② = 10½ pp

list of "problem" types for t. TSQ.

176.21 — 177.40 is a good picture of t. General operation of t. "learning part" of t. TSQ. 181.06 — .19 (Also Top of P. 181) (<181.19 R> is impt.!) Discusses t. "Scaling problem" (& 181.19R Gives a kind of Criterion for Soln. — its a very impt. problem!)))
A long discn. of Heurs & Scaling 181, 182.32 — 183.33

There are << 10 impt. ideas in .17 refs: I should be able to list them & dscrb. them "Briefly". [later, make more detaile dscn as a kind of PAPER! for Myself

1) T. ~~main~~ picture of TM operation of 176.21 — 177.40
2) Heuristics & "Scaling": 181.06-19 (top of P181) (181.19R)! 182.32-183.33
.29 3) The diffy of 192.28-.30 , 192.33 — 193.06 , 193.30 — 194.10
190.14 — .40, 191.17 — .40 — 192.01 ffff →
This involves pc's of concs (A&B) used to define (once) a very useful conc. (C); C is used a lot
A & B are not used again! Should A&B have hy pc?
.32 4) In t SAARB TSQ's; T. only abss (= concs) discovered, were solns. to problems. How bad is this? Can it be avoided?
5) The idea of Global v.s. local references: T. invention of cond, imp formalism
6)

01:194.40 SN In doing TSQ's, The large ~~availble~~ available HDD's (~ 40 Gby) will make it easy to store TM's ~~state~~ after each TSQ session. This makes it possl. to "Backtrack" if I ~~suspe~~ suspect that I've led TM down an inappropriate training path.

.02

.04 I can also try sections of t. TSQ in various orders.

SN Make lists of types of Problems for TM; first t. broad list: [Induction of ⓐ Time series ⓑ unordered sets; Inv probs; OZ probs.]

Then Alg, Geometry, Physics, Chemistry!... Then perhaps more specific problems in these areas:

Then: Simple induction & "learning" definitions in Math by example:

Various Math problems: Easescalated by "Hints": (I can study t. effects of "Hints" by backtracking (.02-.04) & redoing t. problem.

The "Advice channel" (McCarthy paper): Related to "Hints".

"Telling" TM soln. — This is an extreme variant of "Advice" or "Hint". I can either have special 'channel' for this, or simply insert t. soln. into TM by suitable pgmg & reasonable pc's of component concs.

Perhaps use term "Abs = Abstraction" instead of "Concept" — use Concept for valid meaning in computational lrng theory

"STUDY" Problems: These are given to familiarize a person w.a particular domain, so he may ~~learn~~ discover relevant concs. Often this is done w.o. t. teacher knowing (concisely) what t. "relevant concs" are! The "Study" problems" may be given as a kind of "Hint" or as a 'Hope by t. teacher (who may be = t. student), that ~~the~~ these probs "may" yield needed insight. STUDY Probs can be used in 2 ~~general~~ extreme ways ① T. teacher knows what concs are needed & knows t. Study problems "contain" them ② T. teacher doesn't exactly know what concs are needed, but suspects t. Study probs "will" have useful concs in them. [Various Mixtures of these 2 extremes]

SN Would it be possl. to do useful TSQ/TM development (at t. Beginning) using very small CJS problems (almost exclusively). So it would take < 10^4 for a soln. (maybe < 2^a for a soln!) ? T. Q is: Could I investigate/demonstrate all of t. important useful ideas using small cjs.? Using, say a 1 MHz machine & machine code, this small cjs could be ~ 10^7 or 10^8 trials. — which is not so small! For t. SAARB TSQ examples, I think cjs trials was "very large" — & I expect that these problems should have been ~~de~~ solved much faster using good heurs — designed ~~[illegible]~~ to facilitate "scaling"."

$43\frac{1}{3} \times 1.25 =$

$5 + \frac{100}{3} \times \frac{5}{4}$ $\frac{80}{3}$

$\frac{100 \times 5}{3 \times 4}$ $\frac{125}{3} = 42$

$= 542$

SN APL has a v.g. set of "functions". — So that combinations of them tend to be "interesting" (to Mathians!). Could I use a subset of them as "Primatives" for learning certain areas of MATH? □

197.01

198.01 spec

8.23.00 Bulo

.011196.40 So: Go thru 163.01 ff: See how all of this fits into 6. Learning Algm of 176.21 ff

8.23.00 Bulg.

.01: 196.40 SN An aspect of t. TSQ That I hadn't recently much Considered.

.02 It is t. Soln. of INV probs.: e.g. Proofs of Alg. Theorems.

My thots about building up good abss from simpler subabss stem mainly, perhaps from Considering problems of this kind. I would use certain soln. techniques many times in diffnt problems. ~~[illegible]~~ A particular soln technique would acquire a hy conditional pc for problem in which it was appropriate.

This seems much diffnt from ordinary induction on an unordered seq. — which is what 176.21 ff is oriented toward — & t. probs of 194.29, .32 arrise in. They may not arrise in a INV problems corpus.

.10 In ~~a continuum of~~ t. SAAR TSQ — I could start w. simple induction of defns & simple Alg. notation. The solution of linear or higher order eqns., could be formulated ~~regarded~~ as induction problems, INV probs or OZ probs.

An impt ~~a~~ favorable feature of induction, is that its mainly what an infant (Machine or human) does in t. early part of its life.

Hvr, one may also view t. Infant TM as Schmidhuber: that its trying to maximize its mean Gain — & that it uses induction as a main Tool toward this Goal.

Perhaps t. Main thrust of t. present work is to get TM "started" — then, using MCT, add various kinds of problems to its Repertoire.

One very early idea I had about TM_2: That it wouldn't be of much Value until TM had accumulated enuf experience to successfully work on OZ problems of that kind. This idea can be generalized: i.e. we can't expect TM to do anything very clever, until it has had (usually as part of corpus) adequate experience in t. ~~[illegible]~~ domain ~~[illegible]~~ according "Cleverness"

I think what I want now, is to set it up so I can ~~[illegible]~~ give any kinds of problems to TM, & it would have ideas on how to solve them. — Induction in unordered sets seems like a good place to start, but it seems to have characteristics of 194.29, .32 — which are somewhat idiosyncratic!

Hvr, I could just do t. induction now, to start off. It will certainly be an impt. mode of operation. It may be inadequate as a "full TM", but I can add other modes, later. I do want to know if there are some essential deficiencies of t. Pure Induction mode

T. Pure Ind mode can learn how to work INV & OZ problems — it can even learn to work probs that are not logic. INV or OZ probs! So it should acquire the cones of .02–.10 ~~[illegible]~~ after it has lrnd to work INV problems.!

199.01

.01: 198.40: SN In Induction probs: When (at t. beginning), I "give" TM a heuristic,
.02 be sure to put it in t. form that it would have if TM discovered it itself!
.03 E.g. Say its learning 1+7=8 ; 3+2=5 ... then 5+3=?
? is very probably a Number. The operation "5+3=" has to have a Number as output. ~~Thus~~ — so t. final funct must be add, sub, mul, div (or whatever t. set of operations w. no. output is at that time). The inputs to t. funct
.07 must be nos. So 5 ż 3; 4 possl inputs, 4 possl functs, so 16 trials.
.08 How we get all this into form in which TM could have discovered it, is unclear!
.09 Assume an adequate corpus for ~~[illegible]~~ these discoveries.

In .03–.07 much of t. "Reasoning" of t. Heurs used, seem ~~[illegible]~~ "deductive" I will have to put all deduction in inductive form!

Re: .01–.02 ż .08–.09 : Putting t. heurs into TM in a proper form is very ~~[illegible]~~ impt.

.13 An approach: TM looks at t. "problem"; "5+3=?". It has an "association" w. t. symbol ? — that it has always been a "number". This "simple" induction implies that for ~~[illegible]~~ various "features" of t. "problem", TM has "associations". Each "feature" will then have its own "property list/set". In this case, t. "feature" is "?" ż its property is "Number".

.13 ff is a very common type of induction — perhaps t. commonest (≐ Bern Seq.).

.19 The next step, to realize that t. final funct. in t. function net must have numbers in ~~[illegible]~~ its Range, seems more difft. Each function that TM ~~[illegible]~~ has been given, or discovers, has a domain (input set) ż range (output set) — In general, TM may not COMPLETELY know t. range ż domain of a fun.
So, each funct has several properties; Range ż Domain are two: Symmetries (if any) ~~[illegible]~~ on inputs is another. Numeric functs (domain) can have anti sym.
e.g. $x-y = -(y-x)$: division has a "kind" of anti symmetry $x \div y = (y \div x)^{-1}$ (except for ~~[illegible]~~ zeros)
[this "reasoning" is beginning to sound like Lenat's "AM"!]

On second thot, .19ff is getting "Too detaild" (no longer "English")

.28 But I do want to (eventually) find a ways to express (almost) all heurs as part of induction — as a problistic narrowing down of t. "Srch Space"

One BIG reason for .28 is that it will make t. program much simpler — easier to write/debug.
I do not want to program a special section for heuristics — I want to have only a simple induction Algm.

So all I have to do is get t. general lrng system planned. The heuristics should not modify it in any way! Hvr, to test t. lrng system — make sure that it can aquire ż use t. info of ~~[illegible]~~ several impt. types of heurs.
(perhaps including "Quick Abort").

$5 \times \frac{2}{3}$ $\frac{10}{3}$

Laws of Logic: Laws of Algebra:

TM should be able to do both of these rather easily, since these laws are useful in induction — i.e. finding regularities in tasks.

[So why did I have so much trouble w. this in SAARB?]

Perhaps t. laws of alg. & solving equs is only ~~in place~~ important if
.06 our induction has ~~so~~ finite CB, Otherwise, all solns of all equs. are defined by t. equation & 1. of pc=1. So CB<∞ is necy before t. laws of Algebra ~~are~~ to become useful in induction.

Re: .06! Even if arithmetic does have cc, if it's ~~[illegible]~~ cc is ≈ ~~[illegible]~~ & the cc of simple reasoning, it would usually be cheaper to do arith than logic, so t. laws of alg would ~~not~~ be discovered!

Even w. small (~~[illegible]~~ ≠ 0) cc of arith, if TM learned to solve equs, t. laws of Alg. would probly be useful. — In fact, it may be true, that to lrn to solve equs, TM ~~[illegible]~~ would have to devr. t. equivt of "T. laws of Alg".

If TM uses Lsrch for solving problems, cc would always be an impt. factor in finding t. Solns.! (& what about "QuickAbout"?)

So: ~~try~~ to find problems for TM, in which "some laws of Alg" are useful.

OR: Just try to get it to lrn ~~[illegible]~~ to solve all kinds of Algebraic Problems:

This is proby "equivt" to lrning discovering "Laws of Alg." (!??)

ABcdefghi

It is equivt ↔ TM ends up w. t. same pc's for ~~[illegible]~~ all concs (& ∴ ends up w. same cc for all discoveries, as a TM that "Knows t. Laws of Alg".

Hvr, w. large enof ssz & enuf cc, TM should be able to discover any descrbble conc. So & how do I "descrb" "Laws of Alg" as a conc?

198.01 — on proving thms

It would seem that T. "laws of alg." should be discoverable from a suffcly large bunch of "Arithmetic Experiments" by TM. — (If it did "Experiments" of this sort.

.28 Another possy: that if part of TM's "kinds of tasks" was included ~~discover~~ Mathematical Conjectures (& perhaps occasional Proof) w. suitable Heurs to help decide if a conjecture were "interesting/useful", —
.31 That it would do better in regular lrng. of Math, Physics, language etc.

.32 — That ~~Cross Domain~~ InterDomain Lrng. (192.14–.27) is
.33 can be very impt. — $\boxed{.28-.31}$ could be an impt example.

I had originally that that Math alone could be lrnd, but perhaps ~~&~~ Math should be thot of as having 2 aspects ① Math as something to lrn: solving problems, proving thms. ② Math as an ART form (Making Conjectures, Devising & proving thms, Making Defns of "potentially interesting" ideas, etc. The direction of MATH Art could be partly (or wholly) driven by TM's examination of t. Content (& History) of Math. — What has gotten

8.25.00 Bulg.

The interest of Mathns is what, in t. long run, has found it to be useful in Science — By TM's INDUCTION on that info.

Going back to "Laws of Alg". After TM has done a lot w. numbers using + − × ÷, it might ask "Are there some "relations" of interest betw. t. nos & these functions?"

Try to find problems in which certain "laws of alg." ~~would be~~ are useful ~~[illegible]~~ in finding solns. Probably, instead of "laws of Alg" .08 it would find many specific rules: that t. "laws of Alg." would be a way to compress, to express ~~[illegible]~~ in a simpler form, those rules. —— T. value of "t. simpler form" is that the laws ~~[illegible]~~ are then more likely to be mutatable to new "laws" of interest. The "structures" of t. laws could be used for "induction by analogy", .13 Which's a very powerful, "creative" Mechanism.

So, in Saarb, I did get something like t. laws of Alg, by teaching TM how to work linear Equs. — It mite then be possl. to go from there to .08–.13: Hvr, even if t allowd cc ~~[illegible]~~ would be large, I suspect that t. ssz from t. existing data, would be too small to allow t. "laws of Alg" to be induced — Tho this is unclear! we could start w. a very large no. of "rules" (: large ssz) so t. "laws of Alg" could afford some compression.

The large CB needed could be prohibitive! We could just "wire in" t. soln. to t. srch. In fact, it probably did take a lot of search time in t. Math community. —— But t. Math community also had other heuristics/ @ the Goals (like saving time in ~~[illegible]~~ +, −, ×, ÷ & operations)

So laws of alg. are discoverable (or attained) by

1) need to speed up + × − ÷ ~~[illegible]~~
2) ~~&~~ uses in solving eqns.
3) A compressed form of t. rules of 2),
4) An orientation toward discovering "simple" rules relating Real numbers ~~[illegible]~~: Rules relating integers gave an entire new area of Math (for many yrs. a Pure Art Form) — then later found Useful in reducing cc of certain calculations — ~~& the in problems relating to [illegible]~~ Also in solving Diophantine Equs.

Re: + & − : they are inverses w. φ as unity (relation of ~~[illegible]~~ + to −)
× & ÷ " " " " 1 as unity. (relation of × to ÷)

$$x+3=?+3$$
$$x+3=7+? \qquad (?=x-4)$$

What I want now, is a (somewhat) general lrng Algm. that I can easily program. This should be general enuf so I can apply it to several Domains. Then I want to try it, first theoretically & then practically, on several TSQ's from somewhat difrnt domains

Also, I want to see if it can detect (in theory) & find (w. available cc) Various (impt / necy) heurs.

Superficially, it would seem that the "Laws of Ag." would be like "the laws of Physics" but over difrnt domains, — But maybe (apparently) not!

After ~~[illegible]~~ being given examples of many eqns & their solns, TM may be able to ~~find~~ discover that the numerical value for x satisfies the eq. if its value is substituted therein: What would "motivate" such a discovery?

.13 We mite teach it what $x^2+3x+2=0$ means as an "eq. to be solved".

e.g. give examples of eqns paired w. their solns, & see if TM can find the relation (≡ Symbolic Regression).

.16: Well, in looking for regys in the data, TM mite notice that in the problems, solve $f(x)$; 3 say, $f(3)=0$ was always true. This is a logic. regularity, but its not immediately clear to TM, that it is useful in predn! — Unless TM knew about approxn, say & could ~~[illegible]~~ think of solving $f(x)$ as a problem of finding ~~[illegible]~~ $x \ni |f(x)|$ was min.

Perhaps it would be a good idea to list a bunch of problem types to see how & if TM could learn to work them. By looking at several types, I may get ideas about how to solve them all!

(SN) from $x+x=2x$, $x+x+x=3x$, $2x+x=3x$, etc., could TM induce $ax+bx=(a+b)x$? or, more relevantly, ~~$ax+bx=3$~~ $2x+3x=7$; could it induce $x=7\div(2+3)$? could it induce "counting" i.e. $x+x+x\cdots x=nx$.

Well, we could have TM learn what solve $f(x)=0$ meant as "learning a definition". Learning how to solve $f(x)=0$ is a quite difrnt problem.

Hmm knowing the meaning of $f(x)=0$ would seem to imply compression in the sub corpus $\{f(x)=0, 3\}$ How TM would deal w. this compressn, if at all, is unclear. It may, indeed, be unable to deal w. it, if the only kind of compressions it knows, would have to be one or more functions of the expressn "$f(x)$" that would be possl. solns.

8.25.00 Bulg

.32 Quick Abort

.01.'20240 → Do I have a clear ~~conul~~ idea about induction [ā inserting hours] so I can try that linear → Quad → cubic learning?

.03 That a = b [redacted] implies b = a! There are 2 inductive aspects of this idea: ① How can it be used to compress a corpus (i.e. what corpus? ā what other regys ("facts") would TM have to "know" ② From what corpus ā what "primaries" could TM "lrn" .03?
w.o. this info, I'd have to insert .03 into TM in a special way.

.08 → It would seem that if .03 were at all useful in predn., that ~~[illegible]~~ there ~~may~~ must exist a corpus that ~~[illegible]~~ .03 can help compress — ā we could "in theory", use this corpus to discover .03.

.11 To repeat the idea of .08: if .03 is useful in any [for predn] particular induction problem, then a large set of problems of that type could be used ~~to~~ as a corpus to "learn" .03.

So .08/.11 mean that any trick that helps in induction can be "learnable" w. a corpus of proper kind.

.16 O.K. then given problem ~~[illegible]~~ cond 1 = x imp x = ?

.17 It would ~~seem~~ seem that .03 would help solve it — ~~but~~ .11 suggests that we'd
.18 need a corpus like .16 for TM to learn .03. However, we'd like TM to be able to learn .03 just by observing numbers and their
.20 relation to each other ā to +, −, ×, ÷.

Why .18–.20 is disturbing: ~~#~~ I like to think that TM could learn just about everything about R.W. from studying books ā perhaps just the internet. Hvr. .17–.18 suggests that this is not so, that we'd need a specialized corpus to lrn certain things!

.25 On the other hand, if any idea is useful in predn., then that "idea" can be learned w. a ~~[illegible]~~ suitable (ā commonly occurring) corpus.

So .25 can be regarded as a Theorem about concs useful in predn.

Now, How to "try out" .25? Find some things I want TM to lrn ~~[illegible]~~ in which the laws of Alg. are useful.

.32 SN on "Quick ABORT" heuristic: By speeding up search, it enables us to get higher PC (in Lsrch) for a given CC. Just how TM is supposed to recognize this, is unclear at present

.35 I guess the thing that disturbs me about "lrng Laws of Alg" — that at first glance, it seemed like it should be easy for TM to lrn such a thing: What is not clear is just when ā when it is not "easy" to write a tsq for a particular lrng. task.

→ 204.01

8.26.00 Bulg

.01 : 203.40 In t. case of "Laws of Alg" (or any other lrng task). we have to ask: what is t. criterion for TM having lrnd this concept? How can we test TM's "knowledge/understanding" of it?

Clearly, it is not legal to ask TM "what are t. laws of Algebra?". TM doesn't really speak that lang. The only legal tests are inductive problems.

.06 What ~~[illegible]~~ inductions can it make? What new inductions can it lrn rapidly?

(This corresponds closely to a Human Teacher Monitoring a Human Student: Trying to figure out if t. student has "acquired" a certain heur, or whether t. student "can work" a certain type of problem)

Consider "equals": If TM knows about "relations" then "=" is a symmetric relation on 2 things. It is idempotent and transitive.

I could ~~[illegible]~~ have TM lrn defn. of relation: Then ask if "=" was a relation, ask if it were idempotent, sym, transitive.

So: A present possl View: That TM can discover any conc. that is useful for [illegible] t. tasks it has been given (if ssz & cc are adequate).

— But discovery of "Laws of Alg." is probly not essential for TM to be able to work most Alg. problems. When TM studies other kinds of Algebra in which a·b ≠ b·a & t. assoc law may not hold, etc. — Then, it may be more likely that TM will ~~[illegible]~~ try to compress t. set of rules it learned in solving

.19 algebra problems — to give t. laws of Alg.

(3 ways to deal w. "Laws of Alg" 182.27)

The (203.35–.40, 204.01–.19) is a reasonable way to look at "lrng t. laws of Alg" (also note 203.25), I still don't feel entirely comfortable about t. Question. (The oversavers of 205.35–.40) still remains!

.24 : 187.17 Look at my initial treatment of TSQ1 (cf 181.5) on 187.01 ff. In a realistic Situation, TM would be just starting up & would have no heurs that it could have discovered itself. Nevertheless, I could just insert a tsq from to "TSQ1" & see how t. Lcost increases (w.o. use of Heurs in t. TSQ).

T. point of the exp. would be to see how t. solns. to difrnt problems in t. TSQ, interact (or are indep) in t. solving of a new problem.

.30 → In particular, I'm interested in how Heuristic info is acquired & applied.

At present, I see no particular diffy in .30: TM just has to acquire t. Heur (which is a regular induction problem) & apply t. results to new (post heur acquisition) problems.

Well, in 187.13 ff, (3.7+4.1=7.8 ....) we normally try ~~[illegible]~~ unidirectional functions ~~[illegible]~~ on t. input expressn. $\boxed{3.7 \mid + \mid 4.1 \mid =} \boxed{?}$ (also, part of input info is "where ? is") We start with t. set of funcs that we have: & we continue by making more complex functions on t. input expression, 3.7 | + | 4.1 | = | The "final" function (usually must) map to nos. (This ~~[illegible]~~ is a heuristic discovery that narrows down t. P.D. for the final function) The inputs to the (identity) unary function are 3.7 or 4.1.

205.01

8.26.00 Bul g.

205

~~39~~ 40

9 x 30 = 270

.01: 204.40: (we can "wire in" t. fact that the numerical operators identity, + − × ÷ only have numerical inputs so this limits our choice.

This learning 3.7+4.1= 7.8, but this new operator will not work with "3.1=?" (t. ~~[illegible]~~ earlier problem lrnd on 187.08 ff) [redacted] So, it now has 2 operators: one for things like 3.7= ? the other for things like 4.2 + 7.1 = [redacted] ? So each is good ½ t. time, so

.07 they are both useful, & / coding compression's obtained. → (.16)

.08 TM's motivation to continue searching for a better code: Unclear!

I did write about this recently, hvr. (177.02) The idea was that we needed "serious theory revision", when t. system's error rate for t. new data was much worse that what we had before (or for whatever reason — worse than what we expected). ~~[illegible]~~ Hvr., this is a very Vague answer! I'm mainly thinking of Motivation for Theory Revision

.14 in Physics.

.16: [redacted] (.07) [redacted] Then (for reasons, like (or better than) (.08 – .14) we look for a way to get

.17 better predn. A common way is to look at the sub corpus in which "add"

.18 was t. rite operator v.s. sub corpus in which "identity" was t. rite operator.

3 = [?] v.s. 7+2 = [?]. We can distinguish them in (~~several~~ 3) ways: One way:

.20 ① 2 symbols v.s. 5 symbols: (Hvr can TM "count" at this point: ?).

.21 ② 2 nos. v.s. 1 no.: ③ ~~presen~~ ~~[illegible]~~ (absence of "+" v.s. presence of "+".

[anyway, t. same kind of problem arises when −, ×, ÷ are lrnd.]

The Heuristic of .17 – .18: How could it have been discovered? —— what hypothetical corpus ~~[illegible]~~ could give rise to it? What operators could be combined to discover that regularity? Seems like a very sophisticated heuristic! :

NO → One way: ~~[illegible]~~ (Even w.o. t. heur of .17–.18) TM notices during execution of cands, . . . ? . . (nothing to notice! on n=?, we try ~~[illegible]~~ add(n,n): On ~~[illegible]~~ m+n=? we ~~try~~ try Ident ~~[illegible]~~(m) & Ident(m).

A (perhaps) reasonable way to get heur .17–.18: It is a way to ~~[illegible]~~ xfrm t. original induction problem into a new induction problem ("OR" xfrm)

So we end up w. a ~~[illegible]~~ "Time Series" w. Elements n = ? → "iden" m+n =? → "add". We ~~[illegible]~~ need a Op ~~[illegible]~~ that has "iden; add as outputs.

I have to investigate the primitive ob functions. How can they (w. op functs) generate t. needed discrimination?

Compare $\left[\frac{m+n=?}{n=?}\right]$ —— 2 subcorpi How to discriminate: The "+" sign is t. simplest way. (The "size" ~~[illegible]~~ [illegible] also quite simple).

~~[illegible]~~:

8.27.00 Bals:

What I really need to do: Examine OBs & how to use them in induction;
How to use them to dscrb. corpi

In general, obs have ~~same~~ inputs as ops. numeric, Boolean, character/string. ≈ Basic "select case" inst

→[I may want output to select from a set of functions, rather than be just True/false]

Mainly, I'm thinking about an op as making a choice among ops. ~~As~~ Obs are a subclass of ops

In t. present case, I want an "ob" that can recognize ~~difnce~~ difrnce betw.

.08 "n=?" & "n+m=?" and use t. info to excite t. proper "op".

.09 ⦿ Presumably, I have an ob that can look at ~~a symbol~~ an element & give 0 if a number, 1 if a [redacted] non-number (see .26)

⦿ Some other possl. (ob functs) of interest: The usual "If a(≥=<)b" T/F.

This last is "Does such a "relation" hold [redacted] among t. inputs

A "relation" can be regarded as a kind of "classifn" of ~~n-tuples~~ "ntups"

— Hvr, all elements of t. relation have t. same n, so "relations" are not t. most general type of set.

.14 ⦿ **One kind of ob** gives: **If** (a) then (b) else (c)

a is an ob & b & c are ops. so .14 uses a to chose betw b & c.

Hvr, what I want is difrnt. from if then else; t. These are usd usually, to control flow of execution: Two do difrnt things in a functional lang.

So, .14 ends up being an op.

The problem I'm working on, is to augment the normal set of funcs used in [redacted] ordinary Algebra ~~[illegible]~~ presumably by adding t. obs.

Hvr, other people in **GA** probably have been using an adequate set of operators. Koza has (usually) been using LISP.

Returning to Nos Moutons! Say S is t. set of non-numerical objects that have appeared in t. corpus this far. So S = (=, ?, +) only 3 things.

.26 ⦿ Assoc w. each is a "ob" that can tell if it occurres at a "place" point in t. corpus, by having a T/F output (Boolean). (see .09↑)

.28 **I could** drop this problem for a while, & assume that Somehow, Somehow t. lang. was able to distinguish betw. "n=?" & "n+m=?"

.30 There is a Q about t. form of input: The input is a set of strings of symbols & numbers. If t. string is 20 symbols long, I have assumed that t. input funct could accomidate 20 inputs, simultly. ~~[illegible]~~ Long equations will be hard to accomodate! A possl. way is sequencial input of t. symbols. An ≈ equivt method; T. inputs of t. functions have addresses for their args, so they can get them from any point ~~[illegible]~~ in an arbtly long input string. — Hvr, best to work on this Q when I x/t. from "English" into "Code" → 210.02

Perhaps model this language after t. way I write pgms in, say "Basic".

Also, if my dscrn of what I do (in English) is simpler, I should be able to put it in more exact form — i.e. a formal lang.

8.27.00 Bulg.

OK: Well "n=?" v.s. "n+m=?" "in English"

.02 So I had ~~Pro~~ 3 approaches of 205.20 —.21 "in English" ① 3 v.s. 5 symbols

.03 ② 1 v.s. 2 "nos." ③ presence v.s. absence of "+".

These involve ① counting symbols ② counting number ③ counting non-nos,

, ④ "counting" "+'s". ⑤ Threshold for no. of counts. ⑥ use of <, ≥, <, ≤, =

.06 for "If" part of statement.

(SN) Unfortly, having 3 parallel codes doesn't help in normal Lsrch — we just use the single one w. best Lcost. But note: 3 ‖ codes has ~ 3 times as much CE, so we'd end up w. ≈ same Lcost of soln! → (there may be something wrong w. this arguement, hvr. — it has been my impressn. that $\frac{pc}{ce}$ would be appreciably ~~better~~ as a stop criterion for Lsrch ~~than~~ is ~~[illegible]~~ normal Lcost ∄ — So "check this out"! Ⓒ) 8.21.90 TMGen: 1.01 – 15.90 is on this problem.

In .02, I'm keeping discussn "vague" in English: Continue in "English". I may not be as ~~good~~ as I once was in xlting from English into Math, but — too bad!

Anyway, in .02 – ~~.06~~, ① & ② have to do w. "size of expressn": I think this is a very fundamental idea — related to pc (≈ "Alg. Complexity"). — So ~~the~~ TM could have this "Builtin".

~~Another~~ "Builtin" skill; in .03 — recogn of "+": This is a "novel" symbol because it doesn't occur in "n=?" so this, too, is a legit way to recognize difference betw. strings.

[SN] A worry about ENGLISH dscn of tsq.'s soln. methods: That a thing that seems simple in "English" may not be! The main example is "Derv of Laws of Alg".

Back to the TSQ.: A possb. sequence of problem types:

1) Learn to evaluate alg. expressns. (not in Polish) — Perhaps try Polish as ‖ tsq — or intro to Polish after both ways!

so $3+5=?$ $3\times5=?$

$3\times(8+9)=?$ $(7\div(3+9))+2$

See if TM can "Get the Generalizace" of any pt. in the tag, — which means it can do evalns. of arby "depth".

Next soln. of eqns ~~x=3 ; x=21~~

~~Solve [illegible]~~ ; solve(x, x=1) = ?? Evaln. of the function "solve".

$x+1=3$; $3\times x+3=1$; $x+x=3$; $x+3x+7=2$.

("discovery" that $x+x=2*x$)

→ ~~Perhaps~~ Perhaps try varying that ~~STARB stuff~~ / on TSQ.

⟷ Hvr., note that one of the Great Breakthrus assoc. w. MCT, was that TSQ's should be easy to write! That any needed conc. should be teachable by examples, hints, "Telling" or "[illegible]"

So, just try doing an "English" TSQ for Algebra! — continue what I've done, including ~~[illegible]~~ heuristics. The heurs are usually —209.01

8.29.00 Bolo

Varieties of knowledge TM should be able to have:

1) Condl. P.D. : Input: desn of INV prob: Output: Pd on desns of possl. solns.

2) " Input: OZ problem desn: output: P.D. on ~~[illegible]~~ Optzn Techniques (OT's): Pc = Proby that that OT would give best G for that problem in available cc.

3) Languages : Pd's over sets of finite objects (possibly ~~over~~ infinite objects) (as well)

4) Given the "Nature" (i.e. source & assoc. info) of a T.S. & a subsequence of it: It has a P.D. on continuns. of that T.S.

~~5)~~

The foreg. are the kinds of info that I've been mainly concerned w.

~~5)~~ Knowledge of π, √2 etc: essentially the ability to calculate them to any precision a/o the memory of a finite no. of digits of them. (⑥ covers this)

.13 6) Knowledge of how to do various types of math problems like solve eqns, compute π, √2, sin 35; multiply, invert matrices, integrate, differentiate symbolically (This includes ⑤)

.15 7) Ability to reason logically: Given postulates, to be able to prove theorems, conjecture theorems; Give a bunch of data, to be able to deduce info relevant to a particular problem.

The "laws of ~~[illegible]~~ Logic" & the "laws of Algebra" can be lrnd inductively a/o their equivalent in predictive power can be lrnd inductively

.19 8) To be able to plan a/o carry out a research program (like "find cure for cancer" or "devise better CPU Hardware" or "find a better theory for physics".)

Re: .13 (6): These can be taught to TM inductively.

Re: .15 (7): " " " " " " " : An impt. Q is: can TM learn to apply this "reasoning" to its own problems"?

Re: .19 (8) I've worked on this problem some: Don't remember if I solved it or not!

.32 One possl. Big Diffy: I thot of starting w. a "simple" TS Q from Algebra only. But it is likely that to discover or even use the needed heuristics, Many more "modes of knowledge" will have to be had by TM. !

8.29 Bdg.

$L_n \equiv n!$ ~~$\frac{L_n}{L_k L_{n-k}}$~~ $\frac{L_n}{L_k L_{n-k}}$

w/?

.01: 207.40. not absly nccy at this pt., but put them in, because I want to know ~~that~~ that I know how to do this! : Also note 208.32

In particular, note 208.16 & (208.16R) ←

One apparently impt idea is ——— — T.M. has to learn to "reason logically" — Many heurs involve Logical Reasoning. Since both logic & Algebra will ~~[illegible]~~ both have t. same kind of "large SSZ", ~~[illegible]~~ TM's will have about t. same confidence in its preds, in either area. We can perhaps make some kind of Formalism for TM that is equivt. to SSZ = ∞. It is hoped that "Truths" obtained w. SSZ = ∞ will not conflict — since we would then have to know ratio of sizes of diffrnt ∞'s! (Difference betw 2 ∞'s mite be impt. e.g. $\frac{\infty_1 - \infty_2}{\infty_1}$ $\left(= 1 - \frac{\infty_2}{\infty_1}\right)$)

The impt. idea here: That TM has to learn to "reason logically".
— That there is no real difference betw. TM learning Algebra (/Math) & learning Mathematical logic (as part of Math) & trying to use ~~[illegible]~~ this Mathematical Logic in obtaining Probabilities.

SN Many Heurs will have to be fnd. by OSL. We could wait for 1 of "SSZ" to 2, but this would be wasteful of SSZ; Also, it is essential that TM have ~~good~~ facilities for ~~[illegible]~~ doing OSL.

.19 So what I have to do, is write a TSQ in English; & Then ~~[illegible]~~ figure out t. needed (or "good to have") Heurs in English. Then figure out what sort of training in logic ~~[illegible]~~ and Math, that TM needs befor it can discover/use each heuristic.

Perhaps Try to find where I've already done much of .19.

Perhaps Reread Original Proposal for Sol 89. It even had estimates of Timescale!©.

But I ~~x~~ shouldn't be "Too Carefull" in constructing TSQ! T. technique is supposed to work w. not-so-perfect TSQ's.

So in writing t. TSQ & Analysing "in English": I can use various amounts of "Heuristic Guidance" To start off, I just want to get ~~[illegible]~~ a feeling for just what (& how much) has to be done. Ideally in "Steady State," it should be possl. to train TM using not much more than a TSQ that has been designed "IN English". — That if it doesn't acquire ~~[illegible]~~ a certain conc. w. acceptable cc, I will analyse what occurred, &/or just ~~[illegible]~~ give a "hint" or give it (or teach it) what I think ~~[illegible]~~ is t. relevant heuristic.

.34 I think t. latest TS Q analysis: 207.01 — .17; → 204.24 (205.08–.14) dicty 2!
205.07 ~~[illegible]~~ — .40 206..28 — .40, 207.01 — .49; 209.01 ff!

Dittos! 205.08–.14

Main points: start w. ~~x~~ ~~[illegible]~~ 3 = ?
Then 3+7 = ? ~~[illegible]~~ 3 = ? is solved easily because there are only a few/ functions (primitive) that give nos. as output. That TM should notice that "?" has always been a number, is impt. — ~~[illegible]~~ This seems to

Bolg

be a normal kind of Inductive reasoning. —— So Then TM only tries fnctns That have nos. as outputs.

.02 [The form of input to TM, sequential a/o parallel will effect encoding lang, but I want to delay ~~[illegible]~~ seq. v.s. || until I xlt from English to "code"] ⟷ see →

Q about format input — sequential or ||? 206.30–.40

After it learns 3=? we do 7+9=? [It is able to do this — again, noting output is a no., so only a few fnctns have to be tried.

From corpus containing both 3=? ż 7+9=? TM gets it rite about ~~50%~~ of t. time — which is v.g. compression.

Motivation to ~~improve compr~~ try to compress further: well ① it can be given by user, ② for Older More "MATURE" TM, it will have ~~[illegible]~~ some idea as to How Much compression expected in Math, (ż ~~50%~~ accuracy is not v.g. for math.)

To improve compression: TM ~~[illegible]~~ divides ~~the~~ corpus into 2 parts: ① Those using identity f as soln (3=?) ż ② Those using sum( ) as soln (3+8=?). It must then distinguish betw "3=?" ż "3+8=?"

← What is reasoning behind this?

.16 This can be done by ① size of string (3 v.s. 5 ~~[illegible]~~ symbols), ② no. of nos. 1 v.s. 2

.17 ③ no. of non-nos 2 v.s. 3; ④ Presence of "+" symbol ~~[illegible]~~

So at this point TM got a ≈ perfect score. (Assume "Presence of +" t. criterion is not used here, because later we will have to ~~[illegible]~~ revise the other decision methods. ≡ = "Backtracking ... use of (parallel alternative) codes)

← if we use several || codes ("oversearch" W.L.srch.) then Backtracking is not necy, later.

Next do 3−7? This is easily solved, but then we get it rite only ½ of time, since sub( ) ż add( ) would both be used here. We have to distinguish betw 3+5=? ż 7−2=? By "string matching", it's easy to see that "+" ż "−" are the ~~[illegible]~~ only things that distinguish t. two.

So at present, we use say ①, ② or ③ of .16–.17 to distinguish betw 3=? ż 4+7=? This also gives us 3−8=?; we then use the "+ v.s. −" to further distinguish betw add( ) ż sub( ) as solns.

.30 [We could at this point, reformulate t. system, ż use ~~presence of~~ "absence of + or −" to invoke recogn. of 3=? — i.e. t. identity operator. Whether we would do this at this pt, is unclear. A "Person" mite notice that this reformulation ~~[illegible]~~ gives a shorter over-all code. Unclear as to how to ~~get~~ Motivate TM to look for a shorter code when ~~the~~ predn. is ≈ 100% rite!

Doing 3×4=? ż 7÷2=? involve no. new ideas.

8.30.00 Bulg.

Doing $3*(4-5)=?$ would seem to be quite diff'x. at this point. TM may associate * w. mult( ) & − w. sub( ), but this is rather vague. Trying [redacted] mul(3, sub(4,5)) is a reasonable possy.

Similarly, it would learn $(4+5)*7=?$ and $3*(4+8)=?$ ect: 4 contains $4*(8\pm3)$ & $(8\pm3)*4$ to be lrnd. Somewhere along this path, it should begin to do some "hyperorder lrng": by noticing that there are symmetries, similarities in its "rules" for various situations. Such hyperorder rules be ↑ pc of corpus,

— Also, when new kinds of problems are given, they ~~[illegible]~~ ~~[illegible]~~ solve the new problem immediately or help speed up needed srch.

What I'd like, is that at some point TM should ~~[illegible]~~ discover what "parens" "really mean" & induce a general rule for evaluating any alg. expression in Infix notation.

O.K. so that would be an interesting **TSQ:**

Another TSQ: learning $3=?$ & $2+3=?$ $2-3=?$

~~solve(x, x+1=3)=~~ Solve(x, x+3) = ? Solve(x, x+3) = −3, the soln of $x+3=0$.

solve(x, $x+c$) = ? c is a literal; & ? is −c in this case.

It may be necy to get TM to learn mult & div first (?).

After TM learns to Evaluate arby Alg. expressions, try teaching it solve linear equs. — Some "non-linear" equs. ← meaning of this isn't quite clear. Well, equs in which some alg. Manipulation is necy, but final equ is "linear" to be solved.

.24 → Perhaps I want to get TM to the point where its using the logic a/o math it lrnd, to help find models for new problems.

.27 **Long Range Q!** Just how much work will it be to get TM to the point where usually one can teach it w.o. getting into details about how TM is going to lrn it! i.e. ~ to teaching a human person?

This Q may depend on Domain: i.e. one might be able to do it in some domains, but need more training before it can be done in other domains

Proby Sub-goal .24 is a large step toward Sub-goal .27

Anyway, to get .24, I may not need a very large TSQ: Even in simple problems, if one wants to work them w. very small CJS, one must use much "logical reasoning" & "Mathematical Analysis"

An impt. Q is how to get this Mathl/logical reasoning to arise automatically, w.o. any special prmg. on my part. → 216.01

9.1.08 Bulg.

The defn. of Adequacy of an English Soln of a TSQ. .05 — .10

1st a Share o'n pad ~~[illegible]~~. I may need to go.

.01 : 211.40 : A possi. Example of "Much Reasoning" is in t. Substitution Heuristic used in soln. of linear → Quadratic → Cubic ~~[illegible]~~ equs
Also the "Planner" heuristic that breaks probs into "or," "and" nets. — This devises sub-goals.

.05 → SN I have been a bit worried, recently, about mapping from ⟨English desc of t. TSQ & its soln.⟩ → to → a program (SW, HW). This shouldn't be too hard.! [If t. English soln to t. problem is Legitimate then there must exist a {probabilistic} practical search method (perhaps ≈ Lsrch) to implement it. So return to t. sheep & write t. TSQ's! → see 219.19

.10 → So .05 really defines t. Adequacy of an English Soln. of a TSQ.

O.k. So let's Go back to t. TSQ of 209.34ff: How "complete" is it? — What needs to be added/changed to get it to satisfy .05 → .10?
So start w. 3 = ? : T.M. has a bunch of functions available: at least iden (x → x), add, sub, mul, div.. The input consists of 3 "things", 3, =, ? a "thing" is something that can be an input to a function. At this point, = & ? are not "things" because there are no functions that can use them as inputs.
For each t. "Things" (other than nos.) that have appeared in t. corpus thus far, we can define a function that recognizes it: Has "T" as output if input is that, "N" otherwise. This SW/HW idea gives some way to estimate pc of various functions (usually primitive functs at first).

We also have a function that recognizes Numbers; {It is the "not" of the "or" of all other recognition functions.} I may want a function that is T if its assoc symbol occurs anywhere in t. input string. If it does we can look at each position to see if it's there.

Well, O.k. say we have all of these OB functions: for the 3 = ? problem, they could be tried at first level, but they would give "T" or "F" out — never a no.

Eventually, I'll want a means for TM to recognize that a funct w. numerical output is needed — & thereby ↓ t. amount of search needed by a sizable factor.
.30 So TM "Notices" that output is always a No., then
.31 concludes that only functs w. numerical output can be tried as "final function".

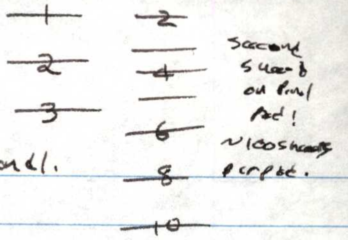
Just how do we realize t. resultant pc↑ of .30 → .31? What Mechanisms can I use? (I want to desc. t. Mechanism "In English" w.o. going into
.34 HW/SW details). → 218.01 see

Perhaps skip this "detail" at this point, but make list of these "lacunae". When I have several, see if I can workout a technique that deals w. several (or all) of them.

→ 218.01

9.2.00

Bulg



216.134 spec

.01! 216.40 : One Approach: One has to predict "?". From pure uncondl. probty, we find it ~~[illegible]~~ h has always been a "number".

That + − × ÷ have always had nos. as outputs — we can give TM a pseudo corpus that tells it this — since ~~[illegible]~~ as an infant, it wouldn't have much ssz of this sort.

The discovery of t. "working backwards heuristic". This can be obtained

.07 2 ways: (1) purely empirically, TM notices that t. Macro function must have a terminal function that has numerical output, if it ~~[illegible]~~ requires that t. Macrofunct ~~[illegible]~~ have a numerical output. TM would have to have acquired t. needed concs ("Vocabulary") befor this could be "noticed".

.11 (2) Because needing a number for "?" and "sum" always has had numerical output" have ~~[illegible]~~ very large ssz, TM would consider using Logic (ssz ≈ 0) on these statements. [Also, proper "adequate" "Vocab" "concs" needed befor this can be done]

.14 The methods of .07 and .11 ~~[illegible]~~ need probly t. same concs (vocabulary) and probly. some difrnt concs.

The maybe .14 isn't so true! T. main difrnce betw. deduction logic and induction is that in induction, one traces out all possl paths — even those w. low pc. In deductive logic, most paths are of Zero pc., so one doesn't bother tracing them out. In approximate induction we will not trace out very low pc paths, so we can approach "Deduction" "Logic".

SN Say we ~~[illegible]~~ "tell" TM ~~t.~~ some laws of logic a/o laws of Algebra. If this info is on "T. Advice Channel", it can be regarded as a "Hint" — in t. sense of reducing t. cost of searches for TM.

So: Ok.: we have given TM enuf info so that its lrnd that "?" is "usually" numeric, that "+, −, ×, ÷" all are "usually" numeric input, numeric output. TM always constructs new cands using old cands and useful sub ~~[illegible]~~ trees. In making final trees, it could "respect" any info that it may have on I/O of t. various Operators.

281 6.28 ·08 .28 Normally, t. Mops are constructed "Backwards" "T. final output func operator first. ← ?

Well we coult construct them this way: start by doing all possl. functs w. desired output "Type", connected to all possl. legal input types.

Next, to add ~~[illegible]~~ one more function to t. "Widget", replace each input to t. function w. all possl. functions, and have t. input to that funct. be all possl. inputs.

To insert 1 more funct, replace all input lines by ~~t~~ all possl. functs.

.28 ff seems to get a difrnt result from ~~the~~ method of function generation that I used on (143 ~~130~~.25 — 131.18), 200 on symmetries 136.35 on Realization in "Basic", which builds up functions starting w. input lines. This method is not so good, if one a certain "Type" for output!

On t. other hand, it may be that .28 ff is not correct! — that it doesn't get all of t. functions w. a given output "type".

→219.01

9.2.00 Bulg.

.01: ~~#~~ 218.40

F₁ is t. output function.

f₂ is t. possl. additional funct to f₁

f₃ " another possl. additional function to (f₁, f₂)

Actually, this analysis need be done for usually no more than 3 or 4 functions, because the pc's ~~became~~ become so low, due to t. very ~~[illegible]~~ large no. of possys.

So drop this for a while (Note 143.20: Discussn of ~~Symmetrices~~ Symmetries in functs, can ↓ no. of possys by a lot!)

The Present Problem: To understand how ① Heurs are discovrd by TM: what form t. discovery takes. ② How are these discoveries implemented in ↓ ~~[illegible]~~ cc/≷ for Lsrch? I.e. How do we use these ~~[illegible]~~ "discovered heurs"?

.12 I'd like t. heurs to be in some "standard form" to facilitate their applicn. (.17)

.13 SN In General "Types" of input/outputs of functs could much reduce no of possl. combinns. The formulae of 130.25 ff didn't consider this. 143.20 discusses Symmetries; Possibly ~~[illegible]~~ a ~ method could be used for "TYPES": I'm thinking of using mainly 3 types Number, Boolean, ~~[illegible]~~ String

.17: (.12) → Try to put this Q into as exact form as possl.: Exactly what is t. Main problem Now?

.19 216.10 One Big problem: (linked to 216.05→.10): How are regys (≈ concs), compressns found? ~~when the~~ After they are found, how are they used for predn.. How do these new found regys modify subsequent searches for regys (≡ compressn)?

.23 "Soln" to .19: "Inviscount all of t. regy found thus far" Try all possl. codings of t. corpus in ≈ order of pc. ~~[illegible]~~ Store all signifctly useful codings. (i.e. signift compressn are obtaind)

A possl. Approxn. to .23: We are considering only codes that are operators. So we have a conditional pd, betw. input & output. We start w. a set of ~~primative~~ primative functs (of perhaps ≈ eprip). We code ~~cand~~ cand. Operators as functions dscrbd by trees. [I have to work out t. details of t. coding of these trees — I will try modifns of "Z141"]. Hopefully, these trees can be dscrbd in ≈ pc order using t. "Huffman technique" (130.19; 142.01) 143.05 ff, 144.01 discusses ● PSG's (phrase Str. Grammars) as ≈ models for "tree search" — derived vs. non-derived ~~[illegible]~~ languages.

So, anyway, we list these cands in ≈ pc order using ≈ Hufman trick. Each function has a pc assigned to it, so we can do the "Huffman listing": Any signifctly "short code" ~~whenever~~ for t. corpus, becomes a newly defined Operator (≡ function).

~~Also~~ After defn of this new operator (we ~~may~~ may be defining several of them in "ll codes"), we have for each such new macro function, 220.01

9.3.00 Bulg.

.01; 219.40 - a code for t. corpus: This is a "2 part code". Part 1 is t. "Precorpus" that consists of the primitives, followed by ~~then~~ all of the definitions of ~~functions~~ functions (that terminates w. t. defn of t. last Macro funct). This is followed by t. defn of t. corpus in terms of that last Macro Function: This lasting in t. style of 214.1, in which t. pc of any element will depend on how many times its been used in t. code thus far. [I haven't yet figured out how to do this ~~in~~ for ~~finite string induction~~ Unordered set induction US Induction v.s. TS induction (time series)] TS induction vs. Finite string

Dol

US induction
TS " "

— Think of CFG's. — How t. various pc's for choices get updated w. each "use" of them. I do want to write this up in some detail!

~~This process was t.~~ My not understanding this process, was t. cause of much confusion about pc's of functions & "sub-functions" in both t. coding of t. corpus & t. coding of new functions & sub-functions,

SN If I really work well only during morn (& sometimes better going to bed?) spend much of day prepping problems so they are in form ~~that is most easily worked on in Morning (/ before bed)~~

Still, Much confusion in my mind about difference betw. Stochastic Operator (function) & a Stochastic CFG. [Well: one Big Difference: CFG is a stoch. operator for a constant (invariant) input. !

.18 One Try: say we've coded t. corpus (a MTM corpus) using a string of definitions culminating in a certain Macro funct, Fo. We get a new bunch of I/O pairs that don't seem to fit well. What do we do? : Well,

.20 (i) We try our operators in PC order: (?) This would mean starting w. t. primitives & using each of the Macro Funct's that worked in part of t. past TSQ. Actually, none of them could work & they all give it zero pc to t. corpus — → they have all been tried on smaller parts of corpus & failed! so diff't to compare them. Next in order of PC, might be taking Fo &/o other "Recent" Macro OPS & ~~Mutating~~ MUTATING them, by inserting & substituting [redacted] different functs for sub operators (sub ~~string~~ trees) within Fo and other recent Macrofuncts. ~~Occasionally Mutating (ALOSL)~~

.27 Actually, MTM problems are always INV probs. — So maybe try Lsrch. In this case, TM has to "recognize" t. new kind of problem (recognition [redacted] involves combining Concs., just like any other problem.)..... "Recognition" is a kind of "Categorization" which is an impt. kind of knowledge

.32 that must be acquired.

.27-.32 is a common approach to t. problem of .18-.20. We Try to find a minimal code telling how t. "new input" differs from all of t. old inputs. (One ad-hoc way is t. passage of time:) Actually, time may not be that "Ad-H"! The new (untractable) input may be identical to an input in t. more distant past — but "Times change" — so TM has to adapt. One way would be by finding a new soln. for t. recent data, & give wts. to t. ~~and~~ old & new soln. — So we are not

222.01

9.4.001 Bulg.

"MCT : How to do everything Rite, Every time."
Titlo for Talk.

→ This Changes it from a MTM to a NMTM problem! ←

.08 220.40 100% sure of t. new soln. T. relative wts ~~will bias~~ should bias more strongly toward t. "recent" soln., as time passes. >>>

While 220.27-.32 is, indeed, a common approach to this problem, I do feel it is somewhat A.H. & I'd like a more General approach.

This is in line w. my wanting a very simple/comm pgm for All of TM's problem types & "Modes".

It may be that my considering only MTM type problems in 220.18-.20 is where the trouble is. [In general, TM is supposed to have this

.10 ← This is MCT → very general Cond'l. PD that contains, stores, all of its "knowledge", including Heuristic "knowledge".

The Macro funct. sought in 220.18-.20 doesn't contain enuf info to be very useful for Heuristics (← Heuristics are usually "NMTM distributions — i.e. pc's are not always close to 0 & 1)

Hwr., t. MTM d.f. of 220.18-.20 could be (is) a part of the more complete P.D. of .10.

→ I think the PD of .10 is my Simple, unified model for TM lrng. It is t. P.D. that TM's spends its cc "Updating".

But how does my approach in Sol 89 fit into .10?

Sol 89 was making new cand. concs by combining older, concs that were found to be useful. How does S89 fit into MCT?

This seems to be t. rite Q! In fact, whenever I get into trouble, I should always ask: "What does MCT have to say about this?" — or/ More specifically what does MCT have to say about t. SAARB TSQ's?

So go thru a Saarb-like Tsq. & see what MCT says about it ... how it should "really" be done. T. General idea is that t. P.D. helps solve problems, & after each problem is solved (or even just "worked on",) t. P.D. is modified, to "contain" this new info.

.28 So for t. ANL problem, we start w. our primitive functions, & means of combining them. T. functs themselves have pc's, as do t. operations that combine them. This gives rise to a P.D. for all functions. They are searched in Lcost order to try to find one (or several) "Levin Search". that do ANL as far as t. corpus goes. So we find some functions that satisfy us, (so far).

.33 How do we then Modify t. P.D.? Well, we add these → To deal w. t. new examples that "Don't fit" t. old Macrofunction; functs to t. set of primitives: but what pc's do they get?

Also note: in .28 we used an unconditional P.D. In .33 we need a conditional pd. One of t. things t. cond'l. P.D. considers,

.37 is "What to do when we have to "revise a theory"." Given a denote a "theory Revision situation" — we want a P.D. on techniques to Deal w. it.

223.01

.31 On t. UNIVERSALITY of a PRIM. Rec. Funct Generator.

.01! 222.40 : While 222.37 is close (of just what) we want to do, we may want to "Elementalize" (break into Tentative AND or OR) sub problems, (that may or may not solve t. main problem). { Remember that in breaking a task into sub tasks (AND/OR not) part of one's time is spent trying to prove (show) various of th. tasks are impossl.! }.

T. cond'l p.D. of 222.33 is t. MCT P.D. It characterizes t. entire corpus thus far, (as well as t. techniques we've used to try to solve probs (which is now part of t. Corpus). Hvr., in this prelimy discussion, I may not want to Consider t. Entire Corpus .... but this is unclear just now.

Another point! Storing t. Cond trials as part of t. Corpus, would seem, at first glance, to be pointless, since t. trials contain Zero info! T. trials are completely dscrbd by t. "startool TM" & t. "Lsrch algm": — Hvr, t. foreg. assumes $CB = \infty$. For finite CB, it may be worth while to store certain info in t. Lsrch (other than final results). An extreme (tho a bit useful) case would be storing very many of t. short codes of t. corpus, — t. less short ones being good cands for later "Theory revison". [T. Revision Problem"]

12:55 { So T. Big Q is "How is t. Big (MCT) P.D. updated (revised) after a (successful) Lsrch? How will TM's subsequent behavior be modified? ← (More General Q).

Consider t. early T.S. Q. 3 = ? ; $3 \overset{\times}{+} 4 = ?$: We could start w. a simple P.D. over a (potentially universal) set of functs / functionals. (possibly a universal or Prim.rec. Universal Algm) .36 .22

SN — It is not clear what a universal Prim. Rec. Algm is: Perhaps if $A(\alpha_A, x)$ is a Prim.rec. algm, A (α_A is t. descrn of a P.R.algm, X is its argument), Then if $U(\alpha_U, x)$ is t. Same funct, implemented on Universal Machine, U; Then for all α_U, α_A $|\alpha_U| - |\alpha_A|$ is bounded by a constant that is a function of A & U only. This would be t. case if A were a Universal Algm.

Or, say A^1 & A^2 are 2 algms that can generate all Prim Rec functs, Can A^1 "majorize" all generators of Prim. rec. functs & not be a Turing) universal function

.31 ⟹ Since t. prim rec. functs are effectively enumerable, A(N, x) would be Prim. Rec. Universal (N is t. index number of t. funct. desired), & could describe only Prim Rec. functs. I suspect that one couldn't get signif'tly small "N" w. a Unc!

1:36 Its not clear that my PrimRec. lang. is Universal in t. sense of .31: but Drop this for now

.36: (.22) consider t. set of successful trials: ① for 3 = ? find t. number in t. input. set ? to that no. (Equality function) ② for 3+4=? 3 solns: ① ② find 2 nos. a, b; set "?" ← Sum(b,a) Sum(a,b). } Same pc for these 2 solns.

1:51 ③ Much hyper pc soln: if "+" occures, set ? ← Sum(b,a) Sum(a,b) else ? ← a (a = any no. in input) ⟹ (224.01)

9.5.00 Bdy

.01: 223.40 [Unclear as to whether I want to include considerations of "?" being to be a number & t. final funct into Macrofunct having numeric output. ("TYPE" info). The "Type" info could be an invariable accompaniment of all symbols., so each ~~item~~ has 2 subparams: Type & Value.]

.05 Or I could let all symbols be same Type, & let TM figure out about "Types"

The .05 would making looking for "+" diff't! TM, say remembers all symbols that have occurred, so it can decide if they occur again. (i.e. so t. "=" can be xformd into a Boolean funct. — By using "Types", I ↓ t. no. of Possys Tremendously! This is because t. no. of Numbers, is enormous! A way around this is to use only one sample per example, & ~~[illegible]~~ use 32 bit random integers. Even so, I'd want to give new random nos.
3:34 ; 4:00
for each new example type.

Apt #3 #4

But I was mainly interested in running a ~~[illegible]~~ not-needy clever TSQ — Since my techniques supposed to work w. all

.16 Speaking in general terms: After a new subcorpus has been worked on & ~~[illegible]~~ a soln. for it is obtained: How is Big P.D. updated? First, t. params of old concs are changed.

.17 Def BPD

Second, new ~~phrases~~ concs added to B.P.D. (Big P.D.) w. assoc. params.

.19 Third ~~Next~~, TM_2 can work on ~~[illegible]~~ entire P.D., TSQ to look for better regys. Possibly a complete Recoding, or partial recoding, or Coding of Code.

~~.20~~ Can I use a sort of arby B.P.D. & see how t. new subTSQ

.21 experience changes it?

.22 My impression: In a TM working on MTM probs, there are 2 kinds of sub-Pd's: One for Math probs — in which pc≢1 or φ & other PD's used ~~for~~ to guide heuristic search on Math probs & perhaps other

.25 PD's used ~~[illegible]~~ for NMTM-type Problems.

Noted Before!

To start. .16-.19 + (.19-.21)R. maybe O.k. I'm not sure it addresses .22-.25

SN I had considered a usual Max of 3 or 4 functions to be combined; but this limit could be much exceeded if I had a particular method of combining functions that was often used. (This is a "Meta Prog" idea) One example is "loops", a combination of at least ~~4~~ "functions": ① Counting function ② Main function ③ threshold of termination function (Boolean) ④ Initialization. (Unclear how ④ is a "function" — it could be an input)

.36 Hvr. Check on Peter's Thm. It may be that ① can always be ~~&~~ j:=j+1 & initial value can always be 1. So t. loop functional has input n & function ②. Essentially, what we do is take function ② and add another ~~[illegible]~~ argt. to it, "n".

~~&~~ If ② has ~~[illegible]~~ n argts, we need k different funcks in

9·6·00 Bulg.

$$\left(\begin{array}{l}\vec{x}_0=\vec{z}_0 \text{ (initalzn.)}\\ \vec{x}_{i+1}=\vec{f}(\vec{x}_i)\end{array}\right)$$

(2): Essentialy, $\vec{x}_{i+1} = \vec{f}(\vec{x}_i)$; repeat n times.

.02 A big Q is 224.36: do I need an arby "counting funct" w. an arby form of "stop" threshold? — This would be useful in telling, say, when an approxn. procedure should stop. — But do we need this for Prim Rec. Functs? — See Kleene "Introdn. to Meta Math".

unfillg 225.0221 Ex 5726 cucma.

.02 isn't actually a central Q: t. Q is: to what extent is 224.16–.19 "adequate"?

{ Re: 224.16–.19: Consider various possl. "Solns" to TM's problems.
Can each discoverd Regy be put into t. 224.16 ff forms?
Can what TM lrnd from t. Soln. (t. Solving process) be put into 224.16 form?

In general: 2 parts: (1) soln of problem (2) Info from t. "Soln. process".
An extreme example: say t. problem is $\sqrt{2}=?$ One part of soln is 1.414···
T. other part is how this soln was obtaind: t. number 1.414··· mite be stord as a soln to $\sqrt{2}=?$, since this problem may occur again. Whether or not it is stord depends on cc of storage & cc of computation & expected time before info will be needed.

IN GENERAL in 224.16: This can be done in many ways,
.18 each w. its own "of completeness". In one way, t. B.P.D. consisted of a set of functions, each w. its own pc, & a set of combining rules (functionals) each w. its own pc. In 224.17 (first) t. pc of the functs & functionals are changed. In 224.18 (second) New functs (& perhaps
.22 functionals) are added, each w. its own initial pc. The "New" functs will be Macro functs used in solns to probs & Functs. represented by subnets that are used frequently ("These subnets" — I haven't found TSQ in which they are useful & this
.25 is a source of "Unease".) Re: New functionals: I don't know if this is necy, or if so, how they would be discoverd.
In (third) 224.19, the revision of t. entire B.P.D. Code: This also needs much
.28 work: It may involve discovery of "New functionals" (.22), (.25).

Anyhow .18–.28 is only one way to construct & update B.P.D. How well defined, how efficient, how complete it is, is unclear at present.
.18–.28 is an "Eng" desn of (2) lrng. system. I could go into more detail, or Generalize it more. — Make it Vaguer: More Englishish!
A MAIN Q: Can All t. induction methods I can think of, be fitted into this (.18–.28) Model?

AN APPROACH: Work some (Human-wise) problems & see if I can fit t. techniques into (.18–.28).

9-7.00 Bulg.

SN Consider folg. problem: How would TM have (1) Motivation (2) Ability to work it: Given f(x) from a to b is monotone ↑, to find $\int_a^b f(x)dx$: Discover Simpson's rule: Bounding $\int$ betw. upper & lower limits.

(3) Could TM have motivation & skill to discover higher order versions of this algm when $f'(x)$, $f''(x)$ were monotonic on the interval? ~~[illegible]~~

And: Find order of approxn to use v.s. no. of pts evaluated $\ni$ ε cc was min.

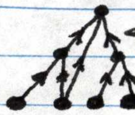
Would TM have to have a kind of "AM" like "motivation" to work on such problems? For the initial problem, the problem definition might be "find $\int_a^b f(x)dx$ within ±.001 & prove this is correct." (or make it "very likely" that it is correct).

Perhaps study "History of Math" to get ideas of / ordered sequences of concepts.

A possy. is that the function nets I've been considering, ~~do not~~ are not always able to ~~[illegible]~~ desc. all solns. to problems. If not, then concs are more general than functions, & I can make desns of problem solns as "conc nets" (as in Sol(89). The updating of the params. will be eas w. function trees)

Woops! There's a big difnce betw a function (which must be a tree) & a conc, which must be a "net" of concs, but a conc net is somewhat more general than a tree! — Tho all function ~~nets~~ trees are conc nets all conc nets are not Trees.

.22



This conc. net: it is not a tree.

.23

Woops! A function net need not be a tree either! (.22-.23) could just as well be a function net!

So it looks like Conc nets & function nets are isomorphic. Neither need be "TREES"

The idea is that any idea (≡ conc) is formed by combining other (earlier) concs. (This is an English statement/descn of the process)

If the concs have simple unconditional pc's, it's easy to update those pc's: They are independent & are obtained exactly as in Z(41) - so pc ∝ no. of times each conc. has been used in ~~[illegible]~~ solns the problems that have been accepted. In the more general case, they are not "indep" but are conditional pc's: The conditions can be arbly complex. I ~~[illegible]~~ Guess updating indep conc. pc's ~~[illegible]~~ may be simpler (viz Z(41)) .... but updating condl. pc's seems much more diflt.

On second thot, updating (function/conc) nets may not be so trivially covered by Z(41) techniques!

→ 227.01

.01: 226.40. Consider Function & Concs. Separately.

First, Functions: To Construct a function: Start at t. top (terminal) function: Using t. corpus of successful Macro functs: Each funct has a pc of being chosen at "top funct", — but we will, at first, want to pool this info, because, at first, SSZ's are small. So, say each funct or connection to t. input, has a certain pc (initially, say, they are all t. same). After t. first funct is chosen, Its inputs have to be chosen, & they, too have t. same D.F. over all of t. functions & (Macro-funct inputs) So, given any Macro-funct, as well as the assoc pc's of the functs & Macro inputs,

Def we can assign a pc to t. construction of that M-funct. (≡ Macrofunct). These probability assignments can be done exactly like in an advanced Version of Z141, with [redacted] "long" definitions.

Def .13 The Pre-corpus, consists of the M-inputs (≡ inputs to t. Macro funct) followed by t. primitive functs. We then define our first funct. It can be one of t. M-inputs, (→ in which case, it becomes one of t. M-functs) or one of t. primitives: If it is a primitive, it is followed by its inputs, which must

.15 be M-inputs. α The next function can have as inputs, any previous function output, or any m-input (Loop Recursion) to .15α. The p.c. of each choice made
= (no of possl. identical choices)/(no. of possl. legal choices)

.17 So. if an input was f_3 & f_3 occurred N_1 times in t. previous code, & there were N_2 legal choices in t. past code that could have been made,

.20 t. pc is N_1/N_2: This is same as normal Z141 coding. (also ≡ "Laplace's Rule")

.13–.20 Tells how to code/get pc of any M-funct. This pc will be indep of t. pc's of any M-functs that this m-funct does not "refer to" (That does not "refer to" in this derivn of that M-function — there can be parallel derivns of this M-funct, that use entirely different M-functs in their derivns).

The isomorphism of Concs & functs can be used to assign pc's to Concs in a formally identical way to M-funct assignment of (.13–.20)

In a Conc. net, each node in t. net is a "combination rule" w. several inputs: t. inputs are "lower order Concs."

.30 In funct. nets: Conditional pc's can be obtained empirically by case counting, but I suspect that this will not give much info usually, since SSZ's are

.32 too small: Probably condl pcs will arise from "reasoning", analogy, etc. → (228.01)

Re: (.13–.20) (T. method of): That gave t. mathemat value of t. pc. To actually get it, each Funct would have assoc. w. it, (its "slots" ≡ "property list") a N_1 & N_2 value (.17–.20) So we could rapidly compute (& update), its pc.

(.13–.20) seems fine for functions & t. same for Concs. Since funct. analysis is simpler than Concs., Do function problems first. — But t. analysis will apply exactly t. same to Conc nets. One big pending problem is Conditional PCs (.30)

9.8.00 Bulg:

.07 Present Branch of Activity : Put this on t. "STACK"

.01: 227.40 : Re: Analogy (227.32) This is common structural properties of concs. Usually a conc will have several poss. dervns. Only using certain of t. dervns, will "structural similarity" w. another conc. be recognized. So holding "parallel codes" can be useful in this way. Also, if we suspect that 2 concs may be analogous, we could examine alternative codings of both of them.

.07 → So: Maybe do this: Do Alg. Inp. using only 227.13–.20 (& possibly t. simple condl pc's of 227.30) See how far I can go w.o. needing more complex concs, than "Functions" or more carefully constructed condl. pc's.

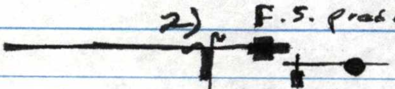
.10 Some Q's: will 227.13–.20 work for only MFM functs, or will it also work for NMFM functs (stochastic Operators)? Note that B.P.D. (c224.18 DEF) is a stoch Operator

.10 is an interesting Q! → Because it deals w. a confusion that I may have had betw. t. pc of a function (which can be deterministic or stochastic) & t. [redacted] params t. derb. t. condl. PC. of a Stoch. Operator. → see 247.19 to at least 249.30 for discn: I'm not sure this was complete soln. to t. ~~[illegible]~~ problem! Stochastic

I usually consider 2 kinds of probilistic operators/functions:

.16 1) T.S. predn. (regressn) (symbols a/o numbers)

.17 2) F.S. predn. (languages) (called "Symbolic Regressn" by Koza)



Give Examples of stoch operators I may be able to discover using models like 227.13–.20. Well, there is, for T.S. predn, Z(41), w. Wolff's Breakthru ~~form~~ (of "reparsing every once in a while"). Note that this is a coding method, & doesn't ~~[illegible]~~ directly give a prediction operator.

The T.S. predn method I have of defining all post fixes of t. corpus, & giving a wtd sum of their predns., is closer to a stoch. operator.

It would seem that t. method of Z(41) could be applied w.o. much modifn, to a corpus of unordered finite strings (a "language") — (Also I have all these ideas on PSG discvy)

A very BIG Q in all of t. forge. To what extent will it be able to deal w. t. "Scaling Problem": The ↑ in no. & variety of available concs: t. necessity of having heurs (condl. pc's) for t. concs to narrow their pc's down to t. application in which they are to ~~be~~ (presently) used.

So: ~~[illegible]~~ To make a TSQ "Acceptable": I have to show how each new needed conc, is derived from older, useful, concs. Also, I have to have condl. pc's for these sub-concs, to show why they were likely to be tried as components in t. (present, relevant) situation — 187.33ff

.36 I think an idea of t. "ob-op algebra" was just for good condl. pc's: One way (not t. most general) to get them is by building up a set of "ob" concs that build on one another hierarchically, to form "categories". These "categories" are then "correlated" w. t. op operators

Avg 3# (kcof → 2936 136
cogc 3860 179
pushing limits of lithography 385#

.01 So we can know under what conds (its more likely) That we should try these operators.
In general, t. rate of Growth of t. "Ob" vocabulary, (conc set) must be at least as rapid as t. growth of t. "Op" conc. set. — These rates Can be "Served" by TM₂ toward approx equality. One Signal that there are "too many OPS per OB" is that t. cc of search for acceptable ops is too hy. — (tho there could be other causes of this last diffy!).

Re: 228.07–.17: This Problem needs clarifn.! I don't have a clear picture of how discovery of Stoch ops is done: how "categorization" (cond'l. pcs for search) fits in, etc.

I may want to do a good review of : A more detaild dcrn of just how I expect this Preliminy TM to work.

Probly best: outline how non-stoch Ops found; First: Random (w. a few kinds of stoch ops: (228.16–.17)

On t. coding of a seq. of MTM problems: (solns. are functions)
The TSQ is A, B, C, D, (each is a set of examples of a function.
Given A, the (operator) soln is F_A.
" A,B " " " " $F_A \hat{} F_B$ where F_A is a dcrn of an operator in terms of prim. funcns.
F_B is a dcrn of an operator in terms of prim. ops & F_A. F_B solves (A, B)

"Similar subtrees" in many cases — coded by ~~similar~~ Similar strings.

.22 (NB) Even w. only 2 functions, F_A & F_B can have common sub-functions
.23 (≡ sub-trees) that will ↑ pc of t. code. →
Given A, B, C, $F_A \hat{} F_B \hat{} F_C$; F_C is a dcrn. of an operator
.25 in terms of prim ops & F_A & F_B. Also, t. remark of .22–.23 holds even more strongly for t. code $F_A \hat{} F_B \hat{} F_C$ than for $F_A \hat{} F_B$. (More likelyhood of (above-threshold-frequency sub-trees) in a larger corpus).
When Common subtrees are found, they are added to t. function list (w. proper
.29 assoc. pc's) for t. construction of new trial functions.

{.22–.23; .25–.29} seems to clarify a lot of my confusion and "unease" about this lack in my SAARB TSQ.

I suspect that t. no. of ~~examples~~ new example types in the SAARB TSQs was far too small for t. effect of .22.23 & .25–.29 to be ~~observable~~ observable!

Also Note: It is possl. to "not need" the ability to detect sub-trees, of .22–.29 if one puts examples in t. TSQ's that use those subtrees as solns. (No! In t present system, this would not work; Even if I gave TM a problem in which a particular sub-tree was a soln. (say t. sub-tree was f_0). Then f_0 would never be a M-funct. — we would (230.01

9.9.00 Bulg.

.01:229.40 have to have an Ob recognize when Fo was relevant, & have this (Ob, Fo) object, adjoined (added on) to M-function. — Or have a general reformulation (total code revision) to fit Fo onto t. M-funct. — But in general, Fo would never (by itself), be an ~~a~~ M-funct — (unless it was ~~a~~ t. soln. to t. first ~~[illegible]~~ set of probs. in t. ~~T~~ TSQ.

In fact: I had this idea of teaching TM definitions (in Algebra) as ~~[illegible]~~ an impt part of t. ~~T~~ TSQ. The this could, indeed, simplify TM's learning of impt. concs: I think t. mechanism of finding "common traces" [Or much ~~[illegible]~~ more diff: OSL of such concs!] would be necy. ~~[illegible]~~ before TM could use those concs in solving new problems.

T. Main TROUBLE w. OSL, is that it requires ~~e~~ either an enormous amount of searching (for functions), or some kind of INFO-Retrieval — (≈ "associative" Memory).

One approach to t. OSL problem would be to look at cases of subtrees that I'd want TM to lrn, & try to find ways to ↓ cost of such srch — either ~~either~~ by ↑ of pc a/o by some hardware method to cheapen associative srch — or whatever

Case Based Lrng is OSL, but I think it models t. Concious aspects of t. process, only. There are prob'ly many strong, idiosyncratic, (individualistic) techniques of categorizn, association, that are not "concious" — perhaps many kinds of "Analogy" (≡ Structural Similarity).

.22 One kind of OSL that may be common: Say I give TM examples of a "named" ~~named~~ conc. to learn ("Definition lrng"). ~~The [illegible]~~ T. concs that are lrnd in this way, occupy ~~a~~ special ~~[illegible]~~ kinds of "positions" in t. resultant M-funct. When new problems arise, TM looks at those positions as being good cands for OSL.

(a mild Genzn. of .22) Another approach to OSL! That the M-functs are normally "factored" to some extent, so we do have Obs & Ops & correlns betw. t. two — So TM could do a search over all ops that had a high correlated Ob — in t. OSL srch. i.e. "which trick (ops) have been useful in t. past".?

.33 An elementary TSQ that could be very revealing! : MTM tsq, small amount of simple lrng (say ANL). Then we teach it a defn: Then we give it a problem in which that defn. would help solve it. This is OSL, & I am concerned w. t. mechanics of how TM would deal ~~with~~ w. it. I think t. problem is easier than looking for common sub-traces (— because there are many many sub-traces to look at — in this Case Based lrng, OSL, we only search ~~examine~~ a much smaller part of t. corpus for relevant cases (I think!) (231.01

9.10.00 Bulg 1

.01: 230.40 In general, [redacted] Lrng. definitions involve creating an OB. (funct. w. Boolian Output). Normally, just lrng a definition is not very useful in itself. It can become a useful OB, only after it has been correlated w. corresponding OPS. (or a corresponding OP.)

(SN) ~~I~~ I have been considering only HTM problems that were "instantaneous" — as opposd to problems involving a sequence of operations: usually obs followed by ops: The obs told "what to do"; t. OPS "did them". Or, we can think of t. "ob·op" combinn as a single OP, ż just have a sequence of ops.

Re: Lrng "Definitions" ~~ǂ~~, ż OSL: If an Ob has ever been "useful," it is a Cand. for OSL. Hvr, certain "Utilities" are more "useful" than others. Useful in a taught "Definition" will have a higher proby of being a Cand. for OSL. Hvr, in general, when a problem is presented to TM, all of t. obs that TM has lrnd, are applied to it to understand it (t. problem) in as many ~~possib~~ ways as possl. Altho t. no. of OBs that TM haves will ↑ (perhaps linearly) w. ~~the~~ time; t. (amount of time) needed to find a match, will probably ↑ much more slowly.

(.19) I'm thinking of "OB ~~recog~~ recognition" as simple sub-string recognition! That an OB is represented by a string or set of strings. To recognize ~~anyth~~ one of a set of cand. strings in a longer string, is a well defined problem. T. cand. strings can be in lexical order. — So recognition is like my "soln" of t. DNA sequencing problem. (use of B trees): Other relevant methods: Zato Coding, Hash coding (see Knuth's Volumes ǂ on this last — or perhaps he has refs to other work on directly related to t. problem of (.19)) Maybe ask Simon!. Ask Alex.

Zato Coding
Hash Coding!
See Knuth for

(.19) occurs in Z(4) in reparsing t. corpus.

Since .19 appears to be an impt., often occurring operation: perhaps Special H.W. can be used. For Decrypting such string matching H.W. is available — but I don't know if t. H.W. is always revised w.r.t. "Moores law": A large amount of Parallelism can be used, ż of course, any very large computer will be highly || (say Beowulf).

Hvr. Obs can be Boolean combns. like AND, OR; NOT. Still, recognition of t. components of t. Boolean Expressn. can be done via (.19).

Also, there are Hierarchical means for Recognizing OBs. — so a part of a ob is recognized, so a large set of Cands are now ~~[illegible]~~ "super Cands" — which ~~narrows bown~~ narrows down t. problem. 232.01

9.11.00 Bulg

.01: 23.40: Another technique for Ob recognition is that used in LZ coding. (very fast! & very "cheap" very little RAM needed!)
— Hvr, in contrast to LZ, I would also store info on how many times each substring occurred.

SO! t. OSL problem may not be so "Computationally intractable"

T. theory is more than a bit simplified! In many cases, an OB is far more complex than just ~~stry~~ sub-string matching. I.E. to recognize that a certain [redacted] substring is an equation, or that it's a linear ODE, etc. is certainly more complex than string matching! A More Realistic Model would be Pandemonium — in which each OB looks (in ||) for things that are "a case of it". A Q would be t. cc involved in this. Also, whether one mite not be able to save cc by doing sequential ~~testing for Categorizing.~~

Do a bunch of Examples of Definitions lrnd & perhaps [redacted] other forms of Concs. that have to be lrnd by OSL. Go thru both t. lrng of t. Conc. & t. ~~app~~ ~~[illegible]~~ later applicn. of t. Conc.

Normally, we give TM a new chunk of corpus (≡ "sub-corpus"). Say it learns to work probs of t. corpus; ~~then~~ (even w.o. recognizing t. new sub-corpus, this problem solving will much ↑ pc of coding.) But next step is to "recognize" t. new section of corpus. This is an "Ob Job". — ~~[illegible]~~ presumably made much easier if some of t. nec'y Concs have been lrnd as "definitions".

How to use Neg cases: We just have to categorize t. input into 1 of 2 categories. TM assigns pc's to each category: t. score is t. max value of $\prod p_i$; p_i being t. proby assigned to t. "Rite" category.

O.k. so: Defn. lrng ~~[illegible]~~ problem is given:
α, examples_i, Def. ← Say this is format for learning that examples_i is a case of α. ~~[illegible]~~ TM is then given queries of form "?, examples_j, Def."
or perhaps Def, α, examples_i, $\binom{N}{Y}$"; then Query is "Def, α, examples_j, ?".
Say TM finds Ob_α that gives rite answers for the [examples_i] sub-corpus. We also have to tie ob_α to "α" and "Def": Assoc. w. α & Def are Boolean operators that recognize them, B_α, & B_{Def}: so t. final operator is Ob_α ~~and~~ And B_α And B_{Def}. (~~[illegible]~~ I've ommited some details, here). Say OP_M is t. previous M-Function — so we want t. new M-funct to be Ob_α ~~when~~ if B_α and B_{Def}; else OP_M ~~[illegible]~~.

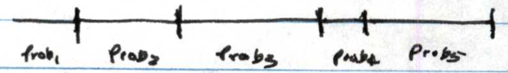
In general, t. format t. M-functs (at first) will be a list of ifs: See 235.16 for criticism

.37 1 If ob_1 then OP_1
2 If ob_2 " OP_2
⋮ ~~etc~~.
n If ob_n then op_n.

234.01

9.12.00
Bulg.

.01 : 232.40 : T. form of Soln. of [redacted] 232.37 –.40 is o.k. for a certain type of Corpus: One that consists of a sequence of separate, clearly defined, problem types.

.02 

[I think that each of these "problem edges" ~~E~~ gives TM₂ some info about when to spend much time on a "single problem", rather than t. entire past corpus & expected future corpus.

This form of Corpus is quite "restricted" but none, t. less, impt., difficult problems can be put into this sequence. A big Q is: Can I put t. Understanding of English text into this formalism?

.08 Superficially, Obj can be a device that counts the no. of probs that have occurd this far. — but I can outlaw this by making TM regard the problems as "Unordered". — by giving TM no functions that could detect this order. On t. other hand, I may want TM to be able to detect "Recency": So if I teach it a defn., then in t. near future of that defn, it will expect problems using it. (This is a rather broad "hint" — perhaps an "unwanted one.")

I probably can arrange so that TM ~~will not~~ cannot do .08: — but I'd like to know just what/other restrictions on TM's capability this would imply!.

Actually [redacted] if TM did .08, it would be unable to solve new problems! In fact, the soln. form of [redacted] 232.37 – 40 is also very limiting! If I give it a new problem that is not of t. same type as [redacted] any of its training problems, it will not know what to do! — Yet, after

.22 it has solved a seq. of diff't problems, it should in some sense, be very smart)

● Well, Look at it this way! ● Say TM has worked t. TSQ up to a certain

~~.23~~ point. We then give it a problem: "string ?". How should it best get a P.D. [redacted] for "?"? Well, presumably it trys to get short codes for "string α", — α being various [redacted] possi. continuns of "string".

.~~23~~ 27 Actually t. situation at .~~[redacted]~~ 24 isn't so difent. from t. situation when

.28 TM is given t. first ~~problem~~ example of a new problem set.

In .27–.28 TM has to get a pgm. to produce $\alpha^\frown\beta$ (where "β" will be eventually replaced by "?" in t. query aspect of t. trng.) . . . .

.33 Well; 2 comments: ① If a person were given a totally new problem under

.34 those circumstances, he wouldn't know what to do. 235.10 ② The [redacted] TSQ given to TM was really of t. form [Q_i, A_i] — but it was ~~broken~~ into distinguishable ~~parts~~ subsets [Q_{ij}, A_{ij}] "j" is t. name of one subset.

Consider t. TSQ: Solve linear eqns, Solve quad eqns, solve cubic eqns. We can consider this seq. as 3 subseqs or 1 big seq.

235.01

9.13.00
Polya

9.13.00 Joyce
~~783 8785~~

.05 :234.40 i.e. we can use t. info about t. 3 divisions or not. In t. [redacted] latter case, we mite regard them all as "equations" to solve.

Also, even when .TM clearly uses t. structure (234.02L) and t. recognition Ob, 232.37–.40, we can regard TM as solving 1 category of problem, with 232.37–.40 being an analysis of t. problem that tells TM what to do. — Hvr, I guess "232.37–.40" is a poor analysis of t. problem: it doesn't tell how to extrapolate itself into new domains: It is rather A.H. (even tho it has a lot of "learning in it" (234.22).

.10 :234.34 232.33–.34 may be t. rite idea! — That unless TM has been specifically taught to generalize input analysis, it will not be able to do it — & shouldn't be expected to be able to do it!

So, one Q is: 234.02L is a desirable structure for lrng: & it does get info init (234.22) ~~[illegible]~~ w. t. soln. format 232.37–.40: Can We somehow use that form to get more useful lrng?

.16 A Very Serious Criticism of t. 232.37–.40 soln: T. search for OP & t. srch for Ob are separate & not related: Normally, I'd like t. input to create a condl p.d. for t. search of OP's: This should ↓ cc of srch tremendously! It's a really broad area for creation of Heurs!

So: while 232.37–.40 mite work for a relatively small corpus, it would not have enuf heurs in it to scale up to a large corpus. —

As t. corpus Grows, t. no. of concs available grows, — w.o any narrowing down
.23 of t. search that would be obtained by "looking at t. problem". → (236.16

.24 To t. extent that 232.37–.40 doesn't "tie t. lino" on MCT, it is deficient.

Ideally, TM should use info on past (corrdn/relations) betw. Ob & assoc'd OPs,
.26 to help in each new search for an OP.

.27 One way to do .24–.26: After TM has (using 234.02L & 232.37) done a reasonably large corpus of example probs., TM should try to find useful, (predictive)
.29 (problistic) relations, betw. t. OPs & their corresponding obs. This "scales"

O.K. — Since t. corpus is small when one just uses (232.37, 234.02L)

Later, one really needs t. relations of .27–.29 to ↓ t. cost of srch.

Unfortly, for a infant TM, discovery relations of t. sort needed in
.33 .27–.29, may be too difclt. — It may be necy to put this in as AH.

Another way to deal w. this! Use t. strict 232.37, 234.02L TSQ form, until TM has acquired enuf "smarts" to do t. induction required in .27–.29. — Hvr, unless one has very great IPC (maybe 10^15 bits/sec)? it may not be practical to do this, because before TM is smart enuf to do .27–.29, it will run into "scaling problems" — I could do t. partly AH thing of .33 to help — to avoid need of super hy IPC. — equivt. to A.H., would be different degrees of BROADNESS of HINTS. 236.01

9·~~14~~·00
9.14.00 Bulg

.01: 235.40 : So, O.K. Say I have this set of ~~[illegible]~~ [problem$_i$, ob-op soln$_i$]
.02 D ~~of~~ corpus ~~of~~ [illegible] & solns. of t. "$\overset{STSQ}{\text{Simplified TSQ}}$" (of 232.37 – 40, 234.02L)
.03 For this set of pairs (of .01R) I try to find a problisht operator (as in 235.27 – 29) to ↓ t. p.c. of that set of pairs. I could do it in an operator form of solving [Q$_i$, A$_i$] in which I assume "no info"
.06 in t. [Q$_i$] set.

So .03 ff sounds fine: Note t. stuff on hy IPC & "Hints" of 235.23 – 40

T. original problem set (TSQ) ~~&~~ "STSQ" was a [Q$_i$ A$_i$] form, w. "clumps" ~~[illegible]~~ [Q$_{ij}$, A$_{ij}$] ("j" is a "clump name"). So we ended up w.
.11 ↳ {Q$_i$, OP$_i$}$_{i=1..n}$ & we want to extrapolate this sort w/l work w. Q$_{n+1}$
t. problem of .11 is just like any other [Q$_i$ A$_i$] prob, except that up until now, I've been doing MTM probs & this one is NMTM (problisht). (OB)

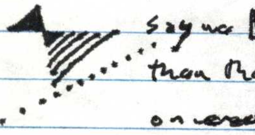
→ In .11, I $\overset{\text{should}}{\text{~~[illegible]~~}}$ be able to use OP$_i$ w.o. t. associated "recognizer" for t. "j" group.

.16 This gen'l. ("IM$_2$-like") method is also useful for incorporating hours [redacted] info t. Lsrch, [redacted] to deal w. "Scaling" — it deals w. 235.16 – 23

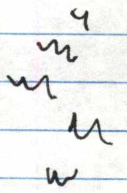
What I'm doing in .02 is a "recoding" of t. "soln" of a (.01R).

STSQ of .02 does give a legit. code for it's corpus: Th. code is one
.20 single final M-funct, along w. t. recognition Ob of 2~~3~~32.37 – 40.
Hvr, this code can be much compressed! (.01) is one way to do it.
The corpus + t. code of .20, ~~[illegible]~~ can create the data string of (.01R).

From (.01R), any stoch operator that can get a good pd of soln$_i$ from problem$_i$ as input, could ↑ t. p.c. of t. M-funct of .20. One way this works:



Say we ~~[illegible]~~ are able to ↑ t. pc of one of the prob: soln$_i$ pairs of .01R: then the pc of t. final M-funct is also ↑, since t. pc of t. final M-funct depends on ~~[illegible]~~ t. pc of each previous M funct. If all of t. ~~#~~ component M functs have their pc ↑, t. final M funct has it pc ↑ even more (perhaps n cost of final M funct is ≈ Σ n costs of all component M-functs.



How the stoch operator ↑ pc of component M-functs: It looks at t. problem$_i$ & makes a kind of p.d. on possib solns. This p.d. can be very direct, or it can work by making narrower conditional p.d's for t. functions that are components of soln$_i$. ("these "conditionals" can be on ~~[illegible]~~ t. partly-constructed ~~[illegible]~~ funct for soln$_i$.) — in either case, t.
.35 pc of soln$_i$ would ↑.

Next, I want to go into more detail on t. "entire system":
.37 First, we have this set of sets of [Q$_{ij}$, A$_{ij}$]. "j" is named subset of QA's of common "soln". At first pass we do [Q$_{i1}$, A$_{i1}$] first: we do an Lsrch
.39 over primitive functs to find ∃ F$_1$ ∋ F$_1$(Q$_{i1}$) ≅ A$_{i1}$. (Since this is at first 237.01

.01: 236.40 'a MTM tsQ; "ā" "13" = " in 236.39)

When we solve problem 1, we go to problem 2. We solve it by Lsrch, using primitives + F_1, to obtain F_2. For the entire corpus $[Q_{ij}, A_{ij}]$ $j=1,2$

.04 F_1 works about ½ the time; F_2 works about ½ the time. This gives a stochastic Operator: which we try to improve, by finding obs that can look at a Q_{ij} & tell if $j=1$ or 2, so it can tell if F_1 or F_2

.07 is applicable.

We then do $[Q_{ij}, A_{ij}]$ for $j=3$ and integrate it into a single Grand Funct (≡ M-Funct) as in .04. .... we continue this ~~[illegible]~~ w. higher and higher values of j, until the cc of solns is too hy: They are too hy, because we have too many functions to construct our new trial function from — & these functs all have about same pc.

.12

a/o (Non-Conventional) TM could do analysis of Wholistic Medicine (it may be that there is almost always no good data to conclude much — but TM could need a lot!)

.13 [.... Next, we have to go into Next Phase of (236.01 – .06): We want to find

.14 regularities in the code for the operator that does $[Q_{ij}, A_{ij}]$

.15 SN Perhaps before we go to .13: [In the soln. process of 236.37–237.12 I had been thinking of just using functions that were primitive or were F_j types of functs, or Obj types that identified j of $[Q_{ij}]$. I think I will need more:

Dat I will need simple OSL: Also, OSL (& MSL (Multiple shot ~~[illegible]~~ lrng)) & ~~[illegible]~~

.18 learning from subnets (≡ subfunctions)........

SN: TM could easily lrn meaning/criterion for "Soln to eq" & Soln.(X, X+7) = −7, T (True): How would it go about solving "Soln(X, X+7) = ?, T"? IN FACT, any INV problem can be put to TM in this form, & the best way to solve it is Lsrch (if TM has proper hcws!)

SN Perhaps Make "Graph of My Recent Prgs in TM work, so I know where I am, & what needs be done.....

I think the critical work now is .15 — .18 & then .13 – .14.

.28 In 236.37 – 237.12 we can have just one "i" value (≡ 1, say) for all ~~[illegible]~~ j

This will work if we use long random nos. for examples. — This tech will prob'ly not

.30 work in ~~non~~ non-numerical domains (& prob'ly wouldn't work in "Number Theory" either!)

236.37–237.12 is a kind of Gross Elementalizn of the Task. It is of some interest in that it appears able to do some lrng. — (perhaps at a serious level?)

(.13 – .14) & (.15 – .18) are improvements. (.15 – .18) is conceptually simple, but the details of implementation could be very diff't to get it to work w. ~~[illegible]~~ acceptable cc.

.36 (.13 – .14) = (236.01 – .06): Using the idea of (.28 – .30) (i = 1 only), we have this set of $[Q_j, Op_j]$ pairs (Op_j = $Mfunct_j$), & we want to find ~~[illegible]~~ a good ~~[illegible]~~ stoch operator $\ni F(Q_j) \approx Mfunct_j$

.39 ~~=~~ Or $\ni$ ~~$F(Op_j)$~~ $\prod_j P(Mfunct_j \mid Q_j)$ is max (Mult by pc of F.) 238.01 .13 max

9.16.00
Bulg

.01 : 237.40: Actually, we should save much cc by doing 237.36-.39
After every "j" problem is solved; since it should make soln. of t.
next "j+1" problem may easier (See 236.16-.35 for discussion of why & how)
Using the i=1 only idea of 237.28-.30, we can think of TM's
work as "linear regressn" "Sequential predn", in t. folg. [redacted] way:

In linear predn; we use info upto $j=n$ to ~~predict~~ get a pd. for x_{n+1}.
After ~~[illegible]~~ we are given t. true x_{n+1}, we use this to get t. pd. for x_{n+2}, etc.
~~In TM we~~ In all cases, t. pd for x_j is based on t knowledge of
of t. true x_i values for $i=1/j-1$.

.10 [α] In TM, similarly — knowing the Mfunct$_i$'s for $i=1/j$ and
the Q_i's for $i=1/j$; we get a d.f. for Mfunct$_{j+1}$. We use this
d.f. over Mfunct$_{j+1}$ to do an Lsrch to find t. ~~[illegible]~~ "actual" Mfunct$_{j+1}$.
.13 we then do $j \leftarrow j+1$ & loop to (α) (.10)

.10-.13 is very interesting! Somehow we are able to get this
d.f. for Mfunct$_{j+1}$, that incorporates all of our previous knowledge
of t. [Q_i, Mfunct$_i$] for $i=1/j$!!

MF = Macrofunct

.17 Def MF It would seem that after t ~~[illegible]~~ search for t. MF$_{j+1}$, & finding MF$_{j+1}$)
there is no need to find a way to find j as a funct of Q_j. —
i.e. finding an ob that identifies j from Q_j. — we don't
.20 need this in .10-.13!

Well, what happens after we find MF$_j$, and we are then given a
problem Q_5 ($j>5$ & Q_5 is one of t previously done problems).
Then, presumably, TM has t. pair (Q_5, MF$_5$) available to it, so it
will give ~~[illegible]~~ P(MF$_5$ | Q_5) a very hy pc.

.25 → My impression is that MF$_j$ doesn't need to include ~~[illegible]~~
info on ~~[illegible]~~ how to work ~~[illegible]~~ problems Q_i for $i<j$! MF$_j$ need solve Q_j only (N.B. .32-.34)
[.25] has to be lookd into! Along w. .17-.20, this gives a
Quite different picture of TM than 236.37-237.12! It looks
.29 less el., more capable, but I'm not sure its total
.295 One fear is that my "Stochastic operator" Model of a condl ~~[illegible]~~ d.f., is flawd — that
.31 it may assume that there is No Info in the [Q_i] set.

→ see 241.01 – 242.05 for good understanding of t. "Stoch Operator" Model.

.32 Actually, in .25, ~~[illegible]~~ t. funct that finds i as a funct of Q_i need Not
be very ~~[illegible]~~ SMART (~~[illegible]~~ nor need it have useful info) — All it has to do is find out
.34 how Q_i is different from all $Q_{i'}$ for $i'<i$ (!) — So perhaps
(.10-.13) seems much better, more direct than (236.37-237.12)

[Hvr. N.B. 236.37-237.12 was meant as a prelim. clzn. of TM: A kind of
"Study problem": In .10-.13 Getting t. D.f. for MF$_{j+1}$ from info in [Q_i, MF$_i$]
$i=1/j$ is not so easy! — But it may be ~~[illegible]~~
always eventually, necy. [.295-.31] is a possl. worry.

239.01

9.17.00 Bulb 3.60 − (.44) = 3.16

so 3.16 v. is ok.

3.16

22 Ω at 3 volts

$\frac{3}{22} = \frac{1}{7} = 140$ ma. 600 ma hrs → ≈4 hrs.

.01 : 238.40 : I'm really not sure how impt t. Q at 238.295–.31 is. (on info in [Qi] set) — see 241.01–242.05

Perhaps drop it for a while & see if I can continue w.o. answering it.

.02 O.K. Say we have this [Qi, MFi] i=1/n ! We use it to obtain [redacted] P(MF$_{n+1}$ | Q$_{n+1}$). for v.g. version

I think I'll have to see how I would work such a problem; To get some feel for how T.M. would do it.

.06 A Perhaps Relevant Idea: for a given sequential corpus, "a", we can have a "Ideal Summarizing Machine, M", so that Random inputs to M, give ~~continuations of~~ outputs w. pc's ≡ pc's of possl. continns. of "a". M is a complete summary of all regys in "a".

.09 Normally, "Summary Machines" are approximations of this ideal summarizing Machine; → 242.07

An interesting Q is: [redacted] if we concat. a string α to "a", how does t. Summarizing machine M, get modified? Well, if α is not very long, we simply put lots of random inputs into [redacted] M, until they produce α, & continuations of α. These continuations of α will have t. pc of continuations of a⌢α.

There is an corresponding "Summary Machine" assoc. w. t. [redacted] "Prob. of an unordered set of strings" I don't immediately remember how this works: See that Sol 99: "T. Computer Journal": "Two kinds of Prob. induction" → 241.01–242.05 13 v.g. on Summary Machines.

Anyway an approx (summary machine) for a t. ~~unordered~~ bag of strings: would be a stoch lang. For PSG's, (stochastic) also 242.07

Each such lang is defined by a set of production rules & prob. choice probys.

T. prodn. rules correspond to NT's (which correspond to sub-languages — or sub-grammars)

.21 When we add new data to t. Bag: we usually modify t. Grammar by ① changing t. fn. probys ② Modifying t. Grammar by adding new NTS & assoc choice pc's.

.23 ③ More diff. & less likely: Deleting certain NT's & adding others. (Harder to do).

In any case, we end up w. a new Summarizing Machine (≡ P.D.)

.02 is t. problem of updating a "Summarizing Mach.", Hvr., its difnt from normal updating of a stoch Gramm, because its a Conditional P.D. that has to be updated. O.K., but ≈ t. same things in (.21–.23) hold — we want to change pc's (if any), add new NT's or functions & perhaps revise t. [redacted] structure of t. MF. ⋮

More specifically: When a new Qi Oi is given, we understand a new operator a/o try to combine t. new Opi into t. old one in a compressive way; a/o revise t. entire system in a compressive way.

A GOOD APPROACH: Watch how I work a specific problem of this sort: Express my soln. in t. Progg. Language ("concs"), then perhaps Genz.

.35 Hvr: My impression is that Demonstration of t. (kind of behavior I'm interested in,) ← see 240.03 for what this means! can only occur w. a fairly smart T.M., (i.e. a rather long TSQ). T. regys required are rather complex & invisible (in) via a small corpus. Only by A.H. insertion could t. needed concs. be visible in a small corpus.

240.01

9.18.00 Bulg

I guess Q_i^1 is the ~~symbolic~~ part & Q_i^2 is the numeric part

.01: 239.40 : 239.35 is another example of a kind of TM behavior that cannot occur until TM has reached a certain "sophistication."

.03 Specifically, the behavior "looked for" in 239.35 : After a fair no. of $\vec{Q}_i$, A_i pairs <in which Q_i is of form Q_i^1, Q_i^2, and TM is able to work problems w. input Q_i^1, $\tilde{Q}_i^2$: here Q_i^1 is an input that it has trained on: $\tilde{Q}_i^2$ is part of the input that is New. ~~&~~ it is able to get the proper $\tilde{A}_i$.> TM is given an $\vec{Q}_{n+1}$ in which both Q_{n+1}^1 & Q_{n+1}^2 are entirely new to it, & it is able to get a

.08 "reasonable" pd. for A_{n+1}.

9.21.00 Looking at 241.01→.05 ! In particular, 241.20→40 and [proof of adequacy 242.01-05], It is clear that the $[Q_i, A_i]$ T.S.Q. is adequate, & that making stochastic functions $P(A|Q)$ using ALP should be adequate. — That the only reason it didn't seem to work, was that ~~my~~ the $P(A|Q)$ functional forms I was considering, were too limited — Not "Universal" — or not able to express the needed regularity, (Universality is usually unnecy)

(238.10-13) & (241.01-242.05) seem to tell how to do TSQ's !

I had ~~x~~ 3 kinds of Induction that I wanted to demonstrate in TSQ's:

1) Simple acquisition of concs that ~~w~~ were solns (MFuncts) as in the Soumb TSQ's

2) Finding subfuncts / sub ~~■~~ nets (not necly trees) & using them for new induction.

2.5) doing 2 for both OSL & Multiple MSC.

3) A more elaborate kind : say (.03 - .08) (≡ 239.35 - 40) This involves NMTM truely problistic predn.

(.03 - .08) seems to be a v.g. example

.22 → Look at (.03 - .08) say $\vec{Q}_i = S_i, \vec{N}_i$ N_i is Numeric (random) part. S_i is symbolic part

Say F_i is the [redacted] functional soln. to $\vec{Q}_i$; so $F_i(\vec{N}_i) = A_i$.

~~we~~ after solving several (n) Problems, we have this set of pairs:

■ $[S_i, F_i]$ $i = 1/n$. My Earlier, simpler ideas ~~[illegible] allowed F_{n+1} to~~ gave a pd. for F_{n+1} based only on the unordered set $[F_i]$ $i = 1/n$.

Now I want the P.d. for F_{n+1} to ~~[illegible]~~ include all of the info In $[S_i, F_i]$

Ideally, this can be done by a (set of) function(s) that map $S_i \to F_i$ $(i = 1/n)$.

.29 The map is from a kind of string, to a function.

Superficially .29 would seem to apply only to ANL, but in theory, it could cover solns of any kinds of equations. — so, in theory, TM should be able to learn assignably diff't task via this ~~t~~sq. type.

.34 Bookmark (where I was) ! 1) Trying to do simple TSQ 2) Trying to devise

.35 simplest possl. Learning machine that would work on any TSQ's. → ~~[illegible]~~ 243.09 244.37

The recent (241.01-242.05) stuff makes it clear that a pure $[Q_i, A_i]$ TSQ would be always "solvable".

The idea of (.22 - .29) of my Symb type tsq as a special kind of Q, A tsq is very nice

243.01

9.19.00 Bulg.

.01 CONDITIONAL PROBTY!

2 forms of Data: [redacted]

Convergence thm 242.01

On Conditional (probty / complexity). It should be possl to devise a thm like S78T3 for Conditional probability! This would seem to be closely related to MCT. Anyway, say some corpus has been generated by some condl. Pd:

$P(y|x)$. T. corpus itself is $[x_i, y_i]$. If we assume that (x_i, y_i) pairs in t. corpus are unorderd, then this is an example of Bag prediction (≡ unorderd set of finite objects)

Actually, we'd need $P(x)$ as well as $P(y|x)$ to generate t. corpus. Will t. error bound in [redacted] $E(P^M(y,x) - P(y,x))^2$ be indip of t. x d.f. ?

$P(y) = P(y|x) \cdot P(x)$ If $P(y|x)$ & $P(x)$ are finitely desrbable then $P(y)$ is also "finitely desrbable".

Hvr, there is also an approxn of $P(y|x)$ that is "indip" of $P(x)$ — that does not use info in t. X d.f. It tries to find a $P^R(y|x) \ni$ $E\,(P^R(y_i|x_i) - P(y_i|x_i))^2$ is min ?

More exactly, it trys to find P^R's w. ~~short~~ desrns $\ni$

.16 $\prod_i P^R(y_i|x_i) = \max$ or, more ~~exactly~~ $\boxed{2^{-|R|} \cdot \prod_i P^R(y_i|x_i) = \max.}$

.16 is the form that doesn't much use $P(x)$ info. (?)

.17 Finding a $P(x,y)$ assoc. w. t. corpus $[x_i, y_i]$ is an (apparently)

.18 difrnt way (better way?) to get t. approxn to $P(y|x)$

Q: does $P(x,y)$ imply a $P(x|y)$? : (Yes!)

Note that $P(x,y)$ (implies / gives) both $P(x)$ & $P(y)$.

.21 $\boxed{9.21.00}$ We are dealing w. 2 difrnt Goals for our model! In once, we want $P^R(y|x) \ni$ .16 is true i.e. $\boxed{2^{-|R|} \cdot \prod_i P^R(y_i|x_i) = \max}$

in .17–.18 " $\ni$ $2^{-|R|} \prod_i P(x_i)\, P^R(y_i|x_i) = \max.$ ≡

~~So, taking logs in both cases; the first way gives wt. to all x_i.~~

~~The second wts them in accord w.~~ or .17–.18 is $\prod_i P(x_i) \cdot 2^{-|R|} \prod_i P^R(y_i|x_i) = \max$

It differs only by t. factor $\prod_i P(x_i)$ from .16

So in .17–.18 we have t. same conds. as .16, except that t. model must also give a good Pd. for $[P(x_i)]$. When the $[x_i]$ are from my TSQ, the $P(x_i)$ d.f. is somewhat "A.H." & is related in a weird way to $P(y_i|x_i)$, so I may not want a TM to try to model $P(x_i)$ — particularly at a cost of a less good $P(y_i|x_i)$

So! for t. present, $\boxed{\text{use .16}}$: It also has t. advantage that $P(y_i|x_i)$, (which is what I really want) is obtained in a direct way: in .17–.18, $P(y_i|x_i)$ is obtaind from ~~[illegible]~~ $P(x_i, y_i)$, the joint distrbn, and involves division by $P(x)$. $P(y|x) = P(x,y)/P(x)$: $P(x)$ is obtaind from $P(x,y)$ by ~~[illegible]~~ integration or summation

$P(x) = \int P(x,y)\,dy$ (or equivt Σ form.)

There are impt. situations in which .17–.18 is better than .16 — in which t. $[x_i, y_i]$ corpus is from R.W. entirely & not modulated by my "Heavy (A.H.) Hand"! (242.01

9.21.00 Bulg.

"Summarizing Machines" (.07)

.01: 241.40: The convergence theorem for $P(x,y)$ (or $P(x)\cdot P(y|x)$) is about the same as that for Bag of strings. In fact the "Bag of strings" convergence thm ~~[illegible]~~ follows ~~[illegible]~~ as a corrolary of .01, when x = constant or x = Λ null

Probably best look at present proof of Bag of strings convergence thm
.05 (in ~~[illegible]~~ MCT files) to devise proper proof for .01. (.15)

.07: 239.09 "Summarizing Machines" that are theoretically complete, need never backtrack.
~~[illegible]~~: The only reason we ever backtrack is that our "summary" is imperfect.

On convergence of $E\sum_i \left(\ln\frac{\delta_i}{\delta_i'}\right)^2 = E\sum_i \left(\ln\frac{\delta_i}{\delta_i'}\right)^2$:
~~Also~~, for S78T3 ~~[illegible]~~ and for Bag of strings:
we do have constraint that $\frac{P_i}{P_i'} > c$ and the convexity of ln.
~~[illegible]~~ — that limits excursions of δ_i from δ_i'.

.15: (.05) On 241.01–.40: In the case of 241.16: what is the default, (A.H.) dcvn.?
we need this to tell if we are making progress & how much progress.
Also, how does QA system work, when there is a body of data to use for reference (e.g. an Encyclopedia)?
MCT would seem to hold the key!

In much of my thinking in 241.01 — .40: I had been thinking in terms of "MTM", in which ~~[illegible]~~ there was one formula that successfully did all predictions, & the next best formula was very far behind. For a small corpus, perhaps this is not true... that in such a case, a more "soft" probilistic dcvn would have higer PC. — This is true in MTM w. small corpus. There are 2 aspects of "Corpus Size". One is the Numeric part: easy to get it to be effectively large by using long random nos. Second is the symbolic part: No easy way to ↑ size.

If $Q_i \to A_i$ is 1 to 1 then there is ~~a~~ single code for the corpus.

If $Q_i \to A_i$ is 1 to ~~[illegible]~~ many, ~~there will be a code~~ then there will be many codes — ~~[illegible]~~ anyone of which does the corpus exactly.

The normal way to think about this is eq. 241.16. ~~Each poss~~ ~~There is a p.d. for each~~ for each Q_i, there is a p.d. of possl. A_i's.
In a MTM corpus, of much length, these P.D.'s are ~~very~~ narrow. — ~~[illegible]~~
Almost all wt. on One A_i, but all other ~~[illegible]~~ ~~[illegible]~~ A_i's have non-zero wt., since they have possl. codes.

.36 So, ~~[illegible]~~ in a NMTM corpus; say ~~[illegible]~~ A_{10} has ~~[illegible]~~ mainly 2 ~~[illegible]~~ "possl." Q_{10}'s: Q_{10}^1 & Q_{10}^2. Then if they are of = p.c. then both will have codes of about same length, for that Q_{10} input. Essentially we
.39 have a ~~[illegible]~~ Prob. machine w. 2 inputs. One input is Q_i, the other 243.01
~~244.01~~

.01 ; 242.40 is "random" input, or just all possl. inputs. Its this "Random" input [redacted]

.02 whose ~~code length~~ we're ~~interested in~~ [redacted]

I think [242.36–.40; 24~~3~~.0~~4~~–.08] is t. rite way to understand this stuff. → "Actually, there are 3 inputs to t. machine: ① an input that defines t. machine behavior: Its length is |R| of 241.16 ② The Q_i input of 242.39, ③ t. "random" input of 244.01–.02. R

"Epistly" speaking we want an R & bunch of "③" (random) codes

.08 $\ni \approx \{|R| +$ [redacted] $\sum_i$ |code for A_i|$\}$ is min.

.09 ① To what extent have we solved (mode progression) 240.34–.35? → 244.37

② Just what was t. Q,A; TSQ I was most optimistic about? 240.03–.08; .22–.29

What we have is our T.S.Q $[Q_i, A_i]$, we start w. a few Q_i, A_i pairs $(i=1/n)$ We find a R pgm and a set of codes for t. $A_i (= a_i)$ w. smallish $|R| + \Sigma|a_i|$. Using that R, when we put in Q_{n+1}, we get, using random input on #2 input, a d.F. for [redacted] A_{n+1} ← is that what we want. & we then do trials (Lsrch) until we get t. rite (given) A_i as output. This, of course, assumes that R doesn't change. In fact, as we get more & more $Q_i A_i$ pairs, we will want to Revise R. — This has been all very theoretical thus far. — I want to see what these operations correspond to in a real TSQ.

Say we are doing MTM w. $\vec{Q}_i \equiv$ s_i, N_i [like 240.22–.29] We end up w. a set of solns $F_i \ni F_i(N_i) = A_i$.

Consider that t. $[s_i, F_i]$ correspond to a xform of t. original $[\vec{Q}_i, A_i]$ [redacted] set. ,

↑ Not "Bag" in this case

Given the $[s_i, F_i]$ $i=1/n$ How do we get D.R. of F_{n+1} from s_{n+1}?

.25 One Gross simplifn: we have set of primitive ~~set~~ functs, we then want to find a set of functs (w. ~~wts~~ Wts.), so that w. t. primitives, we can use this ~~set~~ set of functs to generate $[F_i]$ w. max pc. [we haven't yet included $[s_i]$ info] ← Includes OSL

.28 These new ~~set~~ functs will be subgraphs of the $[F_i]$ $i=1/n$. functs.

~~After~~ Next, we look at t. inputs $[s_i]$ — try to ~~find~~ generate them from a set of primitives, & find larger abss ~~that~~ so that t. course set $[s_i]$ has ~~small~~ small total dcrn.] Similar to .25–.28 on $[F_i]$

Next, we try to find "parts of the $[s_i]$" that correlate w. those corresponding F_i or parts

.32 of t. corresponding F_i. ~~We do that~~

.33 We do all of .25–.32 so as to minz. (.08)

So .25–.33 is a rather gross dcrn of t. kind of activities one wants to do to minz .08. It is a very "elemental" way to do it: to break down parts into sub-parts & correlate them.

There are certainly more "global" ways to Minz .08!

.25–.33 are some simple induction methods. To ~~find t.~~ get t. mechanics of them & to find more methods, Do run thru a TSQ & see just what

.39 techniques I use.

9.23.00
Bulg.

.01: [In 243.25-.33: Make sure I understand just how each operation ↑ pc
(≡ ↓ 243.08).

N.B. The ANL TSG I did in most detail in *Saarb* was somewhat non-ed, in that it simply tried for a short overall psm. in 243.08: Perhaps look at that psm in some detail — find out just *how* it did this.

.06 I think t. way it worked was: We started out w. ~~[illegible]~~ start w. primative funct set. S_i, found a F_1 by Lsrch: ~~[illegible]~~ Adding F_1 to t. funct set, we find a F_2 ∋ it works for both S_1 & S_2. ∴ α Adding F_n to t. set of functs we ~~try~~ use Lsrch to find F_{n+1} ∋ $\{F_{n+1}(S_i) \to A_i \; (i=1|n+1)\}$
.10 Then n ← n+1 & loop to α

.06-.10 suggest an improvement to itself, that *mite* make it better than **243.25-.39**: We do .06-.10 as stated, but the functions we use, to generate trials are much *more general* than those used in .06-.10

A more Global (but still a bit el.) approach to .06-.10: In hunting for
.15 F_{n+1}; we *are* interested in "modifying" F_n, because F_n is already a short code. Similarly $\{F_i\}(i|n)$ are all short codes. Also, any common sub~~graphs~~ functions (subgraphs) in $\{F_i\}(i=1|n)$ would be useful
.18 in ↑ pc of all of t. F_i $(i=1|(n+1))$ [I think I may have discussed .15-.18 not so long ago!]
.19 Anyway .15-.18 is a justifn. for using ~~E~~ $\{F_i\}$ $i=1|n$ & parts of those functs ~~to~~ to construct trials for F_{n+1}, & it tells us (I think) what pc's to assoc w. them: In particular, it enables us to quantify
.22 **OSL** — which is quite impt. in this problem (I think).
.23 It *may* be possl. to ~~[illegible]~~ augment t. techniques of .15-.22 by those of 243.25-.33: looking for good sub functions (over part)
.25 in both t. $\{Q_i\}$ & $\{F_i\}$ sets *and* correlns between them: Any such regys ~~[illegible]~~ found would ↑ pc of ~~[illegible]~~ F_{n+1} in .15-.22. (The the F_i & Q_i in .15-.22 mean somewhat different things than the
.28 S_i & f_i in 243.25-.32.)

So: .15-.28 is essentially t. method used in ANL in Saarb, w. (I hope) a better understanding of the theory (theory = 243.08 and 241.16 and 241.01-242.05 <242.01-.05 is of particular import>) Also .19-.22 gives theoretical basis for pc's of component functions in ~~[illegible]~~ Lsrch trials for F_{n+1})

Also, because of MCT, I have a greater variety of data types I can put in t. corpus — which make TSQ writing *easier* ~~than~~ than at SAARB!

T. forgo. Stuff is all for QATM. It ~~may~~ *may be* that t. general way I do t. TSQ in t. forgo. .15-.28 can also be used for OZ problems.

.37 Also Note (Book Mark) ~~[illegible]~~ sub-goals of 240.34-.35: How general is t. lrng. techniques of .15-.28? Would it be able to work reasonable tsq's. and deal w. t. *scaling problems*? T. "correlations" of *.25* may help a lot.

Bols

65
16
81

3 PM Sunday / Sol.

The Analysis methods of 244.^{23}–.28 may not all apply so nicely to the QA problems! While [Qi] sets ï [Si] sets can be identified ï "understood" (minimally coded) the things corresponding to the [Fi] sets of 243.25–.33 are not as clear!

.05 Hvr: 244.15–.24 should be regarded as a guide in lrng TSQ's: That I will write f. TSQ: decide on what things TM needs to learn, what kinds of things it needs to know about, ï see if I can fit it into f. schema of 244.15–.24! This scheme may need

.09 "Augmentation"! → .15

One main Q was: "How far could I get" w. a given scheme for lrng? Presumably, at some pt. of complexity, a scheme would become ~~"Extensibly~~ "Universally Extensible" so w. suitable TSQ, it would do the Equivalent of learn any needed Modifn. of its scheme

.15 (.09) → SN Another ("Very English") approach to TM: Write TSQ: Write down all the heuristics, tricks, cones, that I seem to need to "lrn" that TSQ; Then try to devise a formal model, using ideas in [redacted] MCT. 244.15–28 would be a good first approach to such a model. (≡ .05–.09)

In .15 ff: [redacted] After I write f. TSQ ï f. "solns" in English, Make up a compact lang. that expresses Easily f. heurs, concepts, etc, used in the solns.

60

SN A common "Meta heuristic": when a problem is solved, or when something unexpected occurs: Try to see how this new (conc / info) fits into f. past — to ↑ its pc. (Or in a more advanced TM — into f. expected future.) — So, what kinds of probs would it be useful for? This puts "hooks" ï "condl. pc's" on f. new conc., so we know (under what circumstances) ~~when~~ to try it in f. future.

OK. So go back to those TSQ's! 181.5, 182 have some stuff, but I think that were extensive listings of things to put into TSQs.

180.24; 167.24–.34 ; 173.10–.16
180.34

173.06–.08 Gives a big set of TSQ refs
— Also possl. future Extensions

If I get stuck! review ~160 ff (or 167ff), lots of Good ideas;

→ Q: How did I get "Stuck" in 160ff? — What were diffys?

Perhaps go thru history since '60 ; look at each of diffys that occurd ï how each was solvd.

As I see it: The way it's Done: I start w. a stochastic language that is universal 2 ways

.37 ① it can express anything (implied by cond ②) ② Its expressions are at most, a constant "longer" (or pc < by > a constant factor) than any other stoch lang.) [cond ② implys cond ①]

< Woops! in .37–.38 stochastic lang → stochastic Operator >

246.01

Bulg.

.01 ~~[illegible]~~ 245.40: **Next**: ~~[illegible]~~ using the stochastic lang, we input Q_0 — we put in 2ux "random" codes (Lsrch) until A_0 comes out.

— We then make a "summarizing machine" & do Q_1, A_1 the same way.

245.37 ff is a bit "too general". More Narrowly,

— I'm thinking of a stochastic function-language. (Prob Prim-rec funct lang — not quite Turing Universal — can't express Ackerman funct)

.07 Anyway I use this lang to get A_0 from input Q_0, using Lsrch.

Next, I modify the lang so that $Q_0 \rightarrow A_0$ is more likely, but I keep the lang ~~[illegible]~~ "Universal" (prim. recursive universal).

Then I repeat w. $Q_1 \rightarrow A_1$.

ABC

.10 In .07, if we get all codes for $Q_0 \rightarrow A_0$, we have a "complete summary machine" & we should be able to "reinsert our "random input" using Q_1 as input, & eventually get A_1 as output.

SN N.B. our "universal" machine probly has to be able to make "Definitions". (at least !).

.15 In .10, After "input #2" creates A_0, Input #2 is concat on to input #1 (Input #1 "defines the current Operator").

Woops! .15 wouldn't work (I think)! After creating A_0, with the concatd code on input#1, I don't think the system would be more likely to give A_0 as output (w. input Q_0).

.20 ⟶ I have to work out the "Summarizing Machines" for stoch Operators! → .25

Perhaps in .15, ~~After~~ Before creating A_0, the machine put its "state" on the stack. After creating A_0, it goes back to stack state but for new input, it remembers how A_0 was created, & is ~~[illegible]~~ more likely to try it w. whatever new (Q_2) input is.

.25: .26 look at "2 kind ~~of~~ ~~[illegible]~~ Prob. Induction" for summarizing machine for Bags. I did have a "summarizing Mach" for Bags but it used a less convenient Model:

A better summarizing Machine for Bags:

.28 **We list** all stochastic ~~[illegible]~~ operators $S_i(x \rightarrow y)$ in aprip order (i is order, $P(i)$ is assoc. pc.

The summarizing machine ~~[illegible]~~ is equivt to: Chose a S_i with probly $\propto$ $P(i) \cdot \prod_{j=1}^{n} S_i(Q_j, A_j)$ & use it to evaluate $S_i(Q_{n+1}, A_{n+1})$ (All possl)

.33 { ~~In practice,~~ The measure "store the S_i's w. the hyest $P(i) \cdot \prod_j S_i(Q_j, A_j)$" The fewer we store, the more "Backtracking"

.34 we have to do.

The next problem is how to do .28 [redacted] → .34 in a practical way

— Heuristic Methods to ↓ cc.

eq .33 induces a pd $P'(i)$ on all stochastic operators, $S_i(Q_*, A_*)$

When a new Q_{n+1} (or Q_{n+1}, A_{n+1}) comes in, we'd like to try them in $\frac{P'(i)}{cc(i)}$ order, but it's not so easy to do, if we just have, say 1000

9.26.00
Bulg.

.01|246.40 S_i's w. hy $P'(i)$ [If we had all S_i's in $P'(i)$ order, then this would be fine & would get a v.g. pd for $Q_{n+1} \to A_{n+1}$. [redacted] What I want is a way to approximate $P'(i)$ in a way that gives a useful, searchable distribn on i, for [redacted] Lsrch.

.05 One way might be to look at $\vec{S}_n$ ← 10 dim. vector, which is t. 10 best S_i's that I've found, for $[Q_j, A_j]$ $j=1/n$. (So: 10·n S_i's.) & "pass a grammar thru them."

.08 Or possibly "pass Grammar thru" t. 1000 best S_i's for t. current value of n.

SN in Sol89 (Israel): §5 "How Updating is done" : We use Lsrch to solve several probs. We take t. set of solns & pass a grammar thru them. This then gives us a p.d. to use Lsrch to solve t. next problem.

When described this way, t. Lsrch is not "conditional", so its P.D. is indep of t. nature of t. problem. In .05—.08 we do a similar "passing a grammar thru", but we have a conditional probability distn., $S_i(Q,A)$. [Note: $S_i(Q,A)$ is usually not a joint P.d. on Q & A; It's t. pd. on A, as a function of Q.].

So t. P.D. on S_i is supposed to be "Problem-Indipendant."

.19 One Q is: Just how do I express a condl p.d.? Can I express it like those "function/concept "nets"? Well, a p.d. is a type of function, but it is a very special kind of function — So we'd want to restrict t. representation method so it would generate (almost) only P.d.-type functions (output on [0,1])

One sure method is to simply use an approxn of ALP: to get $P(Q,A)$, we have many functions that map from Q to A, & we put in Q & then t. prob'y of A is ∝ t. no. of functs having that A as output. [redacted] We may want to weight these ⟶ functs, for greater versatility.

It would be neat, hvr, if we could just put in Q & A & get a P.d. or, put in Q & get a list of t. most likely A's w. their associated pc's.

We can get a P.D. on ordinary functs, by considering t. net that represents their generation. Each choice in t. net (each edge) will have a certain pc.

Perhaps a better way: Consider t. set of (non-probilistic) functions $[F_i]$

.33 that generate $Q_j \to A_j$ $(j=1/n)$. Say P_i is t. aprip of F_i, the i-th such function. Then we pass a grammar thru t. P_i-wtd. set of functions F_i. T. functs are represented by nets, and we can look for common sub-nets in t. set of functs. There is a finite set of such functs, so we may be able to normz. t. $P_i \leftarrow \frac{P_i}{\sum P_i}$.

Hvr, t. meaning of "Passing a gramm. thru this set of functs" is not clear, because SSZ is not clear, and SSZ has to be clear better : 248.01

9.26.00
Bulg.

.01: 247.40 We can have a "corpus" whose code we want to min. This enables us to tell if a code element (a "regularity") is legit. in derbg. the "corpus"

.03 In the present case, we know the true $\overset{\text{stoch}}{\text{grammar}}$ for the set F_i. It is the apriori on $[F_i]$ with zero wt. for all funcs that ~~[illegible]~~ aren't 100% compatible w $[Q_j, A_j]$ $j=1|n$. We want to approximate the rest of the D.F. from this

.06 ~~[illegible]~~ with sampler, $[F_i]$. Is this the "surface reconstruction" problem?

Probly not: I think I've been here [.03–.06] befor: I may have not solved it. It may have been in SGA, in which I wanted to approximate $G(\vec{x})$, given $[G(\vec{x}_i)]$ $i=1|m$ — an interpoln. function. [i.e. a set of pairs $[\vec{x}_i, G(x_i)]$]
I normalized G, so that $G(\vec{x}_i) \leftarrow \frac{G(x_i)}{\sum_j G(\vec{x}_j)}$
Then I wanted to pass a grammar thru the $G(\vec{x}_i)$'s — which were like a P.d.
Lots of trouble!

There is an A.H. soln, that assigns p_i to F_i for $i=1|n$ & zero to all other funcs.

.14 Perhaps try for a short dern of the set of pairs (p_i, F_i) $i=1|n$.
or (F_i, p_i) in the sense that we want an operator in which we input F_i & get p_i (or close to p_i) for $i=1|n$. We want the operator to have

.17 a min. dcm.

.18 SN [redacted] Did I ever find a way to deal w. ~~[illegible]~~ the pc of "correcting" p_i when $F_i \rightarrow p_i$ wasn't exactly rite? Well, one way is to have $F_i \rightarrow p_i$ be a pd rather than a deterministic function. We could have a deterministic funct, then a Gaussn D.F. w. common σ^2 for

.22 $i=1|n$. This seems Not BAD!

So (.14–.17) & (.18–.22) may solve the problem!

It looks like a non-linear regressn. problem ("How many $\overset{\text{params.}}{\text{coefs}}$ to use?")
Well, this "n.l. regressn problem" may not be so easy! In fact I did try to model G_i as a function of $\vec{x}_i$ in my recent work on (GA/GP). — But I didn't think of it as a n.l. regressn problem!

~~[illegible]~~ Still, it looks like a rather hard problem! n.l. regressn or not!

One trouble! I do know the grammar that obtains the p_i from the F_i. (in 247.33)
p_i is the apr of the F_i & is usually easy to find — in that F_i are generated by some (known) stochastic grammar & p_i is the pc assigned to F_i by that (known) grammar.

.33 So the question is: is there a cheaper way to derb. the set $[F_i, p_i]$?
Well, we may add more info — say on $\overset{\text{negative}}{\text{negative cases.}}$ — in which case ~~[illegible]~~ the old grammar that generated $[F_i]$ would give ~~[illegible]~~ wrong pc's to the Funcs that should be given near zero pc. I will probably have a lot of F_i's that I know (empirically) do not fit. ← GOOD!

Hvr, see 252.29 for discussn!

~~The poor~~ Derbg the positive cases only is easy; its pc = 1; its just the apriori grammar (pc = 1) generating the F_i's from the F_i — total pc = 1.

9:26.00 Bdg.

22 × 16¾ → 27.65" diag ←

> 22 × 16½ → 27½" pretty good ← 3,4,5

If we list the [F_i] that satisfy the constraints, this will be a complete dern. of a set of F_i's that excludes all other F_i's. Its total pc

.03 is $\boxed{\prod_{i=1}^{n} p_i}$ This is the A.H. dern of the set of positive & all known negative cases.

So .03 is pc of the default (analogous to "random") case. Any shorter dern is of (some) interest. — derns of smaller pc are usually not of interest unless we have a way to add up a lot of them. $\boxed{\text{ABCDE}}$

But perhaps all we really want is a way to get functions that are "close" to F_n. — or to [F_i] $i=1/n$. Actually F_n is the only funct. that satisfys all of the constraints — only its thusfar of [F_i] will fit [Q_i, A_i] $i=1/n$. We use the rest of the F_i to "expand the sample" in a "soft", approximate way.

No! { On second thot, p_n is the pc of F_n, & F_n is enuf to do a complete Dern of [Q_i, A_i] $i=1/n$: positive & negative cases!.

I want to derb. [F_i, p_i] $i=1/n$ plus [F_j, φ] for f_j that I've tested that did not work.

I guess F_i ($i<n$) are all impt also, because they are all attempts to find F_∞ (the F that works w. all Q_i in an optimum way.) ... But I've forgotten about the problistic nature of F mapping Q→A.

But still F_n is "Uniformly Better" than F_{n-1}.

Should I try to find a grammar that just gives F_n hy pc, & very low pc for all other F_i that I have tested that fail for one (or more) Q, A ?

I could often just have one positive instance of F_n ∴ small ssz (possibly more if I "Oversearch")

Re: use of data of F_i ; ($i<n$)! These can be considered to be "approxs of F_n" ! Each has a pc of "correction" to bring it to F_n accuracy. Usually, I imagine, this pc of correction is quite small so these $F_{i<n}$ mite not be worth considering (?). — Maybe not so small!

In the first place, $F_{i<n}$ have a prps >> that of F_n. "Correcting" F_{n-1} could involve just "recognizing" Q_n & then giving A_n or a suitable function for $Q_n \to A_n$ (n.) — Total pc could be not

.30 much worse than that of F_n. → [See 249.33–.40 ; 252.01–.28 for ending and a kind of resolution of this problem]

[redacted]

CRITICAL

.33 247.19–249.30 is an attempt to solve a very impt problem: How to develop a good stochastic function / operator that maps $Q_i \to A_i$ [Q_i, A_i] $i=1/n$.

.35 There is a standard ALP soln. to this problem: Say [F_j] is a set of functs of apriori [p_i] ∋

.36 ~~∀j~~ ∀ i,j $F_j(Q_i) = A_i$. Then $\sum_j p_j F_j(Q_{n+1})$ is a wtd.

.37 d.f. over possl. "values" of A_{n+1}. — i.e. a d.f. for A_{n+1}.

So just what is 247.33 ff. trying to do?? Well consider .33–.37: If we only have a few F_j's that do [Q_i, A_i] $i=1/n$, then we will only have (at most) a few / 252.01

Meta Problem of TM (.34)

.01: 249.40: I want to do a good review of ~160 to 249: T. present idea is to
.02 write a TSQ. & write out detaild heurs for humans: Then develope method
.03 of lrng. along with this: 160-249 discusses various problems: solves some:
I should be familiar w. These (probs/solns) while I'm doing .02 ÷ .03.

It should be possl. to do .02 w.o. (.03) ← i.e. developing lrng. method. I would list all needed heurs, then look into Q of how each heur could have been dcvrd. — which heurs (if any!) were needed for each heur discovery, ect... down to
.08 a Basic set of heurs or a primotive set of functs/functionals.

Some impt. probs/ideas in TSQ design:

1) The idea of local v.s. Global (symbols/nos/vars): So if I ask TM.;
If x+3=7 what is x? It must realize that x & x+3 etc are to be used in this problem only, but that "Most" other concs (unless specifically told to be "local") are Global & apply to All problems

2) Is [Q, A] format an adequate / good format for an initial tsq? — for all TSQ's? Conceivable all problems could be "usefully" put into this form. — e.g. If its a ① TS, ② bag lrng, ③ inv, ④ OZ, problem, we could have Q w. an additional component that tells which of the 4 it is so, ①(TS, begin of TS): "A" would be next element in Time Series.
② (Bag lrng:, finite bag : Part of Qual): A is next symbol in Qual or rest of Qual.
③ (INV, Function, [redacted] F, F(x) = y; y is Given): A is to find x.   60
④ (OZ, G(·); cc : A to find x ∋ G(x) is Max in time ≤ cc.): A   12

[Note, hvr, Most INV probs are solvd by converting them to OZ probs   4:45

TM could learn a lot by watching an O.T. at work: The O.T. would find a sequence of $\vec{x}$'s in attempts to Maxz $M(\vec{x})$: TM would try to find a p.d. that characterizes the $\vec{x}$'s distribution. TM would not have to know $M(\vec{x})$ in order to do this, but TM could regard this as a kind of Condl p.d., w. M(x) being part of the "Condition" (≈ Question).   Miye at 3:30 4: 4:20 2 5

By watching many O.T. for many kinds of OZ problems, TM might get a good idea as to how to estimate $G(\vec{x})$ very fast & thereby do OZ probs very well,   1 30 1 .50

Note that part of the OZ problem is t. cc of a trial: this would have to be appreciated by TM!   20 oot

.34 Work on "Meta Problem" of How I should work on Designing TM! Taking into account my ↓ power of memory. Sounds like t. "unsolvable" OZ problem, where $G(\vec{x}) \cdot F(T)$ (mach soln) is to be maximized. F(t) is my failing Memory!

1) Some ramblings: What was bottleneck after Sol 89? — Why did I
.38 not finish TM @ Saarb? Reasons: ① T. TSQ's very hard to write ② They solns → (251.20)
3) I have made several impt. discoverys since Sarb: (251.01

ⓑ

Very impt. problem ("Bottleneck"?) ↓ How much progress on problem of interaction of tasks? in "AND" "OR" nets.

Meta TM prob:

"WON"

.01: 250.40: 3) (cont.) They are: ⓐ Sort of soln of (what to work on next) for complete AND/OR net.

.02 ⓑ M&T proofs understanding ⓒ Understanding of TM2's Gor.

→ The main Subgoal is Understanding English text.

→ A BIG Q is: How much info do I have to introduce (≈ A.H) into TM (by TSQ's usually) before it can learn much from less carefully designd Text? How will this threshold depend on IPC of Machine Usd? How well can we do w. 10^{10} ops/sec, 10^{11}, 10^{12} ... 10^{15} ops/sec? Or, more exactly, how well can we do w 10^{n} ops? (≡ cjs)

For a given tsq. I will have to put in fewer steps (less "Skinnerian") if I allow large CJS. [redacted] Say $> 10^{10}$ [redacted] or 10^{11} steps

If I have smaller cjs, I'll need more examples. So the Q is: What is

$\overset{N}{\text{(no. examples)}} \overset{\times}{\times} \overset{C}{\text{(mean CJS of the examples)}}$ = total ops needed for TSQ:

N × C: Can we minimize N × C (for a TSQ that goes up to "Understanding text".) by varying C (then N is a funct of C: N(c) — so minz c · N(c) by varying c.

c · N(c) could be small for small c, but we would then have a TM that had no experience (i.e. have no Heurs) for problems w. large CJS.

Large CJS's use certain heurs that are not practical for small cjs: — e.g. large initial overhead.

So, In General, we may want to use as large a CJS as we can afford; its N(c)·c. (that is the total cc of the [redacted] TSQ.)

≡ Total cc of Corpus

.20: 250.36: ⓐ Only usd concs that were complete solns of previous problems. ⓑ There was no use of subnets (subfunctions) of previous solns. I recently (since ~160) wrote about this — why it occurd. I think it is intimately involved w. how "Big", how complex the TSQ is, and the exact nature of the induction system used: what the form of the inducting? indubing lang is, etc. In the TSQ's ideas of .... ? what did I have in Mind?

ⓒ The soln method didn't "scale" well — as TSQ continued, too many functions in "Storage": (see 253.12 on this "scaling", see 253.02 for defn of SAARB

Scaling

.27

SN In "inserting Heurs" we do this so that TM could have lrnd the heur by observing enuf ~ cases in the past. What SSZ should be used? It may depend on just what the heur is: but try OSL: just one case in the past.

9.30.00

Bulg

Olympia's fund US

STRI or S2RI (.35) (.38)

String to Real ~~Prediction~~ Induction

.01: 249.40 : possl. values of Anti : If we are looking for a particular known Anti, then we may well be unable to find it in the small set of possl. Anti's!. — so we would really completely "lose" on this search.

247.33 ff tries to deal w. this by extrapolating t. set of functions, [F_j] of 249.35-.36, by "passing a stoch Grammar" thru [F_j]. T. spirit of this is that of Z141 — (passing a stoch gramm thru a set of strings).

A major dfty is that of SSZ. we only have a few "strings" (w. wts, P_j) to pass our grammar thru, & I don't have a clear idea as to how to do this . . . It may, indeed, ~~be~~ be a not viable way to solve this problem.

.10 I had t. idea of extrapolating t. set of pairs, [F_j, P_j], w. ($P_j = \phi$ for F_j trials

.11 that did not pass thru [Q_i, A_i] i=1/n.) 247.33 — 249.30 tries to deal w. this problem. Note that because of t. large no. of negative cases, the SSZ for t. problem of .10–.11 is quite large.

A somewhat dif'rnt approach to 247.19 "How to express a cond'l pd:" Stochastic Grammars (like in Z141) are p.d.'s; they are unconditional pd's: If t. Grammar were a function of Q_i, and A_i were given a P.D. by that Grammar, we'd have what we wanted! Hvr, it's not clear as ~~to~~ how to do this — the deterministic (or even probilistic) mapping from Q_i to (a) grammar(s).

In General, it may well be that I will get a better "feel" for what kinds of cond'l. P.D.'s I want, when I try to express my "English" total (including hours) solutions to TSQ's in t. form of cond'l. P.D.'s. At t. present time, my ideas on this would have to be too general.

So this is a good motive to go to t. point [250.01—.08] ← ≈ [163.16 / 163.01]!

.26 in which I write a TSQ in English & write solns & all hours in English, then try to devise a language & induction rules to acquire t. TSQ

.28 viz t. solns & hours in (.26)

"early work" (G)

.29 N.B. t. idea of 248.33 of including negative cases, may not help much! A v.g. fit could still be obtained by deriving t. ~~[illegible]~~ [F_j, C_j] set exactly & giving $C_j = \phi$ to all other cases: (particularly if there are only a few $C_j > \phi$). Hvr, still it is possl. that we may get a grammar that gives approximate values of [C_j] ~~the~~ that has a whole greater PC.

.35 Anyway: Consider t. problem of "curve fitting" [F_j, C_j] to be "not well solved"

2 impt. cases thus ① F_j is set of funct, C_j are known reals & zeros for neg cases.

② F_j are functions or strings denoting objects in [GA / GP] & C_j are their known scores. → Much work on this

.38 DEF Call it t. String to Real ~~Prediction~~ Induction "STRI" STRI.

N.B. Stochastic Langs is one kind of STRI ~~problem that I have "sort of" solved.~~ functional form

9.30.00 Bulg.

Parkinson's Law: "Work expands to fill time"
To get work done fast, allocate less time to it!
e.g. TM. → 255.01

.01: 252.40 That a [redacted] STRI problem [redacted] soln. can have. Hvr t. hvrs of Grammar discovery have not proved to be very useful in solving STRI!

.02 SN I wanted to use y STRI soln. for [redacted] SGA! My "soln" of Koza's "3+2³" 11 input Muxer problem was simply Lsrch over a simple function space; then add to t. set of functions, any new funcs found that ~~seem~~ ~~clear~~ were much closer to t. Goal. — This seems close to what I did in t. SAARB TSQ's. This suggests a "Back Door" soln. to t. STRI problem!

— So if I have F_1 & F_2 & F_3 w. hy C_i values, I just add them to t. "function pool" w. wts, perhaps ∝ their C_i values. The functional combinations can be at most of depth 2 or 3, because of rapid combinatorial explosion (w.o. heuristics). We also have rapid ↑nce as no. of functions to be used in t. combinations ↑.

.12: One trouble w. this [redacted]: By itself, it does not scale to larger probs — t. no. of functs available grows (≈"linearly") w. no. of probs solved — so it Eventually becomes impractical — unless one devises heurs to "narrow down "which functs to try".

One aspect of t. "Scaling" problem

[redacted] STRI

.16 Some Common Mappings from strings to reals: ① strings represent ⓐ equations or ⓑ Systems that have evaluable parameters. Example of ⓑ would be a string representing an electrical ckt: [redacted] i an ω; the real (or complex no.) is t. impedance at that frequency. Or [redacted] the string describes a network & its input; t. real (or complex) output is t. ~~that~~ output. The representation of SPICE ckts by Koza is a good case in point. — The "output" Is t. [redacted] "fitness function" of that ckt.

A common way to discover t. string → real relation is to experiment w. t. ~~to~~ numerical params in t. "string".

A string can represent a physical system (say in QM) — t. "real" can be some ~~pp~~ computed property of t. system (Energy, velocity ...)
Often, t. "text" will set up t. eqns & t. "numbers" will furnish "parameters";
Occasionally Numbers (integers, say) can help set up eqns., but this is usually either a specialized kind of continuous param or t. nos. are [redacted] descriptors that are assigned "symbolic" meanings in a rather A.H. way.
— So they have to be lrnd via larger SSZ. •

This is true in ~~Koza's~~ Koza's systems: "Reals" & "Symbols" are dealt w. in different ways in GP.

In assigning pc to strings (stock lang.) t. strings usually have no nos. numbers. in them. This is ALP's way of Bringing "Quantity" from "Quality". It is a very special kind of functional relation. What it does is related to "counting" t. no. of times a certain feature occurs. We can get a pc-like number by suitable normzn. — or by multiplying such normzd nos. together.

Considering this ↑ way, it seems clear, that usually, STRI

0.3.00

Bulg.

depends very much on info that TM has about what the "symbolic" part "means", I had decided that I couldn't really work on this STRI problem until I had a less "abstract" idea of what to expect in ~~t~~ relevant functional relationships.

Well; now I have a somewhat better idea of what to expect!

ANL is one simple example of STRI. (253.16)

Another type would be recognizing certain "substrings" or "structures", counting them & having alg. functs. of t. [redacted] various "counts". ~~This is what is done in "Grammatical Induction"~~ Passing a Grammar thru a set of strings is one way to get a number (≡ pc) for each string.

Also: "Spice" is an extreme case: Looks at symbolic (ckt. dcrn) & nos (param values) & calculates its path in ~~[illegible]~~ time: ~~[illegible]~~ (≡ seq. of numbers) or ~~[illegible]~~ any other param of resulting system.

Or to invert "Spice" Given params to design (symbolic) ckt & params (nos) (This is what Kozz's filter design pgm did).

The dcom of functions in terms of prim functs involve mainly symbolic (not Graphs) ~~[illegible]~~ dcom + prim funct + 1, 2, π, e, γ ~~[illegible]~~ & some small integers as "params". e.g. $\sin x = (e^{ix} - e^{-ix})$ ~~/2~~ (divide by 2 because there are 2 functs in numerator ... (from "Counting" operation).)

0.400 Bulg

1:11 22
39 / 150 4 18

80 x 50 = 4k/yr.

.01 Admin: Parkinson's law: "Work Expands to fill Time":
Early in TM rsch, I felt that TM could easily take rest of my life: It looks like I've been arranging it to do so!

A (possl) way out: Make outline of what needs to be done on TM: (A sort of ~~[illegible]~~ PERT (?) diagram), w. indications of interaction, a AND/OR net of Tasks:

Describe each task clearly & tell how much of it I've done, & how much more time I [redacted] ~~expect~~ expect I need to "finish" it. (diffnt "'s of "finish" may be possl.)

For Most tasks, ~~[illegible]~~ Give them "Names" along w. clear definitions of them.

→ Also Arrange Journals so I ~~know~~ know where stuff is; Also so I can refer to a page ~~&~~ or set of pp.

Stuff since Saarb; since S89; since S86 are of most import.
Work on S 89 was prob'ly very impt: (Also work on S86).

- ~~[illegible]~~ Main Subgoal is "Good enuf" Understanding of English, to begin to read & understand English text, so that % of understanding ↑ w. time.

.17 • Subgoal before that: Understanding of Algebra or some part of Math. Enuf to work probs of some difty.

- Sort of Subgoal: To be able to do NMTM ~~[illegible]~~ problems — "Soft" induction in the area of Math & eventually in other domains.

.21 • Devise TSQ for Algebra of .17

.22 • Devise Lrng. strategy for .17. & for general [Qi, Ai] problem ~~[illegible]~~ is a genzn. of (.25), i.e. STRI problem

- PSG discovery: Probably very useful for various kinds of Grammar models... Grammatical induction.

.25 • STRI problem: Given $\begin{bmatrix} \overset{\text{string}}{\vec{X}_i} & \overset{\text{real}}{r_i} \end{bmatrix}$ to induce $P(r \mid \vec{x})$ for new data.
String, Real, induction:
This is important in learning OT's & understanding "fixing" GA. | or to induce $P(r, \vec{x})$.
This is a special case of .22R [Qi, Ai] induction

Superficially, it would seem that QA lrng would be a rather general soln. to .22 & .25 — both impt problem areas.

.31 I think I felt that I wasn't doing so well in thinking about .22 "in the abstract" —
.32 that a better way to deal w. it was to write a TSQ for Alg I along w. "solns" in English, w. all hints I could think of. — then express this "soln." as a lrng. strategy i.e. (.22)
While the ~~[illegible]~~ lrng method would be somewhat tailored to this TSQ, I hoped I could generalize the technique in a useful way. ~~The~~

Then I would continue the TSQ & ~~[illegible]~~ solve the probs "completely" in English & see if the ~~techniques~~ previous techniques for lrng. was adequate — or whether new ~~[illegible]~~ lrng. techniques had to be added.

0·6·00 Bulg.

m/μ

m/m | m   μ/ = m | ±⊕ ABCD

.01 : 255.40 : So there are 2 ways:

1) Work on ~~-~~ 255.22 ~ [Q_i, A_i] lrng problem directly, abstractly, in rather general case. [Q, A can be ~~all~~ or partly symbolic ; numeric

2) Do 255.31 –.40 — which seems very promising.

In 255.22 I had t. updating problem: i.e. say I had a good induction model or d.f.

.07 for [Q_i, A_i] i=1/n ; How do I update the model when [redacted]

.08 Q_{n+1}, A_{n+1} comes in? . I did do work on this & t. soln. was a "Complete Summary machine" ~~&~~ then ~~there~~ was no diffy: If t. summary was incomplete then

.10 Backtracking is warranted/ ~~necy~~ neccy.

.11 Another tack would be: If f(Q, A) works up to Q_n, A_n,

.12 what is t. a priori for t. functional form, f — so we can search on it for a fit? → see 267.29–.35 for possl. relevant remarks! maybe soln!

Can [redacted]([redacted].07 –.10) & (.11 –.12) be unified? — It would seem that they should be!

Re: .07 –.10 & Summary Machines: T. stuff about "Backtracking" is

.15 correct if "summarize" by only including t. shortest codes of t. corpus. — in this case, "Backtracking" retrieves some longer codes for t. corpus, that are now relevant. BUT there are ways to "summarize" ≢ .15 :

— Just what are they? Consider an approximate code plus a correction code. Is that a case? Also what is done in "Theory Revision" in Physics?

.20 In Physics — (say introduction of Relativistic phenomena) we first decide that we are working in a new area (hy velocities / energies), so we "label" t. area, so we can distinguish it from t. old area ("old corpus"). We then make models for t. "new area" specifically. When we get these models, we then try to get a "grand theory" that unifies t. ~~old & new~~ models for t. old & new corpi —

.25 A "Grand Reformulation" that ↓ ~~overall~~ overall code length (This is ≡ 257.34–.40)

.26 This would amount to finding a good model for (Q_{n+1}, A_{n+1}) (or a sequence

.27 of [Q_i, A_i] i=n+1/n+m) and a way to identify [Q_i] i=n+1/n+m.

.28 — Then attempt to "unify" t. new model into t. old.

.26 –.28 is easily applied to learning "3+7" then "5*9".

.30 There is t. problem of ~~deciding~~ This deciding how ~~to~~ big m should be in .26 –.27. This means — deciding how much of t. corpus should be covered by t. new model. Well, we just fit t. model to small m, then we see how large an m we can take & still have t. new model fit!

T. above assumes t. corpus is very nice about introducing new concepts sequentially: say t. new corpus was 3+7; 5*8; 4+2, 4+7, 3*7 ...
Here "3" & "7" etc are just 3 & 7, they are not (as) random nos. —

.38 this means we cannot simply code 3+7, then 5*8, because t. model is too big to ~~be shown~~ afford any code compression w.r.t. 1 or 2 examples.

(257.01)

ABCdetg.

~ "BAD"

'01: 256.40 Well, this 256.38 may well be an example of a "BAD" ("imperfect", "corrupted") TSQ. {I wrote this report "Perfect TSQs & the cost of corruption" — on this problem} — I had the idea that it could very much ↑ cc of search, & sometimes make it prohibitively large.

My more recent ideas, hvr, were that we shouldn't depend on a TSQ being near "perfect". Maybe for infant TM, we may use perfect TSQ's — but later on, I expect TM to be able to deal w. very imperfectly constructed TSQ's!.

.09 One way would A.H. deal w. 256.38 (& with a somewhat more general dif'ty of this kind): TM finds an operator that works 3+7, then looks for other Q.A.'s in which this operator works, then tries to find a way to identify the QA's in which that operator works.
.12

Actually (.09–.12) is not so A.H. It amounts to a way of "partitioning" the corpus in a useful way!

(N.B.) the problem of ~~the~~ 256.38 & its "soln" (.09–.12) is an example of an unordered "Batch" corpus — Harder for TM to learn, but much less likely to be "imperfect": That it can be a realistic corpus for TM, & very good, in the sense that TM has some idea as to what its "future" problems will be. This enables it to deal w. what would ordinarily be done by TM2 — i.e. to have a feeling for what "indices" are "interesting", in the sense that they ~~are~~ seem likely to be related to "advanced" parts of the "Batch" corpus. — I also vaguely remember this "Batch corpus" somehow enabling TM to deal w. cc [redacted] minimization in a fairly non-a.h. way: — I'm not sure just how this was supposed to work! — I would seem to make the writing of TSQ's a lot easier! → 258.06
.26

Note, hvr, that my initial goal is to get info into TM! — That the "sequentiality" of the corpus can contain impt. info, & I start by giving TM that info "free". Later, I can wean TM from this by giving TM augmentations of the corpus in larger & larger "chunks", until I am adding chunks that contain info that TM will not be able to deal w. at the time. So he will learn to recognize these corpus areas & [redacted] realize that he is not yet able to deal w. them. → "Textbook" → 258.06
.33

.34 So this ≡(256.26–28) is ≡(256.20–.25) a standard, simple, method of theory revision used in physics: Old model (M) doesn't work on Q_{n+1}, A_n. Find new (usually narrow) model (M') that will work on Q_{n+1}, A_{n+1}; find way to recognize Q_{n+1} as different from
.28 Q_i $(i<n+1)$: (Unify / Integrate) M, M' → $\tilde{M}$, which is a shortening of code for M, M'. → (note some aux. problems! 258.15–.17). → 258.05

.39 ↳ Note one non-a.h. way to do (.34–.40) → 271.34–.40

Th. most

.01: 257.40 While 257.34 is "not bad" it is not optimum. It is el. T. most non-ol. way would, when Q_{n+1}, A_{n+1} come in, do a direct srch for a good model for $[Q_i, A_i]$ $i=1/n+1$ directly: For this to be even mildly interesting, we must have a reasonable apriori on t. search space of A = f(Q) functions + (≡256.11)

.06 252.33
257.26 A Commonly Occurring "Batch corpus" of this kind, would be t. problem of getting TM's to (understand / compress) a large English text. A v.g. way for TM to work on this would be to first find parts it could compress easily, then find more diflt parts, etc. I expected that TM would lrn to "read ȧ understand" English text in this general way, but I hadn't thot of it as a "Batch Corpus" w. TM going back later to try to code or recode a section that it didn't "understand" t. first (or 2nd) time around. I really want to get TM to pt. at which it can do "Batch corpi" of this kind!

It makes writing TSQ's very easy!

One recurrent problem: TM has to decide when to temporarily quit work on a part of t. corpus ȧ work on some other part — (i.e. How Good a Code is "Good Enuf"?) (hoping to eventually return to that part

.15 later (ȧ wiser)). This Judgement also occurs in e.g. 257.34–.40: We find M' for (Q_{n+1}, A_{n+1}). How good a model must M' be before we go on to

.17 t. "Q_{n+1} recognition problem" ȧ t. "M, M' unification problem"? we mite do one M' then if t. recogn. ȧ t. unific part didn't work well, go back ȧ try for a better M'. (or even a better M! Super back track!)

Could I get TM to learn Algebra in "Batch Mode"? I would start out (perhaps) w. a "sequencial" "linear" TSQ ȧ switch to "Batch Mode" instructions as soon as feasible. The "sequencial" TSQ has more info in it — the explicit ordering really makes it a lot easier for TM.

— Best way, start out in sequencial Mode, then see if I can get TM to work in "Batch Mode".

[One of t. Good pts about "Batch Mode" is that it mite be easier to define TM's long Term Goal in terms of this batch. If t. Batch is of infinite size! we want TM to understand, work as many probs as possl. in a given time. This means we have to define TM's "Horizon"!

.31 Also the trade-off betw. Good compression for a small part of t. corpus,

.32 v.s. weaker compression for a larger part of t. corpus. I think t. Horizon ȧ t. tradeoff of .31–.32 Must Be User Selected.

One possl. Good aspect of "Batch mode" TM could be like a Grad. Student ȧ Ask Questions of User occasionally.

Another Aspect of TM's TSQ's that I've sort of neglected is t. "ADVICE CHANNEL".

.38 Another idea I've neglected: That one of t. big reasons of my wanting TM to work on probs of large CJS, is that only in such problems is (259.01)

.01:258.40 t. "overhead" of developement of various search enhancement techniques feasible. How much Time to spend on such developements is a TM_2 problem — Having to do w. "Horizon" as well as expected cc of soln.

Hvr., in early TM, I will be giving TM problems that will ~~bear~~ ~~help find, or reduce CJS~~ or be ~~search enhancement techniques.~~ be ≈ search enhancement methods, or will be close to ~~these~~ such methods — in t. sense of reducing CJS of study such methods.

.08 Still, t. "English TSQ" approach of 255.32–.40 is very attractive, & I'll prob'ly do it. — Hvr., I want to have as much Theoretical Understanding as possl. Befor I do it, to guide my ~~[illegible]~~ creation of what I hope to be a very general lrng

.11 algm.

.12 SN My functional lang. didn't use Recursion, because I felt it could be faster in execution w.o. it ~~[illegible]~~. Hvr., it seems clear that ~~recursion can~~ t use of recursion can vastly simplify t. expressn. of many algms. — Even t. simplest recursion in which a function calls on itself, can make a "lot of simplifn." Lenat may express all ~~recursions in 2 parts:~~ recursive functs in 2 parts: ① The final simple value of ~~funct~~ t. funct for a certain (usually low) value ② The self-referential definition part.

So I can use recursive defns of functs, then have a compiler that translates them into loops (or some other ~~[illegible]~~ fast, low cc, form).

So ~~a~~ one slot can ~~be~~ Recursive pgm. Another "Slot" could be a low cc version of t. pgm. We "Mutate/modify" t. ~~recursive~~ pgm, because it is a very ~~[illegible]~~ ~~low~~ hy pc expressn of t. function it derbs.

O.K. so now, T. main problem is my state of understanding of t. Theo. Soln of t. lrng. problem: 257.34 is reasonable approach: for t Theoretical optimum: 256.07–.12 (Summary Machines) 256.11–.12 (problem of apripd for L srch)

.27 for "Summarizing Machines" I want a more General model other than t. one that ~~directly apps~~ simply takes a bunch of t. best codes.

For t. "problem of apripd" of the (Q,A) ~~[illegible]~~ d.f. — Perhaps all I really want/need is an updating algm for when we add a new Qn, Ans. This may be tied up w. t. form of t. Summarizing Machine of .27.

One diff w. Summarizing machine — I had no idea of other models than

.34 a set of short codes: In fact any f.d. is an approximate "Summarizing machine". So stochastic CFG's, linear regress w. ~~[illegible]~~ stated σ^2

.36 are all cases of Summarizing Machines.

T. method of updating, will depend on t. Model (= Summarizing Machine approxn). Most Models don't do OSL. Stoch langs ~~update~~ by ~~modifying prodns defining new NT's & by general Modifn of Grammar.~~ 261.01 defining new NT's & by General Modifn of Grammar (new loops, etc. ~~undefining~~ old NT's...)

0.7.00 Bulg.

HARDWARE

To run Many ~~co~~ computers in || for Lsearch

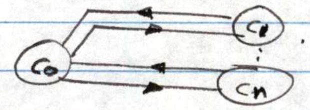
One (control / king (leader)) computer : = C_0 C_0 gives search ranges to each C_i ($i=1/n$). worker computers.

[redacted] C_0 has an input register for ~~co~~ each C_i ($i \neq 0$).

" " " output " " " C_i.

So C_0 has 2-way communication w. each C_i.

~~Ea~~ when C_i gets its range of problems, it starts work on them.. If it solves one, it tells C_0. C_0 (scans its inputs for statements of soln). C_0 then tells all C_i that are working on t. solved problem to stop, and rcv. new (problem ~~ex~~/search range.) C_0 also tells C_i's that have solns. to send them to C_0; C_0 ~~rec~~ receives them & stores them. ~~Whenever C_i~~



Whenever C_i finishes a particular trial, it looks at its input from C_0 to see if this problem has been solved yet.

This ~~s~~ arrangement would many involve only ordinary I/o ports.

— ~~Hu~~ Hvr for ~~a~~ C_0 to give C_i a new problem/search range and " C_0 to receive soln. from C_i, will require a serial port, (or || port)

I really don't know how "ports" work — how to address them, & how t. communication cables are to be set up. T. || port is ordinarily 2 way.

As [redacted] I see it t. C_i machines are all ordinary normal machines but C_0 has n I/o ports : These are actual H.W connectors. — perhaps each has a latch.

Alex felt that I could prob'ly do what I wanted w. a central ~~eth~~ ethernet "server" & a bunch of computers connected to it. — So all computers would be "ordinary".. Is there much overhead time for this ~~Ether~~ ethernet system?

I was thinking C_i need not have a hard disk: Maybe 64k ram & a floppy to start it up.

David G. Willis

Googel

.01 : 259.40 : ~~Another~~ kind of P.d. is that used for [redacted] T.S. regressn. Here we update by modfn. of params, ↑ no. of coefs, use of non-linear coefs:

In both Stoch langs & T.S. regression, I'd like a seq. of updating operations that are "Universal", in t. sense that ALP is universal, & always ~~[illegible]~~ finds any finite describable regy, at ssz & cc ove large amt. ← (like S78T3)

.06 ~~[illegible]~~ In regressn, by employing many linear & n-linear coefs, one can approach

.07 a kind of "Universality", since any funct can be approxd this way.

Similarly, feedforward ANN's can approx any function — in some ways, better than .06 – .07

Tho, I think even ANN is not really "Turing Universal" in t. sense of S78T3

Rissanen's work may be relevant, but I find him very hard to Read!

I do have a paper of his, Somewhere on Generalized Regressn Paper Given at that conf I attended, ISIS conf Abstract on p4 of ISIS conf proceedings.

A purely numeric form of $[Q_i, A_i]$ is : Q_i is a numeric vector & A_i is a scalar or vector. This is t. general/N.L. curve fitting problem.

.15 Again, using many N.L. coefs. or ANN can do t. job. → (.22)

A nice way to approximate ALP is ≈ what Riss does: He lists a large (perhaps infinite but enumerable) set of Fcns, & he uses t. ~~[illegible]~~ / base System in S64₂ (T. one I descbd as "Meanryplacs").

So, in General Induction, we can [redacted] use a wtd Mean of Predn. methods, wts. [redacted] ∝ (apriori method) * (pc of corpus [redacted] wrt. that Method); This should be applicable to all kinds of QA problems; Symbolic Numeric & mixtures.

.22 : (.15) Another interesting possy! Say we have k numeric inputs and a set of functions of various types. We can make all functional combinations of ~~[illegible]~~ order using n functions & k inputs.

.25 N.B. In .22 we also have to include Constant inputs) These are Continuous params in addition to t. normal k inputs This Also Applies to My Earlier analysis of "functional Composition" of 130.25 — 131.18

.28 Anyway, one of t. ideas of ~~[illegible]~~ my functional lang, is that any (prim. Recursive) funct. can be descbd by a finite n composition of primatives. (This is much shortened if definitions (as in Z14 notation) are used ~~[illegible]~~.). T. notation is a preorder corpus string as in Z14) & pc's are automatically assigned, as in Z14. → see 262.12 for recursive defn's

There is, how, t. problem of listing t. functions in ↓ pc order

If we allow t. set of primative functs to be both symbolic & ~~[illegible]~~ numerical, we can get a ~~grand~~ grand P.d. over all ~~[illegible]~~ Q → A functions ~~[illegible]~~ string.

Some such primatives: ~~[illegible]~~ AMS ~~[illegible]~~ M → True else False.

Boolean functs of T, F → T, F. ; T. various Numerical functions + − × ÷, etc.

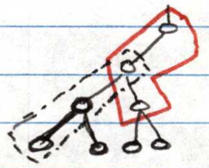
Various Control functions.

Actually, t. set of functs used in "BASIC" would be about adequate. — It does both strings & numbers.

262.01

.01: 261.40 : The developement of 261.28 -.40 is fine, but its pretty much t. same as Alp: except that I use a kind of extension of t. TRMC,
.02 and I automatically sum over many of th. codes, [Thru t. Z141 method,] to get a pdf of various functions.

.05 In coding Function nets, is there much of a parsing problem? —
I think there mite be: I could descrb this net ~~[illegible]~~ using either t. red macro, or the [illegible] macro.



So, if I have a certain set of Macro functions — each w. its assoc. pc, I will have to try reparsing it, every time I make a new def [redacted]

~~[illegible]~~ "Macrofunction" definitions (this is Wolff's idea) → (.18)

,12: It mite be possl. to write regulare Recursive Definitions using t. idea of 261.28 -.40.
T. recursive defns. that would be enabled would be ~~of~~ t. form:
Defn. of F(●): F(1) = 1 : F(X) = X*(F(X-1)). [defines F(x) for positive integers only]
I'd have to find a way to express this in t. Z141 formalism.
First Go thru t. formalism of 261.28 -.40 in some detail.

"REFORMULATION"

.18 : (.12) Actually, this "Reparsing" could amount to an important "Reformulation" of t. code [Theory]! One time it may occur is when a new defn is made or t. pc of ~~[illegible]~~ old defns has changed much — Both occur w. Growing Corpus. While this may be a common type of Reformulation, it is probably not ~~t.~~ t. most General kind.

One kind of "reformulation": TM lrns "1+2 ; 3*7, 4-8, 6÷3;" as individual ~~cases~~ classes! Then it looks at the soln expressns for ~~[illegible]~~ all of them & creates ~~a~~ simpler expressn that covers all cases. — & T. "reparsing" of .05 ~~[illegible]~~ does not cover this. In this case perhaps "Integration" is a better term than "Reformulation"

The reparsing of .05 could result in an entirely difrnt looking code for t. corpus!

Anyway: T. problem I've been working on, is to descrb. a less El. way of doing Q,A induction than 257.34 -.40. 257.38 -.40 is pretty "standard" in sciences, I imagine. On t. other hand, the Non-El method of 261.28 -.40; 262.01 -.02 is close to ALP, but w. t. advantg of (.01 -.02). As is, it is not ordinarily practical, for a large corpus, but it could be used for a small augmentation of t. corpus (like Q_{n+1}, A_{n+1})

Tho 261.28 ff is probably usually used to evaluate a parsing of a corpus. The actual functions defined, are normally found by other means (as in ~~[illegible]~~ ~~[illegible]~~ t. old ~~[illegible]~~ Z141 ... for finding nouns by noting freqs of occurrences.)

263.01

Bulg.

.01:262.40 Another tack on this problem: stoch CFG's give a pd. on finite strings.
— So its a sort of soln. to the Big induction Problem. It works by assigning pc's to various
& rewrite rules (or Matrix Multiplication rules.), Z141 does this also. —
for BoB the grammar & the (proper) corpus.

Hvr, the function dscrbd in 261,28–.40 are deterministic: Could I do a similar
thing & get problistic functs? Well, as a matter of fact I do! The functions
are constructed problistically, so what 261.28–.40 does, is give a P.d. on
.08 Functions. These act (in //) on Q_{n+1} to give a p.d. on A_{n+1} (!)

.09 → It may well be that "Backtracking" is the only ≈ non-el. way to
get better models in a "Turing/Universal" way. [In the follg discussn, (QA_i) [≡ Q_i, A_i]
is meant to be a large chunk of the corpus w. a common soln. (common Macro function)]
So, say I get a soln fn for [QA_i: $i=1,n$:] Fn doesn't work on the QA_{n+1}, hvr.
So I backtrack: When Fn was found as a soln for we also had
other solns that were not as good (if we did any "oversearching") — perhaps
many of them. In "backtracking" we try these solns on QA_{n+1}.
If none of them work, we go back to the QA_n & look for solns that work for both
QA_n & QA_{n+1}: If we can't find any we may backtrack to QA_{n-1}:
To try to find a soln that works for QA_{n-1}, QA_n & QA_{n+1}

Hvr, Normally in inductive coding, we don't use such Non-El. Methods of Backtracking.
We use heuristics that narrow the search down considerabley & in doing so
we find solns faster, but we tend to Miss certain (V.G. impt.) solns that a Non-El
.23 technique would find.

In General, when I work problems using 257.34 a/o other El. methods,
I will try to see how I can make them less el.. Also in 257.34, the
(Unify/integrate) phase ~~sould dor~~ could sometimes employ the Backtracking
of (.09) ff.

I want to write a review of recent ideas on Non-el'zn of
257.34. Also, Perhaps expansion of the idea of 261.28–262.102 (A. Non-el soln. of T. problem)
plus the note on P.D.'s of 263.01–.08

.20 T. reason this is impt.! One of big advantages of ALP over other induction formulations
is that given an induction system, it suggests how to improve it. I really want a system
that is "usefully open ended". I know that w. finite CB, I cannot expect to
.23 get best possl. induction. T. Q is, given extra CC, how best to spend it.

One way to spend it is on hyp pc trials that haven't yet converged.

.25 I think the spirit of the Soln. to .20–.23 is: we start out w. a general UMC —
biased by whatever ideas we can think of. We use this UMC w. Lsrch to find
solns. to simple problems. We then bias the UMC in view of the solved problems.
We then Continue Lsrch — usd the biased (tho still Universal) UMC.

Hvr, the trouble w. .25 (if it is indeed the trouble) is that the

264.01

.01: 263.40 — "Summarizing Machine" usd to incorporate t. solved problems, is not a complete Summarizing machine, so we have to use "Backtracking" like in 263.09–.23. This is rather non-el backtracking, & my impression is that it is very expensive.

.05 I'd like, perhaps some kind of Mix of the el. method of 257.34–.40
.06 and a Non-el. method (like may be backtracking, but not exactly 263.09–.23.)

So .05–.06 seems to be where I am now.

I want to be able to use Heuristics, yet I want to be sure t. system is Universal.

It may well be that "Scientific Method" as practiced by t. Sci Community, is not universal, that it depends too heavily on certain heuristics — That it does tend to "Paint itself into a corner" so much, that the amt. of backtracking (≡ Recoding) that would be nec'y, is quite impractical! e.g. drop all of Modern Physics & start over!

However Modern Physics summarizes an enormous amount of data — so any new formulation would have to track its models closely! — This is what variations on standard Relativity & QM & "standard model" do.

Unfortly, t. Sci community doesn't much tolerate dissidents!

.18 → Another tack: I think one of t. original ideas, was that one starts (say t. TSQ) using a ≈ ~~all~~ complete set of instructions: that t. initial problems are such (A "Matching" betw. initial probs & initial inst. set)
.20 that t. instruction set makes Lsrch find solns rapidly. α After solving a few new problems, ~~the~~ (The P.d. is Modified in various ways. new defns are made & t. various insts have different p.c's) — Making it easier
.22 to solve what were before, excessively diff't problems, loop to α → (.31)

Note that a full MCT TM can use practically any kind of info to ~~[illegible]~~ improve its (P.D). A very impt. source of info is ~~[illegible]~~ t. set of Traces of its own attempts to solve problems. This is t. source of Many heuristics.

.31 (.22) During the loop of .20–.22, The (Machine/PD) remains universal!
.32 All searches are always made over a universal (inst. set/PD).

→ I have to Reconcile .18–.32 w. t. idea of approximate "Summarizing Machines".

.18–.32 Does seem to be a very (GOOD desirable) way to run a TM!

A kind of "Summarizing Mchn" ~~[illegible]~~ more like t. spirit of .18–.32, is what one does in linear regress: After each new data pt. comes in, one updates all of t. coifs — Of course this induction system is not universal.

.33 261.28–.40 is a complete (Machine/P.D.) formalism. The Remark of 262.01–.02 Makes this system more than just a ≈ UMC for ALP. It does induce a pd. The changes automatically as we code a corpus — we get changes ~~variations~~ in pc's of various concs, & we introduce (define) new concs as we code t. corpus. A hour would seem be (probably) a further biasing of t. pc's — perhaps in a way that depends on previous codes.

Hvy notes: 267.01–.03

~~Star~~ Sears
Silvertone Guitar ~ 1960
Sell to that Guy who wanted to Buy it?

.01: ~~264.40~~: 264.33-.40 suggests that parallel coding trials can influence one another!!
The way this works is: We try a certain code for t. corpus but it fails.
A MCT.TM could learn from this one failure (OSL) and perhaps influence the pc of a code ~ to t. "faild" one!

So how does 264.33-.40 relate to "Approx. Summarizing Machns"?
(I probly want to consider various induction systems & see how they can be made Universal & how they do "summarizing") e.g. linear/nl regress, ~~NN~~ ANN, stoch langs (c.f. & C.D. Grammars) the P.D. of 261.28-.40 and others that I may think of. Does this suggest any amusing classes of Predictive functions?

Def) Sumes Consider summarizing Machns (Sumes): After a UMC has coded ~~[illegible]~~ part of a sequencial corpus, the ratio of 0's to 1's in its shortest code, could be a kind of very weak summary: But in general, this ratio will be ≅ 1 because t. code is "random"
~~If we can~~ We can, hvr, consider non minimal, ll codes: They need not ~~[illegible]~~ be "random":
Consider a pure Bern Seq: T. pc's of its symbols are a complete descn of all generated regys in any of its corpi. — So these pc's are a complete summary of t. p.d.
In t. case of 261.28-.40 The probys are a complete summary of the Bernoulli-like regularitys in t. corpus: These probys also include several "defined Concs".
Conceivable I might find a way to express Conditional probys. (like t. proby ~~[illegible]~~ vector of symbols that follow α.)
Presumably, w. these probabilities, t. proby of creating t. recent corpus, is maximized.

.23 A Big Q is: Can any conceivable P.D. on [redacted] all poss. functions, be expressed by something like 261.28-.40? This formalism expresses or can be modified to express, all conceivable functs. — But all p.d.'s on them?

.26 An analogous Q about P.D.'s on strings: Can all poss. P.D.'s on them be expressed by a Bern-like seq in which various ~~[illegible]~~ finite strings can be defined & given probys.? I would say that this is extremely unlikely! — But consider it anyway.

.29 Well in .26 not all, say we consider only p.d.'s on finite strings that are derivable from summing over t. pc's of all infinite strings that ~~[illegible]~~ they are t. prefix of. So t. pc of a string must be monotone as one moves along it.

.31 Well, I suspect that whether or not 261.28-.40 is "complete" in t. sense of .23, — that that approach is pretty much what our "Scientific Method" is — and possibly its as well as can be done by humans (??)

Th. Q of .26-.30 is interesting! Surely we have more "Universal" methods of assigning pc's to strings! — Probly we can apply them (analogously) to 261.28-.40
Are ~~HMM~~ Hidden Markov Models more powerful than that ≈ Bernoulli Model.? I think so! T. Bern model is a HMM w. 1 state! So 261.28 could be augmented into a HMM. 266.01

0.10.00 Bulg.

CSL equiv. to Linear Bounded Automaton ≡ Trmc

→ see Hopcroft–Ulman "Intro to ~~Automata Theory~~ ... " p225

No: CSL ~~[illegible]~~ lang is equiv. to a finite Trmc w. a tape containing input ~~[illegible]~~ only that can be erased & used for work tape.

.01: 265.40 Well, HMM's correspond to finite State Machines! Which can approximate full Umcs! Hvr, CF Langs & CS Langs are more powerful.

There is some belief that CS Langs are Turing Universal. See what extent this is true. — It would have to allow [redacted] some shortening of [redacted] strings. — i.e. rules like ACBD → AD otherwise, ~~[illegible]~~ it would always be possl. to tell if a string were in t. lang.

It's Not!

Since strings can be made to represent all functs in a universal way, a strong problistic lang on strings should give us a good ensemble of [redacted] pd's on functions, **GOOD!** Hvr, I don't want t. (constraints on P.D.'s on strings) ~~&~~ (265.29–.31). derived on They are not what we want. We want an arby ~~[illegible]~~ pd on finite strings — so like L. Vit's ~~[illegible]~~ Discrete D.F.: Umc's that stop at end of string. T. P.D. assoc. w. a stochastic Lang., rather than sequence extrapolation

There are 2 problems here: First (t. easiest) to get the "language" of 261.28.. to be actually universal! (or Prim. rec. funct. "universal") — say to model Lisp or another functional lang.. Second (hardest): to get t. P.d. on those functions to be universal. I have been using Bernoulli Models, F.S.M models, CFG's, CSG's, Umc's (The "Chomsky heirarchy") Stack Machines

.18 One approach: See what kinds of models are needed to express various heuristics that I will need. [for a list of some of them see my "How to solve problems" sheet.] ● ┴ ●......

One impt. Heur: "Express problem in a difrnt Lang." (A Necessary meta heur is to keep one's eyes open for new "Languages"). Also, t. process of preparing (modifying) t. problem to express it ~~[illegible]~~ nicely in t. new lang, may involve (inventing/discovering) new useful concs that could be used in other ways ~~[illegible]~~ for "compression" — or to simply help solve t. problem in a more direct way.

OSL seems to be outside t. "2 code" model — outside t. Chomsky heirarchy.

Hvr OSL can be dealt w. by auxilliary Mechanisms (Presumably!)

.28 A vague Model of learning that stays universal: We start w. a set of universal concs, so that by suitable composition, we can create any function (a universal inst set in t UMC sense). ~~As we~~ The set has initially all = aprips. After we solve some problems, ~~the~~ we have new functs & we have modified pc's of primitives, but our system is still Universal. ≡ (264.33–.40) which was

.33 a Big "Accumulation Pt." in my recent thot.

.34 As for the "Summarizing Machine" that idea may not be so good in present context. For ~~a~~ practical ~~[illegible]~~ induction, T. Summarzn Mchn. ~~Sums mchn~~ is little more than an approxn to full ALP. The model of 264.33–.40 is also an approxn to ALP. It is my impressn, hvr, that since 264.33... retains its universality at all times, then at all times a sufficient ↑ of CB ~~[illegible]~~ must solve t. problem (if it is solvable!)

6.11.00 Bulg. Rev 162. is previous review

.01 A possl. trouble w. 264.33–.40, is that while it's "universal," in t. sense that it can code any function, it is not (ordinarily) universal in being able

.03 to express any possl. P.d. on all functions.

.04 Hvr, when one begins t. TSQ, (QA tsq), t. system can and does do an L srch

.05 over all possl. functions. It does so & it ends up w. a simpler distribution over all possl. functions, T. almost all of t. wt. will be over only a few functions.

If t. corpus is large, we will have a wt. over all functions $F_i: (Q_i) \to A_i$.

wt. will be ~~[illegible]~~

I'm confused! .04 seems to start out o.k., but but but....!

After the L srch of .04–.05 (if it oversearches enuf), it will have several functions that fit $[QA_i]$ $i=1/n$ exactly

Perhaps a Summary of t. last wk's work: (since 155.01)

① Mainly work on QA lrng.

② 256.07–.10 T. updating problem for QA_{n+1}! No problem if complete (CB=∅)

.15 ALP is usd! Otherwise, Partial ALP is, in general, impractical, (tho Occasionally it can be usd w. Backtracking as an effective updating method).

[In .15 when I spoke of "Backtracking" I meant "BLIND" backtracking: Here, Fn doesnt fit ~~Fn~~ QA_{n+1}; so one simply goes back & finds for Fn solns that fit up to QAn & also up to QA_{n+1}. One should try to find lots of them, because QA_{n+2} will cut down their number again — perhaps to zero! (from this pt. of view, no of o.a. fcns ↓ exponentially w. n, so it gets to <1 fast)],

There are 2 Modifns of t. Backtracking Appr. that make it more practical.

① When Fn does work on QA_{n+1}, we don't just do ~~Blind~~ Blind search on ~~QA~~ data up to QAn: we do Heuristic Search — Being aware that t. solns must fit the known QA_{n+1} as well. We do this, say, in physics, we know QA_{n+1} & we know what ~~QM~~ they have to have to fit up to QAn. This latter is characterized by equations that ~~[illegible]~~ summarize t. data & any new trials must closely approximate these equations.

.29 ② ~~When I day~~ In t. updating Discussn of 256.07–.10, I was considering discrete codes, & a finite number obtaind by "Oversearch". In fact it may be often possl. to ~~[illegible]~~ obtain ∞ codes that fit. e.g. linear regressn: for every ~~sort~~ set of coefs, we have an assoc σ^2 & ∴ a code for t. corpus. The pc's ~~assoc~~ w. each code will depend on its σ^2 & on how many coefs it uses. As new data comes in, we always have codes in this set

.35 that will fit it.

.29ff may be an answer to 256.11–.12

③ Th. Common Method of Updating in Physics, (& possibly in t. TSQ's of Saeeb: ANL)
256.20 –.28 is one only discn of this: other discn in 257.34–.40 ~~256–267.~~

268.01

Rev

162
267

.01:267.40 (4) Batch Processing: 256.30 — 257.~~33~~; 258.06 — ~~.40~~ :

(5) English TSQ 255.32 – .40; 259.08 – .11 ; 266.18

(6) On Recursion in ~~[illegible]~~ T. lang TM uses! 259.12 ff.

(7) Why we want TM to work on BIG problems! 258.38

(8) On "Summarizing Machines" 259.34 - .36 ; 266.34 – .40 ← these 2 are not v.g. refs!

.06 (9) T. General (supposedly Universal) model of TM, but is also practical. ← recursive

264.19 – .32 (Use of functional language: possibly recursive: → 262.12

266.28 – .33 ; 264.33 - .40 ; (261.28 – .40 ~~which~~ gives more details on how this is supposed to work)

0.11.00 Bulg. Cosma Shalizi

TELNET Visiting for > yr. ~~Matsosaa~~ Edo.

[redacted]

.01: 268.40 Definitions of ~~of~~ Universal Lrng [redacted] ~~Algoritm~~ Algorithm

[SN] It would seem that A R(x) would be expressible by simple a costal loops!

1) for QA lrng:

Given a corpus $[Q_i, A_i]\ i=1/n$. T. Algm gives, as cc↑, successive approxns of $P(A|Q)$. [redacted] which we call $'P_n^{*}(A|Q)$ (for cc = essentially $'P_n^{*}(A|Q_{n+1})$. we want $\lim_{r\to\infty}$ to be within a constant factor of t. ~~true~~ true $P(A|Q)$ whenever such a P exists & is describable by a finite string (variations when P is ~~[illegible]~~ partly dscrbd by ~~[illegible]~~ continuous params)

Furthermore, if ~~[illegible]~~ t true $P(A|Q)$ is indep of ~~[illegible]~~ n, then this "constant factor" will be indep of n for $'P_n^{r=\infty}$. (or another way of looking at it, as $n\to\infty$, there is an upper & lower ~~[illegible]~~ bound for this constant factor).

→ Essentially, all I'm saying is that t. successive approxns. to $P(A|Q)$ should have t. conditions of Sol 78 T3 and t. will converge nicely to t. rite values, at the proper rate.

{N.11.00 Partial rec. functs are effectively enumerable. Better to use them!}

[redacted] One way to do this: (~ to ALP): Consider only Prim. Rec. functs.

.15 Let ~~[illegible]~~ $P_i(A|Q)$ be a enumeration of t. functs ~~[illegible]~~.

we can assign them $p_i = 2^{-i}$ and this is a complete prefix set since $\Sigma 2^{-i} = 1$.

This isn't much diffrnt from ALP & has all same disadvantages.

.18 Ideally, we'd like to list all functions $P_i(A|Q)$; ~~#~~ $\left(\prod_{j=1}^{n} P_i(A_j|Q_j)\right)$ * $\Sigma P_i \equiv \phi_i$ ·· in order of ↑ ΣP_i is t. apriori of $P_i(\cdot|\cdot)$. | List them in order of ϕ_i

~~Then try them in ... order of ϕ_i ...~~

268.06 Gives

.21 268.06 Gives ~~266.28~~ refs to 266.28–.33 & other dscrns of a lrng system that seems to be universal.

At any time, if we use a nite ~~[illegible]~~ cc, it will ~~[illegible]~~ eventually try t. correct function.

— so it will ~~however~~ be worse than "t. correct function" by a ~~constant~~ constant.

But this constant is ~~[illegible]~~ based on t. aprps of various functions, & these change as it ↑ & we progress in t. tsq. If t. lrng system is any good, t. constant factor, will ↑ w.n (I assume!) ← ☺ Hvr, if t. rite funct is included, any of t. lrng systems I consider will probly give it a high wt. (No! — they will give it a hy [redacted] apsi part but t. apri part need not be so hy — in fact,

.29 [if I don't have a adequate soln to t. "Scaling" problem, t. apri part will be

.30 "Swamped out" by many, many new conc. defns!

In ~~[illegible]~~ .15 ff I wrote about a MTM system for QA probs. The "functions" I try ← are not problistic. But deterministic. are accepted only for a perfect fit over i = 1/n. For i > n they give a not infinitely sharp probability distribn. {Actually false. In .15 I used $P_i(A|Q)$, a "soft" problistic funct}.

Hvr, if, in t. problem of ~~[illegible]~~ .15, I regarded it as a NMTM, I ~~[illegible]~~ write want to use problistic functions for trials. One simpler way is to use deterministic functions w. an "error cost" function" for correcting them. Formally, this is t. same as t. deterministic treatment of .15 ff, but it seems to make search easier, because we get some result for every function we try. Also we get (useful) grey feed back to ~~[illegible]~~ steer t. search over functions.

This is true only if $P_i(A|Q)$ are not problistic, but if I create yes/no functions.

270.01

0.12.00 Bulg

Try to find TELNET on Netscape.
② Intr explorer.

.01: 269.40 : There are 2 common ways to get corrections: ① If output of function is binary seq., the error would depend on no. of output bits & no. of error bits. I found 2 reasonable functions that did this o.k.

② If output is real (or complex) m.s. error can give amount of info needed
.05 for correction.

A more general form of function (of 269.15, .18) is $f_i(A|Q)$, in which $f_i(A|Q)$ can take arby forms. In 269.15 I usd Prim. Rec. functs; so they were enumerable, and I could assign pc = 2^{-i} to the ith one.

But perhap that is not t. pt.. We can (perhaps) easily find a set of functs to such that is "Universal", — that will work for ~~a~~ simple problems that are "matched" to t. set of universal functions. The Q is: when we "learn" we modify wts on t. [redacted] sequence of functions [redacted] was search on: Do we retain universality?
.13 If so, what happends to t. "constant factor" <does it expand (desirable) — or
.14 get smaller (Undesirable) ?>

Probably its easy to retain "Universality": The big Q's are .13-.14.

.16 269.21 -.30 discusses t. "Constant factor" problem.

.17 ~~SM~~ Going back to the "Summarizing Machine Problem:" w. a complete
.18 (ec = ∞) Summarizing Machine, there is no problem (except, perhaps osc (?)).
.19 For a summarizing machine that uses only a small no. of codes, this no. will ↓ exponentially
.20 w. corpus length & eventually we have no codes & have to backtrack.

One way to sort of deal w. this is to have a very large no. of codes — i.e. every trial we make is ~~a good~~ an acceptable code, but in most ~~ca~~ cases, t. pc of correction (.01-.05) is very small & t. code gets very small pc.

However, we can hope t. same large no. of codes for t. corpus as n ↑, because every time n ↑, all of t. old codes ~~&~~ are still valid, but they have diffnt amounts of "correction pc". This [redacted] seems to deal w. t. exponential ↓ in t. no. of valid
.27 codes in .19-.20. [redacted] ←

However, [redacted] the .18-.27 will give lots of codes, I suspect that they will not be good codes, because they are all t. same initial function, but w. various amounts of correction for each corpus ↑. Think in terms of a coding Q,A ~~problem~~ pair sequence. We start w. 1000 functs that map Q into A. ~~For~~ Each of t. functions can do t. entire corpus, & each will have corrections for each $Q_i A_i$. So, essentially, we try each of t. 1000 functs on t. entire corpus & we add in t. needed corrections for each to get it total bit cost & → pc est. I don't see much "learning" taking place.

Could I do ≈ .18-.27 with new def functions, changes of wts. & "reformulation"? ① would it lrn? ② would it be universal? ③ would t. "constant factor" remain large enuf to be practical.?

Using a "Gray" fit criterion, it would be possl. (perhaps) to use conventional GA (or GP?) to find short codes! The "corrections" need not be part of t. code" 271.01

04/3.00 Bulg.

.01: 270.40! as far as t. GA GOF (goodness of Fit) is concerned: It ~~is~~ is simply part of t. info that
.02 computes the GOF scalar.

.03 Hvr, I think t. Main Problem: My expected Method ~~for~~ for TM is 257.34-.40 Is it universal? If so, does t. "const" c remain large, — or does it stay large enuf to be practical?

.06 Well, in 257.39.., we normal (at first) design a TSQ so that "c#", (which is essentially ≈ t. C.J.S.) is ~~reasonably~~ reasonably low.

Perhaps t. real "Bottleneck" of ≈"Universality" is "Good heuristics." TM could read Math Literature, try to verify proofs, etc., w. t. goal of finding heuristics in this Corpus. Perhaps TM could look at Notebooks of GAUSS & ~~R~~ Ramanujan, etc: General books on heuristics mite be of some value.
T. Book "Numerical Recipes" & some Books by Knuth may have many heurs.
Also try Carr book (that ~~R~~ Ram.. got his training from) for heurs.
~~My~~ My "General Problem Solving" list.

So, is 257.34... "THE Method"? (Many details need to be worked out).

SN There is t. general Q of "how ~~shortest~~ short a code is "adequate"."? In MTM, this is easier to answer, but even then, If we have a very long proof, would like something shorter!

.20 On t. adequacy of 257.34... ; This seems to be t. way t. Sci Community works. There is t. general Q. of whether "T. true models of t. Universe" are accessable via that path. It may be that by starting out on a difrnt path, one would end up w. considerably difrnt models: some of which mite be far better in certain Domains, than current "Science".

.25 So: I think that t. **Problem is not** [redacted] "Is 257.34 Universal", but "Is it likely to give ~~bad~~ good models for Various Domains" — i.e. Does it have ~~[illegible]~~ "reasonable" CJS's for t. "True Models" of those Domains. — A system could be universal, yet have excessively hi **CJS for** impt problems, for t. **TSQ** that we would use for it.

As is, I am assuming that I can give TM an education close to what a human would have, plus its ability to read ~~much~~ much more than a Human. The Selection of "what to read" by TM will probly be Much difrnt from that which a Human would chose! So I mite want to Tell TM what areas to read.

.34: 257.39: A way to Non-elz. 257.34 abit: when TM is given (Q_{n+1}, A_{n+1}), It tries first to find a function that will solve $[QA_i]\ i=1/n+1$. ← (The non-el problem)
This can, be done by direct Lsrch, using a suitable guiding pc function.
Hvr, One heuristic way to do this is 257.34-.40 i.e. They first find a funct to do (Q_{n+1}, A_{n+1}) only, then ~~[illegible]~~ (unify / integrate) this funct into t. ~~old~~ old one for $[QA_i]\ i=1/n$.

(Actually, this is not a bad non-elzn!)

272.01

Bulb

.01: 271.40: 257.34··· can further be genzd. so its more "MCT TM". The Q's can be inv. probs, OZ probs, ~~induction of Time Series or Bag induction~~, and t. A's can be t. answers to these problems. [6 would work in an obvious way for inv. problems. For OZ problems, t. Q would be t. problem dscn: M(·) & CB. ; t. "A" would be a reasonable result. [This doesn't seem reasonable!]

Anyway Bag induction could be done w. null Q's & various A's. ~~A's ... of Q's & null~~.

How to do Time series induction is unclear.

Anyway: A better way would be to do as MCT says: Have 4 (or 5 if we want Q,A as a special mode) modes: ① Inv, ② OZ, ③ T.S., ④ Bag induction, ⑤ QA. .09

All info from any soln. or trace of a soln. is put into t. **Grand PD**

All searches for solns. are simple Lsrch's (except for OZ problems — in which case we have Lsrch over O.T.'s (which are non-simple objects)

T. "Grand P.D" also contains info about O.T.'s. Given a OZ prob dscn as input, the "Grand PD" gives a P.d. over appropriate O.T.'s (≡ opzn. techniques).

Def GPD There seems to be an inconsi in my ideas about t. GPD (Grand Proby Distribn). In an OZ problem, t. GPD would give a direct ordering of trials. (a cond'l pb., w. t. prob dscn as "condition". On t. other hand, in heuristic search, one has sub-goals & t. GPD tells one how to achieve these subgoals. — which is not compatable w. GPD order t. INV trials directly. → Note 273.27

.20 One way out: Given a INV problem: 2 ways to solve it:

① Use Lsrch on GPD's directly obtaind cands ② Try to break down problem into AND/OR Net & solve it. .22

↑ This part is unnecy: ② Does it anyway whenever it is reasonable.

[SN In all INV probs: A proof that there is no soln. is also an acceptable soln. So usually one tries to find soln and prove there is none (unless one can prove soln. exists, which disproves non-existence of soln).]

Actually, .20 - .22 is pretty much a standard Heuristic. At t. Begining TM will not be able to do much in t. direction of finding AND or OR Breakdowns of problems but later, these will become very common. My Genl. Soln. of t. AND/OR net may become impt. — (Tho I have a general Methodology, I've mainly worked on "OR" problems and even in them I'm not so sure of t. soln.) .30 273.02

Most Math probs of any dffty at all, are broken down into AND/OR nets of problems. Xfmg a problem into an AND/OR net can involve Great Intelligence!

Both AND & OR skills will have to be lrnd.

So: TM would do Lsrch (for INV problems) only if they were "atomic" I.e. not breakdownable into AND/OR nets. : Unless any of t. other 4 types of (.09) could be "decomposed". Sometimes Time Series & Bag induction are decomposable, "OR" by xfmg t. original data in an invertable way. (This may be done in the linear → quad → cubic soln. of eqns!)

A time series or Bag, mite be better viewd as t. AND of several T.S. or 273.01

.01: t. ~~[illegible]~~ Bays mite be considered as t. AND of several Bays.

OZ problems can be ~~[illegible]~~ formally xfmd by a monotonic function of t. Gorc.

In M2R problem $x \to \ln x$ or $x \to e^x$ often simplifies an optzn.

.03: 272.30 One ~~[illegible]~~ on this heuristic: start out w. Lsrch on a problem: if it isn't solved by a "certain time", then ~~[illegible]~~ try breaking it into a AND/OR Net.

This "certain time" would ~~have~~ to be lrnd by TM.

In solving probs (all sorts) by Lsrch, t. Lsrch is not always to find a

.07 direct soln to t. problem. — T. Lsrch is on "What to ~~[illegible]~~ do next" → .27

Def ~~#~~ Info on ~~#~~ tasks of this sort is contained in GPD (Grand PD)

.11 $\boxed{\text{SN}}$ On Updating GPD: In linear regressn after n data pts, one has this vector of coefs, $\vec{a}_n$, that "fits best": when data pt. n+1 comes in, the search ~~[illegible]~~ for t. update on $\vec{a}_n$ can be in a "small region" around $\vec{a}_n$. If $\vec{a}_n$ is a k dim. vector, this search is over a k dim space, say over a small Gaussian hyper space region:

In ordinary linear regressn, t. data in this space is usually isotropic; but

.17 in non-linear curve fitting, it ~~need~~ need not be isotropic. ● Perhaps t. easiest way to do this: Th. ~~#~~ $\vec{a}_{n+1}$ that gives zero error for t. $n+1^{\text{th}}$ data pt, is a k-1 dim space ($\vec{a}_n$ will normally not be in this space). Draw the shortest line from $\vec{a}_n$ to this k-1 dim subspace (it will be orth. to t. space). We can then compute just where on that line t. "best fit" $\vec{a}_{n+1}$ should be. The tradeoff is betw. error in t. $n+1^{\text{th}}$ data pt. v.s. error in all of first of t.

.23 datapts.

To what extent is t. general problem of updating GPD (for or by (numeric / symbolic) functions, similar to .11 – .17 & in particular .17 – .23 ?

.27 (.08) Perhaps an Easier Way to get heuristic info into GPD!

A "soln" to a problem is not just t. final soln, but t. trace of t. soln — w. perhaps emphasis of t. "correct" decisions path" in t. trace.

This is analogous to: $\sqrt{81}$ "is not just "9" but t. process by which "9" was discovered. A ~~[illegible]~~ simple case would be t. soln. of

.32 (linear (non-linear) Equs. see 275.13 & 275.03–.14 for relevant "Comments" on this →

.33 (.27 – .32) looks like a very impt idea that I've forgotten! → 275.01

I tend to think of $\sqrt{10}$ as an oz problem (inversion of x^2 ~~[illegible]~~ — finding $x \ni x^2$ is close as poss. to 10)

But " " " " " solving linear equs as inv. problems, tho they can be solved as oz [redacted] probs.

One kind of "Hint" for solving linear equs would be to given example trace. The "degree of Hint" would depend on how close t. actual problem was to t. example.

274.01

0.14.00 Bldg

~ $\boxed{\text{REV}}$

rows
162
207
268

168.40 ⊂ Reviews

5 types of probs:
Inv, Oz, T.S. induction, Bay induction, Q A.
1 2 3 4 5

.01: 273.40 Impt Ideas since 268

① (269.01) Defn. of "Universality" of a learning Algm.

② Idea that ①↑ is ≈ equiv to small CJS for all of t. problems in t. TSQ. 271.06

③ à 269.29 that we certainly have to deal w. t. "Scaling" problem, if we want "universality"

.05 ④ For t. Model of lrng of 257.34.. : ⓐ One kind of very impt. heuristic 27?34 (≡ Solve problem "locally" then unify "globally") — can be any of 5 types.

ⓑ Generalizing ~~257.34~~ 257.34 ... : so "Q is descr of problem" (INV, OZ, induction, QA, ...)
"A" is the correct Answer. (272.01 is in this direction, but not as good!)

~~③~~ ⓒ Use of standard AND/OR net for soln (as heuristic) (272.20)

ⓓ Defn. of most problems (like solving eqns) is that "A" must be t. ~~Trace~~
"~~Think~~ Trace" of t. soln. (not t. soln.) [Note 275.03: TRACE contains many properties ("slots")

.10

What (beyond ~~.05~~ .05–.10) must be added to 257.34 to make it a "complete" lrng. Algm? Well several aspects of it haven't been worked out.. A big one is (unification/integration) ⊂ (257.28) — it is a "special case" of t. more general

Def LTTM$_2$ "Limited TM$_2$ problem" of making a ~~hypra~~ hyper pc model for t. known $[Q_i, A_i]$ set.

Woops! — there is t. original corpus $[Q_i\, A_i]$ set, but more important is t.

Def) discovered $[Q_i\, A_i^{TC}]$ set: $\boxed{A_i^{TC}}$ is t. Trace soln(s) for A_i..

Do we want TM$_2$ to work on a hyper pc model for $[Q_i, A_i]$ or $[Q_i, A_i^{TC}]$?

Perhaps BOTH! from a LTTM$_2$ pt. of view A_i^{TC} implies A_i (≡ can be derived from)

So they would be equivalent.

Hvr TM$_2$ (unlimited) is mainly interested in near future & somewhat more distant) Depending on "Horizon" as Given by USER

future ~~averages~~ of cc — (perhaps ~~weighted~~ weighted cc — wts. ∝ "importance" of problems)

6.15.00 Buly.

274.13
a kind of
Review

.01: ~~[illegible]~~ : (273:33 Spec) : 273: 27-33, on using t. TRACE of TM's soln. to a problem as t. TM's remembrance of t. "soln." of that problem!

.03 Well, t. "Soln" can have many "slots" properties: The "Trace" is t. most general one: The actual final "value" is another (kind of "summary"), The cc is another, Perhaps other impt. properties of t. trace. (was it recursive? Did it have loops? What where its I/O "Types"; Range, Domain . . .

Anyway, I had previously been considering using (only) the function trace of t. soln. as t. "soln". The "function trace" is t. same as t. trace, in cases where funct traces are possl. The "trace" is more general: can be usd when t. lang. has loops ~~&~~ a/o recursions. T. trace also has cc info. ȧ t. Funct. trace — well it does also, since it is = t. Trace for t. problems in which it is usable.

.13 So 273. 27-33 is really only a (mild) genzn. of what I've been doing
.14 w. my "Functions as Answers" approach.

Another possl. Genzn. The ~~f~~ Trace functions I've been using correspond to Deterministic Automata. Hvr, a more general lang is non-deterministic automata (in which a state may go to ~~1~~ or more other states), ȧ its probabilistic variant, in which t. state transitions have pc (Hidden Markov Models)

N.B. Tho t. HMM allows loops of states, it is still much weaker than cf languages. (Phrase Structure More generally Trace langs ≡ PSL's)

In MTM we want >1 "soln." to t. problem if possl. Do I want to express this set of solns as a (Non Determ. lang, or a stochastic lang?

The Trace of a soln could involve exploration of t. paths of a ~~a~~ Non-Deterministic Lang.

Given a INV problem, Giving a stoch lang. as a "soln" is a way of narrowing down t. pc for Lsrch.

→ SN The First GHT says that $\frac{pc}{cc}$ order is best for min. Σ cc of soln. Hvr, this may not be our Goal! We may want to be getting info: in which case, we may broaden our search! (Perhaps the ~~a~~ Monte Carlo srch of GA?)

In fact this may partly justify t. otherwise ~~a~~ very-wasteful Mt. Carlo srches of GA. Hvr, GA usually (or always) doesn't use t. info very well — i.e. it doesn't use ~~[illegible]~~ info about "poor fit" trials very well. It does get more "Diversity" than purely honest "Lsrch.

T. main idea of t. TSQ: In t. initial TSQ, I will be very aware of all t. concs. usd ȧ what TM's cjs is for all probs. I will use Lsrch for solns. I think that ~~even~~ this is ≈ t. only system that will work! To have acceptable cJS for a problem t. nec'y concs have to have been acquired by T.M. After TM has acquired a reasonable no. of Good concs, I want to be able to devise TSQ's w.o. ~~detail~~ my detaild understanding of their Conceptual Structures. After TM has done enuf TSQ's I expect it should be able to continue w. less ȧ less care on my part in writing TSQ's.

.01: 275.40 T. most immediate Goal is to acquire a fair understanding of Algebra: (possibly enuff to work somewhat hard probs.). Next, learning to understand English Text about Algebra.)

T. ^(Olrng.) methods of 274.05–.10 should be unncy to specify. T. ~~[illegible]~~ entire lrng. techique should be expressed in t. "English soln." of t. problem.
Hvr, 274.05–.10 may help me express T. English soln. in ^(clear) unambiguous for.

.07 Also in t. English TSQ & soln: When a heuristic is found or we just want to incorperate t. info "discovered" in a ^(Soln.) ~~soln~~ ~~to~~ ^(of) a prob. in ^(t.) ~~a~~ TSQ, — First Tell, in English, just how this new info is supposd to influence subsequent Searches for prob. solns.
Then, find a formal lang. in which ~~[illegible]~~ modulns can be inserted so as to modify t. P.D. expressd by t. modify lang. so it expresses t. heur or t. knowledge lrnd from t. problem soln.

66 $8\times8=2$ $8\times\frac{2}{3}=5'{\times}5'$

.13 → So t. English TSQ & solns. should ~~[illegible]~~ guide language design.
By "language", I mean TRMC or "P.D. assoc w. a TRMC".

At times I don't know just how I want t. info in t. newly solved problem to influence future searches. — I often have a qualitive idea, but no key as to a quantitive idea. T. quantitative part usually comes from ALP analysis.

→ Hvr, I do have to express what I'm doing in some exact lang, & as soon as I do so, ALP is able to give me pc's. I need a lang that can express heurs as well as "problem solns".

[SN T. no. of Bits in a heur or rule doesn't nesly measure its pc! Any "OR" statements mean the rule ~~is~~ ~~a~~ really ^(has) several parallel codes so pc ≈→ pc×2 for each "OR"!
"And" statements ≈ divide pc by 2.]

So, say to start, I have a lang. to express functions. A set of primitive funcs & ~~[illegible]~~ composition rules

.25 Consider $3\left(\begin{smallmatrix}\times\\ \div\\ -\\ +\end{smallmatrix}\right)4=\left(\begin{smallmatrix}12\\ .75\\ -1\\ 7\end{smallmatrix}\right)\leftarrow\boxed{?}$ type problems: Say TM recognizes "nos" & "symbols (= names.)

.26 add, sub, div, mul operators are available; They go from 2 nos. to 1 no.

.27 So it is "natural" to try them on t. 2 nos. in t. problem to get t. soln.

I don't have a lng. for T.M. to do such "reasoning".
The fact that "sum", say maps no. pairs into nos., is an "empirical fact" found by TM thru statistical analysis.

[In general, all of Math, including axioms, can be regarded as "statistical info" obtaind w. very large ss's & t. very likely to have pc close to 1 or ∅.]

So: I want to show how t. "reasoning" of .26–.27 is ~~obtained via sta~~ t. outgrowth of statistical observations on t. corpus. It ~~may~~ should be possl to assign ∅ pc's = ∅ol to natural ideas of I like. — i.e. the generating P.D. should be at least that flexible!

There is t. general Q of how to express various Constraints obtaind by "discovery" into t. P.D. in a way ∋ .Lsrch could still be usd effectively. ^(also see 277.29 & .32 for good ideas on this.)
I had this idea that PD's could be expressed in many difrnt ways ∋ ~~[illegible]~~ that 277.40

0.18.00 Bulg.

.01: 276.40 One big trick in problem-solving Analysis was to view a pd in various ways — thus enabling Soln. of a problem. ~~[illegible]~~ "Viewing in various ways" is a Major Scientific Heuristic. (e.g. change of coörd system; Time domain ⇔ freq. domain; Minz ⇔ Maxz; G or c ⇒ monotonic function of G or c;)

One (apparently "IMP") idea is that t. Trmc formulation can express any desirable P.D., ī in this form it seems ~~to be~~ easy to implement Lsrch.

One apparently "simple" way to create a Trmc from a P.D. is to list the strings in pc order, then assign pc's via Huffman Coding. Every finite string has a p.c., so one starts w. null, 0, 1, 00, 01, 10, 11 ··· ī assigns p.c. to each ~~[illegible]~~ via t. known P.D. We then ī somehow assign Huffman Codes to them. ←(No! This is wrong! We want to assign Huffman codes to all strings of same length (since their ~~≥~~ Σ pc = 1).

So first assign codes to 0 ī 1, then assign codes to 00, 01 ī 10, 11: The code for 00 say, will be " code for 0", w. extra bits.

→ This will not work! A code for an n bit string will be n bits long!

Perhaps read Cover-Cheng Paper. (But actually, I don't think this impt. in t. present problem: ? See .17

.17 So an impt. Q is: Is it feasible to put all P.D. into Trmc form ī for Lsrch? Can I do ≈ Lsrch w. a less constrained form of P.D.? All I really need is cands. (or strings) in ≈ P.C. order. (Perhaps knowing ≤c at each pt. in derivation could ~~[illegible]~~ relax need for exact knowledge of pc ordering)

Anyway, I want to look at t. P.D.'s that I will be using, see how ~~the~~ Various Heuristics ~~effect~~ their pc ordering: see how they effect Lsrch.

As I see it: T. main Q is, how problem solns affect t. PD (since Heurs can be expressed as "Prob. solns" w. suitable SSZs), in terms of ordering of pc (ī $\frac{pc}{cc}$)

We don't want t. ~~[illegible]~~ approx. ordering of trials to leave out any very promising ones — tho it could include many ~~[illegible]~~ low pc trials. — This is an analysis of t. ~~[illegible]~~ relative badness of different kinds of errors in ordering Trials.

.29 One way to get reordering suggested by a Heur: To look at t. reordering that occurred in t. "Hypothetical" pseudocorpus that discovered t. Heur in Qustn. — Perhaps make an "analogous" reordering in t. problem that t. Heur is to be applied to.

.32 (SN) Perhaps → Every time a problem is solved, a "demon" is created to ~~look~~ look for probs of that "type" in t. future. (or for "Analogous problems"!). Periodically, various demons will be (unified integrated), perhaps w. some compression as well. Mindful of Selfridge's Pandemonium.

A nice feature of .32: It can deal w. OSL. It ~~[illegible]~~ can be an Impt way to deal w. "scaling" (280.34)

278.01

0.19.00
Bulg.

.01: 274.40: ok: Back to ~~to~~ t. ANL TSQ of 276.25 [redacted] & t. Heur of 276.26–.27.

It is possl. that TM has studied t. 4 arith. operators statistically & "knows" much about their "slots" properties.
[With random nos as test objects, its unlikely that T.M. would discover that division by Zero is imposs.!] Perhaps t. best way to put this into to TM would be by "hint" / or (Some kind of "Telling". Another way: use ~~[illegible]~~ $\frac{3}{4}$ = ~~[illegible]~~ .75 but $\frac{3}{0}=1$, $\frac{4}{0}=1$ or $\frac{3}{0}=3$; $\frac{4}{0}=4$ etc. or $\frac{\pm 3}{\phi}$ = largest positive possl float. $\frac{-3}{\phi}$ = most neg. possl. float.

Some integers have special properties.
Another way: T. special nos, φ, 1, π, e, 2, 3, ~~[illegible]~~ −1, −2, −3, have special properties ← Another "Lenat Heuristic"

One nice heuristic that Lenat used: When a new object type was discovered/invented, AM would investigate its properties in a systematic way: There was a certain ~~[illegible]~~ set of "slots" & it would fill each of them (to varying degrees).

But Back to 276.25 "in English": Perhaps stay in English more: Not too much formalizn. at this pt. Yet! The t. "details of formlzn." of 277.29, .32 are v.g.! T. [redacted] heur 276.26–.27 can also be in Eng., ~~[illegible]~~ As a separate problem, we omit each how TM acquired that info.

So do 276.25 w. at least 2 heurs! ① T. operators mapping 2 nos. into a no. ② The similarities of t. structures of 3+7, 3×7, ~~[illegible]~~ & of ~~[illegible]~~ add 3,7; Mult 3,7 …. The general idea of ANL (recursive interpretation of expressions) would ~~be~~ ~~[illegible]~~ consist in a recursive fashion a/o one having a search.

A Main line of rsch: That we may want to concentrate on "ordering of trials".
.20 So, in English, figure out rules for ordering trials, then [redacted] find way to assign pc to these trials.
These 2 operations ~~may~~ may interact a bit, but see if its possl to do .20.

So: we start "knowing" T. function needed will be a combn. of t. 4 binary alg. functs & perhaps ~~other~~ other functions. Since final funct. must ~~[illegible]~~ have numeric output, The reasoning about "Types" (numeric v.s. Symbolic) can be easily formalized via "Type" language/algebra. The 4 ~~ari~~ binary arith operations are all t. same ~~wrt~~ wrt Types on I/O. — So this is a big simplifn. of t. problem. For complexity level of 1 ~~[illegible]~~ functions, there are only 4 possl type maps. $n_1, n_2 \to n_3$: for inputs 1,1; 2,2; 1,2; 2,1:

[Also, there is t. possy of ① functs w. no input ② Unary functs of inputs
The only unary functs we have are x (Identity), & $\overset{2x}{x+x}$; $\overset{0}{x-x}$; $\overset{x^2}{x^2}$; $\overset{1 \leftarrow \text{unless } x=0}{\frac{x}{x}}$

.30 Also "unary" functs: $x+a$ (a arbitrary const.) $x-a$; $x \cdot a$; $x \div a$. — So sort of, 2
.31 functs, but a can have diff't a priori in $x \div a$, $x \cdot a$.

Re: .30–.31 drop it for a while: Just concentrate on t. 4 arith operations & this particular
.33 "Type" heuristic. One trouble is that TM has had no "previous experience" w.
.34 function concatenation, so it couldn't have derived this heur. → 281.06 Actually, it doesn't really need that heur just now, because t. problem is so simple. In fact, Much (or all) of TM's early lrng will be w.o. heurs. It will use "large" CB's for solving these early probs, but, in general, t. probs will be "easy". After this phase of lrng, various heurs can be discovered, because there is a large enough corpus of ~~data~~ ~~tabase~~ on which to base TM's discovery of heurs. (Appart from OSL discovery, that needs

279.01

0.20.00 Bulg.

Oct 5, 00 Nature: 2 arts on Conc. lrng
research, other ~~c~~ Comments

.01: 278.40: only 1 or 2 cases ... actually "1½" cases)

.02 So just try to do a fair amt. of t. TSQ w.o. "Heuristics". — Even if $\overset{(CB)}{CJS}$ is quite hy. When TM has enuf "experience" I ~~may~~ then can try adding sequences of data for TM's discovery ~~of~~ of heurs; see what CJS is needed! A large CJS may be tolerable, since it could take many yrs. for t. Sci. community to discover a new heur.

T. "w.o. Heurs" of .02 may be a bit extreme! take: $\overset{x_1 \;\cdots\; x_5}{3+7=10}$ 5 arguments.
4 binary functions; 5×5 possl. ~~in~~ input configns for each:
So we quickly find ~~x~~ Add $x_1, x_3 \to \boxed{2}\, x_5$ ä add $x_3, x_1 \to x_5$.
We get +, −, ×, ÷, in this way: now w. $x_1 \; op_i \; x_2 \to x_5$ ~~we~~ when we use random ~~operator~~ op_i = random (+ − × ÷) we still get good corpus compressn, but not "perfect" (whatever that means!) (.19)

.11 SN In "Batch Mode", TM mite develope "Curiousity" ä various other behavior patterns ~~simi~~ appropriate to a "creative Scientist" rather than a "Pure Engineer".
— How this works: TM finds that "experiments", "investigations into ~~the~~ properties of functions" — inventing "ä using functions" all do result in often useful abss for future problems. Hvr, unless TM has a v.g. criterion for "interestingness", t. ~~~~ long-run yield of this behavior will be small.
So, t. Sci community (ä Organic Evoln) develope "Criteria for interestingness".

.19: (.11) Some useful functs $f(x_i, x_j) = T$ if $x_i = x_j$; ~~else~~ else F
T, F is somehow used to "control" other functions.
$G(T; \alpha) = \alpha$ ä $G(F, \alpha)$ = undefined.
Fine way to do it "correlns". Th. exact Mechanics of how to have the functs (primitive or non-primitive) controld by "Ops" is not yet clear.
In t. "ob-of algebra", correlns were usd.

SN On 250.34-.40; $\overset{251.01-.02}{251.20-.27}$ I discussed just what t. Bottlenecks were, after the Saarb. TSQ's, ä how I have since ~~dealt~~ developd methods to deal w. these Bottlenecks. It would be well to expand (a) Just what were t. Saarb TSQ's (b) More detail on t. bottlenecks ä just how I would now be able to deal w. them.

First, t. TSQ's themselves! (1) The ~~ANL~~ ANL TSQ ending w. recursive evaln. of Alg. expressns. Expressns were in Polish notation, A Pushdown stack was used in the "derby lang" of t. Model: W. Paul may have suggested that part. (2) A TSQ to learn to solve linear alg equs ä thereby discover t. equivt of t. "Laws of Algebra". (3) A given (substitution ä "desubstitution") heuristic, usd for solving linear equs, then, when $\sqrt{x}$ was added as primitive/soln of quad equs, then when $\sqrt[3]{x}$ was added as primitive, soln of Cubic equs.

280.01

.01 : 279.40 : Bottlenecks ① Dif'ty of writing TSQ's ② The ~~#~~ solns. obtaind only used
.02 complete solns. of previous problems (no sub-trees of functions used). ③ The solns. didn't "scale" well — as t. TSQ continued, t. no. of concs in M any f nonlinearly, so ~~t~~ t. pc of t. concs ∝ ☆ & it became more & more dif'lt (more cc) to solve problems. Problems w. "soln depth" of k required cc ~ $\boxed{n^k}$. In human prob
.06 Solving, cc ~ constant or a slowly ↑ funct of n — more like β^k, where β is about indep of n. → (.34)

↱ Idea of task has k params: AND/(OR) of 2 tasks gives a new ~~[illegible]~~ task with params: Also try to prove any task is impossl. ... Try conserve given cc

Since Saarb: Several impt. developments; ① Sort of understanding of
.08 Soln. of AND/OR nets (WOA problem) ② Soln of MCT ③ Better
.09 Understanding of TM_2's Gorc. → (281.01) ④ Some general Heurs for solns of TSQ's:
e.g. Find Fn ∋ An = Fn(Qn), then (Unify / Integrate) Fn into t. previous soln upto n-1.
This last was done using Ob-Op algebras & ~~[illegible]~~ observing "correlations" betw. certain Ob. outputs & t. ~~[illegible]~~ successful use of certain Ops.

Re: .01 : ① Dif'ty of writing TSQ's: Seemed much eased by MCT — since a much larger scope of "problems" could be put into t. TSQ. In particular, t. use of "Learning Definitions Inductively."! ~~[illegible]~~ (Definitions are usually "Told" to a human
.18 student). One effect of this is that t. need to discover useful sub-trees in useful functions, would be much relaxed. {A Mixed Blessing! — since we do want TM to have this skill — ~~[illegible]~~ Time consuming tho it may ~~[illegible]~~ be!} → [This is because t. "useful subtrees" (like x^2 or maybe sin x ?) would be learnd as "definitions" & would ∴ be ≈ "final solns
.25 to problems" (.01–.02)↑.

Also, I had t. idea that I should be able to use practically any TSQ usable by humans, — just as long as it didn't contain need for R+W. specific info: (That I might be able to insert such info in TM, or modify TSQ so it's not needed. SO: use TSQ from Human Text Books. ② Re: subtree discovery: see (.18–.25) But also it may be that t. TSQ's I was giving, simply didn't have t. complexity of concs, such that sub-tree discovery was needed. {Tho it would seem that in eq. solving
.34 ~~[illegible]~~ subtree discovery would be useful!} ③ on "Scaling" (see .02–.06): The ~β^k is obtaind by finding suitable heurs, for each conc., so TM "looks at t. situation" & this suggests certain concs should be used; this ~~[illegible]~~ ↑ t. pc's of these concs enormously! Also, see 277.32 ~~[illegible]~~ for a possl. way ("Demons") to realize heurs of this kind: Also a way to discover them.

→ (see 283.32)

4 □

.01:280.40 : ④ TM_2's Goal; (280.08–.09) Main breakthru: that much of TM_2's goal must be "USR supplied" e.g. the "Horizon" & perhaps the ratio of TM_2 cc to TM_1 cc (this perhaps this is part of the "Horizon".). "Values" like relative cc to spend on "Cancer cure," v.s. "Theory of Everything" must be USR supplied.

⟷ Could I then just take the Soar ANL system & modify it to fix its ~~deficiencies~~ deficiencies?

.05: 278.34: ~~[illegible]~~ Re: this objection that TM has "no history" at this initial pts. so "it can't have lrnd heurs" : I should be able to "bend the rules" a bit; to enable heurs at this point. I want to understand this heuristic (lrng/insertion) as soon as possl.

Well, ~~overgeneral~~ T. lang for just lrng. the alg operator notations is quite simple — no where near universal. It only has num, num → num functions! It has no Boolean fncts, no way to implement recognition of strings, etc.

I was thinking of a simple functional lang w. definitions implemented ~~[illegible]~~ This Z141 "precorpus" notation. It assigned pc's to codings of the corpus in perhaps a better way than simple TMs would — since it automatically summed over many codes to get pc's. At present, I can start w. any set of Basis functions, & it can generate all compositions of them, & make definitions & uses definitions of any functions it can create. Hvr, it can't yet, do loops or recursions.

One way to introduce loops ~~[illegible]~~ a/o recursion, was my use of functionals. I think I can do both Loops & ~~[illegible]~~ (simple) recursions using functionals. I have written about how to do loops recently, using this ~~[illegible]~~ formalism:

```
~~[illegible]~~ a = ~~[illegible]~~ 3
for i = 1 to n
~~[illegible]~~ a = (a+3)(a/2 - 3)
next.
```

The inputs to the loop are "3", n, (a+3)(a/2 − 3) ~~[illegible]~~ real, ~~Integer~~ function ~~[illegible]~~ (real → real).

The output is real no ("a"); The inputs is (a_0 (= "3")), n, f (real) → real.

```
~~[illegible]~~ x = x0 , y = y0
~~for~~ i = n0 to n.
y = f1(x, y) : x = f2(x, y)
next.
```

input, x_0, y_0 ~~[illegible]~~ real real ; n_0, n : integers (int int)

fncts f_1 real, real → real), f_2 real, real → real)

output, x, real; y real.

$$\left(\frac{13.27}{11.64}\right)^{11} = 4.23$$

11.64 ✓ 13.27 ✓

.32 (L)

```
~~[illegible]~~ f(0) = a
for ~~[illegible]~~ i = 1 to n
~~[illegible]~~ f(i) = G(f(i-1))
next.
```

← This is beginning to look like the loop form of a recursion! (R)

E recursion! ① f(0) = a
② f(n) = G(f(n−1))

Defines f(n) for ~~[illegible]~~ all integer ≥ 0.

Which is also done by

So (.32 L), the loop form is equivt to (.32R), the loop form, but is of lower cc (usually) hvr, (.32L) seems to have a more complex defn. I think that in fact, it does not,

(282.01)

0.23,00
Bldg.

ABcde ABCDE
ABcde ABCDE

.01: 281.40: In that 1982 Lenat Paper (why AM & Eurisko seemed to work), he gave an example of a recursive defn of "2 strings being of = length." Could I ~~sim~~ simply xform that into a loop? Perhaps it is possl (even easy!) for "simple" recursions (is this prim. rec. functs?).

.04 → It looks like I could ~~use~~ use the Z141 "f recorpus" formalism to define loops & recursive functions, in terms of constants & previously defined functions, and get suitable pc's for the functions defined. (.35)

.06

So what I need now, is some more primitive functs. to enable TM to ~~recognize~~ recognize conditions (ops) & control functions ops.

What I want, is a nice ~~way~~ way for Boolean ops to control numeric ops.

A possl way: F(True, f(x)) = f(x), F(false, f(x)) = undefined.
I don't like this because it could lead to inconsistant ~~exps~~ expressions.

or $F(T, f(x)) = f(x)$ $f(F, f(x)) = \phi$.

we want a funct to be $f(x)$ if $\alpha = T$, but $G(x)$ if $\alpha = F$.

so $S(x) = F(\alpha, f(x)) + f(\neg\alpha, G(x))$.

We would define a new funct. $S(\overset{\text{real}}{F(x)}, \overset{\text{real}}{G(x)}, \overset{\text{Boolean}}{\alpha})$

Analogous functs can be defined if (F & G are Both Boolean → Boolean) or (Bool → real) or (real → Boolean.)

Functs of this sort would be enuf (I think) to map 3+5 into add(3,5), & recognize "+" → "add".

Maybe ~~of~~ Types of vars: ~~xx~~ real, integer, Boolean, String.

Note! Recursive functs are normally defined for integers only. The extension to reals can be done by say $\sin'(x,n) = [n \cdot \sin(x)]$ so we can get arbly close approxns to $\sin(x)$. This is not, htvr, the way I had recently been considering defns of real recursive functs. The use of X! to as an example of a recursive funct is o.k., [redacted] as long as it is clear that X must be an integer in this type of defn. It could be used to define X! for all reals if one ~~defined~~ gave all values on say (0,1) as a boundary cond. — But in the case of X! this would be diffclt.!

.35: (.06) Anyway, this Z141 technique is fairly general, so I could end up w. a Universal density function on all finitely describable functions. "Universal" in the sense that every describable function has pc > 0.

This is not yet (at! close to) the universality of the Universal pd's assoc. w. UM's. (283.01)

0.23.00 Bulg

ABcd.

.01: 282.40! Each integer finite string (usually, if not always), defines a function: Usually many integers will define t. same function.

SN T. function defined by a sequence of integers. The upper bound of each int. in t. sequence is uniquely determined by t. integers that occurd befor it.

Can I somehow map all (one) integers into functions? Well, yes: Since I have a p.d. on functions, I can assign ~~integers~~ code words to them, that form a prefix set.

Hvr, since $\Sigma P_i < 1$, all integers ~~without regard~~ strings will not necly represent strings of functions.

.10 Anyway, I guess t. Q is: Given a finitely describable probty density function on all finitely describable functions, will the density function on functions of 282.35 multiplicatively "Majorize" it. (> it ~~for all~~ functions (within const. factor))?

Since ~~all~~ each funct. is represented by a finite set of ~~set~~ finite strings, — Then the universal p.d. on finite strings (t. "Discrete" d.f.) will induce a universal d.f. of functions. (maybe!) .10 Is where I have been before in discussing Universality. <269.01 ff>

.16

5.5 → 7

SN It looks like, recently, I have been "flexing my arms about," w.o. good direction: Perhaps go over where I am, what has been done & what needs to be done! Perhaps look at previous reviews of This Q.

1.5%
10%
150

SN In QATM, the Certain sets of QA's are in Unordered Sets (Bags), The "Sequenciality" The "what comes befor What" in t TSQ is of vital importance. It may be well to think of any t.s.Q. as being essentially a (sequencial) T.S. problem! (Except for "Batch mode" Thg)

→ On t. other hand, it would seem easy to write TSQ's for TM! If I know t. concs needed & their heirarchical structure (The "Concept net"), I should be able to make up examples, & TM shouldn't have much trouble learning t. Concs, ~~using~~ using Languages like 282.35. The main problem (perhaps) is to have adequate heurs, so we can "Scale up". ~~In general~~

.32 In General, a Heur must be good enuf to narrow down t. choices ~~that~~ so t. info needed for choices does not ↑ (much, if at all) as corpus ↑. (is 277.32 adequate?) [see 280.02—.06; 280.34—.40.]

I think in my Sol89 analysis of TSQ's & "Conc nets", I did not face t. "scaling" problem. I.e. I thot that a "Conc net" when obtained, would be enuf to insure TM could devr. t. needed concs. NO SO! t. heurs of .32 are necy ~~$~~, — If t. CJS is to be kept below some upper Bnd as required in .32.

0.23.00
0.23.00

Bulg

.01:283.40: So, it would seem that what I should do is devise a conc. net for t. set of probs I want to solve, then find heurs that suitably reduce t. CJS's.

If necy, any conc. should be learnable by TM thru induction. Perhaps, for "practice" see if I can do this for arby concs.

T. simplest kind of conc. is that defined in "Computational Lrng Theory" (Valiant et al.)

Some examples: ① an example of correct addition: 3+7=10; T. "example of addition" is t. conc. Neg examples: 7, 7+ (meaningless); 3+7=2; 3×7=21

Perhaps learn "addition" example then "correct addition" example

My mind is rather vague on t. forgg. concs. — just what I want them to be — just how they are to be combined, used. "correct" & "addition" could be a combining of concs.

SN Suppose I simply took certain English words that referred to concepts, & Gave TM examples of them. This kind of info, would later be of use to TM in lrng to understand English text.

122 SN A (perhaps Novel) approach to TSQ Design: List problem types I want TM to solve. Give equs./functs/strings that solve them: Give several
.24 alternative solns when possl/relevant. The functs are closer to a conc. net than t. problems. Hvr, for humans, the English (vague) dcrns of t. problems have vital heuristic associations.

So, perhaps in line w. Lenat's AM & Eurisco: Each conc. will have various "slots": Some will relate to equs, functions: Pgms: Recursive, non-recursive forms, ~~Also~~

But also Associational ~~Baggage~~ [Lenat said that Humans found this more useful (usually) for devising heurs, than t. pgm stuff] — seems very reasonable: This would include t. "name" of t. conc. (In English — mnemonic, meaningful for human debugging & following operations of TM). When conc. was ~~or~~ dcvrd/introduced. T. context of t. conc. [where it occurs: what kinds of probs it occurs in ~~[illegible]~~. Also, t. "subject area" (Algebra, Chemistry, ect) generalizns of conc, specializns of conc.

We may devise a "special language" for TM to deal w. t. "Assoc slots" info.

In .24 "Soln" can mean several things: 3 most impt. are: ① ~~[illegible]~~ single number ② ~~Equation~~ function, giving "①" as funct of problem dcrn ③ The Trace 285.01

0.24.25 285

Bulg.

The Fletcher School of Law and Diplomacy
Tufts University

The Edward R. Murrow Center
of Public Diplomacy

ABcdefg $\int_{-\infty}^{+\infty} e^{-x^2}dx$
ABcde

Edmund A. Gullion, Dean

Medford, Massachusetts 02155

*③ "TRACE(s)" of t. soln(s).

.01: 284.40! Of t. soln. : for very simple probs (& some <u>not</u> so simple), T. Trace is t. same as t. function (soln.) definition: It is a pgm (or equivt.) that defines t. function. In ~~a~~ other cases, t. trace will be a <u>sequence of steps</u> that result in t. ~~form.~~ final Soln.. As a simple example, t. final function mite be very simple because of cancelations, or substitution of a ~~very~~ very simple expressn for a complex one. T. trace shows these cancelations a/o subsitutions being done; T. "final soln." does not.

One ~~a~~ (sometimes) easy way to teach, is to give ~~e~~ an example of t. working (trace) of a specific problem (This was t. usual way they used to derb genl. solns in t. very olde days of Math.)

Anyway; TM must then <u>Genz.</u> This specific solns. so it is able to work similar problems.

e.g. show how to solve linear eqns: 2 linear eqns in 2 unks, 3 linear in 3 unks ...

Then can TM genz. to N eqns in N unks. ⟨N eqns in N unks.⟩

So: Try out <u>284.22 ff</u> : Some ~~[illegible]~~ possl. elements of t. TSQ,

.20 3+4 = [?] problem: <u>Soln.</u>, sum 3, 4.; <u>desired trace</u> (at first) sum.3,4 ;
This trace does <u>not</u> ~~[illegible]~~ include Unsuccessful trials. I'm not sure if this will <u>always</u> be true! An <u>unsuccessful</u> trial <u>can</u> have [illegible] impt. info that leads to correct Soln. —
— <u>If one has Suitable Heurs.</u> Otherwise, <u>not</u> ~~—~~ which is t. 4 Case in (.21)

Next set of probs. (or single probl. w. long <u>Random</u> reel): 3*4 = ? .
First soln is mul 3,4. This compresses t. corpus of 3+4 = 7, 3*4 = 12, where "3" & "4" are long random nos: If th. random nos. are long Enuf, we
.29 can compress further. (W.o. this further compression, we get ≈ 50% compression). T. further compression is ≈ 100% ~~as~~ length of random nos. ↑.
(corresponds to ssz if we have > 1 ~~[illegible]~~ example ~~of~~ but <u>many</u> <u>small</u> random nos. — length of 1 big random no. is equivt. to Sum of lengths of
.35 many small random nos) → (286.16) →(286.16)
.36 In <u>fact</u> we can use (.29) ff for MTM to figure out <u>when it has solved</u> many (if not all?) Math problems!

(286.01)

Bulg.

The Fletcher School of Law and Diplomacy
Tufts University

The Edward R. Murrow Center
of Public Diplomacy

Edmund A. Gullion, Dean

Medford, Massachusetts 02155

.01: 285.40 Say TM knows what params in the problems are "random nos." (corresponding to ~~[illegible]~~ general real variables.). As lengths of reals → ∞, compression should → 100%, or for given length/L of random vars, Length of Code of Corpus should be Konstant + f(L), where ~~f(L)~~ $\lim_{L \to \infty} \frac{f(L)}{L} = 0$

f(L) could be a constant (or perhaps ln L)).

Hvr, in Math, 0, 1, ~~*~~ neg integers, positive integers, can be exceptions that would not be caught by Random nos. The most common exception (by far) is division by Zero (and associated exceptions.)

.16: 285.35 (Dropping 285.36 At for the moment): to continue 285.35:

We need to distinguish betw. "+" & "*". "In English"; + → sum; * → mul;
But we have to find + & *; Then decide on a functional formalism (op, or algebra?). (The "finding" * + occurs when 3+7=? alone has to be lrnd.

So: 2 problems: ① Deciding where the nos & symbols (≡ strings) are.
② Deciding a Formalism for functional form of + → sum, etc.
Actually, I don't have to decide at this pt. I have a good idea as to how ~~[illegible]~~ much "Info" is involved, & that's all I need.
I can make a (reversible) decision at this pt.
When the problem 3+7=? comes in: There are 5 positions.
The possl. inputs to "sum" are 3, +, 7, =? : Only 3 & 7 are defined —
"sum" immediately rejects +, =, ? w.o. computing output. This saves a lot of cc, because sum, 3, =, say, doesn't have to be compared w. 10.
In fact, for sum, =, 3; we don't even try 3 before rejection when = is put into "sum" operator (??).

Next, How do we recognize "+" & use it to "excite" sum?

287.01

The Fletcher School of Law and Diplomacy
Tufts University

The Edward R. Murrow Center
of Public Diplomacy

3+7=?

Edmund A. Gullion, Dean

Medford, Massachusetts 02155

.01 : 286.40: Well, try Identity function: $id(\alpha, \beta)$: α, β, can be strings or numbers: output is T or F. (Boole).

We also need a way for Boolean variables to control numeric variables:

One suggestion was $Cn(\overset{\text{Boolean}}{\alpha}, \overset{\text{numeric}}{n}) = n$ if $\alpha = T$, $= 0$ if $n = F$.

So to chose + → sum, * → mul.

.06 $sum(Cn(id(\alpha,+), sum(n_1,n_2)), Cn(id(\alpha,*), \overset{mul}{}(n_1,n_2)))$

A rather complex expression! But much symmetry.

Estimate rcost: $R_{Cn} \cdot R_{id} \cdot 5 \cdot 5 \cdot R_{sum} \cdot 2 \cdot 2 \cdot R_{sum}$

$5 \cdot 5 \cdot 2 \cdot 2$ is 100 already! The symmetries $\frac{1}{2}$ will be interesting to evaluate!

Say we have just sum, sub, mul, div, cn, id, sa $\overset{\equiv \text{ same}}{}$ $sa(x) = X$.

7 functions. so $R_{cn} = R_{id} = R_{sum} = 7$ so $7^4 \approx 2500$

.20 $2500 \times 100 = \underline{250{,}000} = \frac{M}{4}$ already! If we didn't have sub & div its

.21 only $5^4 \cdot 10^2 = 25 \cdot 100^2 = \underline{62{,}500}$ only. A factor of 4, better

W.O. invoking symmetry} better. t 2 halves of t. sum. t. entire expression would be $\left(\frac{M}{4}\right)^2 = \frac{10^{12}}{16} \approx 6 \times 10^{10}$.

Reachable but bad. Since id & sum are sym, we get an additional factor of $2^4 = 16$ so $\frac{10^{12}}{256} = 4 \times 10^9$. Also, symm for final sum ⇒ 2×10^9.

If we did this not using "sub & div" as in .20 & .21, we get an additional factor of 4×4 so 2.5×10^8.

However while these symmetries ↑ pcost they don't influence kcost or bcost.

— which is t. real problem.

.30 If we could implement Random choice w.o. replacement, would get pc! Pseudo random trials would do this. Perhaps HASH coding would be more general.

← Not always! This present case Appears to be a special case in which Kcost >> bcost. — whether it is or not, t. main Q would be, how much CE to spend on that trial? Hvr, when this is not relevant, we can get ≈ rcost to control search! So: When is it not relevant?

288.05

0:25.00

Bolz

The Fletcher School of Law and Diplomacy
Tufts University

The Edward R. Murrow Center
of Public Diplomacy

Edmund A. Gullion, Dean

Medford, Massachusetts 02155

.05: 287.40 While 287.20 may be of great eventual importy I may want to return to main track: How to implement the symmetry of the 2 halves of 287.06 — & how to get more heuristics in, in general.

A diffnt track! Can parts of the function of .06 have ≠ compressions ≥ ½ ?

.10 More reasonable! Try to Implement The idea of finding which ~~en~~ functs work for each problem, Then "correlating" the functions w. the problem descns.

Suppose that we had found equns to 287.06 as solns. to many diffrnt probs:
I have indicated in Red on 287.06 the ~~constant~~ invariant & ~~constant~~ constant parts the eq(s). The variables are $\overset{7}{\alpha}, \overset{7}{+}, \overset{7}{\text{sum}}, \overset{7}{*}, \overset{7}{\text{mul}}, \overset{2}{n_1}, \overset{2}{n_2}$ | say 7 positions
rcosts!

So rcost = 50×50 × 28   2500 × 28 = 70000 = 70k

This would be the rcost of using the function 287.06 : we mult by the rcost of the function itself (which is probly hy).

.19 So at present. (.10) seems like a more reasonable approach!

.20 What we do, is first get the set of pairs: $\begin{Bmatrix} 3+7=? \\ \text{add}(3,7) \end{Bmatrix}$; $\begin{Bmatrix} 3*7=? \\ \text{mul}(3,7) \end{Bmatrix}$ ect.
So this becomes a. new Q.A. problem. To a human, it seems trivial, but what tools are needed/used ? "=?" are constant! 3,7 correspond to 3,7 (long random nos)

.25 + → add ; * → mul. But how to implement this? (.30)

.26 the ob "+ was seen" correlates w. the op "create sum" "

.27 .19-.26 may be an instance of a more General Ideas: perhaps: "Tools at various Partial solns: Mutation, Crossover (combine) them for total final Solns. Mutations Crossover" Should be part of normal L srch. They are usually "OSL" cases. "OSL" Cases. .29

.30: (.25) In (.20) we want to find what 3 is a funct of ; ~~&~~ what 7 is a funct of, what "add" is a funct of. Clearly 3→3 & 7→7 ; the mapping could be made by "correln".
Another way to look at the "functionality" In .20   3+7.., = 3*7 differ in only 1 symbol
ad 3,7 = mul 3,7 " " " " "

So the functionality must [redacted] be betw. those symbols.

.35 On "correlation" "+" "*" is a predictor of "add" "mul" w. "100%" accuracy so pc ≈ 1.
~~we look for~~ in .20, we ~~[illegible]~~ test various symbols in top row for efficiency in predicting symbols of bottom row. This is good because it works in "Grey" cases → (289.05)

0.26.00

Bdg.

.05 : 028.40! as well as "Black & White". This "correln" is a standard Heur. for detecting "Causality".

.06 One main Reason I'm not 100% happy w. 288.20–.25, .30–.40! I don't see it in a "Standard form" as comming from a real or [illegible] hypothetical TSQ.

But perhaps drop objection .06 for a while. Later, I ~~[illegible]~~ will (I hope!) have experience w. Discovery of heurs, & I will then have ideas about how to ~~[illegible]~~ deal w. .06.

.11 **So:** I've more or less gotten TM to do t. tsq $3 \begin{smallmatrix}+\\ \times\\ \div\end{smallmatrix} 4 = ?$. T. ~~[illegible]~~ cjs's are all very small.

first each $3 \begin{smallmatrix}\times\\ +\\ -\\ \div\end{smallmatrix} 4 =$ is learnd, individually, then 288.20 is usd, followed by t. "correln" Study of 288.35. This is all pretty much in "English" so t. details have yet to be worked out.

Next: to learn evaln. of arby alg. expressn. Best have all notations in Polish. This is simplest. Later, a prob. for TM will be xltn ~~[illegible]~~ betw. polish, ↔ Rev. Polish ↔ infix ↔ polish w. parens, etc.

~~Polish~~

Perhaps a fundemental Operation for TM, is substitution of expressn for a variable or expressn → express or expressn → variable. Another possy fund. operation is use of Push down Stack.

+ (3,4) → sum(3,4) ← it knows : to get [illegible]

$\frac{3}{7} =$

+ 3, * 8,9 → sum(3, mul.8,9). This is **not** very interesting! TM just does "+" → "sum", "*" → "mul", etc. ~~[illegible]~~ The stack is not explicitly usd.

perhaps have TM lrn infix → polish (← lang. of CPU.) Just try this out! See if any interesting things are lrnd. [Perhaps try Polish ↔ reverse Polish.]

It may be easier, more "consistent".

~~[illegible]~~ ~~[illegible]~~

3, 4, +	r polish	3 ~~*~~, +, 4	infix
sum, 3, 4	polish	sum, 3, 4	polish.

In general, t. ease of discovery of anything will depend critically on t. notation.

Also, I could try parens w. polish & r. polish. : Any of these variations don't make much difrence in learning + − * ÷ ~~[illegible]~~ individually. ~~So~~ look for difrences (if any) in more complicated combinations of + − * ÷

Def pol, rpol. T. problem now is evaln. of arby alg. expressn. I will try r. pol (= rev. polish) → polish notation.

3, 4, +	~~→~~ R pol	3, 4, +, 7 ~~mul~~ *	
sum, 3, 4	pol.	mul, sum 3, 4, 7.	

Write out some possl. "final soln(s)" that I'd like : see if I can lead TM to it (them)

.37 A desirable soln. : replace 3, 4 + by (3+4) obtaining 7, 7 *, & evaluate this.

Perhaps a recursive program! If $3, 4 \begin{smallmatrix}+\\ -\\ \times\\ \div\end{smallmatrix}$ occurs, rewrite expressn w. its evaln. Repeat until → 290.01)

0/30/00
Bugs

.01: 289.40 no longer possl. If result is number that's t. result.
The idea of "rewriting" & of "Recursion" could be primitive or "heavily hinted" concs ("heavily hinted" means taught w. many big hints — like examples of word problems.

SN Use of example ***traces*** of solns. mite be an easier way to do early training than giving simply probs & solns. I may use a Mixture: Use t. "Trace" example only for probs too hard for TM to lrn other way.

Actually, 209.37 isn't such an unlikely thing to try. Say TM knows about "substitution" (& Rewriting). This substitution always ↓ no. of elements by 2, so it does "move toward a soln."

209.37 works for any of 3 lang. xltn. problems! — It xforms prob. into a new prob. that seems simpler than t. original! The "OR" net. — which I considered to be a major "primitive" planning heuristic. → So This Looks Pretty Good! It is in English — so its still quite flexible

.15 In t. "OR" plan: "Any way I can "simplify" t. problem?" Well, in t. past, substitutions ~~term~~ (in t. form of evalns.) always ↓ t. length of an expressn — (One possl. "GPS" measure of vector hill ht.), so its a good thing to try. → see .35 for OBJECTION!

SN T. routine of 289.37 is not really "recursion". It is More like a "Do loop" with "until" or "while". As such, it has less cc than "normal Recursion". Most simple recursions can be easily represented as loops of this sort, & t. ~~a~~ pc of dcrn is identical to that of recursion, because it has t. same defining "parameters".

SN Solving Eqns of all kinds by progressive "simplifn" (.15) is a commonly
.24 used technique. The use of an invertable substitution is a "special case" (as in soln. of linear → quadratic → cubic & possibly quartic & possibly quintic, since in all cases, one adds a new kind of function that enables t. soln. to be found (for linear quad cubic, its ~~[illegible]~~ linear inversion $\sqrt{x}$ & $\sqrt[3]{x}$; for quartic & quintic, I don't know yet ← perhaps look at soln. of quintic ( )

.24

Substitution (& perhaps inversion of substitution) should be primitives in t. language or very easily defined from Primitives.

I want to Estimate pc of this "soln" of .37: Also estimate its cc, since I need to know $\frac{pc}{cc}$ (or $\frac{cc}{pc}$ = cjs)

.35 Using t. "OR" Heur of .15 Woops! in .15, TM doesn't yet know that "doing a legit substitution leave t. value of t. expressn. invariant." ← These are all concepts foreign to TM at this point. ~~[illegible]~~ Would it be feasible to teach TM these ideas before having it ~~aqui~~ aquire the heur of .15?

291.05

N.1.00

Bulg.

MEDFORD TUFTS UNIVERSITY MASSACHUSETTS 02155
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

.02 290.40! I m.to teach TM

About equality: that it is invariant under equal substitutions & other ~~[illegible]~~ unary, binary, etc. x fns: e.g. if a=b then f(a)=f(b) for any f, a, b.
Maybe not exactly true if F is Floating pt. function! Tho, TM works probilistically;
These relations are usually true in term (so 99% accuracy); later TM will try to find how to get closer to 100% accuracy by looking at error cases:

Perhaps it could lrn about t. "division by zero anamoly" this way!

First teach what equality "is": Examples, pos & neg.

SN pos & neg examples: ~~[illegible]~~ There are usually far fewer models that need be considered if neg. as well as positive examples are included ... ~~But still~~ One can have very A.H. models (w. + & − cases) but t. A.H. models look very A.H. — They have very small pc.

So 3=3 3=4(no): 3+4=7 ← This is rather complex! TM has to find ~~So~~ sum operator in its internal lang. An impt Q., perhaps is "In what order to introduce concs to TM?".

perhaps ~~first~~ teach "identity", then equality. Identity is ~~[illegible]~~ a relation betw. strings & strings or nos. & nos.

So: Is 3+7 a string or a no.? In Lisp, maybe, its a list. Its evaln is a number. So maybe identity should relate strings to strings only.

3+7 is not "identical" to 10 tho "3+7 equals 10" because evaln. is implied ~~[illegible]~~ "equals".
"identity" can be a ~~primative~~ primative conc (string α)=(string β) means (evaln of α)=(evaln of β)
I may want equality of 2 nos x & y to be primative: it ~~[illegible]~~ is tested by ~~[illegible]~~. evaluating x−y: if result is zero, then x=y. TM will lrn that if x=y then y=x.

TM will lrn. to evaluate t. function Solve( ):
.35 e.g. Solve (x, x+3) = −3

.36 eval ($\underbrace{3+7}_{\text{string}}$) = 10

.01 : 291.40 ! The present approach seems v.g.. Perhaps review recent work! Have I been "drifting about?".

Superficially, The "OR" plan of 290.15 is an "English" descn. of how to solve t. problem; Trouble is, it does seem to use several concs that seem distant from "primatives".

So it suggests 291.06 ff ~~[illegible]~~ i.e. To teach TM what "equality" means:

SN This somehow seems closely related to teaching TM what "quantity" meant — But seems more "exact"!

291.06 ff is beginning to sound like LISP! Distinction betw. strings & evaln. of strings, etc. Perhaps all of t. concerns ~~that~~ I had about what ~~[illegible]~~ I meant

.11 by various expressions: (e.g. distinguishing betw. evaln of a string & t. string itself) → .18 would be automatically taken care of by LISP.

T. slowness of LISP can be taken care of in various ways!

1) Lisp → Forth 2) People in GP (Genetic Pgmg) (comm.) ~~[illegible]~~ translated ~~[illegible]~~ Kozo's Lisp stuff into Machine Lang. ~~[illegible]~~ a/o CC, C++.

Perhaps have TM lrn Lisp (as a TSQ!): Another possy is to lrn MAPLE.

.18 : .11 Another thing I was confused about, that Lisp deals w. in a possibly o.k. way: t. idea of Binding temporarily: An assignment statement is only true, until t. next modifn. of t. Variable.

Also, t. idea of Local v.s. Global variables: In Basic, we use local variables in Srtns. All other variables are global.

Also: I have a file on "Functional Langs": It includes a long review Art. on "functional langs"

N 10.06

SN A possl. way to deal w. large (or ∞) cc of trials in GA. Use ≈ Lsrch: i.e. For each GA run, ~~select each~~ have a k value, & truncate a trial as soon as cc. pc > k. For t. next GA run, do k ← 2k.. Pick some reasonable value of k to start t. "Series of GA runs".

.30 SN On t. possy that t. Z141 d.f. of ~~[illegible]~~ (≈) Lisp functions, is a universal d.f. on all functions. To show this: Consider these functions:

F_1 ~~[illegible]~~ maps integers into Lisp functions: it is able to map all possl. functs from integers.

F_2 maps integers into integers (essentially what a UMC does): Pr ∼ giving a universal Discrete (stopping) D.F.

.34 $F_1(F_2(\text{integer}))$ maps integers into functs with t. universal D.f. on functs.

$F_1 F_2$ is a function realizable by LISP — it has $> \phi$ pc. (finite cost).

So if we give LISP random inputs, this D.F. ~~[illegible]~~ will ("Mult Majorize") ~~[illegible]~~ $F_2 F_1$ (G is any finitely derivable function $F_2(F_1(G))$)

.38 ~~functions~~ — so it, too, will be a universal D.f. on Functions. 263.06

Q: Is this a non-trivial result?

N 10.00
Bulg

SIGILLUM UNIVERSITATIS TUFTENSIS 1852 PAX ET LUX

.06:292.40 MEDFORD **TUFTS UNIVERSITY** MASSACHUSETTS 02155

THE FLETCHER SCHOOL OF LAW AND DIPLOMACY

Administered with the cooperation of Harvard University

Well, what is defn. of a "universal d.f. on functions?"

Say G is a D.F. on all functs G(F) = pc of F. ; G(F) has a finite defn.

If $G_U(F)$ is a Universal d.f. on functs, then $G_U(F) \geq k\,G(F)$ for all F, for some k > 0. (k is indep of F : is same for all F)

If 292.38 is true, How "picky" do I have to be in evaluating pc's from Z141? How bad is MDL?

Note: tho t. F all have finite defns, they need not be effectively enumerable. (?)

[redacted] Does **292.30ff** prove Universality for t. Z141 → Lisp d.f. on functions?

.16 **Two Previous Discussns of "Universality" of D.f. on Functions!**

282.04 −.06, 282.35 − 283.16 || 269.01 to at least 270.16 (270.18 − 271.02 maybe relevant)
271.03 − .40 seems to be very relevant.

.19 Back to 292.30 − .38 ; $F_1(F_2(\text{integers}))$ ← 292.34 maps integers to functs, & gives a "universal" d.f. on functions. ("Augmented Z141") is a way to map integers onto functions .... it is a possl. "F_1" The "functions" that F_1 describes, are from integers to integers. ..... → 302.16

271.20 switches to t. problem (perhaps more relevant!) of t. adequacy of 257.34 ... (which seems el. — tho 271.34−.40 perhaps makes it less so.)

.235 Since 257.34 is prbly what I will use, & it will guide t. intellectual path of TM, ~~28~~ 271.25 suggests that t. Universality of 257.34 (or of Augmented Z141) — w. low

.25 is not as important as; "will it give solns of probs that we & no it ~~e~~not cjs?". T. idea was that I would lead TM along a path that a human could follow, using human heurs'; that TM could (w a large tho possl CB) learn itself. —— After going along this path enuf, it would have enuf **heuristic** power to ~~contintue~~ continue w.o. much guidance on my part.

T. rôle of "Universality" of TM's model, in t. prgg. is unclear.

One simple (tho possibly impt.) way! When TM searches for a soln, (that is a function) it really is able to try all possl. functs. That t. "soln" is beyond its CB, is a

.38 **necy** restriction (debility) of all "Real" mechanisms.

This Makes t. TSQ more impt. than Universality Even w. a non-universal lang, an adequate TSQ can bring us close enuf to solve a problem — But universality w.o. a good T.S.Q — one can (in practice) solve very little. Hvr. even w. a V.G. TSQ., we need a lang. that is at least strong enuf to express all t. concs. we need. Probly, it mite as well be Universal. → 304.01

Perhaps a very impt idea is **284.22 ff** ← (Approaches) Novel way of TSQ design. 294.01

MEDFORD
TUFTS UNIVERSITY
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University
MASSACHUSETTS 02155

.01:

T. Main goal of TM is "compressn." Ideally, this also makes the future work of compressn easier. One part that would make it easier is if "significant" ("Significant" (not "significantly")) Longer codes are included. These are codes one would normally "Backtrack" to in Theory Revision.

Would "indexing" t. corpus be of use here? Perhaps list all things one mite do to make t. tasks of t. future, easier.

At least ~~[illegible]~~ 4 kinds of future tasks: ① Augmentation of Corpus — we have to code t. new stuff. Prediction of ~~[illegible]~~ t. future is perhaps best way to do this (also see ④)

② Modifn. of tasks/goals of system ("~~[illegible]~~ S.W. Maintenance"). If system is maximally modularized (so each part of system mite also have alternative S.W. codes ... (≡ ways of doing it)), this makes Modifn. easier.

③ System should have many "adjustable params" to deal w. changes of applicn. of system.

④ Theory Revision: when t. predn. system of ① no longer seems to work adequately. ~~[illegible]~~.

H19.00

Marcus!

MEDFORD TUFTS UNIVERSITY MASSACHUSETTS 02155
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

This is a reply to Marcus Hutter (and ~~partly~~ to part of Jurgen Schmidhuber)'s "Learning how to lrn".

Consider t. problem to be RTM, w. h not specified but CB specified. Each ~~[illegible]~~ "trial" by TM is somewhat expensive, — so cc of the response by t. usr. will, in general, be computed differently from TM's internal compn.

"horizon" = no of trials to be used.

As first approxn, consider RTM w. h = 1. This is a more or less "Straight OZ problem". For h = 2 t. problem is to make 2 trials ∋ t. best of t. 2 is max. This allows "experiments" (which h = 1 does not allow). (I think Jurgen was concerned w. h = 1, only).

If TM has intelligence ≥ that of user, this can be a very dangerous activity! TM will try to manipulate t. USR into giving it a high score..

For h > 1, I think t. problem may be a "Dynamic Prgmg" problem. There are also aspects of the (unsolved) G = G(t, x) problem, in which G decreases w. delay.

11·22·00 Hutter

297 ~~2~~

MEDFORD

TUFTS UNIVERSITY
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

MASSACHUSETTS 02155

.06: 1.40 :

Int. [redacted] Problem of 1.18-.30 At each pt in t. process, we have a sequence of Painting, Goodness pairs: $[pn_i, G_i]$ $i=1|n$.

.09 → We could regard it as a simple OZ problem; We want a painting $pn_j \rightarrow G_j$ is max, & we want to find it within time T.

We would have to use a rather sophisticated ~~OZ~~ OT if we wanted anything of much value in

.12 a reasonable time. → (Note 4.12!). An "OT" is any string that derbs ~~gives~~ a discovery strategy.

T. Method of 1.18-.30 is perhaps a greedy method, in which each choice of painting to present, is t. "Best possi." that TM can do at that time.

More General (& Less Greedy) OT's allow for "Experiments".

By allowing t. most General kind of OT's (Those describable to a UMC) we get t. best possi soln by Lsrch.

.20 So .09-.12 really is a "Theoretical adequate" soln. of t. problem!

I was thinking that the "Cure Cancer" -(research direction) problem, was close to the "Paint on accord. to good painting" problem.

In t. painting problem, feedback to each "Trial" is a scalar.

.24 (It could be more info, ~~like~~ Like ~~[illegible]~~ a more detaild discn. by t. "patron"

.25 of just why he liked/disliked t. particular "Trial".)

.26 Anyway, in "Cure Cancer" t. feed back is in a very general form.

Supposedly, if we Generalize t. idea of O.T., it could deal w. any ~~[illegible]~~ possi. form of feedback; so it could deal w. t. feed back of .24-.25 ~~[illegible]~~ and even t. very general type of F.B. in .26.

Finding a Good OT ~~[illegible]~~ of this more general kind is t. same kind of problem (w. what looks like an identical type of soln) as my older, simpler OT's w. only scalar F.B.

1) Most ~~[illegible]~~ probs are solved by OZ or OT. T. invention of a good FF for a INU prob to convert it to a OZ prob, is an impt skill that TM would acquire early.

.33 Furthermore GPS with vector Goal, could be implemented by this more General OT.

My impression of t. OT's used in "Lsrch": That one would best use only 1 OT — ever! Using >1 O.T. means loss of info betw. O.T's — so ~~[illegible]~~ OT's can't use info that other OT's have requested — They may even duplicate trials

3.06

MEDFORD TUFTS UNIVERSITY MASSACHUSETTS 02155
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

.06:2.40 Made by other OT's.

If t. trials are very Expensive, this would be very inefficient!

Selecting t. [redacted] "Most promising" OT thus far considered by TM could be quite good — if t. OT selection was like I envisioned it in MCT.

In MCT, we have this "Grand PD" that looks at a problem (≡ t. "conditions") & outputs a pd. on all possl. soln. trials. In t. older style, if t. input were a OZ problem,

.12 it would output a pd on all OT's wrt. that problem — A possibility is that as TM

.13 matured, it would end up w. almost all wt. in one P.D. — Seems likely!

[I was concerned that in t. "Cure Cancer" problem of 2.20, t. results of experiments would be "problistic" — But in fact, all results of any action are "problistic" as far as TM is concerned!]

SN It would be well to integrate logical/mathrical reasoning into t. "Lsrch" or t. MCT — since much development of OT's involves reasoning of this sort.

I did write some about this in t. last 6 mo. ... try to find it!

But do go into t. "Cure Cancer" problem in some more detail! I think I did Not solve it, last time around.

→ (SN) perhaps t. ideas on non-Lsrch ~~[illegible]~~ suggested by .12 → .13 would be t. best way to deal w. t. $G = G(x,T)$ type of OZ problem!

→ SN for "Any Time" problems, use Levin's Time-shared Version of Lsrch for OZ probs

In General, if G(·) is expensive, ~~[illegible]~~ Normal Lsrch may be "rather poor".

In "Cure Cancer" prob & other OZ probs w. expensive G or c! ① cc of G or c may be large & ~~[illegible]~~ constant & known by TM. ② cc of G may be known in advance for each trial — by a formula $f(\overset{\text{trial elem}}{x})$ ~~[illegible]~~ Given to TM. ③ ~~[illegible]~~ cc of G or c is only known by TM as its own induced p.D. from previous observation of G, cc pairs.

→ This last seems close to t. "Chess" problem. In Classic ALP, one can't ever get a d.f. for t. cc of a trial — but in Resource Bounded ALP, one might ~~[illegible]~~ legitimately make an approx. guess.

If "Cure Cancer" is an ≈ INV problem, (so one knows when it's been done — or a certain threshold of goodness of soln is assumed) then t. ideas of t. First Gambling house thm are relevant — one wants to minz E of Σ cc before soln. is found.

4.06

.06: 3.40 : In General, t. Methods of MCT can be usd. w.o. using Lsrch. Tho, in many cases, Lsrch is ≈ OPTIMUM.

W.R.T. OZ problems, Lsrch seems ~~[illegible]~~ O.K. for induction problems. — either Sequential or unordered — say w. fixd corpus. Lsrch is v.g. for finding t. unnormed PC of a corpus. } Note.

.12 T. "Method of 2.09–.12" automaticly tries to "simulate" t. Expensive Gorc & it does this in a non-el. way.

Normally in induction problems, one does not use Lsrch (in t. usual sense) for trying cand. codes. ~~[illegible]~~ (in which any trial is or is not successful)

Hvr. induction is an OZ problem: for such OZ problem, TM finds a P.D. over appropriate OT's! → see 302.27

Usually we arrange so that every trial results in an acceptable code: we do this by computing t. amount of "correcting" that would be needed to xform a "non-code" in to a fitting code. More Generally, since induction is an OZ problem, TM recognizes it as a "special kind" of OZ problem (all OZ probs are "special" since t. dern of t. problem determines what ~~[illegible]~~ kind(s) of OT's will be tried), and uses special OT's to solve it.

Hvr, there is a common type of induction problem in which we do use lsrch: These are t. induction problems that I had in early Algebra learning, — ~~[illegible]~~ like ANL.

~~[illegible]~~ We don't use Lsrch for diflt INV probs (we usually reformulate them as OZ probs, w. suitable FR .... or as GPS probs w. vector Gorc (2.33), ...

→ Yet, INV probs should be ≈ optimumly solvable by Lsrch: "If all needed heurs can be put into t. "Grand PD"." □

→ For OZ probs, if they used Lsrch & "all heur into weaving "Grand Pd GPD"" then it would, indeed,

.30 be ≈ optimum! — for same Reasons as for INV probs. → 301.15

I want to look at Several heurs: to what extent can they be expressed as Modifus of t. "Grand PD."?

Take t. ~~[illegible]~~ AND/OR Net, "Plan" heuristic

Remember: In most INV probs, we're searching not for just a soln, but for a strategy that will solve t. problem. Lsrch is over these "strategies".

In t. "cure Cancer" problem: In view of past info, what is strategy w. max proby of solving problem? — one w. min expected CC of solving problem? ← First G.H. Theorem. First G H thrm.

→ A Heuristic would be part of t. dern of any strategy. This idea of a "Strategy" is one that 5.06

N23.00 Hitter

300~~5~~

N.B. .18-.40 should
Be in Bulg file
Perhaps make Hitter
pc 1 thru 5, → Bulg 296 thru 300.
Bulg 295 .13 on Hitter.

MEDFORD

TUFTS UNIVERSITY
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

MASSACHUSETTS 02155

.06: 4.40! To Mm. Lrost. — but can

.07 it deal w. "Quick Abort"? [Actually Quick Abort & perhaps heurs like it, couldn't ↑ TM speed by very much! The expected CS is always ≈ CCS/PCS = CJS. → 301.11

Re: Use of Heurs: Ordinarily, only 1 or 2 heurs will apply in a given case; so ~~not~~ not much of an "exponential explosion": Once a heur is chosen, it could closely guide t. soln. of t. problem (not much ↓ in pc).

.11 GPS is ~~are~~ and AND/OR ~~planning~~ are fairly General Heurs, applicable to almost all problems. GPS for INV. problems: AND/or Nets for (I think) both OZ & INV.

.13 Another Very General Heur ~~[illegible]~~ "Use analogy to successful ~~unknown~~ soln. of ~~a~~ known problem"

Using heurs in this manner ("when a problem occurs, find assoc. heurs to guide soln") Makes it sound like AM, Eurisco etc — Lenat — But it certainly seems much different in spirit from Lenat! — Certainly no "Teaching by Brain Surgery")

I guess a big Diffnce: My TSQ's are much less "Skinnerian": Much larger CJS than Lenat's. — Also, they are always problistic.

.18 SN In/recent retrospect: One of the ~~most important~~ thorny problems that I recently ran into was teaching TM Definitions of various concs. My mind was very vague as to what relationship was betw. many concs. & examples of that conc.

.21 I had hoped that my Reading about details of "LISP" would help here. → (.24)

.22 T. 3 Heurs of .11-.13 are really quite impt Heurs! Usd almost always (GPS & And or nets) "Analogy" of successful soln. used maybe less.

.24 (.21) In .18-.21: I think my ~~&~~ own internal ideas of what concs. meant, were too vague: E.g. "t. idea of "quantity". — Yet vague as they may be, concs of this sort can be very impt. in lrng! I think t. problem was that I did not recognize that t. conc. I had in mind was really quite vague!

So, T. Q is: how do I implement this vague conc? — More Generally how are vague concs. conveyed ("implemented") to TM?

T. eval/quote notation of LISP is probly very useful for ~~the~~ describing concs. — but still vague concs will occur, & I need them in that form — just how to deal w. this & when to deal w. it.

Wrt. explicating vague concs: Perhaps just give myself lots of examples & give a no. (maybe pc) giving how much t. conc. is represented by ~~[illegible]~~ each example.

Thinking Vaguely is an impt. method of induction/genzn.

Perhaps "Fuzzy Reasoning" is ≈ what I want.

301.06

~~[illegible]~~

N^{25}[redacted].00

Bulg.

30[redacted] ~~6~~

MEDFORD

TUFTS UNIVERSITY
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

MASSACHUSETTS 02155

:06 :5.40

The heurs like 5.11-.13 [redacted] do seem expressable as modifns of t. pd. What Ray says is ∴ :
"This heur has a pc of .3 of success"; (T. heur itself will be a sequence of instructions (some of them often problistic) on how to solve t. problem) ← T. form is expressable as a modfn. of t. rand. GPd

.11 :300 :07 : [ON Quick Abort] : Perhaps it would be possl. for TM to discover t. "Quickabort" heur
.12 by "Logical Reasoning", rather than [usual methods of heur dscvry]. I haven't really gone into t. rôle of Logical reasoning in problem solving & heur invention.

Lsrch OZ probs.
.15 : 299.30 : (Lsrch for OZ probs..., & optimality of) : T. idea in OZ probs is to use Lsrch over t. P.D. of OT's.
So, say a person, α (non TM) has a better way to work a particular OZ problem than TM.
Then α must have a ~~way~~ reason to chose a particular good OT for that problem, & that reason must be based on past experience &/o Logical reasoning. [If those reasons were built into TM's GPD, it should be able to get about t. same results.
.20

The argt. on .15-.20 would seem to hold for "practically any" OZ problem Type:
— Including those w. "very expensive evaln. functs"

So t. weak pt. (if any) of .15-.20 (re: optimality of Lsrch for INV. probs as well) is t. "Logical reasoning" part: (also in (.12)). "Logical reasoning" is a special case of problistic reasoning [redacted] — but w. pc's close to 0 & 1.

.26 So t. Q is: How can TM lrn. to reason "Logically"?

Well, suppose we first teach TM Math & Formal logic in a purely Theoretical way, so it can solve problems but knows no relation betw. ⟨math & Logic⟩ & other problems that it is given. We then give TM problems, β, in which its Math/logic knowledge is relevant, & it could find there is an "analogy" betw. math/logic solns, & t. solns. of problems, β.

It mite also be possl. to teach TM "Practical Applicns." of math/logic along w. t. Theo. understanding of Math/logic. "Practical Applicns" are w.r.t. TM's own normal problems.

→ Try to find a "practical problem" for TM, in which math/logic would be useful.

Well: TM$_2$'s work. — initially, TM$_2$ is mainly Lsrch, but it could be made aware of t. more general goal of min E(cc) for total soln. or for "T. horizon Time".

→ An early approach I had to ANL did use Logical reasoning!: I didn't know how to do it, tho. → 305.06

N28-00

Buls
8

MEDFORD TUFTS UNIVERSITY MASSACHUSETTS 02155
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

.06: On Summarzing Machines & Prediction.

Say One has a corpus, C, & one has found "several pems" that are not bad for this corpus. Predictions can be made in 2 ways:

.10 1) Wtd mean of t. several pems.

"THEORY REVISION":

.11 2) Devising new pems: These use cons usdin t. "several pems", w. appropriate wts.

Hvr., T. Mechanics of .11 is not clear: It may be that .10 is almost always used — unless it seems to not work anymore; In which case one goes to ".11" in "Theory Revision" mode ← .35

to not work anymore: In which case one goes to ".11" in "Theory Revision" mode ← pilot.

D3.00: Z 141 using a ~~[illegible]~~ complete lang. (like lisp, say) generates a Universal P.D. on strings.

.16 One (Apparently) big Q is: Can one generate a Universal d.f. on functions this way?? → .23

Some recent writings (~~[illegible]~~ within last 10 or 20 pp of Buls) suggest that this is not very ~~[illegible]~~ important — that what is more important is a good TSQ. Even if t. D.f. on functs was Universal, it could assign a very small pc to t. "correct" function. A good TSQ assigns hy PC to t. ~~soln~~ known Soln., even if t. D.f. on functs is not Universal. (SEE 293.16 for Biblio. of refs on "Universality")

see → 293.25, 292.13, 271.25

.23 (.16) If we had a nice way to map strings into functions, that would do it! — (Say we need functs of single strings only). We might be able to ~~[illegible]~~ do this very easily in "Lisp" by restricting t. set of [redacted] symbols in t. Alphabet that generates "Lisp expressions". (Note 293.19 on this)

.25

.16-.25 May have a very Simple Soln.! I'll just have to think about it in more exact Detail!

.27 Anyway: In MCT, I was thinking of t. 2 kinds of induction problems (Seq vs. Bag induction) being "special kinds of problems" — T. other 2 kinds are OZ & Inv. probs. In fact, Induction probs are treated by MCT as "Just another [redacted] OZ problem": Th. GPD looks at t. induction problem, then because it is an induction problem, it uses special OT's for it. T. nature of t. OT will depend further, on whether its seq. exmpl. or ~~[illegible]~~ Bag induction, & whether t. params are continuous a/o discrete.... Also t. source of th. problem will help determine t. OT.

O.T.

Well, "Theory Revision" may be a standard kind of OZ problem: but in induction, if OSL is possl., th. OZ version of t. problem becomes more complex (?)

303.06

D·6·00

Bulp
ē
TSQ's

MEDFORD

TUFTS UNIVERSITY
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

MASSACHUSETTS 02155

.06 : For TM to learn

(1) good set of OT's ~~[illegible]~~ (2) when each is to be applied to a given Problem.

(1) → Getting set of good OT's: If I insert a set of OT's that I think are good: I should factor them as much as possl., into a compact code. This makes their form to be more likely one learned ("by itself") by TM.

(2) The assignment of an OT to a problem can be made on basis of (a) source of problem [Does it come from a Math, physics or Chem" Domain? i.e. is it a Math or a physics or Chem problem] (b) Type of problem; Is it an induction problem? Is it continuous or discrete? (c) "Logical Reasoning" (301.26)

What are Main Problems (Bottlenecks) in TM rite now?

.19 1) TSQ design.... (see .26 for a perhaps novel approach).

2) Clear understanding of what T. Lrng Algm. is! 257.34 (Augmented by 21.41) ... ē its non-el version 271.34–.40. 271.25 ~~suggestion~~ suggests that 257.34's "universality" is not as impt. as "will it enable low cjs solns of probs?" ← (see 293.25 ff).

.26 3) Look at 284.22 ff. for an approach to TSQ design. --- see .35 also

4) T. mechanics of putting OSL into all induction systems. (Probly ~~a~~ an IR system is needed)

5) How TM should operate so that its future tasks will be easier (294.01).

6) How TM can use logical reasoning! How TM learns to do Logical reasoning: what TSQ to use?

.30 See 301.26 ff. One new idea: Simply have TM do logical reasoning. Then figure out how TM could have

.31 lrnd this Technique. → 305.06

.35 : .20 TSQ TRACK (not nearly Now): Take bunch of Math probs. Solvable by humans. Write down how I seem to Solve them. — Then how TM can ~~[illegible]~~ ~~be then~~ lrn to solve them — At first, not nearly using "known concs", but eventually reformat it into this form, ē devise a TSQ for them. I "sort of" did this for linear, Quad, cubic eqs. — deriv of Solns. was at a "relatively hy level (English)". This seems V.G. How does it differ from 284.22? I think 284.22 ff is more into t. details. — But in fact, both .35 ē 284.22 ff can (ē should) be used. I think .35 is more general. ~~2~~ 304.28

D.8.00 Bulg. "MAIN LINE"

↓

.01: (293.235–.38)R : "Expressing a [redacted] Conc" means that we can "define it" so it becomes a single word, w. [redacted] an associated pc. This enables us to "compress" by using the Conc.

284.22 is a simpler way to design TSQ's. It's not as complete as a conc. not, but closer to a TSQ. → [See Also 303.35 ff]

.05 So, in view of (293.235R)ff, all I need is a good TSQ, & a lang. adequate to express its needed Concs. "Z141" along w. the "LISP" will perhaps be adequate for obtaining pc's of concs.

So how does t. elaborate treatment of OZ problems in MCT fit into .05 ff?

Well; MCT is about GPD updating. How is .05 related to GPD updating? — Presumably, GPD updating is just another OZ problem. Like Most OZ problems, it has special techniques for doing it.

→ SN Like OZ probs, Each INV prob. can have (a) particularly appropriate way(s) to solve it. GPD looks at an INV prob. & gives a PD over Soln techniques for that particular INV problem.

I had assumed that t GPD would simply give a pd for t Lsrch of ~~[illegible]~~ any INV prob. presented to it: (Probably) not so! As I noted in t past — Most INV probs are solved as OZ (or GPS) probs, not via t Lsrch.

→ It looks like I am gradually Abandoning Lsrch as a Universal prime problem Solving Method!

— Tho in many circumstances, it does seem to be optimum.

Also, it may be that many ostensibly non-Lsrch methods can be (usefully) expressed as modifn. of t pd. & (usually but not always) amenable to Lsrch. (There is, of course T. "Quick Abort" problem!)

.24: 303.40 Consider all sorts of probs, at all levels: e.g. "Cure Cancer", Devise Good unified theory for physics"; Devise V.G. painting from scored set of paintings (do it for poems, music, etc.). Make big list of interesting problems that I'd like TM to be able to solve. Chess, Checkers. (Good evaln. functs; Cheap, effective)

.29 On Prob. Slvng. in General! First time one solves a particular kind of problem, it's pure [redacted] Lsrch. $\frac{cc}{pc}$ depends on t. pc of t. primitive concs used to derive soln.; Only "Simple" probs (simple wrt t primitives) are practically solvable.

Next, one solves probs by "OSL" find problem in past ~ ~~[illegible]~~ cont to present problem & use " ~ " soln. ~~[illegible]~~ techniques.

Next, when one has 2 or more examples of a problem type, it is easier to decide on pts. of similarity of t. prob. & soln. so cc of Lsrch for soln is much reduced.

It is possl. to solve probs w.o. t "OSL" phase: The first & second times one solves t. problem it is by pure Lsch in both cases. After t. 2nd soln., one recognizes it is ~ to {prob. & soln.} of previous problem. By eliminating OSL as possl. soln. method, one is probably at a very big disadvantage, but it is still

.37 possl. to do TSQ's & acquire "big intelligence" ← (perhaps) —

In t. SAARB work on TSQ's, I ran into t "SCALING" problem: → SPEC 306.01 → 305.06

D 8.00 Bulg

305

Logical / Mathematical Reasoning:

How TM can do this,

" " " learn to do this.

301.12
303.31
.06 : 301.40

MEDFORD

TUFTS UNIVERSITY
THE FLETCHER SCHOOL OF LAW AND DIPLOMACY
Administered with the cooperation of Harvard University

MASSACHUSETTS 02155

(See 303.30 - .31)

.08 Some areas where TM uses Math to do induction:

1) Linear Regressn: we want a set of coefs to minz. Σ sq. error — an OZ problem.

In General, each OZ prob may have its own Soln. mechanism. Consider a normal analytical OZ problem: To find the max of $\sin^2 x + 3x^2$. People lrn how to do this. We could teach TM to find ~~a~~ extremum this way. It would first have to know diff. calculus. (which is one way: Probly Archemedes could have worked it w.o. dif. calc!)

2) There has been much work done on Continuous optzn: "Info Based Complexity": Joe Traub et al. Also, on discrete optzn. (to some extent). TM could be "taught" all of this.

[redacted] D·10·00

Bufo.

.01: 30~~4~~.40 spec : I put ~~all~~ solns of all problems in "Memory" w.o. indexing them, so pc of memory use → $\frac{1}{N}$ (N = no. of probs solvd) — A clearly non-viable method of prob. soln. ← Int. SAARB + tsq's,

This mite not be too bad if one accessd memory only once in t. soln. of t. problem: But for 2 or 3 ~~acc.~~ memory accesses, t. process becomes ~~quickly~~ quickly of too low pc.

Some simple methods of indexing: ① Recency of occurance ② Source of problem (Math, Physics, Chem, etc). ③ Similarity of Structure of problem defenition to that of previously solvd problem ("Analogy"). ~~[illegible]~~ (③ is perhaps Most impt.)

So, Given a problem: Subproblem is: What is/are ~ previously solvd

.10 problems? (or previous ~~un~~ un solvd probs?!) ← To avoid unsuccessful techneques of t. past.

.11: 304.37 : ~~On~~ Actually, how much of a trouble is OSL? (≈ φ!) —

~~At~~ W.O. OSL, when a new problem comes in, we look for "induction systems" that have been obtaind w. SSZ ≥ 2. This does involve some searching. ← An "ind. syst." is a problem soln. that's common to ≥ 2 problems that are in t. corpus

Hvr, t. searching is easier than in OSL, for 2 reasons: ① There are many more "Solvd problems" to search over than "induction systems" < perhaps not quite so! ~~When we search~~ ~~we~~ normally search over t. total of solvd probs that have not been put into induction systems + induction systems: t. cardinality of these sets is usually << t. set of all solvd probs; & perhaps not ~~much~~ much larger ~~[illegible]~~ ~~[illegible]~~ than t. set of all "induction systems"

② In an induction system, the relevant features of t. problems have been "distilld out" (= isolated) — This makes it much easier to tell if it is relevant to t. new problem. On t. other hand, Comparing an unsolved problem to a solvd

.24 problem — it may be difflt to decide on t. relevant pts. of similarity.

On t. Basis of (.11 – .24), it is not always clear that one mite save any cc at all (~~or~~ comparatively even) by not ~~doing~~ doing OSL.

.29: : Re 304.29 ff: (on Prob. Solving ~~[illegible]~~ mainly by ~ "Case Based Reasoning")

Often we will not find previous problems or "induction systems" that will solve t. present problem or be relevant to it. Pure lsrch is not often t. way to go.

We'd like (at least) to be able to ~~[illegible]~~ do lsrch over combns of concs larger than primatives. — But t. system of 304.29ff only puts prob solns. & "Induction Systems" in memory (i.e. "defines" them). There is no Mechanism [redacted] to obtain intermediate sized ~~[illegible]~~ concs: (only ~~[illegible]~~ primative concs & "Induction Systems")

One (not completely A.H.) method: Factor the "Induction Systems" to ~~[illegible]~~ ↑ the pc of t. entire set. The "factors" are sub concs of all sizes.

This "factoring" is part of a fairly normal part of prob. solving: i.e. compressing t. all-over-pgm found thus far. ~~[illegible]~~ We then do Lsrch over the parts (= concs) to solve problems. It may be ~~[illegible]~~ that our ~~[illegible]~~ using 307.01

D·11·00 Bulg.

.01:306.40 Solns of previous problems (could ~~would~~) be a special case of Lsrch, — So would have an integrated system.

I have been considering 2 kinds of General Prob. Solving Methods:

.05 T. Older Method: All probs solvd by Lsrch: T. Pd for each Lsrch is a cond'l. P.D.; the "condition" being t. problem defn. The unproved assumption is that all heurs are expressable as modulus of t. P.D. ("G. rank P.D.") (Its known that "Quicksort" is not included) A (perhaps) associated assumption is that this probabilistic form of th. heurs can always be obtaind from t. Data that t. "Human form" heur was obtaind frm.
— This "Data" includes (manual logical reasoning)

.11 T. newer method: 304.29 – .40, 306.01 – .10; .29 – .40: Idea is that new problems are always either solvd by raw Lsch or by finding probs. (or sets of probs) that they are ∼ to, & using ∼ soln. methods.

T. new method seems closer to t. way I solve probs, but I'm not sure about its "completeness". Could I do a TSQ from Elementary Algebra, using this Method?

D·12·00

.15 Part of t. "older" approach was to try to do TSQ ∼ to that of Humans — doing, say, Elementary Algebra. How this fits into t. older Model of .05, is not immediately Clear! It seems to be like t. newer method of .11. — Perhaps I can integrate .05 & .11 !

One idea is that I would do .15: That .05 would enable me to express all ~~two~~ prob soln. techniques & Heurs: Then perhaps .11 would be a major heuristic.

A Main idea of (older) .11: That whenever I (or anyone else) solves a problem, it is by putting together concs. that describe t. soln (method)., so a/sub "adequate" problem becomes t. acquisition of t. subconcs — usually by solving simpler problems.

A (recent) idea on writing TSQ's: → 303.35 Amass a large set of problems, w. one or more "solns" (= method of soln = "trace"). From this, see if one can make a Conc. net & order t. problems suitably.

303.35 & 284.22 are on this ≃

Another (perhaps related idea): To write out how I solve probs on a very low

.30 level (of English). E.g. "How to evaluate Alg. Expressns": & Find a part that I can evaluate. Evaluate it & substitute it, obtaining a new expressn. Recurse to α until I have a pure no. or until I can't find any sub expressn. to evaluate.

I could write a Lisp pgm to do this: T. Lisp pgm would be "factored" into concs. that would be lrnable from other problems.

As a Heur for above; A subsn. of an evaln. "Simplifies" t. expressn — thereby bringing it closer to goal. This would be an "Adequate" heur to solve t. problem. It is an example of an "OR" decomposition of a problem. (AND/OR nets)

Re "Simplicity" a shorter expressn is usually "Simpler", Hvr. in t. present case,

308.01
309.01
Spec

D.12.00 Bulg.

The Horizon problem: Rather unclear as to how to define ~~t.~~ final goals. Say one wants to ~~get~~ (mean) yield per unit time to be max over "long time". This is equiv. to max total yield for "long time." So one would do meta problems first. (always first) — So one would never do ~~[illegible]~~ non-meta problems.

One way to deal w. this is to spend say 50% on Meta probs., 50% on non-meta probs. Also, it may be nec'y to put time scale on Meta probs: how far into future is it oriented? So at least 2 params needed ① t. ≈50% ② t. time scale.

An automatic time scale mite be found in t. inability to predict t. nature of future problems to more than (say) 10 yrs into t. future. (This 10 yrs would ↑ as ~~[illegible]~~ IQ of TM ↑ beyond that of Sci Community)

So, anyway, since I will be ~~giving~~ assigning TM lots of params, I have no qualms about giving it these params. People who want their machines to be "truely autonomous" don't like this, but ~~I~~ I certainly don't want a "Truly autonomous machine"!

D.14.00
Bufs.

.01:307.40 ~~[illegible]~~ getting monotonicly shorter "equivalent probs" would usually be adequate to solve t. prob.

One big Complaint about t. SAARB ANL soln. was that it was not t. way a human would solve t. problem, so it was hard to ~~find the~~ identify useful concs in t. soln. In 307.30 ff we are close to t. way a Human could solve it. We Do seem to need t. concept of "evaluation" — which may be a softly ~~[illegible]~~ impt. idea so we can make it ≈ "primitive".

One Big direction of ~~[illegible]~~ TSQ work was to start at t. Beginnings & work up to a very
.10 Very Smart Machine. A Perhaps Better Approach would simply work on "patches"
of TM's training at various levels, then "patch" t. patches together.
Working on Patches of various levels will give me a better idea of what t. real
problems of TSQ design & t. TM ~~[illegible]~~ Learning Algm. are.

.10 certainly isn't a new idea, but I haven't really been doing much in that direction. An extreme case of .10 ff would be to take an existant "Expert System" (such as a Symbolic integration pgm in Maple or Macsyma or ...) and ~~[illegible]~~ slowly convert it to a lrng system. — Try to find out how each of its routines could have been "discovered".

8.12.00 Bulg

164.40

.01: **ON MUTATIONS, & CROSSOVERS (Recombination)** : as SSZ = 1, SSZ = 2 resp.

On mutations as SSZ = 1 (L = SSZ + 1 = 2). This is fine. We can use any regy we can think of that ~~are~~ (seem to be) appropriate to the "Domain" & as a basis for generating t. mutated form of t. parent.

In t. Case of Recomb. SSZ = 2, (L = ~~[illegible]~~ SSZ + 1 = 3), we are in about t. same situation as SSZ = 1; we are much controlled by regys observed in t. General Domain of t. problem. For each such domain, we must learn how best to design recomb. techniques.

For Recomb, t. main idea that ALP contributes is that child should have as many common parts of parents as possl. (but having part ~~[illegible]~~ from one parent is also good (a Mutation))

As Befor, my impressn. is that in certain domains, a well designed mutation scheme can be much better than an inappropriate Recomb. algm.

One could evaluate a given Mut. Algm using t. "present population". We could get its mean & varc. in predicted G of child.

Similarly we could Evaluate a recomb. Alg. (for 2, 3, 4 or any no. of "parents").

.17 **A META** Problem: To devise new (Mut/Recomb) Algms. & select best ones (A GA-type problem). What language what Mut, Recombs. to use for any "Meta problem"?

Also, t. Goal for t. (Mut/recomb) Algm isn't so obvious! — we want an ~~i~~ M/R algm that Maximizes "rate of ↑ of ~~[illegible]~~ Gmax".

↳ An alternative View: To get a good, cheap pd out of t. existant Corpus of Cands. ($\equiv p(X, G)$). — This is in t. Direction of ~~a~~ "TM_2"

Another view of .17, would be that we mutate or recombine ~~[illegible]~~ a ~~p~~ (small?) population of (Mut/recomb) algms. (eventually, this scheme might become "Recursive" so this mode of mut/rec. ~~[illegible]~~ could also be (mut/rec)ed.

Re: Human Creativity: An experienced Scientist will solve difft problems rapidly: his pd's for trial solns are narrow, but often correct. — His searches are short. Often, hvr. they will be not interesting solns.

The novice Scientist will have a less narrow search. ~~[illegible]~~ algm. It will be much slower in finding a soln to a problem, but because of diversity of trials his solns to problems are less ordinary, and he may solve problems unreachable by t. "experienced Scientist" a/o he may find useful unexpected things along t. way.

A very Successful experienced Scientist will have a somewhat diffrnt direction of search than most. Also, if he spends lots of time, he will eventually go to strange unusual places and find great discoveries.

Very poor Scientists & Engrs: ~~[illegible]~~ use Excessively Elite srch. → 186.01

8.12.00 Rule:

Mutations, Recomb, Search, "Creativity".

.01: 185.40: Also, poor ~~re~~researchers do not use good search methods: Many quit after trying only ~~the~~ high PC trials. They don't properly consider hypoths in Lsrch order. e.g. ~~—~~ 2 cnc. w. pc's of .1 & .01 should be given no ~~[illegible]~~ less wt. than 3 cncs of wts .1, .1, & .1. This latter is ~~[illegible]~~ spuriously Elitist in spirit (consider only hi pc. ideas — Several hi pc ideas - rather than one low PC or a mix of hi & low.).

A merit of Much of Koza's work may be that he has (because of not so good prob search methods) a very broad, very diverse set of cands that he tries. ~~E~~ It takes him a long time, but he is able to solve very dif'lt probs. in "Creative" ways

I have this (Mild) fear that doing a search in Lsrch order may be "Non-creative" due to low diversity. — But I think that on t. average Lsearch (using all available info) will get me the best solns. in ~~available~~ available time. If we give it enuf time, it will be "creative" & ~~[illegible]~~ (on t. average) more so than non-Lsrch based searches

8.21.00 Bulg.

160ff ~~necess~~ Review Review
outline of Fund TM — 163
~~Princeton~~ Princeton/166
Genido / 76.21
~~mem~~ McPeak. 172 .11,34

IPC of Human!

Gary Kasparov

The IPC of (≈) Deep Blue, t. machine that beat Human world chess champ. Said to be ~ 10^{12} flops. very vague in terms of bits/sec, hvr.

Got some idea of speed by [redacted] no. of Board evalns/sec. ← which may give.

Hvr, no. of "flops" ("Bits") in a "Bd. evaln" is very unclear; Well from ratio, I can get no. of flops in a bd. evaln.

→ Perhaps read up on just how ~~De~~ Deep Blue actually worked

The speed of "Blue Gene" is to be 10^{15} flops (or 2×10^{15} flops)

~~Eo~~ Check on this by other comparisons, other estimates of speed of Blue Gene.

"Blue Gene" speed is also given ~~as~~ as multiple of Deep Blue. —

But much of this can be vague & inconsi.

What Human Computing does well ① fast parallel optical & perhaps acoustical processing. ~ 10^{11} ~~[illegible]~~ bits/sec.

② Fast ∥ info retrieval. This could be, by far, the most massive amount of computation done. The amt. of HW (dendrites) used, could be ~~e~~ Enormous.

→ This last could be impt. for OSL: to find situation in past, ~ to present problem or "relevant" to it. "Associational" lrning: Perhaps done by ≈ Zatocoding.

So large "Content Addressable Memys" would be a very cheap way to get what appears to be "Enormous IPC"! — Hvr, at present time there is not much market for "CAM", so it (perhaps) not cheaper than what is now, t. conventional way to do information retrieval.

The fast optical & acoustical (& maybe touch — tho this can be slower) need not be used in a TM of great intelligence.

Latest civilian Super Computer cost/performance. \$ 45M for 6 T flops:

\$ 7.5 ~~m~~ μ /flop = \$ 7.5 per M flop, \$ 7.5k per Gflop seems hy.!

An upper bd on human IPC: cost per flip ~~[illegible]~~ in ergs is En. Time ≈ h

so $\boxed{En = \frac{h}{t}}$; $\frac{1}{t} = \frac{En}{h}$

$\frac{1}{t}$ is bits/sec for 1 bit. ; ~~power~~ for N bits w. $\frac{1}{t}$ sec. each: $\frac{NE}{h}$ Joules used.

$\frac{E}{h}$ ~~[illegible]~~ $E = \frac{h}{t}$; power = $\frac{E}{t} = \frac{h}{t^2}$; $\frac{1}{t^2} = \frac{E}{ht}$ = power for $\frac{1}{t}$ bits/sec.

so $\frac{1}{t}$ = bits/sec = $\sqrt{\frac{E}{ht}} = \sqrt{\frac{watt}{h}} = \frac{\sqrt{watts}}{\sqrt{6.626\cdot10^{-34}}} = \sqrt{watt}\cdot 3.885\times10^{16}$ Bits/sec

$h = 6.626\times10^{-34}$ Joules ~~[illegible]~~ · sec

~~Eo~~ $\sqrt{watts}\cdot 3.885\times 16$ bits/sec. ~~for~~ 25 watts (human brain)

this is only 1.942×10^{17} bits/sec. ~ 2×10^{17} bits/sec.

How much of t. 25 watts is used for computing, is unclear. ✓

There may be a bug in this reasoning, hvr. ; ∥ proc gr. w. bits faster ∝ √power may give ~ φ power for much ∥ procg!

Also, t. Q of how much power (if any) is needed for ~~[illegible]~~ Memory access

8.31.00 Bulg:

EBL

A possl. model ~~for~~ EBL: A problem ~~eat~~ is given in a particular domain, in which
.02 one has much "knowledge", both domain specific & logical/skill.
There is enuf to solve (or almost solve) t. problem deductively, using that (.02)
"knowledge": but there is a fair amt. of cc involved in th particular process of deduction.
So: Next time a "similar" problem occurs (presumably TM has
good enuf, relevant enuf categorizing concs to realize ~~~~ this particular
relevant "similarity") it reduces t. cc of deduction considerably by
patterning after the previous case. This is a bit like CBR,
(≈ OSL)

Bslg.

$\boxed{G(x)\cdot f(t)\ \text{: "t super OZ problem".}}$

on the "Super OZ-problem": $G(x)$, $F(t)$: G is Gorc for candidate x, ~~[illegible]~~ $F(t)$ is monotonic ↓ funct of time: Both G ä F are known to TM. find x in time t ∋ $G(x)\cdot F(t)$ is Max.

[More *generally* Given $G(x,T)$ w. $G(x,T_1) \geq G(x,T_2)$ if $T_1 < T_2$.]

Use technique of MCT to deal w. this:

TM has many cases of $\boxed{OT_i(G_j(\overset{x}{\bullet},\overset{t}{\bullet}),\ G_{jk},\ T_{jk})}$ ↓omm

.07 $OT_i(G_j(\cdot,\cdot), G_{ijk})$ ← each ij will usually have several (often many) "k" trials giving diff. x_{ijk}, T_{ijk} ä G_{jk} values. .09

OT_i is t. "R optzn. technique. This is true, tho each OT_i will have a single stopping criterion that may or may not be much.

From data on many OT_i's ä G_j's (TM's Life experience)

.11 TM is able to obtain a condl. p.d. giving $\text{Proby}(OT_i, G_j(\cdot,\cdot), G)$

It is the p.d. of OT_i giving t. Gorc G for problem $G_j(\cdot,\cdot)$.

.13 From this and (.11) cond. p.d., we obtain $\text{Proby}(OT_i, G_j(\cdot,\cdot))$ which is t. proby that OT_i will have t. best (over all OT's) Gorc, for problem $(G_j(\cdot,\cdot))$.

.16 NO! Using .13, one does an Lsearch over all OT's. {The usual Q comes up. "When does one stop" — Each OT_i tells one (as part of its defn. "when to stop")

.19 .16 is *Wrong!* One simply picks the single OT_i that is most likely to

.20 give max G for problem $G_j(\cdot,\cdot)$. The data used for t. PD of .11 and .13 (which is directly derived from .11 w.o. any new data) can include (& probly will) include info for several Trial T_{ijk}'s for each OT_i (.07–.09 ä ((.07–.09)R)) The *Data* is (.07–.09) The Resultant PD's are .11 ä .13.

.26 O.K. Now consider regular OZ problems. I was not (I believe) ever able to show that Lsearch was (practically) near optimum for OZ probs (tho it's seemed to be for INV problems). — Could it be that for Normal OZ problems as well, ~~[illegible]~~ (i.e. $F(T) =$ constant) The soln. of .19–.20 is best? i.e. just pick t. ~~Best~~ OT_i that is most likely to be best & stick with it.

In both the $G(x)\cdot T(x)$ ä t. classical OZ problem; As we work on the problem, t. pd's of .11 ä .13 may change, but we cannot usefully use that info *directly*. What we want to know is the pd over the OT_i's involved in a *jump* to those OT_i's. In t. case of many OT_i's this involves a "startup cost" — one can often use data from trials thus far on an OT. for info on ~~[illegible]~~.

.37 prospective new OT_i's. → 214.05

→ In much of t. forgg: One must spend *time* computing probabilities that various OT_i's will be best. There is a nice tradeoff betw (214.01

8.31.00 Bulg.

.01: 2.13.40 That time & t. time usd to make t. TRIALS themselves. In t. Forgo discussion, I have assumed that cost of a trial is >> usual cost of ~~pron~~ making pc estimates. We note, hvr, that t. cc spent on pc estimates can be arbly large: it's not clear how ~~much~~ much cc to spend on them!

"On use of Forth" for Cand Geng.

.05: 2.13.37 In general, for even "regular" OZ probs. It would be best to start out w. t. OT that is most likely to be "Best". ~~Work on it until~~ stay w. that OT unless/untill t. data on cands tried thus far (or any other info), makes it clear that (including cost of switching & startup costs), ~~it is worth~~ The Best OT is no longer OT₁ but OT₂ — so we switch to OT₂, Then to OT3 or maybe back to OT₁ — depending on How things look.

The Success of this method will depend much on how Good one's PD is. —— but t. same can be said about normal Lsch for OZ or INV probs

,14 A Big Advantage of ~~picking a~~ (in Normal OZ probs) Of picking t. "Best" OT & staying with it untill it no longer looks Best!

.16 If There are several OT's all about equally Good — all "at top" of t. Goodness Range —— Normal Lsrch will spend about t. same amount of time on each of them, rather than just pick one and work on it (usually to t. end) — This last is usually t. most efficient way.

One serious disadvantage of this last approach is that we will get much less data on "non-top" OT's, so we will tend to not find anything Better! It is an "Elitist self conforming Hypoth"

Is this Criticism also true for the G(x)·F(T) Genr?

9.1.00 Bulg.

(1) $\frac{pc}{cc}$ vs $\frac{2^{-k\,cost}}{cc}$ &

(2) Failures in Many || ~~[illegible]~~ procs.

(3) MONTE CARLO TM (MCTM)

In 1999, world production of 32 bit cpu's for computers was ~ ~~[illegible]~~ 100M units. M units at 10^9 bits/sec is 10^{15} bits/sec! ≅ ICP of "Blue Gene". 10^{15} may be > human, so 100M units may be ~ cap of sci community — so "over threshold". In 10 yrs, ICP → ×100.

One problem w. || machines of this number, is failure of individual μproc [IBM's Blue Gene] is supposed to have a way of dealing with this.

Another way might be Monte Carlo progms. In 1990 (or 1991) I did a study of Mt Carlo choice of cands to try. The prob'y of trying a particular cand$_i$ was $\propto$ its $pc_i^{\,\epsilon-1}$ (or was it $(1-pc_i)^{\epsilon-1}$?) Anyway, the goal at that pt. was to get the effective use of ~~pc~~ $\frac{pc}{cc}$ as ordering of trials rather than $\frac{2^{-k\,cost}}{cc}$.

If it worked ☺, it might also be a good way of dealing w/ occasional failure of individual μprocs.

In general, my work/interest in Monte Carlo methods of search has been motivated by the possible use of "Crummy Components".

9 . ·00
Bulg.

Correction of Errors in Languages/(from Grammars): Several "Takes" on this:

.02 1) ~~Recent~~ Stuff I wrote Wolff, & my corrections of it. — I had 2 ~~3~~ somewhat difrnt "solns". — Hvr, .06 ff is the nicer way to do this!

2) ~~Meth~~ Method of correction used in ~~the~~ linear/n.l. regression; the model gives a prediction & its free we have a P.d. for ~~[illegible]~~ "correction".

.06 ~~3)~~ This method mite be ~~[illegible]~~ adapted to language probm, use ideas in my version of of S78T3 for Languages! — So that for languages, each bit in the corpus is given a Pd.

.09 In fact, .06 is ~~[illegible]~~ a v.g. way, since we get from the error theorem, an expected ~~error for Bain~~ ~~[illegible]~~ RMS error (& ~~[illegible]~~ L K kulbkr? error) for the predns.

Gram Err Thm.

G E thrm.

Call the analog of S78T3 for unordered finite strings "The Grammar Error Thrm." v.s. S78T3 "The Time Series Error Thrm" or "Regression Error Theorem" So us "T S error thrm" = S78T3 or its corolary.

TSerr Thm.

TSE thrm

Hvr, "Symbolic Regression" ~~usually refers~~ may refer to (& usually does) unordered ~~[illegible]~~ finite strings.

Hvr, I have to work out more details of .06: Just how its applied to all sorts of langs. — I think the corrections are to be applied to each bit of the corpus as it is generated. This Area Needs More Work!

9.3.00 Bulk

ANN: Genzd Hill Climbing.

In hill climbing, "Momentum" is often used to guess at direction (& size) of next step: i.e. Say $\vec{x}$ is t. config. vector, & we are trying to find $\vec{x} \ni G(\vec{x})$ is max. ($G \equiv$ Gene).

"Momentum" $(\vec{x}_t - \vec{x}_{t-1})\cdot(G(\vec{x}_t) - G(x_{t-1})) = \vec{m}_t$.

Our next trial, $\vec{x}_{t+1}$ would then be $\vec{x}_{t+1} = \vec{x}_t + k\cdot\vec{m}_t$, where k is a factor to be inserted by USR.

Actually, k depends on our estimate of t. second derivative of G w.r.t. $\vec{x}$, & we can estimate it empirically from past (perhaps x windo) data.

.10 say $\Delta G = m\Delta x + \frac{1}{2}s(\Delta x)^2$ ("s" ≡ second derivative)

we want $\Delta x \ni \Delta G$ = max $m + s\Delta x = 0$ so $\Delta x = -\frac{m}{s}$

We find m & s by doing a curve fit over the last r set of data

.13. pts. (or use an x windo of width, r).

We can only do .10 – .13 once, hvr, since it keeps $\vec{x}$ on a simple st. line in $\vec{x}$ space. After we do .10 – .13 once, we might do a trial in a random direction. — Hvr, size of step is unclear. Size of step could be related to t. "Noise level," which we could have obtained in t. previous .10 – .13 calculations.

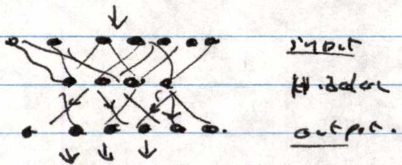
233.01

9.12.00 Bu'g

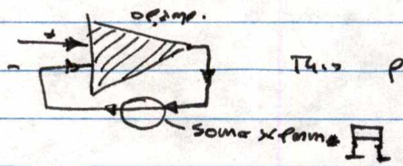
ANN, Fractals, Fractal Compression, CFG discovery

07:221.40 Discussn. of Simon (Grad Stud. w. J. Pollac at Brandeis.)

3 layer ANN



Output is fed back to Hidden layer: This is analogous to



This performs inverse of xfmn. A.

Say "A" is a Parsing Algm. — Say its a Matrix Parser w. Real (not Boolian) wts. — Suits for a stochastic CFG (or CDG)

The params of A a/o t. wts in t. ANN are adjusted using standard ANN a/o GA rules.

I Guess that we want to adjust wts. of ANN & params of A so that t. ~~inputs~~ outputs of t. ANN is ≈ ~~[illegible]~~ its input.

Doing Matrix Mult. w. Reals, is much more expensive than using Boolian values! So one mite use stochastic ~~[illegible]~~ Boolian values in t. Matrices! — This would then be like IFS (Iterated Functional Systems). Instead of a probty value of P_i in a matrix element, T. element would be 0 or 1; ("1" w. probty P_i, hvr.)

Consider a CFG:

$\Sigma \xrightarrow{p_1} a\Sigma$ $\quad C \xrightarrow{r_1} Cd$

$\xrightarrow{p_2} a$ $\quad \xrightarrow{r_2} \Sigma$

$\xrightarrow{p_3} bC$

Here, we have 2 N.T.'s: Σ & C: They Correspond to Matrices. The p_1, p_2, p_3 correspond to wtd. addition rules on Matrices. There are 3 terminals: a, b, d.

We could start w. a very large no. of possl. production rules for each N.T. So Σ would have 10 proby values assoc. w. it. — As we continue t. process, hvr, t. we ~~[illegible]~~ discover that only a few proby values are ≫0.

We mite start w. t. Grammar

additional rules

$\Sigma \rightarrow$ dbdamr } These are t. elements of t. Corpus! $\quad C \rightarrow a\Sigma$

$\rightarrow$ edbdrm } This is t. A.H. Grammar. $\quad \rightarrow \Sigma a$

$\rightarrow$ } We start w. it & mutate it toward a $\quad \rightarrow bz$

v.g. "Goodness of fit" criterion. $\quad \rightarrow \Sigma b$