N4.03
3 TM

ABC Defgh

CMN : Continental Mass Mat?

.00 : 396.40: Since a PST is a rather general pgm, this "view" enables more than just modifs of t. PD₁ : It could include "pure ~~cc~~ cc savers" like "Quick Abort".

So perhaps my present position is that, for IND, PD₁ is t. final Goal: Read each PD₂ is a ~~[illegible]~~ "search heur." — That thus would work for each of t. items of 391.05 - ~~[illegible]~~.40.

.07 We can start w. a simplified form of "Phase 2" For each problem coming in, TM tries to categorize it in terms of which PST is best for it. At first, perhaps there are only a few PST's & only a few types of problems. TM can try various

.0 PST's on various problem types & obtain a matrix giving mean performance (≡ Goal: (presumably (maximized)) for each problem type, PST pair.

The categorizing functions (Like or ≡ t. R functs) will be initially trained by indices given by trainer — later, Irng of ~~[illegible]~~ these R's will be done by TM, w.o. indices — or infrequent indices.

[An index can be regarded as a kind of "Hint".]

Note; I think the "R functs" are being used in rather difrnt ways from My previous use of them in t. early section(s) on QA lrng.

Also think about "Soft" R functs for both applicns. T. "matrix" of .40 is a component of a "Soft" (S-funcs) R methodology.

.21 OK — Say t. R has chosen a particular "PST" (of t. 391.05–40 types): What next? How are these PST's implemented? Well, t. "R" system for QA is implemented as ~~[illegible]~~ in t. early section(s) of t. "Report". Any of t. other techniques of 391.05–.40 can be individually implemented — but how to MIX them? Depends much on what techniques are being "Mixd"!

≈ (9:45 a.m.)

Ideally, for IND: T. input Recognizer decides what PST(s) to use to Solve it. This is done in Both Phase 1 & Phase 2, but in Phase 1, t. operations are much simplified because

.56 [illegible] of really is Phase 2; Its ≈ an early TM model (at IDSIA) in which I had a function that would look at a problem & decide how to treat it (perhaps take a look at the older work).

Hvr, what I want in Phase 1, is a kind of "Minimal" "system that can ~~do~~ do ~~[illegible]~~ useful induction: perhaps helpd w. broad "Hints" by trainer.

Would 396.20 approach be good at this pt.? — I do want as minimal a system as possbl. Just try to apply each of t. sreil methods of Sec 396.04–.11 to t. Algebra TSQ.

A "Study Problem" for each technique of ~~396.04–.11~~ 396.04–.11, try to devise a training Sequence

PP 292—396
see previous pages for more detail

cop. in Page 11 item 26 ← bad typo fixed

10 Perhaps spend much time in Costa Rica working on a clear exposition of what the problem is; what the soln. is, & how its relevant to TSQ construction & soln. in "Phase 1"!

It does seem to be one of the most impt Q's in understanding Phase 1. Also it enables us to understand the equivalences of various forms of PD_2: Such as:

1) The 391.09–12 idea: an apparently rather general form of "Context":
How best to use past pairings of $QA_{1\cdots j}$ to associated O^j, for various j, to suggest a O^{n+1} for $QA_{1\cdots n}$.

2) The Recognition function forms of O^i.

3) ~~[illegible]~~ The use of definitions, token frequencies of O^n to give a pd for O^{n+1}.

4) The General idea of a SUMAC: This may be merely a different way of looking at all of these "equivalent" PD_2's.

I Want to show how each of the effects of 391.05–.40 work in all of the PD_2 models.
391.05–.40 plus 394.22–395.40 seem to cover about all of the ideas; but while I do seem to understand it now: I'm not sure those ~2½ pp. would be enuf to explain it to a "Me" of 6 mo. from now — much less one 2 yrs in future!

A nice approach: Consider a simple TSQ, from Algebra, say. Consider different methods of finding short codes (≡ methods of ~~[illegible]~~ finding Regularities) for this problem: How do they fit into the ideas of .04–.11 (or 391.05–.40) — or other general "Pd2" ideas?

23 To what extent is there a real separation betw. Pd_1 (the apriori) & Pd_2 — the ~~[illegible]~~ Pd that Guides Lsrch? Both are ~~[illegible]~~ conditional pd's on type of a (soln. program string) as conditional on the (problem descn). Presumably, after a large corpus; PD_1 would be somewhat like a PD_2.

.23 this is disturbing! It puts us back to asking what PD_1 is the ~~[illegible]~~ pd of! — which puts us to Phase 2!

But I also think of Pd_1 as ≈ AZ: A fixed; [illegible] Unconditional pd for induction

[NB] A recent ~~[illegible]~~ (Re) begining of the "2 Pd's" dfty: ~366.06 ff

Maybe soln. of 2 PD's problem. ~~3~~ ~~3~~ 394.22 − 395.03
(395.01−.02 in particular) 395.03−.40 is also quite important!

Same on HP 500 | 256M Ram.) 3433: win 98 Second edn. 192M Ram 4.10.2222A

00 Could find "solns" fast (in t. sense of satisfying constraints) but not necc'ly of high "up tite"

.01 pc (≡ hy PD$_1$). ((Getting "solns" that "work" is always possl. by an A. H. code or a

.02 "Promiscuous" ~~every thing~~ "code — both of low pc, but easy to find ~~&~~ fast)) → e.g. see (.18−.22)

.03 Perhaps .01−.02 clarifies t. problem enuf so ~~that~~ it is more or less solved!

.01−.02 considerations may enable understanding of: [t. probs summarized on 391.05−.40]

① Use of "R" (foreign) functs: Constrains form of conds, but can lead to finding "fitting" codes faster (tho not necc'ly shortest).

② ~~Ward~~ In trials to QA$_{1\ldots n+1}$ corpus, we also find that O_i worked w. QA$_{1\ldots i}$ (i ≤ n).

10 Part of this is in form of ~~rescan~~ defns of new tokens & modified pc's of tokens that are used in trials for t. entire QA$_{1\ldots n+1}$ corpus. I'm not sure I really realized this in my most recent discussion.... that those "token defns," & pc modifns were (partial) ~~pc modifns were~~ summaries of t. "previous corpus". ABCDE ABCDE abcde abcde □

So actually All of t. techniques of 391.05−.40 involving t. "2 ~~any~~ PD's", are also approaches to SUMAC. Conversely, SUMAC is just another way of saying that we want to be able to use t. fact that we have codes for certain parts of t. corpus, & we'd like to use those codes to help code t. rest of t. corpus

.18 $\boxed{SN}$: (.01−.02) codes for QA induction ① T. A.H. code: It makes a table of all Q → A pairs. If a Q has ~~[illegible]~~ k difrnt A's; Then each of t. A's gets pc = $\frac{1}{k}$.

20 ② T. "promiscuous code" for all Q's t. d.f. on t. output A's is t. universal D.f., or, if we want a more easily constructed D.f. Say λ is t. mean no of bits in t. A's: Then an A that is b bits long is

.22 given pc ≈ $\left(\frac{1}{2}-\frac{1}{2}\frac{1}{\lambda}\right)^b \cdot \frac{1}{\lambda}$ or some similar function.

All methods of Sumac or "using t past" can get stuck if finite CB is used: One way out is always "Backtracking" to progressively greater depths. ← ~~in all~~ finite CB systs — So The Need Backtracking or some other Technique to "get that work done".

~~So~~ Usually, assoc. w. each "Sumac" there will be bias introduced by reordering of trials. ~~It is~~ Is it usually diffi (or impossl) to guess at t. Magnitude or direction of t. bias (??) — IS IT?

30 I ~~[illegible]~~ kind of feel that I understand t. problem(s) of 391.05−.40 & the "2 PD's" problem(s). An attempt to explain: (.01−.02) was t. break thru: All of t. methods of PDs are reorderings of trials meant to get a soln quicker (a code that "fits" for inv. or one that "fits" for induction. Note t. introduction: 394.22 ff explains how t. conds (which have PD's) for induction ~~[illegible]~~ can be ~~[illegible]~~ treated as attempts to solve INV problems.

~~What should they~~ This Does seem like a very impt. problem in Phase I .... Perhaps t. most impt. problem. It seems to integrate various areas of search. So I do want to remember & have a record of just how this works!

03103
6.28.03
3PM

for __9AM.__

.00 : Report Revisions Abstract.

1) ~~Section~~ ⑧ on update algm: Add to ⑧: In early phases of Learning we use a ~~simple~~ relatively simple update algorithm ~~[illegible]~~. ~~In a later phases, we shift to a [illegible] more complex~~ an algorithm that's ~~[illegible]~~ better ~~able to exploit regularities in the training sequence~~ ~~is able to exploit more complex regularities in the data.~~

~~When~~ Later ~~[illegible]~~ the accumulation of data enables us to ~~shift to~~ use a more complex ~~[illegible]~~ update algorithm that ~~is better able to exploit regularities in the data~~. ~~able~~ to recognize ~~more complex regularities~~ a broader range of regularities.

.08

10 There are two Extreme approaches to designing an intelligent machine.

In CYC (ref), Lenat gives the machine much factual information in a very direct manner. The system's ~~[illegible]~~ learning capability can be used to interpolate ~~[illegible]~~ — to fill in the knowledge it was not explicitly given. Some

In ~~our present~~ our system, the amount of information directly inserted into machine is ~~minimized~~ kept as small as we can afford. Almost all of its knowledge is obtained by inductive inferences from the sequence of problems given to ~~[illegible]~~ it by the trainer.

If ~~CYC is~~ successful, CYC ~~it~~ will have a large, encyclopedic knowledge base ~~from~~ which it will use to solve problems. On the other hand, if successful, our system will

20 be very good at discovering new knowledge, based on fewer facts and more ~~[illegible]~~ on inductive

.21 integration of ~~a smaller number of facts~~ these facts.

.22 081.03 On the "2 Pd's" problem. I had first of the PC as being in 2 parts: the function's the evaln of ~~function~~ PC of corpus. For many corpi not ~~the~~ true: for d-Predn: aps of corpus written = 0 or 1 only. ~~[illegible]~~ ~~[illegible]~~ for s-induction, using 310, we have ~~about~~ effectively same situation — we look for "short codes" — (in nootbits). For induction models w. both continuous & discrete params, I found (recently) a way to get about same effect using "Monte Carlo search".

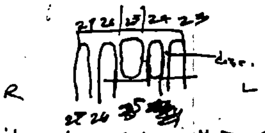
30 If the corpus has many individuals than can be individually evaluated, we can do "sampling" of corpus to reduce srch time: Also look for ~~[illegible]~~ "difft" cases in corpus for testing (this would prob'ly not work for SM "outliers".)

.36 Anyway, a possible approach is that the ~~[illegible]~~ aux info used in the srch could be either ① __logic part of a prog.__ or ② __useful for ~~[illegible]~~ search only__.

→395

① causes no theoretical diffys. ② hvr, __does__. A search containing ②-type info

3TM.



00 [: Perhaps a most impt. idea to include in "T. Report" (or at least My version of it), is
386.10 - .27 : The Idea of "Lenat's Expert System," & "a Pure Induction Machine" as being
.02 2 extreme [redacted] ways of obtaining an Smart Machine.

Also, recently parts of abstract of use of WOW makes incorrect.

.03: 392.40 Re: t problems of PD_1 v.s. PD_2 (391.05-.40):
Consider t R function "Solution": As desc'd in t "Report" "Recommendations" in t 5 update techniques for merging R's & inventing new ones: Synthesis & Analysis resp. — I think
— So we can grow & consolidate (areas of interest/parts of science/ ...).

10 : (space 392.40) This is an impt Q.: One aspect of it! If we stay in t "same R" can we simply directly work on improving t. assoc P.? — Or is each R a subuniverse of unbounded Complexity
— So ~~we~~ we treat each R as a separate (Ro linked by common ans) universe — to be solved by creating "sub R's" — (& sub sub R's ... ect)?

Another BIG Q! Say we are looking for a New P for t. corpus of a certain R.
How do we do trials that recognize t. fact that certain O_n's worked w. most of t corpus?
→ This puts us back at t beginning of t. problem of 391.05 ff, .09 & impartiality.
.19 One Common method, is t use t. tokens & their pc's (including t newly invented Tokens of t. last successful O_n.)
20 also, implement OSL ~~[illegible]~~ which is → idea something like considering O_n & large sub trees of O_n as
.21 "sort of" tokens.

[0.28.03] So far, several Big Methodologies for Phase 1

Newish → ① Mix of Expert System & Mach Lrng. ~~[illegible]~~ (.00-.02)
② (.19-.21) Use of pc's of Tokens & definitions of New Tokens in ~~[illegible]~~ O_n (including OSL considerations)
③ Use of Recognition Functs (see early sections of Report on QA Lrng).
④ "Gen'zd Context." I don't know how this fits into 1, 2, 3 — whether (2/or 2 or 3 are a [redacted] part of 4.

30 ⑤ "Quick Abort" by pe heuristics (My impression is that this may be d. only "Other" type of Gaur. T. main type is Models of approach to Lsrch. Perhaps (probly) this doesn't include Lrng during Lsrch.
⑥ see methods of 391.05-.40.

0.25.03

∠ ABc   ABC ABCDE
AF BC~~ABCDEFG~~

3TM

5 — depth of context.
7 → Backtracking

.00 A rewrite of 391.05–.40: Reordering: 6, 2, 4, ①

1 hard · 3 Refine & succ. · 5 31 by · 1 2 pd's

6 hard problem
- ② uses of solns to earlier probs
- ⑤ depth of contexts 4 scaling 1 .. i dont 2 pd's
- ③ R functs & ce (see stuff written on R functs)
- ⑦ backtrack.

T. ideas of Mixing E.S. & M.L. in constructing TSQ's for TM. Dfrnt roles, ~~etc~~. Much E.S. is too fast lrng, but poor Genzn. Much M L gives good Genzn but slow lrng.

Maybe mention in my comparisons to Lenat.

10 SN — If O_n works for $QA_{1/n}$ Then a "neighborhood" of O_n is a set of O's ⊃ [redacted] t. [redacted] pc they give to the $QA_{1/n}$ corpus is within Δ of that of O_n.

Hvr, that kind of "neighborhood" is too difft to delimit/construct.

We want O_n to be in a form ⊃ we can easily make cands that ~~will~~ have hy proby of working well on $QA_{1/n}$, (or "most" of $QA_{1/n}$).

.18 0.26.03 In "Phase 1" there are 2 sources of info: ① apriori (PD_1) ② history of trials (.19R)

.19 [redacted] $[(QA;O)_i]_{i=1/n}$: I have been considering ② to be "external context" because it is not part of PD_1 || Hvr. ② doesn't seem to be really "statistical info"!

Because t. srch was not done in "strict pc order" or "at random", t. results will perhaps be "BIASED". How to characterize this "BIAS" is unclear.

.23 Whether (.18R) is "legit statistical info": T. Q is: to what extent does it work? This mite be addressable as a statistical problem.

SN While I had considered Gound/Mut/cross as a way to get cands. for O_{n+1}, I really haven't (proposed/found/suggested) any good mut/cross's — OTHER THAN Recogn. functs. → Whoops! → see 393.19! → (for a common way) →

T. recogn. functs will (eventually) have to be (fuzzy"/gray/probilistic/s functs):

30 At first, t. R functs are ~~single functs of~~ directly defined by t. "Indices" that are furnished by trainer.

Later TM has to induce them, since t. trainer doesn't always give those indices.

So t. initial TSQ's will ~~be~~ each ~~be~~ for a single "index type" i.e. they will be for just one recogn. funct. — So no recog. funct is used, at first.

Tho if [redacted] one had 10 different indices, one could have 10 dfrnt P functs, & these P functs could share cones. → (spec 393.03)

.38 One ~~@~~ Impt Q is: As one ↑ corpus of TM, do we have to ↑ no. of R's, & keep t. no. of cones in each R constant (or have upper bnd for no. of them). → (393.10 spec.)

3 TM

.00 : of new [low freq] words as ssz ↑?: (In smaller ssz the new words had cose counts=0.) My impressn was that the pc↓ was too rapid (as a functof ssz) for this to be t. complete explann, but it mite be a component that reduces t. size of t. residual effect.

ABC

ABCDEFG
HIJklmnoff

.04

.05 : 390.30 So list impt. ideas:

. 1) T. "Two pd's" ⓐ pd₁ (AZ aprip) ⓑ pd₂ pd involving "context" $\begin{pmatrix} QA_1 \cdots QA_n \\ O_1 \cdots O_n \end{pmatrix}$. | 375.04 refs.

| Note 390.31

.09 . 2) 387.22-.24 : That one goal in our srch for O_{n+1} is to use t. facts' that

10 $(O_i \text{ works for } QA_1 \cdots_i)_{i=1 \cdots n}$ — That we would like (in some sense) O_{n+1} to be a "modifn/augmentn" of O_n that retains all of O_n's old predn capabilities & in addition does Q_{n+1}, A_{n+1}. T. idea is to reduce t. srch for O_{n+1}.

.12

. 3) (390.11 389.20) An impt. part of (2)(.09) is that certain ways to accomplish this (e.g. t. Recognition functs) result in much reduction of cc in testing conds

. 4) An early use of context was that it was necy to deal w. "scaling"..... that the uncondl pc' for tokens got too small (as corpus size ↑ ... or more exactly, as no of concs. in lang ↑ (see 390.35 391.04 for discussn).

20

. 5) Definition of "context" (390.21, .25, .31-.34): for a "token", if t. pc is "unconditional" there is no "context" used. Any "conditions" on t. pc of a token are part of t. "context": Context can be "internal" (part of original AZ (= pd₁)) or "External" (all other contexts)

390.21,.25,.35 ✓

. 6) T. present problem seems to be a major bottleneck in deciding how to do "Phase I". Designing ~~a~~ lang. TSQ's.

. 7) T. form of Pd₂ (see .09(.22)) should facilitate "Backtracking" when necy.

30 ✓ (Note that "Backtracking" is necy part of any SUMAC (system w. CB<∞!) See 375.07 for a genl method to do backtracking. .... Also NOTE ⑧ (.32)

.32 8) There is the SUMAC ("Summary Machine") w. backtrack idea, that is closely related to all of this. ~~Summary~~ SUMAC w CB=∞ is ideal by itself; Hvr. SUMAC w CB<∞ needs "Backtracking" (at least) to enable it to continue a corpus. ~~Perhaps back~~ Backtracking to an adequate depth (maybe (AT WORST) down to zero corpus length) which means starting AB INITIO.) will eventually work, but could be VERY expensive.

9) T. "QuickAbout" type of srch Heurs Don't seem to involve 2 kinds of PD's — but they mite be part of PD₂. : They are an impt. kind of "Phase I" heur, hm.

0.23.03

3TM

P_0 4×10^5yr
P_U 8×10^7yr

6.4 × 7 = 45 7×10^8y U235
4.5×10^9 yrs U238

.00 My present impressn is that t. 2 PD's (PD₁ PD₂) — PD₂ was t. result of an attempt to solve t. problem of $\boxed{387.22-.24}$ (Note 388.02)

Now it seems that PD₂ was (at least — perhaps mainly) an attempt to solve 387.22–.24 .....

But ~~does~~ does it have any other value(s)?

.05 .. Stated a somewhat non-std way; we want t. PD of O^{n+1}, in view of ~~[illegible]~~ (condition)

$(\{Q_j,A_j\},O_j)_{i=1/n}$, and $\{Q_{n+1},A_{n+1}\}$ ~~[illegible] may not want to include Q_{n+1} in t. conditional.~~

.07 The PD₁ info should also be available.

I think .05–.07 may be identical w. my general "Context" soln. from some time ago —

.10 (a yr. ago?). → ≡ 388.20

.11 A sort of aside: when we use Recognition functs (as one kind of approxn for .05–.07)

We have a very IMPT serious second effect i.e. usually we only have to test a small part of t. corpus on t. new cands. This may also be tied up w. t. soln ~~to~~ t. "augmentation" of O_n needing to be much smaller (hyperc) than a srch for an O_n that must satisfy ~~several or~~ many or all of t. $(Q,A)_i$.

My impressn is that this business of not having to test much of t. Corpus, is a very impt part of t. Human Updating process

Re: t. "R functs": when we feel that we have a v.g. R function that really defines a set of phenomena that ~~[illegible]~~ should have a good, common P funct (P(A|Q)), then if t. pc for a new QA of that R is too low, we really want to revise that P rather than

.20 ~~Change the R~~ revise the R

N.B. ... ~~That~~ an R funct is an impt. kind of Context. ////

.21 Is "general context" ≡ "t. condition" in condl probty? Looks like it! — So t. idea of "context" is to find additional arguments for "t. Condition". In t. simplest type of t.D. of tokens we have ~~[illegible]~~ unconditional pd's — so a common (or Null) context.

.25 Note that "Context" can be for a single Token, or for an entire O_{n+1} soln, (or for a A_{n+1}?). ABCDEfghij

.26;.27;.28;.29 SN ~~In~~ 2 PD's of 387.15–.19: Use simple Bayes model for t. rel wts of t. 2 PD's. — (we assume t. The 2 PD's of 2 pd's operate in 11.)

Try writing "Summary" (as if to a novice) of "context" ideas and t. PD₁ vs PD₂

.30 — problem. — 391.05 →

.31 Context of a token, say can be of 2 kinds: ① internal: due to regys in t. O_{n+1} being constructed — it is part of PD₁ only. ② External: All other contexts: The $O_1 \ldots O_n$ context,

.34 Whether its a Chemistry or Math-problem, ~~[illegible]~~ any "sequential" info.

.35 T. original motivation of using context was that w.o. it, growth of t. language would cause t. pc's of tokens to ↓ so that as corpus ↑ in size, — More exactly, as no. of Concs. in language ↑. it would be harder i.e. harder to solve problems

SN In math. langs, as one ↑ ~~[illegible]~~ size of corpus, pc's of tokens does ↓. It acts as if t. tokens didn't have normalizable pc's! Could t. effect be due to the normal rate of introduction

3TM

Monday 7 P.M. | 617.817.6325 ← Phil organ cell
85 chesnut hill rd | 617. 225 5700 ← phone at large
Chos off ~~Beacon~~ B.line. | For Rollo.

F

.00 : (spec 388.40) : T. idea of Mut/cross of O_n for O_{n+1} trials uses info that O_n fit corpus up to $(QA)_n$.
We also have t. info $\equiv$ that for all i, O_i fit corpus $(QA)_{1 \ldots i}$. — which is info useful

.02 for Backtracking. e.g. $O_{n-1} \to O_{n+1}$ has to fit $(QA)_{n,n+1}$ in addition to what O_{n-1} fit. (.10)

SN It mite be time (politically) to publish or prove NT (Necessity Term)er ALP.
If we have a ~~m~~ Algm that provides, for each ~~ea~~ possl. string, a seq. of approxns to a certain (same) incomputable D.f. — Does this D.f. have to be same as some Universal D.f. — i.e. can we associate a finitely describable UMC to it? .
Perhaps re read the older "NT" (~1979) paper.

.10 (.02) → For a simple sequencial corpus (sequencial predn), There are at least 2 (standard) ways
.11 to augment a code for an (part of t. corpus: 1) Terminate t. first subcorpus w. an end symbol (This costs ≈ $\log_2 n$) and code the new section "ab initio" or using regs in initial string. n = length of sub corpus

.13 2) Simply try continuations of t. codes for t. initial corpus; ~~The~~ They will (usually) give continuations ~~contins~~ that don't follow t. desired contin, but for each such code, we can add in "correction bits".

For BAG induction (a proby QA induction which can be regarded as a variety of Bag induction)
This is not so easy. We mite do .13 (corrections), by having t. system output something & have a standard way of modifying (Mut/~~cross~~ cross) t. outputs of t. system. Assoc. w. each modifn. would be a pc. This could be a "fixed" (conditional) pd, or t. pd could "evolve"

.20 as we have more cases from t. past.

A way to do .11 is by Recognition functions.

0.23.07 SN I was wondering about need of t. ~~Present~~ work on Action Algm. Evaln. (AAE)

.23 by David & Others. One impt. use of it is SM (ordinary SM strategy evaln).

.24 Another, more complex, involves choosing a stock & strategy pair so as to maximize expected future yield.
I haven't yet found nice ways to do either of these problems using t. Univl. D.F. (UDF) | TUD Turing Universal distribn
In t. case of ~~.23~~ I did try "Yield coding": There was a "reason" why it was N.G., but I was not ~~convinced~~ convinced w. that objection.

.30 It is not clear how it could be applied to .24.... (But perhaps it can!).

A trouble was: that a strategy wasn't legit unless its "yield factor" paid for pc of its descrn — so one needed an enormous ssz to tell if a strategy was any good. Actually, this is a common diffty when one is making predictions that are much smaller than noise!

Oct 22, 03

3TM

00:387.40 : If we have a good Mut/cross, PD2, then it will implement 387.22-.24 and we should get ~~[illegible]~~ an acceptable soln. in 387.38 w.o. much searching (= cc).

The idea of 387.22-.~~22~~24 was a genzn of the idea that in some coding environments, one can ~~[illegible]~~ make "Modifns" of a code sequence by augmenting $\overset{O_{n+1}}{x}$... (by "continuing" it).

387.22-.24 is a bit based on the idea of the "prob'y of $\overset{O_{n+1}}{x_2}$, given $\overset{O_n}{x_1}$" — how many more bits do we have to add to $\overset{(O_n)}{x_1}$ (or to its derivs) to obtain $\overset{O_{n+1}}{x_2}$?

In 387.22-.24, hvr, we add much more info than just $\overset{O_n}{x_1}$, to produce $\overset{O_{n+1}}{x_2}$:

We note look at allot the $\{O^i, O^{i+1}\}$ pairs of the past & obtain a probilistic relation to "Mutate" O^n to get good O^{n+1} cands.

While 387.22-388.10 is a not-bad understanding of "the 2 approps": I'd still like to understand the form of 266.26-.40! [I suspect that 266.29-.36 has the main idea.]

The expl. of 387.22-388.10 is actually not bad .... (It certainly could use more detail) — but is it the only justfn. of the use of PD2 for srch? ~~[illegible]~~ Is the idea of 266.29-.35 an essentially different aspect of the utility of PD2? Perhaps read the stuff leading (& post) to get better idea of "what was going on" then!

Some ~~[illegible]~~ refs: 363.14-.21R (ibid. .21R ff also seems ~~[illegible]~~ relevant)

~~[illegible]~~ 366.26-.40 ~~[illegible]~~ ibid

367.18

EN) 361.35-.40 ~~seems to be~~ is very relevant to 387.22-.24: — "It's the Recognition function" idea. It is one important way (& I'd like to genz it if poss'l), to make it easy to get "modifns" of O^i_n, that will ~~[illegible]~~ generate O_{n+1} cands ~~[illegible]~~ that all automatically work w. $(QA_1 \ldots n)$. This also saves a lot of cc! Certainly Recognition functions are much used by humans. I do, hvr, want to genz them a lot. — In general, our models for O^i include recognition functs. Look at my analysis of R functs in the Report & see how they can be genzd — in particular, in ways ~ to the ways humans seem to use them in induction.

The Recogn. functs of 361.35-.40 is one very common approach to the problem of 387.22-.24; The Mut/cross idea in PD2 is a rather large "jump" as a genzn of "Recogn. functs". I'd like a "set of intermediate genzns" to bridge the gap betw R. functs & "PD2" (≈ Mut/cross).

The forg. ideas are intimately tied in w. idea of Backtracking (q.v. 375 references). — That when it seems one cannot solve problem by Modifns (Mut/cross) of O^i_n, one "backtracks" to O_{n-1} or O_{n-2} .... : I had this "soln" to the backtrack problem on p 375 (reference). — This will have to be integrated into the "general" soln. of the problem of 387.22-.24

Oct 21, 03

3TM

00 : 386.40: (sfac) Troubles, while sequencial info. can be put in by t. framer, there will be much less of this (if any) ~~[illegible]~~ when TM is working real problems in RW w.o. a "framer".

[SN] I had idea that t. form of ~~[illegible]~~ t. O^2 soln in 385.34, was indep of ssz. — This is not true for s-induction: (Is it true for ~~d-induction~~ (d-induction) ?). In s-induction, larger ssz gives us a sharper ~~d~~ d.f. on values of ~~constants~~ parameters — Gives a narrower distribution! — In d-induction, there is no point of repeating a corpus element (in pc) → t. corpus is a set not a BAG

So its really a difnt distribn.

10 : 386.40 (sfac) : Re. t. 2 Goresos 386.30 : This is t. same problem as I seem to think I solved on 366.29-.40 — except that I don't ~~how~~ see how it solves it! It mite be well to go thru t. old stuff & try to summarize t. argts., & understand just what was done. Th. problem is of much import in phase 1.

T. idea of 366.37 was that phase 1 needn't be to picky or accurate!

.15 (SN) Conjecture: we have 2 p.d.'s A & B (cases cost) : If ~~[illegible]~~ we mix them (give each 1 wt, say) then If they have = wts, then if one has a sharp D.F. for a given argt. & the other has a flat D.F., then the sharp d.f. will win. If both have sharp D.F.'s, we will get a bimodal d.f. & both choices will be impt. If we think of t. 2 ~~[illegible]~~ corpus, due to ~~[illegible]~~ $O^1 \cdots O^2$ (sequential) v.s. t. [QA] corpus, → 390.26

.19 ~~if either~~ has a hy p.c. choice, t. ~~[illegible]~~ combination will give that choice. <ABCDEFGabcdefg ← Fine! ABCDE

20 So we have these 2 p.d's A & B (α − cases, cost − seq.) : How do we best use them to get good induction?

.22 [SN] Using QA corpus Only: Say O^2_n works for $QA_1 \ldots n$; To ~~get~~ a O^2 to work for $QA_1 \ldots n+1$, we want to use t. info. that "O^2_n worked for $QA_1 \ldots n$", — So we want ~~[illegible]~~ mut/cross of ~~[illegible]~~ O^2_n —

.24 or, t. ~~corpus~~ population of $[O_1 \cdots O_n]$ — But it seems like t. final arbiter is t. QA corpus Gore via AZ.

368.35-.40 seems to be related i.e.

Using Lsrch does put condsin ≈ pc order, so one ends up w. best $\frac{pc}{cc}$; There may be ~~[illegible]~~ larger pc's available, but they have larger cc's. If we use $PD_2(O^2 \ldots)$ to guide Lsrch, & PD_1 as [illegible] Gore — How will results differ from using PD_1 ~~[illegible]~~ to guide Lsrch & 30 PD_1 as Gore?

A Guess: The PD_2 Guidance will give ~~[illegible]~~ us maybe not best pc solns, but it will get them faster than using straight PD_1 Guidance — So maybe better $\frac{pc}{cc}$? (All "pc's" are viz' PD_1).

Perhaps PD_2 would find a soln. befor PD_1

In t. (B) search, we have 2 vectors: $\overset{\alpha}{a^2}$, t. aprip of O^2 ; and $\overset{\beta}{\prod_{j=1}^{n} O^2(A_j|Q_j)}$

In Lsrch, we chose O^2's in a^2 (PD1) order, then mult. by assoc β, we look for maximum products

.38 In (A) " " " " PD_2 " " β. " " " " "

Oct 17.03

3PM

QA corpus (case 2) A → PD1

$O^1 \ldots O^i$ corpus (case 1) B → PD_2

00 : 382.35 : A possl. answer: Any info TM obtains in any way: thru external corpl or thru its own experience in solving problems (Traces) can be included in t. GPD, & can affect the h( ) functions of any possl. problem, PST pair.

We may not wish to include certain dependency possibilities because we apriori feel that "their cc is > their Utility" — i.e. "not worth the time".

≡ Were I was on "in fixing up the ~~rep~~ report" w. a summary of "~~all~~ How things were — what needed to be done ... etc". See 380.26 ff.

10 → I guess t. problem in both $\binom{TM}{c}$ & $\binom{PE}{d}$ is this: There are 2. extreme ways to do them: ① The "expert system" way of p'g'ing in all need heurs to solve all of t. problems (no l'ng at all) ② No "p'g'ing of heurs at all, just start w. primitive set of cncs & suitable TSQ. — preferably w. as large CJS's as are ~~practical~~ achievable w. current technology.

[Actually, ~~we want a mix of~~ ② may be too hard to do, take too long. 3.42 13.7
We want a mix of ①② : Some Q's are: ⓐ how much of 1 v.s. 2, ⓑ how to implement various "speed ups" of 1, by hints, p'g'ing in some impt. concs, (what ~~appr~~ aprxs to give these conc).]

20

The BIG Q then is how best to mix ① & ②. Usually, any of ② is bad in t. sense of TM not (nearly) being able to learn t. ~~heur~~ heurs (or whatever) that were "p'g'd in": [Ramanujan was example of Intelligence w. important parts missing: yet v.g. intelligence!]

To some extent, one can't do ① completely: There is initial choice of primitive cncs — but this can ~~[illegible]~~ contain a (relatively) small amt. of A.I.-ness.

.27

Case 2 (366.02-.03) (Based on QA corpus) A

Case 1 (366.01) B GNOS ~~&~~ $O^1, O^2, O^3 \ldots O^n$ extrapoln.

30 : 385.40! Say 385.34 is t. "True Gore" & 385.35-.40 is a modified aprx used for search. Presumably, t. only reason B mite be used, is that it would realize that certain ~~[illegible]~~ solns w. very good "A" have excessively large cc. Ideally,

$$B^{(x)} \propto \frac{A(x)}{cc(x)} \quad \cdot \quad \gg \frac{A}{cc} \qquad X \equiv \text{cand.}$$

— Which would seem to be not much of an advantage since L such trials are in ≈ $\frac{PC}{cc}$ order

anyhow

In T. discussion of 385.35-.40, we are using ~~[illegible]~~ random & sequencial properties of t. T.Q.

→ See 375.04 for ~~[illegible]~~ refs. on Rizz[illegible]. on 366.29-.40 is that I Solved it.!

space → 387.00
→ 387.10 space

Oct 15, 2003

3TM

"RAYMOND'S REVIEW" .16

②

00 : (SN) for Dowe's Paper: In Intro! After doing the Paper & all elections; THEN! "We ask" — How does This work fit into a Prior work on G: Univl. d.f., MDL, MML, etc.

In ~~QAMTM~~ QAMTM! we have to lrn a certain set of Concs, in ≈ a certain order. If we run ~~int~~ into trouble, we have to "Back track". Normally, T. Sci community has trouble deciding when to back track! ~~but~~ but it is working NAMTM problem.

For QAMTM, we may be able to do back tracking ~~w~~ w. a simple backtrack criterion/threshold.

[I do remember a recent (w.i. TM Test 2 wks or a Month) soln to t. Backtracking problem: see 375.07 for Refs]

10 Along w. QAMTM lrning of ~~#~~ d-funct-type concs, It has to lrn Contexts which are d-functions. Also, in Lsrch (& probly many other srch Systems) one has to lrs an agryd as Conds — which is an SoFunct — but this agryd is more or less "given" & for some extent, known. T. Contexts are less routine.

Oct 15, 03

.16 [SN] [Raymond's Review!] This is 2 Things ① A listing of How & where to get recent & older papers on t. web. ② Book/paper reviews by specific editors. These need not be up to date! Shalizi, e.g., would be a good book reviewer. Also, refs to Amazon's book reviews & Does Barnes & Noble do book reviews?

20 But we would be mainly web-oriented; so a person usually could get t. Material (or a good review of it) on t. net.

Existing "Reviewers" Shalizi, McCarthy, Minsky, Hartel(?), Gunkel(?), kurtzweil, Nature, Science Daily, Sci Amer. New Scientist.

We could Mix This w. t. Dartmouth '5C idea of "Boolean Editors"

Some IMPT ideas in TM rsch!

1) The ~~base~~ "WON" soln for often! Use of $h(t_i)/t$] max as Goal. $\left(\approx \max \frac{pc}{cc} \right)$

2) That Lsrch is ≈ optimum if "all info is in P.D." (does ① enable us to put all info into PD? — well, when we are able to use WON, then Lsrch is obsolete! — The WON only makes 1 trial! ☺.

3) We can design TSQ's by looking at TSQ's for humans, identifying "adequate" heuristics. — Then design TSQ's to learn heurs (w. perhaps "HMB").

We must factor Heurs a/o "PST's" & put factors in Dictionary.

4) For QA induction; t. actual Goal is Max : to find 1 or more O^i's ∋ This is large as possbl.

.34 $$\sum_i a_i \prod_{j=1}^{n} O^i(A_j \mid Q_j)$$: Then use $\sum a_i O^i(A_{n+1} \mid Q_{n+1})$ to get a p.d. on A_{n+1}.

.35 5) In searching for solns to 4) one can use sequential info in t. ~~TSQ~~ TSQ (corpus): Also any ~~C~~ Contextual (local a/o General Contexts) — Remember — Humans do get this info.... Doing induction w.o. it is an unacceptable handicap. → 386.35

3TM

Forms Needed. 1065 is K-1

4952 ✓ Ipd.

1116 ✓ foreign tax credit. f's 2 {5 notes}

6781 ✓

2001 4757 ~ 2PM should start action 4PM or 5PM, blood clot should be well formed.

~13:30 ast time for urine

①

.03: 383.07

1) What is main Bottleneck in present QA system? (Says tentatively, "Phase 1").

2) ~~$~~ Side Q's: Is phase 1 by itself feasible? would it be good enuf to work problems ~~[illegible]~~ of enuf diffy/complexity so it could go to Phase 2?

3) Also, Re: the phase 1 phase 2 dicotomy: essentially diff't ways to Lsrch. Phase 1 is Lsrch Phase 2 is Won.

4) Just ~~[illegible]~~ ~~[illegible]~~ how is "Adaptive" done in Phase 1, & is it really effective?

— Is it even Legitimate? (I.e. not h.H. pseudoprediction),

5) Is there any way I can use various "Machine lrng" techniques that have been developed already Like GA, ANN, RNN, Machine translation Lrng (large corpus), New String Compression Methods; Methods used for Music Style Lrng, ... etc.

Any method that would speed to entry of system to Phase 2, ~~[illegible]~~ could be useful — even if they take lots of cc/mm ~~[illegible]~~ 2/0 ssz, initially.

Use of GA for O² srch. i.e. in srch to find ~~[illegible]~~ Good O².

In Lsrch, we solve it as a INV problem! So "fitness funct" is a Yes/No type — Not any good for GA! Well, we could have a "partial function" that operates ~~[illegible]~~ correctly on some of the QA's but not on others. We should be able to express this as a "partial coding." — Hvr. This diffy occurs mainly in (d-funcs MTM mode) Not in (s-funcs NMTM mode)

IN ~~[illegible]~~ d-funct mode, Did we use "Recogn Functs" or was this for s-functs only?

IN ~~[illegible]~~ s-funct QATM: I guess a main diffy would be conds assigning Pc=0 to some correct Ai's: Or ~~[illegible]~~ very low Pc's so, in Lsrch, it would take very long to find them.

N.B.: The basic idea that I have about ~~$~~ QA lrng — (& all lrng), is that the main idea is one lrns to do simple problems by ≈ random srch. α That one then uses the concs used to solve simpler problems, to solve more diff't problems. ~~[illegible]~~ loop to α

I should really try to get TM to work like this. ~~[illegible]~~. — to get to Phase 2: Phase 2 seems to me to be a really ≈ optimum ≟ "Self improving" system!

0[illegible].03
3TM

00:382.40: So, in Lsch mode, its not clear as to what the guiding pd is to "adapt" to — what "is" a v.g. p.d. ? Well, an optimum pd would give very hi pc to a funct. that solved all of problems very fast. So perhaps a goal would be total time for soln. of all (Inv.) problems. ($\Sigma T_i = min$) Hvr, we have to assign pc's! [Hvr, $T_i = \infty$ for some (unsolved) problems].

Instead of ΣT_i: some orderings: ΣT_i compares 2 PST's that solve same number of problems — $T_i = 0$ if time $= \infty$. otherwise solving largest no. of problems w. $T_i < k$ is a kind of Goal.

.07 So the pc could be the possibly that the function had max? of that Goal. → 384.03

0.5.03

3 TM

382

.00 : SN "Max Cross Entropy" may be related to "QA induction & its "Convergence Thm".

In simple MEM, one has constraints between pc's. One is Given a/ say non-linear model w. a finite set of probabilistic params, and one has constraints among the params.

In Max Crossentropy: MAYBE One is given a perhaps known data set, Q, & an unknown data set, A, T. relation of Q to A has a finite no. of probabilistic params to adjust & some known constraints on these params. Max Crossentropy defines a solution set of params. It is same as ALP in the cases covered — but the QA model is relevant, ~~[illegible]~~ the conv. thrm for QA induction is relevant.

.09 ENQ: If one has adequate ssz, T. incomputability of ALP is not relevant. } ... now

.10 — Hvr, in A.I. ssz is rarely "adequate": Is incomputability of ALP then relevant?

SN In "Barn Seq." (=AZ generated), Laps rule is simply implemented to name all codes.

Could OSL be done in ~ way?

We start w. precorpus. ~~[illegible]~~ T. first token is selected at random from precorpus. If output matches true corpus and a symbol is requested, we have fixed pc (= α) of continuing in existing code ~~corpus~~ or [pc = 1-α of] requesting a new symbol by randomly choice from extant code. We continue until "stop" or we ~~code a no~~ print a non-corpus symbol. — At which pt. we backtrack to last unexpanded node.

So each code is obtained by starting at precorpus, going forward, then occasionally looping back to pt. in ~~[illegible]~~ "code thus far" then going forward, etc. : ☒

Clearly any code can be written this way, since at worst, we can just code single .20 tokens from precorpus.

.30 : 381.40: So, while one may start out using the Lsrch-generated corpus, It's probly necy when in WON mode to generate the needed corpus more intelligently (More oriented toward the needs of WON). Also note 378.31 on "partial work" on any problem being heuristicly useful: How to ~~[illegible]~~ get this info into the h( ) functions, is unclear.

→ A Guess is that the trials obtained in WON ~~[illegible]~~ "prob. solving Mode" will not be adequate .35 for subsequent improvement h( ) calculations. IS THIS TRUE? → 386.00

3:00
3:13 P
15
~~18~~
23

Essentially problems: ① When in WON Mode, what is good way to deal w. need for augmented corpus for better h( ) d.f.'s. 378.02 - .09; .20 - .40 ~~[illegible]~~ discuss this to some extent. — Views it as time spent in self-improvement.

② WON & Lsrch seem to be quite different in how they work:

3TM

$9\frac{1}{2}$ $\frac{9\frac{1}{2}}{9} \approx 12\ \frac{1}{3}$ $12\frac{2}{3}$ 88.6
9 →

[redacted]

direct

00 : 380.40: So [redacted] direct [redacted] Lsrch is a real possy. Now, what about t. "PST Grammar"?

.02 It is a pd on PST's. ~~We have to convert it to something like a~~ (.027) Is this PST Grammar a purely "Phase 2" object? || How we get empirical info on Prob-solving is not. PST Grammar is unclear. → see [378.02 -.09, .20-.40 on "Grammar t. set of PSTs"]

Re: (.027) & we can use the elements of t. Grammar as part of t. "Reference Machine" in t. "straight Lsrch" approach.

T. "empirical" info on PST; → h( ) info that we get from t. straight Lsrch problem solvers may not be so good. The "problem solvers" for most problems will be very similar, because ~~[illegible]~~ "adaptive" Lsrch trys to find a ~~common~~ common pst for all problems.

10 Q: Lsrch seems (a bit) non-el in that it looks at t. problem & tries to find a soln. It tries to find a "Universal" function that will do that for all problems.

Phase 2 breaks down problem into 2 parts: first, look at problem, decides which PST to try first. (Then, by watching ~~trial~~ attempted soln, it [redacted] modifies t. functional parts & may decide on ~~[illegible]~~ a difrnt trial. So it is an algm, but (I think) a v. good one.

function of parts

While "adaptive" Lsrch may be rather non-el, it is also not very "perceptive": i.e. it only knows which "√" conds pst's have solved which problems, but it doesn't look very carefully at just "how good" each soln was.

"Phase 2" is a development of t. Mxt [redacted] Mixed corpus Thm — in which all of t. "info" in TM is in t. GPD, … MCT =

20 → Also, using "pure Lsrch" it's not clear what t. ~~[illegible]~~ "guiding pd" is about. What is it t.ed of?

0.5.03 MCT was designed to deal w. this Q.

Phase 2 Differs in Spirit from Direct Lsrch, in that in Direct Lsrch (or Adaptive Lsrch), it is not clear as to what t. "Adapted p.d." means: what is it t. pd of. Adaptive?

In Phase 2, it's quite clear. In phase 2 we have a complete set of PST's, generated by a universal Algm. <T. Algm also assigns an a priori pd to t. PST's (But we never use that a priori pd as such).>

.27: 378.40 Somehow, we get a corpus of $[PST_i, problem_j, T_{ij}, \overset{1}{0}]_{i,j}$ data: fail or succeed. Then we use QATM to induce a p.d. on t. pc of soln. of any PST_k, $prob_l$ = $P(T_{kl} | PST_k, prob_l)$. Time

T. corpus of .27 could be obtained by applying Lsrch (unadaptive or adaptive) to solve a

30 set of problems. For a small set of problems that are very Disparate (difrnt from one another), there is not much difrnce betw. adaptive & non-adaptive Lsrch.

.32 [There is a discussion of ec spent on obtaining t. corpus of .27 on 378.02-.09; .20-.40]

So there are these 2 essentially difrnt methods of trying to solve problems. — Is one better than t. other in some ways? Is one "Uniformly" better than t. other? Can they be combined? (which is, I think, what is done in t. "Report" as of now).

We start out w. Lsrch. We get lots of data: Almost all of t. Inv. data is failures, — the t. bit/oz data covers the space of interest better. We can use this data as Corpus to get h( ) functions for WON, but its not such a great corpus [see (.32) for ref. to comments!

see 382.?

Reinforcement Machines RTM .04

(Spec)
00: 379.12 : Rewrite of 379.00-.12 refer to Now / sections 2.1, 3.1 (Braso INV/OZ using Lsrch only) → .24

SN last ¶ of §3.1 needs rewriting..

.04 SN It is possl. to take Reinforcement machines use it solve OZ problems, w. o. any of the problems specific to RTM: i.e. We give RTM [redacted] as inputs the descr. of a OZ problem {M(x) and t₀},

We then give it reinforcement M(x) every time — so it learns how to do OZ problems.

10

~~The~~ advantages of this: It may be possl. to get TM to do a lot of automatic "Self improvement" in RTM mode — to develope a lot of techniques for §I. This would also be useful in pure "OZ" mode.

→ Correction for Second last ¶ of ~~?~~ of Report.

probability distributions, {h(t)} for the Timer c.f. will take .. New the Prob

From each ~~?~~ ^prob. distribution, h(t) we obtain a figure of merit, ~~?~~ ^α, that ^α gives an optimum ordering ~~??~~ ^for its associated trials ~~?~~ Fᵢ functions, ~~?~~ we ~~use this figure of merit to guide our trials.~~ Then use this α to ~~?~~ ^order our trials.

20

~~?~~ Last lines of § 3.2 : .... To ~~?~~ improve their solutions

~~This gives~~ ^These improved probabilities give us better h(·) functions, which enable us to {recursively} ^eventually improve the h( ) functions and so on. This can result in exceedingly good h( ) functions and exceedingly good solutions ~~?~~ to both, ~~inversion and or~~ optimization and inversion problems.

.26: .00 (See 379.00-.12} — to be rewritten):

In [redacted] both a (Inversion) and d (optimization) of Phase 1, There are several approaches possible. [Note to rjs: These several approaches have corresponding ~~?~~ approaches in ~~?~~ a & b of Phase 1]

30 Actually, the methods listed in 2.1 & 3.1 — using simple Lsrch (w. possibly diffnt ref. machines for INV vs. OZ) is not ~~very~~ good. It uses the same D.F. ^(adaptive) for all problem types! ~~The D.F. is "Adaptive", it still uses the same D.F. for all problem types.~~ After it has solved ~~?~~ several problems this way, using Lsrch: It will begin ~~?~~ Phase 2. || On second thot, We really only need 1 ~~?~~ d.f. for all problems!

.36

We want a single function that looks ~~at~~ problems and decides on soln. method. ~~?~~ T, Pargo; ^.36 ff I wrote w. INV in mind. For OZ problems, a single pst can look at an (OZ) problem & decide what ("sub" PST) should work on it. This → ~~?~~ PST ~~?~~ is a single pst that does all ~~?~~ (OZ) problems.

Change section Numbering in § 2 & § 3:

~~Subsection~~

§ 2 is ^main section ~~?~~

§ 2.1 ~~?~~ Solution ^by Lsrch

§ 2.2. A mult improve method.

§ 3 is main section

2.1 soln. ^by Lsrch

2.2 a Multimproved...

Go thru doc looking for 2.1 and 3.1

2.1 → 2.2

3.1 → 3.2

examine each Change! There will not be many

See that FN 3 ~ p15 comes out OK.

2.1 occured many times; 3.1 only once.

Check out use of P_1 etc. last line of ~~§ 3.2~~

Just what eq. is referred to? Mention & eq. no.

I'm not sure it is meaningful

N.B ENV | OZ

00: 378.00 Spec: In both Ph1 (c, d): we could just start out w. a general reference UMC, & do Lsrch. The tsq's would have to be ~~be~~ very long before we got it to work interesting problems.

To ~~avoid this~~ deal w. this, we start w. a Ref. UMC. that has as s[illegible]s, ~~procedures~~ procedures, macros that can be easily combined to yield PST's that are known to be useful.

We could get a similar effect w. a TSQ, but the TSQ would have to be very long.

The advantage of doing it w. a TSQ is that when ~~[illegible]~~ TM has finished the TSQ that teaches it a large set of useful TSQ's, we are fairly certain that it can continue to discover new PST's of these kinds in the future.

.10 If we decide not to use a very long TSQ or ~~if~~ we ~~[illegible]~~ are very careful in designing the initial set of macros, we may get a TM that can continue to discover new PST's that ~~we~~ we were unaware of — but we would be less certain than if we had used the very long TSQ for training.

.12 .00 – .12 needs to be written up better. ——→ 380.00

Phase 2. Phase 2 is one of the major ~~the main~~ goals of this system. In Phase 2, the system uses inductive inference to improve its updating and searching techniques for both INV and OZ problems. Since inductive inference can be regarded as an OZ problem, Phase 2 ~~the system is~~ recursively improves its technique for improving itself.

20 We ~~are~~ ready to enter Phase 2, when the ~~[illegible]~~ operator induction system of Phase 1b, has had enough training to effectively ~~work~~ ~~the kinds of problems involved in Updating — as described in Sections 2.1 and 3.1.~~ ~~do [illegible] be effective in~~ implement the updating schemas ..... described in sections 2.1 and 3.1.

Another Major goal is learning to understand English. Though it is possible ~~for a~~ to do this ~~machine to learn some English~~ in Phase 1a (deterministic prediction), it would ~~[illegible]~~ ~~Phase 1b~~ be much easier ~~[illegible]~~ for it to learn in phase 1b (probabilistic prediction). ~~This learning would be [illegible]~~ It could, be done even more rapidly and ~~more~~ reliably after the machine had gotten to Phase 2.

30

It will be noted that the various Phases and subphases need not be ~~done~~ implemented in the order we've given. OOPS is able to solve the inversion problems of ~~of~~ (Phase 1c), but ~~not the probabilistic induction problems of Phase 1b,~~ it has to be seriously modified if it is to solve the probabilistic induction problems of Phase 1b.

Serious Criticism of WON ~~of~~ feasibility of getting good h( )'s: .02-.09, .20 ff

.00: (Spec 377.32) NB P361 has some good stuff on it not included in 377.00-.40

Phase 1 c : (Discuss creation of "Grammars" for both c & d. We could optionally not do {grammar w. factored examples} — but then TSQ has to be Much longer) → 379.00

.02 SN In WON exo~~[illegible]~~ation, we will normally not "exercise" many of the PST's, so we wouldn't have much data on them: ∴ poor h( ) estimates.
As part of the WON system, one will have to "exercise" "various PST's for info purposes" not merely because ~~they~~ these PST's are "good bets". The devising strategy as to what PST's to try on what problems (not merely on the present problem) sounds like a diff't problem! When working on problem P_0, it may be once necy to try a few sample PST's on P_0 to get a good idea for lots of h( )'s. → .20

.09

.10 — Not mentioned as 1mpt subgoal (Milestone): English [redacted] understanding. It can be put into Phase 1 ⓑ form

— ~~Make~~ exactly 2 NMOATM (Ex: con bom Phase 2 as well ... should be if we ever get to Phase 2!)

SN ~~[illegible]~~ P.D. obtained by by full Univ. D.f. are "unbiased"; But ~~[illegible]~~ many (I'm not sure if not most or all) approxns can be very biased. E.g. one can Prefer only those codes that give certain predns. There may be ways to do unbiased approxns: ~~[illegible]~~
e.g. State "srch technique" & CB. ~~[illegible]~~ How Do I define "srch technique" so it wouldn't be biased? It should be simple defn., perhaps common to all Reference Machines.

.20: (.09) → Actually, for INV problems, one usually doesn't get much info if CB is small:
For small CB we usually fail to solve it, & this tells us little. If we do solve it, then we quit! So we can't get "positive" info. preceding our h( ) evals. — A corpus of this sort is ≈ useless to get info for WON.
For OZ probs, it's diff'nt. For any CB and any Problem and any PST, we usually get some info — i.e. how much G was obtained. (If CB is too short, hvr, No G will be obtained. — This is ≠ −∞, because we are interested in "Expected values" & the "−∞" screws this up. (It's part of an "improper linearization of f. G")

Hvr, INV probs are usually converted to OZ problems. — or perhaps to "GPS form"
— having a vector Goal [— corresponding to "vector h( )". — The result is that by working on a problem a little,
.30 one does get some useful feedback, [redacted] w.o. [redacted] completely solving the problem.

.31 Most Generally any ~~partial~~ partial work on any problem using any PST, can yield info re. any PST (same or difent), working on any Prob (same or difent). The way the ~~[illegible]~~ "partial work" info is obtained & is mapped into the h( ) df.'s Varies considerably in all cases! The exact (statistical) logical process by which this occurs, is quite unclear at present.

→ 381.27 Being relevant Continu.

~~9.30.03~~ 9.30.03

3TM

9.4 ··· 930
335 377
42 pp and $\frac{42}{28} = 1.6$ pp/d.

pp $237\frac{1}{3}$, $237\frac{2}{3}$ } Area on Augmented Table of Contents

00 : 376.40 : so Road maps Back to ~~B~~ 361.00 ~~ff~~:

First, an outline of the system and its parts:

Phase One has three parts: ⓐ ~~Math~~ ~~the Q~~ Learning Q A induction (or Operator induction) for deterministic problems. These are problems in which there exists a function mapping each Q_i into its associated A_i, and finding this function using Lsearch is within the ~~available~~ computation capacity of the system (i.e. it doesn't take too much time or memory)

ⓑ Learning Operator induction for probabilistic problems. Here there exists an unknown probability distribution $P(A_i|Q_i)$ ~~[illegible]~~ ~~[illegible]~~ relating ~~the Q_i and the~~ Q_i and ~~A_i~~ A_i, and finding this function ~~[illegible]~~ using L search, is within ~~the capacity~~ computation capacity of the system.

ⓒ Solving Inversion problems using Lsearch.

ⓓ Solving ~~[illegible]~~ Time limited optimization problems using Lsearch.

Phase Two ~~[illegible]~~ Uses the search and update methods of sections 2.1 and 3.1 to solve inversion and time limited optim~~ization~~ problems.

How far have we ~~gotten~~ in this program?

~~In (Rev 94?) we designed~~ Phase 1 a We have designed a training sequence for the deterministic problem of learning ~~[illegible]~~ to evaluate algebraic expressions. This training sequence ~~was inadequate~~ ~~[illegible]~~ suffered from the "scaling problem" — solutions to successive problems took longer and longer. ~~We need to design training sequences and~~ ~~[illegible]~~ ~~context~~ To deal with this problem, we ~~[illegible]~~ have to find ways to define and discover "contexts" (section 1.3) and ~~[illegible]~~ design training sequences ~~in which this discovery can take place.~~ to facilitate this ~~[illegible]~~ kind of discovery.

Phase 1, b ~~[illegible]~~ For probabilistic ~~[illegible]~~ operator induction, we have ~~[illegible]~~ mentioned some kinds of languages that might ~~be~~ used (section 4), but there are many more ~~[illegible]~~ ways to represent probabilistic operators. The ~~design~~ selection of ~~[illegible]~~ suitable ~~[illegible]~~ representations ~~to~~ these operators should be ~~implemented~~ made concurrent~~ly~~ with the design of the associated training sequences ~~to these problems~~. → 388.00

~~Very~~ Superficially → Phase 1, c ~~So~~ The training sequences for Inversion problems using Lsearch are very similar to those for ~~Phase 1 a~~ the ··· deterministic prediction of Phase 1, a.

Phase 1, d In Lsearch for Time limited optimization,

Discussion of design of problem lang for ~~[illegible]~~ OZ Inv probs.

9.29.03

3T My

00 : 375.33 can be usd for **General BAG induction!** ↙ Not so general : The PPM method as shown works only w. "string ~~like~~ (like objects" (see .07 ff)

1) How can we use t. Methods of PPM to associate pc's w. R functions (Like AZ or OOPS)?

2) Probably I can improve PPM.

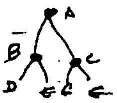
3) PPM is available as BZ2 [prgm. for compressn/decompressn.] which I have in D:\BZ2 folder

So I can actually try it out on real problems.

4) Can I make a variation of PPM that is for unorder [finite] BAG of strings?

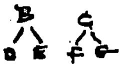
.07 Re: Expanding t. "context" idea to function trees & pc's of Tokens:

T. trouble w. "Tree" context, is that it isn't "linearly ordered" ← is this true? I think so;

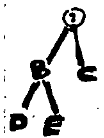
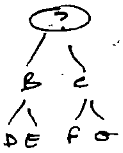
Consider funct tree:  Node A is a "mult" so mult is t. "Token".

50 Hvr. A = B·×C : B and C are tokens of equal strength

Unless we consider B, C (≡ C, B) to ~~Be~~ "t. context"

A deeper context is ~~a~~  I.e. ~~[illegible]~~ 2 layers.

~~[illegible]~~ Clearly t. ~~[illegible]~~ deeper context is desirable, it as relevant CSZ for it (If it always gives same Answer!) ☺.

Any way, say we have previous data on  But not on  then what?

Well, actually, I was thinking that any context ~~is~~ usable:

T. wt. of it depends on its SSZ and its inherent/dem complexity [redacted]

20 I can see how to apply old Z141, ~~[illegible]~~ — since it was really designed w. tree genzn. in mind. Generalizing PPM into trees may not be as easy.

T. Q is: would Z141 (using contexts only) give as good or better results than PPM?

There did seem to be a point at which PPM seemed not to work in a reasonable way ... i.e. When larger contexts were considered its entropy increased a small amt — But later the got a method w. arbly large context: I don't know if it ended up working better.

NB, t. paper: {Cleary, T, T, 1995} "Unbounded Length contexts for PPM" Is quite Understandable! See my notes on it.

93002

30 Back to "t. Milestones" : §8 as written, discusses only TSQ writing: This is perhaps O.K. — But I should refer to §8 — which is oriented to this! Also ~~[illegible]~~ §5 about TSQs:

Probably Almost all of present §8 should go to ~~section~~ §5 on TSQS, Troubles:

OOPS is not introduced until ~~after~~ §5! So perhaps, in section 5, ~~refer to~~ Say that" ~~we will di~~ — [later §(?) will will discuss OOPS: a pgm that uses TSQ's ~~to~~ for Adaptive Lsrch." §8 discusses the feasibility of using the TSQ of OOPS as the beginning of a longer program of (~~[illegible]~~ very general learning.)]

perhaps a better way to (say that entire sentence).

.00 : As soon as I get Roadmap done: Do bibl. review very impt! I'm really forgetting impt. things I've done in last yr!

↓Recent Items of IMPORTANCE↓

1) On srch for O^i funcs for S induction: fast way (impt.) 370.32-374.08

.04 2)(363.21 R) ff: "Paradox" of 2 kinds of apryds: One for ~~[illegible]~~ opt'zn criterion, & other to guide Lsrch for the first criterion. (367.19-.22 also) (366.29-.40) is said to resolve the problem.

Interpretation of Discrete & Continuous (drawings) srch for

.07 3) How To do Backtracking in S-QA~~M~~ (NMTM) ~~[illegible]~~ 369.04-.13; .19-.21

36.21 is on 589 FN#2.

This Routine is indep. of our choice of how O^i is modified to produce O^{i+1} cands. ← (i.e. & p.d. on O^{i+1})

We do discuss various ~~[illegible]~~ formalisms for devising O^i cands. Some involve sequential

.10 continuation (as in OOPS), others do not.

.09

NOTE: As I write the "Road map" Make list of probs that need to be solved, so when I go thru my Bibl. review, I can recognize impt. needed ideas!

9.29.02 SN on J's Godel mchn: In it's ~~[illegible]~~ srch method for proofs: for practical operation, "Hand waving" is sometimes OK. — Heuristic "proofs", plausibility argts. are OK.

These considerations may enable us to relax Gödel's "unprovability" results.

TM normally works w. probability — so instead of Logic, it uses probabilistic logic (reasoning)

(If we give logical rules pc's of $1-\epsilon$ (small ϵ), this will usually give results same as normal logics as $\epsilon \to 0$; But it will not work for infinite sequences of logical rules!

20

BZZ BZ2 is a Prediction by Partial Matching (PPM) compression technique that does get pc for each coded symbol. Its signif'ly better than LZ.

It can be used for categorization (perhaps) in folg. way: Say we have instances of category i: $[D_j^i]_{j=1|n_i}^{\leftarrow Bag}$ Make a corpus consisting of the "Bag": ~~[illegible]~~ with members of Bag selected ~~as~~ at their freq. count in the bag. Make a long corpus. Say we have a new symbol sequence, X. for this corpus append x & find the P_x^i.

We have several other possl. categorizations, each w. its assoc. data BAG. for ~~[illegible]~~ each category^i, compute P_x^i. These probys give rel. proby that x is in each category

30

If our randomly Generated corpi are very long, the preds having almost all wt. will be to various elements of the corpus.

.33 One way to avoid this: For each category data Bag, randomly permute the ~~[illegible]~~ elements and P_x^i for ~~[illegible]~~ x following it. Do this random corpus repeatedly — obtaining P_x^i for each — Then Use mean of all of the $\hat{P}_x^i \equiv \overline{P_x^i}$. Use Bayes for the different category bags, to get ~~[illegible]~~ rel pc of x in each category

9.27.03

3TM

00 :373.40 : T. Q being — just how much error can we tolerate? Also, using a small part of the corpus usually means that peaks are broader & easier to find.

.02 : 373.13 → An alternative View: If Model$_i$ has pc_i, we spend a time $\propto pc_i$ on it — so we minimize $\sum$ no. of test pts $\propto pc_i$.

373.25 ff on the effect of f( ) (≡ h, say) on the srch appears to be a diffrnt. tack: But important condtn.

T. Discussn. of approxns using only parts of the corpus, needs more work! It seems: Fri: 6:45

.08 Very relevant to S.M.

.10: 370.06 : → SN 9.28.03 Some Random Notes:

1) On "Complete" prefix sets (a) A prefix set is "complete" ⇔ every string (finite or semi infinite) is an extension of it.

(b) Any pref set (complete or not) can be divided arbitrarily into subsets — all of which are prefix sets. The subsets may or may not overlap.

(c) for ~~[illegible]~~ OOPS extension to be used for a 3IU, Its legal pgm inputs can be 2 non-overlapping prefix sets — one for S, the other for R.

(d) S can be extended from O^2 to devise new trial O^{2+1}'s.

(e) Hvr. 370.19-.20 holds: Its best to do TSQ & simultly decide on good representation(s) for 3IU (or ~~[illegible]~~ s-functs in general)

.20

2) In ~~[illegible]~~ revision of section on "Road map".

2) ~~[illegible]~~ Phase 1. (a) QA. d-functs (b) QA s-functs (c) Inv. via Lsrch (d) ~~[illegible]~~ OZ via Lsrch

Phase 2 (a) Inv via Won (b) OZ via Won.

Point out that these Milestones don't always have to be in that order. OOPS is a realization of Phase 1 (c). It can do Phase 1 (a) and a limited kind of ~~[illegible]~~ (d) (by searching for "improvements" in Inv. mode — ~~[illegible]~~

Letter to Juergen:

.30 Received your ~~[illegible]~~ Godel Machine paper: I don't yet see how it can be more universal than my own system — which is capable of ~~[illegible]~~ effectively ~~completely~~ replacing itself completely by ~~a~~ an arbitrarily better System. ~~However, your [illegible] machine is (I guess) a reinforcement machine — which has different [illegible]~~ ~~[illegible]~~ I have not read much of your paper, however, so it may become clearer ~~[illegible]~~ with continued reading.

~~[illegible]~~ For my paper, you suggested including a section on the self-development of the system. I have included a section (6?), but am not satisfied with it. I'm working on an improved version. Grace will be back in a week or so and help with the revision, ~~[illegible]~~

Cheers — Ray

9.27.03

3TM

00: : So, t. initial strategy is to randomly (or possibly regularly) do trial pt. ~~[illegible]~~ w. number of trials $\propto$ pc_i of $Model_i$.

* Say all pts start out w. low f( ). As soon as we begin to get hyer f values, we do trials near those f ~~[illegible]~~ values. We want them as far as possl. from other hy f pts, yet we want them to have hy f. — This gives us hy $\int f\, d(\ )$ parameter. The density of trials remains $\propto pc_i \times$ local f( ) value ("f" is pc of corpus in view of the model).

I certainly don't have exact details of such worked out, but ~~[illegible]~~ this I gives an

.09 outline of direction. a grup of models is $\boxed{\text{I. Main idea}}$: That t. density of trials in the region w.r.t. models

.10 of a model $\propto pc_i \times$ (apparent pc of corpus in that region)

After we find a region w. a peak in or near it, use the general Non-linear optimiz. method. There is a standard method; if it doesn't work, I have my own method — involving local quadratic approxn, then jump to peak of quadratic approxn,

.13 to make new quadratic approxn; loop to α → 374.02

Now t. forgoing ~~≡~~ should be used if there is no other "proved best" optimiz. method — e.g. in linear regress, one could use t. forg. method but there is a much more efficient way. We need .04 th. to find a good region and an idea of its size so we can use general N.L. That this method's working is indip of no. of dims of params is of interest.: Saves much time evaluating Models.

Still, hvr, we don't want to have many params because for a given ~~≡~~ ssz, the predns

.20 are not so good ... but whatever t. no. of params t. forg. will give us t. best predns — in locally linear predns ~~[illegible]~~ ≡ errors $\overset{<}{\propto} n^b$ I think, where b is t. no. of params.

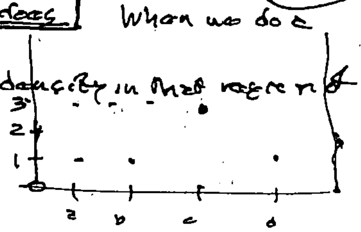
$$\text{No! } E_{\mu}\textstyle\sum(\text{err}_i) < \overset{\text{this is cost of discrete part of Model}}{c} + \tfrac{1}{2}\, b \ln n$$

I'm not absly sure of the "½" factor

½ ln k cost

.25 $\boxed{\text{N.B.}}$ The forgoing trick mainly has to do w. W_i of each peak. It doesn't much concern t. H_i of each peak, Well perhaps it $\boxed{\text{does.}}$? When we do a "test point" we get an f to value. The subsequent sampling density in that region of

.30 param space is $\propto f_0$. Say we have 4 test pts



~~[illegible]~~ c is now of interest so we try pts ½ way betw c & b; & betw. c & d. This new info suggest new trials to make. We want to find t. width of t. "peak" at c — also want to know if f gets hyer than at c.

Another trick: When we are ~~so~~ just starting out, we just want a rough idea of which models might be good and what params to use. To do this we ~~[illegible]~~ save much time by using only a small, random, part of t. corpus. As we get more info, each test pt. is more valuable & more exact values of it are useful, so we use a larger part of t. corpus. Why? Hvr. it is clear that much error in F is bad.

9-27-03

3JM

.00 : Say I'm just working on one model. ~~It has~~ It has a small w but large wH.

For any trial, the expected yield is wH, but the variance is very large!. If we do $\frac{1}{w}$ trials, we will get maybe one hit w. yield H. I.e. mean yield per trial is $\approx wH$, $= \mu$. $\mu = wH$ $\mu^2 = w^2H^2$

The Σx^2 is H^2 ~~on an~~ mean sq. is wH^2; $wH^2 - (wH)^2 = H^2(w-w^2) = \sigma^2$

which seems large. So $\sigma = H\sqrt{w-w^2} \approx H\cdot\sqrt{w}$

$\frac{\sigma}{\mu} \approx \frac{H\sqrt{w}}{Hw} \sim \frac{1}{\sqrt{w}}$ So we need $\sim \left(\frac{1}{\sqrt{w}}\right)$ trials to get some idea of how large $\mu = Hw$ is —

But I think we need no. trials of $\sim \left(\frac{1}{w}\right)$ (which is square of much) to have a good chance of getting one hit! (This seems paradoxical!) so $\frac{1}{W}$ v.s. $\frac{1}{\sqrt{W}}$! ??

.10 **No, No!** we did $\frac{1}{W}$ trials: got one hit so yield for total $= \mu$ was wH. ; σ^2 was about wH^2,
σ " $\sqrt{w}\cdot H$

so we got $\frac{\sigma}{\mu} = \frac{\sqrt{w}H}{wH} = \frac{1}{\sqrt{w}}$ after $\frac{1}{w}$ trials.

(This is mindful of my Monte Carlo method of trying to find out how many coefs to use in linear (or non-linear) regression) ...

While the $\frac{\sigma}{\mu}$ may be quite large, if one does get a hit, one has an idea that wH is quite large & w is small. The value of $\frac{\sigma}{\mu}$ doesn't give a good picture of one's "state of knowledge" after $\frac{1}{w}$ trials — After one has found a "hy pt." one knows ≈ what w is, and ≈ what H is.

Another tack: After finding an unusually hy pt. on some model, try pts near that one — try ≈ "Hill climbing".

.20

Given Several models of different apri∅ How much cc to investigate each?

Should cc be ∝ apri? Start random trial on highest pc model.

After k unsuccessful trys we think $w < \frac{1}{k}$, so we do trials on another model with lower pc, until we know that its w is $< \frac{1}{k}\frac{pc_1}{pc_2}$. — So until we get some hits, we timeshare trial time ∝ pc of model.

After we find a few high pts, the local pc is higher & we do more time share in that region.

.29 We don't have to do random trials: we can do uniformly spaced trials. Then double no.

.30 — of pts by putting a new pt. ½ way betw. 2 old pts. — Then double again, etc.

.31 Each time we double & find no "great val(ue of pc)" we halve our knowledge of w.

— so a soln. if found is narrower & worth less — So for each model, mean dist between pts is $\propto \frac{1}{pc\ of\ model}$. ∴ density of trials is ∝ pc of model.

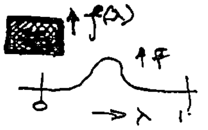
An idea that comes from .29–.31 (on doubling no. of pts): it would seem that having >> 1 dimension would really ↓ value of each trial! I.e. no of pts needed to narrow down hypervolume by ½ is $\propto 2^{\text{no. of dims.}}$ No: Say n dim space w. k^n pts, uniformly placed, then volume of an empty region is $\frac{1}{k^n}$: i.e. $\propto (\text{no. of trials})^{-1}$.

So dimensionality seems to be irrelevant.

9.27.03

3TM

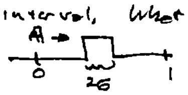
00: : Say the p.d. is 1 dim: [redacted]



f. on the param is λ $0 \le \lambda \le 1$.

We take uniform dist. of ~~#~~ λ on 0,1 interval, What is expected value of largest λ found?

Say we do n trials. Simplify



with n trials, the prob'y that none will have a peak of 0 is $\left(1-\frac{1}{2\delta}\right)^n \approx e^{-\frac{n}{2\delta}}$ If $n=\frac{1}{2\delta}$ prob of find ~~[illegible]~~ being A, is $1-e^{-1}$

So it would take $\sim \frac{1}{2\delta}$ trials to find the peak. The argument holds in > 1 dimension as well.

Another way to do this would be to sum the pc's of trials: Just summing the largest 100 trials thus far, would be fine. Just a simple sum is fine: In floating pt., neglect ~~rider~~ of very small contributors!

10 — If there is only one peak, The point of ~~many~~ prob'y is irrelevant, and the sum of pc's gives the wt of that point.

If there is > 1 peak we should keep track of the sum separately for

.13 each peak, to give the relative wt. of its prediction.

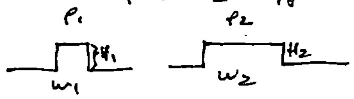
TM Goal: To find simplest poss'l. path to solve arb'y diff'lt. prob(s)

SN This may be how it differs from Godel.

In .10–.13 if we merely want prediction, we can actually "sum" all of the predictions of the various trials, w.o. actually computing the probs of each predn.!

20 What 100ff seems to be saying: If we have a discrete set of d.cr. of models w. continuous params, A ~~not bad~~ V.G. way to search is to just ~~try~~ discrete models in pc order & try random continuous params. Maybe have the time spent ~~on~~ on random srch ~~for~~ for a discrete model, be ∝ the discrete ~~pc~~ aprip of that Model (?).

Say I have 2 discrete models of aprips p_1 & p_2 resp. They have "~~[illegible]~~" equiv of 2δ widths w_1 & w_2 resp. How best to search?



Also peaks have hts H_1 & H_2: w_1 & w_2 are apsl known! ~~p_i~~ w_i are apriori unknown.

do the trials on cases i.

Say I spent [illegible] What is expected yield

30 — The mileage I get per trial is ∝ $w_i \cdot H_i$. [redacted] expected "yield" from 1 trial on case i

is $w_i \cdot H_i p_i$..

.32 After many trials ~~[illegible]~~; perhaps make the relative no. of trials on model i

.33 ∝ (mean yield per trial) of model i..

Well, say Mod$_1$ has a large w_1 but not so large H_1; Mod$_2$ has small w_2 but much larger ~~[illegible]~~ $w_2 \cdot H_2$.

So we would spend most time on mod$_1$, untill we get a hit on mod$_2$

If we start w. random trials: At first we don't know w_i, but eventually we begin to get ideas of how big they are. I think we get into an expected $w_i H_i$, eventually;

3TM

Kind of BreakThru on Evalns of continuous ~~params~~ params of predictive funcs, O^2. How to do Lsrch over a set of cand funcs

(.32 to at least 374.08
373.09–13 is a conclusion: But other simpl ideas follow

.00: (5 Pac 369.35): If the [redacted] S input: R input were some how des anyushed — Perhaps this would fix
distinguishable.
..369.26–.35? ::

The (363.14–21L) model merely postulates a (probablistic) ordering of the O^2 (cands model).

2 ideas: 1) First "extending" O^2 will not work, unless the prefix set R is not complete, so

.06 there is a set S' of pgms, of which none of R are prefixes. The S continuations come from S'.
(see 374.10 to 374.20) for a ~~[illegible]~~ relevant discussion of prefix sets → 374.10.

2) I can use one of the regular inputs to AZ as an "R": The way it works: R's always consist of a binary sequ with end symbol Δ. The pc of Δ depends on the distribution ~~[illegible]~~ of R's: If there are many long R's then Δ is small.

10 Δ ≈ (mean length of R)$^{-1}$. ∄ we can analyse this [R] i.e. by assuming CB or by using finite CB. In the latter (practical) cases, we will assign pc's to 0,1,Δ so as to maximize pc of data. This will be Lap's rule — so pc of each symbol will be ∝ its freq. for that particular ~~[illegible]~~ Q_i. ← (I'm not sure of this! The counts maybe should be over whole corpus).

This I will probably depend much on how many A's we examine for each Q_i. Would like to do a lot so that Normalizing them would be a good approx.

Anyway, this could work ok as an "R" input. This

.19 [Hrs, I think best approach is to: Rethink ideas of (363.14–21L) on C^o ordering, &

20 actually do a t.s.Q. Then decide on good form(s) for "R" input.

So write up the Road map: Making these ideas clear. ABCDefghijk ▨ (.19–.20)

Also write up alternative possl. forms of R, of ∃ I U — or more Generally of O^2. e.g. Bernoulli forms (≡ AZ). functions w. params, like $ax^n e^{-bx}$, etc. (we only need to know ≈ $\frac{n+1}{a}$)

361.00 It is a good ≈ "Outline" of the road map: It does need filling in, & expansions of just where the (diffys problems) are — & how Bad they are.

SN A common type of S function: Given the central (highest pc) object, is a way to assign pc's to objects more "distant" from the center. This is a "Clustering" approach.

30 — One could have multimodal dils — w. ~~[illegible]~~ > 1 "center". Each "center" could have a wt. So the pc of a pt. is the 1/wtd sum of ~~[illegible]~~ its distances to ~~[illegible]~~ all of the centers. Definition of "distance" simple.

.32 SN Finding peaks of continuous pd's. (S-funcs). Getting the ≥ probability (ALP) Search the Time $\frac{cc}{pc}$ where cc is from per trial & pc's ~~[illegible]~~ integrated pc of the model(s). This is (I think) true of one regs Monte Carlo Lsrch. This is very interesting, because it automatically gets around the $\boxed{\text{wasteful duplicates}}$ of normal Lsrch. Look into this carefully! It would be a "ε real Break thru" "If TRUE" Actually, I don't think one uses "Monte Carlo Lsrch"! It's more like random search.!

This d.f. — Tree us down to any specific → ~~The~~ d.f. that determines

00: 368.40: Well we have the pro-tem. CB = CB₀. We can search all of the Q_{i+1} to Q_{n-1} successes, with CB = CB₀. If no success, try all $Q_1 \cdots Q_{n-1}$ successes w. total CB = CB₀ — for entire $Q_1 \cdots Q_{n-1}$ set. (Note that the total time for each level of i = n-1, n-2 ... etc will be ≤ CB₀ because the sum of the pc's will be ≤ 1

.04 N.B. I think I'm using 363.14-20 as the model for 3IV. It is very "sequential" & Mindful of "OOPS". Hvr ~~..~~ this doesn't ~~specify~~ the order in which O^i cands are to be tried — it could be via OOPS, or AZ or any "Grammar" or "GA ~~..~~ Mut/cross scheme".

The "tree labeling" of 368.23 need not be a "tree": It's just a list of certain cands that have been tried, & it has to be in a form that makes the info useful for whatever Kind of srch. I decide on.

10 So in backtracking from i = n, say we go to all cands that are known to work on $Q_1 \cdots Q_{n-1}$ and have had a "time out" (~~which is~~ the value of which is ~~...~~ on the "list") we test them Using CB₀ (for ~~entire~~ time out cand. ... The list shows how much cc used on each cand thus far.)

.13 Idea, then, w. CB = CB₀ we test all cands that can do $Q_1 \cdots Q_{n-1}$ & have some "timeout" available for extension. If no success, we include " " " " $Q \cdots Q_{n-2}$ in the srch, we same CB = CB₀

9.26.03 SN It may well be that after TM has been "modifying" older O^i's to create new O^i cands — & doing this by Cut+inn only, that TM will invent ~~internal~~ functions that make it easy to modify O^i's in this way. — Unfortly, this is a kind of "Hand waving" Argt. — it would be well to ~~to~~ have good ideas as to what such functs ~~might~~ be & put them in as "primitives", or design the Ref mechanism around them.

.19 → we continue down to Q_1. If no success CB₀ ← ~~..~~ 2 CB₀ & try again.

.20 NB (In this (.13–.19) routine, we don't have any Backtrack thresholds for pc: Which seems
.21 Strange; since in Human Science we do. What's going on here!)

SN In one of my Papers ~~#~~ TB181 or Sol 86, I had a way of (my Alg notation) very small CJS ~~...~~ ~~...~~: I may have used "Logical reasoning" — Look at it!

.26 Will the model of 363.14-21 work if I make next cand an extension of the previous cand? Say I insert O^i, then Q_j. It asks for input; we add on α and it gives A_j, the right answer. ~~Next we put on Q_{j+1}~~ Starting w. O^i, we put in Q_{j+1}. No subseq. ~~...~~ insts give A_{j+1}; so we backtrack; In backtracking, we try extensions of O^{i-1}

30 I think the trouble is, If we try any extension of O^{i-1} to work new problems, then If Q_r is the input O^{i-1} was designed for, so w. Q_r as input to O^{i-1}, A_r was produced (after a single insts), then any trial that is an extension of O^{i-1} will tend to produce output w.o. asking for input - which ~~is not~~ means that output/part

.35 is of probability 1) (usually untrue (?)). → (370.00 Spec) →

SN Remember (Lsch) AZ has 2 kinds of inputs, also: ~~...~~ the initial input & the "regular" inputs. The regular inputs can contain pgm material. Presumably, all inputs are "universal".

370.00

00: 367.40 ; $\boxed{\text{ASN}}$ It would seem that (except for use in making $\underline{\underline{p}}d$'s) "oversearching" is counter productive: i.e. one is more economical by doing Backtracking: This assumes that one remembers all of the Tree search info nec'y to go back to do the backtrack. — Otherwise, w.o. that info, backtracking becomes more expensive, & extremely "oversearch" O^i's is more economical.

Is this true? : That if one has a known (say Trainer-given) pc threshold for acceptance of a O^i, then backtrack v.s. Lsrch is completely well-defined.

Well, for each value of i, one has to know a pc threshold.

Remember: we want a O^n w. th $\frac{pc}{cc}$ = max. (actually, we want pc = max, but....).

.10 ~~As soon~~ As soon as we find Q_{n+1} doesn't fit O^n, we go back and continue searching for new O^n's; looking for one that works for $Q_1 \cdots Q_n$ and Q_{n+1}. We do this srch w. a certain $\underline{CB_1}$.

If it is not successful, we go back and look for new O^{n-1}'s that also work for Q_n & Q_{n+1}. The $\underline{CB_2}$ for this Backtrack is unclear. If not successful we backtrack to find a new O^{n-2} that fits $Q_{n-2} \cdots Q_{n+1}$, using $\underline{CB_3}$. ←

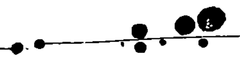
So: what are $CB_{1,2,3}\cdots$? Each time we go back, we have a larger set of conds to search; for a more restrictive (i.e. larger) corpus than that set had been originally srchd for. The size of this corpus, is always the same: i.e. $Q_1 \cdots Q_{n+1}$.

An alternative BackTrack method: → .10 Q_{n+1} doesn't fit O^n go back continue search for new O^n that fits $Q_{1..n}$, w. $CB = CB_0$ w. same CB_n.

20 as soon as one is found, try Q_{n+1} on it. If it doesn't fit go back to

If CB_n is "timed out"; look for a new O^{n-1} that fits $Q_1 \cdots Q_{n-1}$ using CB_0.

If found, search for



.23 At each pt. in the such, we have this tree w. nodes. The nodes we are at are integers from 1 to n. those in which one of the Q_i have "failed" (i.e. the pc backtrack threshold for $Q_1 \cdots Q_i$ has been exceeded. Also other nodes are labled as to 1) how far in the TSQ they will go, & how much time has been spent ~~[illegible]~~ currently testing for the next Q. — & to "failure": what level of failure? How low are the corpus dist'ns?

Thresholds

For time being, assume pc_i Back track Thresholds are o.k. (inviolateable)

Later, these are not so! (369.20-.21)!

.27 A possi. search strategy: Given the pc_i Backtrack thresholds (This tells TM when a O^i is "acceptable" for its corpus). Given a $CB = CB_0$; We test O^n on Q_{n+1}; if it doesn't "work" (w. this criterion) we expand all nodes that got to Q_{n+1} & had timed out for Q_n, and expand them w. new time CB_0.

30 — If we use up all Q_{n-1} nodes w.o. timeout goto Q_{n-2} nodes (w. new CB_0). Search the entire tree w. ~~[illegible]~~ $CB = CB_0$ by first doing Q_{n-1} nodes, then Q_{n-2}, etc. Do this for the whole tree.

If no soln. is found $CB_0 \leftarrow 2CB_0$ & do it again, etc. Double & redouble until soln. is found — ~~[illegible]~~

[Note: $CB = CB_0$ means timeouts is $\frac{cc_j}{pc_j} > CB_0$.] ← pc_j means total pc for that trial; cc_j " " " " " "

We do end up w. a kind of "locally maximal" $\frac{cc_j}{pc_j}$ (if we ever find one).

If we don't find one after a certain CB, we relax the pc_i backtrack thresholds (i.e. decrease them), and go thru most of the entire search again (since any failure is w.r.t. a higher threshold than we have now.

.37 Well, Keep .27-.37 in mind! It may be possi. to do it w.o. the Time-out pc_i's of .27. We just hunt for a O^n that will give a soln — and we end up w. a soln of ~~[illegible]~~ max $\frac{pc_j}{cc_j}$.

— But then, how do we decide when to Backtrack?

9.24.03
3PM

"Backtrack threshold" ≡ Criterion for Deciding to Revise Theory

.00; (see 366.40) : 366.29–.40 seems to resolve t. "2 aprys problem" of (363.21 R)

On t. other hand: lookat (363.14–.21L) — T. Thing that revealed t. trouble. T. Grammar can be used to order t. ~~t.~~ cand evals, (AnyThing can be used to order t. evals!) — But t. impt thing is t. final logit apry evaln (via AZ, say). T. estimated cc of soln. & t. CJS estimate will be wrong if Gramm ≢ AZ; but t. result will be O.K. for induction if we use AZ for t. final acceptance criterion for a cand, & we use some Backtrack threshold for pc —

.08 Given by trainer or some other criterion TM devises, perhaps in attempt to simulate Trainer-given

.09 ~~[illegible]~~ "Backtrack thresholds." → (See 369.04–.13, .19–.21 for what maybe V.G. Backtrack Algs)

10 Using a non apryd for search: We end up w. a soln that's not necy t. best pc/cc; We still have our external Backtrack threshold from trainer on (.08–.09)

It should be easy to compare t. "GA" pd. on t. O² with the AZ p.d. on t. O², since we have to compute both for each O². Presumably t. "GA" pc product is larger — or else we would not use it (?) (A possible reason ~~[illegible]~~ Maybe true, this is that "GA" could

.17 contain info good for srch, but not exactly t. same as AZ.)

18: (363.21 R) Th. Technique of (363.14–.21R) for ~~[illegible]~~ phase 1 srch, seems OK.

The initial "Ordering of t. O²" can be done by a Grammar w. corpus $Q_1 \ldots Q_n$ or by

20 using AZ. (366.29–367.17 discusses this; (367.00–.17) is particularly relevant).

~~"Both" to Grammar~~ The AZ final evaln. functs should use context, & t. Search Grammar would

.22 have to use "context" or equivt, if it is to deal w. "Scaling".

.22R Note:
361.35 : This is an impt Q:
T. discussion gave 1 poss. soln. (t. "R" recogn. funct of §1) — But t. continuation of t. Discussn. did not get anything that helpd; It wasn't clear how OOPS' continuation property would be usable in a 3TU.

OK So Back to t. Road Map of 361.00

Some Bottlenecking areas are 361.35–.40, and 367.22R
T. desirability of something like OOPS, is able to modify O² in various ways that retain its ability to solve $Q_1 \ldots Q_n$, yet give different final types of solns. for Q_{n+1}.
I will have to study t. R ~~[illegible]~~ Recogn. funct method of §1 again — also work on ways to make t. R's "Soft", s-functs (rather than 0 functs). Note that t. "R" system really does deal w. t. problem. .... Is it t. "Best" way? (Remember Phase 1 doesn't have to be "perfect" — But it does have to be good enuf to get to

30 "Phase 2"

9.25.03 SN Examine this process of Generating O² s.t. cc. v.s. t. distribution on O² obtaind via AZ.

Start out by considering (CB=0) for AZ (perfect induction). For each new i ∈ $(Q_1 \ldots Q_i)$, we get a progressively smaller set of [O²] that are consis w. that (augmented) corpus.

The actual set of O²'s ~~t.~~ has property that w. small i's, cc of ~~soln~~ O² solns are small & Incg ↑ w. i (say ≈ linearly?). So, while for small i, t. set of O² is very broad i.e. many O² w. hi pc that fit corpus, — T. Distribution over t. O² gets narrower as i ↑.

9.23.03

3 TM

.00 : 365.40 : IN [redacted] t. first case, which was, defines " Context" we look at $O^1, O^2 \cdots O^n$ & try to predict O^{n+1}. ~~It~~ is a seq. ~~concept~~ predn.

.02 In t. second case, we try to extrapolate A as a funct of Q for t. QA corpus: In constructing a cand. O^{n+1},

.03 The ~~frequency~~ (pc) of a defined concept will depend on ~~the~~ how many occurrences of it in O^{n+1} only

In t. first case, I think t. frequency of a defn. is over t. entire O^i seq.

Also " " ~~the~~ O^{n+1} will usually (but not always) be a relatively simple modifn of O^n.

→ I have worked on this problem many times: T. most recent one involved t. GA idea ↗and t. corpus $O^1 \cdots O^n$. .... but I don't think I ever realized that there were 2 Pd's involved & just how to justify or Understand that!

.10 N.B. Context has to actual modify t. final pc's; otherwise we get the "scaling effect", so cones cost more & more & problem solns cost more & more.

Re .00–.03: In t. first case we have to predict t. whole O^{n+1} T. parts that usually stay t. same (in t. "corpus") have hy pc in O^{n+1}. So a conc. appearing in all n of $O^1 \cdots O^n$ will ~~be~~ very likely in O^{n+1}.

In t. second case, t. fact that token Δ has been useful in many QA's is of import. If it was usd in 8 out of 10 cases; maybe give it a pc of $\frac{8+1}{10+2}$? (or more exact Lap's rule).

Note: Say we have O^n that has worked o.k. for $QA_{1 \cdots n}$. It doesn't work for Q_{n+1}, A_{n+1}, hvr. We could devise a new function, F(Q) that did give $F(Q_{n+1}) = A_{n+1}$, & why pct ~~[illegible]~~ for $i \leq n$, F works occasionally, but not great. So we use a problistc mix of [redacted]

$$O^{n+1} = a\,O^n + (1-a)\,F .$$

In t. second case, t. fact that Token Δ, has been useful for Many Q's, is mostly important.

rnktol.com
Talk1000
1000min ½¢ min

~~From~~ Yahoo!

Perhaps write Road map: listing parts that are unclear &/o Need most work.

Bot write up 1st clearly:

Then do Biblio review of last yr. or so of TM. Go thru, listing impt. topics & cross refs,

Make list of ~~at~~ impt. ideas in o. of order of importance. (some will be ⩳),

Re read v 365.00 ff: Hvr, first, try to get "soln" to problem on 365.01 ("2 ~~people~~")

At this time, I thot I solvd t. problem! — later, I didn't ~~see~~ know how to solve it — See 391.05–.40 for impt. ideas in later approach

.29 9.24.03 Back to "2 kinds of aprx" (363.21R) It seems clear that ("GA") case 1 (.00) can contain legit aprx components, like "context" — but it can also contain "Quick Abort", which is not legit.! " (≠ case 2) (case 2 ≡ (.02–.03))

.30 ~~See~~ In Phase 1 (≠ case 1), we don't have to use all heurs possl" ← Phase 2 can do that, — just use legit aprx, like "context" The O^n corpus can be usd as a heuristic source of possl. regys, (case 1 presumably, (.00)) but they have to be tested viz coding of t. QA corpus (Or some way of conversion so that regy is known to be Legit

.36

.37 On t. other hand, in Phase 1, we don't need perfect induction" so we mite use case 1 "regys": Just how bad those illegit regys can be & still not screw up TM, is unclear. ~~[illegible]~~ ~~up TM, is unclear.~~

→ It would be well to make list of t. similarities & contrasts betw. regys. contained in t. 2 kinds of Pd spec

→ 367.00

9.23.03

BTM

.00: 364.40 to be discovered via which/~~fnels~~ cons.

.01 (363.38 spec) → The [QA]$_T$ corpus & the [O^i]$_i$ corpuse are defining goals. [O^i] involves how to obtain more + more -less acceptable codes for [QA]. These codes are not/normally optimum either in pc or in cjs.
363.38

But I think the [O^i] corpus also contains Logit reps. of the [QA] corpus,

The O^i corpus is [redacted] obtainable from [QA] corpus if the order of QA corpus is given. "Obtainable" if one is given the heuristics for finding the O^i from the [QA] sequence, ~~&~~ — and its ordering.

To what extent is the [O^i] corpus a code for the [QA] corpus? Well O^n, the final O^i is a code for the entire corpus. — Anything that ↑ pc of that O^n does ↑ pc of coding the corpus — But we are using info in ordering of the QA's. Hvr, is there any info in the O^i corpus that is not available by looking at "useful subfunctions" in O^n, (the latest O^i only)? .... There is the sequential info of [O^i] corpus.
.10

.11 Note that this "Grammar" method of using reps in the O^i corpus to find fast, good reps ~~[illegible]~~ in the QA corpus does seem to be what I (& probably other people do).
.12

So it's clear that I really have to understand .11–.12 before I can use it w. confidence — & before I can really optimize it or even set its params. in a useful way!

Note that the O^i corpus is being used to find good O^i, at once. "Good" is w.r.t. the original "a prop(?)".

.17 I'd guess that the Gore of the "Grammar" is to set asly a ln(pc)/as possl, for ~~each~~ QA corpus per cc expanded. (IS ≈ the Gore for WON) — But it's narrower in its methods than WON is. The Grammar only ~~[illegible]~~ modifies pc's of Tokens ~~BS~~ WON can ~~[illegible]~~ create any PST to solve the induction problems (I'm not sure this is more powerful than modifying pc's of tokens, hvr!).
.20

I'm not making any headway on this! Try defining the problem clearly:

~~B~~ Top goal is maxm of ~~$\sum$~~ $\sum_i$ ~~π~~ $\prod_{j=1}^{n} O^i(A_j|Q_j)$: "Best code for Corpus"

We have, in the past many examples of ~~[illegible]~~ Pairs, ⟨O^i's & associated corpi, ($[A_j|Q_j]_j^i$)⟩ would help here!

~~Is there~~ Could we, by studying that set of pairs, induce a ~~[illegible]~~ good QA corpus → O^i function — say an S-function ~~[illegible]~~ ?

← [This was a "Top Goal" in some earlier work on ~~[illegible]~~ QATM ← It still may be a v.g. "Top Goal". A criticism of it is that it is "a phase I methodology" i.e. Try things that are close to what worked before ... it has no concept of optimization, or even improvement!] ←

→ A simple example is a simple "Context". We observe from past codings that the ~~[illegible]~~ token ~~Δ~~ Δ, say becomes more likely in a clearing problem (or If the previous token was *), ~~[illegible]~~

.30

~SN On sample size for various cones: The TSQ could ~~encourage~~ encourage TM to give more hy pcs to certain cones by repeating their appearance in the corpus. Repeating the same (QA) K times would have no effect in MTM ... Maybe effect in NMTM? — ~~[illegible]~~ probably. but then it would be best to vary the A's for that Q: in line w. what I feel the true proby distribn. is.

I think the difference betw. a context & normal modifns of pc's by defin. & repetition, is that context uses the O^1, O^2 ... O^n corpus. This does not. [The "corpus" is the [QA]$_{set}^{pair}$

9.23.03
3TM

Infinity Unbound!

00 : 363.40 : For me writ'g this up in very clear way, so, reading it later, I can quickly get "Up to Speed" on t. main problems/diffs/~~bottlenecks~~.

Also do some biblio review of recent ideas on TSQ's. I think TSQ's shouldn't e.g. be such a diff'lt problem!

SN A real possy! That Designing TSQ's is not such a big problem (except perhaps, at highest level, to get TM to acquire very "creative" heurs) e.g. one idea: write a TSQ for Humans, that doesn't use concs that Humans have special access to: It should be possl. to find suitable heurs that TM can lrn, perhaps by broad "hints" or by ~~more~~ "wiring in".

I'm not sure ~~how~~ of t. importance of t. problem of "2 apples" (363.21b–.40): It may be that I've more or less solved it, but can't remember soln (like t. celebrated footnote in S89!).

.10 [If an idea is "fairly obvious but diff't to explain", so "no need to write it up", then it is Not Obvious and should be written up clearly]

.12 SN on t. S89 footnote! #2 That Blind srch can simulate non-blind heurs.

I Guess "Blind" meant one didn't look at "why" t. trials were successful or faild. Also (perhaps) that one didn't remember anything about past trials (???). ← I guess this is implied by — If one didn't know "why" past trials faild, ~~no reason to rem~~ nothing gaind by remembering them.

Hvr. by allowing "Concs" that TM can acquire, to be able (as arguements) to include info about previous trials (or any other of TM's activities), T. trials are no longer "Blind" tho in some sense, they may "look" like a Blind srch. I guess because choice is only guided by Pc assignd to cands (& as secondary factor, t. cc of each cand).

.20

This may be what I was thinking about when I wrote that foot note. So t. ~~F~~ "FootNote" may actually be "correct", but t. "meaning" of "Blind srch" is "distorted" "mangled" a bit! ☺.

The conditional "Suff't powerful set of Concs" means ~~that~~ that they can look at previous trials ... why they faild, why they almost succeeded, & trials for other problems that did succeed.

Perhaps it is not "diff't to show" that any legit heur is of this form. — it includes

.30 any info that ~~can~~ could have been obtaind by looking at past trials ... successes or failures. ~~How~~ Legit heurs are obtaind in only 2 ways ① statistical study of past ~~su~~ successes, failures of trial ② Logical reasoning.

I may not ~~be~~ have been ~~thinking~~ considering "Logical Reasoning" when I wrote that FN, but it certainly is a powerful source of Legit. heurs. So, if we allow pc's of cands to depend on entire past history of testing + "a prio" + logical reasoning, Then we can include all heurs (I guess ... maybe even "Quick Abort"!). However a search based on such a P.D is no longer Lsrch, because t. pc of a trial depends on ~~t.~~ results of previous trials. — So t. Optimality thm(s) about Lsrch & CJS ~~...~~ bound, do not hold (i.e. we can't compute CJs of a soln in advance of t. search! — Tho we mite be able to estimate it if we knew which concs need

3TM

00:362.40 Insert Q_7, ~~then~~ try various pgms to write various output A's.

So back to just ~~Q~~ for Q_7, insert Q_8.

But if we put in an arby Q_j, how is this handled? We have long S, There is a good chance that TM will ask for no more input & go on output (∴ pc=1). If we put in any of the previous Q's that it works w. it/will ask for no more input & will print out an output (pc=1) Bad!

I really don't see how I could get it to work in any simple way!

Actually, the OOPS system really wasn't designed w. anything like this in mind! The "additional code" was meant to bring it to a new system which it would, for several Q's, have outputs w. no new needed code. I was a MTM system.

How to modify it for NMTM is certainly not clear!

.10 Perhaps it would do no harm to restrict our system to one in which it reads S first, ~~then~~ but as soon as the ~~a~~ Part of Q is read, then we can't ~~ask for any~~ request more code for S: ~~Any~~ subsequent need for code is R. We put R input in PC order, so we will be able to guess a pc for the output ~~whenever~~ find it.

→ 367.18 spec

.14 So say we have S trials listed in pc order (via our "Grammar", apriori, GA input/codes

.15 Scheme, or Whatever.

We put S into the machine, along with ~~a~~ a Q_i.. It may or may not ask for more ~~R~~ instructions, but anything it asks for, ~~sees~~ before it "finishes output A" is regarded as "R".

366.29-.40 seems to Resolve this

.20 We can use J's OOPS/stack machine. The pc's assoc. w. the trials ~~are~~ are obtained via Lsrch rule,

.21 but the ordering of the trials is via .14-.15. → So this seems to be assoc. tg 2 pc's w. each cand. ("21R") Sounds very Weird!

One is apriori, the other is a pc for Lsrch based on history of ~~the~~ past trial(s)!

Lsrch $\hat{P}$ | Grammar pc $\hat{P}$

Hvr, the Basic Idea of Using Lsrch for induction, is that ~~[illegible]~~ used in a natural way, one tries cands in apriori order, so we tend to find codes of the highest pc. If we use a grammar to order the trials, we [redacted] have no assurance of obtaining ~~good~~ very good ~~codes~~ codes.

A possl justifn of using the "Grammar" is that it gives a kind of "apriori" ~~[illegible]~~ & that perhaps its OK to try to Make it for our corpus.

In my early work on "Context", I think I started around ~~[illegible]~~ this area: thinking about things like (.21R), but maybe not exactly.

.14-.21L
is this "soln" I'm using for the S function QATM (NMTM)
The Grammar may include "Context" ideas.
The Backlock cnlarma's a Secondlist of a problem.

.30 Hmm! Is the Sequence of Good O's a form of a kind of regy in the corpus? It does include Sequencial info, hvr. I may want to allow ~~seqnn~~ some sequencial info: The TSQ idea exploits Sequencial regys to an impt. extent. The Grammar is a kind of regularity in the regularities

— So its a logic regularity! I have to clarify this!.

The [O's] seq (or set) that is the corpus for the Grammar, certainly doesn't include lots

.38 of impt. regularities! → (365.01 spec)

→ Try to write a summary of the problem! Defaults soln(S).

Then make a more detaild drn. of the road map w. indications of Major "bottlenecks" (if any) → 364.00

00: (Spec 361.40) : J's "OOPS" formalism of "trying to get [redacted] Q^{i+1} by "continuing" Q^i is an attempt to deal with this. Is it a "good" way? Is it t. most "Convenient way"?? What is t. most convenient way?

One way that seems a bit Gross: The QA's are "sequential" in a certain sense; So t. O^i for each new QA is "difrnt" because "i" (t. "time" index) is difrnt.

Certain problem types solns are time dependent — others time invariant

But (anyway) drop this (softly) for awhile: But do consider t. idea of 361.35–40 again.

We do seem to have "R" functs that put problems in a particular "Area" w. a common function that solves all of t. probs in that Area (Recog. equivclass).

In Retrospect I had kind of avoided thinking about that part of QATM — But it is an important part & a not bad step toward solving 361.35ff.

In t. report, I treated logical (Yes/no) R's only, but I think I wrote a bit about "soft" R functions (≈ fuzzy sets — Gray categorizations of categorizers),

One reason to use d-R functs was that I was (perhaps) mainly thinking of MTM (d-Predn) — OOPS deals w. R's I think by having functions that decide themselves, which argument to "accept" (i.e. no output at all for certain inputs).

Hvr, I'd like t. R's to be more "visible", not hidden inside some predn function.

The Unhiddenness means that t. assoc. predn functions P, can be updated on a small subcorpus.

I vaguely remember this use of functions that determine their own range, as being good for MTM only. (see 349.29 – 350.15)

Hvr. in §1 on QATM I was using logical "d-funct" R's to assoc set of functions, $[P^i]$.

In t. case of OOPS, t. situation seems a bit difrnt: T. functs it implements can do 3 things: 1) present no output to any input: i.e. stop or ∞ loop w.o. printing.

.27 2) ask for more pgm input — after which it may or may not print.

3) print output (w.o. asking for more pgm input) & then stop (well, stopping is unnecessary in certain kinds of problems. — But we may want output to be "unerasable" by TM.

OOPS uses .27 (2) to "extend" S: I might use it to put in R.

Ah! Big thing! In t. normal OOPS! we put in new Q, it asks for more input: whatever extra input it needs to get correct (or any other) Ai t. pc. of that Ai. We insert a new Q, but only after, we have inserted augmentations (≈ R) that gave t. correct Ai for previous problems (In general there will be several such augmentations — we start by picking t. shortest one. Any other augtns can be used (if needed) for Backtracking.

Say we do $Q_{7,8,9}$ & they all give outputs w.o. asking for more inpts. Then to get t. P.D. of various outputs for each $Q_{7,8,9}$: Go back to end of Q_6 pgm input.

61
245
37?

.00 : The Road Map: Its principle Milestones — Markers of Progress.

[Much of this will be ~~a~~ maybe all section of t. report. But I also want it for my own use: A clear statement of what's been done and what needs to be done.

An outline of this: **Phase 1:** This consists of first QATM (MTM d-funct solns) and then (NMTM s-funct solns). ~~[illegible]~~ Adaptive Lsrch

.09 is usd to find solns. Discuss creation of s-Grammar for $O^1 \ldots O^{i-1}$ to find cands for O^i. [Various forms of PD or "s-Grammar".

.10 → Next part of Phase 1; INV & OZ problems: Given a ~~site~~ set of PST's

.11 by trainer: to use them to solve these probs by Lsrch. ~~Note to expand the set of PST's by making a Grammar for them.~~ is this part of Phase 2?

~~Next Phase 2~~ In .10 & .11 (INV & OZ probs.) we start out by using t. universal ~~[illegible]~~ d.f. on functions to put a d.f. ~~on~~ (~~Grammar~~ Grammar) on PST's.

We have separate Grammars ~~for~~ (PD's) for OZ & INV problems,

.20 ~~[illegible]~~ The ~~[illegible]~~ Grammar becomes "Adaptive" ~~[illegible]~~ in a way same as .09: We use t. previously successful set of PST's as a corpus to define a Grammar. This can be done by usual formal Grammar methods a/o Generalized Mutation (crossover idea— (or other methods) The idea of t. "Grammar": to list cands in pc order! [Also give pc's! to enable Lsrch. In QATM t. cands were O's; In INV & OZ, t. cands are PST's.

I Guess we're then ready for Phase 2. We have a set of PST's & problems they have solved — & ~~what cost of~~ cc of each soln. from this we can get h c j functions. & Run WON srch & update during Won srch,

Maybe do it this way → { It's possible to start out w. Won & GHT1 search & no update during srch; then ~~graduate~~ to "update during srch".

.30 Also modify sections 3.1 & 3.1 so GHT1 is written first.

Perhaps discuss details of Lsrch for 2IU v.s. 3IU problems &
" " ~~[illegible]~~ randomish choice of Q's for O^i evaln. (statistical Generaln)
also finding hardest Q's in corpus for O^i evaln (for Quicksort).

.35 [3.21.03] → On t. other hand, there seems to be a characteristic of usual "normal" Human problem solving, that when one searches for a soln. to a new problem, one normally doesn't have to worry about t. soln. still fitting old problems! How does this work? → Note Band 367.22R

The Recognition algms (R) is §1. seemed to help ~~[illegible]~~ with this. → Spec 362.00

This R system was designed to be like t. way humans often deal w. this problem.

00:359.40 : On t. mechanics of L srch for ZIU: OOPS does it one way.

As always, t. pc assigned to a pgm, S, [redacted] depends on t. [redacted] amount of t. pgm ~~that~~ that was used, when its output A is finished. We want max length needed for ~~any~~ all of t. Qi.

Actually, t. way OOPS does it seems v.g. Qi is already in t. machine; we put in tokens (I guess it always asks for them) until it prints an output and stops or until it prints t. desired Ai (~~there~~ Ai ~~may or may not~~ may or may not require that it stop. — it need not stop if we have a ~~semi~~ semi infinite Ai string (~~[illegible]~~ like a seq. to be predicted)

With a new input, Q it runs until it asks for ~~[illegible]~~ S input or until it prints desired output a/o stops or if it prints an output that cannot be correct (i.e. it disagrees with a symbol w. t. desired output.

10

How do we modify this for ZIU?

Well, we could have 2 distinct places to get instructions: S and R. In this case, it would be poss. for TM to ~~try~~ try "continuing" an old S to get a hy pc "A" output — so it could minimize t. length of R for ~~[illegible]~~ t. existant Q.

The IPs for S and R ~~are~~ are individually ~~[illegible]~~ (settable changeable). At each time we have Smx and Rmx which tell how many insts in S and R thus far. The 2 inst. pointers can be set from 0 to Smx+1 and 0 to Rmx+1, resp.

Instruction pointers

20 We may want to ~~have~~ have default ~~[illegible]~~ instructions for S and R (?) — Certainly poss. to do — but I don't see any Advantage Gained;

We also need instructions to tell TM whether to go to S ~~[illegible]~~ or to R for next inst!

IMPT point

It may be that I'll want to wait until I have a TSQ and I ~~[illegible]~~ have ideas on just how to expect TM to compute pc's!

Another possl formalism ~~for~~ ZIU: It reads S until it ~~[illegible]~~ gets to "Stops S" and "Go to R" state. Then it switches t. Inst. pointer to t. first inst of R.

.27 t. Rush t. "spirit of ~~it~~" is that S says stop, then R goes on to tell exactly how to construct Ai.

My [redacted] own intellectual Soln. to t. problem presented to TM should be expressable in ~~[illegible]~~ (.27) form.

30 A Q that I'm still uncertain about; whether t. OOPS idea of "continuation" is a good idea. Superficially, it would seem that t. continuation method of mech. of a ~~Emdr, we~~ O^2 cand would work in only certain carefully constructed TSQs. That in general ~~[illegible]~~ t. $O^k \to O^{n+1}$ would be a very general mutation process, $\bar{s}$ Grammar (~~[illegible]~~ df) construction process.

9.19.03
3 PM

Also Note 357.32–.40 is "basic routine for ~~Adaptive~~ 3 IU "Lsrch"

(See 358.2)

.00 : 358.40 → So (in view of 358.31) 358.25–.26 may be good Enuf for phase I. T. trainer would perhaps have to give the average (ie. ..) threshold for backtracking — But this "average" should only be used when t. corpus to be coded is fairly long.

9.20.03 $N Re: OOPS: How much memory of token frequencies is carried over from previously successful trials? My (strong impressions)

.06 1) when working on T.O.H., say for n = 3; It will do 2 modes of srch: ~~[illegible]~~ One will have no memory of n = 1,2 solns & will start ab initio to try to get solns for 1, 2, 3. The other will use the successful pgm for n = 1,2 & its token frequencies, & try to "Extend" it — i.e. add on bits if & when required.

10 2) when working on T.O.H. it will not remember token frequencies from Grammar problem (other than via "frozen" data bank … thru (usually) "boosting")

Remark! In .06, it would seem very wasteful to start ab initio, since a great no. of cands are known not to work w. T.O.H. n = 1,2 It would be best if t. tree structure for t. soln. n = 1,2, were retained, so (w. this info) notes on which branches failed, which had "timeouts" at what time levels). There is possibly some mention in t. paper ~~[illegible]~~ about retaining "freezes" of trials ~~[illegible]~~ i.e. srch trees — I thot they mite be usable for ~~[illegible]~~ parallel Lsrch — But I'm not sure of whether OOPS did, indeed, (1) ever have (2) retain; that info.

20 **So I'm** still not sure I understand t. details of how OOPS works! (see .29 for a clue!)

However the presently successful Tower of Hanoi T.O.H. 1,2 pgm is a very special pgm on t. tree: it's (usually) a soln to 1, that was extended to get a soln. to 2 ~~[illegible]~~ So perhaps t. failures in t. srch for it wouldn't cover much of t. srch tree, (if this is true) I.E., t. srch for t. (1,2,3 solns) would have to take much longer say a (factor of several, longer) than t. srch for (1,2 soln.)!

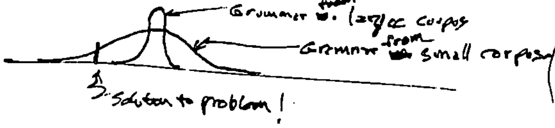
On t. other hand, J tends to be dismissive of a factor of 2 or of 10, for that Matter!

.29 Its possl. (Maybe even likely?), that when OOPS trys to find soln to TOH 3 in "Ab initio" Mode, it doesn't even

30 —— bother w. trying to find a pgm that solves 1,2 as well! — It ~~[illegible]~~ trys to solve 3 only.

.31 : 357.40 ! T. method of doing 3 IU that I had in mind, was 357.32–.40 when we look for soln to $(QA)_{1,n+1}$ we use as corpus for t. lang. of which Q_{n+1} is to be part (i.e. extrapolation) $O^i_{1/n}$. Presumably, OOPS is a "special case" — a particular kind of Grammar for this corpus.

Presumably, when we backtrack, we go to a Grammar (O' extrapolation) based on a smaller corpus, so it is more "general", less restrictive — More likely to have rite answer — but perhaps still of low pc!



NB: If large corpus Grammar is "very weak" pc's are not very "sharply peaked" then Backtracking will not be as effective. (But unbacktracked Grammar ("very weak") will not be v.g. ("sharp")).

00: (Spec) 357.40 : For each search technique, we will have to devise a TSQ that will allow that technique to be successful. !

Some more different ways to "slowly zero in on t. final soln": We have 1. continuous corpus

$[Q_iA_i]_{i=1}^{n}$ We try to find a function of Q's that / for each Q_i divides A_i space into 2 (or maybe more) parts — a "good" part & a "bad" part. We try to get it so that t. no. of "true" A_i that are in t. Good part is Max. (This is a categorizn. problem so 3ID maybe & very likely SVM can be used.) Hvr, as (derbd / stated) This will not work! A categorization that accepts all A_i would get max score! T. corpus has no "negative cases".

.10 Hvr, I did have a related idea! That in many QA situations, there are many A's that are equally good (say variations in phrasing, grammar, etc to answer a Q in English). This results in any one of them getting a very low pc — But if one doesn't realize what's going on, one will

.15 think that a low pc for t. answer means "failure" & need ~~need~~ to Backtrack (≡ theory revision)

.15 sounds serious! One wouldn't know when to (revise theory / backtrack / decide present model is inadequate). When I last thot about criterion for Backtracking, I thot scientists ~~[illegible]~~ learned how to decide: — but often t. sci. community as a whole was very uncertain.

Hvr, this seems diffnt: it seems that Humans get feedback on t. degree of goodness or badness of a

.20 reply (as in RTM.... which I'd like to avoid!).

Well say ~~[illegible]~~ TM spent an "equal amount of cc" on each Q_iA_i ; — But it's not clear as to when this is Being Done! TM normally optimizes a set of ~~[illegible]~~ QA pc's. So perhaps

that's it! TM optimizes a O' for $QA]_{1-5}$ say then goes on to $QA]_{1-6}$.

.25 → T. "backtrack" Q is: "When is ~~[illegible]~~ O'_{1-5} (so bad that / not good enuf so) ~~[illegible]~~ O'_{1-6} has to be revised?"

~~.25~~ 26 Since t. Q of .25 covers several QA's we mite expect a certain ~~[illegible]~~ "average impct"

~~.26~~ .27 ~~[illegible]~~ For a not very advanced TM, .25 may be as good as we can do. ← Note .31 → 359.00

On a higher level, perhaps TM mite want a search method that over the

.30 "long haul" gives t. highest mean impr. per cc.

.31 Remember ~~[illegible]~~ "Phase 1 QATM" (Doesn't Have / isn't supposed) to be terribly smart! — it only has to be good enuf to get Phase 2 update working. — It's really not expected to approach Phase 2 ... it just has to be "Good enuf to get off t. ground".

Full English could be bad wrt. .10-.15 (many alternative solns). However, at first, the English being used will not be ~~[illegible]~~ complex enuf for that. Also, TM will assign greater pc to t. "shortest" version of an answer / I.A. Richards 400 words?

SN How complex is "BASIC ENGLISH"? It has a small vocabulary — but is t. grammar simple & unambiguous? "Loglan" maybe unambiguous, but vocab is large.

9.19.03
3TM

00 : 356.40 : See if I can use other existing induction systems for parts of ~~[illegible]~~ T.M project. for categorizn.

E.g. Bayesian ~~Belief~~ nets (BN), GA, ANN, Decision Trees, S.V.M's

(S.V.M's Decision trees) are used for categorization Probs — which is a kind of ~~[illegible]~~ QA problem.

Can I Generalize them to work more General QA problems?

Re BBN's: Recently, I came to a good, simple understanding of them: — which I've forgotten! See if I can remember, & Do write it down.

Similarly w. Max Entropy / Max cross Entropy ← (This maybe from Chris' stuff

For Max Entropy: via ALP: If ~~there~~ there are n, unknown pc's, [p_i] i=1/n : known constraints betw. the p_i:

then for ssz = ∞ (L = ssz) The best code (~~[illegible]~~ ≡ assignment of pc's to the [p_i])

10 is when $\prod p_i^{p_i \cdot L}$ = max (L is total ssz) i.e. ~~[illegible]~~ $(\prod_i p_i^{p_i})^L$ = max.

so $\sum p_i \ln p_i$ = max, subject to constraints

I then thot I found a bug in this argt! What was it?

Now: what was the Min cross entropy idea? $\sum p_i \ln(\frac{p_i}{q_i})$ = max, w. q_i known? w/ known constraints on p_i?

I don't know what q_i are.

BBN's: Discrete vector input : probabilistic discrete vector output ∃ ~~[illegible]~~ vector s-operator.

If input vector has n_i possible values for i^th component, there are $\prod n_i$ input configs.

each can give a different D.F. on each component of the output vector. These are very high dim.

20 Spaces & can't be learned with any reasonable ssz.

They simplify problem by assuming the vector s operator is composed of a net of smaller vector s-ops. The internal values of vectors may or may not be "hidden".

In not hidden, they can be "lrn'd" rapidly.

Anyway, I'll have to read about these.

That (Duda, Hart, Stork) Patt. recog. book will have suggestions on what kinds of models to use for induction in various parts of "Alpha" (≡TM).

The idea of the set of "Milestones": They are (I think) achievable sub-goals —

30 That they can be achieved w.o. any (or very little) Backtracking (≡ modification of previous Goals)

So ~~[illegible]~~, if true, this set of Milestones is an achievement!

.32 [3NS] Thinking about solving 2IU then 3IU: Perhaps not so easy! I think the idea of 3IU search was to first find a reasonable soln for ~~[illegible]~~ Q_1A_1. Then, using the (updated modified) "grammar" (≡ apriori), Look for solns to both ~~[illegible]~~ $(QA)_{1,2}$, Then (update modify) grammar & Look for solns to $[QA]_{1,2,3}$ etc. In each case, when we are looking for a soln, w. a certain corpus, the Gen'c is quite clear. A problem is to devise a TSQ so that this search technique will work!

(PAF 359.31 spec)

see (358.00)

3TM

00:355.40 : Soi wrt 2IU 354.18–.40 + 355.15–.19 is main idea/approach: To "fill it in" a bit more;

3 General kinds of Criticism of 2IU update schemes:

1) T. technique is slower (in real Time) ≡ cc. than it could be.

2) For reasonable CB, it isn't able to discover certain imp. regys.

3) T. technique is "slow" in sense of needing much ssz for imp. regy discovery

re: 3 (For CB=∞, 78T3 tells necy limitation on 3) – if we apriori, think a regy is unlikely, it will need more ssz. to discover it.

.09 (1)(2) are somewhat related. For finite CB, there must be regys "invisible" to t. system.

.10 w. this necy limitation of .09, an induction technique may be simply inefficient of cc;

Neural nets & most GA mite possibly have this deficiency."

[N.B. when I speak of cc, parallel processing usually doesn't help. cc is computing cost & "time" if we use many parallel processors]

.16 From a practical stand-point, when I write TSQ's I will (presumably) have good idea (or actually know) what t. imp. regys are. In this case I will see what kinds of Grammars (≡ PD's on unordered set of finite objects) are how good, & how th. Grammars need to be revised to "catch" t. kinds of regys that are present.

.19

.20 For **T. Report**: Either just my own version, or for t. actual report:

Write t. kind of detailed "road map", sort of "mile/stones" that I now envision for t. entire TM project. Phase 1(a) 2IU; Phase 1(b) 3IU; Phase 1(c) ≡ Use of Lsrch for OZ, INV probs: Building up of a good set of PST's;

AZ does 2IU, but maybe recursive funcs not so good.

Phase 2 ≡ Use of more rapid Lsrch, Building of Grammar (or equivt p.D.) of PST's — that extrapolate PST's in a "universal" way (limited only by CB).

This mite be done in Phase 1(c) as well.

Where English lrng fits in, is unclear: It requires at least 3IU, hvr (S-functs), So it can be tried any time after S-functs are "enabled", and it has lrnd enuf for us to be able to "discuss" what it has lrnd.
.30

In conclusion, it seems that t. (early) part that's in worst shape is "O² search", for orbed 2IU or (t. problem for 3IU is solved similarly but it has additional complexity; (See 355.20–.30 on this last.). "O² srch" need a way to probability order candt O²'s.

Also "Contents" is in Bad Shape — Not so BAD, but

354.18–.40 + 355.15–.19 are very "Weak": Not much done on that problem. It also is worked on while TSQ's are designed.

A "Grammar" is one way (by defn. a grammar is a P.D. on unordered BAG of data) — important case where there is no repetitions is a "SET".

.38 (355.15– ~~355.19~~) This Grammar (or equivt) should be designed along w. t. TSQ design, (.16–.19).

SN I was considering a Grammar w. t. O² Corpus being an unordered set. It mite be better to assume the O² are an ordered Sequence, & use Sequential predn.! Sequential predn may be easier (partly because Error Correction is easier (?)) — But seql. predn may be more appropriate!

3TM

00: 354.40 : But ~~[illegible]~~ t. essential Bijo Q's about backtracking & updating seem pretty much same for 2IU & 3IU. (Note 352.09 on 138.20 : This does suggest a way to go from a soln. to 2IU to a soln. for 3IU. It gives a kind of such strategy — I think part of it ~~[illegible]~~ May be OK.)

So 353.16 is right : work on 2IU for now. Mainly I'm considering 354.18 → .40

354.18 – .40 covers pretty much what I've been thinking about. Is there any way I could mix in some ideas from t. OOPS model? His idea that ~~[illegible]~~ t. pc of each token was to be computed by t. system itself: sounds like "Context", but (more specific specified in more detail?) possibly more recursive.

As is, t. system (vaguely) describd in 354.18 – .40 mite be workable, but it certainly needs

10 more work : Particularly t. "Context" part ← (Tho I expected to put this in when I analyse my own problems solving).

Perhaps try to write up 354.18 – .40 in as much detail as possibl. I think t. same "grammar" idea is applicable to generation & lrng. of PD's on PST's.

.18: spec: 354.40 3.18.03 So essentially, I've divided t. 3IU TM into 2 phases: Phase (a) is 2IUTM, Phase (b) is 3IUTM! In phase (a) we get O^i's in pc order. One definition of this is:

~~Def~~ $pc(O^{x})$ = $pc\,[O^i]_{1,x}$ / $Pc\,[IO^i]_{1,x-1}$ These pc's involve inventing Grammars

.19 for unorderd sets of finite (objects strings). : Can we use Same Ordering formalism in Phase (b)? (3IU)? I would think so.

this is an using O^i as a sequential "pred'n" problem! → 356.38

20 ~~Each~~ After we're able to do .18 – .19 w. some facility, we go to 3IU TM : We get. O^i's listed in ≈ pc order of approx., Then we have to do tests on t. Corpus. This can be done statistically. We will try to find $Q_i A_j$'s that are ~~particularly~~ particularly diffly so we can quickly abort trials. Also some kind of strategy for jumping from one O^i, Q_j trial to another $O^k Q_m$ trial, on basis of "pc thus far" ← includes "pc's thus far" of corpus (in 2IU, pc of corpus is always 0 or 1 only)

138.20 : ~~[illegible]~~ errors has what look like useful ideas on this.

In 3IU, in doing Lsrch for ~~[illegible]~~ R for particular QA, we will always have a ≈ lower bnd on t. "length" of the R needed: it's (≈) t. longest R tried thus far — we try them in (≈) order of R length (actually R + log₂ cc order)

30 Anyway, in 3UI Given t. PD on t. space of t. O^i, ← 3 & 2UI "Grammar". t. problem of finding t. one in a good "3IU Core" is somewhat "well defined"! Seems "solvable" or "not bad approximationable" — can be pretty solvd w.o. any

.32 extra epistemological analysis.

So, at present, t. main problems ~~[illegible]~~ in TM seem to be 2IU {i.e. search for O^i for 2IU.} & TSQ design.

I want to do 2IU for a variety of TSQ types, so more likely to be relevant to 3IU problem!

Maybe work on 2IU tsqs a little, then look at 3IU tsq's to get idea of what 2IU TSQ's mite help. TSQ's

3TM

.00: 353.40: e.g. [redacted] doing [redacted] Q = n → A = $0^{(n)}1^{(n)}$. ($0^{(n)}1^{(n)}$

or Q = n, [redacted] for n ring Tower of Hanoi as output.

for [redacted] n=1, 2, 3 .... He could find an extension of S that would fit:

Tho this did not (in either case) lead to recursive solns.

That O^{k+1} should be a "small" modifn of O^k is reasonable, but that t. modifn. should always be in t. form of an extension seems unlikely. Hvr, sometimes an extension is a reasonable trial.

My impression is that t. general idea of Mutation or "General cross-over" is more reasonable.

I had t. idea that something like this might be possl. in "Sumacs", because it did work exactly rite for CB = ∞.

10 Hvr, this idea that t. Q itself would help decide how much of S it needed .... Seemed like a very Attractive idea! .... Maybe Too attractive! ☺.

In Sumacs, backtracking was essential, & t. Q was, could one do o.k. [redacted] w. a limited amt. of Backtracking? Say almost always < "3" or "4" or so.... levels,

.18 In a [redacted] corpus presented Sequentially (i.e. a TSQ), A Q. is: "How much" & in what form, is [redacted] info carried from O^k to O^{k+1}? T. GA "Mutation" idea (w. perhaps also "Context") seems fairly General. T. idea of a "Grammar" summarizing

20 t. past — [illegible] softly limiting t. srch for O^{k+1} by t. stochastic (soft) rules of t. Grammar. T. "Grammar" could just be t. set of primitives & sub-functs (Macros) — i.e. "AZ" = Augmented ZMI — that have been found useful in t. past; Then a Bern d.f. on them, conditioned (necessarily) by General "Context". (I really don't yet know if "context" is good enuf to keep t. CJS bounded to usable level)

Tho one can (& usually does) retain previous entire O^k's as "tokens" in AZ, I'm not sure this has enuf of t. idea of preferring "small Modifns" of t. previous working O^{k+1}.

30 It may be possl. to express small changes in O^{k-1} as a kind of "OSL" — i.e. by defining things that have been (thus far) used only once. This could give a Conditional pc of O^k trials in terms of O^{k-1} (info distance, "implication complexity"....) → see 355.15–18 for definition. This seems closer to "mutation". Perhaps in making a "Grammar", t. most recent O^i's, should be given more wt. as part of t. "Corpus" that this grammar is trying to describe. The decay of wt. w. distance .... this λ can be initially selected by Trainer's spec. & slowly optimized over t. life of TM. → $\boxed{355.15-19}$ is a good continuation of this Thread.

This is ≈ t. "recency" "Context...." "recency"

spec (355.15)

355.00

3.17.03

3TM

00:352.40: I want a picture of how NMTM (QA) works w. 3IU — How it normally updates (w.o. Backtrack) — (& how it updates w. Backtrack) ← 352.00 top; .02ff is about [redacted] backtracking)

Say we have a O^i & we put Q_n in (O^i works ok. w. $Q_1 \cdots Q_{n-1}$), TM asks for input: we put in R before there is any output. If no update is needed, The [R_i] prefix set gives a d.f. on A_n. If its an acceptable d.f., we leave it alone — i.e. we leave [redacted] O^i as the "Pre-R" ~~[illegible]~~ program.

Perhaps I'm doing this wrong! OOPS as is: is a 2IU! It can't do S-induction properly!

But ~~OOPS~~ the OOPS Ref machine can act like a 3IU. S_1 (input #1 that defines the O^i) Is any input that occurs before TM looks at Q_i.

.10 One way to change S_1: ~~[illegible]~~ S_1 was the result of a tree search, over pgms that did not look at Q when ~~until~~ 1 more inst was added, it may or may not look at Q.

Actually, O's OOPS is a 2IU, & it isn't much different from a 3IU: The only difference is!
In a 2IU, after we put in Q_i, A_i comes out.
" " 3IU " " " Q_i we have to put in R_i before A_i's come out.

It would seem that the update process would be the same, — except for the conditions of acceptance of a O^i trial

.16 So let's work out the ~~3IU~~ 2IU case: A possible trick; Don't put Q in until it's decided that "S" (the head end of O^i) is "all in." So Q has this "Null" value until S_i has been "processed".

& Superficially, this doesn't sound good — The decision when to put in the Q, ~~[illegible]~~ becomes part of the defn of O^i.

.19
.20 Another way would be that Q is in all the time & we generate S (by "inst's"). As soon as an inst looks at Q, ~~[illegible]~~ That inst is regarded as the last inst of S.

If O^i loads part or all of Q into a register, O^i doesn't end until that register is used in some way (?). This is a "picky" point — The criterion of when S "ends" .... we can decide later. For the present, let the first inst. that accesses Q (So the system state henceforth is a function of Q) be the last inst of "S".

Anyway, in 2IU, once S is in, (& Q is already in), we add no more input. The output (of Λ or an actual string) will be regarded as the & may be [redacted] right or wrong.

This (2IU) Seems quite diffnt from OOPS: In OOPS, the machine reads part of S, then reads

.30 Q: On this basis, it may give output or it may read some more of S, then perhaps have output which may be final, or it may read more of S before final output. By reading Q OOPS decides how much of S it has to read. So: Is that a "good idea"? — is it a reasonable way to "Run a Ship"?

I think the motivation of that formalism is that it enabled one to "add on" [redacted] code to deal w. the (expansion ~~extension~~) of the corpus. It did not of course, always work, in which case "Backtracking" was needed.

Hvr, the problems in which this sequencial expansion of S, worked, were natural(

3·16·03

3 PM

00:351.40: One way would be to do meditns in conditional complexity order, which is ≡ conditional pc order. Given present O^7, what are O^k's that are (conditionally) "implied" by O^7 — in ≡ pc order.

This seems to **Not** use t. "Grammar" ideas of 350.00 ff!

.05 $\boxed{SN}$ To a large extent, we Have searched the space of O^7's. We know many O^7's that will use the part of t. current Corpus: Is there any way we can search that takes

.07 advantage of this past work done? (.12) Also Can we somehow combine this w.t. Grammar (Genetic Algm) ideas of 350.00ff?

.09 $\boxed{SN}$ In ~ 138.20 (which I decided wouldn't work!) I did a search on O^7's and R's together

10 so that t. sum of their descr. lengths (≡ product of pc's) was t. Lsearch "length" parameter.

Try to desc. t. present problem exactly.

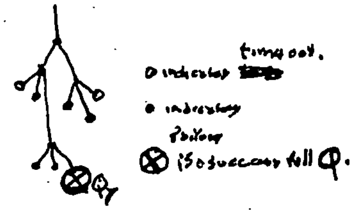
.12 (07) → $\boxed{3.17.03}$ Consider OOPS's ref. machine: Any machine that works for Q_n has done an exhaustive

.13 search of all pgms "simpler" than itself, so no simpler pgm will satisfy $\left(\begin{smallmatrix}A_1 & \cdots & A_n\\ Q_1 & \cdots & Q_n\end{smallmatrix}\right)$.

→ [**Except if a larger CB is used**]. However, as soon as a new Definition is made (.12–.13) is no longer true. (?). ← Check this statement! Not so sure its true!

At any time, t. state of t. system is representable by a tree, that gives t. various token trials that were made. Also, t. "success points" & what level of Q_i was obtained

20 at each pt. Also Failures and "Time outs" must be distinguished (shown separately). also total pc at each pt.



So if finding an acceptable Q_8 seems to be taking too much time (by CB but no soln) we drop back to Q_7 & spend more time on its "timeout" nodes. If they give a new Q_7 in "reasonable time" we go on to try to get Q_8 by "continuing" it. (t. previous Q_8 trial was as a continuation of an earlier Q_7).

[A problem may be if t. successful of t. Q_7's are almost t. same, so failure to do Q_8 by continuing one of them, would make it very likely that another of those Q_7's would not be any good as a basis

30 for soln. of Q_8.]

$\boxed{SN}$ One trick to (perhaps) save much time: take "Q_8" t. latest problem, try it as an input to O^7. If O^7 prints out wrong A w.o. asking for more inputs, Then we must backtrack to O^6; we put Q_8 into O^6 & see if it gives wrong outputs w. no more inputs; if so, we backtrack to O^5, etc. T. foregoing are "fast aborts" — they say that the O^7 isn't recognizing Q_8 as an essentially new input — and it gives wrong reply to Q_8.

The existence of this "trick" makes it clear that t. system will normally try to find a way to identify new kinds of Q's that need extra R inputs

353.00 →

3TM

00:35040: Consider QATM doing MTM probs: [first Lsrchs for soln for ~~[illegible]~~ $Q_1 A_1$. Next it ~~[illegible]~~ Lsrchs for soln to $Q_{1,2}\, A_{1,2}$, using possl. "mutations" of first soln ($\equiv O^1$). For soln to all QAs up thru Q_n, we look for O^n. ~~[illegible]~~ T. search will be over a space in which ~~[illegible]~~ $O^1, O^2 \cdots, O^{n-1}$ are "examples" to be extrapolated. We want some kind of "Grammar" w. $[O^i]_{i,n}$ ~~[illegible]~~ as data.

To start let us assume OOP's-~~[illegible]~~, Forth type ~~[illegible]~~ reference lang., So to work problem$_k$ we can try O^{k-1} & it ~~[illegible]~~ can (sometimes) recognize that this is a new problem — & either give a soln or ask for more code. If this does not result in a soln (we try all possl/codes in Lsrch order), Then make S-Grammar from $O^1 \cdots O^{k-1}$ and use it to generate cands.

this "grammar" may be something that "Grows" i.e. we have a new grammar for each new Q. t. Grammar for Q_k is obtaind by modifying t. Grammar for Q_{k-1}

Examples of Grammars: ~~[illegible]~~ The AZ system w. defns & Lsrch rule: Since it assigns pc's to ~~[illegible]~~ O^i tries, it is a P.D. $\equiv$ "S. Grammar" Any methods of finding regularities can potentially ↑ pc of "Corpus of O^i's" & ∴ make a better Grammar.

→ [[Context can also ↑ pc of "C^icorpus", but it seems difrnt from ~~[illegible]~~ what AZ [redacted] normally does. [redacted] Perhaps our corpus is regarded as part of a more general "Macro Corpus"?]] Needs Clarifn.

Conceivably one could try a CFG or CSG

OOP's ref lang also Gen a P.D. = S. Grammar.

SN on Universal D.f.'s on S. Funcs: Take th. Universal Discrete D.f. on finite strings — Since any P.D. ~~[illegible]~~ on strings can be represented by a pgm ($\equiv$ string) then this d.f. can induce a P.D. on strings $\equiv$ representations of P.D.s on strings ∴ a p.d. on P.D.'s. [I think .20 ff is what 3IU does : If we use a 2IU to represent the final output p.d.'s. Hrn. .20 can use arby methods to dscrb t. final output pd's — they can be Monte Carlo, continuously parameterized function, ···· etc.

Re: NMTM problems: T. Discussn. of .00 ff would seem to hold as well In hunting for 3IU functions, rather than 2IU functions — so not much change in Methodology when we go from MTM to NMTM.

[[I'm not so sure! In 3IU, we have the "R" input to try to create the A_i. In OOPS, we ~~[illegible]~~ have same thing — how much cc to use on this ~~[illegible]~~, before deciding to try a new O^i? Is this t. same as "When to Backtrack"? [If so, its t. Q of how low a pc does t. current Model have to go before we revise our Theory ($\equiv O^i$) current O^i. Also, how much revision to try. How Big ~~[illegible]~~ conceptual leaps to consider? CJs?

Well, maybe this last is "~~[illegible]~~ "No Problem" — i.e. just continue search in Lcost order. This will, to some extent, involve going back to short codes that did not yet converge [redacted]. ← This means undoing recent Defining Definitions

[Say Trainer give pc threshold for new A_j — so a O^i & R have to be found so $O^i(A_j|Q_j,(R)) >$ a certain pc_0. (actually Trainer we may want total pc to be > a certain Threshold.)

I'm temporarily saying this so we can get on to t. rest of t. "Backtrack" problem.

3TM

RESOLN of $\boxed{3\,IU}$ v.s. $\boxed{2\,IU}$ Universal D.Fs ;

3 IU

349.29 starts it but
349.40 has main idea.
350.00 — .15 is Main Argument
— explann.

00: 349.40 ! For 2 IU, we want a funct that gives "right answers" for all Q_i. In a stochastic world, this is often impossl, because for the same Q_i we can have several difrnt. answers in the corpus.

So it seems clear, That in general, 3 IU & 2 IU can't be identical (or even ≈!).

(3)
2 IU are universal, but in difrnt ways (Just as the Discrete & Continuous Universal distributions are continuous, but in different ways).

For TSQ, PST ideas — See Polya!

2 IU is for MTM: if $\underbrace{Q_1 \to A_1 \;\&\; Q_2 \to A_2 \;\&\; A_1 \neq A_2}$ Then there is no MTM soln, & no 2IU soln
for 3 IU of course, " " Then This is easy.

The 2IU soln gives a d-funct. for all old—(corpus) inputs, but it can give a S-funct for new inputs.

.08 I imagine that its usually a very narrow (S-funct, or almost d-funct).

(No) for any new input (not in corpus), There exist pgms that do the corpus exactly, but do arbitrary for
Not No!

10 the new input. Eg. say S_1 is a pgm that does entire corpus correctly, Then if Q' is the new input, consider the pgm: If input is Q Then outputs A_t else invoke S_1.

A_t can be any string we wish. This pgm is clearly much longer than S_1 alone! Ⓠ, but it is of "similar length", so

.15 all strings A_t can be output. ⟶ so This argt. does not really contradict .08

⟶ So, by starting QATM w. MTM problems only, we are getting a machine rather difrnt

.17 from one that will do NMTM. While this may be a good way to start studying TSQ's (30.)

$\boxed{SN}$ A way to get a kind of Universal S funct distribn: To derive a S-funct, we simply list the primitives in the AZ language, The result will be a PD on Functions (as AZ always gives)

20 To get the P.D. on output strings, for a given input, we include that input as a primitive and generate outputs of the system in pc order, by chosing Tokens in the usual "Laplace rule" way. Since the pgms will not be long, the changes in pc's of tokens given by Lap's rule, will usually be not very impt. — We can usually get p.c. of an output by getting all inputs w. Their constant Bernoulli pc's. This may be a faster way to do trials than Lap's rule . . . .

Except that when we make a definition, all of the token pc's must be renormed.

— This renormn is done automatically in Juergen's "OOPS".

$2^{10^{8000}}$ possl. relations in Human Brain; $\langle 10^{3000}$ bits data$\rangle$?

30: (.17) It is signifntly difrnt from a NMTM TSQ. Say we started w. MTM & got a fair distance into Algebra. To start trying English would be a NMTM problem!

So its desirable that TM have fair amt. of experience w. NMTM before trying English!

Or: Could we use English as its first NMTM corpus?

We could start out QATM as a NMTM, but w. Algebra Probs to start out.

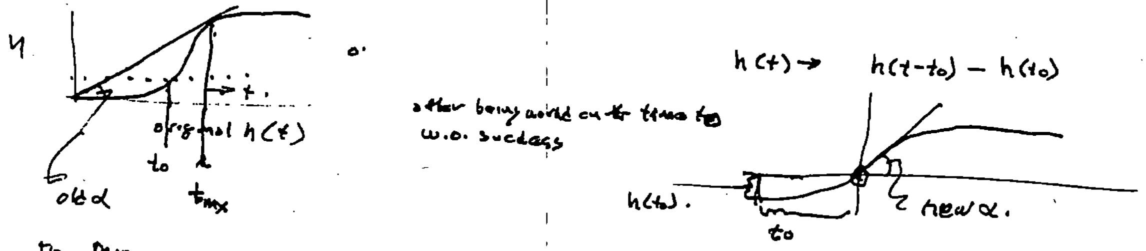
It would always get one code much shorter than any other, but when it didn't, the codes would all tend to extrapolate exactly the same w. new problems. (This is using 3IU)

(By MTM I imply 2IU distribn.)

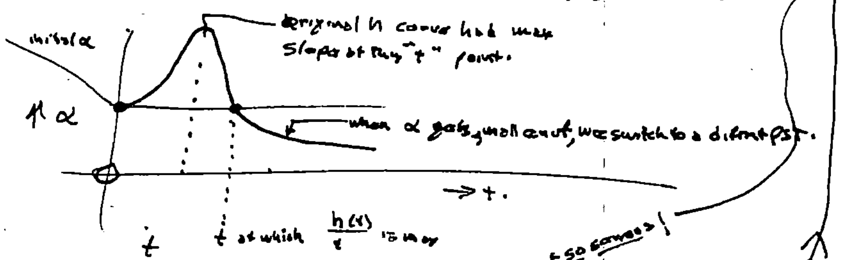
351.00

00:348.40 : no new definitions into O^2 during an entire problem soln. Presumably, before (or after) each problem, we will do a lot of serious updating of O^2 ... often involving serious "definitions" of regys/concs. This last (348.31 ff) seems to make Phase 2 much more practical!.

.03 Indeed, it's (possl/likely) that when we are working on a problem & have not succeeded by time t, that there is a (relatively standard way) to modify the α of t. current PST/problem pair. One possl. way:



10 In this case, α increases (so no chance of frustration) until t = tmx. at which time it becomes t. original α. for larger values of t, [redacted] α ↓.


original h curve had max slope at this "t" point.
initial α
↑ α
when α gets small enuf, we switch to a diffrnt PST.
→ t.
t
t at which $\frac{h(t)}{t}$ is max

May be not so serious! After we reject this PST because of no soln up to time Tmx (or even > Tmx), ~~Then~~ If we do ll updating, then the PST's that are "very similar" to t. one just discarded will also be discarded.

ABCDabcde 1234567890

Not so serious!

20 The discussn. of .03 ff. does not consider modifn of α of PST's that are "similar" to t. one being worked on. This is a serious criticism of it. Does this criticism apply to 348.31 ff as well? I would guess not, since t. α's of all PST's would change continuously w. t. new date, t.

On t. 348.20 corpus: how to deal w. it; how to get good O^2's! To start off, use Phase 1 induction — which isn't so bad! (It's just not "T. Best"). Phase 2 induction looks for past regys in t. solns. of "similar problems" since all of t. O^2 discovery problems are, indeed, similar in ~~[illegible]~~ impt ways, this could be a "not bad" (i.e. acceptable) prelimy soln, to get to Phase 2.

.29 [SN] I had t. idea that t. soln. to MTM (d functs) was formally identical to soln of NMTM (s functs)

30 Now, it seems that s-functs are a lot more complex! — Is this true? It may be that t. reason I expected much difrnce in solns was that the [O^2] were a sort of [redacted] s-functions, rather than d-functs ← (which is what I was thinking about when I wrote about identical solns of MTM & NMTM). — But, superficially, that shouldn't make any difrnce! Any d-funct is a typed s. funct. Is it a dogma that t. d. functs be partial recursive? (i.e. have no output for certain inputs)? Look back on my recent work on this; I got idea that somehow, t. 2 methods were not equivt! See 317.29

.40 (317.31R) AH! I think I understand how 2SU differs from 3SU: Both are universal D.F.s but

See 390.00

9.15.03

3TM

WON in Phase 2 need not take much time at all (.31)

.00: 347.40: "Good concs for Phase 2" includes many "basic", "connecting" concs — so that the ~~[illegible]~~ Set of concs is not only "universal", but it is "usefully universal" ← (whatever that means) (Maybe that's a ≠∃ complete sub set of concs w. reasonably hi pc.)

I can get Phase 1 to solve induction probs over a great variety of Domains, but the domain of Phase 2 update seems "special", different from (Most) (all other?) all other domains.

→ It has to do w. guessing the appropriateness of PST_i for solving $prob_j$.

.07 [Well, so I got this set of psts and experience using them for INV & OZ probs.

.08 [i/o Simply the use of Lsrch to find PST's for OZ & INV probs

? → [Unfortly, solving QA probs in Phase 1 doesn't seem to give a useful corpus for

10 Phase 2 update!] ? ← This (is) seems to be only one PST.

Well, I could try solving QA using a set of OZ solvers — As if it were a normal OZ problem! [superficially, the usual Phase 1 method of induction would not be included, since it takes advantage of special properties of the search for inductive Models.] Anyway, Looking for regys in a corpus is a PST that seems much different from the usual Lsrch in Phase 1 (tho in "usual Lsrch for QA, we do (as part of Adaptive Lsrch) look for regys in the corpus: but I'm not sure its the same "corpus".

ABCD12345678910

Perhaps it would be best to get "phase 1½", in which pure Lsrch is used for OZ & INV problems. Getting lots of PST's for various kinds of OZ probs would be helpful in getting to Phase 2 update (I think)

20 → Note that Lsrch for O^2 w. $[pst_k, prob_i, t_{ki}]_s$ corpus would be of little value unless the reference Universal dist. had been updated w. suitable concs!

Even if Phase 1½ were V.G. at OZ in general, its not clear that it would be any good at finding regys such as are needed for .20! Perhaps if .20 was a "V.G. corpus", the regys would be more clearly visible! m U m U

→ Certainly a first, most diff(t) part, would be to get a good corpus for .20: This involves Many pst's, many problems, Many PST's solving many of the problems (& presumably not doing so well on others).

30 Next (perhaps) would be correlns of various features of the PST's & Probs & the "t"s

The PST's will, presumably, be in factored form. The problems may not be in factored form.

.31 SN Ideally, "Hopefully"; when we get a good O^2 for a corpus, there are continuous params of O^2 that are a funct of continuous params (among other things) of the $[PST_k, prob_j, t_{kj}]$ corpus. In particular, if we have a O^2 for a corpus, and we add a new triplet of data to the corpus, usually, to reoptimize O^2, we will not make new definitions, but will simply modify continuous params of O^2. This would make Parallel updating during "WON search" not very time consuming at all. In fact, it may be possible to not put ≠ 349.00 spec

9.14.03
for
3TM
SPEC

.00: 346.40 : perhaps t. kind of reasoning used in probs like Miss & Cannibals, would be useful:

Also, The Logical reasoning in Th. RAND Program Prover of Simon's Newell's Logical Theorist (LT)

T. pgm. GPS has been Genzd somewhat; so I could see just what useful things it could do. One ~~Gen~~ Genzn of GPS used "Macros", Rather it would use a combo to ~~ach~~ achieve "Subgoals"   See KORF "Lng to solve Probs by searching for Macros".

Devising "Subgoals" for TM is easier in one way than for a Human, because in many areas of Math, a subgoal (i.e. a lemma) can be rapidly tested by TM w. random (trials examples), rather quickly.

The first problem is first: Is it INV or OZ? If INV, we work it by GPS (Vector Genz), How to devise t. score is a subgoal, but criteria for it are unclear).

.10 If it is OZ, there is only 1 goal. I think I'd want to further categorize t. problem: Discrete or continuous: ~~E~~ Pure continuous or Discrete + continuous (i.e. find functional form, then Optz. ~~a~~ continuous params).

I think t. main problem is discrete params. So study how I solve such problems.

Anyway: From pure statistics (no "reasoning") we look at success/failures of Various PST's w. various problems. We look for features of PST's & Probs that would lead to best more LCJ curves. When TM can "reason" statistical methods are mixed w. "reasoning" — but I don't know just how.

9.15.03 Basicly, t. way I expected TM to work: It would start out in phase 1 in QATM,

.19
.20 & learn enuf of t. right Concs, so it could do $[PST_k][prob_Q]^{t_{kQ}} \to O^i$; $O^i(Pst_r, Probs) \to h^i_{rs}(t)$

In order to do use "Parallel" updating, TM must be able to ~~find O^i's rapidly~~ modify O^i's rapidly, when a single ~~(single)~~ $(PST_k, Prob_\ell, t_{k\ell})$ changes, otherwise, ~~TM~~ TM will have tried all Th. "similar" cands (that it should have rejected) before t. O^i changes enuf to reject them.

Would it be possl. to use GA rather early, to find O^i's ~~E~~ for QA, ... but more important, for Update (like .19R)? Actually, what I had in mind originally wasn't Much difnt from GA. I had ~~then~~ all the old cands (t. "population") of O^i's & their ~~Sc~~ assoc. Scores (≡ "fitness function")s. From this set I would produce new trial O^i's.

~~Then~~, I expected to use this General form for updating QATM (Phase 1)

.30 It would seem like a good idea to have TM start working on t. update problem using update data, as soon as possl. (presumably update data is more relevant to update, than QA data ~~&~~ & QA solns is). — But we need to have a good set of Concs to start out with — before Update induction can "Get off t. Gnd.".

It would be good if I has some ideas on what update induction consisted of — What Concs it could use/need, so I could properly orient ~~a~~ t. "Phase 1" training.

Two purposes for Phase 1: ① Get good Concs for Phase 2 ② Trainer has to lrn how to write TSQ's.

348.00

03TM

.00: ∴ Perhaps study properties of integers: sum dif products exists. — quotient sometimes.
3 ÷ 0 is meaningless — No "value" is assignable

So teach it to solve ~~problems~~ equs in which solns are integers, ($\frac{8}{3}$ is meaningless)

Introduce fractions, Then more kinds of equs are solvable.

So teach how to solve them.

Maybe start w. pos. integers, then neg integers then fractions:
Or fractions before neg integers. (maybe try it both ways!)

.07 In each case, give equs to solve.

$\boxed{SN}$ Consider H. Freudenthal's "Lincos": How to teach Extraterrestials about Earth, about our language. Would it be useful to read This Before trying to teach stuff to TM? — Presumably the ET's

10 know a lot more than TM (to start). I do have a copy: but can't find it.

Re: .00 – .07: At the end of this TSQ: would I be able to start English trng? There are various "facts" TM could know: Say "inequalities". Is 3 > 0? This "English" could be concernd w. "Present Tense" only.

Is "3x+7 = 3 an equation"? ← is this a "useful Q"? Would the concept "equation" be useful to TM? It might be useful for analogy. It has several properties: ① possible ~~[illegible]~~ literals. Possible solvability of some literals in terms of others or in terms of constants.

.20 It seems quite possl. to continue .00 – .07 easily, by introducing new operations, functions. e.g. After all linear equs are solvable, introduce $\sqrt{\ }$ operator. Next, perhaps, complex nos. From this, it would seem that fairly complex probs could be solvable. Introducing $\sqrt[3]{\ }$ could give solns to cubic equs using the "invertible substitution" heuristic

From linear equs, instead of quadratic equs, one could solve simult. linear equs.

I could, perhaps, draw a tree graph, showing various alternative paths of cons, & problem solns.] rnd.

I could avoid the ~~[illegible]~~ diffics of inexact compns, by restricting TM to literal eq. solns.

9.14.03 2 Impt Goals ① English ② phase 2:

.30 For phase 2, Fitting the a, μ, σ^2 params to $h(\ )$ — or rather optimizing in a continuous space, is probly not hard. T. problem is for TM to be able to look at a [pst_i, $prob_j$] set ([redacted] & [redacted] have good enough "understanding" of the relation of pst, prob ~~columns~~ to be able to devise a good OS for that kind ~~[illegible]~~ of corpus. — to get assoc $h_{ij}(t)$ (for inv probs)

Kinds of Reasoning useful for this induction problem: Categorization of PST's, & probs. So we can correlate "types" of PST's w. "types" of probs wrt. expected soln. times. ("a" & μ).

If all TM has is these categories, then if one pst in a class, fails to solve a problem, we may want to discard all of the pst's in that class.

Spec 347.00 →

9.12.03

03TM

5:48 "52 turns" on DVD. 5:24: "Hyper" is slowing down.

5:59 1152 steps

.00 (Spec 344.40): Perhaps it would ~~need~~ not need "logical reasoning": that statistical reasoning would be "good enough" — Maybe more than adequate! Is logical reasoning (d-functs) really a subclass of statistical reasoning (s-functs)? It may be that d-functs consent ideas to ∞ more easily! The usual s-functs that TM uses do not ever get p=0 or p=1, because of "universality". The result is that $(1)^{\infty}=1$ but $(1+\epsilon)^{\infty}=0$. So Reasoning about ∞ ("All") could be quite different between d-functs & (universal) s-functs.

As I recall, present systems to do integration by "heur pgmg" are just given a set of heurs & set loose on the problems. Any difficulties fixed up by A.H. adjustments.

How to work this into a "TSQ" is not Clear!

.09 — Maybe try ordering the heurs in the sense of which heurs need what other heurs — so a "partial ordering".

10 This would help ~~[illegible]~~ in writing a TSQ.

The ordering of .09 is different from the partial ordering obtained by having common concs (or actually using a previous heur as part of the code for a new [redacted] heur ← ?).

Actually, I think that ~~[illegible]~~ what I [redacted] had in mind was to give TM the heurs needed to do much of symbolic Integration problems! Then I help "factor" the heurs, so TM can make a Grammar, ~~[illegible]~~ → a p.d. over them. Actually, what are in existing "Integration" pgms are not really heurs, but rather PST's.

20 What's really necy: to get a set of PST's for integration: To factor them ("by hand"). Try to figure out a way that the PST's & their component concs/heurs could be lrnd, or "coded in". We want to end up w. a System that will not only be able to integrate, but able to discover/learn new/better ways to integrate! (Possibly including the invention/definition of new functions that would permit the "closed form" integration of many new functions.)

. Is there a Book on how to do integration? Normally a Calculus book will have only a few tricks: (like integration by parts, substitution of variables). I do have 1 book w ~ 30 pp on integration.

30 Substitution of Vars, & Int by parts are main tricks. Some special tricks: all rational functions are integrable by expressing them in certain simple forms.

The Olde alternate problem is Elementary Algebra because "TSQs" for humans have been written. Instead of spending much time on elementary algebra, problem of function inversion, Assume that this is "primitive knowledge".

In elementary algebra, they ask Qs like "is this no. even or odd? Is it an integer, is it rational? Is it algebraic? These [redacted] concepts are sometimes useful, but infrequently.

~~[illegible]~~ Suppose we want TM to learn to ~~[illegible]~~ find x ∋ $x+1=7$.

If he has some experience w. integers, he knows x must be an integer, so try 0, ±1, ±2. . . .

ABCDE
Aabcdefb
hijklmnop
DABCDEF
Gabcdef

9.11.03

3 TM

: While the meanings of "numerically solve (x³+3x−15, x) and symbolically Integrate (7x²+sin3x, x) etc. are clear, Its not clear how to teach TM what a "vector" is — or an "equation" : These meanings ~~we~~ should depend on how I want to use them, & how I expect TM to use these concepts, So I'll have this seq. of "Milestone" problems, then I have my own cores. I used to solve them. One Gen. I use is "value" (341.37), Also this T Einstein hour: (342.00)

* I may be able to write TSQs to do symbolic integration as well as numerical & literal solns of eqs; & dif. eqs. (ordinary & partial)

To trn to do these things (even "very well") I'm not sure it would understand much about algebra, or what numbers were. Would I be able to teach it much english based on its knowledge of these Domains?

So in Phase I I will have 2 kinds of trng: (1) definitions of sytms, macros, or whatever a/o ~~[illegible]~~ functional subtrees. (2) Contexts.

Context seems very general. The pc of a Token can be any ~~[illegible]~~ function of the state of the System & any "Environmental" info (like "This is a Geometry problem"; "This problem was given to me by Joe"; "The CB is rather small for a problem of this type" : The previous sequence of ~~[illegible]~~ problems were mainly in linear Algebre (recency context)).

[Very often, context data like the above, would have too small SS to be definitive. Only by using "logical reasoning", one mite be able to pool info from disparate contexts.

BIG Q! Say I was able to write a TSQ for TM that trnd to do symbolic Integration: as well as & Better, than current pgms (Macsyma, Maple, Mathematica). AND, it was able to go beyond & Learn to do new integrals by itself, define (and suitably Name) new interesting functions. Could I get it to do other areas of Math: e.g. could it ~~[illegible]~~ lrn to solve eqns easily? Could it lrn to work on Riemann Hyp.? Or would it be necy to give an aux TSQ to teach it elementary Algebra? Could I teach it enuf Physics for it to usefully look for a model for "Cold fusion" (or any other anomalous (or "useful") phenomena? Could I get it to recognize when it had found something Useful?

.31 : 342.03 SN Rel Einstein Hour (342.00) : ~ Hours: Define idea of "Vector" then treat sets of objects like single objects (x-ing them etc) to solve eqns & to solve other kinds of problems. Other ~ hours: "Duality" (Genzd) if a set of objects, satisfys the axioms of a diffnt set of objects₂ then all theorems about (1 2) have corresponding theorems about (2 1).

Would a useful "Integrator" be possl. w.o. any understanding of Elementary Algebra? ? Probably it would be able to do "Logical Reasoning" (343.23), → (345.00 spec)

3TM

00 : After we have Several "successful" functions that solve milestones, ~~[illegible]~~ part of "context" discovery, becomes well defined! We just look at ~~[illegible]~~ each of t. tokens in each of th. functions & try to find ~~[illegible]~~ contexts of tokens; (configs of tokens) that make ~~[illegible]~~ certain

.03 tokens (less more) likely. There are, tho, many other kinds of contexts. By watching myself work ~~problems~~ t. problems in t. TSQ, I should be able to propose various kinds of context.

~~[illegible]~~ "boosts" in OOPS (& its gains) ~~[illegible]~~ is an impt kind of Context. Also the Recognition functions "R" in § 1 of "TS Report" ~~[illegible]~~ is an impt

10 class of Contexts.

T. context of .00–.03 is a "looking ~~Backward~~ Backward" model. It tries to get a P.D. for each token based on t. tokens "preceding" it in t. function sub~~tree~~, that it is t. "root of": I'm thinking of a "functional lang." like Lisp ..... For T. forthish lang in OOPS, ~~Context~~ this kind of context is different! It is t. state of t. entire system when t. Token in Question was introduced.

Actually, this is true in Lisp also: Since instructions often have "side effects" that modify meanings of functions. Tho all system states must be brot about by previous tokens (≡ "code"), it may be better to regard "context" as system states.

20 code an causal system states.

FOR The Ruff Outline of TS Construction at 342.26 ff, considerations of "kind of Reference machine" should be irrelevant (?).

.23 A possl. trouble; In my thinking about my own problem Solving, I often (maybe almost always) use some "logical reasoning". I have to either "teach" that to TM or put Some of it in as "primatives".

I may want to use something like t. notation of Maple (or Mathematica or ~~[illegible]~~ MathCad) — Because its very specific & ~~[illegible]~~ almost always unambiguous

30 T. language history Macsyma ⇒ Maple ⇒ Mathematica This last may be a more integrated System, Since it was directed by ~~one~~ one person for its early history & it had some benefit of t. earlier Systems. I do have a Big Manual on Mathematica.

On t. other hand, ~~[illegible]~~ Lisp (or "scheme") has perhaps not many built-in concepts of t. kind I want ~~to~~ TM to "lrn" t. meanings of.

T. short "book" "a Tour of Mathematica" lists various things it does. Some of them, it would be useful for TM to lrn & to lrn to Extrapolate; like solving Eqns. & integrating.

3.10.03

3TM

Fix Bathroom
Clock.
6 volts?

00:341.40 : (2) T. Einstein heuristic: To solve equs (& perhaps other problems) write t. expressn to be solved for w. "x" as t. unknown object. Transform t. equ ~~[illegible]~~ in ways that would be legal if you know what x was. Do this until it's clear that ~~[illegible]~~ what t. soln. is, or that there is no soln, or

.03 many solns. → 344.31

Consider Classic A.I. probs & solns written for them. They are usually search procedures, Heuristics are ways to narrow down t. srch.

One kind of TSQ: use ~~[illegible]~~ a regular text book and see what t. student has to know in order to work each problem. Quantify just how much additional "knowhow" t. student needs

10 for each problem. If that knowledge is learnable, find ways to "teach" by examples, by Minimal "hints". Compute CJS for each ~~proble~~ knowldg increment. — ~~[illegible]~~ ~~[illegible]~~ (w.o. training/lrng) See if the CJS seems to go as we move along t. (Macro) TSQ. Even if its largish, if it doesn't ↑, then it may be usable! — Humans may have very large IPC. Into proc capacity.

.15 Say for Elementary Algebra: draw up sequence of human-type problems & see how it deviates from a desirable TSQ. Then see if I can find stuff to

.18 augment it so it is an acceptable TSQ

At a very elementary level, there are problems in foundations of

20 Math. Analyzing via Lisp may help here.

We must use Lisp to tell it machine "what t. problem is" T. machine could have an understanding of Lisp: and so that any problem could be presented to t. machine.

.26 For .16–.18 Get big Milestones. Estimate ~~[illegible]~~ CJS Distances (but in Bits or powers of e or 10 since its logarithmic) Then fill in ~~[illegible]~~ before t. Big Milestones to get smaller CJS's .... ~~Hardware~~ Moores law is .7 to 1 bit/yr. (BJS) ~~(binary [illegible] = $\log_2$ (CJS))~~

BCJ
Def: ~~[illegible]~~

on BCJ Binary Conceptual Jump

30 First run: writing each "Milestone" soln. indeply: Get / indep BCJ's (very large).

Get BCJ's when Milestone Solns able to use common Cases w. previous Milestones.

At this point ~~[illegible]~~ the BCJ's of t. milestones should be ≃ (comparable) but probably well beyond achievability, — Also maybe ~~[illegible]~~ BCJ's may ↑ ~~[illegible]~~ w. each milestone because of "Scaling" effect.

To Get t. ~~[illegible]~~ BCJ's ~~[illegible]~~ to be comparable, we probably need

Contexts to ↓ BC's at first

9.7.03

3TM

.00 : An/Very impt idea in reasoning is GPS: T. "Vector Goal". To lrn to solve a/DVV problem, first lrn to design Vector Goals; ~~[illegible]~~ Seems related to "Multi-Objective Optzn" — but I'm not/so sure it is. In GPS, we want to get all vector components to zero. In Multi Obj Optzn, it seems quite difnt. Tho, as a sm problem in GPS, we may have to decide if one trial is Better than ~~[illegible]~~ another in a scalar sense.

Consider classical heuristic ~~[illegible]~~ probsolving: (Maybe look at old AI Encyc). With a given set of heurs, one can solve a certain set (sometimes an infinite set) of problems. Can one make a Grammar, a p.d. ~~of~~ over most of t. heurs & see if we can find a way (a TSQ) to teach those heurs? Better, would be to find a set of problems initially not oriented toward t. heurs, but for which t. heurs were appropriate } Meaning What?

.10

Re: [PST$_i$] set of PST's: I could use a Grammar for a set of Human PSTs "casecount = no. of time "usually used"" in sorted by User! Each PST could be wtd by ssz! — to Give val/"apriori" importance. ~~Later they could be m~~ Those wts could be modified by empirical success/failure of t. PSTs. T. ~~[illegible]~~ Unconditional pc of t. {PST$_i$} are useful when we have little experience mapping problems to appropriate PST's. Later TM will learn to find "h( )" functs for each problem, PST pair.

Given a c(f)Grammar (or any constructive Grammar w. definable pc parameters) ~~One Parses Various~~ To (adjust/match/optimize) t. pc parms to t. corpus: Parse t. corpus with t. unprobablized Grammar; Use t. freq. of use of each choice to get values of pc's. (Laplace rule, say)

Parsing can be done via Matrix or ~~[illegible]~~ possibly simpler way.

Parsing for General ~~[illegible]~~/Generative Grammar using UMC, & ~~[illegible]~~ ~~pc's for umc "table"~~

~~[illegible]~~ may be diff$ to Parse!

[SN] In Many Domains (Maybe MOST) parsing t. corpus is not a problem/ T. Corpus has already been parsed & t. problem is to define new combinations of Old concepts

A few TSQ ~~Copying~~ Writing Methods!

1) Forward 2) Backward 3) Known AI projects w. heurs: Factor heurs & get TM to build ~~[illegible]~~ grammar. This is perhaps very ~ to (2): Also ≈ to ideas of Trainer factoring set of PSTs & ~~[illegible]~~ having TM make Grammar.

.30

3.10.03 4) Mix of 1) & 2): Write TSQ from one "state of knowledge" SOK to another "SOK".

→ My impression is that normally TSQ writing employs all of t. techniques used in 1) thru 4) That there are a small no. of operations that are used in all TSQ design.

In General/(writing TSQ's) & (Problems Solving by t. trainer) are very similar:

Hvr, writing TSQ's is perhaps ~~[illegible]~~ best regarded as a "Not well defined problem."

.37:34.34 Consider writing TSQ's w. 2 (at least!) primatives: (1) Some expresss have "values" assoc w. them. In an expression that has a value, that value is invariant if any subexpression in it is replaced by its assoc. ~~[illegible]~~ value. [is this discoverable? — I think it's always true]

9.6.03

03 TM   Bipstc.tex) on world

/ Sent ~ 605

00 : ~~Dapt [illegible]~~ Some Features of Revision. 2.0

1) Better way of improving Update Process. ~~[illegible] [illegible]~~ A technique is used that appears to be ~~[illegible]~~ superior to ~~[illegible]~~ Lsearch. pp ~~[illegible]~~ 14-18

2) ~~The method of using the~~ How to use ~~A [illegible]~~ information on failed trials ~~of~~ for improving updating, pp. 15

~~4~~3) Two methods ~~of~~ of realizing ~~[illegible]~~ probabilistic functions for optimization. p 18,19

~~5~~4) More ~~[illegible]~~ comparisons of Phase 1 with OOPS. ~~pp 22-24~~ (18, 19; 22-24) → also Section on state of Rsch.

~~5)~~ ⑤ The summary of "~~state of~~ ~~[illegible]~~ Rsch" is not v.g.

10 ~~4) Two forms of Probabilistic functions to enable [illegible] [illegible] induction stochastic [illegible]~~

Dear Jan M!

Here is ~~[illegible]~~ Revision 2 of report. Main ~~[illegible]~~ new features.

1) Various Corrections of Typos and some ~~[illegible]~~ expansions of explanations

2) Better way

20 : On writing TSQ's! One of the early diffys seemed to be related to very ~~[illegible]~~ fundamental Q's about basis of Mathematics: ~~[illegible]~~ ⓐ Position of symbols on input register! ⓑ Just what I wanted TM to learn/to do. It may be that the present QA formalism can clarify this. ⓒ This diffy might have been repaired by using Lisp ~~[illegible]~~ "Quote" notation.

.24 Some random ideas on this! Write TSQ ~~[illegible]~~ w. rather large Milestones, using "large" Concs. & Contexts. Try to get idea of size of pc's of needed Cncs, Contexts.

.25 I had some diffy explaining the idea of "Value" of an express. Is this still a problem. "Value" seems to be an impt. idea — That it could (partly) be defined recursively.

Re: .24-.25: Would that work on solving eqns; discovering ~~rules of~~ laws of Alg; Solving linear, quad, cubic eqns, be useful?

.30 A TSQ to solve ~~linear~~, then non-linear eqns without be bad; i.e. have to go ~~from~~ to Quadratic to Cubic could be just again have that it uses ~~[illegible]~~ for other problems as well.

.34 "Value" is assoc w. some strings, but not all strings. (see 341.37)

I think that I felt the idea of "Value" was essential in understanding Algebra — But ~~[illegible]~~ actually, TM could solve eqns w.o. having that aspect of understanding Algebra. Perhaps it would be easier for TM to pick up the ~~[illegible]~~ "Value" idea later from maybe Analogic Reasoning.

9.5.03
03JM

00 : Contexts are of 2 kinds (at least(1)) : (1) R functs : which are d-functs : (Recogn)

.00 (2) Most contexts are S functs. So in working on d-induction, we will want to use S-functs for context (at least).

It is of interest that these contexts will, as "S-functs" have a certain "Bernoulli" ~~flavor~~ ..... i.e. We will notice a certain context (which is a d-categorization), then we will correlate that context w. various Token uses ~~[illegible]~~ & "correln" is ≈ Bernoulliish.

At ~~first~~ first t. no. of tokens will ~~be~~ small, but eventually, there will be an enormous no. of them.

A General S-funct, ~~wh~~ would be able to take as input, t. denot. "The Current Context" (which can be any possl. context) & hv as output a p.d. on current (& possibly newly invented) tokens, or it could have an arby sequence of tokens as output — & assign pc's to all possl. sequences of output tokens

---

SN Note my rather General approach to "Phase I" : related to my ideas about Genzd "G/A". Main idea was how to get from one "System state" to a better "System State".

Perhaps: For each type of problem (category, context) t. "no." (or entropy) of possl. solns must be kept below a certain level, or those probs can't be solved.

~~[illegible]~~ Review t. 2 Main ideas on Phase I : (a) TSQ vs. context (a "1st Genzn")

(b) The "GA" idea of modifns. to "genz" past successful "Macrostates".

← . context ideas of .01 → 10 need clarifn !

Footnote for § 3.1 ; Page 17??

\footnote {The sophisticated statistician will ~~not~~ note that while the solution to ~~the~~ any optimization problem is invariant if the utility function G is modified by a monotonic, possibly non-linear, transformation — ~~that~~ that the value of γ in in equation(11) ~~will~~ will not ~~be~~ always be invariant under such a transformation.

Equation (11) is correct only if the ~~GM~~ utility function, $G^{\#}$, is "linear", i.e. the ~~[illegible]~~ utility of $G(x)$ with probability 1, is the same as the utility of $G(x)/p$ with probability p.

If $G^{\#}$ is not linear, and no equivalent information is available, then, while the solution to the optimization problem is "well defined", the solution to the "strategy optimization" problem is not well defined. ~~[illegible]~~

fortunately, several important utility functions, ~~[illegible]~~ such as time, and money are linear or linearizable. }

9.4.03

3TM

Work on this later.

00:337 : → Middle of Abstract! The system does not start w. "Series preds" — it starts w. QA induction.

To start off, the machine learns a particularly general form of induction: — Given a sequence of question answer pairs, it must extrapolate an answer to a new question. The questions and answers can be strings and/or numbers, so this particular model covers very many kinds of learning problems.

in section 1.1 (early TM) I do d-induction! In this case getting pc's of preds — ~~of the~~ — the non shortest codes will usually be much longer than the shortest code

The use of alternative codes for backtracking, is impt, however.

The section 1.2 on updating w. R functions has to be proofread.

Re: The footnote on p8 about whether P is "large enough" —— Possibly augment F.N. w. part of letter to D of 27 Nov 02.

↳ Re Burra: Anth. Prin.
SUN 2003
S [illegible]

So: done thru §2.1. Start §2.2

Rect. distribn.

(SN) Rect distribn, or $\sigma^2 = 0$ are N.G. They give zero probys for cases w. $t < \mu - \sigma$. also failure for $t > \sigma + \mu$ are given proby ϕ.

Also the other 2 d.f.'s may not be so hot — The tails on them are very narrow

So very small pc's assoc w. data far from μ! — Worry!

SN Would I now be able to read Cowan's "AGI" properly? — also be able to read his Book w. interpolating reinterpret, properly? — Mainly use his listed hours. Also it probably does record facility thru the no. of hours.

An Immediate Approach!

1) Write/Bib. review of recent. impt. ideas.

2) Draw up TSQ for Phase 1 a) First, do it w.o. contexts.... CJS's can be quite large, but only due to seemingly large costs of concs used. Perhaps Peter Bumgardner already did that! Actually it's not clear as to how feasible it is to do TSQ (w.o. context) first then add context. It's more reasonable to do them concurrently ~~simultly~~, so I'm sure an adequate context is possible, before I continue the TSQ development in "that direction".

3) One (perhaps reasonable) way! Write up a TSQ for Humans, that seems "humanly reasonable". This "TSQ" will be BIG concs: I then must find ways to factor and/or "contextify" them adequately.

4) Note that the R functions (Recognition functions) of §1 of the report, are a variety of Context. Some functions automatically choose their contexts by only accepting a certain "domain" of input values. I may want to integrate these kinds of "Context".

9.4.03
3TM

4 Probability distributions on Probability Distributions: How to realize Universal Distributions on Probability distributions.

5 ~~4~~ Training Sequences: How to design a sequence of problems to train the system. ~~The input information for training the system: How to~~

6 ~~5~~ Efficiency of the system

6.1 ~~5.1~~ Updating: How much time to spend on ~~updating~~ updating.

6.2 ~~5.2~~ Ultimate limits of the system: An argument ~~An argument~~ ~~that the system can become~~ There may be no ~~[illegible]~~ upper limit on performance of the system.

6.3 ~~5.3~~ Genetic Programming: ~~How Genetic Programming and the present system~~ ~~[illegible]~~ Relation of Genetic Programming to present System

7 ~~6.~~ Related work ———— (perhaps Genetic Prog. Should be in here!)

7.1 ~~6.1~~ Relation to Lenat's CYC

7.2 ~~6.2~~ Relation to Schmidhuber's OOPS

~~7.3 Would cause ~~annoying~~ [illegible] problem (But find [illegible] to section 6.3~~ ~~6.1 ~~[illegible]~~ After Eq. 2.1 This is only valid to that section~~

8 ~~7.~~ State of ~~the~~ Research: What has ~~been~~ done, what needs to be done.

I can have more extended titles in the sections themselves! — so no diffy in T.O. Cont.

How to design ~~a~~ a sequence of problems to train the system

9.3.03

3TM

"Contexts"

.00:336.401 (#6 (cont.)) difficulty can be dealt with by discovery of "contexts" (Section 1.3). The discovery of contexts must be ~~[illegible]~~ ~~[illegible]~~ ~~[illegible]~~ integrated with training sequence design.

SN 1) ~~Perhaps~~ This new section is ~~[illegible]~~ all on TSQ's! Perhaps this whole discussion should be in the section on TSQ's!

2) Discuss [illegible] of ^normal^ root ~~[illegible]~~ problem because of ↑ in no. of concs. As no. of probs ↑, ~~[illegible]~~ → ∞, this need not converge to a constant! — "Context considerations" prevent divergence. ~~[illegible]~~ Give Generalized defn. of Context, as anything that modifies the ~~[illegible]~~ pc of introduction of a "token" ~~at a particular~~ pt. in Generating a cond. A context defn is legit ⟺ it ↓ root of corpus (may be OSL "exception")

337

SN Write to Marcus: Re: "Dynamic MDL" I would think that anyone smart enough to discover Dynamic MDL would immediately ~~[illegible]~~ transition to the Univl. Dist., which ~~[illegible]~~ takes care of this in a very natural way. Hvr. the idea of 2 part codes (≡ scientific Laws + predn using these laws) is very deeply ingrained in the sci community. Even normally very intelligent people like Wallace & Gell-Mann are unable to ~~[illegible]~~ transcend this bias. Perhaps only younger scientists like yourself ~~would~~ will be able to deal w. this.

Contents:

Introduction: ~~[illegible]~~ ~~description of~~ ~~[illegible]~~ Description of overall operation of system — ~~[illegible]~~ ~~[illegible]~~ Page 1 Maybe Unnec.

1 Inductive Inference: Detailed operation of Q, A induction.

1.1 Early training: Operation of system at beginning of Training Sequence ← Maybe very specific.

1.2 Updating: How system is modified after ~~problems are solved, or~~ problems are solved.

1.3. Scaling: ~~How problem solving [illegible]~~ Time needed to solve successive problems normally increases ^very^ rapidly. How to prevent this.

2 Inversion problems: Solution of P and NP problems.

2.1 Improved Updating and Search Technique: How the system uses its Optimization capabilities to improve its update and search methods.

3 Time limited optimization problems: How the system solves these problems.

3.1 Improved updating and search technique: How the improvement ~~[illegible]~~ techniques for Inversion problem solving ^problem solving^ are applied to Optimization problems

9.3.03

3TM

$337\frac{1}{3}$ } $337\frac{2}{3}$ } exist

"Contexts"

.00:336.401 difficulty can be dealt with by discovery of "contexts" (Section 1.3). The discovery of contexts must be ~~planned~~ ~~[illegible]~~ integrated ~~simultaneously~~ with training sequence design.

(SN) 1) ~~Perhaps~~ This new section is ~~[illegible]~~ all on TSQ's! Perhaps this whole discussion should be in the section on TSQ's!

2) Discuss limit of normal rcost ~~par~~ problem because of ↑ in no. of concs. As no. of probs ↑, ~~[illegible]~~ → ∞, this need not converge to a constant! — "Context considerations" prevent divergence. ~~[illegible]~~ Give Generalized defn. of Context, as anything that modifies the ~~[illegible]~~ pc of introduction of a "token" ~~at a particular~~ pt. in Generating a cand. A context defn is legit ~~as~~ if the rcost of corpus (maybe OSL "exception")

[SN] Wrote to Marcus: Re: "Dynamic MDL" I would think that anyone smart enough to discover Dynamic MDL would immediately translate to the Univl. Distr., which ~~[illegible]~~ takes care of this in a very natural way. Hvr. the idea of 2 part codes (= scientific laws + predn using those laws) is very deeply ingrained in the sci community. Even normally very intelligent people like Wallace & Gell-Mann are unable to ~~[illegible]~~ transcend this bias. Perhaps only younger scientists like yourself ~~would~~ will be able to deal w. this.

is relatively easy. Writing long training sequences that give the system great capabilities in many domains is more difficult

Comments on re-reading, Abstract: "starts off by learning to solve Inv. & OZ probs," — False! it starts w. QA probs.

I may say that I'll describe the "steady state" behavior of the system.

It says in Abst. that ~~a~~ Beginning TSQ's for Lsrch are easy to write because of ~~CJS~~ availability of COS. — Not so easy!

Intro: ~~[illegible]~~ QA predn: "Functions that define classes of problems." This is done in § 1, but it is actually a type of "Context"

"At first we will use simple Bern. prediction" — I don't see this being true!

Lsrch is mentioned in Intro w.o. explaining what it is! (It is introduced ~~mentioned~~ in ~~the~~ Abstract)

3TM

00 The general theoretical framework of the system seems to be sound. The main unsolved problem is the design of a suitable training sequence ~~as was discussed in section~~ [redacted] (~~se~~ Section 4+1=5).

.04 We expect that the training sequence and the set of contexts needed to deal with scaling ~~solve~~
05 will have to be developed concurrently.

In OOPS, we have a system that has solved a set of problems using Lsrch. ~~Would this be~~ Can we consider ~~them~~ those problems to be as ~~a kind of first step~~ [redacted] ~~a training Sequence?~~ ~~a set of early~~ first steps in a training sequence?

10 In ~~the~~ early training we expect the conceptual jump sizes to be much smaller than the [redacted] 10^{10} needed for the Tower of Hanoi ~~encountered in OOPS~~. We ~~would~~ consider ~~Hanoi~~ The tower of Hanoi problem solved by OOPS to be a very difficult problem — to be solved much later in the training sequence.

While OOPS did show useful transfer of learning from an earlier problem, the training sequence was much too short to exhibit the scaling difficulties of section 1.3, and ~~at~~ in its present form, it has no means to deal with scaling of this sort.

20 → insert .04-.05

22 \Section{ State of Research: What has been done and What needs to be done } ← Is this the last Section.

This report outlines the workings of the Alpha system and gives some mathematical details of its operation. At present, it appears that the Theoretic foundation of ~~this system~~ Alpha is sound. The main immediate problem is the design of a suitable training sequence for it.

~~E~~ OOPS is a program that has solved a set of problems using Lsearch. Could we not use these problems as the first steps of a training sequence?

#6

30 The main value of OOPS has been in showing how Lsearch could be used to solve significantly difficult problems, ~~a~~ and that it could use information from earlier problems to improve solutions of [redacted] later problems.

It is not clear ~~that~~ how the two sets of problems solved by OOPS could be continued to solve problems of increasing difficulty. It is relatively easy to write short ~~seqs~~ training sequences that don't go anywhere. ¶ We expect to start our training sequence with very easy problems: ~~[illegible]~~ the C.J.S. [redacted] will not exceed 10^6 — much smaller than the 10^{10} needed ~~[illegible]~~ for OOPS to solve the Tower of Hanoi.

~~A very serious problem not~~ dealt with in OOPS, is the problem of "scaling", in which the time ~~is~~ needed to solve a [redacted] problem is multiplied by a sizable factor for each new problem. We expect that this ~~number~~ →337.00

3TM

00 The general theoretical framework of ~~the~~ system seems to be sound. The main unsolved problem is the design of a suitable training sequence ~~(as was discussed in section~~ [redacted] (~~of~~ Section 4+1=5).

.04 We expect that the training sequence and the set of contexts needed to ~~[illegible]~~ deal with scaling

.05 will have to be developed concurrently.

In OOPS, we have a system that has solved a set of problems using Lsrch. ~~Would this be~~ Can we consider ~~this~~ those problems to be ~~as a kind of first step [illegible] a training sequence?~~ ~~a set of early~~ first steps in a training sequence?

10 In ~~the~~ early training we expect the conceptual jump sizes to be much smaller than the [redacted] 10^10 ~~encountered in OOPS~~ needed for the tower of Hanoi. We ~~would~~ consider ~~the problem of~~ The tower of Hanoi problem solved by OOPS to be a very difficult problem — to be solved much later in the training sequence.

While OOPS did show useful transfer of learning from an earlier problem, the training sequence was much too short to exhibit the scaling difficulties of section 1.3, and ~~[illegible]~~ in its present form, it has no means to deal with scaling of this sort.

20 → insert .04–.05

.22 \Section { State of Research: What has ~~been done~~ and what needs to be done } ← This is the last Section.

#6

This report outlines the workings of the Alpha system and gives some mathematical details of its operation. At present, it appears that the Theoretic foundation of ~~the system~~ Alpha is sound. The main immediate problem is the design of a suitable training sequence for it.

~~E~~ OOPS is a program that has solved a set of problems using Lsearch. Could we not use these problems as the first steps of a training sequence?

30 The main value of OOPS has ~~been~~ in showing how Lsearch could be used to solve significantly difficult problems, ~~a~~ and that it could use information from earlier problems to improve solutions [redacted] of later problems.

It is relatively easy to write short ~~seq~~ training sequences that don't go anywhere.

It is not clear ~~that~~ how the two sets of problems solved by OOPS could be continued to solve problems of increasing difficulty. ¶ We expect to start our training sequence with very easy problems: ~~[illegible]~~ the C.J.S ~~[illegible]~~ should not exceed 10^6 — much smaller than the 10^10 needed ~~[illegible]~~ for OOPS to solve the Tower of Hanoi.

~~Another A very serious problem not~~ dealt with in OOPS, is the problem of "scaling", in which the time ~~[illegible]~~ needed to solve a [redacted] problem is multiplied by a sizable factor for each new problem. We expect that this ~~[illegible]~~

→337.00

9.3.03

3TM

.00 "Roadmap" 324.01–.30 first attempt at expo
327.10 second attempt at expo

Our Approach: To write what I actually have to do: for my own use: Then simplify it so that others can follow it:

.04 So: First thing is QATM: i.e. induction & devise t.s.Q: Assoc w. this are: design of Reference Computer & primitive set of insts. First design comes in Conc. nets — Then find Contexts to make [redacted] cjs's accessible. This involves designing a language to desc. Contexts.

Next s-induction, again design TSQ: A good representation of s-functs is needed: 3IU is ok., but other methods are heuristically useful (& impt.) List various s funct representations, & situations in which they are best used. All of them should be available for GaTTOZ's option.

After many problems have been solved, we have a set of PSTs & empirical data on how good they were for various problems. We are able to using improved updating methods of § 2.1 & § 2.1.

The set of PST's can be regarded as a language & our system has to extrapolate this language by devising a s-grammar for it. Ideally, we would give the machine TSQ's to train it to do this.

Another way to speed up the system's acquisition of a good Grammar for PST's, is give it a set of PST's that the trainer feels are very good. To more easily facilitate the systems' making a s-grammar for this set of PSTs — The trainer "factors" the set of PST's ~~to some extent. The factors~~ into a set of concepts that can be combined. The system can be "given" these factors, in the form of augmentation of the existing set of instructions, by functions or operators or subroutines that correspond to these "factor" concepts. Another way (perhaps resulting in a more intelligent system) devises a TSQ leading to the acquisition of each of the factors. If the system acquires these factors this way, it is better able to understand how to use them — how to combine them.

To do .04, we its usually best to design the TSQ, then choose a set of primitive ~~primitive~~ instructions that best facilitate that TSQ.

OOPS has shown that Lsch can be used to solve problems, but the set of problems it solved are quite different from what I'd consider to be a TSQ. The CJS's in OOPS were much larger than those that I'd normally consider. I expect the problems to be relatively close to one another

9.2.03
3TM

Spec
00: 331.40: In 331.30ff: It would seem that ~~[illegible]~~ in the norm course of TM's ~~[illegible]~~ doing a TSQ, it would consolidate its info, by expressing all of its PST's as a ~~[illegible]~~ stoch ~~lang~~.... i.e. "Grammar". (IS 331.00 – .17 Relevant? ~~[illegible]~~ — it would seem not to be).
To "first order" we make a grammar of useful PST's only. The Grammar doesn't tell how useful or ~~[illegible]~~ for what problems we should use what PST. So just the ~~[illegible]~~ PST grammar ... No "G" assignment. Tho, because it is a grammar based on positive instances only, it must be a stoch Grammar.

Good!

10 [S N] Notes for report: ① Maybe expand, 331.30–.40, 334.00ff for "Roadmap" section.

② Expand comments on OOPS: ~~[illegible]~~ Manhattan $(\frac{n}{4})^4$ factor: (there were ≈11 systems in the Tower of Hanoi soln.)

.12 ③ A critical ~~[illegible]~~ aspect of the ~~[illegible]~~ system ~~[illegible]~~ is its ability to "edit" old programs — break them up and reassemble the parts to make new promising trials. ~~[illegible]~~ In the present system there are only a few such "editing" instructions, and only one, ~~[illegible]~~ boostq, has been shown to be useful. ~~The editing facility of the system will have to be significantly expand~~ Certainly many ~~[illegible]~~ more new editing instructions will have to be added. Ideally, the system will learn to use ~~[illegible]~~ them. While OOPS does have facilities for
.20 "noticing" that certain instructions have been more useful than others, as well as facilities for defining macros, it is not yet clear ~~[illegible]~~ as to how the system would be able to use
.22 those facilities to learn to ~~[illegible]~~ do useful editing.

#5

insertion in §7.2.

Perhaps ~ quote J on use of "copy editing". That this is critical, yet the present system has to be much augmented before it could do that.

④ The OOPS mention that it only solves INV probs. It can solve OZ probs to some extent by a simple trick, but any INV prob. solver can use that trick — (it's not a v.g. trick). OOPS can be trivially modified to work induction problems. More modif. needed for OZ probs.

30 Tho .12 – .22 isn't terribly relevant to Alpha, include it anyway — Criticism of other ~~[illegible]~~ work is an impt. part of papers.

00 : This will be a Biblio listing of impt. ideas:

1) S-functs. (324.33-.40 kinds of): (333.00ff expos on S functs for Report)

2) List of refs to impt. ideas that "needed more work" 511 or 513?

.05 3) <u>2 impt "Breakthroughs"</u> ⓐ A way of understanding just what is being looked for in the O² of QATM. — Specifically, what info is being used to order the O² trials. (see 327.21 for Biblio version) ≈ (322.31 – 323.18)

ⓑ That Generating the set of useful PST's is an induction problem about [illegible] to the kinds of probs one solves w. TSQ's. (~~3~~ 325.06 – .40)

.09 3)b is impt. partly because it clarifies role of (Context) so we can quantify it. → Also note 325.28 on Genzd Context.

10

.20

30

00 Section: Probability Distributions on Probability distributions

In the QA induction of section 1, we look for probability distributions, $O^2(A|Q)$ such that ~~φ~~ eq(2.1) is maximum.

If the induction problem is deterministic (only one possible A for each Q), then ~~a~~ we need ~~a~~ a probability distribution on deterministic functions, ~~[illegible]~~ $O^1(Q)=A$. The language AZ ~~φ~~ (appendix A) describes a universal distribution of this sort, as does the FORTH-like language used by OOPS ( ). Everett to Juerg's paper.

It is often possible to use languages of this kind for probablistic

0 induction as well. In sections 2.1 and 3.1 we use the ~~[illegible]~~ probability distributions $h'(t)$ and $h'(G)$ ~~[illegible]~~ for updating Inversion and Optimization problems. In both kinds of problems it is often reasonable to assume ~~[illegible]~~ that ~~[illegible]~~ $h'(\ )$ is a monomodal distribution.

For Inversion problems, $0 \leq t \leq \infty$, so the Gamma distribution,

$h'(t) = a x^r e^{-bt}$ is a reasonable ~~dam~~ approximation.

For Optimization problems, $-\infty \leq t \leq \infty$, so the Gaussian distribution,

$h'(G) = a e^{-\frac{(G-\mu)^2}{2\sigma^2}}$ is reasonable.

~~An even simpler approximation [illegible] rectangular dist~~

20 The rectangular distribution is very simple and can be used for both Inversion and Optimization problems. It is defined by

$h'(x) = a$ if $\mu-\sigma \leq x \leq \mu+\sigma$,

otherwise, $h'(x)=0$.

x may be t or G.

For Inversion problems, the figure of merit is $\alpha =$ max value of $h(t)/t$. In which case $\alpha = a\sigma/(\mu+\sigma)$

For Optimization problems the figure of merit is γ, (eq 9): and

30 $\gamma = 2a\mu\sigma$

330.20 — .22

329.02 — 329.25

9.2.03
3TM

333.15
REV/Index

00 : (SN) In t. WON problem, we look for suitable h(t) & h(G) functions, 9:45P 9:49P

As of now, I've been looking for h functions that most accurately descr. t. cu time t or G range of t. (TSQ, problem). ~~T. criterion is~~

This is not exactly t. rite criterion. All I ~~want is~~ ~~to~~ to know which PST has t. "best" h.

Perhaps all I want is a funct that looks at a PST, prob. pair & gives a γ for Inv. or α value. Σ for OZ ~~[illegible]~~ Goo for such a function? No. of errors in categzn in past? Minimum Mean sq. error in γ or α predn? Here we are ~~[illegible]~~ mainly interested in finding PST's of ~~[illegible]~~ (near) best γ/α . .

0 T. (possl.) advantage of ~~[illegible]~~ looking for good h( ) functions, is that devising h( ) functions is a normal activity for t. GCPD.

This (.00 ff) problem needs More Thot! : No time available just now! T. Basic idea is that discovering/inventing PST's (F_i's) of hy α, γ, could be a more directly Solvable Optzn. problem. Finding t. h( )'s as intermediate step, maybe unnecy; wasteful.

.17 Is 334.00 ~~ff~~ relevant?

The new section (332.00) on PD's on PD's: Make this follow section 3 (on OZ probs).

.20 ~~[illegible]~~ We add a new eq. after eq. 2, so eq 3→4, 4→5 etc. I have to proof t. whole thing, looking for eq. nos. to be changed, relabled,

The next Big topic is t. "Road Map" 327.10 "Status of t. system: what has been done, what needs to be done.

Refs: 327.10 — on how to do expo (Also note 324.00
324.00 – .18 ← intro'n.
324.19 – 30

322.31 : That t. prob. of ordering O's for trial is ≡ GA problem of finding good Conds.

30 : : On Trng & extrapolating t. set of PST's: We can start w. solns to probs that TM ~~has~~ has obtained thru normal Srch, — using a lang. not particularly designed for general prob solving. We then could try to ~~teach~~ various PST's by suitable TSQs. The set of PST's obtained & t. (unconditional) P.D. on them could be ~~to~~ subject ~~of~~ to Extrapoln (Genzn) & ~~[illegible]~~ ultimately could be used in t. improved updating & prob solving Srch methods of WON. This assumes we got a good set of PSTS by having TM (i.e. TSQ & t. Namer) work forward. A equally good way would be to design a PST grammar by ~~to~~ "working Backward", from known Good PST's — to ~~[illegible]~~ factorization, & to continued/rec factorizations until a reasonable set of primitives is obtained.

→ 334.00

9.1.03
3PM

00 ; 282.40 : ~~T~~ from 282.40! ~~γ or [illegible]~~ for Gaussian is not used, γ is for Gaussian
α is for Gamma

T. calcs at ~~30~~ 282.30–.40 for the α slope ~~[illegible]~~ are wrong

we want $\frac{\int_0^t h'(t)\,dt}{t}$ to be max $\boxed{\frac{h(t)}{t} = \max}$

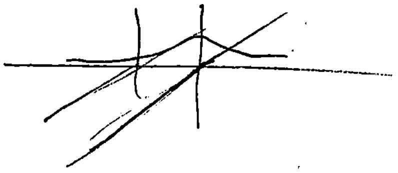
~~[illegible]~~ $\frac{h(t)}{t} = h'(t)$ If h' is ~~[illegible]~~ Gaussian dist. its not so simple. — Gamma distribution $h(t)$ is inc. γ funct.

T. α value, which is max of $\frac{h(t)}{t}$; at which pt. $\frac{h(t)}{t} = h'(t)$

doesn't seem to be easy to calculate for $\boxed{h'(t) = a\,\cancel{t}^k e^{-bt} \quad a\,t^k e^{-bt}}$

h(t) = inc. Gamma funct. See Bureau of standards for functional eqns, etc.

10 for often problems, we want first moment of ~~x~~ h'



$\int_{-\infty}^{+\infty} x\, e^{-\frac{(x-\mu)}{\sigma^2}}\,dx$ is ~~[illegible]~~ easy,

$= \underbrace{\int (x-\mu)\, e^{\cdots}}_{=0} + \int \mu\, e^{--} = \mu \int e$

so its or μ2   so its $\boxed{\mu \text{ times the } \overset{\text{mean}}{\cancel{\text{[illegible]}}} \text{ value of } \sigma.}$

20 : ~~28~~ 328.40 In eq (8) we may set

$$O^{i}(G^{j,l}) = a\, e^{-\frac{(G^{j,l}-\mu)^2}{\sigma^2}}$$

.22 in which a, μ and σ are all functions of $\tilde{G}_j$, t_j and F_l. → 329.02

in text.

.24 : 328.24 In both kinds of problems, it is not unreasonable to assume both h(t) and h(G)
.27 are monomodal distributions.

30

9.1.03
3TM

00 ~~To opt For eq( ) we want to find three scalar functions of ( )~~

.01 ~~Such that eq( ) is maximized~~

.02 : 330.22 → The universal functions of AZ or of OOPS would be ~~adequate~~ for ~~[illegible]~~ finding suitable forms for these functions.

In many cases ~~[illegible]~~ we can assume $\sigma^2=0$, ~~[illegible]~~ which makes h' ~~[illegible]~~ a ~~delta~~ δ function and simplifies our search for optimum μ and a. ~~δ delta~~

delta

For more general induction problems, we need a universal distribution on probability distributions. One way to ~~[illegible]~~ obtain such a distribution uses

10 a three ~~[illegible]~~ input universal machine. All three inputs are prefix ~~[illegible]~~ sets.

The first input is a finite string, S, that describes the function.

The second is the ~~a~~ finite string, ~~[illegible]~~, Q, the "question".

The third input is a random binary ~~[illegible]~~ sequence.

For fixed S and Q, we have a machine with random input — inducing a probability distribution on the output, ~~A~~ just as in the usual universal probability distribution.

In the present case however, the S and Q inputs may not define a universal distribution on the output.

The foregoing formalism ~~[illegible]~~ describes a universal distribution over all possible probabilistic relations between Q and A. For every ~~[illegible]~~ describable probability

20 distribution between Q and A, there exists at least one value of S that ~~[illegible]~~ implements that distribution.

Tho it is possible to realize a ~~[illegible]~~ 3 input device of this sort using ~~[illegible]~~ AZ language, it is easier to implement using the ~~FORTH~~ FORTH like languages

.25 used in OOPS.

30 lines .00 - .01 need ~~[illegible]~~ Clarifn.

~~In eq(8) we may ~~[illegible]~~ say $O(G^{s,l}) = \frac{a}{\sqrt{2\pi}\,\sigma} e^{-\frac{(G^{s,l}-\mu)^2}{2\sigma^2}}$~~

~~in which a, μ and σ are all scalar functions of $\tilde{G}_j$, t_j and F_l.~~

~~We want these functions to be such that eq (8) is maximized~~

5.1.03

3TM

Thu

0 : 329.40 : 2 items: ① Grand plan: ② s funcs/d funcs.

②: AZ ~~major~~ gives universal PD on d funcs.

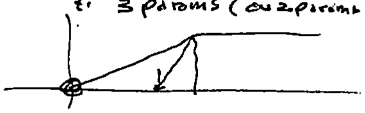
OOPS gives " " " " "

In theory a pd on funcs (like $O^2(A|Q)$) can be regarded as a d-function A, Q.
≡ s funct

~~So~~ AZ/OOPS could give a univl. pd on s funcs of this sort.

Hvr, many funcs are not 1) ≥ ϕ 2) ~~[illegible]~~ $\int_{\text{Range}} < \infty$. (i.e. normalizable).

For h funcs, we ~~[illegible]~~ may confine ourselves to monomodal distributions ~~[illegible]~~ such as ~~the~~ Gaussn or Gamma or ~~[illegible]~~ box car functions. We then have to simply optimize the 3 params (or 2 params if $\sigma^2=0$)



0 Discuss 3 I U. Advantages: Universal; always normalizable; easy approxns. Since its univl. it is incomputable exactly, but ⟶ ↑.

~~[illegible]~~ Probability distributions on Probability distributions (See 224.33 on "s-functs")

In ~~[illegible]~~ QA induction, ~~eq — looks for~~ we look for ~~a~~ ~~[illegible]~~ probability distributions $O^2(A|Q)$ such that eq (2.1) is maximum.

If the QA problem is Deterministic (only one possl. A for each Q), then we need a probability distribution on functions, $O^2(\cdot|\cdot)$. The language AZ (appendix ..") describes a universal language of this sort, as does the ~~The~~ ~~language~~ Forth-like language used in OOPS ( ). ~~is also of this [illegible]~~

unclear)

~~For probabilistic QA problems the [illegible] languages usually [illegible]~~ it is ~~sometimes possible to use these lan-~~

It is sometimes possible to use languages of this type for probabilistic induction ~~[illegible]~~ as well. In Sections ~~2.1~~ 2.1 and 3.1 we ~~[illegible]~~ use probabilistic functions, h(t) and h(G) ~~that are used [illegible]~~ in updating ~~[illegible]~~ Inversion or and Optimization ~~problems~~ problems.

sections 2.1 & 3.1 are not directly about QA. But updating INV & OZ does use funcs of prob. pd.

2 . insert 330.24-.25 not so.

(~~Since we are only interested in the relations~~ between ~~centre of the~~ moments of ~~these~~ These functions, ~~we may assume them to be monomodal distributions.~~ Inversion problems, for Inversion problem In Inversion, the t of h(t) ~~[illegible]~~ must be ≥ 0.

~~In the [illegible] of h(t) we have [illegible]~~ use so the ~~[illegible]~~ Gamma distribution, a reasonable approximation.

30 $h'(t) = a\,x^n e^{-bt}$ is ~~[illegible]~~ our ~~[illegible]~~

~~for h'(G)~~, ~~the [illegible]~~ In optimization problems ~~[illegible]~~ h(G) is ~~[illegible]~~ unlimited: $-\infty < G < +\infty$ so the normal distribution,

$h'(G) = a\,e^{-\frac{(G-\mu)^2}{2\sigma^2}}$ ~~is~~ is ~~[illegible]~~ a reasonable approximation. ~~[illegible]~~

See 282.30 for calcn. of γ for Gaussn & Gamma.

2 u · σ · 2

~~We might~~ In ~~[illegible]~~ either Inversion or Optimization we might use the ~~[illegible]~~ vector of the distribution:

$h'(x) =$ ~~[illegible]~~ a if $x > \mu-\sigma$ and $x < \mu+\sigma$ ~~[illegible]~~

otherwise, $h'(x) = 0$ ~~[illegible]~~ $\boxed{\alpha = \text{~~[illegible]~~} \quad a\,\sigma(\mu+\sigma)}$

give eqn. x can be t or G.

Give eqn. on 330.20

For the ~~[illegible]~~ of Monitor, αx, of inv probs

8.30.03

3TM

REV.20

2 IMPT Breakthru ideas in TS Qurrity.

Itinerary
State of t. System : What Has Been Done; What Needs to be Done
Master Plan
Road Map.

10 : ~~3~~26.40 : Various methods of "correlating" these potts w. problems, & problem Classes.

In t. Case of OOPs : Looking at Tower of Hanoi & that 1 param. can have any integer values Could suggest that a recursive soln. mite be looked for. One mite Analyse it logically; "If I had soln for n (or n & n-1) could I get a soln for n+1 from them?

0 : 324.32 ("Roadmap") 324.19 322.31 ff ← This is about how O² srch is very ≈ to GA & clarifies GA as well as O²srch! is about t. Road map (not such a good term!) : "Master Plan"? State of t. system (Resource) what's been Done, what needs to be done.

Perhaps start out by deriving TSQ construction (as a thing that needs be done) Refer to sections in Sol 89. Discuss Making of Conc. nodes, Also finding suitable contexts: How a context lang must be designed : How it leads to short derns of t. sequence of prob solns. (T. things First cues are functions of are not "free" — they have to be ordered so that each has pc cost.

~~Discov~~ That PST's have to be found, listed. — That TSQ writing frams trainers as well as TM.

10 (325.40 5 pac) : So 2 General "Breakthrus" : 1) T. similarity of T PST induction problems & t. General probsolving routine of TM. (325.06–.40) 2) 322.31–323.18 A very General

21 discussn. of How to ~~#~~ Get Good O²'s for [redacted] QATM. T. General idea is ~ that of GA : Use of Past set of O², Raw corpus G's, to extrapolate to t. new O² & its corpus. Ideas of Mut/cross (suitably Genzd) & mut/cross as examples of induction w. SSZ of 1/2 ... then idea is to use SSZ = total corpus. Note that this is weak form of Optzn : Its trials try to be "~" to previous trials : No ~~[illegible]~~ attempt at true "optzn". But this Discussn seems to make O² updating ∈ a "well defined problem".

Just what Do I need for report | see 292.33 for list.

Claims

0 → 1) Road Map.

2) Table of Contents

$\frac{pc}{cc}$ srch = GHT srch.

3) Insert 321.10 –.17 ← Not obs: ~~[illegible]~~ This is to be $\frac{pc}{cc}$ srch = GHT srch.
If could be inserted, then explain why updating during srch is better.

4) More Comments on OOPs & how it differs from α.
Mention: it only works INV. probs : It can work Induction probs by Lsrch & OZ problems — sort of,

5) I have several lists of Desired modifns of t. Report (292.33 is one)

6) ~~Intro~~ : Claim list. : Why This is Good approach : Mention other General induction methods

o.k. — 7) How Smrbs work 3IU ~~and~~ 324.33.

8) Fix equation nos. ... omit ~~[illegible]~~ Original eqn. nos.!

P.18 : §5.1 line 3
eqns 6 & 7 !
primary 6 & 8.

8.29.03

TSQ writting .00 ↓

CMI ≡ Coding Method of Induction.

$\frac{2\,kk}{40}$ = 50kby/p.

00 (3pac 325.05) TSQ writing: Start w. defincs (MaR): Write solns to several mildly diff problems (related problems) factor down solns to primitives.

Also, in addition to listing impt funcs to lrn (some funcs identify types of Math object) — Do much on Context of each funct Rx calls when it is used.

This is pretty t. main problem: It ~~gives~~ tells us what primitives

09 to use & how to combine.

> Unfortly, much "Context" info maybe created by "logical reasoning", which is only available "Later" (I really don't know how much Logic TM has to know before it can do t. kinds of reasoning needed for "Context" determination — Or for other kinds of use of "Logical reasoning" in TM.

I ~~Think~~ ~~[illegible]~~ .00 ff should be t. main approach to writing TSQ's. T. reason is, that various aspects of it relate to much of t. rest of TM's design.

Not only t. concs & their Context, but perhaps methods of search & any other

30 aspects of prob. solving. E.g. writing t. TSQ this way should alert me to **Non-Lsrch methods of solving problems**

So 2 BIG PROBLEMS:

① **TSQ** writing {Conc nat + contexts of concs.}

② **PST** finding/discovery/implementation : Designing PST Grammars; factoring set of PST's

Q: A Conc nat seems Great for Lsrch: But is it also useful for other prob. solving techniques (PST's)? It may be that Lsrch is t. general

30 "glue" that holds the system together — t. "recatatif." "recetatif" = t. betw. opera songs.

In general, Lsrch searches over PST's for INV & OZ probs. Normally, its ordering of PSTs isn't very good — it just uses a pr pd of all functions (say via AZ). But

One simple Class of PSTs is just a **pd on Tokens**: This may include arbtly complex tokens. This "pattern" can be like BoostQ: One info stores just t. increments [Δ numerator, Δ denominator] in tokens (+ new defns) that were used in a prob. soln. These patterns could be combined linearly, w. perhaps an additional "Backgnd" set of all of t. cnsts. — or enuf to make t. set Universal. To ~~use this~~ use this PST, we just use this pd on tokens for Lsrch. (Perhaps we should be allowed to make new defns, in t. Lsrch. A pd on a single token can be regarded as a "pattern" — 327.00

WE MAY ALSO VIEW This as a kind of CONTEXT

4-29-03

3TM TSQ writing.

.10 : 324.40 : SN In "Comp Sci" there are various "Algos" like FFT, b trees, various sorting Methods. TM should be able to "discover" any/all of these.

A Good source is Knuth's 3 volumes: He also has problems that give "degree of Diffy" — ~~#~~ One might use these params to help make a TSQ.

.04 : (Spec 324.40) We have 2 kinds of Grammars: ① — ("derived" Grammar/language) One in which every ~~production [illegible]~~ step

.05 of a derivation gives a ~~[illegible]~~ a2 ; ② one in which only "terminal leaves" are a2's. → Spec 326.00 →

.06 : 326.00–09 : Describes Initial TSQ construction: Starting w. Several ~~[illegible]~~ related, somewhat diffy probs & ~~[illegible]~~ find sub-concs. (hopefully, many common subconcs a/o sub$^{(k)}$ concs. & assoc contexts.

0 [The problem of lrng & Extrapolating a set of ~~[illegible]~~ PST's is identical to .06ff.] I thot of it in terms of factoring the ~~#~~ PST's into common Concs & ~~common sub$^{(k)}$ concs & find~~ — but instead .06 ff is just not relevant to PST lrng as to learning to solve any set of diff. probs.

By Regarding "PST's" & "probs" as similar, we can perhaps get cross-fertilizn. ~~w~~ with regard to ideas on how to work them. In the case of PST's, I was thinking of having trainer "factors" an initial good set of PST's. ← Going Backward. An alternative way would be to have TM learn all of the PST's as "terminal probs"

20 in a TSQ. ← (But to write this TSQ, usually one has to start out by the Trainer "Working Backward". ↑ Going forward.

Alternatively, I might take a set of "2nd Terminal probs" of some diffy, then have trainer factor them, & factor more the factors, etc. } A normal way of writing TSQ's.

[illegible] In either Backward or forward implementation, TM could help the trainer, ~~[illegible]~~ if TM has large capacity Search capabilities — So that the "factors" could be "large grained" (fewer factors$^{(k)}$)

N# Goographic .com.

.28 Genzing .29 "Context" Re: "Context" (≡ "when to combine 1 or more Concs) & how : Context can be generalized to be any d. or s. rule that tells one how to do this.

0 ~~[illegible]~~

31 For both TSQ lrng & PST discovery/lrng, it's not really necy (At First) to have TM Work ~~any~~ problem or devise any PST's: It would only be necy to compute the pc's of such solns (≈ CJS; but with contexts included). Hvr, after these ~~TSQ~~ Concs & PST factors & Contexts have been put into TM, we ~~expect~~ that it should be able to solve new problems & create new PST's out of order of & more diffy than the ones "programmed in".

Spec 322.20

8.28.03

3IM

00:323.40 : Summary of "STATE of TM"

on How to list O's in ≈ pc order.

.01 T. way it works: STarts out w. QATM: Uses ideas summarized (in 323.~~40~~ –.18) to get good O^2 functions. Eventually, TSQ gets to point where it can find good h( ) functions for WON search — Then it eventually switches to WON search.

There is a ~~necessary~~ Part Missing:

For WON we need a set of PST's. At first, a smallish set is inserted into TM. ~~Next~~ Next, it is factored (Mainly by trainer), & TM (a/o Trainer) makes a grammar for the set & is able to genz. the set of PST's. T. Gramm. is, ideally, Universal, so all conceivable PST's are describable by the Grammar. T. entire process of Generating the Grammar, starting from the Initial set of PST's & their partial factoring, to generating more example PST's: to deciding which ones to try on new (& old) problems, to finding way(s) to generate new PST's that are likely to have hy "slope (= max $\frac{h'}{t''}$)", & finding PST w. ~~max~~ max or (near max) slope, must be investigated.

325.06 – .40 Is a V.G.! Approach to Generation of appropriate PST's !!

0

I could start w. optzn methods, since this is the commonest PST, i.e. ~~a~~ devise a Grammar a/o factorizn. of PST's.

.18

.19 I really need a more detaild itinerary/roadmap;

20 (.01) is o.k., but we can't ~~do~~ apply QATM to the WON problem until we have a Set of PST's. We could insert a reasonable set of PST's for oz problems. Then have TM do Lsrch on them for/optimization solving problems. From the data generated on PST_z, $prob_j$ (failure times, soln $time_{zj}$), QATM could, in principal, obtain "h" functions for any new (or old) PST working any new (or old) problem. To do ~~a useful~~ an effective job on this QATM would have to be pretty smart in the relevant domain — & have to have a suitable TSQ to get those "smarts"

Next, we can expand the set of PST's ~~$~~ via the trainer — who would also partially factor the PST's: TM's being able to use this info in an effective way... ? could it?

30

32 For Expo: T. detaild roadmap would be a good thing to put in the report. (327.10)

33 Kinds of S Functs: 1) a IU (2) P(A) = ~~a~~ a f-funct of params of A : e.g. $P(A) = a\,t^r e^{-bt}$ a, r, b are functs of other params of A; Note that "t" is a param of A, also.

3) S-Langs, Born seq. S = $p_1 \to N\,V$, $p_2 \to N\,V\,N$: N = $p_3 \to$, $p_4 \to$ V = p_4, p_5 This is a way of assigning p's to A's: Is it a case of (2)? In 3) "N, V" & "N V N" are partial den.

~~what~~ does this correspond to anything in (2)?

4) Monte Carlo representations: (3) is easy to express as Mt Carlo, (2) is "not": why not? (1) is easy Monte Carlo. (325.04 space)

8.27.03
3TM

(P)SMP : (Potentially) Smart Machine on the Planet
PSME
↑ Earth

.00 : Basically, t. GA model of 322.31-.46 captures t. idea of how this initial QA induction is done! We used t. known (O^j, G) pairs to create a "Grammar" that assigns a G distribn. to potential Cands.
T. System also includes a Method for finding promising cands (by expected G),

SN We have a G funct. based on prob corpus? Is it linear? (we would need to know this for optimum Optza.) ≡ Has it been "Linearized"?

So that's it, & t. Q is, to devise Good Grammars & Evaluate .10 Grammars that are now ~~being~~ used in GA & ~ problems.

.11 This Seems to make t. listing of O^j cands in pc order, a "Well defined Problem" 322.31 – 323.10

The use of "Context" of various kinds has to be integrated into this as an aux. source of Info. → See 325.28-.29 for a very General defn. of "Context".

T. stuff starting w. 321.00-.09 is V.G., then T. Discn of 322.00-.30 has some good ideas, esp t. discn about GA (starting at 322.31 and → 323.20 makes ~~@~~ early QA induction a "Well defined problem". .18

.17 In this form, QATM ~~is~~ is able to induce h functions that are used in INV & OZ problems by getting a & μ (& perhaps σ²) of $h(x) = e^{-\frac{x^2}{2\sigma^2}}$ & $h(x)\ x^r e^{-bx}$

$h'(x) \sim a\,e^{-\frac{(x-\mu)^2}{2\sigma^2}}$; $h'(x) \sim a x^r e^{-by}$.

.20 – for OZ probs | for INV probs.

claims! 219.28R
T. (potentially) Smartest Machine on t. planet
(P)SMP
PSMP

.21 In this (.17-.20) case TM just finds μ, a, σ² values ∋ [illegible]

∋ $a_o^j \prod O^j(t^{j,e} \mid G_j, s_j, F_e(\cdot|\cdot))$ is max (max likelyhood) } § 21 of report – P13.

$h'_e(t) \equiv O^j(t \mid G_n, s_n, f_e(\cdot|\cdot)) = a\,t^r e^{-bt}$

a, r, b are related to a, μ, σ² in a ~~[illegible]~~ simple way.
See 282.30 –.40 for this

211 $\sum_{\mu} \leq \rho(U_i) < -2\ln\rho_\mu$ for Marcus

207 why My Normeniz [illegible]

192-252 [illegible]

.30 : 8.28.03 T. problem at .11 would seem to be a "fairly" well defined induction problem. ("fairly" because "Context" is a fairly "open" term) — Hvr. I assume that TM is not nearly smart enuf to work on this problem as such. (.11) asks for ad hoc hevrs to do this — ideas usually derived from my observations on Human prob. solving & induction.

SN Re: Normed Approxn to Unnl. Df! ~~[illegible]~~ In any approxn., one can't know t. normalized form. We haven't gotten all codes for all continuations, so our approximate normzn will ~~always~~ ~~always be~~ often be too hy, or too low. It still may be a better bet than t. approximate unnormed distrbn!

→ 3.24.00

Murry Danofsky
202 Medford st. Somerville
617 625 8960

00 : So t. main Bottleneck for the report is just how t. sfunct (& perhaps d-funct) search is done.

One way: Using Lsp's rule! ~~After~~ we discover O^1 for Q_1; then, looking for a O^2 that works for both Q_1 & Q_2: Th. pc's of t. tokens used in O^1 are all (at least) doubled: so if there are k ~~tok~~ tokens in O^1 this is a pc ↑ of 2^k for trials using all k tokens: and even more if O^1 uses a token > 1 time!, We also have t. "definition" of O^1 as a token (perhaps).

321.00–.09 is a good way to make new trials "n" to older successful trials.

Another Good trick is J's "patterns": These ~~[illegible]~~ can be subsets of instructions. One way to Use them is to "Boost" w.r.t. them ("Jurgen"). A subset of insts is a ~~[illegible]~~ usefully defined

10 object, if we can use it to ↑ pc of previous corpus (including, of course, t. proof defining it).

"Context" is another heur that modifies pc of a Token at a particular point in t. coding of an O^j trial. — "Context" can be of various kinds, of various generality, connecting nearby ~~[illegible]~~ as well as more distant aspects of t. past codes & corpus, to t. pc of t. current Token trial.

So: I know how to start out w. 1 or 2 Q's & build a O^1 "from Scratch".

The next trials for Q_3, Q_4 ···, will be (mut (~~[illegible]~~ cross) of past O^j's. For pure mutation, (SSZ = 1) we want trials in "info distance" from t. single known soln.

for 1 or 2 Q's, We may want to oversearch, to get a larger population to start with.

1) problem of getting t. mutations in rough pc order. Breaking up functionals, etc.

20 Simply substituting One token for another is unlikely to give a mutation that works both with old problems & with a new problem. This is because t. programs are "Minimal"; — No! note

(Well maybe substitution could still work. It may have about = pc; so it was selected, but mutations could give pc's not much lower or even same)

So first order mut is subs of any ~~[illegible]~~ token k-input function by another k-input function.

Second order: Substitution of branches by newly grown branches (Growing a branch is normally expensive, but in our "funny" branches are very small, ∴ "possible").

Third order: substitution of subtrees by ~~others~~ locally generated subtrees of same I/O capacities.

When we have several models: for 1 Q or for several Q's! We may be able to express a certain subset of t. models as having a common part (or maybe a few common parts) and various subtrees

0 stuck on the common parts.

31 Th. problem is identical to that of GA: We have a population of known G's: to extrapolate — to get a pd. of G for each describable cand.

In t. case of O^j's we have a lower bnd on G for each: if we search more, we might find more codes.

§5.3 (p18) of report is a reasonable discussion of this as a way to ~~[illegible]~~ "start off" t. system.

Perhaps use GA for this beginning of t. system. T. GA could be made to ~~[illegible]~~ improve itself by varying its own params & by improving t. mut/cross functions. ~~[illegible]~~ This last is essentially induction on t. set of ~~[illegible]~~ cands that have been evaluated.

8-27-03

3TM

0: 320.40 : One way to relate the new O^j trials to previously successful O^j trials: The new trial can use O^j or parts of O^j (sub-trees) as part of itself. This, to some extent, ~~implement~~ can implement OSL. Also, if certain "parts" have been successfully used in several previous O^j's, this is a "Big Plus" (Common previous successful (Trees subnets)) Boosting is a v.g. similarity type: "use insts of previously successful trial, + subsets of insts + perhaps other insts. However if a pgm is in the more distant past, boosting should be much less likely to think of it as "close". The "attenuation rate" w.r.t. ~~pas~~ distance into past — at first inserted by trainer; later it is adjusted by TM to reflect its experience.

.09

0 — (SN) Rewrite section of report on WON search! — This way: When max slope order is obtained — one search method is ~~Gall~~ Graub. House search — just search in that ~~order~~ (max slope) order. Then introduce || updating as significant improvements — ~~etc~~ — At least 2 reasons why ~~both~~ it's better: ① uses ~~recent~~ info on recordering ② Avoids correlated and identical trials.

Disadvantages; ~~the~~ improved meth(s) ~~do not~~ are not as heuristic schemes probably do not have property of "constant factor worse than best". (The "constant factor" can be 2^{100}.) except for availability of "frozen" pgms via "boosting" — or other "patterns" can be available.

17

One BIG Difference betw OOPS & α: α looks for O^j to solve all probs up to now. OOPS starts anew for each problem. (I think this is related to its solving INV probs only) α can simulate this by using "indices" for each problem type — which it later learns to do without.

0

In OOPS: After it has worked a problem, it tries to work subseqt. probs by "continu" of the earlier soln(s). To do this, it has to look at the problem & decide whether continuing is needed. In my latest 3IU model (319.32 - 320.16 320.01 - .08 in particular) (Then 320.17 - .19), any "continuation" of the old soln would be categorized as R_i-type inpt. Note hvr OOPS is not a $\underline{\underline{S}}$ ~~pa~~ function discoverer — only d functions — so "R_j" is always part of O^j.

28 (S.M.?) —> Also because it solved "T. Identifn. problem".

(SN) Gold's "induction": I had previously discarded it because it didn't consider ssz — But it can be regarded as a mechanism for d-induction, in which it gives predns based on any particular corpus: It has models in some enumerable order — for any finite corpus, it picks the first consi model in the enumeration. It does not consider $\underline{\underline{CC}}$, hvr. I was put off by his claim of "Identifn in the limit" — which looks like larger (possibly ∞) ssz (Also "T. Identifn. Problem" (so 2 bad errors)) (Tho for any finitely durable μ, there will be a ~~point at which~~ finite length of corpus that is "adequate")

8.26.03

3TM

.00 : 319.40 . be part is "acceptable"

01 { I had thot about t. Q of "What should trigger backtrack" In R.W. we usually have some idea as to how much accuracy to expect from "t. current Model". When this is signifly violated, we start hunting for a new ~~M~~ Model/theory. In Physics, if Theories even form so that only "expt error" can give results not exactly same as Theory. But "Expt. error" can be arbly large.

This ~~[illegible]~~ trigger threshold for backtrack (also how much time we spend on it — how seriously we take it) may be part of "Meta lrng" Our "experience" in similar situations in

08 t. past. (Mind full of t. Grue soln & ~~t~~ t. problem of Grow proby & "How many coins" problem).

0 For "Backtracking": Just how is this done? One way is to search exhaustively over O^j models in pc order, using a kind of AZ model ~~to~~ give apps. ~~Anoth~~

Another way is by Mutation or/biased crossover (w possibly >2 parents). T. idea here is that after trial O^j's have worked ~~awhile~~ on part of t. corpus w. some success, one wants to use this info to essentially modify apriori pd on O^j. Mut/cross is

16 One (rather large) set of ways.

17 (319.32 — 320.16 320.01 — .08 in particular) is o.k. "in spirit", but I still need more detail. We start by writing "O^j": I guess O^j is "over" as soon as it looks at Q_i. Any subsequent

19 code inputs is part of R_i ("by definition"): R_i is done when output stops ("normally") but

0 Sometimes we can have alternate pending ~~of~~ part of A_i & asking for more R_i — so sequentially, we have a pc assoc w. each prefix of A_i (This is always true for t. complete Universal QA distribn)

Since, initially, we are only interested in R_i codes for a specific A_i, we will quickly reject any R_i that produces any bit that deviates from that A_i.

"Normally" we could have code reading from O^j & from Q_i, but we are devising this formalism so that this is by definition, impossl. It makes it possl to divide up t. ~~c~~ code input ~~[illegible]~~ ~~[illegible]~~ uniquely into a O^j part & a R_i part.

I'm not yet sure t. fory does this, — but say it did! We would have to find a O^j that ab initio, was able to do all Q_1 to Q_n correctly. Rather Unlikely! — and not much in t.

0 Spirit of TSQ's. what we want is a O^j that will work Q_1, then a "small modifn" of this O^j that will work both Q_1 & Q_2, then ~~another~~ another "small modifn" that will work Q_1, Q_2, & Q_2 etc. If I want to do "small Modifns" (mutation/cross) then Lsrch → AZ would be good, since such ways can be used to make small modifns that are always meaningful.

Juergen arranges so that if O^j works Q_1, then O^j must work Q_2 or an extension ~~etc~~ O^j will work Q_2.

I was doing INV problems only (I think — but I'll have to go thru just what OOPS does to see if that's true!).

8.25.03
3TM

00 : So T. Q is: How to efficiently search T. ($|O^j| + \sum_i |R_i|$) space.
One approach: Start w. O^j, ~~[illegible]~~ R_1 search.: we want to find a O^j & $|R_1|$ is not very large. —
when we find such a O^j, we test all subsequent Q_i's on it. For each Q_i we find ~~[illegible]~~ one or more
of T. shortest R_i's. Say we had an O^1 w. an O.K. $|R_1|$. Then we see how large R_2 is. If $(|R_1|+|R_2|)$ seems
too large, we change to O^2 & try ~~[illegible]~~ Q_1, & then (if R_1 is O.K.) we try Q_2. If we do ~~[illegible]~~
worse than ~~[illegible]~~ O^1 we try a O^3 (or possibly go back to O^1).

0 : NIPS 138.29 : $\boxed{8.26.03}$ continuation of $\boxed{\text{NIPS 138.23 – .29}}$ — This turned out to be N.G.: finally, what looks like a reasonable soln is ≈ 322.31 ff (≈ GA). Induction w. ssz = 1, 2, 3 … n.

If we have already coded O^j, A_1, we have "equiv" code length
≈ "$|O^j| + |R_1|$": By working on O^j a/o R_1 we get ll codes that ↓ "$|O^j|+|R_1|$". T. only way to
↑ this sum is to ~~have~~ start ~~code~~ to try finding code(s) for R_2. So with just O^j & A_1 to code
~~[illegible]~~ we can only ↓ equiv. code length. When we think we've spent "enuf time" on O^j, A_1
(i.e. T. amt. of ↓ in equiv code leng per unit cc is small), it's time to start coding A_2!

17 T. "Top Goal" is ≈ "Mincode for $O^j + [A_i]^n_1$"
A way to do "Top Goal" (.17): Do a "rough code" (first code found for each A_i) for
O^j and $[A_i]$, then go back & try to reduce code lengths by working on O^j or the individual R_i's.
20 Getting any O^j code for T. entire corpus is a Big Job!
Perhaps ~~we~~ start w. O^j: try to code as many A_i as possible until ~~[illegible]~~ T. total code length
"per A_i" is too large! Then we Backtrack/switch O^j trials. What is level of "mean code length ~~[illegible]~~ per A_i"
23 to ("expect"/use as threshold): Maybe find it ~~from~~ from "Previous Experience". (see .24–.27)
24 We could start out by searching for min ($|O^j|+|R_1|$) ~~[illegible]~~. When we get to pt. of "diminishing
returns", we estimate cc per −Δcode length that was obtained. This gives a rough idea
27 for T. "Previous Experience" of .23. In case of $|O^j|+|R_1|$, unclear as to whether "mean
code length" should include code for O^j. $|R_1|$'s code is pathologically short because O^j was somewhat
30 A.H. – designed for it. But if we did an Lsrch for min ($|O^j|+|R_1|$) then presumably T.
code length for $|R_1|$ is "reasonable".

32 When we code a new A_k, we find our first code by Lsrch. If it is "poor" or we find ~~code~~
no code w. existing CB, we backtrack to a new O^j ~~of~~ Perhaps better to use mean bcost per A
as backtrack trigger rather than just "excessive" bcost for latest A. ~~[illegible]~~ (lack of a code for latest
A will ↑ mean ~~[illegible]~~ "bc per A" & trigger Backtrack to change O^j).
If T. new O^j trial gives worse mean ~~[illegible]~~ "bc per A" we either try a different O^j or
go back to first O^j & continue ~~trying BAM~~ work on T. "unsuccessful/for now" last A problem.
In early training (Q_1, Q_2, etc) T. trainer will have ideas on how large

8.28.03
3TM

00 : 317.40 A possl. way: Have 2IU ~~[illegible]~~ "simulate" 3IU.

Use t. prefix code methods of 316.22 ~~ff~~ ~~to~~ –.35 : 317.15 ff is prob'y. N.G.

We take all of these codes: for each Q we get a bunch of A's coded w. assoc. wts. We use t. wts

03 to get a pd, which we normz. This sums over all codes.

Now, is .00–.03 equiv. to 3IU? In .00–.03 is ~~there~~ there any interaction betw. t. diff. QA answers?

05 In 3IU, there seems to be much "interaction": ↓ .055 The "wt" of O^j ∝ t. pc that O^j assigns to t. corpus. for new predn

I don't see how t. 316.22–.35 model is doing anything like that (.055)

T. way ~~own~~ one will actually use 3IU is as follows:

09 —— For each Q, we get t. shortest code (or sum of a few codes) for ~~[illegible]~~ more t. ~~[illegible]~~ rite A.

10 We ~~mult t. pc of t. code [illegible]~~ add t. pc of t. original code to sum of shortest derns of each A. We pick ~~[illegible]~~ original code ∋ this Σ is min. To do predn, we use the "best" original code & use many R's to get many A's & their pc's — &

15 we normz.

In ~~[illegible]~~ finding shortest code for each A, we may "oversearch" & get more codes: this could ↑ pc of "correct" code or ↓ it because of pc's of other A's & normzn. Actually, we ~~will [illegible]~~ don't need to oversearch in order to get lots of pc's of other A's — we may get ~~these other~~ many others before we hit t. "correct" A.

20 There is a reference in Nips 33 (or 133) on just how to do searches on 3IU.

If we do t. corpus incrementally ~~to~~ make ~~[illegible]~~ a set of predns for each Q, then we will have a lot of A's for each Q & their assoc pc; & we can normalize them.

We may be able to do .09–.15 using 2IU — in which case its not much diff't from 3IU

A way to use ~~t.~~ OOPS model:

We find a O^j (≡ seq. of instructions). We put in Q_j, there may be some outputs, and t. machine may stop. To this is bad, ~~it is [illegible]~~ this is our only output from Q_j — it is ~~[illegible]~~ given pc=1.

& We hope it will ask for ~~[illegible]~~ input before printing, then ask for more, ~~[illegible]~~, print more, etc. &

30 —— eventually stop — w. ≥ A_j. Using diff't a.o.x inputs, we get diff't A_j's,

— We sum t. ~~[illegible]~~ "p.m. pc's" & normz to get pc's of A_j's.

32 — It may be that NIPS 137.29 ff (but 137.00–138.30 is longer context) is adequate for 3IU & OOPS as well! A search strategy over O^j, R_1, R_2, R_3 ... is discussed/described,

— May be adequate (?). — I don't see its "Adequate" when searching to minz $|O^j|+|R_1|+\cdots|R_k|$ its not clear when we ↑ k (≡ add a new Q).

Nips ~~[illegible]~~
137–138 v.g. on training 3IU
137.29 ff is main routine.
I think this turns out to be Vacuous!
See ~~[illegible]~~ 319.10 ff

I am disappointed w. 137.00–138.29: I remember thinking I had a really good search routine: but its not at all clear as to how to do it.! Searching over $|O^j|+|R_1|+R_2|\cdots$ directly looks like an excessively large space!

It would seem excessive to search over ~~[illegible]~~ $|O^j|$ plus maybe two R_j's in one search. Total code length would be too long.

00: : **IN Report:** Point out that ~~a~~ functions that get a ≥ M from PST, problems are a very small

.01 functs" & need lots of suitable trng before they can do anything useful —

Also t. often problem of finding good PST for a problem (≡ hay $\frac{h}{t}$ slope) ~~can~~ can involve (inventing / creating) new PST's, & ~~may~~ can be a very difflt problem — also involving much ~~[illegible]~~ "advanced" training of a viable sort. So a function to find Max $\frac{h}{t}$ for a largish, but "**finite**" **set** of "already created" pst's, this may not be such a difflt problem.

When these such problems are diffk, one has to decide how much time to spend on them. — use "50% soln".

10 Deciding when to switch from normal Lsrch for WON, will be difflt — we may not need v.g. functs (of .01 – .02) to be better than Lsrch — But we can periodicly try ~~on~~ our functions & compare them w. (such results — when they are clearly getting better than Lsrch — we clearly switch over. In transition period, we may use both in ||.

.15 : 316.35 : T. way to **2IU** seems to work (if indeed its nophsafell)!

We just try all partial functions on t. complete set of Q's ① If a functs gets all n A's correct, we retain it & note its wt. (≡ aprip). ② If it gets some rite, & says "no reply" for rest, it gets its wt as in ① & is used for predn. ③ If a ~~[illegible]~~ partial function gets wrong answer ~~on~~ for any of t. n Q's, it is discarded. ④ to get ~~predns~~ ~~[illegible]~~ future predns for ~~[illegible]~~ a new Q, we use all functs w. non zero wt,

20 & obtaining a dist. over possl. A's, & we ~~[illegible]~~ normalize to get pc's.

Whenever we find out rite answer, we destroy all functs giving wrong answers. This leaves us w. only a few

.22 consis functions so we have to Backtrack to get more of them. → [Hm, it t. Theoretical [cB=∞] case, backtracking is unnecy, since we have already tried all possl. Backtrack functions]

[$ Q? — Is .15 – .22 ~~[illegible]~~ really equivalent to 3IU? → .15 – .22 ~~[illegible]~~ looks like a real ~~embodiment~~ of a SUMAC — w. all its ~~[illegible]~~ "backtrack" diffys.

Consider: In 3IU, a given usable k_i will sometimes code t. "rite" A_i (that occured) & other times it will code a A_i that did not occur. This seems Quite Differnt from .15 – .22!

.29 So it looks like I don't have any Correspondance betw [2IU & 3IU] — NB This doesn't mean 2IU is wrong: it simply means that the correspondence betw 3IU & 2IU is incorrect. : LATER!

30 **One** way to look at 3IU: Its all these || ~~[illegible]~~ coded objects — each w. its own PC. ~~[illegible]~~ Many of t.

31 coded objects ~~[illegible]~~ are incomplete" i.e. they have ~~[illegible]~~ several of t. A_i missing. This is not assigning pc=0 to them, hvr. A "coded object" is a subset of t. [A_i] set — with one or none ~~[illegible]~~ for each i ($i=1/n$)

3IU & 2IU are significantly Different defs → 349.40

A particular corpus is a set of n A_i's one for each # i ($i=1\ldots n$). Do t. set of "coded objects" assign unique pc to every particular corpus? They do but via only 3ID ~~[illegible]~~ indirectly.

00: 315.40 : In general, t. set of R_j's that are assoc. w. a given Q_j will ~~not always sum to~~ sum to < 1, because of partial recursive funcs, so we will ~~be~~ always have to Normalize w.r.t. t. cases that have "come in" thus far (up to current CB).

{ So, for 2IU, for each Q, we use a wtd ~~E~~ sum (wts = pc's of codes) of predns for each Q — then we normalize to get D.F. of Answers. This is / functionally adequate — i.e. it's able to ~~give~~ predns.
— But how good are t. predns? ~~&~~ We may actually be able to show they are identical! — code for code!

T. way we show this: Any 2IU code for t. entire corpus (w. self delimiting exclusions ∴ partial functions*) can be regarded as "part of" a 3IU code for t. entire Corpus ("). A component of!!

Somehow, we have to take the #1 code of 2IU & ~~[illegible]~~ break it into 2 parts that correspond to #1 & #2 inputs of 3IU. Those are self Delimiting

So, as of now, t. 3IU & 2IU ~~codes~~ models are slightly difrnt: i.e. ~~[illegible]~~ 2IU has a single S.D. code & 3IU has 2 S.D. codes (#1 & R (=#3)), (#2 is "Q" in both cases),

Take any (random) 2IU #1 input code: X. Select a fixed, complete (?) s.Delimiting (prefix) set. — S from X, select t. prefix string ∋ ~~[illegible]~~ ~~[illegible]~~ it is a member of S. (This is a unique choice). We have thereby ~~[illegible]~~ divided ~~[illegible]~~ t. 2IU input into 2 parts; t. first part is R, t. second is t. #1 input for 3IU.

T. trouble is, t. first part will always be in t. same complete set S. — So this will not map into t. 3IU codes, because they use difrnt prefix sets for ~~[illegible]~~ difrnt Q's & t. prefix sets are usually not complete. T. completeness is not problem, since t. 3IU can reject any of t. R inputs, but ~~t~~ using t. same prefix set for all Q's seems to have trouble of 315.29-.33. The ~~[illegible]~~ this is not a problem if t. R_i sets need not be complete.

Say Define Z^3 to be first input of 3IU, ~~[illegible]~~ Q^3 is second input, R^3 is third input
Z^2 is first input of 2IU, Q^2 is " ".

Z^3 and Q define the prefix set {R}

If Z^3 is self delimiting (I think it be) — ~~t~~ then R can be random & we will ~~[illegible]~~ know how to select a legal R.

~~[illegible]~~ If, in 2IU, we stipulate that Z^2 has to be read first, ~~then any input after a~~
Well, actually Q has to be read first, then Z^2 is read. Anything read after there is any output, is regarded as part of R. I think that would legitimately map to 3IU.
Actually Z^2 & Q can be read any time: but as soon as there is output, any non-Q reading is regarded as part of R. (not 100% sure of this yet, hvr.) → 317.15 →

(≈ SN) How to do this in "OOPS"! conceivably oops could have a register to contain Q! then, at a certain pt, it begins reading "random" bits (or sequential integers) preparatory to printing out "A".

00 : 31440 ! I'm ~~[illegible]~~ uneasy about t. partial functions — since they don't code t. entire corpus, There shouldn't be as much "sharing" of "bcost of #1 input".

02 ? { Perhaps if a partial funct has $\underbrace{\text{bcost of f, \#1 input pc (R_i)}}$ & it only ~~[illegible]~~ responds to one Q, Then it
.03 { is still a legit code ~~for~~ that sum w. wts $\propto 2^{-}(*)$

.04 —— $\boxed{.02-.03 \text{ may be a critical } Q}$! In order to be "partial" a function has to also contain info
.05 about what inputs to "ignore" (≡ give zero output for).

So : is ~~[illegible]~~ — is .02-.03 o.k.? — Note .04-.05.

A poss. arg't v.s. .02-.03 : Say we have a corpus of A, B.

.09 A is dscrbd by D_A — which does not dscrb B
.10 B " " " D_B — " " " " A.

It would seem that ~~$\exists$~~ $D_A + D_B$ would be needed to dscrb A, B.

12 D_A & D_B would not work well.

13 .09-.12 are perhaps not exactly correct. $\left\{ \begin{pmatrix} D_B \\ D_A \end{pmatrix} \text{ has input } Q_A \text{ \& output } \begin{pmatrix} B \\ A \end{pmatrix}; \text{ with input } Q_B \text{ it has no output} \right\}$ is correct.

(8.23.03): I had t. idea that 3 IU was "covered" by t. "Grammar" one Prod (of sum of all ~~[illegible]~~ partial d-functs) in which ~~[illegible]~~ case it multiplicatively dominated t. 3IU

Some Q's : Is this a legit dscrb? Is it a "proper" a priori (i.e. does it sum to < ∞? .. Is it an enumerable DO [redacted] please)

Dark So t. Grammar method's until "sum of all 2 input unis" ≡ "2 IU"

T. univrsl 3IU is sum over all 3IU's? . Input #1 dscrbs a particular O'. For predn, we use a density dist. over all O's.

Consider density dist's over 2 IU's : (just input #1 is #2 of t. 3IU).

SN In 3IU, in general ~~[illegible]~~ it is usually impossible to have t. same set of R_i's for all Q_j's :
~~This is because [illegible] (i.e. length of shortest R_i)~~ will vary w. Q's. ~~For all Q's,~~ $\overset{\equiv S_i}{}$
~~Say we have a [illegible]~~ for each [redacted] Q_i there will be an Δ of max pc. For all Q_i ~~[illegible]~~ take t.
[redacted] max that is smallest. Say it is of size Δ, Then no R_i can be shorter than $-\log_2 \Delta$,
because if it were then t. R's for Q_j could not add up to 1! They pc's are too small to be able to use that short

29 Another Q is whether it is at all likely that all Q's will use t. same R_i-set.
30 If t. Q's are solved sequentially, Then after t. first prob has been solved, there will ~~be~~ be t. "shortest R_i" that is used in that soln. Subsequent probs ~~[illegible]~~ would not t. be able to have a ~~[illegible]~~ min length that is longer than that "shortest R_i". So t. first problem would determine
33 what ~~set~~ of R_i's to use. ⟶ 316.19-.22 seems to get rid of this diffy by not having "complete" R_i prefix sets — also so they have to be renormalized.

316.19-.22 says that we can use same prefix set for all Q's, but in general, we will not use all of the R_i, & will renormalize.

So, for 3IU, ~~that~~ local assoc set of d-funct will be partial, & (t. assoc set of local d-functs will be partial).

$\boxed{\text{Consider .13}}$ Essentially, A is dscrbd by D_A & B is dscrbd by D_B. There is no point of viewing A & B as part of t. same corpus.

8.22.03
3PM

so: 313.40 : Perhaps a good general way to approach t. problems of 313.31 : That to each value of an S-function we assign a pc ∝ $\sum 2^{-L(\text{code } i)}$ (☉).

.02 New tack: In OOPS, we might get it to mimic 3IU by considering t. "R's" as "Backtracking"

Tho: Superficially, .02 sounds wrong! We ~~be~~ normally "Backtrack" to get a code that fits all problems exactly.

Can we use MDL "2 part Codes" to get reasonable S-functs? or "Dynamic MDL"

In 3IU, one part can be The first 2 inputs, & t. second part can be the R input.
It would be ~ MDL if we only used one O^2 function.

0

AH! In 3IU! when we make separate codes ~~for~~ of t. corpus for each value of R! Each ∥ "R" code consists of the common "first input" (in prefix code form), followed by R (in prefix form). So t. relative wts of t. codes are always $2^{-|R_i|}$.

This view makes it look much closer to the "Grammar" codes; i.e. many ∥ partial d-functs in ∥ with different wts. So each partial funct codes part of t. corpus.

18 If t. same prefix code for R_i were used for all solns., then every R_i in that code
19 would be used in all problems. and t. ~~[illegible]~~ d-functs will be total (not "partial")
0 : — On t. other hand (as is more likely) different pref. codes are used for different ~~so~~ problems, then many d-functs must be "partial" — since certain R_i will be used for some problems, but not for others.

In t. case of .20, it seems not obvious that t. relative wts of t. d-codes should be ∝ $2^{-|R_i|}$, even tho t. R_i code a different no. of problems. → Maybe not so bad!

Consider a single Q,A: T. pc of any single code will be $2^{\wedge}-(\text{len that \#1 input to 3IU} + |R_i|)$ (#1 input & R_i) (#1 input same to all problems & R_i)
I.e. each problem will have its own {#1 input & R_i} — which is t. complete code of any particular soln.
Hence, we select #1 inputs that are short, because the resultant codes all have higher wt.

In (.19 .18) t. ~~[illegible]~~ ∥ d-functs are not partial, they are "total" (they have outputs for all inputs) or all "legal" inputs ≡ "meaningful inputs"

0 So, it looks like a wtd set of d-functs that describe all A's, is o.k. T. ~~[illegible]~~ d functs can be total, but better partial (i.e. not output for certain inputs). Also, we need partial recursive functs if we are to get universality [An alternative to prim. rec. functs]
a new idea from .30

So t. set of all (partial) d-functs that describe a QA ~~corpus~~ corpus does give a universal distr., since t. 3IU codes are a subset of those ∥ codes

Re: d-functions used: We'd like to have functions ∋ ~~[illegible]~~ for any Q input, ~~[illegible]~~ & A output ~~[illegible]~~ ∃ a R that will give that A output.

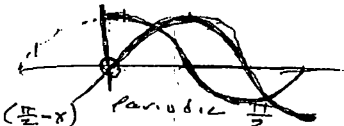
→ 315.00

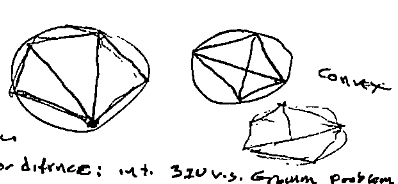
8.21.03
3TM

$\cos(~~x+~~\frac{\pi}{2}) = \sin x$

$\cos(x) = \sin(x+\frac{\pi}{2}) = \sin(\frac{\pi}{2}-x)$

$\cos(\frac{\pi}{2}-x) = \sin x$





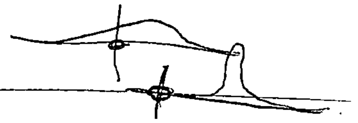
>0 ! 312.40 : This seems like it may have answer a/o strong Hint! { A major difference: in the 3IU v.s. Grammar problems in 3IU, all || codes have been paid for.

Another (perhaps Related?) Tack: Each Pem has assoc. w. each problem, a "confidence level" that it computes (by itself) [perhaps this is ~ to 312.34–.35.]

.03 In the model of 312.30, if a Pem hasn't had much experience in a Domain, then its codes will all tend to be very long: So ~~many~~ many possys of ≈ pc which is perhaps ≈ 312.34

— In the "Real" domain, it would have a broad rather than a sharp distribn.

.06 A Pem of the kind I'm interested in either has no output, or has a $\overset{>0}{pd}$ for every possibl. output.



In the 3IU, each "R" code has an "extra" pc factor $= 2^{-|R|}$. But each R code has a ~~short~~ single predn. for each problem: So perhaps ~~it might be so close that~~ the 3IU is not so close to the problem of 312.30

10

Anyway in the ~~[illegible]~~ idea of .03–.06, If we combine a sharp d.f. w. a broad d.f., giving ≈ wts, I think the sharp d.f. will dominate. The amplitude of the sharp d.f. is ≈ ÷2.

If the d.f. is over discrete things (an ∞ of them) then we have about same effect.

Real Problem of 312.30: — assume 312.35 occurs: if the Pem is "uncertain" it gives a broad D.F. (.03–.06) — How, how much wt should we give to the a.d.f.'s (≡ Pems).

If one Pem has a very long derivn, it can be very sharp, but very A.H. So I'd like to somehow bring into the argt., the fact that a Pem may have coded a long Corpus "successfully". It's not clear how to do this!

20 271.21–.24 & 274.12, .26 are on Grammar → 3IU equivalence.

The 3IU is a wtd sum of (partial) d-functs.

read 269.23–.40! This is why I used the term "Grammar" for the "wtd d funct" model

→ To use J's "OOPS" machine ~~for~~ to Generate s-functs: Work on first problem & oversearch, so we have several d-functs that work p1 (≡ problem #1): Run we try these d-functs on p2.

271. to 274

Some can be extended ~~to work [illegible]~~ (or not) so they work both p1 & p2. If some simply stop w. no output for p2, that's OK. If a d-funct gives wrong answer for p2, that, too, may be OK.

278.22 ff

For p3, we use the same set of d-functs that worked for ....

I don't have an entirely clear picture of ~~what~~ how this works. I understand 3IU, but the wtd set of || partial d-funct model is unclear.

30

.31 So! 2 serious problems! (1) Just how/t and 54 Grammar is equivt. to 3IU (See & read carefully 269.23–.40 (.20R sn ff) reading & correcting this material is good way to familiarize self w. that Domain.

(2) 312.30 "2 intersecting induction paths". Seems like Generally impt. Q.

Is this Q. same as the very old "Blinking Gambler" problem? ~~k~~ k times blinks out of N times, before gambler roles dice, he blinks. — This results in 7: what is pc of 7 w. the two. Blink?

Of interest k = 0, 1, 2 & N is not large.

Is (2) related to "Encyc" problem?

8.20.03

P2.27
Prostate formula
Phone Mo.

3TM

:0 : I really have to find a good way to do predn. w. several (partly ||, partly mutex exclusive) s. funcs.
This seems to be a "well defined problem."

Consider corpus A, B: ~~[illegible]~~ s. funct f_1 accepts A & B ; has ~~pr~~ aprip a_1.
s funct f_2 " B only. " " a_2

f_1 has pc^1_A for A, pc^1_B for B | What is total pc of A, B corpus?
f_2 " pc^2_B for B.

perhaps $pc = \approx pc^1_A \cdot (a_1 pc^1_B + pc^2_B \cdot a_2)$ Probly wrong. The way we should use all of a_1 on Problem B is Unclear!

$pc =$ ~~[illegible]~~ $a_1 \cdot a_2 \cdot pc^1_A \, (pc^1_B + pc^2_B)$

~~$(x \| y \equiv \frac{1}{\frac{1}{x}+\frac{1}{y}} = \frac{x \cdot y}{x+y})$~~

Another way: distribut a_1 over Probs A & B in a way determined by pc^1_A & pc^1_B.
$\ln a_1$ is distributed in ratio $\ln pc^1_A$ to $\ln pc^1_B$.
$\ln \gamma$ " $\ln \alpha$ " $\ln \beta$ "

$$\ln\gamma \cdot \frac{\ln\alpha}{\ln(\alpha\beta)} \;\Big|\; \ln\gamma \frac{\ln\beta}{\ln(\alpha\beta)} = \Big|\; \frac{\ln\gamma}{\ln(\alpha\beta)} \cdot \ln\alpha \;\Big\|\; \frac{\ln\gamma}{\ln(\alpha\beta)} \cdot \ln\beta .$$

$$e(\;) = \gamma^{\frac{\ln\alpha}{\ln(\alpha\beta)}} \;\Big|\; \gamma^{\frac{\ln\beta}{\ln(\alpha\beta)}}$$

P276 : old Search TM

'0 $$A^{\ln\alpha} = e^{\ln A \cdot \ln\alpha}$$

(.30)

(SN) For Report (or next version of Report or MIT story):
Describe various PS methods I will use.
1) QA ... Lsrch. : 1 soln, (det. functs) then >1 soln. (S functs)
2) INV " "
3) OZ " ~~#~~ many || trials.
4) PST's: Advanced methods for 2) & (3), 1)).
5) PST's Grammars for induction.

30 (20) Another way to think about PCs: We have 2 Pems f_1, f_2 that have had, in t. past, 2 different corpi — each has had its own or degree % of success in t. past. We now turn them both loose on a common (problem) Corpus! How to combine their predns? [A not unusual problem!

A Pem can express its lack of confidence in its predns in 2 ways (perhaps):

34 1) ~~[illegible]~~ Its predns can deviate minimally from a certain aprip → (313.03)

35 2) T. sum of ~~#~~ t. probys of its (cond). predns can be <1.

Hint! In ~~converting~~ showing Equivalence betw. 3IU & "Grammar" (|| codes) method ~~spec.~~ ~~of sets~~ of Getting ~~sets~~ universal S functs, do I deal w./ solve this problem? (313.00)

8.20.03

3TM

00 : (309.30 : spec) : On "oversearch" : ~~[illegible]~~ One justifn. is that it does give pc's ! : w.o. it we get one "soln". w.o. oversearch, we really don't know how probable t. ~~other~~ simple "found" soln. is wrt other possys.

Also, ~~is~~ if we use 3 input ~~[illegible]~~ mach. for getting S-functs, we don't have to oversearch to get pc's ! but we may have to backtrack. Oversearching will, hwr give better pc's.

I was worried about SUMAC's (w.o. backtracking, or w. very little backtracking) — I think I'd come to a "dead-end" when the 100 best codes ~~I~~ had reduced down to zero!
10 — But I don't think this normally occurs. We can find useable extensions of t. 100 best codes — But they ~~[illegible]~~ can be very long. A reasonable strat. would be to spend time on many difnt branches. — But in actually backtracking — (or just using a previously oversearched soln. from a lower level) we have to solve at least 1 more problem in addition to t. presently problem.

At present Moment, I'm not really sure about best way to do search. Trying to get codes by Extending previously successful codes can work only if we
20 have certain kinds of languages. So maybe make list of some examples of those kinds of languages.

J's OOPS seems good in this respect. For each new input, it may or may not request additions to t. func. defn. While his stack language seems good and general, I'm ~~wary~~ wary about ~~the~~ t. way he uses it. — How ~~[illegible]~~ solns to previous problems are used by t. system — how pc's of tokens are updated. (I think he updates wrt. present problem or present problem "subset" only. He shouldn't be telling TM what t. subsets are — (other than perhaps by ordering problems
30 as a TSQ, Or if they are labled ~~as~~ ~~[illegible]~~ w. $\boxed{\text{indices}}$

— But his pooling token frequencies from t. same problem (index) set only, seems a bit A.H. — it reduces transfer lrng. to "boost" only,

If we have ll S-codes of t. corpus, this gives ↓ of ~~[illegible]~~ equiv't code length. On t. other hand, ~~[illegible]~~ a set of codes ~~that~~ ∋ each code does not code all problems, but ~~has~~ each has its own "acceptance criterion" — This set has to ~~be~~ have their boost added together.

Actually, t. formul. is excessively simplified. If certain parts of t. corpus are accepted by several S-functs, this should somehow ↑ B cost of that parallelism, →312.00

8.17.03
3TM

APRIP of INTEGERS & REALS $\boxed{.26}$

TIME Varying OZ probs! .12-.25

a Not Bad (but not perfect) soln.

.0: 309.40 : $\boxed{SN}$ In chosing to do WON rather than "Lsrch": I could do it either way. In the earlier optimize Lsr

.01 Version, I ~~did~~ ~~at~~ got these time(t)'s (for INV problems), then I roughly

.02 selected a good set of them viz ≈ $\frac{a}{n}$ gone, then I calculated an ordering

.03 based on "proby of being best".

In WON, I just ~~used~~ ~~$\frac{a}{n}$~~ selected best ~ $\frac{a}{n}$ & worked on it

until (due to ll update) a different PST (i.e. different h(t)) looked better.

I could do it same as .01-.03 — select "Most likely to be best", work on

it until ll update says another PST is proby "Best".

0 The only difference is the criterion for "Best": The one used in WON is easyer to calculate

& does have good theoretical reason for being likely to work well. In fact I was

thinking of using the WON criterion to "Narrow down" the set of PST's to compare for "likely good at ~~being~~ being "Best""

2: ~(189.00-.90)

On timeVarying OZ problems:

In the non-time varying case (say G was ≈ "linear" (291.27-.29))

We have a $h(G,t)$ but we only need to know $h(G,t_m)$ (the CB=Time of

the problem being worked. So we select the h → $\int_{-\infty}^{\infty} G(G,t_m)\,dG$ is max.

Again, ~~as~~ having $h(G,t)$ ~~as a suitable function of t by~~ x G is not a funct of x.

0 Say $\underline{\underline{G}} = G(x)\cdot g(t)$ then consider $h(G,t) \to$ ~~g~~ $h(G\cdot g(t),t)$

for $\underline{\underline{each}}$ $\underline{\underline{h}}$ there will be a ~~t~~ → $\gamma^{max} = \int_{-\infty}^{+\infty} G\cdot g(t)\cdot h(G\cdot g(t),t)\,dG$

$$\gamma_t^{max} = g(t)\underbrace{\int_{-\infty}^{+\infty} G\,h(G\cdot g(t),t)\,dG}$$ is Max

It may be possible to simplify this; generalize g(t) here

25 We will then select the h that has ~~[illegible]~~ largest value of γ_t^{max}

26 On the aprip of the positive integers or positive reals.) → reproduction on 3 04.10

the $2^{-\lg_2^* x}$ d.f. gives not so bad behavior at $x\to\infty$

$2^{-kcost(x)}$ goes to zero for large values slower than any recursive funct?

0 Anyway we are interested in (at least) 3 aspects of $2^{-kcost(x)}$

1) Value for $x=1$ (small x) 2) behavior for large x ($x\to\infty$)

3) At what value of x does the transition take place?

Proby 1 & 2 are time dependent, so "previous Experience" is very imp. } → 304.10 ~~Nif~~ HMG subfile

Marcus suggests that (normed) $\frac{1}{x}\cdot\frac{1}{(\log_2 x)^2}$ is good enuf for most cases.

((Perhaps there is mainly 1 problem: 1 params i.e. PC of integer, $\underline{\underline{1}}$.

behavior at for large x is not critical $2^{-\lg_2^* x}$ is proby good enuf.

for small x $\frac{1}{x}$ is o.k. Value at $x=1$ determines pt. of transition to "large x"

8.16.03

3TM

they were first proposed, but

.00 i.e. those theories didn't look very good at the time. If they fit ~~all~~ the new data ~~[illegible]~~ ~~theories of the~~ they will be reconsidered (unless the a priori probability is still too small()). ~~Awareness~~ Awareness of many alternative theories corresponds to remembering the results of ~~[illegible]~~ "oversearching" — ~~[illegible]~~ ~~[illegible]~~ ~~knowing~~ remembering several alternative solutions of lower ~~[illegible]~~ probabilities.

A novice scientist ~~will~~ does not know about those ~~earlier~~ theories and for his "backtracking" he has to ~~[illegible]~~ to ~~find~~ invent new theories that fit both the old data and the new.

He will ~~be~~ is less constrained in his search for ~~theories~~ theory than the more and may find a very good, very novel theory that fits very ~~well~~, but ~~the~~ experienced scientists, ~~but his search will probably~~ it ~~will~~ take ~~[illegible]~~ will take ~~much~~ more time than the experienced scientist

0 ~~[illegible]~~ much more time to find promising candidates.

~~others~~ It is commonly ~~[illegible]~~ ~~Novice Scientists~~ ~~usually~~ ~~seem~~ believed that the scientific community simply selects the ~~[illegible]~~ single best theory and forgets about the rest (the "also ran").

? This is a serious misunderstanding of the mechanics of scientific progress. The more ~~mature~~ mature scientists are ~~[illegible]~~ aware of alternative ~~[illegible]~~ theories in their sciences and are quick to propose ~~theories~~ theories when new experimental data demands revision.

The results will occasionally be superior to ~~[illegible]~~ most of narrower search of the ~~experienced~~ scientists, but will take much more time.

some confusion betw. "prob" of theory & "a prior" of a theory

20

SN On "oversearching": It can be quite expensive! To ~~get~~ find a code w. pc = k bits longer than first code found, means cB ~~[illegible]~~ $CB \times 2^k$. (~~[illegible]~~ assuming cc ~~per~~ of generation & test is same for all codes) On t. other hand we may find many || codes of ≈ or shorter length than first code found. To raise "Diversity" to 100 codes would seem to usually be too expensive (?). — [would it cost >> 100 times cost of finding single "best" code.?]

Actually, I have no reasonable idea as to how many codes I get for given amt. of "oversearching": For oversearch factor of S, I get all codes w.

$$\left.\begin{array}{l}\Delta cc = cc_0\\ \Delta b = b_0\end{array}\right\} \quad \boxed{cc_0 \times 2^{b_0} < S}$$

How many I ~~get~~ get is unknown. → 310.00 sec

10

"Backtracking" would seem to be much cheaper than "Oversearching".

In Backtracking, we (seems) to get solns w. minimal searching: we ~~[illegible]~~ do as little searching as we can & still get acceptable solns. It is thus, Maximal "Greedy" — & ∴ suspect! Also, essentially no real "probabilities" obtained (for ~~[illegible]~~ ... but probs for new problems are available (automatically since we will often get many answers w. their ~~[illegible]~~ pc's before we get t. "correct" answer.)

8.16.03

3TM (≡ NIPS)

≡ 03TM

>0 : 307.40 : 307.30-.40 ~~[illegible]~~ May be an "impt." "Breakthru" (if it is, indeed, a new idea (G)).

In Sumac, I was always worried about loss of diversity occurring whenever new probs were solved. Say we ~~[illegible]~~ store (100 best solns ~~[illegible]~~ thus far for entire corpus). When a new problem comes in, say only 10 of them "fit" the new data. (we haven't done any searching yet). This "100→10" should trigger a "feeling of unease" in TM. It means TM either backtracks a bit, to find more solns to past as well as present problems &/o in ~~future~~ immediate future, does more "over searching" until we have more solns (say ~100) to each ~~[illegible]~~ of the "near future" problems. 0

"Normally" ~~GGPD~~ GCPD ~~[illegible]~~ summarizes all knowl. of TM up to then. "Backtracking" amounts to "revision" of GCPD based on past info only. "Updating" means revision in terms of new info. So if updating is "difficult" : i.e. "too low" pc's are assigned to new data, One trys "Backtracking" : ≡ Revision of GCPD on Basis of past data. Usually we want to go as little into "past" as possl., but often we will have good ideas as to what part of past needs revision. So for new, anomalous physics results, we don't try to modify theories in sociology or linguistics (But we might in chemistry because it's "close" to physics). 0

So, in certain domains, we have some idea as to how large pc's of new data should be. If our GCPD gives too low values we try to revise GCPD by "Backtracking".

How Sci. Community deals w. anomalous data: 1) Cold fusion 2) flying saucers. (lons/flash radiations as experimentalists ~~[illegible]~~ troubleshoot) Main method is to claim Experimental errors! ∴ no need for theory revision.

Other scientists do propose theories, Do Expts (in Cold fusion).

For end of § 1.1: The foregoing ~~description of~~ treatment of Updating and Backtracking ~~[illegible]~~ has correspondences in the scientific community. ~~When a~~ Suppose a new experimental result is presented, that ~~[illegible]~~ seems to violate current ~~[illegible]~~ theory — which ~~the usual response is to~~ ~~(current theory~~ gives the new result a probability close to zero. We can much increase this probability ~~[illegible]~~ by invoking "experimental error". 50 If the result is replicated by many other scientists, experimental error becomes less likely and "theory revision" ~~[illegible]~~ must ~~may~~ be invoked — This amounts to "backtracking": finding a new theory that fits the old ~~[illegible]~~ data as well as the new experimental results. ~~A [illegible] scientist will~~ An experienced scientist will often be aware of several theories that fit the old data, that were rejected ~~[illegible]~~ — perhaps because of low a priori probability —

8.15.03

SUMAC .30 ff

Nips ≡ 3TM

Could be ~~[illegible]~~ Nameof ppm
a' title of series of papers.
The "summary" is in GCPD — The ~~[illegible]~~ capability of

Arr Sat 16 Aug 13:55
LH 424 — 1800 645 3880
Lv. Wed 20 Aug afternoon

Marcus
ETA 1:49 PM

: SN some topics to prior discussion report!

Backtracking is an essential part of system, since we can't always make adequate summary — unless we use full (incomputable) ALP. ≡ Universal D.F.

1) If INV: OZ are adequate: why do we start w. QATM?

2) How Does QATM ~~[illegible]~~ differ from, say Google?

3) ⓐ How Does α Differ from other Lang Systems in its treatment of NLang?

1498

ⓑ — particularly NL lrng? — ⓑ How different from Statistical (MT) 2ch ktn.?

(2.b) Much less ~~[illegible]~~ apri idea of what words mean, of formal grammar.

Ⓒ QATM "understands" Q's much better than current SOA MT — i.e. its Models of lang & of world are more General, "Better" than ~~→~~ The Rig is a "Moving Target".

4) Discuss ~~by~~ IMPT aspect of t. GCPD: That it enables transfer learning betw any problems solved by t. system — if we decide to minz. code of t. entire GCPD, rather than minimiz ~~[illegible]~~ independantly t. various "parts" of t. GCPD.

5) p5, end of § 1.1: on "Backtracking". Explain: / ~~Th~~ Backtracking is invoked ~~when the system~~ ~~suspects~~ upon suspicion that an error has been made in an earlier decision. We ~~normally~~ first go back to modifying the most recent critical decision. ~~[illegible]~~

"Suppose F^n works ..... ~~But will work on Q, .. Att. if the recognizes only that the decision to use F^n why F.. fail~~" If we can't find one,

~~we~~ [It may be well to give note on previous page (p. ?) saying that / Later we will discuss the utility having several alternate ~~[illegible]~~ solutions w. different assoc pc's gives us a larger set of F's in memory can make backtrac'

Oversearching has 3 benefits:

1) It may give us a soln. of higher pc.

2) It could give several solns. of ~~[illegible]~~ ~~[illegible]~~ possibly differing pc's — enabling us to get a pd on possl. solns of future problems, rather than a ~~[illegible]~~ single soln.

3)

SIU This oversearch / backtrack Business is very impt in Making TM work in "sequential mode" — i.e. SUMAC mode

— In "Oversearching" we (if lucky) get several choices for ~~[illegible]~~ future problem solns. — Thus making "Backtracking" less likely to be needed. On t. average, we want Oversearching to compensate for t. loss of diversity (≡ Adaptive Variety) that we get, when new data comes in & we Reweight pc's of many possl. cncs.

("Conc. trails" ≡ paths thru t. concept net that give a solution trails).

What we want is as many codes for a soln to present problem as possl. This automatically makes ~~[illegible]~~ t. need for Backtracking less likely.)

Has to do w. SUMAC

See 308.00

08·13·03

NIPS

00 :305.90 : 305.30 is a sort of (Lame!) intro : But maybe first make outline of what the section will contain!

General remarks:

03 1) List types of d-functs, s-functs: Give examples,
In list, perhaps include (.03R) & applicns.
Or maybe have this discussn. separate.

2) In introduction, explain ~~why its appl. to ban~~ how d- & s functs are used in [redacted] ~~Alph~~ Alpha α & [redacted] why its impt to have ~~mno~~ them in many forms.

(.03R) Also discuss
~~forms of~~
h . . .
e^{-x^2}
$x^n e^{-x}$ etc.
a, μ, σ^2 as d functions → s functions.

10 3) Discuss L/Learning, using various (S/D) funct "~~by~~ targets".

4) S/D functs occur as ~~targets~~ (outputs) & as solns to induction problem.
(Generators of outputs). Give examples

5) Discuss: ~~#~~1 Input umc., (S distribn can be universal) (S distribn 2 input umc.) (S funct 3 input S funct)
Also discuss (Grammar / language) model. Show how different from 3 ist, yet equivalent.

10^{10}:
10^9/sec 10^6/sec.
10^4 sec : $2\frac{1}{2}$ hrs.

SN A (perhaps) Nice way to get functs w. several outputs: "Vector".
a) As one of the inputs have the index of the vector component desired. This enables sharing of concs. between functions.
20 In the case of the h'(E)'s ~~say~~ say, we can optimize a & μ separately —
This "elgm" usually ~~saves~~ simplified (shortens) the task — But its not a Bad elgm — rather Good, in fact!

(6) For Expo (perhaps) Start w. General (soln(s)): Say 1 2 or 3 input umcs,
Then discuss alternative means & approxns.

Start with Section I on Simple ~~introduction~~ QA induction.
Reword thus: My recent stuff on s-functs could intersect with "the older stuff"
Introduce 3 input umcs. Mention equivalence to other form.
Also, discuss a, μ, σ^2 vector output for h' ~~models~~, models.
30 Discuss AZ v.s. OOPS method

§6 p 18 on "related work": Discusses Starting w. Algebra & evolving into "English about Algebra"
~~Another~~ An important source of training sequence material can be obtained from work on "Expert Systems".

08.13.02
Nips

00 : 300.29 : Fortunately, the ~~incomputability~~ incomputability of the universal distribution is rarely relevant to its ~~application~~ practical utility. For practical induction we do not try to use the Universal distribution. Instead, we use approximations to it. While it is impossible to ~~[illegible]~~ have a useful estimate of how close an approximation is to the Universal distribution, it is easy to tell which of two approximations is closer — This is the one with the (generalized) "shortest code". ~~[illegible] approximations~~ ~~Because we~~ In view of the ~~[illegible]~~ superior performance of the Universal distribution, we want approximations ~~[illegible]~~ that are as close ~~as possible to the~~ to it as possible.

For practical purposes, we usually need ~~to know~~ an estimate of the expected
0 error in prediction. For our approximate distributions, this may be obtained in the same way that other induction systems are evaluated — ~~[illegible]~~ e.g. by cross validation. For ~~[illegible]~~ approximate distributions that have been obtained completely a priori (without seeing the data), the ~~testing~~ training set is of size zero, and we are able to use all of the data for testing our system.

17 229.312 : The ~~[illegible]~~ system is particularly adept at "transfer learning". It performs induction ~~[illegible]~~ on data by writing short codes for the data. If all of the problems in a domain are described by a single code, then transfer of learning
20 between problems in that domain occurs through sharing of definitions between problem descriptions. If we use a common code for several domains, the sharing ~~definitions~~
22 between those domains facilitates transfer learning between those domains. → 3001

(299.18-.40 ... 300.00-.40, 305.00-.22) This is [illegible] for introduction! Not ~~[illegible]~~ necy:

I do want to (eventually) have a good expo on "Motivation" — Why this is such a promising approach. Not needed in this "Revision 2".

I want to ~~[illegible]~~ have good section on "S functions".

Title of Section: Stochastic functions and their Representations
0 or Representation of Stochastic functions
or Representations of Functions.

~~B~~ Both deterministic functions (d-functions) and stochastic functions (s-functions) play critical roles in the present system — both in directly ~~[illegible]~~ implementing the system and as ~~[illegible]~~ models of data, for which we try to find short codes.

Because of the great variety of problems we will be solving, it is well to have a great variety of representations of both d-functions and s-functions, so we can tailor our ~~[illegible]~~ search to the particular ~~[illegible]~~ kinds of problems we are trying to solve.

8.9.03
NIPS

(Dip)

10: 301.40 ~~It would seem that we are [redacted] progressing backwards at a [illegible] rate!~~

We ~~have are~~ proposing a ~~[illegible]~~ technique for optimization that ~~involves two~~ requires at least two new optimizations! ~~[illegible]~~ Is anything being gained?

~~Consider the two opt.~~ The first new optimization ~~is~~ ~~of~~ equation (301.05). Obtaining a good $O^{?}$ ~~[illegible]~~ is useful for ~~all future problems, not~~ not only the present problem, but for all future problems — so it is a burden that is, to some extent, shared by all problems.

~~The other~~ second optimization involves finding the F_R with as large ~~[illegible]~~ $\beta^{2'}_{m,2}$ ~~[illegible]~~ as possible.

Since this is a common problem that is solved many times, we will ~~probably~~ ~~a~~ ~~find a very good solution for it~~ try to find a way to solve it that is fast and effective: ~~[illegible]~~ ~~[illegible]~~ try optimizing (eq 301.05) ~~[illegible]~~ and again in optimizing ~~[illegible]~~ equation (301.25), we will usually be making small ~~corrections~~ corrections to a previous optimization — so the process ~~would~~ need not take much ~~[illegible]~~ time. ~~We alternatively, for a fixed time spent, we will get better good results.~~

The ~~methods of the~~ techniques ~~in~~ area of the present section and of section 2.7 are meant to follow what seem to be common human methods ~~of~~ for solving ~~[illegible]~~ problems of these kinds.

8.9.03

Dup

Nips

Section 3.1

Title (same as §2.1).

.00 : Back to § 3.1 : Improved Updating and Search techniques.

The ~~impov~~ improved methods of section 2.1 can be applied to ~~[illegible]~~ optimization problems as well.

Here we want to find O^is such that

.05 $$a_0^i \prod_{j,l} O^i(G^{j,l} \mid \tilde{G}_j, t_j, F_l) =$$ eq (301.05)

is as large as possible.

$(\tilde{G}_j, t_j)$ describes the j^{th} optimization problem: to find ~~an~~ within time t_j,

10 an x such that $G_j(x)$ is as large as possible.

It ~~$O^i(G^{j,l} \mid \tilde{G}_j, t_j, F_l)$~~ is the probability density ~~density~~ (in view of O^i) that F_l will find an x ~~[illegible]~~ within time t_j, such that $G_j(x) = G^{j,l}$

→ Let us define $h^{i,'}_{j,l}(G^{j,l}) = O^i(G^{j,l} \mid \tilde{G}_j, t_j, F_l)$

After we have found a good O^i function viz eq (301.05), we can use it to obtain 'h'' functions for an arbitrary problem and arbitrary PST.

Suppose we want to ~~[illegible]~~ ~~[illegible]~~ solve a new problem, G_m, t_m ~~[illegible]~~

h'_0 — comma

20 Then for every F_l, O^i will give us a probability ~~[illegible]~~ distribution over $G^{m,l}$.

Since ~~we~~ we want $G^{m,l}$ to be as large ~~as~~ as possible, we will select, for our first trial, the F_l ~~fixed best thing~~ with a $h^{i,'}_{m,l}$ such that its expected $G^{m,l}$ value ~~[illegible]~~ i.e.

.25 $\beta^i_{m,l}$ ~~[illegible]~~ $$= \int_{-\infty}^{+\infty} G^{m,l} h^i_{m,l}(G^{m,l})\, dG^{m,l}$$ (equation 301.25)

Beta

is as large as possible.

We ~~[illegible]~~ apply this F_l to the m^{th} problem for ~~[illegible]~~ ~~[illegible]~~ ~~interval of~~ time $t_m/10$. At the end of that time, we reevaluate eq (301.05) to see if F_l is still the most promising PST.

30 If it is, we continue applying it to the ~~[illegible]~~ m^{th} problem. If not, we apply a more promising PST to the problem. We continue this alternation ~~alternation~~ of applying PST's and reevaluating ~~reevaluating~~ them, until all of our time, t_m, has been used up.

~~In the language optimization tap~~ In working ~~a single optimization problem, we have proposed a technique that involves two~~ new ~~optimizations~~. Superficially,

8.9.03

NIPS

(301, 302?)

>0: 299.40: ~~domains, we get transfer of learning between these domains.~~

0(: ~~305.22~~ 305.22 Another source of power of the present system is in the very broad classes of problems it can solve. These are inversion problems — ~~[illegible]~~ corresponding to the P and NP problems of computational complexity theory. They include solving of equations, symbolic integrations proving theorems, etc. Another broad class of problem it solves are time limited optimization problems. Usually, the NP problems ~~[illegible]~~ we have mentioned ~~[illegible]~~ ~~[illegible]~~ are effectively of this form: we cannot solve them exactly in the available time, so we want ~~as good as~~ an approximation that is as ~~good as~~ possible.

All induction problems ~~[illegible]~~ can be regarded as problems of this type. This includes sequence extrapolation (time series prediction), all classification problems, learning to translate from one language to another, ~~[illegible]~~. discovery of a Grammar for a corpus of data, ~~[illegible]~~ and the improvement of a machine learning system. — So the system is able to work on the problem of improving itself.

There are very few problems in science and engineering that are neither ~~[illegible]~~ inversion problems or time limited optimization problems, so our system does indeed solve a very broad class of problems.

Do go with this.

We believe that the Universal distribution gives the best ~~[illegible]~~ predictions possible. While few people contend this view, there is a ~~general~~ certain (misunderstanding) uneasiness in the scientific community about ~~[illegible]~~ practical ~~[illegible]~~ application of the universal distribution to real problems, since ~~[illegible]~~ this distribution is known to be incomputable. → 305.00

This kind of discn. could be included — but without it.

Otherwise, just say univ. dist. gives best predns poss. — Q show thus?

29
10

SN In summary; our system is able to solve problems over perhaps the broadest possible set of domains. ~~The method~~ ~~Its method learning, the~~. It uses the Universal distribution for learning — perhaps ~~the~~ ~~one~~ the most general, most accurate ~~[illegible]~~ induction ~~[illegible]~~ possible. It is able to use this distribution for both ~~[illegible]~~ learning, for learning within a ~~domain~~ single domain, as well as transfer learning involving several domains.

? wrong ?

There ~~seem~~ is a certain class of problems in which we would (like to know how) ~~how~~ close our model is to the Universal distribution. Such problems cannot be solved using the Universal distribution or by any other means.

leave this out of report.

not ~~exactly~~ clear/correct.

8.8.03
NIPS

0 Defines t. ~~ϕ~~ sub OZ problem is peculiar to t. present problem, The ~~[illegible]~~ PST's that are ~~good~~ useful for such problems are common to all of them. So the Z sub optimization problems ~~that~~ are always problems whose cost is somewhat "shared" over a large number of optzn. problems.

→ 298.34 is perhaps t. nite idea, but it is not at all expressed clearly! Also, I should mention that as with Inv. problems, The ~~sub proble~~ sub-OZ problems will at first be solved by Lsrch — and only in more mature machines will they be solved by PST's that are selected by the techniques of t. present section.

0 Anyway: first write § 3.1! (see 288.32 — 289.06)

11 SN In Now introduction, where I discuss "What is t. Mach Lrng problem": ~~B~~ Talk about success of ML in Narrow fields, but General inability to integrate lrng. from disparate fields — one of Most Noticeable Characteristic of Living Creative Minds. [Also, it would be good to have a ¶ or 2 on just how TM does "Transfer Lrng".

(Poss) Introduction!

18 What is Machine Learning? ~~[illegible]~~ We will define the aspects of Machine Learning that we have built into our system. We have a Machine

0 that is able to solve problems in various domains. After having solved several of the problems the machine ~~s~~ becomes more effective in solving new problems. "More effective" can mean "more rapidly", or, ~~in the case~~ if there ~~is~~ are ~~a~~ different effectiveness criteria for ~~the~~ problem solutions, it will do better with respect to those criteria.

A system will be considered good if it is able to solve problems in many domains, and ~~[illegible]~~ If it learns very rapidly to improve its performance. In particular, it should be able to do "transfer learning" so that

30 solving a problem in one domain will facilitate solutions of problems

31 in other domains. .312 → 305.17 ~~The present system is particularly adept at transfer learning. We do using the universal probability distribution for all kinds of learning. This is done by finding short codes (descriptions) of data in each problem. If the codings of all problems in each domain are done separately for each domain, we can have transfer of learning between problems within a domain, if our problem descriptions are allowed to share definitions and other concepts, in common. When we allow sharing of concepts between problems in different~~ 300.00

·10 (2.92.40 spec) : Modifn of discussn of §2.1 !

For each $h^i(t)$ there will be a value ~~[illegible]~~ of $t = t_0$ at which $h(t_0)/t_0$ is maximum.

We want the F_ℓ such that the associated $h(t_0)/t_0$ is

for each ~~[illegible]~~ $h^{i}_{n,\ell}(t)$ there will be a value of t

The expression $h^{i}_{n,\ell}(t)/t$ gives us the probability ~~[illegible]~~ of success per unit time expended. For each $h^i_{n,\ell}$, denote by $\alpha^i_{n,\ell}$ the largest value of this ratio. Large values are desirable. We then pick the F_ℓ that has the largest $\alpha^i_{n,\ell}$, and used to try to solve the current problem.

0 → After ~~[illegible]~~ 14.22 Which is the best?

Type by hand.

Consider $h(t)/t$. It gives us the probability of success ~~per unit time expended~~. and by t_α, the time at which this occurs: $\alpha = h(t_\alpha)/t_\alpha$

For each h, denote by α, the largest value of this ratio! Large values of α are best.

~~[illegible]~~ The first Gambling house theorem' suggests that we will minimize ~~minimize~~ expected total solution time if we schedule our F_ℓ ~~[illegible]~~ trials so that their associated ~~[illegible]~~ h(t)'s are in α order, largest values first

16

Still need to be added : ~~After~~

1) At end of section 2.1 : Discussion of when to ~~start~~ drop Lsrch & start using "LON" (insert 298.25-.33)

.19 2) In § 3 (on OZ) : Put in section 3.1 corresponding to § 2.1 !

0 ~~B~~ But a diffty is that "WON" uses OZ prob soln. : explain how this works; that Lsrch can be used for this or a U.G. PST that has ~~been~~

2.3 found. → see (.34)

3) Section 3.1 has to be written (carefully).

·25 ~~It will not be possible~~ Normally the techniques of the present section can only be used in a somewhat "mature" machine — one that has learned to ~~optimize~~ ~~the equation ( )~~ find very good O^j function by optimizing eq( ) — also it must have learned to ~~[illegible]~~ how to find ~~[illegible]~~ F_ℓ's of high α value. The decision when to drop Lsearch can be made by the trainer : ~~Analysis for this decision~~

0 This decision ~~can~~ is be made by working problems ~~by~~ using both Lsearch and ~~[illegible]~~ the methods of the present section. If the Lsearch solutions are still better, then

33 the machine is clearly not ready to switch.

34 on the ~~[illegible]~~ Q of .19-.23 : in this "improved" OZ soln, we have a sub OZ problems — so this looks like we are making extra work for ourselves! — But not really! finding a good O^i is ~~[illegible]~~ has value for many more problems than those being worked on now. The other relevant sub OZ problem involves finding a ~~h(G) of an~~ F_ℓ ~~[illegible]~~ yielding maximum E ~~[illegible]~~ P(G) for the present problem. while the set of P(G)'s they

08.05.03
NIPS

293-295 area
293 : 1) is ALP the only way?
~~293-295~~ 2) The HMC problem
294-297

0.0

anyway from 291.35 & .40 G and $H(G)\frac{dH}{dG}$ will not be ~~a~~ proportional (i.e.

$G = a H(G)\frac{dH}{dG}$ will not be true & so 291.35 > 0 will not imply 291.40 > 0 —

so the ordering produced by H(G) will not be the same as that produced by G.

{ If G ~~has~~ be linearized, then for all α ~~[illegible]~~ ; having G with probability α has same utility as having αG with ~~certainty~~ probability 1,

Does this work for negative G as well? $G \to aG + b$ retains linearization if $a > 0$.

0

Time, Memory, Money are usually already linearized; Bandwidth, Maybe.

The moral is: if the thing to be optimized is linearized, don't ~~modify~~ replace it by a non-linear function of it.

It is notable that the optimization ~~to~~ G ~~[illegible]~~ (x) is only a well defined problem if G ~~has been linearized~~ is "linear"; i.e. for all G and true probability, p; 0<p<1, the utility of G with probability p is the same as the utility of p·G with ~~ce~~ probability one. ~~If we do not know if G is~~ Normally, there will exist a monotonic increasing function, H(·) such that H(G) is linear. If we ~~are~~ are asked to optimize ~~[illegible]~~ a non linear G and we are not given H, the problem ~~is~~ has not been completely specified and has not exact solution. We can pretend that G is linear and ~~solve~~ obtain an ~~optimization~~ optimization based on that assumption, but it will be an uncertain optimization..

0

A ~~few~~ few things done that need to be incorporated into report.

§3 1) The present version of section 2.1 "Improved optim..." has many notes & corrections that I haven't yet inserted in "report".

2) Insert ~~288.26~~ ; 30 at 14.20

3) Look at Latex book! See if Maple can insert pictures into ~~[illegible]~~ Latex
Ask Grace to look at Maple Books

4) ~~Insert 291.10-30~~ ~~[illegible]~~ at 13.16 ← (done)

0

Done → 5) ~~In~~ §3: After ¶ "After we solve ... in the present case" (55 ¶)
We have to rewrite the whole thing: 288.32 ff. is a start; Maybe have §3.1

6) Write about when to switch from Lsrch to ~~[illegible]~~. (See 289.10)

7) Insert 298.10-.16 (Done)

● 8) On "Creativity & Transfer Lrng": [299.11 : Also introductory data for Intro ~~what~~ "what a Machine Lrng is"] o.k.
299.11 o.k. not yet
299.18 is n.g. is a "pre" Introdn.

Imp't things not yet done:
1) How to make 5 Bunch!
Various kinds of :
Grammar v.s. ZIU.
etc.

Insertions needed to be typed:
288.26-.30 (not obly nec'y)

Some trouble: 2 "optimizations" used in this "Opt." method!

292

See
→ 298.00