.00: 378.40 : 378.00 – .08 is the main idea! The Q is: To represent only part of all possl. O^j.

We store a sub-distribution O^j. This sub-distribn can be stored in several ways —

à we can store various sizes of O^j seats, but our access ↓ as size ↑ ; | the most extreme case: ©
e.g. we can theoretically

store all of the finitely derivable O^j by giving the stock Grammar & Generator all of them.

A drift from the extreme, is to store a large bunch of O^j explicitely, as $\vec{m}$'s (centers of clusters) — à give a $\vec{\sigma}^2$ (metric) for near-by members of the clusters. The metric (as 378.29 ff)

could be grammatical: but as was pointed out (378.29 ff) simply giving the kernel of $\vec{m}$

as a string, and then concating on code to extend to $\vec{m}$ is not v.g. Better metrics

.09 are needed — O.K.: see (32) ©

.10 The stuff from 378.00 to here, seems to be on impt "simple" approach for the all over QATM

It certainly needs more "Fleshing Out"! 378.22–.23 is a major direction.

I don't immediately see how the "Bootstraps" $TM_2 = TM_1$ occurs, but the discussion of 378.29–379.09 suggests that TM_2 would find a good "Metric". Here "Metric" could mean

.14 a kind of conditional pc: "How much into the seat Q to A or from $(O_n, (QA)_{n+1})$ to O'_{n+1} d.f. ?

(R) is answered exactly by 378.02 so that's not quite what I had in Mind!

Hmr. 378.02 is not relevant to finite cc à finite sets of O^j's. → see (21) – for details

What we want is a way for TM_2 to expand the d.f. on O_n before O_{n+1} is created.

A good "metric" for this "clustering" is needed. The amount of "Expansion" should be about the same

.19 as the amount of contraction wedged by 378.02 → (21) (i.e. like 378.39–.40

.20 [I've been going around in a circle several times in 378.00 to here! Much repetition of same ideas!

.21 .19 After the "narrowing" of the d.f. by using 378.02, it has to be "broadened" again by → (38)

.22 Adding to the d.f. O^j's near the O^j of hyest pg. This might be done by mutation/crossover — But knowing

the "metric" for mutation à the "Crossover Atom", is the Big Problem. I intend to Goat in site on this problem by

From working on the TSQ. Any useful soln. of a new problem should be expressable as some kind

.24 of Xform on the old O_n or old O_n d.f. → see (32) for suggestion (≡ 382.11–15)

I think the Main Impt. ideas are: 378.00 – .06 then 379.21–.24

.26 Also note: 378.22–.28 à 378.29–.40, on "Clustering"; "Metrics": Also 379.00–.09 is a Useful way to look at the problem.

.28 (.22) NB It might work (but inefficiently) to use an "ordinary" GA approach for (22) using Koza-type

Crossovers à perhaps "Automatic function Generation". If it works but somewhat too slowly, we can

.30 .31 use the system to Improve its mutation & Crossover Atoms. (Bootstrap).

.32 382.11–.15 has some ideas on how to define such a metric, using stochastic trays w./ only 1, 2, 3, 4 ... known members.

.33 3/22/01 On the [illegible] the d.f. of 378.07–.11 i.e. that the set of O_n's w. $pc > \frac{1}{100}$ of pc of best O_n

becomes small as $n+1$ à eventually → only 1. ∴ no "ring" hence no Pr.

It is possl. that this set will not always be in size: particularly if it is large enough, to start out!

.37 One Big Diffy would be comparing $P(A_n|Q_n)$ for all these O_n^j's every time $n \to n+1$!

A Kind Improvement on all of this: Express O_n in terms of 389.00–.05! If this is done, then (present method 378.–379)

becomes much closer to 300–302! T. d.f. on O_n is expressed as the product of ind d.f.s of OV-Ops for each

$\{q_i, t_i\}$: $i = 1/n$. It becomes much easier to treat modifns of O_n. Also ? 389.00–.05 tells how to "Expand"

the O_n d.f. after (Q_{n+1}, A_{n+1}) is known

QAM as Name.

A Question Answering Machine.

.00: ; Confusion Betw. equo 377.26-(29) wrong! anyway. ≈ 378.00-.03 :

378.00-.03 seems rite! The ● P_0 (or 377.20): $P_U(A_i|Q_i) = \sum_{j=1}^{\infty} a_j O^j(A_i|Q_i)$ ← No! —

Look at 377.30. Maybe correct maybe should be ● $P_U = \sum a_j^0 P_j(A_i|Q_i) = \sum a_j O^j(A_n|Q_m)$

It would seem that the proof of the QA convergence Rem. would be about identical to the proof for Bay induction!

— Because In bay induction we have $\sum a_j^n O^j(A_n)$; $a_j^{n+1} = a_j^n O^j(A_{n+1})$

In QA " " " $a_j^{n+1} = a_j^n O^j(A_{n+1}|Q_{n+1})$

So Bay and the same except that in QA induction, the O^j of Bay induction is different for each value of n.

I don't know if that changes the proof much, hm.

What is the Rem/proof for sequencial induction (Sol78T3)?

.10 $P_U(X_n|X_{0:n-1})$

See 799.20 ff for recent stuff on "SUMAC" ("Summary Machine") — which is a development of 378.00

.20 I'm afraid my proof of the Bay conversion Rem (w.o. Radix 3 [Thm needed]) of 98TM 111.03-.17 may be wrong!

On ibid 111.10 I have $\Sigma_{K_i} = -\ln \frac{\prod_{i=1}^{n} P_i!}{\Gamma\sum_{i=1}^{n} P_i}$ is bndd. Hm, Rem couldn't be so! We can make n arbly large, & small coef. factors w. diff i are indep, I'd expect the ratios to Diverge as $n \to \infty$. [later in the day: the first real contradiction 380.23-.32 suggests that my objection might "not hold water" ☺.

.23 Go "back" to Sol99 ("Skinner"): The universal inc Prob of PC assignment to bays is

$$P_{M_U}([D_n]) = \sum_j \left(P(M_j) \prod_{i=1}^{n} P_j(D_i) \right)$$

$P(M_j)$ is PC of jth P.P. over $[D_n]$.

$P_{M_U}(\Sigma[D_n]) \geq P(M_j)\, P_j(\Sigma[D_n])$ for any one of the M_j — including the one that generated $\Sigma[D_n]$

O.K., so that Gives the "constant factor" Theorem for QATM!

.30 It looks like 378.00-.03 maybe "Not exactly rite" ← Actually it is exactly rite!

.31 The universal prob. is ~~not~~ 378.03, ~~Pr~~ ≈ Probly $(A_{n+1}) = \left(\sum_v a_j^{n+1v}\right) / \left(\sum_v a_j^n\right)$; Woops! This ≈ 378.03!

.32 T. "Constant Factor" Inequality: $\sum_{j=1}^{\infty} a_j^n \to a_k^n$ for any value of k — including the modal index that generated the data.

On applying 98TM 111.03-.17 to QATM: Yes bcm more probabilistically → 381.07

for the (Q_{n+1}, A_{n+1}): T. ratio of the PC's for A_{n+1} of QATM v.s. Generator (of index "k"):

$$= \frac{a_k^n \cdot \sum_{j=1}^{\infty} a_j^{n+1}}{a_k^{n+1} \cdot \sum_{j=1}^{\infty} a_j^n} \qquad \frac{\sum_{j=1}^{\infty} a_j^{n+1} / \sum_{j=1}^{\infty} a_j^n}{a_k^{n+1} / a_k^n}$$

Could we use Sol78T3 on the "sequence" A_{n+1}?

(As we did for the single "object" (like D_n) in 98TM 111.03-.17

It may work! The each A_{n+1} in the QA corpus has a different Q_{n+1} i.e. a different "Machine" associated, 78T3 may apply anyway. As before, the meaning (& correctness) of "Expected values" is unclear

.00 : 380.40: "On Expected values" re: 98 TM 117.11-12 "①": It may be O.K : for each * "object" in ibid .03-.10
T. "expected value" will be over * ensemble of "possl objects" at that "pt." . [I Guess this could apply as well to a particular
$Q_i A_i$ in QATM] : Also NOTE in * $O^*(A_{n+1} | Q_{n+1})$ * pcls are indp of * Q_i, A_i preceding $Q_{n+1} A_{n+1}$.
Hvr in the universal d.f. preceding
So, any way, what does from 380.30 ff Gives us: Gives us: $\sum_{j=1}^{n} z_j^{n+1}$ (cf 380.36) are order dependent. In fact we
that both normalizers do not depend
all of * (Q_i, A_i)'s that Ran $Q_{n+1} A_{n+1}$.
for $i < n+1$

1) A possl. proof of ~~Existence~~ ∈ conv convergence (Boundedness) Thm.
2) The probly return inequality (≡ constant factor) [380-32]
3) A restatement of how QATM does predn: 380.31 (≈ 378.03)
9/17/01 → T. "Constant factor" An equivl. but more revealing eqn: revealing equ.
380.32
$$\sum_{j=1}^{\infty} z_j^0 \prod_{i=1}^{n} O^j(A_i|Q_i) > z_k^0 \prod_{i=1}^{n} O^k(A_i|Q_i)$$

.10 Back to "T. report": [See 376.-20, 374,24, 370.40 ff for previous work in this direction]
20 Break down * Report into sections: First Dcrb. each section; Then dcrb. it in more detail; Then write each section.

Possl. sections: 1) Theoretical EA's! Discuss one based on simplest Dist.
2) Perhaps give .06, .05 ≈ .04
3) Fig .04 shows First * ~~problem~~ largest pc we got for our Arit. corpus, (Ran via Sol 78 c3) *.
Smaller pcs correct we get for our Answers.
4) 1, 2, 3 may be wrong; but $\prod P(A_i)$ is max; would seem intuitively, to be a reasonable Goal.

□ ABCDE abcde αβγδεη

Suggests that * individual $P(A_i)$'s are big ≈ cf. "rich answers".
5) Show how any universal (2) input function generator can represent stochastic fcns.
Our SAZ system does this and can assign a pc to each such function.

SN on 3 input comp. A, B, R → can be P(A|B) or P(B|A)!

.17 We have shown in section (how SAZ can
.18 * the cause * SAZ system (of §
probabilities them
.19 ~~pcs~~ to each of ~~them~~. How can we use this formalism to represent stochastic relations and
.20 system probabilities to ~~pcs~~ relations?
Suppose F(Q, R) is a function of 2 variables: ~~The F function generator reads all of~~
~~Q when invoked~~ The first variable Q, is a finite string. The second, R is a semi-infinite
random binary string. The function reads Q, then it reads as many bits of R as it likes then
it prints "A" as output and stops. ~~By defining F in a suitable way, we can sometimes~~
~~any problem~~ for any P(,) and any Q, F will have a certain probability of outputting
~~any~~ any particular A. By suitably defining P(,) we can ~~represent~~ use it to represent
any finitely describable probabilistic relation between variables Q and A. We can use

In exchange for promised security (which will have maximal risk) Our Senate has given up all our civil liberties.

As a result, ~~SAZ will further discuss~~ section ~~—~~ to gives us a way to represent
arbitrary probabilistic/prob stochastic relations and it enables us to associate ~~prob~~ probabilities
with descriptions of these relations. (~~Omit~~ lines .19 -.20).

194

.30 The formalism of SAZ to assign probabilities to ~~the proposed AGI theorem~~ descriptions of functions such as F(,).

(Once we are able / With our ability) to generate arbitrary stochastic functions, and assign probabilities to those functions, we can begin to work on the problem of maximizing eq ( ).
~~Assuming we have~~ Suppose we are given a short sequence of small (Q_i, A_i) pairs. ~~and it is small~~.
We can use the formalism of SAZ to generate trial functions, $P_{on}(A_i|Q_i)$
in attempts to maximize eq (.0). SAZ makes it possible to generate functions in approximate
order of their ~~descriptive complexity probability~~ description probabilities. This, in turn, makes
it possible to use Levin's Universal search algorithm (Lsearch) for optimization.

8/19/01
ID

.00 SN Just how does Lsrch work for finding short codes? We can actually find t "code of x"
$(c(x)) \ni \frac{2^{-|c_i(x)|}}{T(c_i(x))} = \text{Max}$ T(c_i(x)) is time needed to test t. ith code for x.

By using more time (cc) we can (usually) get shorter Codes.

So Lsch would work for small enuf corpus, But what next? We'd like to do "incremental Updating".
GA (on Koza's style of function trees) would be dif'lt because testing time for each cand would be very large. We could just test random (Qi|Aj)'s for each cand. This would give apparent G's. Or a certain subset of them that seemed more "critical".
Then "fight it out" among t. cands of hy (apparent) G's.

.09 SAZ generates an ordered sequence of functions (trees). — Each funct can use as subfuncts,
.10 any function (tree) preceding it ....
.11 Could we use t. forgg. to devise a suitable "name" for t "GA"? — When I last worked on this problem t wanted a "Grammar" that would give the known cands & their Gorcs.
Using .09 and the idea of "crossover" being "extrapoln W. ssz of 2" we should be able to get good crossover rules, using a simple grammar & OSL. We can perhaps get "Mutation" from t. same Grammar!
Not Bad!! Needs Work here
.15 **Its a nice way to get extrapoln. W. ssz = 1, 2, 3 ... ect.;**

Th. unclear part is how to get a good association of empirical Gorc w. t. ass of t. lang.
One obvious way is to have a certain common set of (functs, trees, concs.) for all of t. Grammars then for each range of G
Values have a somewhat dif'rnt. Grammar.

Another way (that did not, as I remember work so well) is to use the p.c. of t. stoch grammar "acceptable Sentence" as t. G value assoc w. each as. T. trouble was, that we had to have short derns t. have hy G.
.20 On t. other hand, it should be possl. to make a grammar in which t. "case count" of a particular string α as is made a funct. of its observd G value.

So t. forgg. is o.k., I think I had more or less solved (adequately, protem) t. problem of Continuing the update algm. (past initialization). I don't quite remember just what I had in mind, but!! try
.24 recording that old stuff! ~ 33 (302)
361.12-.20 is not bad! "Hodging On to get On's hats" is not very clear! Perhaps I was thinking of some kind of "Metric" like (.11) or (.11-.15) for a "GA" approach to "Updating"
369.20-.40; (Also maybe 370.00-.29 on GA) ; Excellent; Expanded 300.36 — 301.05-.09 as a "final soln"
to QATM: (parts 300.34-.35 to ... to be resolved. See 301.10-.20 also 301.21-.40 + 302.00-.40
(260.01-.25) "Some Updating Tricks" ← Very good: Some General, some specific.
(260.08-.33)
.30 378.00-379.40 Is a general plan for "QATM as a whole": It has a rather large "unworked out" part, but it's non-t. less Very promising!

(260, 261, 300, 301, 361.12-.20, 369, 370, 382)

Main notes so far on updating:
260.01-.25
260.08-.33
261.12-.20
300.36—
301.05-.09
301.10-.40
361.12-.20
369.20-.40
370.0-.29 on GA
378:00-379.40
382.11-.15

Some imp't ideas Not yet Considered for "report" but which I want for t "Macro Report":
1) Updating by finding Common Subtrees in hy Gorc Cands (= On) (+ OSL)
2) The (Hessian) Gorc for continuous part of optzn of functs.
37

9/21/01
IDoc...

80M mils 40m" $60 \cdot 60 \cdot 24 = 86.4k$"/day
= 463 days.

379.40 ← Most narrowly
378.49

.00: ~~[illegible]~~ on Expanding the D.f. on Q_n before $a_{n+1} = a_n \cdot P_{Q_n}(A_{n+1}|Q_{n+1})$
This appears to be t. main problem of opt. General approach of 378.00–379.40.
379.28 has some 1 deas on this — (there may be other ideas on this scattered thru

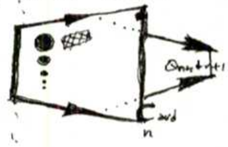
37800–379.40

[right margin note: .00 conceivably would not be necy if t. n. values of Q_i's were large enuf! i.e. t. "width" would not always be a_{n+1} — it may ↑!]

.04 A possibly V.G. approach to "expanding" a pd on Q_n. This pd does constitute a precisely defined stoch lang. (the Σp_i may be < 0, it can be normalzd). i.e. We took a certain set of Q_n's we did t. xfm of .00 on $a_n \to a_{n+1}$! We got a new set of a_{n+1}'s!
Those w. $a^i_{n+1} \geq \frac{1}{K}(a^i_{n+1})_{max}$ are this new set. They de v. a_{n+1}'s assoc w. them. These a_{n+1}'s are relative probabilities: But there are missing members of t. Q_n set, that would have $a_{n+1} > \frac{1}{K}(a_{n+1})_{max}$, but they dropped out earlier in t. learning of t. TSQ.

.10 So, we could take t. present Q_n set defined by t. new a_{n+1}'s and expand it by some grammatical (or other) extrapoln. device. To find t. assoc a_{n+1}'s would be expensive; but one way!
Generate a large no. of cands (e.g. in GA) & test them statistically; until a logenuf no. of
.13 them w. a_{n+1} > threshold have been found. If we can't do it, we are "stuck".
.14 Some ways to get "unstuck": Continue (.00)↑ w.o. expanding: do it for about $n \to n+2$ or $n+3$,
.15 Then repeat .10–.13 & hopefully, it will work (or one will be stuck again & forced to try .14–.15 again.

Just Before (.04)↑ I had t. idea that this "expanding t. D.P." was one way of doing a generalized kind of "Backtracking". That we had to find some way to recover t. Q_j's that were (unfortunately)
.18 deleted several "a_n's" back. One View of "Backtracking": That some time ago, we took a choice of branch of max pc. That we should now go back to that ~~branch~~ pt & examine several branches of lower pc.



.20 This is ≈ t. idea of going back to n–i; lower the threshold enuf, so there are many more cands; so when we finally do $a_{n+1} = a_n P(A_{n+1}|Q_{n+1})$ we will have enuf cands of adequate a_{n+1}.

.22 HM! I had been thinking of using a very large initial set for Q_0. — i.e. small a_0
.23 This would mean they would include (mainly) Q_i's of fairly large dern length. as we do xfm of .22, we modify t. df of a^i_n by hyper hyperbo — but we leave t. dern length of t. Q^i's in t. corpus invariant. Th. dern lengths do not ↑! This Seems like a Bad idea.
The idea of More or less Simultly ↓ width by .22 & ↑ width by ½ mut/crossover etc, seems more reasonable.

.30 We mite get a legit "expansion by backtracking" if in .18–.20 we were allowed to modify t. Q^i's by — perhaps by ↑ their lengths.

Re: .23, I think t. dern lengths of Q_n has to ↑ w. n. (but very slowly we hope) — this Does mean ↑ in size of t. Q^j set.

9/23/01 Q: Actually, t. dern of t. corpus, $[Q_i, A_i]^n$ does include increase w. n, because the R_i are included in t. total dern. length. A question is, does t. size of t. Q^j set have to ↑ w. n.? If t. Q^j_{max} dern stays t. same as n↑, then "lrng" has stopped! — even tho t. response to a Q_i may be "probabilistic" & ∴ involve ↑ in length of dern. of corpus. "No lrng" means "No new Theories" — which mite occur in certain small areas of science, but not (I believe) in most (uncertain) research. It would mean (in t. "2 part code" version of ALP) that t. first part of t. code is invariant (except, perhaps for t. precision of

ID

40 G = 4×10^{10} bytes 3.2×10^{11} bits. ~ ?? = $2^{38.22}$ bits ($\equiv 2^{35.22}$ bytes?)

(826) P

.00: : continuous params. (Like $pc \tilde{\propto} n^{-\frac{b}{2}}$ w. b continuous params)

.01 In General, (say we are doing a Algebra TSQ): we look at the individual soln(s) to t. first set of problems, then we make a grammar that "passes thru" these solns. It is a Stoch Gram, & we use it to search for solns. to subsequent TSQ problems — i.e. it Generates an "Ensemble" & we Lsch on this Ensemble, for t. solns. T simplest kind of Grammar is AZ1 (≡ SAZ) which counts freq. of occurrences of sub-trees (plus OSL). Finding subtrees & & OSL, & estimate of cc.

.06 [This "Search for solns" would not be t. usual non-tel search of finding a O^i of (max) hy pc, but rather "finding a soln to $Q_i \to A_i$ of hy pc" than finding a way to "recognize" present O_i is difrnt from previous Q's. This makes it unnecy to verify new O^i's wrt previous parts of the TSQ. I think that the great majority of induction is of this kind. T. only kind that don't

.10]

.11 involves Backtracking or Equiv.

.12 Also, Periodicly, t. narrow routine of .06 — .10 is "integrated" by looking at a set of recent problems solns (INDex's assigned by instructor or "Recency" attempt. SC types or some other way of categorizing a sub-corpus (≡ "SC")) and trying to integrate those solns w. a "Shorter Code". This can occasionally result in a "Great Break thru" i.e. Much Compression. This is Newton integrating Kepler's & Galileo's "compressive laws".

.15 (Also Give reference to this sort of thing discussed (Briefly — a few P's) in Sol 86 — it's

.16 (Also Give reference to this sort of thing disc... (Also Note 261.13 — .18): this integrates .06 — .10 into a less "el." system)

.17 t. discn of "Why put all info into t. P.D.?". (Also

The "Ob" that is able to "recognize" t. new Q_{n+1} can also help decide what kinds of trials to make to "solve" it (Not so clear, hvr — in .06 — .10 wrt t. "Recognition" function (≡ Ob) is devised "Post Hoc" — after we knew how to solve $Q_i \to A_i$). Perhaps, after we have a lot of ob. cases, we can begin to

.20 extrapolate obs & use past correlns w. parts/aspects of "OPs" to narrow down search for solns to new problems. Hvr. see 261.13 — .18: This tells how to get useful extrapoln as well as "identifn" out of this ≈ ~~Post Hoc~~ Ob

This integration of several Obs in a PC of corpus Conversely — When several different obs are replaced by a single ob, but the OP's that point to many Obs need not be "equivalent", but the OPs that point to many Conversely, also replaced by a single op. So many Ob → OP: replaced by a single ob → OP.

So perhaps outline: How to do Update Global for small Corpus (2)

(2) Incremental (a) using .06 — .10 & also .12 — .16: Also maybe discn. of Obs/ops Algebra?

(b) Using ≈ GA for Global updates: Excessive cc of verifying entire corpus: use of carefully (critical) chosen/sample QA's

(c) Perhaps some other update methods in 260 — 261, 300 — 3??, 361.12 — .20, 369 — 370, 378 — 379, 382.11 — .15) gpp Con..?? ideas?

See 382.24 — .30 for list of refs.

An attempt to list impt ideas!

260. Division of corpus into SC by indexing (recency) (301.29), (261.34), induction from indexing.

360.12: Common to many EA's (a) update pc's of cases (b) discover new sub-cases (≡ subtrees) reparse (

.30 { .19 — .25 find special operator for $Q_{n+1} \to A_{n+1}$, then find "recognition ob" for Q_{n+1}: (Note perhaps impt improvement of 261.13 — .18) (Note (.06 — .11) also (.12 — .16))

→ { 261.19 p.d. $O_{n+1} = O_n$ (t.O'_n, Q_{n+1}, A_{n+1} & other context) v. o. or "correct" 300.36 — .30 & 301.36 — .40 are good discussns.

300.36 — 301.09 is an approach to "final" soln" — this is all v.g. — See 385.00ff for discn of 300.36 — 301.09 v.s. 378.00 — 379.40 As poss "final soln"

369.20 — .40 Summarize (11) key impt update (ideas) methods ← GA built in as impt possl way to "Bootstrap" TM.

What this listing lacks is (378.00 — 379.40)'s matl

See 385.00ff for discn — It has Been inserted as (#11); (.30) was inserted as #10

How does t. "Grand plan" of 378 — 379 differ from Proposals 300.00 — 301 — 302? Well 300 — 302 is a TM$_1$ = TM$_2$ "S.T." system 378 — 379 is open loop (At first!) 300 — 302 seems less "el" — more Genl. (see) → 385.00 —

.36

ID d.f. O_{n+1} = d.f. O_n (d.f. O_n', Q_{n+1}, A_{n+1}) ||| $a^j_{n+1} = a^j_n \, P_{O^j}(A_{n+1} | Q_{n+1})$

[300–302] V.S. [378–379]

[M1] [M2]

(379.36–.40)

.00 (384.30–36) R $\geq$: [300–302] M1: Has t. problem of Generating (d.f. O_{n+1}) from (d.f. O_n) & $(QA)_{n+1}$ & "contextual M60".

[378–379] (M2) Obtains d.f. O_{n+1} from d.f. O_n; $(QA)_{n+1}$ in a very direct way — T. problem being "How to **Expand** (d.f. O_{n+1}) after it had been obtained by "contracting" O_n : we do this by taking pts "near" to d.f. O_{n+1}. T. nature of t. "Nearness" Metric, is t. BIG Q.

In [300–302] (M1) O_n (a a posteriori d.f. O_n) automatically gets a d.f. for O_{n+1}, which we may or may not need to check w.r.t. t. Gore of t. new d.f. O_{n+1} w.r.t. t. entire corpus $(Q_i A_i)$ $i = 1 | n+1$.

In [378–379] (M2) On way to look at it! At d.f. O_n point, we look "near" d.f. O_n for "solns to $(QA)_{n+1}$". When we find enuf of them,

O_{n+1} is "Wide enuf" so that we can go on to $(QA)_{n+2}$ [An improvement would be if "nearness" fnct. depends on $(QA)_{n+1}$ as well as other "contextual info" ← as 300.26 ff) d.f. O_n & any other contexts (300.26 discovered Genzd "context")

In [300–302] (M1) T. search is more "intelligent" than 378–9 if its guided by QA_{n+1} as well as O_n

.09 Conceivably, use 378–9 until TM is smart enuf to do 300–302 usefully. → 391.00

.10

.20 369.20 — .40 Summarizes t. state of research on "updating" up to th. present pt. (there may be other impt. ideas not referenced in it, hvr.)

An orderd way to desrb this!

.23 1) Give Global Gore; (w. poss. wrtd. appendix/f.t. on cc problem).

T. Gore is usually the a set of O^j_n. (≡ "ensemble")

A possb. way do 378. update algm. for SS = ∞. (Maybe give thrm on MS error? or mean of it)

.25 Discuss Using small set of O^j_n's, & what I expect would happen (why it wouldn't work).

How narrowing might be prevented by expanding via a closeness metric or GA mutn/crossover.

Closeness Matrix could be obtained from Grammar fitting (SAZ type grammar),

.28 So p.d. O^j_n, $(QA)_{n+1}$ → p.d. O_{n+1} : $[a^j_n \; O^j]$, $(QA)_{n+1}$ → $[a^j_{n+1} \; O^j]$ ← some O^j deleted, " O^j added

use of fixd algm for expansion

.29 **Better** : pd O_n (p.d. O_n, $(QA)_{n+1}$) → pd O_{n+1} (300.00–302.40)

.30 Do .29 when TM is capable of understanding & usefully working on problem.

Difty .25–.26! Testing a new O^j_n's to be added in! involves statistical testing by trying O^j's on small no. of $(QA)_i$'s (or do "sequential testing" (Wald)). —

.33 — But mainly, I'm worried that t. system would not "understand" that It should be making O^j trials that are certain to leave Old O^j results on almost all $(QA)_i$'s invariant; It should concentrate on $(QA)_{n+1}$.

That GA seems able to work probs of this sort is of interest! Also idea of using "Grammar based"

Hvr, it would seem that w.o. t. "understanding" of (.33) updating would tend to take longer & longer as n ↑.

The hope, is that somehow we'd Get into a region of O^j space where new trials tended to

.34 ⟨⟨ leave all old QA's invariant, — & work mainly on latest $(QA)_{n+1}$ ⟩⟩

A related approach! to try to find cross/mutn that did (.39) → 386.00

9/24/01

ID ID

I D

130 2:14A

1-16-99 w. Paul

(302.00 – .31)

.00 : 385.40 : Lsrch will be involved in some of the forms. (Tho not if GA is usd)
If Lsrch is usd, mention [Incr. doing] srch (so it's not strictly Lsrch! one can't estimate (time) to soln)
Also, perhaps, "context dependance" of pc's of concs, etc. See 300.00 ff. for discn.

.03 —— Another way to get $O_{n+1} \approx O_n(O_n, (QA)_{n+1})$ is by first getting pc's of new concs by
.04 1) Context free: uncond. pc.
.05 2) Varying amts of Context
.06 3) Then try for the "conc" O_{n+1} as a fun. of $O_n, (QA_{n+1})$. → .10 tells how, hvr
How 2) moves continuously to 3) is unclear! — The On 300.00–302.40, it did occur! Perhaps it was a
[illegible] nonal jump! I guess the idea of concs. & condl pc. of concs was only a method of getting
O_{n+1} trials, so if we could do this Directly, it would be Better ("less el.").

.10 —— One way to get to .03 is by Elzn. We have many QA pairs for the .03 induction: [illegible]
One elzn. would be to find good concs & assoc (context dependent) combination rules to create good
O_{n+1} trials. We could start out w. context free concs as .04: This (probly) may be the same as the
pc's of the concs that we use to evaluate the "code length" of O_{n+1} for the final Corc –
.14 As SSZ ↑ we have more cases of relevant contexts, & we use them to obtain condl pc's of concs. From there → .03 or I guess .06 (=(3))
i.e. we wouldn't otherwise, hareness concs, heurs.
Actually .10 ff may be a nicer path to get to .03. One mite get there (alternatively) by GA, hvr, but I suspect
.16 much slower.

.10

.17 The .03 ff. & [illegible] in particular, are certainly good ideas, [illegible] the resultant system does not
.18 deal w. the 385.33 problem of trying to find a modifn of On that preserves (some) old solns & solves new problem.
preserves
.19 384.06 – .20 + (384.01–.15) R Reviews some ideas about this
—— So .00 – .16 is a possibly not bad way to do (300–302) — But .17–.19 are very impt but not included.
.20 A possl. way to integrate: we startout [illegible] TM by [illegible] (using) .17–.19 only to solve problems.
.21 we [illegible] improve? .17–.19 by .10 – .14 &, [illegible] minimizing in .03 1–1
The part I'm least certain about is 2)→3) (.05→.06); An & "soln. outline" in .10 – .14.
But drop this for a while: Say .03 – .21 all would work. Could we integrate 378–379? ($a_{n+1} = a_n \cdot P_{O_n}(A_{n+1}|Q_{n+1})$)

378-9 so far: [illegible] 385.00 – .09 discusses 300–302 v.s. 378-9: 378-9 is easier to [illegible]/prog [illegible] – particularly
If I usd GA for "expansion of the D.F. — But it seems not very smart — seems not to use heurs, [illegible] etc.
It Mite work, hvr. I certainly feel 300–302 is more likely to work!
A possl. reason to believe 378-9 Mite work: That GA was able to solve what seemed to be "Hard problems"
9/25/01 Also 378-9 seems a lot easier to (implement/try) than 300–2. A main thing study needed was the "Distance"
metric — presumably devising Grammars [illegible] using SSZ = 1, 2, 3 etc. & SI write consist of
Improving the Metric — making it dependent on $(QA)_{n+1}$, O_n (& "carton" of correct On d.f.) & perhaps other things. — This would bring it
.30 Close to (.03) (≈ 300–2). The reasoning here seems to be: How can we improve the "Metric"?
.31 An essential Genzn of the Metric is a mode of "expansion/modifn" of the On d.f.:
d.f. On → d.f. O_{n+1} is what we want, & we're obtaining it by d.f. O_n → expanded d.f. O_n $\xrightarrow{P_{O_n}(A_{n+1}|Q_{n+1})}$ d.f. O_{n+1}
2 xfms, They could be in opposite order, hvr. — Tho I'd think that the present order is best
because we do want all of the new Concs to work w. $(QA)_{n+1}$. (?)
So .31 ff does seem to "Merge" 378-9 w. 300-2.

→ 388

[D]

.00 : SN A way to use GA for a "Problem Pool" TSQ:
We have this unordered set of problems (≡ "Problem Pool").
We try various functions (obtained by GA) on the problem pool. At first, No functs solve any probs;
.03 but we cross/mut. ~~~~ the set of functs that solve n problems: As soon as we get a funct that can solve n+1 probs it goes into the "n+1" function pool: so n→n+1 & loop to .03 —
So we have many function pools (one for each value of n), & they recombine w. members of that own pool (perhaps occasionally w. functs of n−1 pool?) So we have pgms diffusing to higher & higher "castes" (caste no ≡ n).
The trainer will observe what problems are diff't to solve (few solns) or no solns at all, & try to devise "bridge problems" to solved probs: especially probs solved by many functs (?).

.10
.11 Out. Problem of 386.18 (solving new prob & preserving solns of old probs (intact!).
Normally, when one solves a problem, T. soln. method is then available for Exactly the same problem. Also it is available for certain known mods of t. problem (e.g. 3+7→10: ~~~ any soln will usually work for a+b→[a+b].
In general, for each problem solved we will (guess) (know) how to extend it to ~~~ "related" new problems, usually by slightly (or much) modifying the soln. I think this may be the point of t. 386.18 question. Normally each problem soln. method will have an assoc. algm. that will tell when that exact method can be used a/o how to modify the soln. method for diffrnt (associated) problems.
Unfortly, this QATM can't make hypoths about what kinds of new problems the newly found soln method will solve, & give pos. or neg. examples to "Teacher" for pos or neg verifn.
.20 The way it works, is that t. method of "identifn" of t. Qs is a short code, so it automatically extrapolates Qs.
E.g. $\overset{R_1}{3} \overset{R_2}{+} \overset{R_3}{7}$ → sum 3,7 → sum $R_1\,R_3$. sum $R_1\,R_3$ works for many examples in t. TSQ., so it's very compressive ; 3286186622 + 41190025 → (??) sum $R_1\,R_3$ w. ~~~ there a big mess here and can is also very compressive.
sum $R_1\,R_3$ is a great extrn from t. original data.
.07 For all 3 register probs; if R_1 & R_3 are nos, then sum $R_1\,R_3$ works if R_2 ≡ "+". ←
(.07) is discovered Empirically.
.30 So, ~~~ .11 — (here) conforms my previous ideas that when one solves a new problem & augments On w. this soln, it is usually (almost always) unnecy to verify that this On will continue to work on all older problems. In fact, analysis like all ff is normally used for all problems. ∓
"Backtracking" is a special, rarely invoked technique.
(or serially / sequentially)
What it looks like is a classic Expert System: the "Obs" work in ll: when one of them "fires", t. assoc "OP" is exted. Since there are an enormous no. of "Obs" to be run (they could run in ll) One should find ways to ↓ testing time: this can be done by ~~~ looking at t. problem & roughly categorizing it (EDS?) so one knows which obs to try first. (trying all (or many) obs in ll saves time, but not nec'ly cc!) — Note 396.26 on Hierarchy of Obs to ↓ cc of "Recognition".

→ 388.00

Context: .06ff

.06–.09 gives t. most General defn. of a "context". i.e. T. "firing" of an "Ob".

.00 : 387.40 : Instead of a single Sequentially written pgm, On will consist of a bunch of **op-ob** combinations: that are allowd to use other ob-ops (or just ops or just obs) as subroutines. I think this may modify, considerably, t. way I will be thinking about these pgms. T. individual ob/op functions can usually be determinst. but can be stochastic. — This last would be a nut to give a stoch output for any funct "containing" a stoch funct as a "subfunct". Any of t. functs can be implemented w. t. "R" input, or they mite be expressed in any of t. other ways. see (.05–.09) R→

.05 R: Make use of variety of ways to express stoch functs. If may be that there are only a small no. of ways to get Ran, & all other ways are combos of functs of this sort & other stoch or Det. functs. 244.07 ff has some stuff on this.

.06 T. mode of (discovery / Construction) of .00–.05 will be same as befor (386.10–.16). When ever we are (during stochastic construction of an (op/ob)) a certain context occures, an assoc Ob will "fire" and t. appropriate pc's of funct. will be enabled. Much cc will be spent looking for new "contexts" that
.09 may usefully modify pc's of sub. functs. → see 396.30 for more on "context" (.06–.09)
.10 T. "Usual Contexts" are ① what functs are being Connected to? ② What is main problem being solved? ③ What (local) sub-problem is being solved? 300.26 was big expansion of t. "context" of .10–.11
.11 Other contexts will become clear as I devise TSQs & their solns.

So: How to go to "context" for soln. to specific problems, To attempts to devise On?

Well, in t. approach of .00–.05, There really Is No "On". There is only a set of attempts to solve t. n^th problem, "i.e. defn. " (Q_{n+1}, A_{n+1}). If all ops used to "identify" new problems are stochastic, Then finding O_{n+1} would mean improving methods of solving $(QA)_{n+1}$ and perhaps improving soln of previous problems. ("improvement" can mean ↑ of pc of answer a/o ↓ of cc of answer for past problems)

.17 The Goal is as befor, to ↑ $\prod_{i=1}^{n+1} P_{O_n}(A_i/Q_i)\cdot P(O_n)$, but On is in a difrnt form from what I was thinking about when .17 was first written. — It's like (.00–.05)^S.

.20 Hvr, it should be possl for T. presently contemplated QA TM, to be given t. problem of improving itself! As befors t. Q is "Orthonormal defn." [t. defns of t. ensembles O_n; Q_{n+1}, A_{n+1}; plus previous QA pairs of this type — i.e. $\left[\begin{matrix}\text{d.f.}\\ O_j\end{matrix}, Q_{j+1}, A_{j+1}\right], \left[\begin{matrix}\text{d.f.}\\ A_j\end{matrix}\right]_{j=n-k}^{n}$ k can = n

Also part of t. Q would be an "index" telling TM what t. problem is.

So, as befor, TM can work on S.I. as a "Normal QA problem" — after it "understands" what t. problem is.

The fact that On has t. form .00–.05 rather than a string representing a single function of
.28 3 variables ($S^\circ, Q, f_\circ$), should make no big difference in my discussn. of how QATM works.

.29 Suppose we give t. QATM of .00–.09 a new problem Q_{n+1} & that Q_{n+1} does not cause any "recognition ob" to "fire". Then TM can't work that problem! Seems strange, hvr, since I can
.30 almost always "work on" any new problem. In this case TM was unable to even work on Q_{n+1} at all! Well, say I was given a problem in a lang I didn't understand — (So first sub-problem is to lrn to understand t. language!). Well, there are problems that I can't work on because I feel they are not well enuf defined.

Hvr, it may well be that a suitably educated QATM could usefully work on any problem that I (or any human) could usefully work on.

Expanded context & perhaps Lrng during Lsrch, were main tricks I expected to use for better updating. (i.e. changing guiding P.D. during Lsrch)

.37 I do, hvr, need a way so that Any Possl. Hvr can be expressed. My earlier idea was that [modifying t. P.d. that Guides Lsrch & Lrng dury Lsrch (say & fine shore betw. Lrng. & Lsrch)] mite be adequate (i.e. .37–.39)
.39 to express practicaly any hvr. Perhaps write a bit more detail on how this can be done — particularly in t. present (.00–.09) QATM scheme.

9/26/01

Def: ID

Def:

D = Det. = Deterministic. D.funct., Dop, Dob.
S = Stoch = stochastic = Probabilistic sfunct, sOp, sob.

.00: 388.40 ; Comma Backmark situation! It occurs in a sc. A certain conc. has had it's case count recently ↑ so it can just now, be used for last 3 or 4 cases, giving (↑ of pc) & change of code for last 3 or 4 cases. This new conc can be Ob or OP.

In General, we want to have t. Ob & OP (found/discovered) for a new QA to be as general (i.e. as "short in code") as possl. Otherwise, it's a very narrow soln, we haven't lrnd much!

— So a large fraction of TM's time will be spent integrating (i.e. Genzing) old solns of old problems.

— Like devising an OP selector that works for many old Q's. Early Obs only select one op. Integrated Obs select (many) different Ops.

"Advanced" Obs will analyse a Q & Generate a set of Ops to solve it. By "set of Ops," I refer to a D.f. over Ops. (See 388.05R–09R)

.10 So: t. initial soln. of a single problem (≡QA) by an OP & finding an Ob for that QA, is an initial element of t. general QA problem. — That improvements of .00–.10, wrt. Gorc or

.13 max $P(O_n)\cdot\prod_{i=1}^{n} P_{O_n}(A_i|Q_i)$ are necy & amount to t. main final Gorc of t. system.

In .13, I'm beginning to lose t. feeling of O_n as a P.D. (≡ ensemble) — The t. $P_{O_n}(A|Q)$'s can be p.d's (ensembles).
(perhaps should be "j" not n.)

Also Note! For a new problem (QA pr) t. system will not recognize Q. After TM has found a good Q→A op, T. fact that This Q was not recognized, should help find a recognizer. — i.e. look at t. logical decision pts. at which t. present Q was rejected. Actualy, t. ob. that is t. total rejection by all previous obs, is an ob, but Even probly not a good one. Is it expensive & in pc? One would have to look at all previous Obs — which sounds Expensive. Also all old obs are Mut. Exclusive: No overlap (usually or always?) | Another possy: Q is accepted but A is given a low pc → 390.00

(i.e. "WHY" did Ray reject R is Q?)

.20 So: A possl. outline of report: ← precisely

(1) Desrb. QA problem: Also Go into How/any problem that can be expressed to human could — Work done by QA/TM — (Had to clear that this is close to a general form for all problems)

(2) (b) Give ≈ Genl. soln. (.13) then give ↗ as apparent | first (a) Give exact genl soln. first using Σ over all O_n^j (≈ .13). (cc = .10) Perhaps give explanation in terms of wts. a_n^j of → 378.00 – .06

so (1) then (2a) then 2(b).

Also discuss .13 when O_n is a d.f.

RECOGNITION Problem

Perhaps write 2 or 3 Reports on GA & IA's ANN

.26 (3) How to approximate 2(b) (≡ .13) — First, do each (QA); in sequence, using sub-functs of previous solns when relevant. Discuss "Recall" problem. Obs, Ops. & t. desgn of 388.00 – .40 on & t. initially fragmented form of O_n's & how to "integrate" it, "An Analogy" we make trial ... using context (uncondl. pc of trial) — then gradually ↑ context.

Also 2 The 378-9 system, w. Criticisms, See 385.00 — comparing 388-9 to 300→302.

.29 In particular discuss $O_{n+1} = O_n(O_n', (QA)_{n+1})$ mode as t. first goal.

Perhaps have a FAQ section at end! Try to antic. objections, misunderstandings.

.30 (4) TSQ's Tell how tsq will be used train both instructor an Instructed. Training by instructor leads to Modifn. of (a) TSQ (b) Mode of operation of TM.

(5) Maybe section of FAQ's.

Infrequently asked Q's.
Occasionally Asked Qs
Sometimes
Difficult to Answer Q's
Impossl " "
Embarrassing to be asked Q's
Embarrassing " "

→ 392.00

.00 : 388.19 : 288.29 discusses what happens if a new Q is not recognized by any ob.

.01 What if several obs recognize Q? How do we (or do we) mix their output d.f.'s for A? (Also Note .37)

It seems that I've worked on this problem before! I had several IA's, each w. somewhat different S & C's (sub-corpi = datasets) perhaps & problem(s) leading up to the analysis of ~ 250. ← Looks very much like present problem!

There is some possy that I did an earlier analysis of a problem closer to present problem.

I vaguely remember something like this: I had a bunch of IA's, & each would have an assoc. SC. Some of the SC's for the diff IA's overlapped somewhere. In this earlier version, the IA's were dets, so they either predicted the rite A or not. For the whole corpus, consider only data (QA's) in which there was some overlap. We wanted to assign wts to the IA's so that if we used a/sum of wts for each & predn, & recorded this normalized sum as a fn of pc, then $\prod_j pc_j$ (of those IA's which were right of QA's w. overlap), was max.

.10 I think the idea was pc_i = (sum of wts of IA's that predicted correctly for that Q_i) / (sum of wts of all IA's that predicted for that Q_i)

.11 In the present case, each predictor assigns a pc. to the correct A_i (see — a pairwise tabulation of overlap of SC's is the same).

.13 Try $pc_i = \frac{\sum_j w_j \cdot p_j^i}{\sum_j w_j}$ (i = superscript) ← (.28, .30) This pc_i would be the pc assigned to A^i, the i'th possl. answer.

So we want to maximize wt. assignments to the $\prod_k pc_k$ is max, pc_k being the pc_i (of .13) assoc w. the correct A of $(QA)_n$ (this product is over only those QA in which overlap occurred).

To do this maxzn, we note that in .13 if we multiall w_j by a constant (it is invariant) so we want to do maxzn subject to constraint $\sum_{\text{all wts}} w_j = 1$ or any other constant.

Could we do L's Multipliers? In the case of .11, I think I found a way to break it as a Bern Seq. Bern Seq.

248.16 does deal w. several A's for same Q. Also Not 248.28, .32

Hm, read 250.09L – .19L It sort of tells what this stuff is "about".

.20 NOTE: T. idea of 250.09–.19 was that I could do the Bern Seq. & / gained Laplace's rule. for this case.

$\int\int\cdots\int p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \cdots d(\ )$ of $p_i \geq 1$, $\sum p_i = 1$

AH! I wanted this $\int$ because it represented the total pc of coding the corpus using all diff wt assignments.

In 250.09–.19 Dany Pris $\int$ meant the difrnce factor using "Lap's rule" vs. "Straight rule"

T. "straight rule" corresponds to using the peak value of the wts, for predn (which is what I had in mind in .13) — which is OK if SSZ is > 5, say.

I guess what we want is $\int\int\cdots\int$ pc's wts, $\sum w_j$ and ideally, we could use Pris for predictors — i.e. see how this $\int$ varies when diff pc's for the new A are used.

.27 Earlier I did work that involved finding peaks of products: I used the fact that N. Linear often involve

.28 (13) To get the peak take ln then $\frac{\partial}{\partial w_j}$: $\sum_i \left(\frac{p_i}{\sum w_j p_j} + \frac{1}{\sum w_j} \right) = 0$ + 1 more eq. — Actually, practically any constraint will do!

.30 If I can't find a simple approxn, one way is to use the peak of .13 (which corresponds to "T. Straight Rule")

T. straightrule is bad for low SSZ! which is probly case of interest, so to get the analogue of Laplace's prob

To do this we use the reasoning that gives Lap's rule, (Pre-multi-dimensional integration) & apply it to ~ eq(.13)

.33 à la (.30 – .33)

O.K. for the present: .13 & .28 is probly good enuf for present. Probably it's possible to figure out a good approxn to "Laps rule" by modifying the solns to the (Str. rule (of .13).

→ Also try various analogies betw. Bern Seq. & present problem, to suggest approaches & approxns.

.37 .01 9/30/01 T. Poggio is fine if all of the (1 or at least one of) the obs given a DP of that "works" acceptably.

→ If none of the recognizing obs. gives acceptable results for (Q_{n+1}, A_{n+1}), Then we have to (a) find a new ob-opdt. to solve (Q_{n+1}, A_{n+1}), & also modify old obs, so they don't recognize Q_{n+1}.

$\frac{a}{x} = \frac{b}{1-x} = 0$

$\frac{a}{x} = \frac{b}{1-x}$

$\frac{a+b}{1} = \frac{a}{x}$

$x = \frac{a}{a+b}$

$\int x^a (1-x)^b$

$a \ln x + b(\ln 1-x)$

$\frac{a}{x} - \frac{b}{1-x} = 0$

$a(1-x) - bx = 0$

$a - ax - bx = 0$

$x = \frac{a}{a+b}$

9/30/01

ID

N.B. I think there is a previous ref. to the [illegible] . go ff. It starts at the top (.00) of a page. 385.00 f .09

[Defs (302.27–.30)]

.00: 385.09: Call [M1] the Model of 300.00–302.40 | call [M2] the Model of 378.00–379.40
(379.37–.40): [d.f. O_{n+1} = O_n (d.f. O_n, (Q_{n+1}, A_{n+1}))] | [$a^i_{n+1} = a^i_n \, P^i_O(A_{n+1} | Q_{n+1})$]

~~The Difference is in~~ Both have at any time a D.f. on O_n.

Their responses differ in how they respond to a new Q_{n+1}, A_{n+1}.

.03 → [M2] first (contracts / narrows) the On d.f. by $a^{new}_i = a^{n}_i \cdot P^i_O(A_{n+1} | Q_{n+1})$, then it expands the D.f. using some kind of metric, or GA.

.05 → Also M2 may do the expansion first, then contraction. The only difference is that in the case of .03, the expansion depends on QA_{n+1} (as well as QA_n) in
" " .05 the " " " QA_n only so its not very intelligent. — but the new replys to Q_{n+1} are completely w.o. "Fudge"! — tho actually, This is not

.10 an issue, there is as little "Fudge" in .03

In fact, I [can't see any difference betw .03 & .05] They both alternate contraction & Expansion. Contraction is a "routine", fixd Algm; Expansion is "Creative", "heuristic"

M1 looks at QA_{n+1} & decides how to Modify the O^i d.f. on this basis: "fudge"

— So, while there would seem to be an A.Hness problem, there is not because we use ALP (which is probly ≅ the contraction eqn of (.03) usd by M2.)

.06 M1 first uses the contraction eq. on a new QA_{n+1}. If it gives "adequate" pc for A_{n+1} then the O^i d.f. is not modified ("expanded"). Hvr, if A_{n+1} gets too small

.08 a pc, we expand the O^i d.f. until an adequate pc for A_{n+1} is obtained.

From (.06–.08), it would appear that M1 & M2 are about the same. — Depending

.20 on what M2's expansion method is; how much it differs from M1's "expansion" method.

— But go over 385.00–.09 (the earlier discussn of M1 v.s M2

[10/3/01] On the General System Implementation: It could be by (1) storing many O^i's at all times; or (2) storing a few solns to each "problem type" or (3) storing a few O^i's but also storing states at impt "Branch pts", so "Backtracking" is easy.

.25 [10/6/01] But remember, the form of O^i is an "product" of a larger set of $Ob_j \to Q_j$ subtns/sub pgms.

The meaning of "Branch pts" or "Backtrack" is much difft, from that of a sequencial pgm — (.25) is more like a || pgm!

≅ like Selfridge's "Pandemonium" (1957): for each QA_j pair we have a d.f. of $Q \to Obs$.

The d.f. could be a single Ob Op or a set of them: We may have the product of a Op d.f. is a Op d.f., or a more constrained d.f.

.30

ID

→389.40

.00 **E**xpansion of 389.20 : (≡ outline of Report)

.01 ① Desc QA problem. Give examples, but also discuss "Generality" (≈ Asking human a Q or describing problem to Human)

.02 ② Give genl. soln. for cc = ∞ : $\sum_j (389.13) \cdot P_{O^j}(A_{n+1}|Q_{n+1})$

Perhaps mention updating routine of $z^j_{n+1} = z^j_n \, P_{O^j}(A_{n+1}|Q_{n+1})$.

Discuss: actual predn algn. for .02 Then some of its properties. ~~(without Count~~

(i.e. $\prod_{i=1}^{n} P(A_i) > c$ · some criterion for "true" condl. pd.

Perhaps discuss approxns w. finite no. of O^j's why it usually wouldn't work

③ Give 389.13 w. d.f. over finite set of O^j's as a kind of approxn. —

.10 ← in which we want to Maximize ≈ 389.13.

SN | In most of this stuff, I'm not sure that I will have much of a d.f. over O^j's:

The individual Ob-ops that constitute (O^j d.f.) may have d.f.'s, or could be simpler ≤ op's

Hvr, if I do find > reasonable soln. to a QA, & these solns. look much diffrnt (i.e. they work difrntly for other Q's) then I certainly will want to save them, either a "normal saving" or by int. D.F.. O^j "branch points" for Backtracking.

⇒ not making all of the solns part of O^j but saving the solns as possl. "branch points" for Backtracking.

I will want to decide just how to present this stuff : O_n or d.f. on O_n^j

Probly d.f. on O_n^j mite be best. Hvr, when I do have several poss Op-obs that solve a given problem difrntly (like 350.01–.20 Then I must have a d.f. over O^j

Anyway, do go into 389.26 (≈ 388.00 ... ≈ 389.19) Some details on how "Normal updating" occurs.

Then $\overset{\approx}{O}{}^{df}_{n+1} = \overset{df}{O_n}\,(\overset{df}{O_n^j}, (QA)_{n+1})$ as a final Goal ... Guided by ≈ 389.13.

.20 Then discuss TSQs: How they are usd to ① Train TM② to Train Me.

Guided

$$\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}}\,dx$$

9/8/01 (From .02 : There exists a formal Solution to the conditional probability problem.

An (unnormalized) probability distribution on A_{n+1} is

.25 $$\sum_{j=1}^{\infty} P(O^j) \prod_{i=1}^{n+1} P_{O^j}(A_i|Q_i)$$

.26 Here, O^j is the jth Conditional probability distribution.

$P_{O^j}(A_i|Q_i)$ is the conditional probability induced on A_i, by conditional probability distribution O^j, given, Q_i (Q_i is the "condition")

$P(O^j)$ is the a priori probability or "weight" of O^j. It is approximately

.30 $2^{-\ell(O^j)}$, where $\ell(O^j)$ is the length of a short description of O_j.

The summation is over all possible O^j distributions.

~~While~~ It can be shown that eq. .25 gives a extremely good distribution function for A_i. ~~However eq .. is always incomputable~~ Tho eq. .25 is always incomputable, it is possible to approximate it. The present paper will describe some approximation methods.

One approximation method immediately suggests itself! Suppose that in .25 we do not sum over all O^j, but only over a very large subset of them, chosen a priori — before we see any of the $Q_i A_i$ data. We have not been able to get that particular approximation to work for any practical number of O^j's ~~but the approximation that we have found most useful may be regarded as a variation on that theme~~ → 393.16

An equivalent, perhaps clearer : Form of eq. .25 : $\sum_j z_n^j \cdot P_{O^j}(A_{n+1}|Q_{n+1})$ where we define $z_n^j = P(O^j) \prod_{i=1}^{n} P_{O^j}(A_i|Q_i)$

393.00

0/8/01
ID

.00: 392,40 : It is possible to get good induction by using only one O^j — i.e. one for which

$$P(O^j) \cdot \prod_{i=1}^{n} P_{O^j}(A_i | Q_i)$$

is as large as possible. If O^j is the best we have been able to find, then we can use $P_{O^j}(A_{n+1} | Q_{n+1})$ as a conditional probability distribution for A_{n+1}.

We can get a better conditional probability distribution

Instead of choosing the O^j a priori, we choose for each n, a set of O^j such that for each of them, $a_n^j = P(O^j) \cdot \prod_{i=1}^{n} P_{O^j}(A_i | Q_i)$

.10 is as large as possible. We pick the 100 best O^j's, then use them in a finite sum with modifications to obtain the distribution function of A_{n+1}.

In general, it is usually quite difficult to find O^j distributions such that .10 is very large — particularly if n is at all large.

.16 : 39 2.39 However, we can get a good, useful approximation to .25, by choosing a set of O^j such that

.10 is as large as possible. a_n^j may be regarded as the "weight" of O^j and the resultant distribution function on A_{n+1}, for question Q_{n+1} is

.20 $$\sum a_n^j P_{O^j}(A_{n+1} | Q_{n+1})$$

.21 For each n, we want to find a set of O^j with maximum weights.

A very practical difficulty is that if n is more than 3 or 4, the problem of finding a good set of O^j's is normally extremely time consuming.

What would be ideal is an O^j set of the proper form,

Exclude do not cross out! We will discuss a method for finding the O^j set for large n by a process of "updating" Say we have found a good set of O^j and associated a_n^j for the data set $\{Q_i, A_i\}$ $i = 1 \cdots n$, then contribute on the O^j being of a certain form, it is possible to obtain the O^j set for n, a Call this set of O^j and associated weights S_n.

.30 Then given Q_{n+1}, A_{n+1}, we have an algorithm that will use Q_{n+1}, A_{n+1} and S_n to produce S_{n+1}. Using this algorithm we can find S_1 by direct search of Eq. .25, then the algorithm can be used to find S_2, then S_3 ... and finally, S_n.

0/9/01

Change notation: $a_0^j \leftarrow P(O^j)$; $P_j(A_i | Q_i) \leftarrow P_{O^j}(A_i | Q_i)$

0/9/01 ID

First draft to Q A problem (See 392.01

.00 There is a formal solution to the conditional probability problem.

.01 $$\sum_{\text{[illegible]}} a_0^j \prod_{i=1}^{n+1} P_j(A_i|Q_i)$$

define a_0, $P_j(A_i|Q_i)$ (392.26-30

" The sum is over all finitely describable conditional probability distributions, [illegible]. $P_j(\cdot|\cdot)$.

If all of the Q_i's ($i=1 \ldots n+1$) are known, but the A_i's are unknown, then .01 gives the universal joint probability distribution for the A_i.

If all of the Q_i's are known, and A_i; $i=1 \ldots n$ are known but A_{n+1} is not known, then .01 gives a universal probability distribution for A_{n+1}.

An equivalent form to .01 is

.10 $$\sum a_n^j P_j(A_{n+1}|Q_{n+1})$$

.11

.12 we define $a_n^j = a_0^j \prod_{i=1}^{n} P_j(A_i|Q_i)$

and the sum is as in .01 over all finite describable $P_j(\cdot|\cdot)$.

.13 expresses the conditional probability of A_{n+1} as a weighted mean of the conditional probabilities induced by all P_j's. — The weight of P_j being proportional to (its a priori probability multiplied by its probability assigned to the known data, (A_i, Q_i) $i=1 \ldots n$.

This solution to the induction problem is known to be very accurate, perhaps as accurate as any inductive system could possibly be. It is, however, also known to be incomputable in finite time with finite memory.

.20 Most legitimate attempts to do induction can be usefully regarded as methods of approximating this ideal solution. From this point of view, we can compare and criticize them.

~~physically realizable induction methods~~.

The present paper is a rather direct approximation of .11. Instead of summing over all P_j's, we only sum over a relatively small number of P_j that have large weights, a_n^j.

Discovering P_j's that have large weights, is the main problem. We look for P_j's such that the value of a_n^j in .12 is as large as possible.

~~Usually we will have a large set of P_j.~~

For each value of n, we will usually have a large number of P_j's with

.30 large weights. As n goes from n to n+1, the list of P_j's of large weights will usually change completely. However, usually we find it possible to express the high weight P_j's for n+1 as modifications of those for n. examples having high weights →315.01

.33 If P_j has large weight for (A_i, Q_i) $i=1 \ldots n$, usually we are able to find a new P_j' that is a modification of P_j such that

Then we will try to find a new P_j' that is a modification of P_j such that

i.e. $P_j'(A_i|Q_i) = P_j(A_i|Q_i)$ for $i=1 \ldots n$, but $P_j'(A_{n+1}|Q_{n+1}) \gg P_j(A_{n+1}|Q_{n+1})$

.36 P_j' and P_j are the same for (A_i, Q_i) if $i \leq n$, but P_j' is much better than P_j is, for (A_{n+1}, Q_{n+1}).

10/01
ID

Somewhere in here, a sentence comparing a_0^j — (type of P_j) (Latterly, the functional form of P_j is peculiar, & the AZ-141 method described is not applicable in an obvious way. The sub-functions are Unordered.

~~The process of creating a good set of P_j [illegible]~~

Given a "good" (high weight) set of P_j for n examples, the process of creating a new set of good P'_j for the data set having an additional new example, (A_{n+1}, Q_{n+1}) is called "updating".

While it is possible to do good induction using only the P_j of highest weight found, the process of "updating" is easier to do if we have a large number of good P_j's for each set of examples.

~~The problem of~~ This "updating" is usually the most difficult aspect of an induction system, and the method used for updating ~~and an important way a very good way to characterize induction systems~~ is to implement it is a good way to characterize an induction system.

Needs revising (more)

Section: Updating the P_j distribution.

There is an obvious way to do updating: For from the set of P_j of high weight for the n & the n member data set, we can construct new trial P'_j by a process of mutation and/or crossover, ~~The~~ as is done in Genetic Programming. The "fitness function is, of course the a_{n+1}^j of .12. While this technique would probably work, the fitness function is very expensive in time, since each P'_j has to be tested on all (A_i, Q_i) for $i = 1 \ldots n+1$.

Taking a clue from the way humans seem to solve problems, we will limit the P_j sets considerably by making the following constraint:

(Insert 394.33 -> 36)

We can do this by ~~constructing each P_j so that it is a combination of~~ making each P_j a combination of n functions: one for each A_i, Q_i example. Each of these functions consists of a recognition part and an operator part. The first part, R_i recognizes how Q_i differs from other Q's, and the second part, O_i operates on the Q_i to obtain a probability distribution over possible A_i.

Often several Q's will be recognized as equivalent, since they all use the same stochastic function to obtain the associated A distribution.

For a particular $Q_i A_i$ it will often be possible to find more than one R_i, O_i pair ~~that gives high probability to A_i~~ that gives a larger a_n^j value in eq. .12. These R_i, O_i pairs will have high a priori probabilities (i.e. short descriptions) and assign large probabilities to A_i.

One way to find such R_i, O_i functions might be via Genetic Programming methods of mutation and crossover of R and O functions that have been previously found useful. At first glance, each R candidate would have to be tested on all previous Q_i's. While this is indeed true, it will be found that certain R functions are able to reject large subsets of the Q's.

R_n^k $Q_n^{k \text{ w.r.t.}}$

finding R_1^k may not be easy: it has to be checked on all previous Q_i's.

10/6/01

0/11/01
ID

.00 ~~[struck]~~ A small number of functions of this sort can be used by many different Q_i's to reject Q_ℓ's not identical to themselves. It will usually be necessary for a new R_i to be tested on ~~[struck]~~ only a small number of Q_ℓ's for which the standard rejection ~~[struck]~~ functions are inappropriate.

.04 [SN] In 394.01 if we sum over ∞ P_j's we get "v.g." induction (By ~~[struck]~~ M square criterion). If we sum over only some of the P_j's Ihypart. Σ is, T. closer we get to 394.11 (w. Σ over ∞), so t. smaller ~~[struck]~~ Σ err² we get!

Σ err² ~~[struck]~~ is bnd by something like $\ln\left(\frac{\sum_{j=1}^{\infty} a^j_{n+1}}{\sum a^j_{n+1}}\right)$ (j = ~~[struck]~~ fruits with a^j_{n+1}) or perhaps $\frac{\ln \prod_{i=1}^{n} P_{true}(A_i|Q_i)}{\ln \sum a^j_{n+1}}$ (over my a^j_{n+1}'s that I've found thus far.)

.10 Perhaps include this discn. in t. report!
I think it is "implied" in my discussion in t. Bare paper — but not as clearly as it might be! (.17)

.12 [SN] In report: Include a Para giving t. heuristic Basis of TM; T. idea that we have ~~[struck]~~ a TSQ — we have ideas on how Human might solve t. TSQ: we have a measure of adequacy of that "Explainn" & : if it is any good, we know how to express it exactly using ALP — if not, ~~[struck]~~ ALP & Max ideas ~~on how to~~ ~~[struck]~~ on just where t. model is deficient, & this often suggests how to repair it. } Perhaps also explain "Witten factor & a of optimum"

.17; (.12) [SN] (or before) (After) eq. 394.01,.11 give Sol 78T3 & its properties. Discuss correspondence betw. them.
"For the model of ~~[struck]~~ 394.11 there is an analogous theorem that guarantees the rapid convergence of .11 to to correct answers." ~~#~~ Perhaps mention more details on how any P(A|Q) induces a pd on each of t. bits of every A_i & just both ~~[struck]~~ 394.11 & t. "source" P(A|Q) do this: So we can ask how ~~they~~ t. bit probys ~~deviate~~: betw. these 2 models; t. answer is "like" "S78T3". See (.04)

= ||
— ||

A B.bl. review of stuff on Ob, Op ideas to implement 388.00ff;

1) 388.00 — 383.19; 390.10 — .40 (A special problem); 386.18; 387.11 — .40 : 384.06 — .20 + ~~[struck]~~ (384.01 — .15)R ; 261.08 — 18

.26 One thing about ob's: very often they are heirarchical, so a few are used to "narrow down" what t. problem is, & then ~~[struck]~~ further obs are ~~[struck]~~ selected to determine more exactly what t. problem is (& .: its soln).

.30 388.06 — .09 Here "Context" is defined by an Ob: It looks like t. most General defn. of "context". It's an operational defn. of "context". Hvr. Discovering t. correlates of a "Context" is dfclt: we have to try it on all previous data to see if it correlates w. any thing Useful. It seems unlikely that we would Construct "context" obs ~~at~~ at random & test them for "usefulness" — T. test is usually too expensive. 388.10 suggests some useful contexts.

(SN) On overall format paper: In ~~discussion~~ introduction; Desb. problem: Desb. ~~[struck]~~ IA's: Mention that GA, ANN, ALP(~~[struck]~~ Complexity based Induction), or SV machines ... can be used to create IA's that will solve t.'s problem. "We will mainly discuss use of CBI"
Also mention that while Max $a_0 \prod_{i=1}^{n} P^j(A_i|Q_i)$ is v.g. thing to ~~optimize~~ Max w. limited time, it is not t. best way. T. "Best way" seems related to Sub-cogni & recognition Algms (= obs)

8/12/01
ID

.00 : 396.40 : On Context! Context can be ~~det~~ (deterministic, sharp) or stoch (probabilistic, stochastic) ("fuzzy"). Contexts are used in (at least) 2 ways: • As Recognizers (Obs) to condition ~~[struck]~~ (D/S) Ops for Predns

• As contexts for obtaining pc's of ~~concs~~ sub concs, used for Conc. Constructions.

& pc evaln. of these Concs.

My, ~~[struck]~~ (Pre-latest) BIG expansion of t. idea of Context was 300.26ff. (300.26 – .33 in particular)

This should be tied up w. cross refs to t. most recent idea of context 388.06 (context = Ob)

300.26 – .33 considers "Context" to be t. ~~[struck]~~ hyer level problem of which t. present problem is a part — as well as th. hyerlevel problem above that, and so on . . .

The Genzn. of "Context" of 388.06 was that a Context (operationally) could be any Ob; i.e. a funct from Strings to Boolian (True false).

Notation Note: P_j^n is jth p.d. at ~~[struck]~~ for data set $[A_i, Q_i]$ $i=1 \ldots n$

$[P_j^n]^{fixed\ n} \equiv S_n$

$S_{n+1} = S_n(S_n, Q_{n+1}, A_{n+1})$ symbolic expressn.

.10 — ~~They~~ They would seem to be two quite difrnt kinds of generalizations!

.12 — So, General outline again:

Dcrn of QA problem explain of why its a "complete"

Defn, Dcrn ~~[struck]~~ of IA's:

~~[struck]~~ workable IA's can be based on GA, ANN, SVM's, C.B.I. — We will discuss all ~~[struck]~~. approach to a soln. via CBI.

Give Genl soln. for $cc = \infty$; tell how Good it is: In paper give details, perhaps Dcrn, perhaps outline of proof.

How can we approach this incomputable soln?

1) Start w. large no. of P_j's. Then $a_{n+1}^j = a_n^j \, P^j(A_{n+1} | Q_{n+1})$

~~Discuss~~ Say we keep all P_j w. wt $> \frac{wt.max}{W}$. (say $W = 1000$).

2) n ↑ cardinality of $[P_j^n] = S^n$ will ↓, when card = 1, no more ~~[struck]~~ lrng can occur

.20

2) For each n, select ~~[struck]~~ w (w can = 1) P_j^n of hyest ~~[struck]~~ a_n^j.

.22 For each n, normally Ξ $[P_{n_j}^i]$ & Σ $[P_{n+1}^j]$ will ~~[struck]~~ have few P^i in common.

If n is large finding these w P_j^n will be very time consuming & time will ↑ w. n.

3) Similar to 2) but given $[P_n^j]$ ~~&~~ mutate, recombine (crossover) until we have new $[P_{n+1}^j]$ of each w, ~~[struck]~~ for top W that are within factor of w' of peak wt. W can be as small as 1 & w' can be 1, but then mut. & recomb. doesn't work well.

We will mainly discuss variations of 3.

For most forms of ~~[struck]~~ $[P_j^n]$, ~~[struck]~~ when we ~~[struck]~~ expand the set of functions by mutation & crossover,

.30 we have to test each candidate function on the entire data set $(QA)_{i=1 \ldots n}$ — As n grows, this

.31 process becomes more and more time consuming. The time to learn a set of n (examples / problems) would be ~~[struck]~~ O(n^2).

.32 (i.e. $\sum_{i=1}^{n} a i = \frac{a}{2} n(n+1)$).

.33 One way to overcome this difficulty is to express P_n^j as a combination of n operators:

Each operator consists of ~~[struck]~~ 2 parts: first a "Solving" part S_n^j that maps Q_n into a probability distribution on A_{n+1} and second a recognition part R_n^j that maps Q_n into a Boolean True or false. If True, Q_n is transformed by S_n^j into the probability distribution on A_{n+1}. Otherwise S_n^j does nothing.

The other R_n^j could all ~~be applied to~~ operate on Q_n to determine which S_n^j should be applied to A_n.

398.00

10/13/01
ID

$\int \ln x$: $d\,x\ln x = \ln x + x$

$\Rightarrow x\ln x = \int \ln x + x$

11:47

.00: 397.40 Would this not take time proportional to n, thus defeating our effort to save time?

· Normally, it is possible to categorize the operators R^j_n, so that by using only a categorization operators on Q_n, one can rapidly eliminate almost all R^j_n's.

.03' Rewrite @ 397.33 ff: One way to deal with this difficulty is to first find probabilistic one or more) functions, F^j_{n+1} such that $P(F^j_{n+1}) \cdot F^j_{n+1}(A_{n+1}|Q_{n+1})$ is as large as possible — $P(F^j_{n+1})$ being the a priori probability of F^j_{n+1} and approximately equal to $2^{-l(F^j_{n+1})}$: $l(F^j_{n+1})$ being the length of the shortest program for F^j_{n+1} that we have been able to find. (Explain how such funcs are found)

.08 Having found such a set of functions, we must find a way to recognize problems like Q_{n+1}, so we will know that F^j_{n+1} can be used to solve them.

.10

.11 At first we will try to find a function that narrowly recognizes Q_{n+1}. When we have recognizing functions for each of the problems in the set, we will try to simplify them, so that their total description length is reduced, and so it takes less time to use them to determine what F to use to solve problems. At first, this is done by creating a hierarchy of recognition functions, such that at the lowest level, the recognition function will decide what recognition functions need be used on the next level, and so on. At the highest level the recognition function decides which F function to apply to Q_{n+1}.

.20 Using a device of this sort, the time needed to recognize a particular Q_n may be of the order of $\log n$, and the time needed to recognized all F functions needed for n problems will be of the order of $n \log n$.

.22 In general, there will be other means of consolidating recognition functions so as to reduce the total amount of time used in implementing them.

SN Explain tradeoff betw. CC & PC in (.08R) Also in finding Good "Recogn Obs" in .11-.22

That this is not an "adjustable Param", in the sense that one inserts it on the basis of past experience a/o "intuitive understanding". It is a **User Input**, which is part of the problem dcrn. Discuss how a "user input" could be devd by a "Reinforcement Machine" — but explain why this is **Not Desirable**.....

10/14/01 Th. main Bottleneck now, is how to dcrb. "Updating" : As of now, in most "el." form; It is in 2 parts: F^j_n & R^j_n. So I can first dcrb the "first order" F & R functions. Next, discuss consolidation of both Types (i.e. F's consolidation; R's consolidation. R's.)

.30 Consolidn. of R's means: say

$R_n(Q_n) = T$; $R_{n+1}(Q_{n+1}) = T$; Consolidation: $R'(Q_n) = 1$, $R'(Q_{n+1}) = 2$,

Furthermore R' has shorter dcrn than dcrn length of Q_n + dcrn length of Q_{n+1} a/o

$R_n(Q_n)$ $T(R_n(Q_n)) + T(R_{n+1}(Q_{n+1})) + T(R_n(Q_{n+1})) + T(R_{n+1}(Q_n)) > T(R'(Q_n)) + T(R'(Q_{n+1}))$

The dcrn length of R' < the total dcrn lengths of R_n & R_{n+1}.

"Consolidation" of R's & F functions like Newton's integration of Galileo & other results.

399.00

0/15/0
ID

Convergence thm for Condl Probys : (33)

.00 : 398.40 : We can start w a Maximally el. S function that is composed of a sets of R's F functs. We end up w a single S funct. that accepts any kind of Q, & gives some kind of A — (the best ever A if larger CB is allowed.)

Essentially, what we want to do is ① Compress S, ② have it work faster: also, it should have some results for past corpus. Compression implies better induction via analog of S78T3.

.05 SO: T. First Ring TM does when a new QA comes in: ① use old S on Q: see if U.G. A appears (= by pc). ② If Not (i.e. Q may have no output, or just a A of low P''), it tries to find F(Q) → by A pc. More exactly $F \Rightarrow P_F(A|Q)$ is largest ave possible.

Notation Note: Perhaps look for "Smaller" functions (= shorter descr. length) rather than "shorter" funcs. "Shorter" could mean shorter exact sentence (= keystrokes) shorter descr. length

.10 ③ Next, it tries to find way to Recognize ≈ new Q.

.11

.12 Probably I shouldn't go into more much more detail on "updating" until I start work on TSQ.

Tho I might do ANL, using the "new model of updates"

SO: in t. report: Do, or — all: Tell how to get t. set of R_n^i, F_n^i (possibly include 400.00 — .19 doc. 10—.19 is summary)

Then discuss "compression" of R_n^i in small sets; of F_n^i also in small sets.

If a set of i's over (unified, coalesced, integrated) then these correspond. R_u^i are also automatically integrated — Also t. Recog'n algms then need to do "less work" (less cc).

.20 But note .12 (into report).

One expository Approach: Just tell how to get F_1^i, then for as large a set of probs as poss!.

If (When) it looks dif'ly, Do $\{F_2^{i'}\}$ for a new set of probs & find $[R_2^j]$ for these probs:

Then $\{F_3^{i'}\}$ for a $\{R_2 z^{j'}\}$ for t next set. Then discuss the idea of 397.31 & skip some cases. (i.e. $cc \propto n^2$ $= O(n^2)$)

Then show "consolidation" of R's (& of F's) reduces time to $O(N \log N)$ or less. $\int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma^2}} dx$

(See 398.03—.22)

ABCDE FGH
ijklmnopqrst

.28 Say we have a new problem; several of R's might easily recognize it, but their R's fail. After we solve t. new problem, we have to modify t. R's that easily recognized it.

.30 This might be done by having a new R (for t. new problem) that "dominates" all other (older) R's.

In Gen I've written about various kinds of overlapping R's & various kinds of

.32 Review & try to consolidate that work: list various kinds of cases (situations of overlap, etc) → 398.00

.33 0/16/01 SN On t. Convergence Thm. for Conditional Probabilities: In t. proof of this thm, I considered $P_F(A, Q)$ v.s. $P'(A, Q)$. P being True condl proby & P' being any approx to it — or already any condl. proby distrib. In t. proof I that I had to assume $P'(A, Q) \geq P(A, Q)$. This is not so.

The assumption had to do with Expected Values of proby ratios ($E_{wrt\ P}(\ )$).

(Try to find previous Reference). My that it's still not clear on this! 4 25:00 neg: what is probly t. rite idea! We were interested in $\min_P E\left(\frac{P'(x)}{P(x)}\right)$: which is t. Kull—Leib. distance & is always < 0 → (spec) 425.00

→ 400.00

.00 : 40 1.40 : SN This finding of R_n^j is usually incremental, but it need not be "easy"! The process of incrementation

But Anyway! Breaking the problem into a recognizing part & a soln part does seem like a good, useful idea.

Mention use of ~~beta~~-type GP as an alternative to search (Give Pros & Cons)

Ordering functions in PC order! In Lisp (& other functional langs), the string-sort, tells when a function is completed, but it may only be a "subfunction". [Look at my ~~dissertation~~ discussion of this in that stuff I wrote in Latex "report"] IMPT!

Also ~~—~~ that stuff I wrote about using the Warm code to sort strings in PC order — is it Applicable in this case?

More generally, suppose we have been given n examples, $[Q_i, A_i]$, $i = 1 \ldots n$.

.10 and we have ~~the~~ a recognition function R_n and a set of associated conditional probability functions $F_i^j(\cdot|\cdot)$ for $i = 1 \ldots m$. Usually $m < n$, since we don't always need a new set of F_i^j's for each new Q, A. F_i^j is the jth function that responds to R_n's ith output.

We are given a new problem, Q_{n+1}.

If Q_{n+1} is recognized by R_n, $R_n(Q_{n+1})$ will be some integer from 1 through m

SN Could R_n output more than one integer? — At first arrange so that it's only one integer. Later we may improve it by allowing > 1 integers & giving wts to the F_i^j sets.

.17 : To define the system more exactly, suppose we have been given a set of n examples, $[Q_i, A_i]$, $i = 1 \ldots n$.

The system has a recognition function, R_n, so that for $i = 1 \ldots n$, $R_n(Q_i)$ gives a single integer from 1 through m. The system also has m sets of $[F_i^j]$ functions ($i = 1 \ldots m$).

.20 If $R_n(Q_i) = k$, then the set $[F_k^j]$ of probability distributions will be used to solve the problem, Q_i.

If $1 \le k \le n$ then $1 \le k \le m$ — i.e., for problems that the system has worked, Rcarmill's loops for a set of $[F_i^j]$ functions that will give a probability distribution for A_i.

If Q is a new question however, say Q_{n+1}, then if R_n recognizes Q_{n+1} it will output an integer k, $1 \le k \le n$, & the resultant probability distribution for A_{n+1} will be

$$\sum_j b_k^j F_k^j(Q_{n+1}, A_{n+1}) \Big/ \sum_j b_k^j$$

$$b_k^j = \prod_i F_k^j(Q_i, A_i) : \quad i \mid R_n(Q_i) = k$$

.30 c_k^j is the a priori probability of the function $F_k^j(\cdot|\cdot)$.

$\sum_j b_k^j$ is a normalization constant

In the denominator of b_k^j, the product is over all i such that $R_n(Q_i) = k$.

On the other hand, if R_n does not recognize Q_{n+1}, $R_n(Q_{n+1})$ will have no output. In early stages of training of the system, we will stop at this point, and say that it cannot answer that question. In later stages of training, the system can try to modify R_n (usually by compressing it), so that it returns its responses to Q_i ($i = 1 \ldots m$)

Discuss the structure of R_n — as a set of if-then-else statements.

10/18/01

ID

.00: 402.40! and also responds to Q_{n+1}

Analysis of the updating behavior of the system, is more complex.

As before, say we have (Q_i, A_i), $i=1\ldots n$ and we have R_n and assets $[F_i^j]$'s that are able to get a good value of A_{n+1}

After the system has responded to Q_{n+1}, we (can) update it by giving it the correct string.

If the system gives an adequate probability to A_{n+1} conditioned on Q_{n+1}, we can do no updating, and go on to the next problem.

If the system either gives no response to Q_{n+1} or gives too small a probability to A_{n+1}, we first find a heavy set of F_{k+1}^j's such that

.10 $\sum_j c_{k+1}^j F_{k+1}^j(A_{n+1}|Q_{n+1})$ / $(\leq c_{max}^j)$ is as large as possible.

We then modify R_n to create R_{n+1} so that $R_{n+1}(Q_{n+1}) = k+1$ but for $i \leq k$ $R_{n+1}(Q_i) = R_n(Q_i)$

i.e. R_{n+1} is the same as R_n except for its response to Q_{n+1}.

(Discuss R_n as composed of k modules: How some must be changed if they give poor responses.

Discuss $[F_{n+1}^j]$ discovery: Discuss R_{n+1} modifn } (a) via Lsrch (b) via GP. (c) other ways ANN, SVM?

For my own use, try to go into more detail on these 2 & also on the more general problem

.18 of updating, in which we want to get $O_{n+1} = O_n(O_n^{df}, Q_{n+1}, A_{n+1})$, using the usual Gore.

Notation: Instead of $F_i^{k(\cdot)}$, perhaps use $p_i^k(\cdot|\cdot)$. (This was used as far back as 397.22 !) ← No! The data of P_n^j was not what I'm using now.

.20 i.e. the way(s) that the new problem differs from the old, could give some clues on how to solve it.

.18 is a/ general updating alg (A more general one would have more agents than O_n .. e.g. O_{n-1}, O_{n-1}, recent Q's, recent successes of probsolving, etc ... (see ≈ 300.26 ff for some possys.)

$[F_i^j]_{and}^{R_n}$ is one way to structure O_n, but not the (most general / best) way.

So we start out w. a certain form for O_n as TM matures, it is able to do more general forms.

0/20/01 → For Myself, write a review (or just read all past) on "updating" — { Is REFranco Piaget PP allen "updating"? }

NB M2 378.00~.06; 379.21~.24 aren't nearly so bad! I was concerned w. the trials taking too long because they would have to do all $(QA)_i$ for $i=1/n$: Could M2 use the ideas of M1 (i.e. Division into Ref Functions } to cognition ↓ verifn. time? Cands that took too long would be discarded, so we would end up mainly w. cands

.30 that used M1-type models. A trouble is, M1 doesn't actually test $(QA)_i$ $i=1/n$.

It uses a logical argument to show they are 0, ½, 1

.32 SN In report do say that Humans do "recognize" a new problem, as/ normally retain all old ways of working old problems. What we want is a method that usually does this, but occasionally is able (needs to) make a transition (backtracking) that can apply to old as well as new problems.

That the "recognition" problem can be further "Broken Down" (≡ alzd) into a recognition Operator for each problem type: we can have higher level recognizers for categories of problems & by level still to cats of cats, ect. This hyper hierarchy speeds up & simplifies (lower $ cost)

.38 the solns.

Do say .32 ~.38 in report — soon after the "updating idea" is mentioned

404.00

0/20/01

ID

= $P_S(A|Q)$

.00 : On t. 3 input form of a universal P.D. : R is equiv. to a 2 input form of a condl. P.d.

i.e. [redacted] t. input is condition, other is "R". So R is a desn of t. output "instead of" Q.

For a 3 input setup, R is a desn of Q in view of Q & t. general system desn, S.

For a system desn S, Q's $Q_1 \cdots Q_{n+1}$; A's $A_1 \cdots A_n$ t. desn of A_{n+1} is

---

.04 One way to write report: ① Desn of Problem, its universality → 410.13

← this was writ here by Mistake!

② Soln: Maybe as I've been doing recently: on little detail,

③ Desn of O_n as sum of kn P_2^i's & R^i's.

~~④~~ ④ Updating: Allow done:

.10 ~~⑤ best~~ Appendices: ② How pc's assigned to Concepts

.18 ① how L srch is done for Induction, for Inv probs.

.20

.30

ID

.00: **407.40 :** The forgoing description of updating techniques is for the early training of the system. The only method used to solve problems was Lsearch. The system was able to learn to solve complex problems, only if the solutions were obtainable by combining functions that were found useful in solving earlier problems.

} This idea of ~~[illegible]~~ + SQ's should be mentioned earlier. Mention Sol 86's 89, but describe the build'g of concs.

In explaining "what t. QA problem is": Say that /general optzn. problems — optzn w. limited time a/o other resources is a poss. kind of problem. Lsrch can do this, but is not necessarily t. most efficient way to do it (Woops! But it is "If all info is in P.D. & we have suitable lrng. during Lsrch.)

.10 Perhaps follow line (.00) with: The system itself is broken down into a problem recognition part and a problem solution part. While this is often a reasonable way to solve problems, it is not the most general way. After the system has had enough ~~successful~~ experience in solving optimization problems, we can give it as an ordinary ~~[illegible]~~ "question" the problem of finding a set of functions,

.15 O_{n+1} so that $a_0^{n+1}\prod_{i=1}^{n+1} O_{n+1}(A_i|Q_i)$ is as large as possible — of system functions ~~[illegible]~~

~~its "side information"~~ ~~[illegible]~~ the set of $[O_n^j]$, that worked well for the first n problems

.17 in the sequence, is given as "Auxiliary information".

(.17R) The concept of "Aux info" must be explained: e.g. An encyc. or dictionary can be regarded as "aux info".

Stated another way,

~~$[O^*_{n+1}] = [O_n]([O_n'], \odot$~~

.20 $$[O_n](O_{n+1} \mid [O_n'], Q_{n+1}, A_{n+1})$$

(.21R) This concept of "acceptable ways" to use Aux info: Rather complex; I'm not sure I have a good understanding of it. See 408.32ff for discn 409. ~~ff~~ may be adequate soln.

.21 give us a distribution on O_{n+1} as a function of

$[O_n']$ the set of descriptions of O_n's of much weight, and

the new problem solution pair $Q_{n+1} A_{n+1}$, which are to be used to update $\{O_n\}$

$[O_n]$ is the set of system functions of much weights and $[O_n']$ is the set of their descriptions ~~description of these functions~~

.27

.28 From .20 we obtain a distribution on O_{n+1}, ~~which we can use as the basis of a set of trials~~ We use .15 to determine which, if any, of these O_{n+1} are good "system functions"

.30 Q: So How does .27-.28 differ from Lsrch Guided by $\{O_n\}$? — it doesn't but $[O_n]$ is supposed to give a v.g. P.d, so Lsrch is very fast.

.32 0/23/01 (SN) on "Aux info" I haven't really solved this problem for finite CB. (I'm thinking of t. pure /compression/ induction problem — not OZ probs.). For induction problems w CB = ∞, One simply maximally compresses Main info + aux info. For CB < ∞, One only compresses enuf of t. total corpus to give good prediction — extrapoln. of a particular part of t. total corpus.

I haven't really solved that problem: One soln. of it is to break corpus into 2 parts & compress t. part most relevant to t. extrapoln. problem. I've written a lot on this since I → 409.00

10/23/01

ID.

Ballroom
Beethoven

$10 \times 10^9 \times 3 \times 10^7 = 3 \times 10^{17}$ seconds
$\times 109 \sim 3 \times 10^{26}$ ns

.00 : 408.40: first discovered the problem (at IDSIA). I may have called it "the partitioning problem".

Thinking about it now, it would seem that one would want to compress parts of the corpus to varying amounts. This would give a set of concepts that were very useful in extrapolating the specific area of interest. That "set of concepts" that seems most important.

Being able to "read an encyc." means that one can compress it very rapidly (low cc).

But the idea of "indexing" an encyc. so one would not have to read so much of it ---- How does that fit in?

→ I think the (~) "soln" I obtained was that one would discover certain methods for doing induction. These methods might involve "how to choose a subcorpus". One learns how to do good induction.

This "soln" puts the problem at a hyer level, does not eliminate it. But one might have more

.10 ideas on how to deal w. the "hyer level problem" — it may be in a more stylized form than diverse forms of the lower level problems. Still it is Juergen's "Learning How to Learn".

The "Hyer level" could be Org. Evoln!

.14 Actually, the "second level" can use the older form of "Resource limited ALP": i.e. "the best codes for the entire corpus obtainable in Rinf given CB" RLP This may not be strictly best, but it's good enough!

We can have much interchange of info between first & second level! Each second level problem can be a regular problem for the first level.

Earlier, I thot of this "second level" in another sense: i.e. R.W. can be this "BIG" corpus, but to make predns, we select out observations/experiments. [In physics theorists are dirty, hvr:

.20 In "My Enery" experiments, say, the predictions we make are mainly for the experiments & for future Experiments. — rather than for answering present Questions.]

Re: the "Aux Information" problem of 408.17 & .32, 408(.17R-21R)

However, one can "learn from experience" (level 2 learning) that under certain conds, one has good chances of getting results inconsi w. previously accepted predn. methods ("theories").

A mode of living that I've not studied, is that of studying works, theories, experiments of others — Reading history of science — Just trying to understand how/why certain theories a/o experiments arose, could be a useful source of "Understanding" for TM. I assumed TM would read lots of Good Stuff, but I hadn't considered how it would make use of it

---

So: up to 407.40, we have a "first order approxn." to a TM.

.30 407.15-.40 is an improving the recogn. funct, so it doesn't take or know operations to do the EL problem. While taking that long would be very bad, it would be tolerable. — Say, if I wanted to demonstrate a minimal TM that could do work interesting problems.

(407.03-.14: this deals all updating operations: I guess I need all of them: Probly even for Minimal TM.

Improvements beyond Minimal TM:

.35 (1) Lrng during L-srch (this may be needed in even Minimal TM) → (More detail on how this is done) → 410.36

(2) Improvements in Recognition Function (of 407.15-.40

(3) Perhaps some trims to the simple first order updating operations

(4) Use of context to assign pc's to Concs (may be needed in Min TM). I should write this up more clearly for my own use. This could result in the final equ. 408.20 (see 300.26ff)

Conditional probs based on context = "conditional probs" = what QA TM normally does! → (spec 410.32)

(Details on this plus it to be done) 410.00

1.377 pp/d.

space 399.32

.00: 399.40 → Various Responses of "set of R's" to a new QA:

① (Trivial) Q is recognized, resultant A has Adequate PC: → Go on to next problem or do "SI".

.02 ② (Normal problem) Q is not recognized: We find some F → F(A,Q) is true: Shall find a R to recognize just Q.

③ Several R's recognize Y. Q: 399.00 – .40 deal w. this situation. If resultant predn gives adequate pc to A, ⊕ Then O.K., stop; if not, go to ②; ~~Perhaps try to find R that~~ (this is essentially ① (≡ 399.28-.30))

.055 In addition to finding Good R for F we've found. Try to Modify the R's that had F's that didn't work well. Modify R's a/o F's. (see 399.28-.30)

.07 **Summary of .00 ff**

- P(A|Q) is adequate — Go to next QA.
- P(A|Q) is not adequate — find new F set to solve this Q.A.; — Modify Rn so that Rn(Qn+1) = K+1 only.

.10 for updating

- ⓐ Q is recognized by 1 R (.00) → Predns are O.K. see 399.00-.40 for assignment of wts
- ⓑ Q " " " >1 R → Predns are not O.K. Go to (.055 – .07)
- ⓒ Q is not recognized by any R. (≡ .02)

For predn: ⓐ If Q is recognized by 1 R use predn of F.

ⓑ " " " >1 R use wts of F's (when available) ∈ (via wts 399.00-.40 if available)

ⓒ " " not recognized by any R. Either no predn, or try to modify (compress) ⊕ some R's so they include present Q. { Alternatively, All R's give same wt to all Q's which is a ~~somewhat~~ Modal of QATM that I've not investigated } (Is Mindfull of Selfridge's "pandemonium"

Another, more conventional way to deal w. ⓒ is to compress ⊕ various R's that seem to be close to Y. Q — since by compressn, we ↑ generality (gain more?).

.19 abcdefghi decrb. Rs.

.20 Good Offline: 397.12 – .32 (.33 ff is one way to say it. 398.03 [10,22] suppose another (better?) way to ~~test for this~~.

O.K. so: Start w. "CP1" Paper: Use first 9 lines (as corrected)

~~Start~~ Here will first give a [very Good] theoretical solution to the problem, involving infinite time and ~~memory~~ storage.

~~We will then~~ We will then discuss various ~~ways~~ physically realizable approximations.

For data set described above, the probability distribution on Pn+1 is

.26 $$\sum_{j=1}^{\infty} a_0^j \prod_{i=1}^{n+1} P^j(A_i|Q_i).$$ (use O_{no}^j instead of P^j.)

Here $P^j(\cdot|\cdot)$ is the jth possible conditional probability distribution relating it's two arguments. — $P^j(A_i|Q_i)$ is the probability of A_i, given Q_i.

Since ~~the~~ total recursive distributions of this sort are not effectively enumerable, we include in our sum, all partial recursive functions of this sort — which are effectively enumerable.

.30 a_0^j is the a priori probability of the function $P^j(\cdot|\cdot)$. It is approximately $2^{-l(P^j)}$, where $l(P^j)$ is the length in bits of the description of P^j. Appendix 1 discusses computation of the a_0^j (this is not language "SAZ").

We can rewrite .26 in the form

.33 $$\sum_{j=1}^{\infty} a_n^j P^j(A_{n+1}|Q_{n+1})$$

Here $a_n^j = a_0^j \prod_{i=1}^{n} P^j(A_i|Q_i)$.

In .33 the distribution of An+1 is a weighted mean of all of the P^j distributions — the weight of each P^j being the product of its a priori probability and the probability of the observed data, in view of P^j.

The distribution .2 6 (or .33) is known to be very accurate. If we use the

.00 z_0^j to express all of our a priori information about the data, this distribution is perhaps the most accurate that could be obtained by any means.

Since we cannot compute this particular infinite sum using finite resources, we try to approximate it by using a finite number of large terms — terms that in 400.33 have much weight. I.e. large z_0^j values.

While it would be ideal to include the terms of maximum weight, it is unknown that it is impossible to know if a particular term is of maximum weight. The best we can do is find a set of terms so that their total weight is the maximum we could find in the available time.

.10 Finding seems to be that of finding the "heaviest" heaviest functions possible in whatever time we happened to have.

Using whatever limited computation resources we have, within our limited time and memory limits that we have, the more time and memory we have, the better are our probability estimates.

In general, better probability distributions as our time and memory our probability estimates will if we search for a longer time.

.15 resources are expanded.

abcdefg

SN One way to phrase the "problem of induction" is to find a set of functions, F^j within given resource constraints, C, (C involves time and space), such that their total weight in 400.33 is maximum. Viewed in this way, induction is a (resource constraint) limited optimization problem"

In 1972 Levin (Sug 72, LiV 93, LiV 97) suggested a universal method of solving sequential search problems that can be readily applied to the problem of finding a set of function of maximum weight.

.20 Levin's method (which we will call Lsearch)

If the Q's and A's have short descriptions and n is small, this can be a very practical method to solve such problems. Since solution time is exponential in n.

to find functions of inductive functions of much weight.

Monotony "exponential in n"

For complex problems and or larger values of n, the method can no longer be used to find "heavy" F^j functions directly, but it can be used as part of a more complex method, that promises to take not much that needs time not much slower than linear in n. → 404.31 (Make this a new sentence)

.25 Lsearch

Let us start with a single Q_1, A_1. Using Levin's search technique (which we will call "Lsearch") we find one or more functions, F_1^j such that

$z_0^j \; F_1^j(A_1|Q_1)$ is as large as we could find in the available time.

When we are given A_2 and Q_2, we try the F_1^j functions on Q_2. If $\sum_j z_0^j F_1^j(A_2|Q_2)$ is acceptably large, we go on to A_3, Q_3. If not, we use Lsearch to find a new set of functions $[F_2^j]$ such that $\sum_j z_0^j F_2^j(A_2|Q_2)$ is as large as possible.

Having found these, we look for a function R_2 that can recognize that Q_2 is different from Q_1.

We use Lsearch to find a function R_2 such that $R_2(Q_1)=1$ and $R_2(Q_2)=2$ with b_0^2 (which is the description length of R_2) as small as possible.

SN We could look for R_2 we want both by (approx) a small cc for finding R_2 and small Q_i.

The ratio $\frac{P_C}{cc}$ is one possible Gate, which is not nearly the best one.

402.00

0/20/01

ID

$O_{n+1} = Q_n(O_n, Q_{n+1}A_{n+1})$

.00: 403.40 : try to get 403.18 (A fairly Genl. soln.) early in t. report.

.01 SN NB: T. system as derbd will not try to make "new discoveries", in t. sense of requesting new physical Experiments. Discovering new Thms; Inventing new fields of Math. This has to do w. "Lookahead" & will be implemented only if we give T.M. suitable goals.

Do mention .01 at end of Report: — That this system is easily adapted to solving problems of that sort — i.e. it can be given goals of a suitable sort —

After Derbing O_n as being composed of R_n's $[[P_i^j]]$ (sets of sets.), Explain that this is only preliminary str. for O_n — Not in general, we limited t. form of O_n only to t. extent that we want 4.26 (i.e. summation to n) to be as large as possl. [Get notation straightened out!]

.10 Some likely early announcements of O_n: "sets of sets" ← See 403.32 – .38

If one tried to optimize O_n directly for each n — viz Lsrch, then cc is exponential in n.

Using R_n broken in to n parts makes cc $\propto n^2$; or nk for k parts each. } for recognition alone.

" heirarchical form of R_n reduces cc to n log n or less.

Since we find P_i^j by Lsrch, it will tend to be short (heavy, hywt, hypL) so it will tend to extrapolate well. genz.

Go back to 401.20 (on Lsrch) [Put Lsrch in an appendix.] Discussing use for induction problems.

Notation $F^j \to P^j$ not so good! I'll want $[P_n^j]$ funcs, $j = 1 \ldots k_n$ — well, maybe o.k. → Use $[O_n^j]$ for set of operators so $[P_k^j]$ would be o.k. to use : it wouldn't conflict w.

.19

.20 This conflicts w $[P_i^j]$ $i = 1 \ldots n$ for k difnt kinds of problem solns.

(SN) If we use GA; for obtaining pd O_{n+1} from pd. (O_n, QA_{n+1}) we could find (mut, cross)'s such that t. cands would tend to leave t. solns of $(QA)_1^n$ invariant. — Tho still one would have to spend too much time testing them (would random choice of $(QA)_i$ help much in this?) Anyway — it does involve learning in mut/cross elsln. — so we would have a hyer order GA doing that.

.28 → So Mention Lsrch for finding O_n^j $O^j = P^j$ of 400.26

so $[O_n^j]$ is t. set of O_n^j (for a given n) that have hy a_n^j in 400.33.

.30 ! That this works for small n (time is exponential in n).

.31 ! 401.25 One way to solve this problem is to have these sets $[O_n^j]$ (for fixed n) broken up into 2 kinds of sub-functions. → 406.10

.32 We will do this by having two kinds of functions: first, $P_k^j(\cdot|\cdot)$. $\sum_{i=0}^{n} a_0^j P_k^j(A_{ki}, Q_i)$ (Define) produces an acceptably large probability for a certain subset of values of i ($\le i \le n$).

.34 Each value of i will have an associated value of k,

Or: Each value of i will have an associated integer function $k(i)$. $k(i)$ is the "kind of problem that Q_i is". $1 \le k(i) \le k_{max}$. Usually $k_{max} \ll n$ since there will be many kinds of problems Similar Problems.

406.00

0/21/01
FD

.00 : 404.40 : We will have k_n different functions [R^i] that Recognize the different kind of problems, Q_j..

$R^i(Q_j) = 1$ if R^i recognizes Q_j; $R^i(Q_j) = \phi$ if R^i does not recognize Q_j : ~~$R^i(Q_j) = R^k(Q_j) \Leftrightarrow i = j$~~ ~~$R^i(Q_j) = \delta(i-j)$ (is this rite defn of δ funct?)~~

.02 Associated with each R^i is a set of functions [P_i^j] that solve the problems that R^i recognizes.

.03 If $R^i(Q_\ell) = 1$, then $\sum_j a_i^j P_i^j(A_\ell | Q_\ell) / \sum_j a_i^j$ will be the distribution function induced by the

If $R^i(Q_j) = 1$ and $R^k(Q_j) = 1$ then $j = k$. So the R functions do not overlap.

.08 Set of functions P_i^j, on the set of possible Answers, A, for question Q_ℓ. a_i^j is the a priori probability of the function P_i^j and $\sum_j a_i^j$ is a normalization factor.

— $R^i(\cdot)$ (i.e. for each value of i) — ??! — $R^i(Q_i)$

(404.31) → Perhaps start out by saying: (after 404.31) → One kind of function recognizes what kind of

.10 problem Q_i is. For each kind of function, we have an associated set of functions P_i^j that solves problems of that kind.

For each value of n, there will be k_n different kinds of problems. Usually $k_n \ll n$, since many problems will be of the same kind. There are w_n Recognition functions R^i ($i = 1 \ldots w_n$). Each of them recognizes one and only

.14 one kind of problem. $R^i(Q_j) = 1$ if Q_j is an "i" kind of problem.

$R^i(Q_j) = 0$ if " " not an "i" kind of problem.

The R functions do not overlap. If $R^i(Q_\ell) = 1$ and $R^j(Q_\ell) = 1$ then

.17 i and j must be identical.

→ (.02 – .03, .08)

.14R From time to time New P_i^j's are found that are able to work many Old problems as well as new problems. When this occurs, it becomes possible to use a smaller k_n than before, a smaller no. of R^i functs and a smaller no. of P_i^j functs. So the over-all system can be much "compressed" — yielding better expected accuracy for its generalizations.

We try new P_i^j's on older Q_j's that are "similar" (have common indices, etc.) to Q_{n+1}: New P_i^j was found for.

.19

.20 There are two critical questions about this induction model:

.21 → use both for ranking — ● How do we use it for prediction

● When we find out the correct answer, A_{n+1}, to question Q_{n+1}, how do we repair "update" the system? i.e. How do we update the sets of functions $[R_i]$ and $[P_i^j]$, so that the system will respond more accurately to future problems?

[margin: Via Bayes... but system very low on that A...]

Let us first consider prediction. When a new question, Q_{n+1}, is given to the system, it is applied to each of the R^i functions to Q_{n+1}.

If $R^i(Q_{n+1}) = 1$ and $R^j(Q_{n+1}) = 0$ for $j \neq i$, $(1 \le i \le w_n)$, then the probability distribution induced on the possible answers, A, is given by .03 — i.e.

$$\sum_{j=1}^{j_i} a_i^j P_i^j(A | Q_{n+1}) \Big/ \sum_j a_i^j$$

j_i is the number of P_i^j functions for that i value.

.30 If $R^i(Q_{n+1}) = 0$ for all i, then the system is unable to answer Q_{n+1}.

If $R^i(Q_{n+1}) = 1$ for more than one value of i, we can use a mean of distribution functions induced by the P_i^j's of all of those i values. $\sum_i \sum_{j=1}^{j_i} a_i^j P_i^j(A | Q_{n+1}) / (\sum_i \sum_j a_i^j)$

.33 When updating the system, we are given Q_{n+1} and A_{n+1}. If $R^i(Q_{n+1}) = 1$ for one and only one Recognition

.34 function, and $\sum_j a_i^j P_i^j(A_{n+1} | Q_{n+1})$ is "large enough",

Then no updating will be done.

If any of the foregoing conditions do not hold, we will update. This will, in all cases, involve using L search to find a new set of $P_{k_{n+1}}^j(\cdot|\cdot)$ functions such that $a_{k_{n+1}}^j P_{k_{n+1}}^j(A_{n+1} | Q_{n+1})$ is as large as possible.

10/22/01

ID

an adequate cond could be violation of say format at 406.33.

.00 : 406.40 : If $R^i(Q_{n+1}) = 1$ for 1 and only one recognition function, then we do no further updating, and go on to the next problem.

If $R^i(Q_{n+1}) = 0$ for all n recognition functions, we must find a new R_{n+1} such that $R_{n+1}(Q_{n+1}) = 1$ and $R_{n+1}(Q_i) = 0$ if $i \neq n+1$. We do this again by an L search over suitable functions.

.03 Update conds:

1) R^i is unique. Prodn is adequate → no update

2) R^i " " Prodn now adequate → update? update P_i^j (for a single Q_i)

3) $R^i(Q_{n+1}) = 0$ for all i: (a) update P_i^j (b) find R_{n+1} of suitable properties via L search.

R^i not unique

.10 4) $R^i(Q_{n+1}) = 1$ for > 1 i: (a) If one of P_i^j is that recognizes Q_{n+1} gives adequate performance; Modify that recognizer that gives best P. , Modify other recognizers (via L search), so they don't recognize Q_{n+1}.

.13 5) (b) if none of recognized i's gives adequate prod, modify those recognizers so that they don't recognize Q_{n+1}, (a) update P_i^j (b) find suitable R_{n+1} we are now in situation 3) so continue as 3).

.14

.15 If there is time available, some of it can be spent "improving" the sets of P_i^j : R^i functions. We may be able to find new functions that solve what have been regarded as different kinds of problems — so they are recognized by different R^i functions. If, for example, we find a new P_i^j that can derive adequate sols for problems of the 3rd, 7th and 8th kinds, Then we will use it as single solution type for all of those types of problems. This "consolidation" of P_i^j functions usually gives code compression, which

.20 makes it more likely that the new P_i^j function will generalize well for new problems.

In addition, we will then want to integrate R^2, R^7 and R^8 into a single function that recognizes all 3 kinds of problems. — which usually gives further code compression.

.247

.24 Another way that we can improve the performance of the system is by expressing the R^i functions in hierarchical form. One way to do this would be to have a recognizer that can quickly decide if a problem is of kinds 1, 2, 5, 7, or 8 as opposed to 3, 4, 6, 9 or 10. This might have the time saving in testing new Q's to determining what kinds of problems they are. This technique can be used again to further subdivide the categorizations of the first level.

.30 of categorizations — thus dividing the recognition time by perhaps 2. By using higher order categorizations, we can have a solution time for the nth problem of the order of $\log n$.

.24 ff could be done exponential better: I might start by saying that the system as described, takes time of about $\frac{1}{2}n$ hrs to do n problems, since working the solving of the nth problem requires the execution of the recognition functions. If kn is proportional to n, this gives a time to solve the nth problem proportional to n — much too slow! We can improve the speed performance of the system considerably by .247 ff

By using individual recognition functions that recognize several different kinds (or categories) of problems we can speed up the system further.

:00: 409.40 ⑤ Any aspect of the Model that is "improvable" can be dealt w. by giving it as a problem, to TM. (but the Q is: Just what aspects need improving?)

N.B. ⑤ (.00) is a critical feature of any Induction Algm that can become "infinitely smart".

① & ④ may be adequate, but ⑤ adds fancy "overkill" (No: ⑥ is also imp^t)

:07 ⑥ Look ahead: "Cure Cancer" ect. Look ahead can significantly ↑ problem solving speed.

:08 SN By using These P_i^j's in 11 (unordered) as the R^i's in 11, The slant of overhead for concs. becomes more reasonable (perhaps): It does seem to be quite different from using a

:10 — simple (complicated) O_n function & obtaining pc's of individual concs. within that O_n function. —

(Then perhaps it ends up the same — if the O_n funct is a simple function of the 11 R^i's & P_i^j's.)

:12 But do examine this!

:13 Copy 405.04–.11 here! ≡ "How to write Report"

A possible Abstract.

Abstract.

:16 CP1 — conditional probability machine.

A rather general form of the inductive inference problem, we are given a sequence of Q,A pairs. Then given a new Q (we are required to find a probability distribution on all possible A's. The Q and A's may be thought of as sentences and answers, or problem descriptions and problem solutions, or else inputs and outputs of some unknown stochastic process. The Q A's in the set need not all be of the same "kind" and may be drawn from very different domains (such as linguistics, physics, biology ... ect.

:20 Our system solves this problem using a set of system functions O_n^j that give the probability distributions for A_i as a function of Q_i for the first n Q,A pairs. The O_n^j are chosen so as to minimize the length of the programs that describe the first n QA pairs.

In the initial part of the system's training, the O_n consist of functions that recognize various kinds of probability when different kinds of Q's and associated stochastic functions that relate the Q's to/ distributions on corresponding A's.

The recognition functions and the stochastic functions are updated, when necessary, by using Levin's Universal search, to obtain functions with short descriptions that describe the data well.

Functions of this sort are known to give good extrapolation of known data.

For rapid learning, the sequence of Q,A pairs must be carefully designed.

:30 We will discuss some weaknesses of the described system, and how they can be overcome.

The Introduction goes over the material in the abstract in more detail. (as examples of QA's)

:32 (409.40) Part of "context" is "kind of problem" — this will in turn involve categories & sub-categories of problems used in the Recogn. funct's. We may be able to get QATM to work on the construction of trial functions for R^i & P_i^j, at a relatively early point in TM's training.

:35 Would it be useful to have QATM/ able to necessarily EXALL itself, in various problems Sorts? (411.30)

:36 (409.35) ⇒ "Lrng during Lsrch": Here we can regard this as a kind of QA tm problem — But I'm uncertain as to just how. Perhaps work out some more detail on how the lrng is done & then see if I can express it as a QATM problem. One part of the trick is to suitably index previous cands (≡ mals) that are in "this search"

The Q's & A's will be $[\underset{Q}{cand_i}, \underset{A}{Corr_i}]$ — we are trying to quickly predict which cands will have by G. (411.00)

.00 : 410.40 : this seems like a familiar "o/de problème"! Occurs in analysis of GA. How do we get [redacted] languages that will enable good G prodn.?

If a cand has hy G, then "near by" cands will get a priori hy G. This is obtained by expressing R. "context" hy G cond. è nearby conds in a language, set that small "mutations of Co have hy G because they and Co can be relatively inexpensively predicted from t. same set of Grammatical Constructs. ——

.05 How t. "nearby conds" get a priori hy G, is unclear

.06 Well, t. "Irny during Lsrch" amounts to this: Say one is doing an Lsrch bg guided by a certain P.D. $P_0(cand_i)$ that was obtain from some Empirical data. After 1 or more Conds have been tried (tested, evaluated), we can use that / that cond & data to augment t. emp data that

.10 $P_0(cand_i)$ was based on — giving $P_1(cand_i)$ — a new P.D.

.00 – .05 is simply concerned w. how we get P.d.'s like $P_0(cand_i)$ ··· i.e. what are some good ways to do this?". It is, indeed, a major problem in analysis of GA.

I can, hvr, simply express it as a "well defined problem" that needs a good soln.

If $P_0(cand_i)$ d.f. changes w.r (i.e. w. successive trials): I did have a way to do the "Lsrch" but while it seemed good, I had no "Bound on soln time"

An Approach to Solving it: Look at some problems (solvd / worked on) by GA.

a) Real ~~number~~ space: Optzn problems.

b) Digital spaces " "

c) Mix Real, Digital aspects of dcrn. — optzn problem.

.20 Try to make approxt. models linking dcrn. of cand. to its Gene.

One way is to have Run Gene that's hard to (or to calculate / optimize), We make a simpler Gene ≈ t. original one, e.g. "Spice's Price" that's much easier to Calculate & easier to find it optimum.

Is "Easier to Calculate" is relevant to probs in which / Gene is very expensive (e.g. to do use of / "Spice" prgm to evaluate Cand chts. — or use of a complex simulation prgm to evaluate a "Broom balancing" prgm. I had Risidea of using a "loco/modal" — so as I moved a pt. still, I'd change models.

In "Expert Systems" are kind of "Plan" was a simplified model of t. system. ——

⇒ it was easy to predict/Solve behavior of G. "Plan" model, & one had approximately that behavior in t. system: So t. "Plan" system was good for generating "Conds" ≡ "trials". →412.05

.29

.30 : 410.35 : (On context): T. most. impt idea here is that "context is / induces t. most General kind of Cond'l pc — it is what QATM normally does / does. Consider t. main problem that QATM works on: Given a new Q_i to find / a Good recognizer for Q_i: to find a good (stochy) Function relating Q_i to A_i. ($P_i^*(\ |\ )$)

For t. "Recognizer Problem": Perhaps t. "context" is all of t. previous Q_j's & their Recognizers. | the Q_j, A_j pairs.

N.B: W.o. context (≡ cond'l p.c.) we have UN cond pc — which is usually ≈ (frequency / or Lapis's Rule) of occurrence of various Concs.

(In finding a P_i^j relating Q_i to A_i one Impt big Context is: all previous $\{(Q_j, A_j), P_i^j(\ |\ )\}$ pairs

Dat these "Contexts" (≡ cond'l pc ≡ cpc) give good guesses for P_i^j, which was Rsn Test.

T. cpc is a good "Cand generator".

1,520 in Penn in 2000

1997 1 death every 13 min. = 3.5K / month, 42K killed by auto accidents in US, in 1997

1988: 49K

5.25 min (½ mo) a Pt.

0/25/01
ID

412

34 $\sqrt{93}$ 2.718   5.436   3+1+.7

1.07 / 34   .092 — 1.143   .07 / 34   .60   × 2   .7/34 ~ ~~.14~~ → 1.12   1.426   .13/34   1.3/34   ~~.03~~ .123 +1.137   1.09

7 : 1.85

2/4	1.? 1.4	
6	5	1.6 1.8
9	8	2.1
	10	2.3 2.2
	20	3
	100	4.6
	200	5.3
	400	6

.00 : 411.40 : Wether cpc's (≡ contexts) ~~w~~ for P_i^j cands would be : Just old P_i^j's. or just (Q_i, P_i^j) pairs,

→→ Another impt part of context ~~w~~ for $P_i^j(Q_i)$ cands. is the Recognition functs { R^* assoc. w. $P_k^j(A_k^{A2}, Q_{kk})$ } and, t. R^S assoc w. t. present Q_i. the recogn functs. tell important things about t. Q_i's.

They can "classify" t. Q_i's in useful ways that can help find good P_i^j solns •

Context ≡ "side info" ≡ "Auxilliary info"

.05 : 411.29 An impt kind of **extremizing problem** is, of course **MDL** : This is a rather easy Score in itself. (just list cands in term Length) : **But t. aux conds for MDL** are also impt, (Say we want t. string (≡ pgm) to solve k difnt problems). Then one kind of score would be Length of pgm + α·(no. of probs solved). This is a common trick in GA solns. Hvr, t. **GPS idea** of a **Vector Score** is perhaps much better. It enables us to have a better idea as to "Just what is Wrong" w. a given cand. — & perhaps suggest How to fix it ......

→ 410.16–.30 : I. Abstract, covers t. main content of t. report : At t. end, I will mention/discuss t. ~6 "improvements" of 409.35 – 410.07 : Also some stuff on tsq's : How to get TM to "understand" what a particular index means — for rather arby problem types.

So perhaps list various writings that I may want to use : in factor "Modified".

1) Abstract : 410.16–.30 → 413.00–.40

~~4~~ ~~Update~~ Prediction and Updating 406.20 – 407.14 ← probly also .28 ?

3 On recognition functions : (404.30 – .34) Ran 406.10 – .19 than

2 Start main part at 400.21 – 401.14 ; then 40~~0~~.20 – .25 ; then continue.

] 400.~~2~~20 is an older outline. See how reasonable this is!

1) ABStract 410.16–.30 → [413.00–.40]

2) 400.21 – 401.14 [9 lines of SAZ paper] ← 414.00 – .11   414.00 – 415.09

Then soln as $\Sigma = \pi \otimes P_i^j$ ~~Rap~~ Rapts BPPX I for 2prip compn.

Finding ~~best~~ set of functs w max Σ ~~as~~ in Run CB13 problem

401.20 – .25 : Relevance to Lsch to find solns (sto) 415.11 – .19

404.31 only 1 line "Break $[O_n^j]$ into sub-functions" ~~—~~

40610 – .19 ≐ 406.10 – .17 (406.02 – .02 ; .08).

Note: underline means italics

.00 ; 412.40 : Title : A Conditional Probability Machine.

ABSTRACT

As a rather general form of inductive inference problem, we are given an unordered set of Q, A pairs. Then, given a new Q, we are required to find a probability distribution over all possible associated A's.

The Q's and A's may be thought of as questions and answers, problem descriptions and problem solutions, or as the input and outputs of some unknown stochastic process. The Q, A's in the set need not all be of the same kind and may be drawn from very disparate domains of knowledge, such as linguistics, mathematics, physics, biology, economics....

Our system solves this problem using a set of system functions, O_i^j, that give the probability distributions for A_i as a function of Q_i for the first n (Q_i, A_i) pairs. The O_i^j are chosen so as to minimize the lengths in bits, of the programs that describe the first n Q, A pairs.

In the initial part of the system's training, the O_i^j consist of W_n i.e. different functions that recognize W_n different kinds of Q's, as well as W_k i.e. associated sets of stochastic functions that relate the various kinds of Q's to probability distributions on the corresponding A's.

These recognition functions and stochastic functions are updated (when necessary) by using Levin's universal search algorithm to obtain functions with short descriptions that describe the data well. Functions of this sort are known to give good extrapolations of known data.

We will discuss the design of training sequences that facilitate rapid, generalizable, learning by the system.

The system described is a very elementary one with several serious deficiencies. We will suggest ways in which these deficiencies can be overcome.

10/26/01

ID

.00: 413.40 Introduction

The problem we are addressing is that of very general probabilistic prediction: We are given an unordered set of (Q_i, A_i) pairs, $i = 1 \ldots n$. We are then given a new Q_{n+1}. The problem is to find a conditional probability distribution over all possible A_{n+1}. (Here the "i" indexes of Q_i and A_i are used to indicate Q_i is associated with A_i. The i ordering, however, is not to be used for prediction.)

The Q's and A's can be strings and/or numbers. The Q's can be thought of as questions, the A's as answers. Q_i might be the description of a problem, and A_i its solution. A_i and Q_i might be the inputs and outputs of an unknown stochastic process.

.10 Since we are considering stochastic models, the data can have "errors" in it —

.11 interpreted by the system as "noise".

We will first give a theoretical solution to the problem, involving infinite time and storage — then a discussion of various practical approximations

For the data described above, the probability distribution on A_{n+1} is

.18
$$\sum_{j=1}^{\infty} a_0^j \prod_{i=1}^{n+1} O_*^j (A_i | Q_i)$$

Here $O_*^j(\cdot|\cdot)$ is the j^{th} possible conditional probability distribution relating its two arguments

.20 $O_*^j(A_i|Q_i)$ is the probability of A_i, given Q_i, as given by the function O_*^j

Since total recursive distributions of this sort are not effectively enumerable, the sum is over all partial recursive functions of this sort — which are effectively enumerable.

a_0^j is the apriori probability of the function $O^j(\cdot|\cdot)$. It is approximately $2^{-\ell(O^j)}$, where $\ell(O^j)$ is the length in bits of the description of O^j.

(Appendix 1 discusses the computation of the a_0^j)

.30 We can rewrite .18 in the form

.31
$$\sum_{j=1}^{\infty} a_n^j \, O^j(A_{n+1}, Q_{n+1})$$

Here, $$a_n^j = a_0^j \prod_{i=1}^{n} O^j(A_i|Q_i)$$

In .31, the distribution of A_{n+1} is a weighted mean of all of the O^j distributions — the weight of each O^j being the product of its a priori probability and the probability of the observed data in view of O^j

.00: 414.40: The distributions 414.18 (for) and 414.31) is known to be very accurate. If we use the a_0^j to express all of our a priori information about the data, this distribution is perhaps the most accurate ~~[can could be obtained by any means]~~ possible.

Since we cannot compute this particular infinite sum using finite resources, we will ~~try to~~ approximate it using a finite number of large terms — terms that in 413.31 have large a_n^j values. While it would seem ideal to include the terms of maximum weight, it has been shown to be impossible to know if a particular term is of maximum weight.

It appears that the best we can do is to find a set of terms of largest total weight in whatever time we have available

see 416.00 — 417.23 for discussion of utility of LS search for induction. (i.e. ≠ ENV-type LS search). ← 491.20

.09
.10 → Insert 423.25 — .32

.11 In 1972 L. Levin (Lev72, LiV 93, LiV 97) suggested a universal method for solving Sequential search problems that can be readily applied to the problem of finding functions of maximum weight. If there are a small number of simple problems, ~~If the Q's and A's have short descriptions and n is small,~~ Levin's method (which we will call "L search") can be a practical way to solve them. For more difficult problems and/or a large collection of easier problems ~~longer values of n~~, the method is too slow to find "heavy" O^j functions directly. L search can, however, be used as part of a more complex method with a search time not much slower than linear in n.

.15
.20 One method of this sort expresses the set of O^j being used, as a combination of two types of functions. The first type of function, $R^i(Q_\ell)$, recognizes what kind of problem Q_ℓ is. Associated with each recognition function $R^i(\cdot)$, is a set of one or more of the second type of function, P_i^j, that solves problems of the i^{th} kind. For each value of n, (the number of Q,A pairs thus far), there will be w_n different kinds of problems. Usually $w_n << n$, since many problems will be of the same kind.

.30 There are w_n recognition functions, R^i ($i = 1 \ldots w_n$). Each of these recognizes one and only one problem type. If $1 \le \ell \le n$ then

P $R^i(Q_\ell) = 1$ if Q_ℓ is of type i (italic)

P $R^j(Q_\ell) = \phi$ if Q_ℓ is not of type i (italic).

A The R functions do not overlap: If $R^j(Q_\ell) = 1$ and $R^k(Q_\ell) = 1$ implies $j = k$

[redacted]

Lsrch for OZ probs:
Lsrch for Induction Codes

.00 ; Trouble on 401.20: How Lsrch is to be used for finding good induction codes is unclear

I think I did write a lot on this at IDSIA month.

For t. induction problem: finding short code for a corpus; one way is to make trials in pc order, using CB = pc·T. When all have been run; try T←2T:

Or use parallel ELsrch:

In t present case, we can only test cands that are complete. — source list

If we have k primitive symbols — each w. its own pc, we can use Huffman codes for each of them:

Then consider all binary codes of length ℓ: They will represent Huffman coded symbol sequ's.

|| WOOPS! when we define a new symbol, (or even use a symbol) the pc's of t. symbols change! — So we have to recode w. Huffman!

.10 We test all terminating sequences for Time T·pc. ϵ is time of one clock or some small time. | Is this practical?

for a given value of T, we test cands w. $T\cdot pc > \epsilon$

For a given T, $T\cdot pc > \epsilon \Rightarrow pc > \frac{\epsilon}{T}$ so we test cands of length $< -\log_2(\frac{\epsilon}{T})$

Doubling T means we consider binary codes 1 bit longer.

In using SAZ to generate codes, I had a mechanism to terminate functions;

Perhaps it is unnecy! If we are looking for binary functions, and we have generated codes we test it. If it fails, we simply use it (& its previous subfunctions) as t. basis for t. next

higher level function.

While t above would probly "work"; it is not really Lsrch applied to an OZ problem.

.20 In this last, we try O.T.'s (optim. techniques) in Lcost order.

I've probly workd on this Q a lot!

Hvr, its a lot like Lsrch for an INV problem! — we want a code for t soln.,

So we try them in $\frac{pc}{cc}$ order. In t. present case, I want a [illegible] pgm,

O^j ; $\ni \left(pc_{O^j} \cdot O^j(A_{n+1}|Q_{n+1})\right)^3$ max. ($\equiv Z$) $Z - \log_2 Z$ is essentially the length of t. code of A_{n+1} (wrt Q_{n+1}).

Probly using true OZ Lsrch would be much better — but it would depend critically on what OT's one had available. one had available (placed) now (good)

.30 So, until one has good OT's perhaps OZ Lsrch is not so hot!

I.E. Th. Lcost of a good OT for t. given problem, would be >> solving it as a INV. problem.

Later when TM has had experience doing OT probs or has been giving a good set of OT's & it has "calibrated" them, so it knows when to use which one — Then it's best to use OZ Lsrch.

We will, in any mildly advanced TM, be concernd, in Lsrch, w. not t. uncondl pc of solns (both INV & OZ) but with the Conditional-on-t. problem pc of soln.

Untill TM is somewhat traind in an area of INV or OZ probs, its condl pc's will not be v.g., & Lsrch using them will be very expensive. → (see 417.00)

CPC : conditional proby ~~Computer~~ calculator

.00 (spec 416.40) : Eventually, TM will ~~be~~ have learnd various methods (variations of OT's) to find short codes: Contingent on t. corpus to be coded — so a conditional pc of t. coding techniques wrt t. corpus (i.e. "condition"). While compression coding is a variety of OZ problem, it's a special category of OZ problem, & will, in a "mature TM" have hy pc's for special OT's to be used for compressn coding.

This Condl. pc for Lsch is a very impt. idea! for a young TM, condl. pc ≈ uncondl. pc.

In general, we would always want to use condl. pc for Lsrch, except that it may be more costly — "cc" wise. Perhaps much more costly. In Lcost = $\frac{cc}{pc}$, the ← does this include cost of computing pc?

cc = cost of computing pc + cost of ~~trial~~ generation and test

.09 ~~Th~~ Discussn of "Context" of 300.26 is really a discn. of Condl. pc.

.10 In fact, any thing that anyone (or TM) ever does can be regarded as being primarily based on condl pc (≡ cpc). — i.e. T. task to be done, t. backgnd info, the info from "related domains".

(Def again)

A statement for t. report ? : That condl pc's are important & all pervasive is very clear, ~~what is not so~~ [illegible] Hvr, using the freq. concept of pc, our ssz's for most ~~[illegible]~~ cpc's of interest are too small to be useful for probty estimates. By employing ALP, we are able to use data of ~~[illegible]~~ very many different kinds, to calculate ~~c~~pc's & so ssz's are much more likely to be adequate. ALP makes it possible to use data from disparate domains of knowledge to obtain ~~[illegible]~~ cpc's in any particular area. 53 51 0951

The idea that CPC can ~~work on~~ be "trained" to work on practically any problem

.20 is one of t. impt ideas in QATM. It is sort of related to ~~[illegible]~~ $TM_2 = TM$.

.21 Another impt idea: that "If all info is in t. P.D., that all possl. heurs can be implemented ~~[illegible]~~ via Lsrch — & that t. result will be about as good as using those heurs." [T. idea is that Lsrch is a ≈ universal mechanism that can express/simulate any heur: if we allow long during t. srch. That any heur can be expressed as ① Modifn of t. Building Pd ② a sort/classifying algm. ~~[illegible]~~ used in guiding Pd during srch]

.23 see 423.20 for objection: re: No Lookahead

On 404.31, we run into notational diftys : up to this pt. we've been using $O^j(\;)$ that depends on n. I want to convey t. idea that we will be using a subset of the O^j that depends on n. [O_n^k] is that set (404.19R) ~~as~~ use $O^k(\;)$

I could mention [O_n^k] then say that [O_n^j] is ~~[illegible]~~ composed of 2 kinds of sub-~~[illegible]~~ functions! R^j and [$P_j^{k,i}$] (or [P_i^k]?)

.30 For each value of n, there will be W_n kinds of ~~[illegible]~~ problems usually W_n can. There will be initially, a set of W_n recognition functions ~~that recognize~~ $R(\cdot)$ that are able to recognize which kind of problem a particular Q_i is. $R^j(Q_i) = 1$ if Q_i is the j^{th} type of problem — ~~[illegible]~~ if not, then $R^j(Q_i) = 0$

Replace 404.31 : One method of this sort expresses ~~[illegible]~~ the set of O^j being used as as a combination of two ~~kinds~~ types of subfunctions. → 406.10

Tasks for Lsrch ~~[illegible]~~ required by system as desbd: — (~~[illegible]~~ these are all in updating?) so viz 407.03: 1), 2) 3). 4), 5)

~~1)~~ 2) update P_i^k 3) find adequate $R^{k_{n+1}}$ 4) Modify certain ~~[illegible]~~ recognizers ~~[illegible]~~ 5)

So: only 2 operations ① Update P_i^k so it gives good pc for smaller Q's of type k ② Find Recognizer (categorizer) for certain sort of positive cases " " " negative cases (spec 420.00)

.00; 415.40 The second type of function, $P_i^k(\cdot|\cdot)$ is a probability distribution. $P_i^k(A|Q_\ell)$ gives the conditional probability (as seen by P_i^k) of the answer A, to question Q_ℓ.

~~The probability induced by the set of functions P_i^k (for fixed i) on A, with input condition Q_ℓ is~~

If $R^i(Q_\ell)=1$ then Q_ℓ is a problem of the i^{th} kind, and

.09 $$\sum_{k=1}^{k_i} a_i^k \, P_i^k(A|Q_\ell) \Big/ \sum_{k=1}^{k_i} a_i^k$$

.10 is the conditional probability induced by the set of functions P_i^k on the set of all possible answers, A, for question Q_ℓ

a_i^k is the apriori probability of the function, P_i^k

$\sum_{k=1}^{k_i} a_i^k$ is a normalization factor.

k_i is the number of prediction functions for the i^{th} kind of question.

More generally, if Q_{n+1} is a new problem/question never seen by the system before, its probability distribution on possible answers, A, will be

.18 $$\sum_{i=1}^{w_n} \sum_{k=1}^{k_i} a_i^k \, R^i(Q_{n+1}) \, P_i^k(A|Q_\ell) \Big/ \sum_{i=1}^{w_n} \sum_{k=1}^{k_i} a_i^k \, R^i(Q_{n+1})$$

("w is s.c.")

.20 If $R^i(Q_{n+1})=1$ for one and only one value of i, then expressions .18 and .09 are the same, except that $\ell = n+1$

If $R^i(Q_{n+1})=1$ for more than one value of i, then .18 gives a maneuver the various kinds of problems Q_{n+1} seems to be.

If $R^i(Q_{n+1})=0$ for $i=1 \cdots w_n$, then .18 becomes a zero divided by zero. The system cannot answer question Q_{n+1}.

.07 Section: Updating

A critical aspect of system operation is that of "updating."

~~The most important question about the system is "How is updating performed?"~~

When we are given a new question-answer pair (Q_{n+1}, A_{n+1}), how is the system modified?

.30 How do we update the sets of functions R^i and P_i^k so that the system will respond accurately to Q_{n+1}, ~~future questions?~~ continue to respond well for $Q_1 \cdots Q_n$ and be likely to respond well to (unknown) future questions?

406.33
407.03

.00: 418.10 : There are five possible kinds of response of the system to the new Q_{n+1}, A_{n+1}. Each has its own updating algorithm.

1. There is one and only one i such that $R^i(Q_{n+1}) = 1$ and $\bar{P} = \sum_{k=1}^{k_i} a_i^k P_i^k(A_{n+1}|Q_{n+1}) / \sum_{k=1}^{k_i} a_i^k$ is acceptably large. [Fn 1] In this case the response of the system is satisfactory and no updating is needed.

2. Same conditions as 1. except that $\bar{P}$ is too small.

In this case there are at least two ways to update:

a. Modify R^i so the new $R'^i(Q_i) = R^i(Q_i)$ for $i \neq n$, but $R'^i(Q_{n+1}) = 0$

Devise a new $R^{w_{n+1}}$ such that $R^{w_{n+1}}(Q_i) = \phi$ for $i \neq n$ but $R^{w_{n+1}}(Q_{n+1}) = 1$

Find a $P^{w_{n+1}}$ such that $P^{w_{n+1}}(A_{n+1}|Q_{n+1})$ is acceptably large.

Find a set of $P_{w_{n+1}}^k$ functions such that $\sum_k a_{w_{n+1}}^j P_{w_{n+1}}^k(A_{n+1}|Q_{n+1})$ is as large as possible.

b. Keep the same $R^i(\cdot)$, but modify the associated P_i^j functions, so that

$$\sum_k a_i^k \prod_\ell P_i^k(A_\ell|Q_\ell)$$

is as large as possible. (Here ℓ ranges over those values of ℓ for which $R^i(Q_\ell) = 1$.) This method of updating is usually better in terms of future system performance, but takes more time.

3. $R^i(Q_{n+1}) = \phi$ for $i = 1 \ldots n$

Find a new $R^{w_{n+1}}$ such that $R^{w_{n+1}}(Q_{n+1}) = 1$ but $R^{w_{n+1}}(Q_\ell) = 0$ for $\ell \neq n+1$.

Find a new set of functions, $P_{w_{n+1}}^j$ such that

$$\sum_k a_{w_{n+1}}^k P_{w_{n+1}}^k(A_{n+1}|Q_{n+1})$$ is as large as possible.

4. $R^i(Q_{n+1}) = 1$ for more than one value of i, and one of the recognizers is associated with a set of P_i^j functions that give an adequate $\bar{P}$ for A_{n+1}.

Modify the other recognizers so they no longer recognize Q_{n+1} but otherwise behave the same.

5. $R^i(Q_{n+1}) = 1$ for more than one value of i, but none of the associated P_i^j functions gives an adequate $\bar{P}$ for A_{n+1}.

There are two ways to update in this case:

a. Modify the R^i's that recognized Q_{n+1} so they no longer recognize Q_{n+1}, but otherwise behave the same. We are now in "Update situation 3" and can proceed according.

b. Find a single set of P_i^j functions such that

$$\sum_k a_i^k \prod_\ell P_i^k(A_\ell|Q_\ell)$$

is as large as possible. Here ℓ ranges over those integers for which Q_ℓ and Q_{n+1} have a common recognition function. i.e. $\exists k \ni R^k(Q_\ell) = R^k(Q_{n+1}) = 1$ → 429.00

Omit this! insert 430.10 – 430.20 instead

[Fn 1]: The criterion for whether $\bar{P}$ is "large enough" will change during the course of training of the system. At first, the "largeness thresholds" will be given by the trainer as part of each problem. In later stages of training, the system itself will be able to induce reasonable thresholds — based on its own experience and its goal of maximizing the value of expressions 414.18 and 414.30.

Note for RJS! See 427.39 – 428.06 for a good discussion of this point — "A quantitative soln." ☺

→ 429.00

.00: (spec 417.40): Related to this is the Q of what is an adequate Pc for a set of P_i^k's? — this P_i^k is designed

.01 [If the set refs on a single Q we expect hy pc — it can be rather s.H. — but if for a larger

.02 [bunch of Q's (individual) some of the P_i^k's can be rather small (i.e. the "true" d.f. has occasional A's at small pC.)

Perhaps the pc will have to depend on the cc available. of the R functs

Also, the R functions obtained may not be so good! If one is bad, its assoc P_i^k set

.07 will give poor pc even for large cc. → (423.34)

.08 Another (advanced) trick in update situation (5) (407.13): None of the set that can be recognized

by the recognizer of Anti gives good pc for Anti: One could try to find a new P_i^k function that

.10 works well w. all Q's recognized by those recognizers! This would seem like a "miniature"

Grand QATM problem: It is, but (1) small no of cases (2) the cases are already very similar (2) one has

"side info, the fact that certain functs work de rather well on certain subsets of these problems.

.08 is an alternative method of update for Situation (5).

At first Glance, it might seem that the problem of .08 would tend to not have an easy Soln.:

we have already tried many "short" functs on parts of the problem: they don't work on the whole problem — so

that we need "longer" functs for the whole problem subset. Well, the original P_i^k funct was

designed for the first Q in the set only. It "just" happened to hit the test in the sub-set. In looking for

a good P_2^k for the entire sub-set, the pc constraints are much reduced, since it's only a good

.20 mean (geometric) of the pc's is needed (see .01–.02). The initial recognition set (423.10)

.21 was based on the first Q in the set, and the $[P_i^k]$ that was obtained for it.

Actually, this first Q and completely defines the assoc R partition function

(Also one has to give a lower bound on the P_i^k ($P_b | Q_R$)). (see .00–.07) What we are looking

for in a "proper recognition function" is a R that will do this recognition

cheaply (in cc), and will extrapolate well to new unknown QA's.

.27 Actually, all the info contained in model is in the first QA of each kind in set of its $[P_i^k]$.

Occasionally a new P_i^k will work well for previous Q's. In this case, it is more "general"

than the earlier P_i^k's & we should try to modify the recogn. functs to implement this —

.30 i.e. so that the new P_i^k is applied to the older Q's. & w.r.t. the R functs:

We could just devise an R funct to recognize all of the old ones plus the new one,

.32 or keep all of the old recognizers. 420324 This would give multiple recognitions for certain Q's.

Which is a modification of this QATM that I have considered. I suspect that eventually,

.34 I will end up doing that — the update process may be much more complex.

A further Ganzn. would allow Recogn functs to assign wts to Q's — this would

be a fuzzy (Recognition definition). Evaluating fuzzy Recogn Algms involves applying them to

.38 Very many QA's — which is expensive. One may develop ways to restrict them

.40 to a small set of Q's in which they have wt. > some ϵ, say.

ID

.100: 420.40 : one mite want eq. of 418.18 with "fuzzy" R^i's, — subject, perhaps, to the modifn. of 420.39 - .40

.02 Actually the multiple Boolean recogn of 420.324 - .34 may make updating easier. — It may make it easier to find R-functs that categorize things in useful ways.

In general one mite have an interaction betn P_i^k's & R_i's : we try a P_i^k — see which Q's it works well on — then try to find a short, fast R^i for it. We can do this many times & get a (set of P_i^k & associated R_i functs. We can then use 418.18 (with Boolean R) to obtain probys, predns.

SN For categorizn. problem of multiple recognition : ID3 would perhaps work. I suspect that it could be much improved. It does have sub-categorization, so distinct R^i's could jointly use them : Also, it tries to get "final" decision in as few "micro decisions" as possl. ID3 may not be ideal, but it may be adequate. It is a more or less "off the shelf" item. (16)

A Q is : would Lsrch be better? For "ID3" & other Categorizers "Features" must be first designed.

.10

.11

SN for finding P_i^k, Lsrch can be used : or GA w. heavy bias on short codes, built into "fitness function". All old P_i^{k-1}'s will be in population to furnish "parts". We could start out in a relatively simple, mixed-type GP, Waterman (w.r.o. recursive functs, say), then add capacity for complexity, as we acquire good P_i^k's to be used as "parent stock".

.16 : (11) — (ID3) : May be not "off the shelf" at all! ~~ID3~~ write T. recogn problem is a binary classifn task, it is to recognize strings, & is rather general. ID3 has to have a vector of discrete properties for each element (CQ) to be categorized. While we mite eventually have such functions, (& they would be useful) they are not immediately available "ab initio" "from ambush". [ID3 needs a "feature extracter" at "frontEnd"

.19 For the recogn problem, it would be useful to try to develop a set of categories (for a ID3-like tree decision process) so that Recognus decisions would be fast. A possl. approach : Start w. simple, "first order" obs on Q's. The output of these can then be combined (Logical) Boolean ways to create higher order obs. A way this might be done : Say we have already found a set of R_i^i's for the Q's thus far. This set of R^i's is slow, but is not particularly "stout" — my pc. In devising Boolean categories (perhaps by Lsrch) a possl. Goal : to divide the set of Q's into 2 approximately = parts. This could be good "first order obs" (.21).

.20

.21

I think there may have been lots of work done on "strictly categorizn" problems. Perhaps extensions of ID3. It's likely that Vapnic's system has been used for this. — My impression is that it is slow, but this may be false & fixable.

For small n, the R^i's will have to be revised after each $n \to n+1$; But after we have a large enough sample for a R^i, revision will occur less frequently (or at all!) — This would be true indefly of however find the R's — tho it would converge faster if we do the "ALP-oriented" Lsrch.

That recent book on pattern discovery by Duda, Hart & Stork will probly have a lot of info on categorizn. problems for strings & perhaps on P_i^k discovery. Will Grosh may have some ideas & refrences.

After (or even before) we have a rather stable set of R's, we should begin to optimize the current set of R, P functions in various ways. We can optimize new R's (for fixed P set), w.r.t speed & pc. For non fixed R, we can try to find P functions that solve a larger no of problems at lower cost, lower cc, than existing set of P's. The R's would then have to be revised.

SAVE 422.00

10/29/01
ID

GA :.20

spec
.00: 421.00 : In fact, the whole idea of QATM is to have 1 P_i^k function for the entire corpus! So one approach would be the "problem pool" approach. We try to solve as many problems as possible w. as few functions as possible. This can be done in an elegant way by "merging" pairs of P's, then repeating the associated R's. This merging of pairs of P's can continue until we have once big P for the entire corpus.

Of course we already have that, using the many R's & P's, but the pair by pair merging produces a cost & a/o cc at each merging.

Deciding which pair of P's to/try to merge is an interesting problem. After chosing the pair, one has the 2 P's & their subfunctions — which should Heavily Bias the search for the "merged P".

.10 T. The associated R functions could also give useful info. We know that the 2 R's & the 2 P's can be combined to yield a single R & P, but at [both r & c] costs that we

.12 want to reduce. 423.10 is a possible way to find good MERGE candidates.

ABCDEFGHabcdefghij

An Unpleasant note on "Partitioning": Given a set of n elements, the info needed to partition it into the 2 specific sets of 2 and n-2 elements is maximum when $n = 2^2$, my "soln" to problem — well I had 2 solutions — unclear as to which was better. The exact nature of apriori info involved would have to be stated to resolve the problem: Branch 1 soln was $\frac{n!}{(n-2)!\,2!}$ which is the "Binomial distribution" Gauss d.f. for that eqn. I say "unpleasant" because that that is $\binom{P}{R}$ dividing the corpus into the 2 parts would be best. — Well maybe it is best if its of small rc, because it does a "lot of work" w. small rc ($\zeta \equiv$ raw cost $\equiv \frac{1}{pc}$).

Def:

.20 On using GP/GA to find O_n: One apparent Big Objection was that it would take too long to test/verify each cand. However, maybe not so. By using statistical testing on the QA's, one can make a Good Estimate of the Goodness of a Cand, w.o. testing many QA's — in fact one can chose "difficult" subset of QA's to use for prelim testing. If a cand passes prelim tests, we test using More (maybe all) QA's. Since the population of parents are mainly vigs over the whole QA set, it may be that if a cand does well on, say, 10 of the QA's, it will tend do do well on the rest of them! If true, this could reduce considerably

.30 the cc needed to find good cands (good O_n's). → See 432.20 for discussion of importance of .20 ff
.31 & how to usefully use the GP/GA system.

A very brief summary: 1) The use of (.20) – GA for finding O_n is fairly non-el, & if it works, could give us e.g. TM. That is non-el, it is inherently slower than much more el. methods I mainly discussed.

.36 2) T. el. method described essentially reduces to finding a P_i^k for a new problem ② finding and revising R" (recognition) functs. The system obtained may be good enough (w. a suitable tsq), to work on problems in practice vastly less el. ways.

3) Even w. the most simple "problem type" version of QATM of .36, there are (423.00

.00: 422.40 impt improvements that can be made, e.g. the (merging / coalescing) of P_i^k functs. (422.00-.12).

Developing Categories & subcats for R-functions (421.19 ff) could reduce cc of solns & also ↓ rc of soln. (420.08-.28) Gives a less el. soln. to "Update situation (5)" of 407.13. It reduces no. of P_j^k's & R^i's. So ↓ of cc & ↓ of rc.

420.20-.32 is a beginning of a discn of just what this "bare bones" model is, & what its basic weaknesses is.

420.27 ff is particularly impt.

10 } 420.21 If $R^5(Q_{n+1}) = 1$ & $R^{10}(Q_{n+1}) = 1$! This suggests that P_5 & P_{10} may be "similar": Try to find a common P for Q_5, Q_6, Q_{n+1} : P_{10} unite to test that P! [

.10 is one possl. heur for "merging" of P's. I will need to find other heurs for this, since MERGING is a very impt Operation.

The Merging itself is perhaps standard Lsrch, but more generally it is a problem w., in t. cases of .10, "side info" of R^5, R^{10}, P_5, P_{10}. → (see 420.08-.21) for more discussn.

A "kind of heur" Suggested by .10: suppose we have this set of R^i's that don't "overlap" (e.g. for no i is $R^i(Q_2) = R^j(Q_2) = 1$, than we can test some overlap (as .10 can be instead) by "weakening" one or more of the R^i — "weakening" means removing constraints — in general "shortening" [illegible] of this.

20 SN I had idea of 417.2 that Lsrch w. Irng could simulate any heur. May be Not so? It would not simulate "Look Ahead": T. doing of "experiments" to gain info (at some cc). ("Look Ahead" May be wrong term. It's used in Pgming Chips, in which several policies are simultly taken — Then all but one is abandoned. Then later — all but 1 policy abandoned.)

(It is Myopic & Greedy)

Perhaps use term "Expt." rather than "look ahead" — until I think of a better term: A non-greedy "Far sighted" rather than (Greedy Myopic)

.24 Insert at 15.10 : (See 422.20-.31 for some encouraging theoretical discn of GA for this problem.) (95 ?) (koz)

.25: Insert at 15.10 One way Genetic Programming, using a Lisp-like functional language, would seem to be a reasonable way to solve this problem. A large population of P^j functions could be mutated and/or recombined w. suitable crossover algorithms and a suitable fitness function. ~~Preliminary analysis~~ A first glance suggests that while this may give very accurate probability distributions, it would be ~~very slow, as compared to~~ much slower than the system we will subsequently describe. This is, however, only a first impression and we have by no means abandoned Genetic Programming as a possible solution technique.

.32 ~~for solving this problem~~.

.34 :420.07 on "Adequacy" of pc's assgnd by System" ~~above~~, This has to do (intimately) w. t. Update algms, since it is t. criterion for discard/acceptance of R^i, P_j^k functions

As was noted in 420.01-.02; sometimes it's o.k. for a pc for a particular Q to be small.

A possl. rule: select a P_j^k set so that [illegible] $\sum_k z_j^k \prod_\Delta P_j^k(\text{@}A_\Delta)Q_\Delta) / \sum z_j^k$ by k is as large as possl. for the amt of search time used.

Probly learn about this "forgiveness factor" → (space 424.00)

10/31/01

ID .....

(Spec: 423.40): Usually in induction problems, there is a "default code" equivt. to the term "By chances" ("Random") or some trivial regularity. ~~Getting a code at least as~~ If one does not, in t. available cc, get a better than that, then one confesses failure & uses default code.

→ So, in t. updating problem, t. main idea is to get max pc in available cc.

There are 2 impt. subgoals of a "Bare-Bones" system:

1) Being able to usefully work on t. "Updating problem" as a "regular problem"

2) Being able to ~~use~~ read & "understand" Nat. Lang. to a useful extent.

50°
44°
6°

Q: Is our Gone to get highest possl. $\sum_i a_i \prod_\ell p^i(A_\ell|Q_\ell)$ or $\sum_i a_i \prod_\ell p^i(A_\ell|Q_\ell) / \sum_i a_i$

or $\sum_i a_i \prod_\ell p^i(A_\ell|Q_\ell)$ v.s. $\sum_i a_i \prod_\ell^{\ell_{max}} p^i(A_\ell|Q_\ell) / \sum_i a_i \prod_\ell^{\ell_{max-1}} p^i(A_\ell|Q_\ell)$ ← This is cond pc of $A_{\ell max+1}$ in view of $Q_{\ell max+1}$

Go Back to (578T3): Say a_i is apriori of code p^i.

Perhaps easier: 414.18 or 414.31 If we maximize $\sum_j a^j_n$

j sums over all codes. That would seem about best. It would get most wt. for t. Corresp. predn "algms". In t. Update case, we want Max $\sum_j a^j_{n+1}$.

Here → say we are extrapolating string X: for best induction we want total wt. of all pgms that of strings that have X as prefix.. T. larger a total wt. of our pgms, t. more likely it is that they will have much wt. for t. "Rite" pgm (??)

Ideally, we'd like to use the shortest pgm; which is max a^j_n; even better, would be to include t. highest a^j_n pgms ∋ their total wt. was ~ 90% of all wt.

While it is true that t. more wt. one has in $\sum_{j=1}^{\infty} a^j_n$ t. closer it is to $\sum_{j=1}^{\infty} a^j_n$ & ∴ more "accurate";

This is not actually t. problem! We are interested in condl. probys — i.e. will weighty sets of predn functs give better predns than less "weighty" sets. ?

Consider predictors of P(1) v.s. P(0) as a function of previous (binary) strings, X. Say F^j is the j^{th} such predictor in t. set. $F^j(x)$ gives a p.d. over 0,1 for t. bits following x, for orby binary x.

Consider t. ~~a wtd~~ a^j_0 wtd sum of these for the n^{th} bit of x (x_n = first n bits of x).

The p.d. is $\sum_j a^j_0 F^j(x_n)$. or let $F^j_0(x_n)$ = prob. of 0 being next bit of x_n

so $\sum_j a^j_0 F^j_0(x_n)$ = pc of 0 by sum of

I think t. end result is that if we use .30 for predn of t. sum,

No! we want to use $\sum a^j_0 \prod_{i=1}^{n} F^j_\ast(x_i, x^{i+1})$ as wt. of F^j after x_{i+1}.

This is f^j's pc of $i+1^{th}$ bit in view of past known x_i.

If we do this, this then t. macro predictor ends up w a pc for t. x_n corpus of its total wt. at that pt. So we can use ~~the~~ Corollary of 578T3 for "true" pc of corpus to compare it to t. true p.d. : So larger is better, since expected $\sum err^2 \leq \ln$ "approximation" pc of corpus

I think t. Algebra is very simple!

exercise user name — 24 hour fitness — 5×24 — rg'sol — prgaaya

(.spec 399.40)

.00: 424.40: Re: the Question about $\ln \frac{P'}{P}$ always being <0: (worried about it: also earlier 399.33–.40)
I think it has to do w. Expected values & t. Kull–Lieb. distances which are always ≥ 0. Hvr, I'm still uneasy about this! Work it out in some detail.

→ Gac's theorem about Serr convergence for semi measures: is this implied by S78T3 corrollary?
No! T. corrolary is for ratios of Measures only.

A It may be better to Consider only 3 Update situations: (rather than 5)
1) $R^2((Q_{n+1})) = \phi$ for all i
2) $R^2(Q_{n+1}) = 1$ for 1 or more i
a) T. best i is adequate.
.10 b) " " " " not adequate.

In t. 5 situations approach 2) (above) was split into 4 rather than 2 cases. i.e. $R^2(Q_{n+1}) = 1$ for 1 i (value ok / value not), for >1 i (value ok / value not) ~~considered if there was only 1 soln. v.s. >1 soln.~~
Doing it in 3 cases makes discn. of "what to do if one has lots of cc for update" easier to desc.

SN Is t. "No free lunch thm / Ugly duckling thm" true ⇔ we assume a uniform d.f.? (I think so)
If so, this is clearly ridiculous: Only t. U. d.f. gives "no induction" any other d.f. gives some induction.

.16 **Exposition:** The expected total error is a monotonic ↓ funct of P. — So maximizing P is equivt to minimizing Expected total error.
? This is the most impt idea in t. paper: There are several ways to find functions w. large values of P. All induction techniques ~~find large values of P~~ find functions w. large values of P. can be compared and evaluated on t. basis of how well they do this.
.20 The rest of the paper will mainly discuss one particular method of ~~approach~~ to devise techniques ~~getting~~ a larger value for P. The reader may ~~[illegible]~~ greater efficiency.
effiecency efficency efficient efficent efficeint
$x^n e^{-x}$

(This will have to be preceded by an explain of S78T3 & its ~ result, then how 418.18, .31 give ~ result.)

SN Perhaps its possl. to show that t. convergence of Serr directly implies convergence of other error measures: This would probably give a lower upper bnd on convergence Rate than Marcus Got.

419.00 ff: Criterion for update is unclear: When do I want to use t. $\frac{1}{\sum_k a_i^k}$ normzn constant?
I do want good predns — (which means use a Normzn Const). On t. other hand, the Serr is ↓ funct of
.30 $\sum$ wt, which does not use a normzn. const!
Consider t. Deterministic Predn (MTM). What indication do I have of "rite answer"? — Well, there is t. default code. — I want to do much better than that! (Default code would be perhaps based on freqs of symbols or of various common abss, cones.
.34 — One approach: Just try to get as much wt. as possl. w. t. available cc. — That's it! —
If "no updating is needed" (because t. predn was v.g.), spend cc on other aspects of wt. ↑ — more gt look for more "global" regularities.
→ .34 seems to deal w. t. problem of occasional hy r cost predns — that must occasionally occur. ←
Well .34 is a non-ali soln, but I have broken down t. problem into R^i & P_i^i functions →426.00

N/2/01
ID

6.27386 stys HSCK
→ 440.552 h   21.77%/yr to date
~ 456.537 expected turn end of year

.00: 425.40, so t. Q is: how much cc to spend on R^i modifn.? on P_2^i modifn? — On modifn. of Older R & P's? On "unifications" of R's & of P's?

Somehow, a Good Prediction (say pc = .9) would signal that one need not spend more time on t. R & P for that QA. .... Not necly a correct conclusion, hvr!

So t. Goodness of (a) predn(s) can be regarded as a heur, that suggests which R's & P's to work on — it Guides t. srch for hype.

Now, It may be that t. TSQ (is/has to be) initally designd, so that (goodness of) predn is a good heur for (allocating use of a) focusing cc, & telling "when to stop". In fact, initial STQ's are designd

.10 so that t. problems are "solvable", so one knows when one has solvd them, & one wants to Go on to a new problem. It is usually in the very advanced part of training that TM is Given problems of unknown solns — Where even t. trainer doesn't know if a soln. is "correct"/adequate.

.40 [Perhaps "an analog" of TSQ, t. trainer could actually Tell TM what pc levels to target for solns to problems — as part of t. "problem desn".]

At first, most of probs will be MTM, so pc threshold ~ .999; But we also want TM to lrn problistic things, so it can find heurs.

An advanced TM should be able to guess how much compression to expect w. a certain cc on a certain (problem/or sub-problem). As TM works on t. problem, its estimates will change.

.20 -

One impt kind of "stochastic" problem: Devising Good concs based on "Context" (a condl pc prob) Here, it is not clear as to "when to stop" — when progress has been made — tho a pc >> default pc is much "progress".

(SN) Tho I want to Maximize wt. T. goal is not max(wt/cc): because any monotonic fn of wt. would also be maximizable — wt^2, ln wt, E^{wt}, $\sqrt{wt}$, $-\frac{1}{wt}$.

— Hm, why did I need a linear ordering of (pc, cc) pairs? — Say I just want max pc for any particular cc. Well, \$100 w. pc = 10^{-10} v.s. \$10,k for pc = 10^{-2}; to chose betw. t. 2 depends on problem(s) being solvd.

.30 For fixd cc or fixd pc, t. answer is always clear! Hvr., paths in cc, pc space may not be possl. to chose between. A kind of Modified "cure cancer" in which some "output" occurs befr final results.

So: A Q is whether .40 is correct/necy. Do we have to tell TM level of pc required for prob. soln. Perhaps it is needed only because we have broken down t. "main problem" (≡ Max wt. of corpus) into "Subproblems" (≡ etc.),

Another possy Mite be: for each problem a certain "Min Compression" wrt Default coding (Same for All probs)

Hvr. — what about TM working on itself — what pc thresholds to use? ——→ 427.00 space

N 2/01

ID ★

.00: (spec 426.40). Perhaps TM could lrn how much to allocate time as a "Meta Lrng" problem!

That may be it! | After TM has had many examples of "pc thresholds for solns" given by (trainer) USR, it will be able to induce them for given problem, — and it has a "GRAND GORC" for doing this — i.e. max total corpus (code) lnt. for a given cc.

So, main outline of TM so far: 1) 414.18, .30 **Max. wt. Gorc**

2) Break down On into R, P functions.

3) Solve R, P problems, guided by pc thresholds / initially given by USR.

— use (Lsrch (using some lrng.)) a/o GP or whatever to solve R, P problems.

.10 4) Continuously be trying to Merge R funcs w. other R funcs → P " " " P & main thrust.

So that's it for "First Level epc"

.12 **Second Level:**
Next level: Get (above) machine to work on folg. items:

1) In Lsch we are guided by a p.d.s a) Prob lrng in this p.d. b) make p.d. conditional on various things — so computing it becomes a QATM problem. [i.e. t. problem itself, previous solns to "n problem"; (similarity judged by common R funcs. t. & other discovered "features")]

.16 2) Solve "Look ahead problems": so it can learn to use "Look ahead" in its own prob. solving.
≡ do "experiments": actions not immediately directed toward option of Gorc.

3) Devise its own criteria for a) indexing b) how much cc to spend on a problem/sub-problem.

.18 4) Work on problem of extrapolating goals of Sci Community..

5) It should learn to devise and solve problems that are oriented toward t. "look ahead" goals

.20 of .16, or t. goals of .18

• 6) Reading Nat lang & and "understanding".

7) Do more "Merging" of R and P functions.

.23 These "cc thresholds" can be mult by 1.5, say, if TM has more time available: Essentially, it gives t. ratios of pc's spent on various sub-tasks → we can regard them as being done in ||.

For "R" (recogn. problems): These are strictly INV problems. One could accept t. first soln found — or continue to search for solns. of higher pc. — Essentially, one would not consider solns of lower pc than t. one found. — so a limited set of cands — & one spends more time on each of t. cands that haven't yet converged. || —

.30 On t. other hand, t. P* problems always have "soft" solns, A "soln" is anything better than "Default". "After a better than default" soln has been obtained, TM could spend equal times on improving R & improving P solns. T. goal is to minz products of r costs of t. 2. So t. problem is — given a certain amt. of time, what is best division of it betw R & P improvement tasks? One could first solve R (w. min $\frac{cc}{pc}$ soln), then spend rest of time on P. — But after R is solved, one should, at first, spend much time on P, but eventually, we get to pt. of diminishing returns. I think one works on R or P

Very Important

.39

.40 ((depending on whether $\frac{d \ln P}{d\, cc}$ or $\frac{d \ln \text{pr } R}{d\, cc}$ is larger. **Second level**: could estimate these 2 derivatives → (428.00)

.41 The 2 derivatives are prob'ly both ↓ functs of cc.

Contextual

.00 : (Space 427.41) : How Level 2 TM (estimates / calculates) these derivatives is unclear. I guess "the usual way": Various contextual clues, averaged over many problems in the past. We keep "records" of that activity. Also, I'd like TM to "Think" about the problem — use reasoning to get better solns. — but this is an aspect of TM behavior that I haven't worked on much. My impression is that there are many "Math/logic" problems that are close to this, that could be given to TM as "regular problems" in [redacted] training to "think about" its "real" regular problems.

.06

.10 Trying to get Eq. for 419.30–.32 correct. "5b" update

Also Find a new R that recognizes all of the Q_ℓ that were recognized by a R^k that recognized Q_{n+1}. Otherwise $R(Q_\ell)=0$.

i.e. if $\exists k \ni R^k(Q_\ell) = R^k(Q_{n+1}) = 1$, then $R(Q_\ell)=1$.

$R(Q_\ell)=$ [redacted] if and only if $\exists k \ni R^k(Q_\ell) = R^k(Q_{n+1}) = 1$.

Discard all older R^k's $\ni \not\exists Q_\ell \ni R^k(Q_\ell) = R^k(Q_{n+1}) = 1$

Describe 5b merging in English w. much care. If I can find a good way to do this, Mathematically, symbolically, do it following English dern.

Discuss how dern length of O_n is "sum" of dern lengths of P_i^j (but this must be made exact: it can be very confusing! — It's easiest to explain if in P_i^j, j has only 1 value: i.e. for each R_n^i, only one P_i. Perhaps say $\ell(O_n) \approx \sum \ell(P_i^{\ell_i})$, $P_i^{\ell_i}$ is the pc of shortest dern of all functions having same i value. $\ell(P_i^{\ell_i}) = \min_{\ell_i} \ell(P_i^{\ell_i})$

We want to ↓ W_n, the no. of Diff. R types, i.e. no. of different P_i^j types.

Try to find best correct Notation — Maybe look in Li–Vitanyi Book?

.20 [SN] The idea of having many ∥ codes (functs) P_i^j (many j values) was partly to ↑ total PC, but more to enable updating. The ∥ codes are able to store many useful concs — They help form a "Population Pool" for GA. However, it would seem that for a given i value, the concs in P_i^j of different j cannot be "shared". On the other hand, with $P_{i_1}^j$ & $P_{i_2}^k$, if i_1 & i_2 are different, then $P_{i_2}^k$ can share concs for all values of j in $P_{i_1}^j$. The "weight" of this sharing ↑ if $P_{i_1}^j$ [redacted] [redacted] assigns by pc to [redacted] the correct answers for the problems it works on.

On the other hand, one would want to share conc betw. different j values of same i value in P_i^j, because "same i" means they work on the same problems, & are more likely to be ~.

.30 So .20–.30 is a "PARADOX"! — funct. derns w. common "i" will not "share", but ↑ pc's by being in "∥".

Perhaps write $a_0^n \approx \prod($ $) \cdot \prod \ell(R^i)$?? (or whatever apriori of O_n is) "O_n" really doesn't have an aprip! The formula for the wt. of the entire dern involves the aprips of P_i^j in the P_i^j; but I don't think there is any way to assign an aprip to O^n! O^n is a set of functs (operating in ∥). An aprip can be assigned to each component function. Perhaps give example of a $[O_n^k]$ set as a cartesian product of W_n sets of functs. On 414.18, .30 We use the set of functs $[O^j]$. For each n, we will have a (often different) set of functions, $[O^j]$.

One way to get the kind of "interaction", "synergy" betw. ∥ O^j's is by "cond'l pc":

→ 430.00

00: 419.40 : ~~[illegible]~~ While this method of updating is usually very time-consuming, it is ~~mmm~~ a very desirable kind of update since it "Merges" P_{ij} functions and it also merges the associated R functions. These operation can ~~[illegible]~~ give a ~~[illegible]~~ very large increase to expressions 414.18 and 414.30. They can also speed up ~~[illegible]~~ considerably both ~~[illegible]~~ recognition and prediction.

.05 Insert 430.02 - .05

10

20

30

N3/01

ID

This is original "Dup" 430

Grace has Dup of this.

ABCde ABCdefg ABCdefghij $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}}\,dx = \sqrt{2\pi}$ $\alpha\beta\gamma\delta\rho\pi\xi$

.00: (spac 429.40): This might be done by using all || O^j's as "context" $\xi \equiv cond(pc)$ for new trial O^j's

This looks like GA(GP). — But I think it just as Relevant to pc's for Lsrch

.02 (for 429.05) This large increase resides mainly in the apriori probabilities of the O^j functions. Each such function consists of W_n sets of R^i and associated P_i^j functions. The description length of O^j is approximately the sum of the description lengths of the R^i and the P_i^j.

.05 By "merging" the functions, we reduce the ~~description of O^j which increases its a priori probability~~

.06 number of functions that need to be described, — which increases the a priori probability of O^j.

.10 b. The fact that several of the R^i accept a common Q_{n+1} suggests that the R^i and their associated Θ ~~Q_ℓ's~~ accepted by these R^i have something in common. They become a good "merging" candidate.

possible to find a single set of P_i^j functions that can work ~~all of the~~ we look for a single set of P_i^j functions so that

$$\sum_k a_i^k \prod_\ell P_i^k(A_\ell | Q_\ell)$$

.15 Not this but this

.16 is as large as possible. Here

17 Q_ℓ ranges over those "accepted" Q_ℓ's. i.e. $\exists k \ni R^k(Q_\ell) = R^k(Q_{n+1}) = 1$

If we are able to find such functions, we can eliminate the R^i functions (that accepted Q_{n+1}) and find a single new R function associated with the P_i^j's of expression 430.15.

.20 presumably, larger than the corresponding weight, using the "unmerged" P_i^k functions.

.21 Actually, .15-.16 is trivial & of no interest, unless we include .21.

Perhaps we should not have 5b at all, & just have this "Merging", as an impt. part of 4.

Update process; if we have time, after update 5a

.27 b. If there are two or more R^k such that $R^k(Q_{n+1}) = 1$, Then the P_K^j's associated with those R^k's are likely important features in common. We can take advantage of this "commonness" if we can find a single set of P_n^k functions that is able to solve all of the problems formerly requiring many P_n^k sets. This would reduce the description length of O^j, and increase its a priori probability, thereby increasing the value of expression 414.18.

Perhaps instead of .10-.20

Eliminate after 429.20

.30 for

.32

.00: 430.40: Summary and Discussion

The system described ~~in~~ operates on a great variety of induction problems. Because it is based on the maximization of

.03 expression 4(4.18), we are certain (.035) that if the regularity in the data is describable as a finite binary string (plus perhaps a finite number of continuous parameters), then if we use L search for maximization, we are guaranteed to find the solution in a finite time. For any induction problems of interest, this "finite time"

.06 is too long, so we must use approximation methods.

The approximation methods suggested, are based on human problem solving. We first identify the nature of a problem, then we apply solution methods

10 that we have found to be successful for that kind of problem. This corresponds to our finding the recognition functions R_i^r and the conditional probability distributions P_i^k.

The "merging" of solution techniques that we employ corresponds to an important part of "scientific Method": the summarization of many "special cases" theories by a grand, more encompassing Theory: Galileo's laws and Kepler's laws summarized and extrapolated by Newton's laws.

.17 The employment of carefully designed sequences of problems to train the system corresponds to the norm in human learning. [illegible] seems quite different from most human training experiences, in which learning occurs → 437.26

commit for present

Insert at .035

We are certain of two things:

First; that problems of widely different domains are able to share their concepts with each other. The techniques of problem solving and the concepts used in physics can be readily transfered to Mathematics, Sociology, ect. } This is not made clear!

Second: .035 If the regularity in the data . . .

.29 : 437.31 It is of interest to note that the training sequence will teach not

.30 only the system, but the trainer! By observing how humans seem to solve problems given to the system, will suggest modifications of the described probably learning techniques and entirely new ones.

One serious deficiency of most present learning systems is their inability to transfer information obtained in any problem to the solution of subsequent problems. There has been some work done in this direction. "Incremental Learning" is perhaps the category that best fits the system we have described. Another, more recent direction of research has been the use of "wrappers" for machine learning — the continual adjustment of machine parameters 432.00

abcde

.00: 431.40 that transfers certain kinds of learning between ~~some problem to~~ successive problems.
We feel that the system we have describe may do this in near optimum manner.

.01

.03 Discuss Ability of system to Do "Meta Irng" — corresponding to Human ..... ? Put this ~ after 431.17

SN How Tolerant is ALP of "errors in Coding"? I mean serious conceptual errors — say, the use of base 2 for one portion an expression & base 4 for another: Or the commiting of the "2141 feature" of assigning pc ∝ no. of times of past use — ? well, it depends on the error type. Some are probly minor — others catastrophic. Committing the "2141" feature could destroy any induction!

.10 Some things to include in Report: (see 427.12)
1) Why "Merging" is impt: a) # of pc 2) # of cc so there can be a linear.
2) Why have many codes in II? Why not MDL? see 428.20.
3) Maybe discuss 427.40 ff on How much time to spend on R v.s. P discoverys.
4) 425.16 ? More discussion of why maxzn. of 414.18 is a Big Deal.
.15 5) Explain how the QA form of problem presentation is ≠ Univl. (426.04

Nuclear ~~Nuclear~~
31 Oct 2001 sunset Vuces

.20: 422.31 → **An Alternative General Scheme for QATM:**

To solve 414.18: I opted to elementalize O_i — Break it down into sub-funcs of certain kinds.
(we have 2 aspects of SciMethod. Breaking problem into Parts (Analysis: & R^2 funcs. Integrating several apparently difrnt parts of a Science (or difrnt sciences) SynResis: "Merging".

The other Option is to solve for O_i's directly for each value of n; (This could be very cc intensive) — via Lsrch or GP. Meanwhile, we are trying to Get TM to work on the $[O^j]_{n+1} = [O^j]_n$ operating on Q_{nt1}, A_{nt1} & other Contextual info). Sot. system

.25 is set all times, ~~maximum~~ maximally "Non-el"

.30 $\equiv [O^j]_n \left([O^j]_{n+1} \mid [O^j]_n, Q_{nt1}, A_{nt1}, \text{other "contextual info"} \right)$

→ so we start out w. Big fast Computer system needed. This is true in till TM begins to be able to solve the problem faster via "Cond. probly" (i.e. QATM problem) This is usable for Both Lsrch & GP approaches.

Use of GP to solve the initial tsca probs might be nice, because we have a good population of O^j conds, & this population is slowly changed as we move thru TSQ.

A serious problem is cc of testing conds but this may not be over powering powering — See previous notes on this → 422.20-.31 ~~[illegible]~~.

Mainly, .30 is the problem of getting a good p.d. for O_{nt1}, as a function of the T. "problem" Q_{nt1}, A_{nt1} 2. The Solns. for Upton $(= [O_n])$, any other "contextual info". Before we get to this point, the conds for O_{nt1} are based only on the p.d. for O_n → the frequencies of use of various sub-functions within the O_n's d.f.

→ A (very) dirty w. .20 ff: That TM has to do a lot of Irng before it Meta Irngs is good and to speed things up — it may take hours to get to the pt. at which MetaIrng "runs" the show

dynamics associates

00: 432.40: O.k.: So what has to be done on report? We have:

1. Abstract

2. "Introduction": Actually, up to 414.13 is kind of an introduction. Its really not an adequate introdn. I will add more stuff that discusses the various sections.

3. (New section.) The paper starts "for the data described (above) → (above in the introduction)

Derbs Gunl. soln using ALP

415.10 — on ∃ GA soln

415.11 L srch

415.20 The P_i^k; P_i eqtn.'s how predn. is done

418.27 Updating thru 419.00–.10 then 429.00–.04 430.02–.06 ends updating.

I have "Summary and discussion" 431.00 –.40; 432.00–.01 maybe do 432.03

α I have to write on TSQ, perhaps of some length.

β " " " " 432.10 : Discn. of first & second Level (machines: or just Meta Problems. (Things to include in report)
427.12 (things of importance to (perhaps) discuss.

I might actually send "first draft" of "Report" w.o. much on TSQ — just refer to S86, S89: Possibly quote a page or 2 — since both are in Latex. Give Mahnert! Put S86 on web.

I will also have to tell how Meta-Problems are designed. Perhaps give examples.

It would be well to show how ∃ any problem types can be represented to QATM.

Mention use of index "fields" in Q. — That learning to understand the domain of a problem is a good way to prepare TM to solve the problem.

γ) I need Section on L srch! — but rather simple, because only IAU (INV?)

probs are being solved. Perhaps discuss T ≤ kT, ll lnch, Monte Carlo ll srch.

δ) ~~When in near term~~ or in the introdn.
415.03 – .08 — This could be made much stronger! ∃ 431.03–.06
The "Guarantee" that we will get about the best possl. P'd for that particular (finite) subspace of $\{R_i A_i\}$.

ε) 2 IMPT Operations: (a) Merging of old R^i, P_i^k's to form newer fncts.
(b) Breaking down R^i's into new subcategories.
(a) Synthesis
(b) Analysis
In both cases we get pcot covgs down.

ι) 432.10 – .16; 427.12

Perhaps derb "TM working on context" as a TM calling on itself recursively, to work on meta probs.

Note Rest of Exp. I derbd in some detail is not recursive. Recursion needs (1) self reference Equs (2) Boundary values.

relevancing? relevant?

N5/01
ID

.00: 433.40 Levin's Universal Search.

ommit

There are two kinds of problems in our system, that are solvable, using L search. They occur in updating the ~~finding~~ recognition functions, $R^i(\cdot)$ and probability distributions, $P_i^k(\,|\,)$.

We will first describe these functions in some de

We will first describe Lsearch, then show how it can be used to update the system.

~~Occurring in updating the system~~

Suppose we are given a function in the form of a computing machine $M(\,)$.

M accepts as arguments, finite strings that form a prefix set.

The "prefix set" condition makes it possible for M to know when a string of symbols has terminated. The problem is to find an input to X so that after M has read X, it prints a and stops

$M(x) = a$. / Meaning that a (not italic) is a known, finite string. Many problems such as solving equations and proving theorems in mathematics / can be put into this form. ~~Solving equations, proving theorems,~~

If we are given a probability distribution $P(x)$, over possible solutions, x,

we can use this distribution to guide our search.

~~One way of doing L search starts with a certain time threshold, T.~~

$T \cdot P(x) > \epsilon$.

We would like to try $M(x)$ on all strings x,

But since $M(x)$ may take an infinite amount of time for certain x, we usually can't do this, so instead, we spend a maximum time of $T \cdot P(x)$ to test string x. If $M(x) = a$ the search is over.

The total time needed for this part of the search is $\sum_x T\,P(x) = T \sum_x P(x)$

Since $\sum_x P(x) \leq 1$ This round takes total time $< T$.

$\epsilon \leq \tau \leq T\,P(x)$

This round of t

If we haven't found a solution, we double T and go through the search again. This doubling and redoubling continues until we either find a solution or run out of time.

If a solution, x_0, takes time t_0 to test, then it is not difficult to show that the total time needed to find it this way will be about $\frac{2t_0}{P(x_0)}$.

If we have a good idea as to what the solution is before we do the search, we can put this idea into the probability distribution $P(\cdot)$, by assigning a relatively large value to $P(x_0)$, resulting in a relatively small value to our total search time; $2T_0/P(x_0)$.

Knowing an upper bound on how long L search will take, makes it easier to design training sequences for the system.

If $T\,P(x)$ is less than the time of one machine instruction, we don't test that x at this time.

P1200 125 96 | @433 150 bad, 300 combined

If $T\,P(x) <$ Time for one computer instruction, we don't test test x.

180 36 | CPM.ps 180 96 ok | 160 not bad | 180 good | 200 Good

There are two kinds of problems that occur in the updating process.

— Finding the recognition functions, $R^i(\cdot)$ and finding the probability distributions $P_i^q(\,|\,)$.

The criterion for acceptability of $R^i(\cdot)$ is that $R^i(Q_k) = \phi$ for a certain set of Q_k and that it is 1 for a certain Q_{n+1}

We can use the mechanism discussed in Appendix I, to assign probabilities to strings — so that we can readily employ L search to solve this problem.

Spec 438.00

N5/01 4/11/02: on t. diff of .00 ff: If we use L srch, for stoch Relations betw Q & A, we
ID May get a stoch soln. much earlier than a D soln. — because D soln. takes much cc.
T D-soln. may have better $\frac{cc}{pc}$. 435

dynamics associates

.00: 434.40 : SN Re: Search for P(A|Q): **PARADOX?**

I've been considering a 3 input UMC: input 1 decribes a function of inputs 2 & 3.
Input 2 is Q, input 3 is R.
If There is a function of Q w. [illegible], say, that gives A_0, then it should be possl to have
a function that is 3 bits longer (w. input 2 = Q_0) with $\ell(R)=0$, that gives same A_0.
(D; $\ell(R)=3$)
(→ May solve t. problem → 436.23)
& yield many $Q_0 \to A_0$'s.

.05 I.E. t. length of input 1 + length of input 3 can be constant, & yield many $Q_0 \to A_0$'s.

T. forgo argument doesn't hold if t. stoch funct has to work for > one Q_ℓ value.

.07 Well, it may be that for a single Q_ℓ, a D soln. is usually a best. — That in "merging" one
gets to S functs, & one gains much pc by "merging". My example in ANL in search are all
D functs. If .07 is true, I may have to revise this R^3, P_i model somewhat
.10 or a lot!

.11 Well actually, for a single Q_ℓ, it may always be true that a single code is by far t. best!
If one has $Q_\ell \equiv Q_m$ but $A_\ell \neq A_m$ then t. P has to be stochastic.
Merging QA's that to be D, can give stochastic P's.
A stoch S-P_i can be later broken up into "special cases" by proper Now Recognizers
— into several D-Ps &/o S-P's. This sort of thing occurs in normal Science!
Finding what was that to be a single disease, is best regarded as several differt. types of disease.
A set of Q_i's that are not identical, but are "expensive to recognize as" differt, can be
Defi regarded as identical & perhaps give a **S-P.** (Stochastic P notion.)
So t. problem is, **how to do updating!**

.20 There is that other model of stoch operators, that I didn't like so much!, A 2 input (rather
than 3 input) UMC. 2 inputs: 1) desc of function 2) Q_ℓ. We had many codes
(# input) for a given output. 414.18 414.31
Trouble is, I that I had a micro proof that 414.$^{18}_{31}$ was correct!
Could I devise a ν proof for t. 2 input umc? At first I found a counter example to 2 input
model — then found t. counter example was wrong!
T. 2 input system does give a pd for each Q_i input, so the equs of 414.$^{31}_{18}$ apply: t. Q13
Is a 2 input model universal? Equivty, If one are allowed to use any [illegible] input Machine...
How can it simulate a pd. ?
~~I suspect that t. 2 input~~

T. 2

.30 253.21-.24 has "counter example" to 2 input Model: 267.00$^{-.09}$ shows t. counterexample wrong.
Hvr, note objection 267.10-.11. — Th. whole thing (2 input model) started at 250.04-.14
Hvr, as far as I can see t. 3 input would work. — & is provable:

Any way: Assuming that I'm using 3 input model, it seems that for a single Q_ℓ,
there will (perhaps always), be a min code for it. i.e. $F(Q_\ell, R) = A_\ell$; we can always find an
$f \ni R = \Lambda$: so $F(Q_\ell) = A_\ell$ & F has a short code.

W. 2 input model, t. predn. is obtained as a wtd sum of all of t. codes: A legit code has
to code t. entire corpus, giving correct answer each time, & each code has a certain "Length".
This corresponds to t. Sequential coding soln of say, 578T3. — regular ALP. — Each code has to

dynamics associates

.00: 435.40 : code each bit of t. corpus exactly. What if $Q_\ell = Q_k$ & $A_\ell \neq A_k$? (.00 R) Clearly t. system as desrbd fails!! (.07)
T. codes are for a Q → A function, so t. codes form a prefix set — like in Li-Vit's Universal Discrete distribn.
T. 3 input code certainly deals nicely w. t. dif'ty of (.00 R)

.04: 432.16 [SN] Do have discn. about how t. QA form of induction is fairly universal — how one can easily "teach" what $\int f(x)$ means, after $\frac{d}{dx} f(x)$ has been lrnd.

(.07) "T. SYSTEM as desrbd is: 2 input Umc: $\text{Input}_2 = Q_i$; $\text{Input}_1 = P_{prmi} = z_i$ | Both inputs prefix sets.
→ t. approp'd assigned to an unorders set of $Q_i A_i$ pairs is $\sum_z 2^{-\ell(z_i)}$ z_i is t. prefix set of dcrns of [redacted] functs betw. Q_i & A_i such that the A_i's are correct for all of their Q_i's

.10 So: T. problem now, is ≈ [435.00] dif'ty: But mainly, 435.11 is t problem: for a say R^i w. a single Q_ℓ recognized, t. best model will be a 'D-function'. (Determinstc, MTM).
After a new Q_ℓ arrives w. same R^i, one can try merging.
The (common) exception is that a QA may be a D. funct. (as in MaPa).
It may be possl. to work
"So t. real Q is how to update?" I mite first look at pure MTM case.

these problems w. th. present update rules: [407.03–.14] ≈ 419.00–.33

Going over 407.03–.14: Condition (1): 1 & only 1 [redacted] $R^i \ni R^i(Q_{n+1}) = 1$ — PC of A_{n+1} given Q_{n+1} is "Hyankr".
[redacted] t. condition of acceptance is t. have, that t. PC for A_{n+1} " is not large enuf. 2(a) is ok; 2(b) is much perfered if possl.
O.K.
.20 cond (2) same as 1 but [redacted] — The approach given is O.k. for MTM: for NMTM, it → 437.00
.21 cond (3) $R^i(Q_{n+1}) = 0$ for all i. Its predns will tend to be not so hot.
may not do harm, but its not yet of much use. ... → [S: Q_i, R] (1, 2, 3) "I think"
.22 w. a 3 input umc, there will be

.23: 435.05 [SN] When we have a really New Q_{n+1}, we note 435.00 –.05 : if we do find >> 1 S, R pair many [S, R] pairs that [redacted] have ≈ same total leng R_i, so we get some extrapolation. So this then we have a good chance of $R \neq \Lambda$, so we get some extrapolation. So this May solve t. dif'ty of 435.00–.05 ! Hvr, I'm not certain that [redacted] soln. of t. N=1 ssz=1 case is usually helpful for t. [redacted] N>1 cases — i.e. it extrapolates at all well!
I may be able to do a lot of lrnng of MTM types using not so clever heurs — Sort of "Minimal Heurs"; But to get even lrnng during Lsrch" t. system has to
.30 be able to do NMTM!

I usd to think that Induction could be usefully done w. ssz of 0, 1, or 2: Certainly t. formulae of 414.18 (.31) enable this. But my R, P formalism may not be so good for it!
Well, maybe that's O.k. — That at first TM may need ssz of 2 or more for problstc induction — but eventually, it will break out of. t. R, P restriction & be able to do more "advanced" kinds of induction.

[So] I want to know if t. MTM problems would be ≈ solved by t. "3 steps of update"!
Also, some ideas of how to deal w. NMTM. Also if not how to deal w. MTM!
for NMTM, Think of some simple ssz = 0, 1, 2, 3 ... cases.

II

(spec 436.22) (spec)

.00 : (436.22): on update rule ③ [crossed out] $\forall i: R^i(Q_{n+1})=0$; For MTM: O.K. I Guess,

:436.22: for NMTM! Unclear: I do have to rethink induction for use of 0, 1, 2, 3 ... . 414.18 .31 is prob correct, but how to approx it in a good way?

For $S\Sigma \equiv N=\phi$ we just make pc of A depend on how easy it is to construct A using available concs.
For $\Sigma\Sigma$ $N \geq 0$ we do same thing, but the thing we have to construct is $A_0, A_1, \cdots A_{N-1} \cdots A_N$. (construct using some available parts"). One might think that $N>1$ makes it possi. to invent new concs';
— But no new concs can be invented as part of a larger function used to express A_0.

This is done as in SAZ (= Augmented Z (41)). I think its essentially what update 2b does.

Update 4: Several R^i accept Q_{n+1}, but one is best for A_{n+1} & is good enuf.
Modify R^i's so that only best & good R^i accepts Q_{n+1}. Seems O.K. for NMTM. = STM | vs. DTM = MTM

Dads [crossed out] It is possi. to do 4a & 4b: 4b is same as 5b & involves Merging
Note that in general, [crossed out] merging of all accepting R^i's may not be feasible/desirable —

.10 but try [crossed out] to merge as many as possible — but may be only 2 are feasible.

.11 N.B. in condition ① (407.03-.14) of update! Every thing works fine — apparently "no update needed"
False! When Q_{n+1} comes in, it is added to the corpus: At minimum, "continuous" params have to be updated,
since no. of cases ↑ by 1. Another possy is that the freq. of use of 1 or more concs goes "over threshold"
so we are able to legitimately define a new conc. (thus ↑ pc).
This kind of update is very fast, hvr, & doesn't involve Lsrch.

.16

Ghostscript
DLL c:\Ghostgum\gs0.01\bin\gsdll32.dll
it's in c:\Ghostgum\gsview
is 271kBy 3/20/00

Do pg no to an CPM (condit probability [crossed out] Machine) C & Learner.

(1) (2)
1) for [crossed out] 414.18 [crossed out] & .31 have eqs nos

⇒ for these eqns, the Σ should not be summed to ∞ — just "j"

.20 The summation is expained in text. later these eqns are used w. for these sums,
Is it made clear later that when we are using these eqns — we are "Maxzing" them by not summing to ∞?
This must be made clear. Say in subsqt.

SN At all "update" times, the goal is to Maxz 414.18. Hvr, with the limited time Available,
there is an ordered list of tasks. First to do the 5 update things but do the "a" parts
which are quicker. If time remains, do "b" parts. Maple Staple Ecad Abcdefghijkl

.26 : 431.20 : in spite of [crossed out] Educational experience that is not at all well designed. [crossed out] that are not [crossed out]. This might be explained by considering that people start out with a lot of "prewired" induction techniques based on many millions of years of

.30 genetic selection. The system being proposed is expected to become much less dependent

.31 on training sequences after it has learned certain important induction techniques. [crossed out] → 431.29

[redacted]

dynamics associates

(spec 43140):

.00: To understand the application of Lsearch to probabilistic $P_i^j(\cdot|\cdot)$ functions, I will first consider ways to represent them, using universal Turing machines. The discussion readily generalizes to other methods of representing functions or binary strings.

Consider a 2 input machine: Both inputs are prefix sets. Input$_1$ is Q; input$_2$ is R. The R inputs are a complete prefix set, so $\sum 2^{-\ell(k_i)} = 0$. (k_i is i^{th} word),

e.g.: 1, 01, 001, 0001, . . . $\sum \frac{1}{2} - \frac{1}{4} + \frac{1}{8} \cdots = 1$.

Hmm, this prefix set could not represent certain P.d.'s. e.g. it couldn't represent $\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}$.

.08 — Since all other prefix sets would have to be used and 1 (w. $P(\cdot) = \frac{1}{2}$) could not be 01 (could not..
Another/ constant prefix set is the set of all binary strings of length n. If n is large, this can represent many distributions

.10 Unless I use .08 w. large n, this will not work! Each Q_i needs its own prefix set.
If the machine reads Q_i, then reads as much of R as it wants, then starts to print A_z, then the inputs to R must be a pref. set; one input can't be pref. of another.

How to get Lsuch (or AZ141) to generate bivariate functions, is unclear.

For Q_i, may be a binary string w. a "blank" end symbol — to this defines the function on R".
If R is unidirectional, and we acknowledge "Q" & its outputs when the machine stops, they "accessible R's" are a pref. set.

.17 → I could write the paper using the Trmc formalism.
· T. Trmc instructions could read S, then read Q, then read R, then output.
· T. insts could be arranged so it does this. As soon as M reads a Q bit, it can't read any more R "; any more Q "

— I could generate any function of 2 variable (Q & R) via AZ141.

.20 Actually, for testing, it would be best to have once read Q first, then S & R.
→ except if I wanted to do several Q's with same S", so probably do S first.

.23 So I could construct "s" via AZ141. — But I'd like to get AZ1 to generate functions of 2 variables.
Well, if there are only 2 objects to be args (in addition to 0, 1, π, e, etc), then that's it!
So in AZ1, I generate a seq. of functions; each funct def. can use previously defined functs.
Any funct can have 0, 1, π, e, Q, R as 1 or more arguments, — or previously defined functs.
Since Q is finite & well defined — no problem w. that argument. for R, it's not clear as to how the AZ1 function should deal w. it. Perhaps have the AZ1 function include a "no. of bits" integer, which it asks for the value of. R.

.28 Using the formalism of the "Sept 8, 2001" postscripted notes, I may want to allow Q_i & R to be one of the "constants" — so they can occur anywhere in a funct. final or intermediate function.

.30 When we end up defining a funct of 3 variables, then R & Q's ok. — the choice of values for each of the 3 slots is still 0, 1, π, e, Q, R.

.31 A possible trick: T. no. of bits read from R is a funct. of Q only. — This means that for each Q we have a fixed resolution of P & a minimum value (other than zero) — Min is 2^{-n}; in terms of length of R.

Ideally, I want the decision to read the next bit of R to be a funct of Q & all previously read bits of R. → 447. 22 - .23 talks how to do this

Perhaps Read about Chaitin's Micro lisp to get ideas on how to do this.

My AZ1 is a rather limited device anyway (no recursion): I could have a sequential reading of R. $F(Q, R_t) = 0$ or 1; if it is 1, R_{t+1} is generated. If not, R is of length t. So F has to have 2 outputs ① and/or ② A.

Schönhage
≈ Variable Storage Machines.

Schönhage:
≈ Variable Storage Machines

dynamics associates

.00 : ₹38.40: Getting back to Lsrch! Wait we want to Search over $\ell(S)+\ell(R)=$ constant.

So We present U. function w. Q a R sequence of length n. It has already ~~Reject R~~ ~~dealt~~ dealt w. all R trials of length $<n$. If R is that contains a tried prefix, then that R is rejected. We can arrange so that if we give U. function a R of length n, then only R like the prefixes that were required. T. function ~~used~~ used not recognize this n bit R, ——

but it may. It could express its "non-recognition" by not stopping, or ~~not stopping~~ I.e. stop w. Λ output or not stop w. ~~Λ~~ Λ or any output.

That might be O.K! say X is a ~~string~~ R string that has no prefix that caused Funct to give acceptable output (all prefixes have been ~~tried~~ tried, hmm). We Run try X. If it produces legal output, Great, no extension of X is used as R. If no useable output occurs, we try X0, & X1 for R on next Lsrch "round".

Will this ~~work~~ work? remember "No legal output" means loop times $> \equiv T\cdot 2^{-n}$

.10 On next round, Loop times will be $2T\cdot 2^{-n-1}$; — The same! No! S can change in long trials so.

So, its not clear!

Oh! each round (= Value of T") we have a new ~~same~~ prefix set! — ~~but~~ and it is = prefix set!

So, a big Q is! Can we do this using || Lsrch (using randomly chosen trials & perhaps)

In || Lsrch, as soon as a trial terminates, it immediately cuts off all R trials that are extensions of it! — So that would do it!

In || Lsrch, if after finding a successful trial, we could still have prefixes of that trial that haven't yet converged — so we don't know about whether they will or not ——

But still, for a given $<B$, we do do find a prefix set. (& this is true in U. T<2T Lsrch also). So perhaps, for each $<B$, ~~we~~ since we have a different prefix set, we also have different predictions!

For Lsrch, for a given S function & a set of parameters we should only try R values of all U. same length — so we'd have to remember which are legal (i.e. have not completed prefixes)

It looks like Lsrch may not be so easy w. this AIT function test!

.25 SN on TSQ: ~~Giving~~ Giving "Indexed" types of "Hints" for Q = 3+5; 7+1, 8+9 each, could be a strong "Hint" In general, ~~this~~ indexing can be a strong kind of "Hint".

.27 The T→2T method seems to work O.K. w.r.t AIT apriori for S functions.

We list the S functions in ~~in~~ apriori order. We then do an exhaustive search over R values, starting w. smallest $\ell(R)=1$, Run $\ell(R)=2$ then 3 . . . etc. We reject R values that don't converge in $<$ Limit $<B = T\,2^{-\ell(R)-\ell(S)}$. So for each S function there are a prefix set of R's.

.30 (Will keeping track of the R's that have converged, be an excessive ability) — At each pt. in time for ~~We will have a list~~ a given T, S, we will have a ~~list~~ value of $\ell(R)$ that we are working on! Then $2^{\ell(R)}$ possible prefixes will be divided into legal & illegal prefixes: We try to continue on the legal prefixes. Then illegal prefixes/given a prefix test has already converged.

It may be possible to keep track of this very cheaply! Look at U. exploration trees for R.

U. routine may be same or identical to U. search that I did in ~~1984 or so~~ U. 1984 report I wrote. (~~Abridged~~ Abridged report.)

In fact, it may be identical! ⇒ "Optimal Sequential Search" was the Title. "OSS 1985"

→ (440.00

dynamics associates

.00: 439.40 : Re: parallel Lsrch: We don't have to do this, because we aren't doing OZ problems yet in lsrch.

(tho. I may end up needing it for that!). On the other hand, || lsrch is more effective — by a factor of 2 (at least) & (probably possibly) by a factor of the search depth.

In || Lsrch, we store all partially run strings & their assoc "machine states".

[redacted] — so it looks like it would be possl. to do this Lsrch, since we will know which [redacted] trials have converged.

No! → Its beginning to look like || lsrch w. S & R will not be any distinct from || Lsrch w. R alone, — (i.e. regular Lsrch). ~~In both cases, we explore all~~

.10 Consider just Lsrch on R alone: In the present situation, say [redacted] R = x was doesn't converge in time [redacted] t; but it is possl. for X1 to converge in time << t!. — This would not be true in Usual Lsrch using unidirectional IO wires or suitable "prefix machines". In the present case we always "try" x before trying X1. "Try" means a complete trial: either it converges or "times out" using present cb. The the prefix set differs for each ~~round~~ "Round", the time to do each round is ≤ T for that round.

For [redacted] || Lsrch on R alone: it's not obvious how to proceede. proceede.

.19 Suppose for fixed S, the (convergent) R do not form a prefix set? 440.19 R They have to: If a R input converges, we don't "try" its extensions.

.20 — Well, in T←2T, they do for each T value!

.21 N10/01 [redacted] Couple of Q's: ① Say S, Q has outputs in short time for most all short inputs (R).

.22 So R=0, R=1 bring me our pets & that's it.! Each have pc = ½. Only 2 Ai's

② I deadly wish'd like an R to exist for every possl A_i!

Actually, (.22 ①) isn't nearly so smart! the S, Q function has to end after the R input

BUT Many functions will do that.

.26 I really haven't done any stochastic TSP's. Perhaps try some Elementary Item Sequs; we index the data w. the name of the TSP so TM can tell they belong together.

I've certainly written a lot about this stoch inductive problem.

.29 • Another kind of stoch induction: [redacted]

.30 A common type of stoch funct.'s a table: for each Q, we have probbys of several different A's.

Hvr, using a UMC w. 3 inputs it's a provably adequate soln to the stoch TM problem.

I guess what I don't like is — I don't see how I can fit 441 into it.

Hvr, I'd like to be able to show how it would work for the learning (.26) & the table of pc's (.29).

In .26 (TLU), we can make separate studies for Q=a & Q=b.

I am reminded of that Guy who tweeded ≈ curve fitting using a ALP! He did a discrete model Search, using a MDL, then [redacted] some continuous optimizer methods to fit the continuous params

This is only a small part of Inductive modeling — it is an important one. [One has to take the Hessian (expect the final pc, hvr.]

→ 441.21 space

dynamics associates

.00: 440.40: superficially, the problem is: There are 2 ways to represent S trees:

1) 3-in pot umc

2) AZ141.

1) is formally fine, but doesn't seem able to model hours of import.

.05 2) Freq. of word occs.

.06 b) looking for common sub-trees.

.07 2) is able to do .05 & .06, but I haven't been able to get it to do Lsrch properly. — T: apparent reason for Phil's (13)

5/11 Looking at "Opt. Seq. Srch.": Comments on Follow:

Remarks & Q's by Eric Hamfeldt: Very Good! I really have to answer these Q's!

.10 Also, that "OSS" paper did seem to have a (general) Model for the Lsrch. Lsrch

(Reading Wind's remarks, suggests that there are some Holes in my understanding of Lsrch!

Mainly Re: OZ Lsrch, hr.

.13 (.07) † funct of AZ1 (as of present), simply take prefix code words as inputs. This appears to be because of the structure of the functions it can create. Re: Umes that accept only prefix code words: The Army I Phil unique about them was that their decision to take another bit, was based on what they had read thus far. Actually, any device that accepts its news how to (seems to solve † problem) accept as code words from a prefix set, can do the same thing. → see 447.22 – .23 for How to Do Thig

So, in AZ141, each function must be able to look at any infinite string of bits, & cut it into code words (if the code words form a complete set.)

.20

.21 (440.40 spec) Another kind of "simple" [redacted] S. induction problem: A single Q is a long sequence of symbols (or of numbers). The TSQ consists of $X_{0|i}$ | X_{i+1} for $i = 1$ to $n+1$. [redacted]

Q_i | A_i

$X_{0|i}$ means † seq $X_0 X_1 \cdots X_i$. % Hoffer.

if the X_i are reals [redacted]: † conds will be Pd's on reals that are functions of † nos. in Q_i. (or an arbitrary no. of nos. in Q_i).

Now if we have a function on $-\infty < z < +\infty$ (and † nos. in Q_i) that is ≥0 and, under Q_i, $\int_{-\infty}^{+\infty} f(z)\,dz = 1$, then it is a cond. for a prob. distribution.

We could, of course take any function & square it to get it ≥0, then normalize it on $(-\infty, +\infty)$ interval. (if it were normalizable). ≠ The "squaring" part seems bad: In general we want † funct to be more "naturally" pd's: like † pd's we get in ALP. → 447.24

.30

.31

.32 One way to get nice, easy prefix sets: use space symbol (=2; so 0;1,2 are 3 symbols) "2" terminates a word. So we have binary words separated by spaces (=2). † system is complete because any (binary) string can be uniquely parsed into words.

$\sum 3^{-\ell(x_i)} =$

word	pc	
2	$\frac{1}{3}$	$\frac{1}{3} \cdot 1$
02	$\frac{1}{9}$	$\frac{1}{3} \cdot \frac{2}{3}$
12	$\frac{1}{9}$	
012	$\frac{1}{27}$	$\frac{1}{3} \cdot \frac{4}{9} = \frac{1}{3}\left(\frac{2}{3}\right)^2$
102		
002		
112		

$\sum_{i=1}^{\infty} 2^i 3^{-i-1} = \sum \left(\frac{2}{3}\right)^i \cdot \frac{1}{3} = \frac{1}{1-\frac{2}{3}} \cdot \frac{1}{3} = 1.$ So it works as expected.

.34

.36 Since smallest word in pc = $\frac{1}{3}$

.37 we cannot represent † pd. $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.

See 442.37 for a kind of soln. — leave out "$\frac{1}{3}$" – use incomplete prefix set!

N/11/01 ID

.00 : As noted before: 441.36-.37 is a necessary difty in any system using the same prefix code for all Q_j's. Somehow, Q_i has to be able to close the prefix set for representing pc's. — Or, more exactly (P_i^k) " " " " "close" " " " representing pc's,

Perhaps look at Cover's demo, that any PD can be represented by a suitable Turing.

Is it very much like Arith. coding? In arith coding, one feeds in an infinite binary seq, & every once in a while a bit of the coded sequence emerges — which is a kind of "parsing" of the input string.

For binary data, the P.d. Always has at least 1 pc > .5, so the argt of 441.36-.37 doesn't apply)

Also note that if we use a Basic binary system (like almost all digital computers), then if we use words of several bits (not necly all same size), then we can always assign pc's to individual bits of all A_i's. Hvr, Arith coding gives just 1 code for each A_i — and it's a bit erratic — sometimes no output for a long window input seq., then a long seq. of output symbols!

.10 What about Hufman Coding? It is certainly not v.g. for pc's close to 1! — & I will be having lots of d-funcs to lrn. — which have pc's ≈ 1.

Write up detailed reasons why the 3 input model & the AZ(?) model are inadequate So an external person (or later me) can work on it.

List possl. approaches, alternative models, approaches.

Also the method of 250.04-.19 : Counterexample 253.21-.24 ← (Counter-example wrong 267.00) Now Counter Example 267.00: Actually 267.00 gives instructions for realizing the D.P. of 250.04-.19!

.16 The Idea is: For each i ($i=1|n$), we have a set of possl A_i^j. Pick some large integer Z. Then we have Z $S^r(i)$ functions: $S_i = A_i^{k(i)}$: The probability that $k(i) = j$ is the probability of A_i^j being the/a correct answer.

A way to get extrapoln. out of such $S^r(i)$ functs : $S^0(i)$ gets the most likely choice for each A_i.

For each value of i, we have (0,1) divided into Z = divisions (0.........1). We place the A_i^j probabilitys

.20 over that interval (they sum to 1). We put the most likely at left & in linear order of pc; least likely on left. $S^r(i)$ will have that A_i^j that occurs over the r^{th} point: it is at $pc = \frac{r}{Z}$.

So, in general, the $S^r(\cdot)$ functions are in order of r, {r=0 is most likely}

So, in general, the $S^r(i)$ functions that will have values A_i^j will be ≈ Prob of A_i^j.

The number of $S^r(i)$ functions

.25 O.K., So say we have a lot of S^r functions. Each one gives A_i as a funct of Q_i.

No one S^r funct can assign much of a pc to the corpus. Also, if the S functions are expressed in $R^i P_i^k$ form, it's easy to get lots of them: If we have "a" values of k for each P_i^k ($i=1|n$) then that's a^n difrnt S^r functions.

If the S^r functs (or the P_i^n functs) all have difrnt pc's (≡ wts.), we will usually need more of them to get

.30 good pc's than if they had all = wts.

So it looks like .25 might work! (it is same as 250.04-.19), 253.37,

Couple of Q's ① Is the method Universal? ② Can we easily express known pd's using the ③ can we express certain impt. heurs using it?

.34 ⦿ → On first thot, this seems like a very hy. cost method of deriving S-functs. — It is necy to derive all of the S^r functs: the total deriv length of all of them is the final deriv length!

Hvr, many have ≈ derivs, so Σ cost is not as bad as that! (Tho it's still pretty Bad!)

.37 HA! Maybe a way around the objection of 441.36-.37 : say pc's are not measure but semi measure — so we don't have to use the "$U_{\frac{1}{3}}$" word if we don't want to! — we do have to normalize, hvr., since we'd be using an incomplete prefix set. A big trouble is normalization! We have to get pc's of

dynamics associates

.00: 442.40: All of hy wt. A's! Maybe estimate it: try ~~WWWW~~ vary by wt. R's: The ones w. no output; we can add up their pc's & get ~~mmm~~ ≈ normzn. constant.

So b. code of 442.32 may be 0.4. Binary strings w. space (or 2) for end of ~~w~~ word.

.05 OR maybe don't normalize (except perhaps ~~for~~ wrt. R's that don't converge)

If a none of the A's have large probys .... Well, $\sum_j p(A_i^j) = 1$ i.e. ~~some~~ some output must occur!

~~[redacted]~~ So what is response to space, (pc $\frac{1}{2}$) when no A has pc > .25!

Well, say more we only 4 outputs A's w/ all had = pc.

Is it possl. for the function to have "no output" — is this distinct from Λ (≡ null string)?

Say the ~~R~~ R's have 4 A's where 00v; 01v, 10v; 11v; All other inputs (including v) give no output. So this would actually mean that no A = no output would be rather likely!

.10 Only if we know this is the case!, can we "normalize it out". (.27)

Well, in all cases of update, we know that an output A must ~~be~~ occur. — so ~~[redacted]~~

We'd be justified in "normalizing out" the nulls.

Still we have to put in a lot of inputs, to get the normzn constant!

$\sum 2^i 3^{-i-1} = \frac{1}{3} \sum_{i=0} \frac{2^i}{3^i} = \infty$ | $\sum_{i=0}^{n} a^i = 1 + a^1 + a^2 \cdots a^n = \frac{1-a^{-n}}{1-a}$

$\frac{1}{3}\left(\frac{1}{1-\frac{2}{3}}\right) \cdot 1 - 2^{-n} = 1 - \left(\frac{2}{3}\right)^n$:

n	1	2	3	4	5	6	7
$\frac{2}{3}^n$	.667	.444	.296	.197	.13	.087	.058

so, to get .058 accuracy, we have to go out to length 7:

$2^7 = 128$; so ~ 256 trials. — But usually we don't need such precision!

.20 ~~[redacted]~~ 29% error would be eff. length of 3 :: $2^{3+1} = 16$ trials.

~~[redacted]~~ — Superficially, in R, the symbols (& bits) all pc = $\frac{1}{2}$ — but actually,

Note so! The pc's of 0's & 1's are about the same, but pc of v is ~~[redacted]~~ much less!

Consider the sequences of R's in a corpus code. The freq. of v will be $\ll \frac{1}{2}$.

So we can legitimately recode the R's.

.27 (.10) Another Q: Pr. single symbol R = v, can be a legit ~~[redacted]~~ arg for Lisp functions.

Hm. If an arg has to be a number, v ~~[redacted]~~ $\Lambda \neq \phi$ at all. Λ would not be a legit number.

Unless, a string means (as a number) "~~[redacted]~~ clear the register; Now load the string, bit by bit."

.30 If there were no bits in the string, result is ~~[redacted]~~ zero.

[How is "zero" expressed in floating pt? It can't be (ϕ, ϕ) since "ϕ" means 1ϕ ("significant digits")

Maybe ϕ, -511 (the smallest possl. no.).

.34 → O.K., so if there are no A's w. pc $\geq \frac{1}{2}$, then for the v value of R, there is no output — which means "undefined" — the ~~system does not~~ machine does not stop ever, if that pat occurs.

(Or maybe it doesn't start? — No, it has to "start" to read Cv.) Well, if it read a binary string of length w, followed by v,

It should look for A's w. pc's $\geq 2^{-w} \cdot p(v)$ — it may find none. ∴ it ~~[redacted]~~. Doesn't stop.

N/12/01
ID

dynamics associates

-00: 445.40: How to code a set of (pc_i's, A.), using a prefix set of (pc_j's; PR)

Put the A, pc's in order of size, largest first;
Put the PR " " " " " "

.03 Probably get A pc: find largest R pc that is < to it & assign it to the A. [α] Take the remainder of the A pc & insert it
.04 into the A pc list in its proper order. Go to the next pc in A pc list, & loop to α. Will this eventually use up the [redacted] A pc's? If there are some PR pc's that are > any A pc, they will never be used, & so the total pc of A (which is 1) can't all be filled in.
Also, if the PR pc's are of finite cardinality, & there are more A pc's than that we can't do it either.
So say the PR pc is of [redacted] Card = ∞.

Even if .03–.04 Doesn't work, it may not be Bad! ... May be Usable!

.09 [R 1/13/01] **Anyway** 443.34 suggests that we can put any finite binary string into the Lisp function.
-10 It can then select a subset of these which it regards as "defined" — for others, it does not stop.
Trouble is "binary strings $\cup$" is a complete prefix set. Any proper subset is incomplete.

Say the function selected only 1ω, 0ω as defined. In one sense, this is a complete prefix set. — In another, it is not (i.e. the "$\cup$" is excluded).
439.27–.40 seems o.k. (w. present ideas of "defined"), but note 440.10→.20: However, note recent note on (440.19R)
So the legal R do form a prefix set. But say x0101 was a pgm that gave a soln. to an INV problem. Suppose x converged. {? Meaning?
.15
.16 We would never ever try x0101 ! | In our present problem, if this were true, then the function S, must be discarded! — Fine.

On the other hand: The pgm on P9 of OSS 1985 is also **Unclear to me now!**

I seem to have forgotten how Lsrch works! The Lsrch is over pgms that are able to produce the soln: $p = H^{\alpha}(\alpha, M, x)$ (M, x: problem defn.); the "Test" is: IS $M(P) = x$ true?
-20 $[\alpha]$ form a prefix set. The p's tested need not form a prefix set.

We are looking for R's $\ni$ $F(Q_i, R) = A_i$. In this case, I think the [R] form a prefix set.
We want short R's. — So this problem is really easier than normal Lsrch.

If $F(\ )$ is a TM, we can easily get [R] to be a prefix set. — we feed in R sequentially, when the machine stops — that's a member of [R] on input.

Look at Chaitin's **"Little Lisp"** for (perhaps) how to do this. ["Pure Lisp" has no "side Effects"]
(I'm looking at a talk Ch. gave at Santa Fe inst. ~ 2006 ?; he starts talking about bit string input.
Chaitin TLisp1.html 47K 11/8/00 on PP200 Chaitin folder on G: or F:
1) read next bit 2) read next (legal) s-expression.

Lisp: Chaitin TLisp1.html, this is a simple intro to Chaitin's Tiny Lisp. Chaitin TLisp1.html is a Santa Fe talk; How this differs from chain 3, 2 Chaitin is unclear.

"Try" can fail via CB or via end of bit string.

[I have his Lisp interpreter in Mathematica. (Lisp.m

Ch. uses stack length rather than Time as CB.
-30 (Try, (Time limit; CB), s-expression, binary data) : when it → CB; output is (value | CB | how much data used? | displays)

McCarthy's Initial Defn of Lisp (In Chaitin Folder PP200), seems very readable.

.32 ⊕→ .15–.16 was a serious Objection to my model, but (.16 R) answers it! ←

For a Given CB, we will have a tree containing a prefix set at terminals. As we ↑ CB the size of tree will never ↑ & will usually ↓.

However, say we are in this tree w. a given CB & we insert 1, 1, 1 ... — & none of them converge. We will spend a total time of $T \cdot (2^{-1} + 2^{-2} + 2^{-3} \ldots) \approx T$ on that branch alone.
On each such branch we could spend ~ T. Well no, we spend $T \cdot 2^{-k}$, where k is the length of R at which the infinite branch starts. So Q is $\leq 2^{-k_i} < 1$?
k_i is the length at which the i-th infinite branch starts. ⟶ See 445.02 → 445.00

dynamics associates

.00 : 44440 : My Letter to Laurn dealt with the problem of a UMC that did not have unidirectional I.O. Its methods should be applicable to present problem.

.02 : 444.40 : In worst case scenario : S ~~never converges~~ for any R : $\sum 2^{-k_i} = N$, $N = \log_2 T$
space
(Here we assume that T starts at "1" which is time for 1 instruction so $N = \log_2 T - \log_2 (T_0)$ $T_0 \equiv$ time for 1 instruction.
.04 This may not be enormous, but it is $>> 1$. (!) .

- So look at (00)↑

.06 Even if the Machine 444.32 −.40 works, it would be very inefficient — it would rarely decid. functions that ~~would~~ only would accept/be provided x set as input. Even if we had a good way to detect such functions, it would be inefficient.

On the other hand, the input UMC automatically accepts only prefix sets. —

.10 Its like Koza using Lisp expressions & crossing them over to get increasingly fit expressions
.11 every time.

Well one way to do it would be by Chaitin's method: to devise a language that would have effectively unidirectional I.O. — for all inputs. Actually, the functions at first ~~2~~ inputs seem under control — only the 3rd input (+ R input) tends to be fixed up.

Well Chaitin had it so that a try could end if CB was exceeded or no. of bits requested was exceeded.
In the latter case, one repeats bit string with extra bit (This is very inefficient)

Another tack: See how I do induction problems, & try to put it into the 414.18 formalism of "Hypothesis of Corpus"

.20 All I need is a method to get various R values various A values
One way is ~~the~~ stack the w. probabilistic productions. In this case, we are asked for a (usually finite) no. of choices: each w. its own ~~pc~~ pc distribn. The functions AZ(4) are generated this way. —
& after each time is generated, we have choice of a "stop" symbol (i.e. final function is defined),
.24 we can derive strings in this way.

We could continue derivation of the S function & w. known input (eg. Q). Actually, one or more of the inputs is normally part of the function definition!

Say "Con" (≡ concat) is one of the (perhaps primitive) functions. con(0,1) → 01 con(0,001) = 0001.

so con(x1 (con x2 (con x3, x4))) = x1 x2 x3 x4 (say x_i are 0 or 1).

F(Q, x1 x2 x3 x4) would be F(Q, con(x1,con(x2,(con x3, x4))))
.30 Which is the code of a function. if it is a commonly occurring function (form), it could
.32 have pc ~ $2^{-4} \cdot p(\text{end symbol})$ — which looks like 44.32. If the p(end symbol) is ∈ (small)
then ∈ will be the smallest that the max pc of an A can be. — so should get ≤ A's of = pc's,
.34 they could all have pc = ∈. But if we had > $\frac{1}{\epsilon}$ A's of = pc we couldn't do it.

.35 $\left(\frac{1}{2} - \frac{\epsilon}{2}\right)\left(\frac{1}{2} - \frac{\epsilon}{2}\right) \in$ $\epsilon \cdot \sum_{i=0}^{\infty} \left(\frac{1}{2} - \frac{\epsilon}{2}\right)^i$ $\frac{1}{1 - \frac{1}{2} + \frac{\epsilon}{2}} = \frac{1}{\frac{1}{2} + \frac{\epsilon}{2}}$ $\sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = 1 + \frac{1}{2} + \dots = 2$

$= \frac{2}{1+\epsilon}$ → $\frac{2\epsilon}{1+\epsilon} \to 2\epsilon$ as $\epsilon \to 0$.

$\epsilon = \frac{1}{3}$ $\frac{2 \cdot \frac{1}{3}}{1 + \frac{1}{3}} = \frac{2}{3 \cdot \frac{4}{3}} = \frac{1}{2}$!

$\epsilon \sum 2^i \left(\frac{1}{2} - \frac{\epsilon}{2}\right)^2 = \epsilon \sum (1-\epsilon)^i = \frac{\epsilon}{1-(1-\epsilon)} = 1$.

good!

So if ∈ is small, it works fine, but then we tend to have lots of binary strings $\binom{?}{?}$ $\left(\frac{1}{2}(1-\epsilon)\right)^n = \left(\frac{1}{2}\right)^n e^{-n\epsilon}$.

.00: **445.40: Re:** T. trouble of 445.32-.34. T. case in which All A's have = pc & there are many of them — **is a Bad production Situation!**

If we only had these 0, 1, ∈ symbols (∈ = end symbol = pc of ∈), we would run into some trouble, but it may be usually not very bad.

But t. fact that this thing doesn't not work exactly is worrysome — I'm afraid it may have some other very bad properties!

.05 A possl. way to do it: Assoc. w. Q are 2 functions! F_1 looks at (partial R input) & decides whether it wants another bit or not., After R has ended F_2 operates on Q & R to obtain A.

Essentially, what f_1 does, is create a prefix set on t. R input.

This would seem to make finding a F_1, F_2 pair that works, have much more r cost than just t. simpler single function! But maybe not: we may be able to find a set of f_1 functs

.10 that covers all cases: say t. $(\frac{1}{2}-\frac{\epsilon}{2})(\frac{1}{2}-\frac{\epsilon}{2})\,\epsilon$ functs of 443.35ff: By choosing ∈, we are able to fit a greater variety of p.d.'s.

We mite do t. coding of t. A's using 0, 1, ∈, then optimize th. ∈ value for that particular Q → 442.00

.14 Another aspect of .05ff! T. R values don't have to be "complete" because they have to be eventually renormed (since some values of R → "undefined A").

Normzn can be timeconsuming, hvr. Another thot is that since t. proby of "undefined" → φ as we ascend CB, this may be less of a problem.

It would seem that w. variation of ∈ (i.e. use same ∈ for all Q's) normalzn would fix up t. objection to 441.36-.37 (i.e. 442.37)! (or see 443.03)

→ Remember hvr, 445.06-.11 — on Efficiency

.20 (So t. model of .05 may be O.K.) 445.02 was a killing objection to a certain model — model of. 444.32 Maybe t. model was 444.09-.16?

Anyway, consider & try to see if 445.02 is relevant to t. model of .05. (Lrelevant to?)

.23 So 443.34 — 444.09 — .16 (fixes first objection) Then 444.32-.40 & 445.02-.04 was a big objection to it!

While we mite have this (incomplete) prefix set, we had no good way to discover it!

.25 It looks like t. objection of 445.02-.04 does not apply! (.25R) Th. $\frac{1}{2}(1-\epsilon), \frac{1}{2}(1-\epsilon), \epsilon$ system is a prefix set for each value of ∈. In doing Lsrch, we are governed by 2 params: T & ∈.

A trial will terminate if $\frac{\hat{T}}{(\frac{1-\epsilon}{2})^n\cdot\epsilon} > T$ (T = trial time) i.e. $\hat{T} \geq T\cdot(\frac{1-\epsilon}{2})^n\cdot\epsilon$

$(1-\epsilon)^n\cdot\epsilon \approx \epsilon e^{-\epsilon n} = (e^{-n})^\epsilon\cdot\epsilon$

For a given n, whether we are over CB or not depends on both T & ∈. Say we fix ∈ & its "mean value" for t. previous corpus (or maybe $\frac{1}{\epsilon}$ had this "mean value") [$\frac{1}{\epsilon}$ = "mean coder lengths"]

.30 Does this model multiply t. Lsrch time by N (t. length of soln) as in t. objection of 445.0-.04? — Looks like Yes!

.31 For a given n when is $(\frac{1}{2}(1-\epsilon))^n\cdot\epsilon$ max wrt ∈ so $(1-\epsilon)^n\epsilon \approx e^{-n\epsilon}\epsilon$ is max: $-n\epsilon + \ln\epsilon$ is max

.32 if $S = \frac{1}{\epsilon}$ $-\frac{n}{S} - \ln S$ max $-n + \frac{1}{\epsilon} = 0$; $\epsilon = \frac{1}{n}$ as suspected! (2nd max.) $\frac{d}{d\epsilon}\frac{1}{\epsilon}$ is < 0 so it's a max $-\frac{1}{\epsilon^2}$

$(1-\frac{1}{n})^n\cdot\frac{1}{n} \approx \frac{1}{en}$ so is this worse than a factor of n?

In $T\to 2T$ Lsrch, Time for soln. is ≈ 2T; w. $T = \frac{\hat{T}}{pc}$ if we divide pc by en, it takes en times as long for Lsrch.

So t. t. system of .25R is at least e times worse than that of .23ff

The .23ff system takes n times as long in worst possl situation. .25R takes ne times as long in (I think) all cases.

3 INPUT "S" functs using AZ141: .22-.23

V.G. way to Get TM to try Complex functions: (.24R)

(SPEC 446.14) dynamics associates

.00: 446.40: F_1 maps from binary string to 2 F.T or get 1 more bit, stop; U = undefined = nonstopping w.o. output is also poss.
How to create a universal set of such functions is unclear; Well, just take any universal funct from strings to strings, & take 1st bit! That will do it!

.02

Reading Leo-Cover 78 paper: In general, prefix coding always has an overhead cost. That "factor of N" may not be a big deal at all! P 332 col 2 Thm 2:

$$f(l_i) = l_i + 2\lceil \log_2 l_i \rceil + \log\left(\frac{2^2-1}{2^2-2}\right) \qquad (2>1).$$

or $= l_i + 2\lfloor \log(l_i+1) \rfloor$

or $= l_i + L\ \log_2 l_i + \lg\lg l_i + \lg\lg\lg l_i \cdots \rfloor + 4$

False! see .10

the "factor of n" may have been like $\lg_2 l_i$ — rather small cost for prefix coding.

.10 I Misunderstood this! In fact the Leo-Cover paper shows that all codes & prefix codes have about the same ability (≡ "mean codeword Length") & for prefix codes, mean codeword length is ≈ Entropy = $\sum_i p_i \ln p_i$

→ (.00-.02) would seem to almost solve the problem of implementing the 3 input model.
We still need f_1 & f_2 — 2 functions — which definitely sounds very expensive.
Tho, the 2 functions can both use the same "pool" of subfunctions. In fact, F_2 (the "A" output) can use F_1 (the "bit output") as a subfunction.
Consider a function w. $\binom{3}{2}$ kinds of output ① I want another input bit ② A^l output ③ also "undefined".
A^l output is a binary seq. w. a "stop" indication: the stop can be an internal state, but externally Accessable!

.20 Is request for another bit a "Side Effect"? If so, it could be possible to have a function w. {no output & request for bit} or {no request for bit & output}.

See 581.11 for Comments!

.22 Another way: If output [redacted] starts w. 0, we request another bit;
.23 " " " " 1, the rest of the output is the "A".

Here "S" is constructed as AZ141, so we have reasonable pc's for sub-functions

.24R SN we could construct rather sophisticated functs, using previous sub-trees, by Monte Carlo Selection! It may be possible to "detect" in this way, rather obscure sub-trees that one would otherwise find difficult to detect in the corpus!
It might be easy to do as an actual computer pgm, rather than as a "tree structure" (which is the usual way I think about it.
One Difficulty, one couldn't take advantage of fact that 2 M.Carlo Generated functs were identical tho they had different structure — this difficulty occurs in non M.Carlo methods as well!
If it is, in general, a serious diffty in L srch.

.24: 441.31 .22 would "work", but Heuristically, I don't see what's going on at all!
.25 So .22 is a "soln" that makes it clear to me that this is not what I'm looking for!
Perhaps closer to what I want: Say I have (ssz) of 0, 1, 2, 3 etc:

(See 581.11 for comments!)

How do I deal w. these cases?
.28 for N=0, I guess we would say: "No output": that is "ssz=0" means the Machine is very young. A more mature machine would always be able to categorize each new input as being "similar" to one or more previous inputs. My idea of (kind of definability of) "Recognition function" is a "Fuzzy one" i.e. stochastic. — I did consider
.30
.31 stochastic functs — I don't know if I found them Bad, or just what. | NB Any Fuzzness param would be ok — it need not be probability! the only criterion for how this is used to get pc's is max'ztn of 414.18
N=1: This "sort of Q" has occurred once before. So we want to Maximize 414.18,
w. 2 cases of Q_i. The Q's may or may not be identical. The Q's all identical, but w. different A^l's is an interesting ("simple") special case.
So maybe study this case, using (.22-.23) — say 2 or more identical Q's.
If N is large, it is a kind of Bern seq. — but the A^l_a's can be "similar" so that supposed to one another — so that could give much deviation from Standard Bern Seq.
for .22-.23 — take a standard Bern seq. 3 different A types
a [redacted] large N. (How would (.22-.23) implement this??)

(See 448.07)

dynamics associates

.00: 447.40 : SN I think the main definite progress I've made on the QATM problem is

1) 414.18 2) The idea of Recognition Functs & assoc. operators:

At this point, the prompt Q is whether I want the R functs to be stoch, or not — a/c whether I want several R's to be able to work on a new Q — & whether I (update / modify) R's so that each Q has only 1 R_i assoc w. it.

.05 3) Another Branch from Q is to use GP to try to make 414.18. | see 422.20 -.31; 432.20 -.40

.07 (447.40 Spec.) Say we had 3 output strings α 0110, β 0011 & γ 1010 & wanted equal probs & say a total of 100 cases.

One way would be to have 3 different P_i^j functs P_2^1, P_2^2, P_2^3; each had a different

.09 one of the 3 outputs for the same Q.

.10 .07 implements a common Reggy type (i.e. Berm reggy). We could do it w. a single P_2^j function, but I don't know any heuristics to find it.

How the method of (.10 - .09) is not at all clear!

· Overlay: We have those 3 funcs. Q→α, Q→β, Q→γ.

The cool work part: for P^1, Q→α is cheapest: otherwise, we have to write "0011" or "1010".

So if Q→α saves much recalc — ⅓ of the time — Still a colossal cost saved?

.16 An easy heur to find suitable α, β, γ, would be: If Q→α works enuf of the time to "compress code" — then it is of interest — A sort of Hill climbing function for obtaining

.18 good functional components.

So (.16 - .18) Gives a "statistical heuristic". We could make this part of Tim's general routine — along w. a way to pod it (or its applicands) into 414.18 for evaln.

Note that 414.18 does not stimulate (or require) a 3 input stochastic function — Nor does it require of stipulate [such] many

Another Common Heur for Probty problems: (related to .16 - .18)

for a certain set of Q, we find F(Q) is correct unusually many times: We then

.20 try to find a way to characterize the set of Q's for which it works well, so we can get it to work a higher % of times.

Many heurs for S. Probs involve ("reducing"/expressing) the reggy (to) of a Berm Seq. —

i.e. the "frequency doctrine"

→ 449.20

.29

.30

.31 Perhaps write up as it is now: Explaining how it works, but also what difficulties are in training Stoch. functs.

So the entire field of devising Heurs for stochastic functs is something I haven't done enuf work w. — Must therefore develope.

N17/01

ID REV: .04 – .19

dynamics associates

.00 : Review: T. present difficulties about 438.00, in trying to apply Lsrch to t. finding of solns for stochastic TSQ's. Up to that pt. I'd been thinking in terms of det. TSQ's (MTM).

.02 T. initial Q was how to apply Lsrch to t. AZ141 model — ≈ AZ141 applied to ≈ Lisp.

Some Impt. ideas: ← This is An Implementation of 446.05

.04 ① 447.22 – .23 This is a soln. of t. problem of (.00 – .02): it makes it clear that this is not t. real problem: (see 438.23 – .31 for impt. detail) i.e. 447.25

447.24 – .40; 448.07 – .30 discusses t. real problem: finding heurs for stochastic inductions putting them into t. form of 447.22 – .23 ("3 input AZ141") or somehow showing how the heurs ↑ (equ. 414.18)

② On t. problem of finding good stoch models of data: 440.21 – .40; 441.21 – .31, (447.24 – .40 (R)), 448.07 – .30

Actually, there is much earlier work on this: try to find refs

Also [redacted] (Aparently) relevant: Methods of expressing PD's.

.10 ③ 439.25 (as BN) Giving "Index type" for various TSQ's can be a powerful kind of "Hint" to ↓ search time tremendously!

④ (438.23 – .31; 438.28 – .31 is main Argument): How to Get AZ141 to make functions of 2 variables (Q & R) only. — by putting Q & R in list as "constants" (This is sort of part of (449.04))

⑤ 448.00 – .05 SUMMARIZATION of main Impt. ideas in QATM This far

So, very little of 430.00 – 448.00 is important! — T. impt stuff is mainly these 5 points (1 & 2 in particular)

Well, not quite so! Some ideas that didn't work: Various attempts to do ≈ Lsrch on prefix ≈ prefix set i.e. "R" in pobs:

1) use of $\left[(1-\frac{\varepsilon}{2})\cdot\frac{1}{2},\ (1-\frac{\varepsilon}{2})\cdot\frac{1}{2},\ \varepsilon\right]$ ← 445.35 : This gave prefix set but was inefficient in coding! Also some reasons why prefix set could be used.

T. diffy of 441.36 – .37;

2) Use of 11 Proxy functions for total $O(A/Q)$. Very hy cost! 442.34. {tho I think there may be a better way to do this}

.19

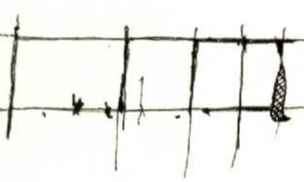
.20 : 448.29: Say I had somehow managed to code a Bern seq. : Could I express t. regy as ① t. set of words (t. Alphabet); t. case counts of each; t. resultant Laplace's rule eq. for t. pc's. — ≈

If I did this, could TM see t. Analogy w. all other Bern. Seq. problems?

How could t. intermediate info: t. case counts, be recognized as important? {Well, They could be parts of t. code ≈ for t. pc's. — (seems to answer t. Q).

How to code t. Bern seq. in t. 3 input umc.? Note that t. case counts are a set of params indep of t. ordering of data: All coding params of interest Must have this property!

Given a set of pc's $[p_i]$ $p_i \geq 0$, $\sum p_i = 1$; how to get a "≈" prefix code for them?



Well, the constant length prefix codes will usually give a v.g. approxn! — but there is a code that gives binary expansion of each p_i.

AH! Say p_i is represented by codes of length $\{l_i^k\}$, $k = 0 \cdots w_i$

.30 Then $\sum_i \sum_k 2^{-l_i^k} = 1$ so there must be a prefix code w. all these lengths.

.32 Given a set of code lengths $l_i \ni \sum 2^{-l_i} = 1$; how to find t. prefix code?

.33 I guess you put l_i in [illegible] order; shortest first: If we start w. r_1 codes of length l_1,

.34 we use t. lexically first r_1 codes of length l_1, next we code r_2 codes of length l_2 — t. lexically "next", and so on.

.35 Will this work? Well, it doesn't matter; an algm. exists & almost certainly can be found (discovered) → See (.37) for correct algm

Unless I happen to think of it: don't spend time on that!

.37 (.35) This is easy; .33 – .34 is rather close to writer: A correct way take t. first w_0 strings of length l_0. Fill out t. bottom with ø's until it's of length l_1; take next w_1 (lexical) strings of length l_1: take last string & fill out w. ø's until it's l_2 long: pick next w_2 strings; fill last l_2 string w. ø's till of length l_3, then pick next w_3 strings of length l_3 ··· etc.

In t. temp. development, no string is t. prefix of any other.

space → 450.00

dynamics associates

.00 : 449.40 : T. "Bernoulli function" : Input is set of codewords (≡ "A_i" values) & case count for each A_i'.
Output is Pc's of each A_i (or) prefix codes for each A_i'. Depends on how we want to implement Lsrch.
This Bern funct. is very commonly used.

.03 After 434.00 – .40 which devs. Lsrch & applies it to R_i^j discovery, I ran into trouble
.04 w. Lsrch for P_i^j (A | Q). — a stochastic function. → (451.00)

[Write about convergence thm for 414.28. Decide on where to put it in Report. See 433.24. 433: lists Parts of Report. See GA: 432.20]

.05 Also, I had objections about t. "5 update modes" ["rules" are 419.00 – .40, 429.00 – .04]
437.00 on cond #3 | In general, I have to go over these update rules to see if
437.11 on cond #1 | they work for S. Ind. (NMTM).

So: update #1 (Everything works fine!) : 437.11 – .16 is v.g., : while I can see how this would be applied to AZ141 induction I don't see how to apply it to t. "3 input machine". Using t. "Bern funct" of (.00)† may help here. — We actually may do it in t. AZ141 part.

.10 How, until I actually understand how to do Bern seq in QATM, this whole update process must be very vague!
Update #2 same as #1 except Pc of A seems too small. Actually, I probably will want to merge conditions #1 & #2. If #1 occurs, update isn't absly necy, — but for #2 it is necy.
In both cases, I guess t. srch would be t. same. Ideally, it would take less time for #1 because update is usually simple (unless new conc has been devd).
update #3: No $R^i(Q_{n+1}) = 1$: 437.00 are correct: try update may work for MTM but its "ssz = 0" (or SSL) for NMTM.
At this point, we don't know if it's MTM or NMTM.
We may leave it alone until we get a larger ssz.
Or, try to categorize it as ~ to some previous problem(s) try various P_i^j functs on it — see if they "help at all"
See if any of t. $R^i(Q_{n+1})$ is "close" to 1 : This is meaningless as stated; But a "closeness" of some value must be devised.
/If it gives 0.8 $P(A/Q_{n+1})$ for a P_i^j, use that; if its a bit low try to modify that P_i^R so that its Π P(·) for its recognized set, is Adequate.

.20 #4 $R^i(Q_{n+1}) = 1$ for > 1 i', & 1 of them works o.k. } Actually, both #4 & #5 are havens for "Merging"
#5 $R^i(Q_{n+1}) = 1$ " " " " & none of them are adequate.

So 2 impt kinds of heur: 1) To devise new R, P., 2) to Merge old R, P.
Type 1 occurs when all old R, P's seem to fail for a new QA: In some (maybe all?) such cases, we may want to wait until we have an adequate SSZ before trying for a new R, P.
Whenever a new P has been found, we may want to try it on older QA's. If it works we have (a) cand(s) for merging.

SN Re: Proof that (for discrete univ p.d.) $2^{-\text{shortest code}} > pc$ (constant indep of what?)

.30 Try to find proof in ① Chaitin ② Willis ③ Li-Vit. : See what order of magnitude it is —
See how it compares w. length of dern. of "True Generalized".

$M(p) = a$; $l(p) = k(a)$ = length of shortest code on which machine stops w. a on output.

$pc(a) = \sum 2^{-l(p_i)}$ p_i = any code for which Machine stops w. a on output.

$2^{-k(a)} > B \cdot pc(a)$ B is const indep of a(?) but dependent on M. What is B indep of ??

T. proofs in Li-Vit, Cover-Leung, Cover-Joy, are all rather long & involved!
Willis has a proof, but only for CB < ∞.
I may try to Willis' idea & run limit to ∞. Say for M, for CB = T, we get $P_{M_T}(x)$ — This is usually for |x| < a certain amt. (that of wT). For each T, there will be a set of x that are assigned Pc's, for certain values of |x|, all x are assigned Pc's for a given T, — also, all shorter x's will be assigned Pc's. (Possibly on t. longest x's)
Re: Willis' proof: if it is true for all CB's, its probly true for CB = ∞! For practical purposes, CB is finite anyway — so a proof of it for all finite CB would be adequate.

→ see 454.00

ID

dynamics associates

Spec 450.04

: On t. form of S. functions used by t. system: I chose t. Σ input-out form because I knew it was universal: & I found a way (447.24,25) to implement it. Here, for Barn Seqns, & probly many other P.D.'s (since many/most) are derivable from Barn Seqns), an output in t. form of a pc, ~~a~~ as digital or analog, (or perhaps M Carlo) form is most feasible.

Could we allow TM to chose t. form that seems best, for a given QA?

Probly before trying to answer this, I should work out more details about how TM ~~d~~ curs Barn seqns. and their params (450.00 is a start).

One simple TSQ might be for a fixed Q: we have several A_i^Q & TM gets their case counts → pc's via Lap's rule.

One possible "Next" would be: Same set of A_i^Q, but Q_i varies somewhat. Try to find relationship

btw. Freqns of A^Q & Q_i.

Th. Bird, most complex form: Both Q_i & t. A_i^Q set vary w. i & TM has to find how t. A_i^Q set varies w. Q_i & so walks how t. assoc. pc's of t. A_i^Q: This is ≈ t. General QA problem.

~~Another~~ variation - say SM or real no. predn.: T. set of A^Q is t. same: indep of Q_i —

But t. Q_i could be vectors. (t. curve fitting problem) — Thus curve fitting problem has a standard ALP soln. Find function of a vector that has best pc mult by pc of data in view of that function.

≈ "solved"

We can do it by having funct chose center & d.f., & give σ^2; or have it give an arby d.f. for each or all of t. data pts. Someone does an exhaustive search over functional forms, then ~~over~~ various ones to find optimum set of params.

Can we modify .14 - .20 so it will work w. difrnt QA's? (i.e. not just real vectors).

If we have A_i be a fixed set of strings, their pc's constitute a vector, so we just have vector curfit — which is a trivial extension of .14 — .20. (T. output vector's for any data set are then evaluated via 414.18)

One interesting Ganzn of .20 - .23 — Part t. set of A^Q is an infinite set of strings (or integers).

If t. A^Q are integers, we must regard them as an input to a function: so we want FC(,) FC(,)

⇒ FC(Q, A) = pc = Max. for $Q = Q_i$, $A = A_i$. Here Q can be a real vector, & A can be an integer. Q & A can be strings a/o nos., vectors, sets of strings, seqns of strings, ect.

But in general, any FC(Q,A) would be OK — just as long as they are Normalizable.

One big problem w. .24 - .27 is Inverting FC(Q,) to find A's of hy pc. — This can be as diff as any General ~~EQ~~ QA problem.

Another Difty is Normzn: Depending on t. form of FC(,) this could be very diff'lt.

Could we use a Mt. Carlo method? Not so easy, since we have no idea as to where A's of large pc's are! That's one advantage of Simplicity: The t. pc's are not normalized: They are usually not far from normal, & we expect that for large SSZ, they would be close to normal.

Well, except for normzn, .24 - .27 might be easy to Update (since we know A_i) — but harder to use for predn (No perhaps we wouldn't spend ~~some~~ such a large fraction of time on "production" predn. Perhaps use QZ search a/o GP to get pc's of large pc. to Answer Q's.

It would be neat if we found a nice way to ~~find~~ search over only funcls that were normzd (or almost normzd) (or easily Norzd)

Spec 452.00

ID.....

20 x Kx B = 20 x 10^{12} by.

500 x 100 = 50MW.

dynamics associates

.00: 451.40 Spec ! It might be conceivable if Σ, A^l were constrained to be reals, or integers — but if they are to be Arby Strings: this may not be so easy! The 3 input Umc begins to look good!

The strings ↔ integers. The int. case of strings, there is no apriori metric — well there is, but not so easy to use in this case (?).

Perhaps it would be best to try a MTM TSQ: then look at the Stochastic Hours for the seq. ∈. Try to find ways in which TM could discover them. (t. CB may be fairly large — It took much Human Effort to discover many of them — sometimes Centurys! ie.

.08 Re: Born Seq. & similar pd's that I have hours for: I can express the rhythm in any form I like — but ultimately, it has to end up in, say, the 3 input umc form. — So the 3 input umc is should have any Prin to do w. hours for NMTM — it is just a way to help

.10 .11 evaluate 414.18. → Note .31 also Note 453.28

If .08–.11 is true — That would solve it! — How to implement hours in 3 input umc form. 3iu 3iu

.12 Def Heuristic T. expression of the should be doable in any universal lang. So try the Born seq. (450.01) — since it seems to be So important in probabilistic Analysis. Consider a fixed Q. We have various empirical A^l's for it; Certain A^l's have occured several times! So Lap's rule is a good way to code it. The Born Seq. might be pervasive enuf so one could get pretty far in NMTM, TROVAKO 453.02

.18 enuf to discover a fair no. of hours. — Using the Born Seq. as one's only NMTM model.

.19 Related, perhaps to Born Seq: A^l's do not repeat much (even Re Q is constant), but somehow, the

.20 — find a numerical function of A^l that gives its frequency of occurance. This could be usefully done even if there is little or no repetition of A^l's.

A rather direct way to do this: try to find a cheap way to code all of the A^l that have occured. This would be a "stoch grammar" — perhaps of the usual stoch. production rule type. Discovery of such grammars: well I have some ideas on how to do this: its 2 (short) well defined problem if we restrict ourselves to certain Grammar types. for the General (Universal) problem of inducing a grammar! I don't know how to start. One might look for short codes for individual A^l's, then short codes for (random) pairs. Then for random triplets: Or look at short codes for individuals: look at pairs of the codes that are "near": then code for those pairs — do similar thing w. n tuples of A^l's. n = 2, 3, 4 ···,

If ssz is large enuf; for each Q_i, devise a grammar G_i; Then try to find "R" G_i as a function of Q_i ⇒ plus, perhaps (some bits) so G_i is a stoch pert. of Q_i.

.30 .31: (11) [N 21/01] Inadequate Re: (.08–.11), it should be possl. to take any heur, expressed in any formalism (≈ "Language"), & use it to optimize 414.18 via 3iu (3 input umc)

Well: Give an Example; Say the Zidl method of coding a sequence of symbols. — This is a Heuristic for Maximizing 414.18. Or even Simpler: we have a Born Seq. & Lap's rule is another Heur Method to maxz 414.18. (Note 453.28)

.36 I have to put a Born Seq. Consider not a Born Seq. but an unordered Set w. diffent freqs for each set member. This problem is easier than the Born Sequencial Born Seq as because in Born Seq. predn, other models are possl. If data is known to be unordered, assigning pr's to each symbol is the best that can be done. — So can one get Lap's Rule? Current "Straight Rule"

Non Steroidal Anti Inflammatory Drug
N SAID

lower rt. = 28 : Lavinds
5 o bit = 29 (Rc)

dynamics associates

.00:452.40 : Perhaps first try to have a clear idea — (perhaps a formation), of just what problem of 452.08-.11 is

.01 — or just t. ~~Barn~~ Barn problem of 452.36 (.11)

.02: 452.18 : On t. usefulness of t. Barn Seq. in Statistics: Situations in which t. Barn Seq. is not applicable:

1) curve fitting
2) linear/n.l. regression (of time series).
3) stoch. Grammars.

(Bernanke?)
Possible applic'ns: 1) Linguistic Analysis of some kinds

.09 2) Cross Correlations.

.10 ... QATM & see how we can solve it.

.11 (.01) Well just draw up a "Barn" problem for w. diff't frequencies "indexed" but identical.
so: fixed Q:, various A's/for (long Sequence:) T. Q's are "indexed" but identical.
(large SSZ).
so: fixed Q:, various A^i's; & Lap's rule is t. first (only ⊕) thing to try.

.13 In General, for fixed Q & large SSZ & small no. of A^i types; & Lap's rule is t. first (only ⊕) thing to try.

We could do a Barn Seq as a sequential object — as a single Q: i.e. after TM has learned to understand what t. problem is — but I think this is a more complex problem than (.11-.13)

$\Sigma = 22$

.15 [So: say we have 4 types of A^i: $\overset{\alpha}{00}$, $\overset{\beta}{01}$, $\overset{\gamma}{10}$ & $\overset{\delta}{11}$, with case counts of 2, 4, 5, 9. — That's t. corpus.

.16 [Q is same for all 22 cases.

.17 I recognize it as a "Barn" situation & immediately involve Lap's rule — which gives me frequencies & then codes (viz 449.32-40)
.17 is all done "open loop" on basis of past experience. I first "recognize" t. s. corpus as a Barn-type problem.
After .17 has been done, TM can verify that t. corpus has been compressed (viz 414.18).

.19R $p_1^{n_1} \cdot p_2^{n_2} \cdot p_3^{n_3} (1-p_1-p_2-p_3)^{n_4}$

.19 Woops! If I just maximize 414.18, wouldn't I just get simple Max likelyhood — i.e. peak of .19R — which is "t. straight rule"? Well, if we sum over many possl. $\vec{p}$ values ....? Lap's rule is obtained by

-20 Integrating over all possl. sets of $\vec{p}$ values.

.22 T. correct way to solve t. problem via Alp: for t next predn: 4 hypotheses (α, β, γ, δ). each results in a corpus w. an assoc. Σ pcost. What we want is normzd values of these pcosts.
Each of these Σ pcosts is ∫ over 3 space of these 3 p_i values, of .19R, (T. integration as well as normzn may involve various "normzn factors".)
The Method of .22 ff would seem to be OSL — but usually not in more complex cases — since we'd have to ~~...~~ get t. Σ corpus pc for all possl A^i contins — & there are many possl contins.

.27 [N24/01] Actually, I should be able to use any ("any" / method of derv) of a hour that I like: (see 452.08-.11, .31 for early note of this)

.28 This has to be true! — Because hours are very heterogeneous — very much given to being easily discoverable w. a suitably modified "Language". I will have to learn to xfm various hours into "standard form" —

-30 i.e. maxzn of 414.18. → Note 456.15 As for TM discovering such hours, first TM has to (discover) learn t. "Languages" that t. hours are expressed in. (By "language" I mean language a/o "point of view"). So I should be able to write up t. hour in whatever lang I like, — Then for t. "corpus" of that hour (the data that gave rise to it — that justifys it) I can note that 414.18 is increased.

Hvr, some hours (maybe most!) don't have a corpus: they are discovered by "Mathical Analysis".
(Add) In this case, one should be able to ~~give~~ conds under which t. hour is expected to be true. — But for such hours, I'd really have to look at one to get ideas on how to deal w. it — on just how (if at all), maxzn of 414.18 is Relevant. — E.g. T. hour could be for ↓ of cc of srch. (so: same 414.18, but less cc!) —— More Generally,

(see 455.00 456.00)

8/22/01 ID

454

How Impt is using pc vs. 2^{-kcost} ? (.04)

dynamics associates

.00: (Spec 450.40): Consider Arith codes: For finite CB, all x's of < certain length have pc's, so one can make Arith Codes for them. In this case, T. code lengths usually $\lceil -\ln_2 pc \rceil$ + k cost of Arith Coding Algm.

However th. Theorem about "$-\log_2 pc$ is within constant of k cost for any x ("within distance is indep of x")"

∴ But cc for "shortest code" could be >> cc for "many ‖ codes".

.04 So: T. interesting problem is: For "small" finite CB; To what extent is $2^{-k(x)} \approx pc(x)$ approximately true i.e. in a search for codes, or Σpc; we usually only code corpi of small kcost; within — at least t. codes we can find in < CB are not long, i.e. kcost is even shorter. For codes of limited kcost, can $-\lg_2 pc / \lg_2$(shortest Code found) be rather small? One application is in $\Sigma(\text{err})^2$ for a corpus. Viz S76 (T3) (correcting C'). If true pc of corpus is C & t. best we can find is C', expected Σ err < $\ln(\frac{C}{C'}) = \ln C - \ln C'$

If we shorten $\ln C$ by Δ; t. fractional ↓ in $\Sigma(\text{err})^2$ will be $\alpha = \frac{\Delta}{\ln C - \ln C'}$; This will also be t. fractional ↑ in "convergence rate" w. SSZ. — it will be fractional ↓ in SSZ we need to get same mean sq. error

10 For problems in which SSZ is a large fraction of project cost (like "Clinical Trials") — This α factor can be very impt. Hvr, for most AI problem it α is .1, it does not pay much; but if α = .5, This can be very impt. speedup.

In many problems, $\ln C - \ln C'$ will be t. ln pc of t. true model wrt t. reference one. — CB need not be ∞ for C'. This would be true for CB = ∞ for both C & C'. (it's always CB = ∞ for C); So C' will be at least (t. ln pc of coding t. model) + t. deviation from C' due to not all codes being found — so $\ln C'$ could be << ($\ln C'$ for CB = ∞) — in fact, t. shortest code may not have been found — so

I'm beginning to get t. impression that using 2^{-kcost} rather than pcost may not make much difference! —

.18 Except for Heur search w. backtracking a/o search in which one does recombn. betw. "t. reasonably good models" We want to retain models that are anywhere near useful, yet are significantly

20 different, structurally, than t. "Best Model" →

Hvr, I really have to Go over This more carefully. T. Goals for Different kinds of Statistical tasks can vary a lot.

① Small ms error is but one Goal. ② Cost of Samples is another ③ In (QA)TM I'll be concerned w. "allover rate learning" e.g. how much cc to get particular "Milestone" (like understanding NL of up to 20%). ④ Path in SM.

→ Also I'd like to be able to characterize .18 - .20 better: What I want from t. "not so good models" that are much different from "Best" model, is: what are t. differences? They amount to an alternative method of coding part of t. corpus (say). That they work at all is of interest: we should try to modify them a/o use them as components when t. present "Best" model no longer works

30 Actually, T. "importance" depends Much on t. Application: In SM, say we use t. "2 part code". The code part that describes t. model can be in MDL or in ALP. The second part is always effectively in ALP. The use of ALP for t. Model can give a much more optimistic (And correct) predn. of future yield

N22/01
ID

"FUZZY Recogn" .11 — .40

(Note 453.02 -.09)

.00 : In General, t. problem of how to Generate S-functs is not a new one! In my attempt to analyse GA from a ALP pt. of View, I ran into diffy of not having good ways to do this. — So maybe list various specific problems of this sort & see if (a) pattern emerge

.02 (1) Old super SGA idea of pc → Fitness; so used of a CFG to get String → pc map, then a ... monotonic funct from pc to Genr Fitness. Hvr, what I think is needed was a way to go from strings (≡ cands) to a d.f on Fitness. (≡ Genr).

(2) General forms of d.f's for curve fits, SM. So this is common to 1)

(3) A common D.F. is "Central" pt. w. pc's of various deviations: "Clustering" is a genzn. of this. Binary strings.

(4) T. very general 3IV form: strings → a/o nos → p.d.

(5) (Sort of) related to (1)(≡.03) Use of a stoch cfg. to assign pc's to strings (a "language") for "Cond'l proby" we want t. lang. params to be a function of Q. (The discrete as well as continuous params must be functs of Q)

As an example, perhaps: Given a corpus of unordered strings. We use a standard simple GRammar Gramar, to stochastically generate deterministic Grammars; Then parse t. corpus via each Grammar (if possl.) & assign pc's to t. choices in t. parsings. This gives a "universal" d.f. overall

.10 →

.11 SN One t. "Recognition" problem: we can have category (cats) & subcats: Geom figure, St line sides, 3 sided, 2 equal sides ; 1 rt <: (Data sets (≡ Bags of strings).)

.13 Also, useful to have problem in > 1 cat (often but not always). Some cats will help in some situations but not in others. Finding out when this is so, can reduce search time. Multiple R's Correspond to dif'nt ways to look at a problem ← This is a very Impt. Heur.

e.g. Problem is both in R_1 & R_2. R_1 gives a good, useful PD (compression) R_2 does not (in this case).

Try to find out "Why": ~~New Cats~~ 1 or more new (or old?) cats that will help tell if R_1 or R_2 is ... best ...

So we usually use an R funct (cat) to tell which method(s) in (.00 - .10) should be tried/used.

N24/01 It would seem Best to have > 1 R possl. for a given problem, initially, but that we normally try to find ways to narrow down t. R funct so it points to t. "Best" soln. for Q. (→ We ask "Why is this $P_i^?$ better than others for this problem?") Perhaps this is t. disadvantage of "MDL": That we don't return low pc Solns, that we may need (w. modifns) later — ("Backtracking")

.20 The I did use > one $P_i^?$ function per Q_i. → I may want to give extra wt. to those $P_i^?$ that were most successful for t. set of Q_i that they were designed for.

.21

N28/01 → I will have to Expand this idea: it seems very Impt! It amounts to a gross overhaul of my initial QATM model, since using only 1 R per Q type was an impt part of t. System!

One way to deal w. this is to have "Soft R's" Instead of yes/no, have a Fuzzy categorizn:

→ I have written about this: The "Fuzzyness" param need not be proby: It can be assigned in any manner: Hvr., T. main Q is how it is usd to guide search for soln., & how effective it is in ↑ α (≡ 414.18)

447.28 - .31 discusses Fuzzy R's — why they would probly be necy: (also a vague reference to previous work by me) on this.)

An interesting property of "Fuzzy" params — it may be that they don't have to be normalizable!

.25 This fuzzyness idea seems closely related to an Earlier TM model in which I had a "Front End" function that would look at t. problem & assign pc's to various IA's (Induction Algms) — These pc's are my initial estimates of "how effective" those IA's would be for that problem.

.30

In t. earlier Version (.28) — I had pc's assigned to t. IA's: Now, I'd have pc's assigned to various $P_i^?$'s or sets of $P_i^?$'s. (?) An apparent Diffy! By having only 1 R^i assigned to a Q, we knew what t. pc of t. A will be. If we have several wtd R's assigned to a Q, t. pc of t. A would be mix of t. pc's of t. $P_i^?$'s invoked by those R's. How do we evaluate α (≡ 414.18) in such a system? — Well, it would work out; t. System would assign pc's to all of t. A's in t. corpus. I'd have to compute t. effective pc's of t. O^j functions of 414.18 — I don't know how hard that is — i.e. t. apri pc's of t. O^j mite be hard to Compute || (419.00-.40) > 950.05 ff; 429.00-.04)

As for "Updating" It mite be done a little Like for t. Previous "5 stages." — We'd still have t. R parts and $P_i^?$ parts of O^j. Another thing that made "1 R per Q" attractive was that when updating, I didn't have to reevaluate t. O^j for any but a few QA_i values. To what extent is it true if R's are "Fuzzy"?

→ (spec 460.10)

ID

Def $\Phi \equiv \alpha$ 414.18 $\alpha \equiv$ 414.18

dynamics associates

.00: (spec 453.40): a heur can be oriented toward cc↓, 414.18↑, or a Goes. Most is some combination of these 2 factors. Usually, the heur will do one or the other (cc or 414.18) so it's easy to tell an improvement has been made. If, eg. cc↓ much, but 414.18↓ a smidgen, it may not be clear as to whether the heur is any good — probly it depends on application. In general ALP has cc=∞, so any heur, will ↓ cc & also ↓ pc (≡ 414.18). In the most general case we have a Goes that is a function of cc & 414.18 — & this function will in part depends on the application. A nutty duty Goes is $\frac{pc}{cc}$: It just so happens that Lenat makes trials in that order.

Ab Initio, I expect that (heurs/methodology) for learning S.. problems will be ("wired in" "inserted by Trainer") That this will be true for quite a long distance along the TSQ. The Chomsky of "Language" involved in many (most?) heurs, seem quite complex & well beyond the capacity of ~~anyone~~ an "Immature" TM!

.10 So, e.g. for Bern Seq-like problems: I will initially "program in" a Recognizer & a P_i^2 function for them. I'd like TM to be able to Generalize on this "wired in" stuff, hur. (Perhaps), the more "open" the (every code) of the heurs is to TM, the easier it is for it to Generalize — i.e. the code for R_i^* & P_i^2 should be as "as simple" as poss.,

Using functions that TM has found useful — that it "understands the properties of."

N 25/01

.15; 453.30 SN This "Being able to use heurs in any form" ("language") is a necy property of any form of induction that I use. — So it is not a particular ability of EURISKO. Any adequate induction system has to be "Universal enuf" to at least be able to do that "language translation" — which is equiv. to universality and the "invariance theorem" becomes relevant: so pc↓ only depends on K (in case of Language).

So it seems clear, that the Problem of Expressing an abby heur in terms of various induction Systems is a problem that must be solved — no matter what form the induction System takes.

.20 Furthermore any inductive system will have to learn the various "langs" used by the various heurs. It may well be that certain induction systems are more "Central" to the set of heurs — so on the average, it is easier for such systems to learn many heurs. If so, it would be well for any ind. system to be able to lrn those particular "Central" induction systems.

So 453.11ff on T2 Bern Seq. predn. is an attempt to Map this Methodology (which I've been calling a "heur") into (414.18 ≡ α) Maxen. (well, it is a "heur" — it is a fast way to get large (414.18 ≡ α) low cc.

N.B. A very easy way to get TM to acquire new (Langs/ways of (doing or Progs) is to have TM lrn "definitions" of relevant Concs. At first definitions would be lrnd by giving suitable TSeq. Later, when TM has acquired some understanding of the suitable language, TM can be "told" definitions of certain Concs. (as a dictionary).

.30 N 26/01 Hur, it may be better to have TM lrn defns in the "usual", purely inductive way — that this might be more beneficial for TM: it would have to put the things defined into the lang. of Concs that it has already (evaluated/understood). W. Human Students, this seems to be the better way.

.36 Anyway, I think I may use do .15ff! Just write out how I do a heur, (for a solved problem) In English: Translate it to a more exact form. Then see what TM has to know, in terms of (defns, heurs, Concs) in order to be able to express that soln. in acceptably small resources — (& perhaps be able to solve that problem w. acceptably small cc).

457.00

ABC abc
ABC ABC

Boston Bicentennial Associates, Inc.

.00 : 456.40 : → Learning to solve (even) Stoch problems, can be understood/analysed viz a conc. net - just like d-problems. } good!

.01 .00-.01 looks Great! So just start at 456.36 : ~~Write~~ write y. soln. in y English, then analyse it into concs, & subconcs & make a conc net that has y. soln at y. top.

Probably y. problem of devising Suitable ~~Rec~~ Recognition algms (y R^2) Can also be formulated as a conc net. —— ie At first, I want TM to be able to do Recogn. about y. same way that I do—

.06 So I'll have to teach it viz a conc. net. .13

.07 SN keep in mind that ≈ ANY problem can be solved viz Lsrch, & if we don't rites soln will be within a Factor of maybe 2 of y. best that could be done. One has to have "all heur. into" y. Pd & one has to learn dumy y. Lsrch. This is related to, 456.36 - 457.06, but I don't think its y. same thing.

.10 What .07 says is that Lsrch can be close to optimum.
A possl. BUG in .07 is that y system that I analysed had no "Look ahead". (≈ Dynamic Pgmg).
Maybe double w. "second order TM" that is Given "Look ahead" as a "Regular Problem".

.13 (06) I could just "warm" some very impt. elementary concs involving probty. Have TM construct its more complex probty concs on that basis. Perhaps Get those Concs from

.15 Wm. Feller's book. (Maybe try to get ~~borrow~~ Second Edition?)

→ What I want now is some examples of TM solving s. problems. Start w. Bern Seq.? —
Or perhaps use Bern Seq. as one of y. "primatives" in .06? If so, how to express it so it could ① Be used to solve probs ② Be combined w. other Concs to yield new concs that can solve more complex/diff(t) probs.
So just derb. how to do it in Eng., then formalize, then "link" to "α" (≡ 414.18)

.20 for statement of problem! say 453.15-.16 : Soln: we Recognizer pays as a Bern problem.

T. characteristics: A small no of A_i^ℓ types; perhaps a few no of A_i^ℓ, w/ repetitions! & Later+SSZ, the better "Bern Seq." Fits. In the present setting, if Q_i is constant, any set of A_i^ℓ would give a Bern Seq. because t. A_i^ℓ are "by definition" "indep". If all t. A_i^ℓ are distnt, t. Bern model doesn't give us compression — but it doesn't lose any either ☺ : we give = pc to all cases, so t. pc is simply t. pc of ~~coding~~ decding. t. A_i^ℓ's.

complete For a "Bern" dcrn, we need dcrns of all of t A_i^ℓ & their case counts: Treat it! assoc. w. that dcrn.

N27/01 Since t. dcrns of A_i^ℓ are self del., we can list t. dcrn's followed by t. case count in integers. T. pc's of t. integers is unclear!

The I could use that approx to $2^{-1.8^* x}$. T. pc's of integers is defined by $p(n) = \sum_{i=1}^{} 2^{-\ell(S_i(n))}$ semi monotonic
My impression is that this was not very bad.
NW From this dcrn, TM is able to make a code for t. set of A_i^ℓ's — its pc is easy to be computed.
It may possl. to just leave the pc (from Laps rule or Sr rule) as is, w.o converting it to Codes: Probably since we will use it in a way that gives same results as Lap's rule.
4(4.18 can use these pc's just as well as "code for 3 i u". — easier, in fact.

SN Riss' $2^{-1.8^* x}$ is for a continuous x d.f. : If we know that x is an integer, my method should give hyper pc's for integers — but I may have tried this & found it not so! — Check into this some time!

→ A concept TM would find useful to create input for t. Bern Seq. a coder (Pc evaluator):
Given a bag of objects, S, & One of t objects A; Card(A, S) tells how many times A occurs in t. Bag. "Cardinality"
t. Bag could be in t. form of a list.

458.00

← = BERNOULLI SEQ

.00: 457.40: So TM looks at corpus of Q,A pairs: First, it finds several different Q_i types: for each Q_i type, it has a set of A_i^l. It gets t. cardinality of each such A_i^l. It plugs this into into the Bern function, (450.00) and we get as output, t. function that maps Q_i's to

.03 d.f.'s on the A_i^l.

There are several operators needed to do .00–.03, but .00–.03 is doable this way, & t. subfunctions need look usefull in their own right (i.e. they should be useful in many other kinds of problems)

.07 The operator of .00 ff is best for Q's that have > 1 A^l: Otherwise, we pool together all Q's

.06 that have only 1 A^l, & we try to find a min code (grammar) for this set. (.20)

Another kind of s-funct. that is ≠ Bern seq, is curfit. In that known form,

.10 we are given a bag of $\{x_i, y_i\}$ pairs : (essentially, QA's). The commonest way is to find $f(x_i) = y_i + \epsilon_i$ with min $\Sigma \epsilon_i^2$, say. This error dist is like $e^{-\frac{\epsilon_i^2}{2\sigma^2}}$ (which is a Bern seq.) ← No! Same values occur rarely

T. analog for strings is: If x_i & y_i are strings; find $f(x_i)$ ∋ $f(x_i)$ is "close" to y_i using some metric — we want Σ of such dists metrics is min. A common Metric: How many bits (much extra code) do we need to Go from $f(x_i)$ to y_i? This is a kind of (condl probly cpc) — but its less diffit than t. general QA problem — because we mite use t. same metric (≈cpc) for a fairly large variety of x_i, y_i pair types.

In general, I want to see how solns to various kinds of s. probs can be expressed, using

.19 certain "primitive" functs.

.20 → (.08) (i.e. .07–.08) in .00–.08 This doesn't always work: Say we have Gaussian noise, p.c. of ϵ is $\propto e^{-\frac{\epsilon^2}{2\sigma^2}}$. Say we saw a bunch of these ϵ's: Most would occur only once, & a few would be repeated (depending on our "resolution"). So we would use t. Bern seq. on t. subset that had > 1

.23 cases of each ϵ. This is a very poor way to analyse t. data. → (459.10)

SN (Get back to (.20)) But first note that for any s. problem, all I need to do is write out a soln in terms of "t. primitives": Do this for many problems then decide on what a good set of "primitives" can be. — Then make conc. notes for t. solns for t. problems.

"Soln" usually means "acceptably by α" but it may mean "acceptably by pc for A_{n+1}" — I have to straighten this out: I have written about this diffy — I think "by pc of A_{n+1}"

.30 had been used in some cases as a hour. In general, I need criteria for when to stop t. srch. (other than running out of cc).

One "prim op" could be finding ∃ xfm (a/o Laplace xfm) — using simple rules or by using Maple or Mathematica or Macsyma — or smaller larger table of xfms.

Similarly integration can be done — either indefinite or definite. TM can "learn" when to "use external pgm" or "do it itself" on t. basis of experience — so as to minz cc. — Just as a person does.

T. primitives may T. stock "primitives" may be derivable (probably are) from d. primitives. If so we should try to get TM to learn them more naturally — rather than have them "wired in". — Since TM may then be able to use t. "sub-primitives" to devise better solns to probs than I know of. — also it (see 459.00)

I.D. small REV.36

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

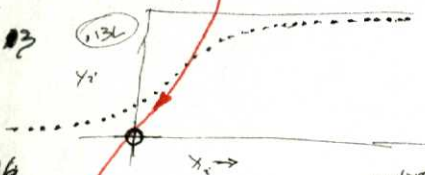
Spec
.00: 458.40 ; would be a better position to "understand" + some to + S. probs it does — "understand" on a deeper level.

→ Go back to 458.20 ! → .10

For the most part, d. predn. & S. predn. will be separate & will be separately maximized. But occasionally I may need to mix them (forget their product max) or for a particular problem, it will not be clear as to whether it's d. predn or S. predn, so I want to try both & compare them. (— actually sum them).
So I will have to (occasionally) convert from one form to the other.

In some problems (like curvfitting) + selection of functional form must be via pgm length, & the optimization of the params of the pgm is for continuous pgs must be (≈ pgm length), so we have to mix the 2 methods, — or translate betw. the 2 kinds of "Language".
— i.e. pc v.s. $\sum 2^{-\ell(s_i)}$

.10: 458.23 For the Gaussian type of problem, there is a common approach: We collect data as X_i, Y_i pairs: Y_i is the total no. of cases in which we've had a value $\leq X_i$ — so we get a jagged curve, that (usually) gets less jagged as $\delta\delta Z$ grows. We can then try to fit a function thru these X_i, Y_i pts.



Say X_i is quantized so it has a certain (uniform) grain size, δ. Some can associate each $(X_i, X_i+\delta)$ interval ($X_i = i \cdot \delta$) with the "pc" assoc w. the ↑ of $f(X_i)$; i.e. $f(X_i+\delta) - f(X_i)$.

So that's a not bad way to solve the 458.20 problem:
But if $\delta\delta Z$ is not small, & there is little resolution, .10–.16 still is a reasonable way to solve the problem — unless $\delta\delta Z$ is so small, one doesn't have enough pc available to decide the function: one must just assume "= pc" for all X_i that have occurred (& zero for non-occurring cases ??!) ← Normalize ← $f(X)$?
.19
.20 Actually, a better way to do .10: Try to find a d.f. (using some grain size, δ) such that the (p.c. of the corpus) • (pc of the function drawn) is max. If the funct. has continuous params, we (.20 L) by the determinant of the Hessian. If det of Hess. is zero, we have wrong dimensionality in our model.
.21

So Go back to 458.20's criticism of (458 .00–.08)

We can use the form model in 458.00–.08 if the A_i^0 are all strings. Usually also if the A_i^0 are a small no. of integers.
Or, say the A_i^0 are finite strings, and there is some function mapping them to integers (several A_i^0 could map to same integer). We then make pc's of strings or the assoc. integers & we normalize them.
If they are not normalizable, the function is rejected.
Better: Have a funct mapping strings onto (0,1) & or any finite interval, & normalize (if possible). the funct so $\sum_i f(A_i^0) = 1$.

A Good thing about the Method of .10–.19: The set of pts of .03L suggests a monotone ↑ funct. We can usually get some kind of fit to it using relatively standard methods.

.30
.31 Well 458.00–.19 (Bern + curfit) is perhaps a Good start for many problems: In general, in a TSQ, I will have other reasons to chose various kinds of P.D.'s for the cands. — TM will have to have these heurs. So I should be able to write TSQ's for S. problems — if I know how to solve these S. probs myself. The Example of 458.00–.06 (Bar seq.) is probly o.k.
I may want to describe another S-prob. soln., to make sure I don't forget the general idea!

.36 For Revised 453.27 starts this present view of how to do S-problems. Then → (end of P 453.40) Then 456.00 to present P. so ~ 4 pp. 458.00–.19 is main ideas, but there are other important ideas in this sequence.
Also 457.00–.06, .13–.45 is v.g. 455.00–.10 has some probs — some of which I've not done well!
Use of > 1 R to categorize, "understand" a Q (problem) (455.11–ff.). Seems like an Empt. Heuristic Principal! Amounts to "Different ways to look at the problem".

N28/01

I.D.

GA (.01)

A Bcde Abcd

.00 : That relatively recent section discussing use of >1 R for a Q was Good, impt. — Find it i reference it in "Mini Review" see 455.11 ff

.01 old [SN] On GA: (SGA) : For cands w. ally. same Gorc (≡ "fitness func. value) I had a way of making a Grammar from any 2 ssz of 2, say: Make a gramma that could have generated both cands: But also use OSL to give somewhat more pc to new cands that use "parts" of t. original pair of cands.

Now → If t. 2 original cands don't have same Gorc value: Devise a parameter that gives less pc to "parts" of the low Gorc cand. — The lowering of pc param is based on diffrnce betw t. 2 Gorcs. — ~~T. way it affects t. p~~ T. amt. by which it lowers "t. pc of parts" must be periodicly updated by ~~Stat~~ statistical studies of ~~the~~ recent emprical results.

.10 : 455.40 ("Fuzzy R's" continued). OK. So lets Go thru t. Fuzzy system more carefully:

A Q_i comes in, T. R function looks at it: Assigns its ~~x~~ soln probys viz P_j — so for all P_j, we have a pc assignd; "a_j" assignd to P_j. (P_j may be a "set" of ~~orders~~ of prems, or only one). ~~x~~ T. system then assigns pc $\sum_j a_j P_j(A_i|Q_i)$ to answer A_i. Now lets look at t. 5 update conds (419.00, 429.00, [450.05])

(1) T. pc assignd to $A_i Q_i$ is adequate. Each P_j involved has had its ssz ↑ by 1 & has to be updated. Normally, this will involve only small modifn of continuous params. Hvr, if ssz is over threshold ~~for a~~ — certain new concepts will be defined a t. effective ~~xxxx~~ P_j (1) may change more seriously (than if only continuous params were slitely changed)
So: Relatively little updating: Sometimes one will not bother to do t. updating in such a situation!
Also Note the R functs have to be updated also because of ↑ of ssz by 1 ‡

(2) T. pc assigned to $A_i Q_i$ is too small. (But is, presumably, > 0). • we will probly try to improve the P_j that were assignd large a_j values for this Q_i • ~~x~~ a/o Look at other P_j's: first t. ones assignd lower a_j than t.
.20 * bygest a_j ~~x~~ — Than later, look at P_j's that were assignd $a_j = 0$ (perhaps), If a good P_j is found for this ‡ problem, the a_j assignment ~~x~~ function has to be modified so this a_j is larger. ~~x~~ (sometimes "larger" than its previous value of zero!)

(3) Q_i is given $a_j = 0$ for all P_j. We have to try old P_j's on ~~x~~. Q_i

(4) and (5) These ~~x~~ don't seem relevant ~~to~~ if > 1 R is assignable to any Q_i. — (4) & (5) are when > 1 R is excited by Q_i

See what corresponds to t. "Merging" here for t. old ~~x~~ 4) & 5) update condition).

In t. present system, it looks like there are only 2 update conds: (1) & (2); (3) is probly very close to (2)

If a • ~~xxxx~~ $P(A_i|Q_i)$ is very bad we might start out ~~xxxx~~ to find an operator that xfms. Q_i into A_i,

Or possibly a $P(A_i|Q_i)$ — But for only 1 case, I imagine that an S ~~xxx~~ function would not be appropriate

.30 Well, actually, (4) may be of interest — if one of t. P_j gives a good P for $A_i Q_i$, but t. a_j is too small, Then we may want to modify t. R funct to ↑ that a_j. We want to do this w. min pc cost, ~~so it will~~ ~~a~~ t. new R funct will extrapolate well. Also, we ~~a~~ want t. new R funct to not change a_j's of other Q's much — so we don't have to reevaluate ~~x~~ much of t. corpus.

So: Consider t. sugg. to be a "framework": T. details of t. update process should be worked out in more detail, when I do t. TSQ. Be sure to try both d-problems & S. problems in t. early TSQ — so I can be sure t. update Algms are reallyadequate.

Also do look at discussions of t. "5 updates" & ~~a~~ Hours & (inadequacies) extensions of them — These may be relevant to "Fuzzy R".

One possy is to use 1R ~~x~~ per Q for d-probs., & fuzzy R for S. probbs — in early ("infant") TM.

N30/01
ID

REV (Sortof)

.00:

.01 So! present picture of QA TM:

1) 414.18 (E α) is t. final Gorc being usd.

.03 2) We can use AZ 141 to assign a prgps to O^j ← (of 414.18)

3) The 3 input umc model will be usd, w. the method of 447.22–.23 being a possl. way to implement it.

4) Two kinds of primitive functions usd in t. reference "umc"

a) Normal, deterministic functs: finite strings → finite strings or (strings a/o nos) → (strings a/o nos).

b) Problistic functions: (strings a/o nos) → pd's on strings a/o nos.

5) Actually both 4a) & 4b) are not both necy, but we do have ways to xff. betw. them. (4a could give Pd's Viz t. 3 input umc trick of .03 (2); PD's can be so narrow as to be deterministic.

Also, all functs that TM "learns" are problistic, since a solution pgm is always possl.

Having these 2 kinds of "primitives" makes it much easier to write TSQ's & solutions to them, & heurs for them.

..08 6) I will write TSQ's & Give solutions & try to find heurs that make these solns. of "accessable cc" (see 586, 589) for both d. & S. type problems

These can be done in usual ways using "Conc nets" (as in 589) [...]

.10 — 7) The O^j functs will consist of a problistic "Recognition" function that will problistically & serious [...]dility of assign. Solution functs to each Q_i • (see 455.11ff on "Fuzzy Recogn") — see 463.28ff for [...] fuzzy recogn. — & some possl. ways to deal w. it.

8) The updating algms for 7) (.10) are not yet clear to me! T. 5 update rules of (419.00, 429.00, 450.05) Could be made a basis of ideas; see (460.10 – .40) ← But this needs much work!

I expect to develop the update algms while working on .08 (6), guided by this (.07ff) set of ideas.

9) In writing TSQ's, I'll use a lot of "indexing" of problems. Also much learning of defns. of various concs. by Ivng. Examples

.20

.30

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

:00 : This will start work on TSCQ & Update Algms.

.01 SNon Heurs: Very IMPT ideas: (1) Partitioning & Recgn. cat. into 2 or more new R funcls. Realization that a former broad category can be usefully broken up into sub. cats, w. solns of hy pc a/o lower cc than before partitioning. [Use of AND to do this]

(2) Merging of Recognn. cats: Realizing that the those 2 cats formerly had different problem solns, ∃ a common soln. for all probs in both cats. This "common soln" could have been one used normally by one of the cats. [Use of OR to do this]

.06 .01-.06 is impt partly because it makes it possl. to "correct" "errors" in cats. — to allow them to "evolve" We probly need more operations than just .01-.06 for good evolution of cats.

Is there any way to devb. a "universal" set of operations on cats — so that any operation can be expressed by a (usually small) set of primitive ops.

:10 Ok: well an initial problem w. TSCQ's was ANL: I was able to get TM to lrn +, −, ×, / easily, in RPN or just (just PN). I wanted them that TM lrn to evaluate

:12 arbit compositions of those funcls. The human heurs, seems to surround the idea of "quantity" — "what was got by evaluating a funct". — That it can be an arg. of a funct, just like a number.

:14 — How to get TM to discover this?

Well, 4+3=7 ; ⇒ 8×7=56 also 8×(4+3)=56 : From this could

Say we do this for several values of "4, 3, 8, 7". Could TM then guess 2×(b+c) = ...

even if b+c has never been 2 the before & 2×2 has never been 4 the before? (2 = level of (b+c)).

After being given several examples of this sort, could TM induce a general rule for

:20 evaluating compositions of functs?

The answer is probly "Yes", but for CQ is, how large is cc?

Intuit w. no simple TSQ : just a small set of examples of compositional evaluations, its possl. to logically induce a general rule, but probly only w. very large cc.

:21 (a) One approach: look at some expressns for a general soln: see what concs might be lrnd that would ↓ its pc. — could these be logic concs. But would have other uses?

(b) Hm, the idea of .12-.14 is the way I think about this problem, & there should be a way for me to express my intuitive ideas, formally.

:27 Re: (b) perhaps recognizing that the evaln of "3+4" is the number "5" are very similar,

:28 so that properties of 1 write be properties of the other.

.27-.28 Is part of a very general heur: If 2 things have several properties in common they are more likely to have other properties in common.

:30 (c) Another view: That if TM "understands" what RPN means, it will automatically be able to evaluate compositions of functions. —

(d) Another possy: Just make up some vague heuristic ideas & make a TSQ built around them — & see what cc we get.

(e) Ask Grace for some heurs: try to implement them/formalize them

:38 (f) Try a GA-type soln: Devise a "fitness function" for a final soln. One such function

:39 might be: How many of a examples does it get right? Also, more credit for longer examples. Say credit = ∑ no.

:40 of bits in problems for which it gets right answer. So this could be a kind of "compression" criterion!

→ 463 space

N30/01

ID

ABCDE abcdefg ZUNIS! ABCDE ABCDEFGH?jabcdefg
abcde ABCDE ABCDE ABCDE abcde ABCDE ABc ABCDE
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcde ABCDEFGH ABCDE

HAMPTON RESEARCH ASSOCIATES
26 Boylston St.; Cambridge, Mass. 02138

ABC ... ABCD ABCDE abcde Zeleza

.00 : 462.40 : [redacted] (462.38)(f) May be worth developing! "A link betw ALP & GA"
T. Gorc of 463.38–.39 mite be fine for GA soln. of "$[O^i_{n+1}]$ from $[O^i_n]$" (!) — $[O^i_{n+1}]$: set i, n fixed, i varies; $[O^i_n]$: set i, n fixed, i varies.
.02 Hvr. if we did that Gorc, it would take much cc to evaluate T. more successful cands! As soon as we find
.04 a cand failed on several Q_i's, we have an upper bnd on how "Good" it can be! — & we may be able to reject it.

SN (.02–.04) is an Example of Q.A. (Quick Abort) heur. Its Gorc is not achievable by Modifn. of t. P.D. Guiding Lsrch. Hvr, would t. Gorc = $\frac{PC}{CC}$ (say) be adequate for this? On some difty of [redacted] .02 : In Human Reprodn via DNA, etc. we do have a kind of Soln to this problem! We have "genes" that create proteins & "meta" genes that control "expressn" of other genes (& perhaps other "meta genes"). All viable genomes can combine — w. usually viable offspring!

Evaln. of cands is Hy CC, hvr.

The "genes" & "meta genes" are a bit like "ops" & "obs" [obs ≈ Recognition Algorithms]

10

.11 (8) follows op of 462.38 After 3+4 → 8 ; 3+4 & 8 are heuristicly linked: so its not so unreasonable to try t. "appropriate" substitution to evaluate t. Expressn.

SN In t. TSQ: When TM solves a new problem, we modify t. P.D. some. If a new heur was used, we also modify t P.D., because t heur was equivt to modifying t. P.D.

[p.1.01] SN Some General Remarks: O^i (of 4/4.18) has t. folg. form: $O^i(A|Q)$:

T. particular subset of O^i functs that we will consider: First look at "Q". → a pd over z. → $[P_i^z]$

Straighten out the indices & z, i, k, l, m, etc.: T. function "R" looks at Q_i; its output is a p.d. on $[P_i^z]$: which are problistic mappings from Q to A. The function R decides what "kind of problem" Q is. For each "kind of problem" "z"

20 there is a set of Problistic functs $P_i^z(A_k|Q_m)$ that map Q_m values problisticly into A_k values. Heuristicly, this corresponds to a Human's deciding that a given problem is of a certain kind & should be treated in a corresponding way.

An impt. effect of this categorization! When the a new kind of problem is solved or an old problem is given a better solution, The R function is usually modified only slightly, and only a few P_i^z functs are modified or created. This minimizes the amount of computation needed to determine whether the new O^i obtained is really better than the old one.

.29 There is a difty here! If R gives a rather broad distribn on z, then any change in one of t. $[P_i^z]$ sets (very z) will change the $P(A|Q_i)$ for many Q_i — which will

.30 make reevaln. of O^i very time-consuming (or hy cc).

Some ways to deal w. this difty: w. pc>0

.32 1) Only have a few z outputs of R for each Q input.
.33 2) When we solve a new prob or improve soln to old one, modify R & t. $[P_i^z]$ so as to minimally
.34 disturb $O^i(A_i|Q_i)$ for previous i values. (.32) ① may be a special case of this.

One way (other than ①(.32)) is to arrange R & P_i^z so that only t. new solns are affected — all other $O^i(A_i|Q_i)$ remain t. same.

.33–.34 Tells part of t. story. — part of what we want. On t. other hand, we want to be able to notice that certain problems are similar but not identical! — That in general t. solution of any problem does impinge to some extent (near zero) on every other problem. In most cases this "impinging" is → 464.00 spec

D1/01
ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: Spac 643.40 : very small, hvr. I want t. system to be able to dcvr. that certain probs, once thot to be quite difrnt., are, indeed quite similar (Merging).

Also, I want t. opposite to be possl.: Probs thot to be all in same cat are best workd on as difrnt cats.

[NB] There is an impt difrnce betw. T. System of S89 say, that solvd OZ & Inv probs — & t. present QATM. — in that in S89 I could have fuzzy suggestions on t. best way to solve a problem. If I had very many, of reasonable wt. this would slow down t. soln.

In QATM, t. wts. are not for "suggested ways to solve probs" — They are rel. wts of predn. methods — having them wrong give bad soln. — its much less forgiving than for Inv, OZ probs

Hvr, in S89, I really didn't have an exactly formalized method for t. entire system —

→ I mite try to do this — go thru S86, S89 carefully: I have to proofread them anyway!

10 So: Present Big Q's ① How to resolve 463.25 : Just how narrow R is.

② Get a TSQ started so I can get ideas on how to do updating, also, fit my genl. system to "RW" ③.

Re: ① There are 2 extremes: i.e. ⓐ t. one I considered 1 i for each Q_ℓ.

ⓑ i has pc > 0 for all i for all Q. (which has difty of 463.25)

.14 I did consider 250.09-.20 was t. beginning of a method that I usd (perhaps later?)

.15 to assign wts. to various (Pams?) when > 1 seemd relevant to a problem (ΣQ_i). → (.20)

.16 → What I may do: start w. approach of attempt to have one i per Q_ℓ. Hvr, empirically, we soon find that we get > 1 i per Q_ℓ. This was dealt w. in 450.20 ff: update situations #4 & #5. There, I tried to arrange so that there was only 1 i per Q_ℓ, but I need not do that! I could readjust wts, so that somehow α was larger (↑), (which may be close to ↑ $O^j(A_{n+1}|Q_{n+1})$). a/o (Modify / introduce new) P_i^ℓ.

20 — Usually, we like to have one i per Q, but sometimes we can't — The $[P_i^\ell]$'s look about equally good,

.21 — or, we can't find a way to modify R to emphasize t. best $[P_i^\ell]$.

Def: $\vec{P_i}$ is t. unordered set, P_i^ℓ ($\ell = 1 \ldots \ell max_i$) Good! (it's not a vector in t. usual sense, but it's a fine notation!)

T. details of (.16-.21) have yet to be worked out, but it seems like a very reasonable approach.

A modifn. of (.16-.21): T. predn. is not a R wtd mean of t. $\vec{P_i}$'s; Instead, t. $\vec{P_i}$ of max wt. is picked for pred. If > 1 has top wt., we use average of them.

Again, if several i are assigned to a Q_ℓ by R funct then we have a "merging heur": it suggests that they (set of i's) are "≈" & so we try to merge one or more of them by finding a new $\vec{P}$ that will satisfactorily solve all problems that point to this (set of i's).

We can try various sub sets of that "set of i's" that we can usefully merge.

---

30 .14-.15 is correct & relevant: I analysd t. case in which sometimes one or more pams would be relevant to a problem (usually only 1 was) & I got optimum wts: but I'm not sure it was exactly t. problem I have now. For t. cases where > 1 pam was relevant — each pam was Gn. a wt. (for all cases by that Pam) ∋ t. corpus pc was max. I guess this will work, but I'm not sure its t. best way. (I did some that about finding t. peak a/o divide by t. Hessian determinant at t. peak. — for t. value of O^j.

This amounts to simultaneous optimization over all of t. wts of all of t. i. Hvr, t. i can be put into "subsets of interaction" — either exactly or approximately. This means that when a new Q_{n+1} is done, only the wts of t. i's in its "interaction class" need be considered. We only consider factors (terms?) in which > 1 i are involved.

Spac 465.00

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

Spec 464.40

"Levenberg-Marquardt method" Gershenfeld "Nature of Mathematical Modeling" p124 §10.4.1

00: On t. optzn calcns needed for 464.30 -.40. The "---Marquardt" n.l. optzn method would probly be fine. One normally has a good initial approxn. available so perhaps usually only one "iteration" is adequate. We can apply cut off interaction wts that are below a certain level a/o arrange so that we never have > 3 or 4 i's applied in a given problem (again cut off range wts) — this will result in smaller sizes of "interaction subsets" so t. optzns will be much faster.

05 Bibl. NOTE: T. earlier work on this problem may have been 267."35" ff (t. "35" Imo is where mine should be!) The discussn runs to 269.00 from 274.09 -.18 (note (274.10 -.14) as a generalized principal. Also: 274.19 ff seems very relevant to present. It would be well to reread & summarize that stuff. That General area of TM notes looks very relevant to present problems — T. idea & FC.) (which is essentially a fuzzy R function) was very impt. in that Model. 274.23-24 was supposed to solve all "Phenomenal" problems of TM; also

10 .15

On 274.23-24: Appssl. Model for QATM: $Gore = \sum\sum z_j O^j(A_i|Q_i)$ (?) maybe not justified!

→ 2 input vmc: ① input decrs O^j. ② inputs O_i: outputs A_i. Each O^j has to decr all A_i's (w pc>0)

O^j is a single code. I think this is just an alternative way to look at t. 3 input vmc model! We just include the R_i inputs as part of t. O^j code. This avoids t. difficulty of some decrs assigning pc=0 to certain QA's. We can think of t. R_i's as being put in "at same time as Q_i's", or all at t. start or all at t. end of t. rest of t. O^j ops.

Another way to look at it: The Q_i's are one by input, t. R_i's are part of O^j decrs; t. A_i's are one by output. Hvr, t. fact that t. Q_i are "unordered" seems to make this different from an ordinary 2 input induction problem. I.e. $\overset{M(S,Q)}{\text{refrence}} = A$

20

So: .11 ff Seems to resolve a Big Confusion: How we could have a set of d-decrs of t. $[Q_i A_i]$ corpus & have them give pc's for any of t. Q's! It shows how this model is a way of looking at t. 3 input s each is a basic function model.

SN Marvin's "Soc of Mind"/Selfridge's Pandemonium: My present approach to TM ts Q's via conc. nets can be regarded as a kind of way to realize these models. Could I use any of t. ideas Marv mentions in "Soc of mind" to help design TSQ's? (4/9/02 — How did I make correspondence??)

30 10 So what we want is a R that usually assigns 1 or a few "i" values to each Q_i — That R can assign apprxt wts to them a priori (or) assign an optimum set of wts, using t. methods of ~ 267."35" ecb (see Bibl note .05). The latter assigns invariant wts to 2 values. These wts are in dep of t. Q_i they are applied to! — so they are somewhat distinct from what I'd normally consider! — i.e. I usually think of i's being assigned to a Q_i along w. a suitable wt by t. R funct. So I feel uncomfortable w. this "dde" wt. assignment! I feel that R(·) should decide on t. wts as a function of t. Q_i's. So t. conflict is between wt being property of Q_i or wt of i being a property of i. At present time, perhaps persue both approaches.

These seem to be Savageal ideas here: OT: problem Q_i & t. soln methods i.e. 3f. problem type" —

Spec 469.00

ID ALP Genl. is ANN applicns

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 On curve fitting using more curve params than data pts.

Say we have n pts & we're using n+k param curves. There will be a (k-1) dim manifold that passes thru all n pts exactly. Hrr, say we want to have m.s. error ϵ^2. What is the volume of region in the k space that gives $< \epsilon^2$ m.s. error? Say we know the params are on $(-1, +1)$ so the hypervolume is bounded. We want V as a funct of ϵ^2. for $\epsilon^2 = 0$, $V = 0$. Is there a usual functional form for V as a funct of ϵ^2 for small ϵ^2? I could probly study it empirically!

Anyway, if $k < 0$ then for $\epsilon^2 = 0$ there are no points. (usually). There will be a min value of ϵ^2 that allows $V > 0$.

The idea here is that for each value of k (≥ 0 or < 0) we will be able to make calcs for the data using a certain amt of info ($= -\ln p_c$). In general, p_c corresponds to the volume in k space, and if $k < 0$, we have to add in info needed to reduce errors.

.10

.12 In the case of $k > 0$ (and fixed), we may be able to do extrapoln to pts not in the original data set. Say for $n+1$ data pts & $n > 1$, there is a certain $\epsilon^2 > 0$, (tho small), we have hypervolume V. If we fix the first n data pts & vary the $n+1^{st}$ data pt, x_{n+1}, V will be a function of x_{n+1}. Then the extrapoln of x_{n+1} from the first n pts should give p_c of x_{n+1} or $V(x_{n+1})$. Presumably, this distribn will be indep of ϵ^2 for small ϵ^2.

.19 I could do a Mt Carlo Simulation: Say the n+k params are each limited to $(-1, +1)$.

.20 We put random pts in the n+k dim space & see how many pts are within ms ϵ^2 of all n+1 pts: The first n pts are fixed, & whenever we get a curve within ϵ^2 of all n first pts, we see what its output is for the $n+1^{st}$ pt. The distribution of these n+1 "predictions" should give its p.d.! We could find the MS of all of these preds. Or, we could make an integrated d.f. plot of how many preds, x_{n+1} are $> y$.

.24 For large n, this is a very inefficient way to do predn, because so few Mt Carlo pts will give a curve w. acceptable ϵ^2. A possl. way to deal w. this: Make ϵ_0^2 very large, to start off. If we find an acceptable $\vec{\vec{z}}$ vector ($\vec{\vec{z}}$ is an $n+k$ vector of the params of the curve) we try pts near by in $\vec{\vec{z}}$ space, & zero in for smaller ϵ^2 until ϵ^2 is $>$ our finally desired threshold. We then start out again w. ϵ_0^2 large and put thru the process again.

.30 This reminds me a bit of feed forward ANN predictions, in which we start out w. random wts, then evolve until we get a certain threshold ms error. — We make a predn. X_0, then we start out again w. random wts & evolve to a < rms threshold & make predn X_1, ect. If we do this many times, we'd get a d.f. for X_0 th. pred. We wouldn't need

"Cross Validation" (Colde Faithfull)

.37 In either case, we start w. random $\vec{\vec{z}}$, and either "hill climb" until $\epsilon^2 <$ threshold

.38 or pick ϵ_0^2 of size larger than so we can hunt for $\vec{\vec{z}}$ w. ms error $< \epsilon_0^2$, then from there we hill climb to ms error $< \epsilon^2$ threshold.

I suspect .12 - .19 is a reasonably good soln. — Is .19 is .37 or .38 better way to realize it? Mt. carlo

.20 - .24 is a kinda/weak realization of .12 - .19

space 467.00

ID ALP Genl. à ↑ ANN applicns

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 466.40 : A possl. criterion as to how large ↓ final ϵ^2 should be. perhaps it should be < th. true σ^2 of ↓ x_{n+1} predn using very small σ^2.

Say ϵ_c^2 is ↓ final threshold oversosi and σ_f^2 is the resultant apparent σ^2 of x_{n+1}

Conjecture: $\sigma_f^2 = \sigma^2_{\epsilon^2 \approx 0} + \epsilon_F^2$: i.e. σ_F^2 is like "smear effect size" on "true" σ^2 of x_{n+1}

If this is really true then there might be an optimum σ_F^2 to use, to obtain a certain/desired precision in σ^2.

A (perhaps) related problem: $k < 1$, but its hard to find ↓ best curve; $n-k$ is still quite large & we are concerned that there may be many local min (of ϵ^2).

In this case we might use Mt. Carlo methods as 466.20 ff. — & I think this is commonly what's done in feed forward ANN research.

Another Mt. Carlo approach for curfit (466.20 ff): Pick random pts in $n+k$ dim $\frac{5}{2}$ (coeff.) space. For each pt. plot its ϵ^2 vs deviation from first n pts; v.s. its value for x_{n+1}. This might make it easier to estimate σ^2 of x_{n+1}.

.15 Another possl way (not Monte Carlo): Look at ↓ k dim manifold that goes thru all n data pts. Get ↓ D.f. of x_{n+1} over that manifold. For $k = 1$ or 2 it might be possl to do this analytically.

.17 → For $k = 0$, n data pts, n params, 1 curve only; ↓ predn would be 1 pt for x_{n+1}! — doesn't sound v.g.! Could $k=0$ be ↓ worst possl case?

As I remember, using linear predn, there would be a certain no. of coeffs (≈) that would give optimum prediction. It would be well to see how much "compression" we get for $n = 0, 1, 2, 3$, etc. — i.e. write out ↓ actual pgm doing

.17 is ↓ Main Negative Criticism of ↓ System. ↓ attraction of ↓ idea is that it gives us apparently good way to analyse ANN & also may suggest approx methods for other kinds of induction problems.

For .17: Say $\vec{z}_0$ fits all n pts; and → x_{n+1}: for hsphere of radius r about $\vec{z}_0$, what is distribn of x_{n+1}?: What is $\vec{\nabla} x_{n+1}(\vec{z}_\bullet)_{\bullet \to \vec{z}_0}$. The dirns of x_{n+1} wrt each component of $\vec{z}_0$.

30 In contrast to .17, it would seem that larger k would be good — giving ↓ system lots of possibilities for many kinds of models.

More specifically on ANN we have (N TNG sets) N data pts) N+k wts. We use cross validation. We pick random wts to start then do vigorous, exact, part of max slope. This gives a 1 dim. (Time ∝ T) path to a peak (it could be a local max). We evaluate Gore wrt Test set along ↓ T path & pick T value for best Gore. This is a 1 dim. optimzn so ↓ expected Gore is not far from observed Gore.

If we repeat this process (w. random initial wts) many times & pick ↓ best of peaks, we get SOY (Spurious Optimization Yield). Similarly using same initial wts but we use path of max slope wrt randomly chosen "cases", we get random paths — & we can pick prior path w. best Gore wrt Test set. Doing this many times & picking best, gives SOY again.

(spec 468.00)

D2/01

ID

ALP Gen 2 & ANN

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

spec
467.40 : We could do similar stuff w. the curfit of 466.00 ff. We pick a random initial set of params, then take a steepest slope path wrt. a [redacted] type error fcn. We have a big set of data to do this on.
We also have a test set, — the procedure is same as in ANN
But I'd like to be able to do it Logically w.o. "cross val".
It would seem that the methods of 466.00 ff & w. logistic should work. It is probably necessary to analyse if the params are linear coeffs (like fitting w. [redacted] polynomials) — But they will be what this is specifically the system that will not work! i.e. the non-linearity used in ANN is specifically diffrnt. than linear coeffs

NB 466.00 ff is not a strictly "Academic Exercise"! I do use this method of averaging over many terms of = wt., in several diferent kinds of problems; One is Jaynes' "max entropy" approach to ANN (using Hessians). Also my coding the Bernoseq. w. 2 code words of all same length. Also my General soln. to the TM core problem of (remaps) 274.10-.14 — which uses a Uniform dsrtpn over continuous params & integrates over them all.
However, in this last (& perhaps Jaynes' [redacted] I think K << 0 i.e. no. of params is << no. of datapts. Perhaps it also works if K >> 0 — many diferent coding methods considered.

But do try analysis of K=0 for linear & the ANN non-linear stuff — e.g. A. Barron 1993? on Analysis of ANN — no of params needed for curvefitting. I'm not sure that Barron worked on exactly this problem, however.

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

(spec. 465.40): "problem type" (i) is a bit unclear. It is assigned by RC's to Q_ℓ: It can be a set of i's (1) or, (2) a set of i's w. assoc wts that RC() gives to those i's for that Q_ℓ. If (1) "a set of i's", each i has its own "comparative wt." ≠ indiv. of Q_ℓ.

One way to look at (2): Each i looks at Q_ℓ & decides how much it (the i) is (i wt. relevant) to that Q_ℓ.

Perhaps (presumably) i "relevance wt" depends on certain "properties" of Q_ℓ (.14)

Slightly "Aside" When one gets a new problem, & one does not feel that any of the "standard methods" are appropriate, one does not normally take a wtd mean of their results. [Unclear! — Depends on whether it's a simple induction problem or a Inv. or many kinds of OZ problems. In my personal QATM, no new "creative" work is done when Q's are answered; the creative work occurs only when updating. — Answers to Q's are always "Routine". A passl. exception might be learning during the soln. of a problem — So one would "update" the "learning" part, during the problem soln.

So! During many problems, impt. "Q A"'s occur! — so the system should be updated before proceeding.

The "default" soln: Well, I could just have R assign wts: But if the R didn't have any good ideas as to what the wts should be, we use "i" assigned wts — that are obtained in a standard way, by averaging over previous cases of several "i" for same Q_ℓ (267."35" ff) (see 465.05 bibl notes)

Remember that R_i's problem is a regular QA-type induction problem, but the data is not QA pairs, but Q_ℓ, successful P_i pairs [also, we can use unsuccessful P_i's as data, but then the Q_ℓ has to be "indexed" so what we want is "Failure" not success.] The Q_ℓ to $\vec{P}_i$ induction is easier than Q_ℓ to A_ℓ induction because there are fewer $\vec{P}_i$'s (Less info needed.) — So we have this "2 phase induction" — first R gives Q_ℓ to $\vec{P}_i$; the $\vec{P}_i$ pass from Q_ℓ to A_ℓ.

Since R's problem of $Q_\ell \to \vec{P}_i$ is a regul. induction problem, all $\vec{P}_i$ will get wts $> \phi$.

I suspect that we will be only interested in the "top few" $\vec{P}_i$: We could try the system w. Various cutoff pts. for no. of $\vec{P}_i$ considered. For the Pred. phase, we can use as many of the $\vec{P}_i$ as we have cc for — usually only a few should have more wt. (The "Cutoff No." could be a "system param," that the system optimizes)

for updating, we try many of the $\vec{P}_i$ to see how well they do in Q_{n+1} thru A_{n+1}. If we don't find any good ones — we try to generate new $\vec{P}_i$'s somehow based on our knowledge of Q_{n+1}, A_{n+1}. Otherwise, we might just try the $\vec{P}_i$'s in order of wt. — (Maybe best, since certain trials will take too long.)

05/01 The "trials in order of wt." — the wts could be optim assigned via A≥14.1. — This is att. to my "What the wts (≡ pc's) are, but presumably the pc's get a better empirical basis as TM features" ☺.

That's an impt. point. We start out w. $\vec{P}_i$ assigned via A≥14.1. As we get more comp. data on how good the $\vec{P}_i$ were for various problems, RC() assigns $\vec{P}_i$'s to problems more accurately (than by RC()).

.37 We have (3) things going on w. R & $\vec{P}$: (1) for a fixed set of $\vec{P}_i$, the pc's assigned get better as ss ↑ & cc ↑. (2) When the best $\vec{P}_i$ do rather poorly on a problem, we generate new $\vec{P}_i$'s for that problem. (3) When pc's assoc. w. all useful $\vec{P}_i$ get too small, we drop that $\vec{P}_i$. (Not using a Threshold here seems "PROCRUSTEAN" — But we may want to don't to save CC in a General System)

472.00 spec

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 Time to Go back & look again at GA: What its faults are & how to fix it!

Faults: 1) When problem is Given, it has to be put into form so that GA can work on it usefully. This involves: Deciding on Population size, type of Mut/crossover & val freqcys of them. Design of fitness function — & probly other things. This definitely has not been "mechanical", & makes much human skill. [initial "start" population]: Also monotonic x-fcn. from y. fitness funct. to reproduction frequency

.04 2) After a problem has been solved by y. system, it discards all info obtained during y. soln. "Kill y. Eqr" — Then conceive and train from babyhood a new Eqr's for newer problem." Very wasteful.

.04 3) Evaluating y. fitness funct. can take almost all of cc of y. soln. Approxns. of

.09 4. Fitness funct. should be used; As we approach a final soln, our approxns must improve or we don't use approxns anymore

.10 Solns. to these problems:-

1 & 2) (a) One way would be to have a sep. set of probs of v. diff (& tsq). These probs would not be "prepared" by y. trainer, but would start out w. essentially "prepared" examples & continue w. less "prepared" problems. This, however, is only one aspect of TSQ.

.18 (b) The problem of y. system at time i, is to find a funct. that can

.19 solve y. first i problems. (See .20R ff for a possible modification of this idea)

.20 As i ↑, evaln. of y. "fitness funct." becomes very time consuming — See .10R for one approach to this. ~~Another approach~~ In addition we can do approx. evalns much of y. time (.09 - .10)

(c) TM remembers by a population of solns to past problems, when a new problem is given, TM starts w. a population mixtures able to solve "similar" probs in y. past. Knowing "what is similar" is a serious problem: Perhaps find way to use GA. to solve it.

Also, only basis of "similarity" it can give reasonable values for

.30 Params est .00 - .04 (Big part of fault #1)

Similarity can be viewed as an individual problem w.r. usual ALP Gerc. So it is a "fitness function" — probly directly usable.

.31 SN I've been Trying at y. GORC as being an adequate soln to all probs upto now (.18 - .19) Perhaps this is unwary: If y. TM can solve y. last 10 probs (or 10 important ones fully selected "Quick abort" probs, this may be adequate & would reduce Testing cc enormously! (Also see .10R)

Another aspect of this: Our population pool does store lots of info about solns, So there is a strong tendency for a cand generated from Pcum to solve many probs in y. TSQ if it is tested on only a few.

One way to Do This is to use GA. to update α (= .14 .18) — notns above argts on why it may be practical.

Thus (temp) must be chosen fort. problem

.10R SN I was concerned w. legacy of evaluating conds in GA, if y. problem to be solved was always finding a good one for y. entire corpus.

We can weaken this cond considerably! There are distinct "Areas of knowledge": Each update affects only (very) a certain area. Also, noticing which problems y. conds fail on, we can order trials so for "Quick Abort" Note .31

.20R SN The idea of a "problem pool" The Gerc, is how many problems solved — or a wtd sum of problems solved — w hard probs count more wt. If GA. is used, we have Mut/cross betw. conds that have solved most probs: We write "index" conds, telling which probs they solved — this info would be used in Mut/cross for new trials

There is a previous note on ≈ .20R within last wk, (certainly within / last Month) I think at bottom of page.

ID

GA .00 | Recursive functs : .13 AK
.24-.25 is important (if true ☺)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

Spec 470.40

.00/470.44 → Another Defect of GA is that I think there should be better ways to get conds than mut/cross: Grammatical or Mapoln., perhaps... I've written a bit on this.

Another very impt. pt.: I'm thinking of a GA machine: Since any problem can be put in this form, w a suitable TSG, it should be possl. to get this GA system to work on all of its problems of 470.00-.10 — to improve itself.

In designing a GA system of this sort, I will probably get ideas on how to do "updating" for the non GA QATM. It might be very useful to design the 2 systems in ||.

.04 A possl. Diffy (& possl. Strong pt.) of the System: I will not be "micro programming" the System: for the most part, I wouldn't know how it's solving problems. I'll have to devise tests & using conc nets, but I really wouldn't know if the system solves the problems using the concs in my conc net!

.13: **Recursive functs:** Simple Recursive functions: The value at one pts X, is related to the value at another pt $f(x)$ by a certain function: so $G(x) = H(G(f(x)))$ The functions $f(x)$ & $H(\cdot)$ have already been defined, & this is a partial defn. of $G(\cdot)$. The rest of the defn. gives G's values at some pt x_0 so $G(x_0) = y_0$ is known. So $G = H(G(f))$ is one functional eq. & $G(x_0) = y_0$ is the "Boundary Value". I have studied $G = H(G(f))$ in the past, as a kind of "difference" eq.

$H^{-1} G = G f$ which is a kind of commutation relation betw G, H^{-1} & f. so if $J = H^{-1}$, $JG = Gf$.

.20 $JJG = (JG)f = (Gf)f = GFF$ so $JJG = GFF$ and so $J^{(n)}G = GF^{(n)}$ ← which is the usual way of expressing

from $G(x_0) = y_0$: we get $J^{(n)}(\underbrace{G(x_0)}_{y_0}) = G(F^{(n)}x_0)$: $J^{(n)}(y_0) = G(F^{(n)}(x_0))$ This gives G for the related $F^{(n)}(x_0)$: $-\infty < n < \infty$

If we can get n to be real or just rational we can define G for a continuous range — but usually integer n (usually ≥ 0) are used. $f^{(0)}(x) = x$.

.24 → This is impt. for defining functs in QATM or any other TMs, because we often have
.25 $G(x)$ bearing a known relation to its value at some pt related to x. (I Pine) — But I should find

→ **examples** It should be an impt. Heur for defining "Recursive" functs.

Differential eqns, Difference eqns, Differential/integral difference eqns are all special cases of Recursive function defns. They all need "boundary values" as part of the defn.

D5/01 TSN on Recursive functs. $f^{(\frac{1}{n})}$ can be defined by $(f^{(\frac{1}{n})})^{(n)} = f$ so we can then define $f^{\frac{m}{n}}$.
However $g^{(n)} = f$ may or may not be solvable: if $f = x^k$ $f^{(n)} = \frac{1}{n} x^{k-n}$ we can define $f^{(n)}$ in this
.30 case, even if n is complex. If we get an "analytic" soln for $f^{(n)}$ as a funct of f & n then we can make n complex/continuous, etc. — Maybe n can be a vector or matrix.

in X^{n} can n be a matrix? If n is a square matrix, any analytic funct of n is possl. (defined).
If we use a coord system in which n is diagonal, then the interpretation is very simple!

ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : (spec 469.40): Re: 469.37: The "improvement" of the pc's of $\vec{P}_i$'s could be done via the methods of "267.35" ff (see 465.05 Bibl notes)

This assigns a pc to each $\vec{P}_i$ indep of Q_e, hwr. But the $\vec{P}_i$'s do get assigned to Q_e, but in a "Logical/(Yes/no) way.

This "267.35ff" stuff does sound rather Gross! — But it probably more or less "works".

.03 Intellectually, it would seem to be more reasonable to gather data on the effectiveness of various $\vec{A}_i$'s on various Q_e's and then tailor our R accordingly. This is what 469.37① is about.

.05 The Ring about "267.35" ff: If we make $\vec{P}_i$ assignments to Q_e in a Boolean way (yes/no), & we sometimes have > 1 $\vec{P}_i$ assigned to the Q_e's, and we want the wts to be a function of "i" only, then "267.35" ff is correct.

If we want to use more info for wts, — make them dep on Q_e as well as i, then 469.37① rather than "267.35" ff is a better way to go.

.10 In an "infant TM" we might use 267.35 because there is little info available. — But, I suspect that there could do a Yes/No assignment of $\vec{P}_i$'s to Q_e's — one could just as well (in terms of §§2) use Fuzzy (soft) $\vec{P}_i$ assignments.

Think about how pc's are assigned to $\vec{P}_i$'s for Q_e's. I had the idea of "kinds of problems" so Q_e's would be "pooled" — a Boolean N/Y type of categorization.

Anyway, 469.37, ①②③ is a nice formulation of the problem; the part that it doesn't talk about is how certain problems are very similar & so can be solved by the same $\vec{P}_i$'s. I think of Ring as a "Boolean" Ring — but, it is, in general not — but people usually think about it in a Boolean way. → I certainly will have sets of problems that are all solvable by the same set of $\vec{P}_i$.

.18 .183

.19

.20 D6-01 So we have a difficulty! 469.37 ff v.s. (.183 - .19): If 2 Q_e's have many similar pc coefs for the $\vec{P}_i$'s, we might consider them to be an "equiv. class". If there are many (or large) coefs, it would be easy to compare the 2 Q_e's.

.22 I'm tempted to use 469.37 on the TSQ to start. Then see how I can fix it in the equiv classes (.183 - .22)

The stochastic R is v for the old FC.) that I used in the 589 model while I was at IDSIA.

At first, the indicators given by trainer will tell TM which probs are "equiv" in the sense of expecting a common D-funct - Boolean $\vec{P}_i$ soln. Then later, TM will induce (S-funct) on the basis of observed, empirical spatial pairs that have "world well" (= given by pc's to R_e's).

At first I will have only 1 R class! It will be solved by a single D-problem. Then that Grows as the TSQ Grows.

This R class will be defined by the index given by trainer.

We may try several "R" classes; some would be S-problems.

After we have a fair no. of R classes and TM has learned to work S-probs of suitable kinds,

.30 we can put in unindexed probs & get TM to assign pc's to them (i.e. "R" function) to sort which $\vec{P}_i$ Fund would be appropriate.

So I want to try a TSQ (maybe only 1 R class!) & see what (tools/devices/heavy) are needed to "solve" the TSQ.

I was thinking of ANL as an initial type. But what about "Definitions" — how would these fit into the column of the ANL TSQ? Would I want each defn to be a new "R class"? — perhaps all defns could be a single "R-class"!

What I may want is a general framework for (my) TSQ's — & a set of mechanisms that may be used to implement them.

473.00

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 472.40 : **Re: ANL:** I could have it lrn +, −, ×, ÷ as 4 separate things. Then learning more complex notation could be based on that (rng).

Alternatively, have it lrn all those operations as a single function, but with "component functions" that "recognize" what to do in the various situations.

BACK in my Mind, is always the idea of a **cone net** — that all problems are to be solved this way. If so, I have to make the concepts available at reasonable pts's when they are needed. One way to locate conc. was by

.06 **"Context"** — How similar the situation in which a conc may needed" was to ~ situations in which it had been used in past. or "senses"

I think .06 says very much what I mean by "context", when it's used to enhance pick of concs to be used in a new problem.

Going Back to ANL: I had Ring (human) concept of "quantity" that I wanted TM to lrn, so it could guess that substituting the "value" of 3+7 (i.e. 10) in an expression that had "3+7" in it, would give a true eval'n for the expression.

.10

A diffrnt approach was to use a machine containing a "stack" & have TM lrn. to use it — which is what I did in Soarb.

Well, the "value" concept; Perhaps its main property is the substitution idea! So if TM understood "value", it would already know how to evaluate any alg. expressn! — So maybe my difficulty in getting TM to understand "quantity" (= "value"), was that I wanted to explain it w.o. telling it about its use in substitution — perhaps an impossl. Thing to do!

So perhaps I should be more skinnerish in teaching TM about "value"! E.g. say:

.18 If 3+7=10 then 3+7+5 = 10+5=15. Or even do it w. literals. (?? — how?).

Anyway make TSQ that suitably infantile human should be able to work; Then devise TM, so that it can also work that TSQ in the same way.

Perhaps teach many of "substitution": its impt in ① eval'n of alg. expressns ② solving eqns.

"Teach by example" induction from examples.

T. Ring to be lrnd is meaning of subs(a, b, c) "substitute a for b in expression c".

Maybe ambiguous sometimes!! a = x b = aa, c = aaa : → xa or ax !

.25 Very Early in Rng — (maybe in BULG?) I practiced up some alg. problems — that did not involve ANL! — Some were about Literals, others simple eq. soln. — Try to find Pgms!

Maybe like: a+x = b, a+3 = b, x = ?

What is probly a major problem: I tend to make assumptions about what is "Already Understood". In Princ. Mathematica they come closer to making clear what all their assumptions are.

One possl. (New) approach: write down lots of "true" expressns, that seem to have adequate "coverage"; then begin asking new Q's. (first ask "old" Q's!)

.30 In .18 TM could learn about substitution — just what "equality" means.

Also, the idea of some info being local to "this problem" & other info being "global" — to all problems. "If a+b=c; b=3 then a+3=c" a+b=c and b=3 are "local" info.

"If" seems to have a special meaning here: is "localization indicator". I think I ran into this idea in the Bulg" stuff referred to in .25.

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

:00 : 473.40 : SN (to some extent) : In doing my early TSQ, I can probably go far in certain directions when using certain minimal inductive concepts. e.g. stick to d-probs: Use very simple "built in" induction for "SI" aspects of the system (mainly Leap's rule -- part of which is built into the AZ141 Notation).
I was concerned that I'd need some ε-"ε-function" thing in even elementary

SN | Re: use of fft. pt. for early TSQ. There is the problem of error size. I could explicitly introduce error size, by giving an ε "ε" for each problem.
Alternatively, start w. integer arith w.o. division: Then introduce division, but write all problems having later needing only simple-division w.o. remainders. Perhaps introduce division w. "remainder" — so all arith is exact. But this amounts to 4 − 3 = 1, 1, 3 (≡ 1/3)
So we would have to teach it how to do arit. w. phrase "3 vectors". It would amount to knowing how to work w. rational nos.

:30 So one/pupil would teach TSQ integer arith — & division would always (in the problems examples) have zero remainder. (If this is passed.... TM will be doing experimental division of 10 by 3)
So I think TM must know that 10 ÷ 3 is not just 3.

:14 SN | Jerry & Marcus have been trying to find general "shortcuts" for doing induction: (w.o. TSQ's)
Can it be proved that TSQ's (= incremental (rng)) is necessary for solns to induction problems w. acceptable cc? If Ls such is, indeed, an optimum way ("If all info is in P.D."), then w.o. TSQ's the cc ε (= cc/pc) is, indeed, enormous. So only by providing a chain of problems in reasonable order would diff'lt probs ever be solved.
:18 "Trouble" w. it — it is that a human, (& presumably, a mature TM) does not need extraordinarily (invent) provided TSQ's to do research. (Not exactly a "Trouble"!) A human T. Mature (TM/scientist) could simply find problems that seem within its CB to solve, & work on them. A mature TM may be using the "Problem Pool" type of TSQ. A mature scientist knows of many interesting problems & works on those that he thinks he may be able to solve, using his current set of concs.
:20 SN | Back a few pp (<< 20 pp ago) I had this idea that the Representation I could represent the pc of a seq. of QA pairs by a set of pairs in ||. On second look, that would be impossible if any Q had ≥ 1 diff'nt A in the set!
Anyway Marcus (perhaps), defines a pd P(A|Q) as [Σ p 3.14, eq(2) of their proposal]
This looks like a 2 input umpc: $P^{\text{Marcus}}(A|Q) = \sum 2^{-\ell(R)}$ M(R,Q) = A.
So Hmm, Marcus is just using a universal d.f. for P(A|Q):

:30 SN | TTSQ's for Sol 56 second easy enuf to write! Somehow I didn't have the problems
I've been getting in writing TSQ's — even for ANL!
Perhaps just write up some TSQ's for a human & give desired responses: See just what info is needed for those responses!
There Diff'lt things to learn & Diff'ty in Conceiving TSQ's to run them — but I haven't really gotten into the "serious problems" yet! There are some fundamental notational "understandings" that I have to try to sort out how to deal w.!
Hm I wanted TM to somehow acquire the concept of "quantity", but actually, I don't really :40 have to do this: Just write out solns to $3 \overset{x}{+} 4 = \boxed{?}$ so then solns to slightly

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00: 474.40: more complex problems, than soln to general ANL problem! Try to devise a seq. such that a human should be able to run it. Eg. start w. depth 1, then depth 2, then depth 3 etc. Also 1 then 2 substitutions at depth 1, etc.

See if idea of substitution can be taught (say do this first);
e.g. subs (a, b, c) means substitute a for b in expression c... See what cases are nasty: or maybe use this as a prim. conc!
Try to list a set of prim concs that would make lots of alg. operations "simple" (MDL).

BUT DO TRY 474.40ff.

In doing, try to get >1 soln, & save all of them as possl. "component containing" objects & for "Backtracking".

.10 T. langs I use for solns, should, at first, be merely "English"! (i.e. ambiguous & nasty)

Perhaps the way I got into that bind was "Quantity"! I assumed therefore v.g. way to write TSQ's was "Top down", start w. soln. to Diff't problem, then find other smaller probs that have parts of the soln, and so down to prim. concs.
Hvr, until I have more Experience w. this & the "Top down" is not nec'ly the best way to start out!

.16 Perhaps try TSQ from Elementary Alg text. — As they occur in text.
See what concs are Necessary!
Hvr, write out Solns in "English" — After a fair amt. of TSQ has been done in English — only then, begin to formalize it. ("Formalize" means "remove (noise/uncertainty)").

.20 Another Source of inspiration for concs (prim. or not prim.) are words in English used to (discuss/describe/solve) math probs. Not nec'ly terms that are formally defined in Math.

In Sarb I did write out solns to some problems of Int.! like linear → quadratic → cubic eqns.
Also a wizard way to "Lay t. Laws of Algebra".
.23 Trying to do Ramanujan's text is also a possby. I don't have to use simple "primitives"!
I can use whatever "prims" seem nec'y to solve & sort. The main Pty of int! is that the concs should be as "universally Extensible" set in a useful way. <Any set of concs can be made universal by adding a small set of concs.>

.27 One Approach! To do many distinct TSQ's of very distinct kinds: (Many of them starting w. a rather large set of complex concs). See if I can learn any Pty about writing TSQ's: a bout update algms: from this exercise.

.30 NB d Elementary problems, like a ANL is very elementary algebra, and diff't because it is often hard to recall (decline) just what the probl. is: what are local, u.s. global conds, ect.
So notice this! Perhaps avoid such "elementary problems"! Perhaps try to devise a TSQ that avoids such probs: like 50|56!
Another unpleasant feature of "elementary problems" is idiosyncratic placement of symbols (476.22-.23)
Maybe start as a more advanced stage!

476.22

D11/61
ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : 475.40 : [SN] Getting QATM to "understand" what an a-oz problem is, is not so easy!
Even harder is defining "Anytime" problems! "OZ" seems easier. Perhaps defn. is related to the "2 part soln." I had for OZ problems in t. IDSIA mouth: I vaguely remember: 2 functions: t. first was inductively evaluated (w. diffy). t. second was an analytic funct. of first (easy to evaluate via Mt. Carlo — an optzn, rather than OZ problem), I first obtained this soln. in doing t. "Mixd Corpos Prm (1998).

I think t. first function is: Given an O.T. α (Pr ·) ; (α is name of O.T., Pr is t. problem descr.
Pr could be $M(\cdot)$ where t. prob. is to find $x \ni M(x)$ is "as large as possible")
and an OZ problem, Pr, CB [Pr is an optzn. problem, CB = its CB)
T. vpred and data on Pr's work on various Optzn problems, $(\alpha^{Pri}, \alpha^{C}, \alpha^{M})$ ($M \equiv$ Gore)
10 to get a d.f. of M as a funct of C, for α for a given Pr.
Given a bunch of OT's & a specific OZ problem, t. second funct gives t. pc that a given OT will give t. best soln. to that OZ problem w. t. specified CB.

→ A possl. Approach: Perhaps insert suitable "Prims" to make t. defn. of . oo easier!
(Thus, in Recursive function theory; Picking t. max of several variables mite be one of t. primitives. — Sometimes t. max doesn't exist & is perhaps a common source of ("non-stopping"/undefinedness).)
This seems to be an essential problem: But If we want TM to do "SI", it should understand what
.07 "Improvement" means! → 480.38

.20 [SN] 279.19 Discusses $F(\cdot)$ (which is a $R(\cdot)$ w. pc's rather than Yes/No (Boolean).
Some diftys in $F(\cdot)$'s "lrng". Also Good genl. discn of $F(\cdot)$'s "updating".
.22 : 475.40 In lrng. "3+4", say; T. fact that this is indep of horizontal motion is ≈ built in as human hdwr.
.23 So I have to either build in recognizers indep of hor. motion or give examples w. various hor. displacements.
.24 List some possl. TSQ's to start.
1) ANL: see 474.40 ff: for sometimes! Perhaps use subs(a,b,c) as prim. ... Other dirctns.
476.22
2) Sol 56
.26 3) 425.16 TSQ from ordinary Alg text! (425.16): One possl. advantage: That it may be Human-lrnable. → .31
4) Ramanujan's text book, 475.23

30
.31 A possl. advantage of .26R: It will have lots of concs missing, but these are things TM must know, & I can insert them as "primitives". My discovering what these "missing" concs mite be seems very impt.!
So write out solns in "English": see what concs. seem to be needed. Do a fair amount of TSQ in "English" before attempting formalization. I may be able to get code length/pc. estimate from "English" dscns.
Pretend that I am explaining things to a neonate!
Mite be (possible/feasible/good idea) to do for ≈ ("English") w. ≈ ANL & some following ideas. Keep "NEONATE" in mind!
.38 Or: Just try what would seem to be an adequate (for Humans) TSQ for "ANL ++" & compute t. cjs's! If they are accessable to me: try them out at these CB's: See if there are simpler solns. Reform them as we go along: they would ↓ t. cjs's of subsequent 477.40

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 476.40 . problems, a great deal! .03

One update-heur. that seems to follow from the ANL work: That one has assigned Qnty to $\vec{P}_i$. $\vec{P}_i$ doesn't work very well w. Qnty: Then Modify $\vec{P}_i$ so that Qnty will "work" (i.e. until Anty tries my PC.)

.03: .00 Try very sparse tsQ to start out with & look at cj's: Then insert various examples to augment the TSQ & see how the subsequent cjc's change.

.04 SN Perhaps get Peter Baumann's TSQ! From Wolfgang -

SN for "Report" Give the (tentative) update schema that I've written up: But make it clear that this is very preliminary:
— That the real update schema will be devised by studying TSQ's — seeing how humans seem to solve them, &
expressing these heurs in formal ways. That. heurs Given seem to work for some kinds of
d-problems & probly will not work well for s-problems.

So: Finish up report & send it out! But first, review my present views of TSQ's! 476.24 — 477.03

.10 — But w. more detail on each approach, so I can easily "get back to it". → 478.04

.11 SN On my proof of the conv. thm for BAGS & for QATM: (see 98TM on the Mixed corpus thm, for the proof.)
Again some strange case! It would seem
That the expected error is localized by the amount of ~~P(Qi)~~ ~~P(Ai)~~ to

$$Z = \lim \frac{P_{\text{Approx to be evaluated}}(A_i \mid Q_i)}{P_{\text{True}}(A_i \mid Q_i)}$$

. But I must look at the whole proof to analyse this
Z can be <0: which I worried about before, & found not relevant!

Another point in this Area: When $\prod_{i=1}^{n} P(A_i \mid Q_i) = \omega$ is small, this means that the Expected value of ~~the~~ Σ sq. err. over i=1|n is small. What it says about the future is unclear! So, is it relevant to the Occam's Razor Q? → 478.00

.20 Is it really evidence that one PEM is better than another? Well, It looks like we did an honest evaln. of the Pem w honest A priori

.21 SN On comparing ALP w. "o/de Faithful" (= O.F. = "Cross Validation"): O.F. gets honest a priori by actually testing
the found parms "apriori": Better data is seen. Alp gets the "a priori" by less certain means, but by
it doesn't waste data (which O.F. does).

.24 So ALP gives us an "Honest estimate" of ~~past~~ past err²: $\overline{\ln P(A_i \mid Q_i)}$:
I Guess O.F. would also. Apart from "Honesty"; what are the σ² of these estimates of the future? (i.e. ALP's O.F.)
Perhaps the σ² are really what the Question is, in comparing them.
From a practical pt. of view, ALP tells one we have to look for good models. O.F. does not.
One could use ALP for generate trial models w. low complexity then use O.F. to test them — which is usually
how O.F. is done: i.e. (low complexity models are usually tried first. (via Occam's role). — But probly w.
O.F. the trials are not systematically for low complexity models.

.30 In O.F. (and probably also ALP) there is usually the spurious Option Yield (SOY) problem,
if one tests > 1 cand. — (which one would presumably do)
In ALP, I guess we only have 1 cand. — so no "SOY" problems — for CB=∞.
for CB<∞, we select out the cands of high pc, in the past. — but we are right them w. wt. ∝ pc's.
I'm not sure how (or if) SOY is relevant.
In QATM, the mean (pc / err) would seem to not extrapolate well into the future — depends
much on how "Similar" the future Q's are to Q's int. TSQ.

D14/01 **ID** CONVERGENCE: Convergence thm for Bags/QATM : Possl. Bugs? 478.00: 477.11 — 20G : 1K, 1000G = 50K

ABCDEabcde 10.0793 =

~~HAMPTON~~ HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 477.20: Another Q about t. proof of convergence rate for Bags & QATM: It would seem that the Pc's would dependont order in which t. Bag items (or Q/A's) occur. — Yet by defn, the true values are order independant. — But t. predicted value are order dependant: a bag element will get a much "better" Pc value if that Bag element is evaluated later in t. TSQ.

.04: 477.20 I want to be fairly sure that TM has acquires all t. needed "little concs", "unconcious concs", "Commonsense concs", — otherwise it will have lots of trouble later.
One kind of (partial) evidence that this is o.k., is that t. cjs's seem reasonable — (usully small) for TSQ's that I expect a human could do easily. Note, hvr, that Skinnerseqs are very easy, but tend not to teach concs needed for harder problems.

.10 Well start again on ANL! Pattern $3+4 : 7$ (Q: R1 R2 R3; A) $A = \text{Sum}(R_1, R_3) = \text{Sum}(R_3, R_1)$
Sum, sub, mul, div are 4 operators available that have numerical argts. TM knows No. v.s. Non-No.
TM knows "identity" for nos, also for non-nos. Since it is a Boolean funct, it can be usd for control.
TM could lrn .10 if "+" is t. only kind of problem.
If we introduce $5 \times 6 : 30$ problems, Sum (R_1, R_2) [R_2, R_1] will work ½ of time, mul ( ) " " " " "

This is still a enormous amt of compression & t. argts are 32 bit random nos!
The trainer could insist on 100% accuracy!

.18 —— A reasonable heuristic! Put together all Q's for which "sum" is correct operator.
" " " " "mul" " " "
.20 — Contrast in
so $n_1 + n_2$ (sum) v.s. $n_3 \times n_4$ (mul) In all cases, (sum, mul) —— $R_2 = (\times, +)$,
Conclusion: If $R_2 = +$ then sum (R_1, R_2) [R_2, R_1]; " $R_2 = \times$ " mul ( ). } call this operator on $R_{1,2,3}$; "O_1".

.24 This works fine! Next "−" we give many examples: only sub(R_1, R_3) works (not sub(R_3, R_1)!)
But before .24 is tried, TM tries O_1 — which does not work on "−".
We could get to .24 by indexing Q — common index for all "−" problems. So it would know they all had something in common. (.33)

.27 **SN!** I don't have to Get TM to solve all t. probs in a TSQ sequencially! If I can't think of a reasonable ~~#~~ lcost soln, then just give TM ~~...~~ a soln, & continue t. TSQ.
Periodicly review t. "probs that TM couldn't solve" & see if (as t. result of my Experience) I can find a way to Get TM to Solve them in a "useful way".

.30 — **SN₂** Befor trip, Get soln. to cubic, so I can try to ~~...~~. Give a soln in terms of invertable substitutions. See is standard soln to Quartic can be done this way
Copy pp 118-121 in Birk/Mclane.

.33 (.27) When we notice sub sometimes works & otherwise O_1 works! Use heur of .18 to decide "−" gives "sub".
Similarly, we learn "÷" → "div." Using same heur.
T. soln we have now is a complex conditional of ÷, −, ×, +, but it always works.

.36 Now look at (1+3): 4. [$R_1 R_2 R_3 R_4 R_5$] T. old ops don't work because of displacement. R_1 & R_3 are not numbers.
A person would notice that "(1+3)" looked a lot like "1+3" and perhaps try sum(1,3).

.38 Actually, I can give .36 after rite 1+3 (.10)

$\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\alpha^2}} dx$ $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\alpha^2}} dx$ $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\alpha^2}} dx$

mean (2) — now (1) — Okie

$\vec{p}$ (pilot removed)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

:00 : 478.40 : At this pt, I mite Get TM to understand "Hor. invariance", as a useful conc. for "( )" and for Irng "substitution".

Perhaps start out w. TM looking at 3+4 & "noticing" that 3 is "before" + & 4 is "after" +.

.03 → Another approach would be list a sequence of solns to probs in t. TSQ & then try to devise a suitable TSQ so t. solns will be of acceptable Least.

Then would be solns to 3+4 ; 5×8 .... 3÷5 ect ; Then (3+4) , (5×13) ect :

eventually 3+(8×9) , ect ... 4×(2-5) · (3+8)×1 ; Then (3+5)×(9+13) ect.

eventually, nested parens.

T. way I'm using Params; I may be able to get no ability if I only use say "close parent" — ")".

(3+5) × (3+(9÷3)) (3+5) × (3+(9÷3)) → 3+5) × 3+9÷3)) ← This is uniquely decodable.

.10 → Alternative to .03 would be a TSQ that I expect TM to be able to learn. — I then write out solns I expect

T. General way to Evaluate Expressions that use parens & an only binary funcs:

.12 & find a subexpression that once can evaluate and evaluate it; rewrite entire Expression w. new value: loop to & until no more.

Can I work this w. unary & ternary "Nary funcs" (using suitable notation).

Perhaps look at Kleene for Notation.

Should TM spend Much time discovering Alg. identity's like ≈ sub(3,4) = Mul(-1,(sub(4,3))) ?

Depends on how useful such activity could be. ABC DE. Initially, finding such identitys would be t sort of thing "AM" would do, because they were "interesting". — would we want TM to

Spend a certain amount of its time discovering things like those "identities"?

.20 Just how would these "Discovery's" fit into TM's general "QA" problem set?

One way would be to have a "special particular Q" that asks TM to find "interesting things" — which it presents as "A". How to get this response as a desired "hypo" for A, is unclear!

It may be that this "exploratory activity" would be a kind of "look-ahead", because it would not be immediately oriented toward "hypo A's". — (Tho it would be in t "long run").

Some tentative goals for early TSQ's: 1) ANL 2) soln. of one single linear eq. (including Simplify).

In solving these problems: (say a Number crunching Prob): Write out, (in English), t. hints or heuris that I would want to give to t. student. See whether these may be really adequate

.30 for t. task addressed.

old "Bad" pen:

So back to 478.60 → .38 : say I find (1+3) then (4+8).

Then do 3×9 ; Then must (8×3) be easier than ?

.33 Trouble is: (1+3) : R "good soln": If R_3 = + Then Sum(R_2, R_4).

R_{12345}

It really misses t. parens!

480.12 space

ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

Generalize

.00: 47240 : Out "UPDATEG ALGMS" I listed in "t. Report": (5 Algms)

Once one has "decided" that Q_ℓ is in "class i" à $\vec{P}_i$ should work:
If it doesn't, we want to modify $\vec{P}_i$ so it works for t. new (as well as old members of t. i class. If this proves difficult, put t. new Q_ℓ in a new i class: try to find a special soln. for it, then find a way to distinguish it (via "R") from the members of t. former i class. We want the "distinguishing" Algm to be as "short" as poss'l., so it will extrapolate (generalize) well.

.10 → This "training TM" is very much like training a child. If t. kid doesn't seem to "get t. point" we have to devise examples, hints, or "talking" to put him "on track".

.12: 47240 OK: Retournez à nos moutons! While 470,33 is a soln, it's not t. one we want — it will be useless for work w. parens. If I taught TM how invariance of t. operations (3+7) would be obtained indep of t. presence of parens!

.17 Well: try another track: Assume that TM somehow found 47012:
What cases would it need? (1) T. idea of "eval. of a stab expression." In Lisp, a sexp expression is any string, & any string can be evaluated. Consider any string of length > 1 that evaluates to a no. In Lisp, paran operations are built in. ANL is essentially "built in". So, in Lisp, I could try to get it to learn to solve a linear eqn.

.20 SN BIG Q: What were t. characteristics of t. SOL56 (problem set/TSQ) that made it easy (or seem easy!) for TM to usefully work on it?
— Well, t. prim. cases that I was using, seemed to match t. problem well — in t. sense that t. solns. to t. problems were (in terms of those cases) very "simple".

In ANL perhaps t. basic "equality": If 2 things are =, then they are substitutable, giving same evalns of expressns as before.

Ah! If TM ended up at 47,33, it would have to "Backtrack", if we gave more complex probs using parens.

.30 D17/01 SN Another BIG idea in t. General QATM system is t. ULTIMATE SI:
The "Q_{n+1}" is all of TM's previous History (Traces + anything else), & A_{n+1} is $\vec{O}^{n+1}$, t. new set of system ⊃ operators. We also have to include t. dependance on C.B(=cc). (we have a tradeoff eq. between pc & cc: $\frac{pc}{cc}$ = Max is one possl. expressn.)
→ It amounts to a general soln. to t. update problem.
We want TM to start work on the problem as early as possl., because it's close to a general, non-el soln. to t. Update problem.

.38: 476.1 Re: Applicn. of QATM to OZ probs: could we could have a "TSQ" of O.T.'s?

D17,01
ED

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00 : 480.40 : **THE BASIC IDEA** of TSQ writing is FACTORIZATION : All of know that I want TM to be Taught directly. I have to put into form of concs, à la standard concs. If I cannot factor what I want TM to learn, I do not understand it well enuf, à I must be able before I can usefully teach it to TM.

Years ago, I had this idea of "Euclidification of Human Thng" — to express all that has to be learned, in a seq. of steps of size < a certain threshold. So that's what I have to do. I don't see any way to avoid this problems — Tho perhaps many of the tricks I've been proposing for TSQ writing, are ways to help in Euclidification.

10 Many Times when I think I understand (procedure/problem...) I really don't, because I am unable to properly formalize it. "teaches" >> status.

A note on Euclid ☺: He had those axioms & postulates that he that were adequate to generate the whole of Geometry. Mathematicians for many yrs thought that he had done this. Much later, it became clear that he had not done this — That there was a fair amount of vagueness in how to generate proofs, etc.

→ How does "Euclidifn." differ from what Lenat is doing in the CYC project? ←

I really don't know enuf about CYC, to say. I suspect that Lenat mainly "teaches" by "telling". V.S. My teaching by examples (usually)

20 Much of early "AI" rsrch was an attempt to formalize human problem Solving. Perhaps I can use this rsrch to formalize it in a learnable way. Say take Slagle's symbolic integration — or perhaps even Moses & Macsyma & Maple & Mathematica. Tho these last 3 are "proprietary" — initial "macros" (Macsyma) work was not proprietary — A Thesis may be much published on Symbolic Math manipulation & prob Solving.

In general, a big part of Human "Understanding" of any thing, is the ability of the Human to express the thing in a small no. of concs that are familiar to the Human.

Symbolic Integration could be an easy area to start with. To start with, the student has a way to differentiate all expressions. (so test trial S's ... if trials of this sort are made)

30 Also a lot of rules w. unknown S's, & some special tricks like linearity, integration by parts, partial fractions & substitution.

It Should be possl. to write out "pgms" in English, w.o. using **Lisp**: An advantage of "English" is that even when they don't work (insufficiently formalized), they can usually be modified (not very much) so they do work "better" (more formalized).

I can probly write a pgm in English (or Logo Basic), that can do a fair amt. of symbolic integration. → Next: How to get TM to that state by "Long".
This is v to my "state" at 480.17

ID

$\int_{-\infty}^{+\infty}$ $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}}$ $\int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma^2}} dx$

raucous

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00 : 481.40 : Consider an earlier "state of knowledge":

TM can, for a certain set of functions, f_i, work the problem $\int f_i = ?$

We can teach it the linearity of "$\int$" i.e. $\int(af_i + bf_j) = a\int f_i + b\int f_j$.

So it could integrate any sum of coded functs, if it could integrate the functs.

A meta-rule: try to get the integrand into this form, if possl. (a sub-goal).

Say TM "knows" $\int n f(x) = n \int f(x)$ if n is any no. & $f(x)$ is any funct'n.

How could it learn this? Say it knows $\int \sin(x) = \cos(x)$ to teach it that $\int n\sin(x) = n\int \sin(x)$;

.08 Say we give it many examples of $\int n\sin(x) = n\int \sin(x)$ is

After it runs .08 would it run $\int n \exp(x) = n \int \exp(x)$?

.10 Perhaps give it examples of $\int n f(x) = n \int f(x)$ w. various n & f, to start: —

would it run good rule?

→ Note: Learning Differentiation is about same as/run Integration

Except that there are fewer rules! (Also they always converge!)

.14 Note: In McC's way of defining recl functs in Lisp: used only for single recursions,

We can always start w. known $f(x_0)$ & generate $f(g(x_0)) = h(y_0)$ & do recursion

$g^n(x_0) = x$; then use $f(y) = h^n(y_0)$. It may (i.e. no n exists

$\ni g^n(x_0) = y$) which is a nasty property of recl functs. — Humph "forward"

.19 Way of implementing recursion is faster & uses much less memory! → 483.10

.20 (Def/dr) If Q is of the form $d/dx(f_1(x) + f_2(x))$ then A is $d/dx f_1(x) + d/dx f_2(x)$"

We have a rather complicated "Recognition rule" for Q. We assume TM already knows it.

"defn" of a function is it would "consider" such a Recogn rule.

Note that Recogn. rules can be rather complex, so that determining if

a given Q satisfies them, can be a diff'l task. e.g. $f_4 + (f_5 \circ f_3 \circ g)$ is of the form

$\beta_1 + \beta_2$ with $\beta_2 = (\beta_3 + \beta_3)$, is relatively simple, but we can have diff'l cases

of complicated nesting.

Also TM may have to manipulate Q a lot to get it into a form that is "Recognized".

Since there are many "recogn. rules", it is not always clear as to what Recogn should be used.

What specific sub-goal should be sought.

→ Some "Recogn rules" imply an ∞ of "easily applied" Recogn rules. e.g. a

.29 recursive recogn rule might be borrowed this way. (.33) might be a case.

30 One subgoal in induction is to find (or generate!) a recogn rule for Q.

Certain induction rules generate R rules & the associated $\vec{P}$ functs.

From these ideas, it may be possl. to see how induction works for differentiation & for integration.

.33 (.29) → A very wierd Recogn rule!: "If Q is of form $\int f_1(x)$ and $d/dx f_2(x) = f_1(x)$ then $A = f_2(x)$"

By (str)ing all functions one can "recursively" define $d/dx^{(-1)} f(x)$ $(= \int f(x))$

D19.01 Consider differentiation! E.g. look at $x^2 + 17 + e^x + \sin x$. It notices that

it consists of functions separated by "+" signs. This is strongly suggests

"AND" net ('Divide & Conquer'). Lets try to factor this: One factor

that would be otherwise useful: A fast recognition of that & things below.

D19/01
ID

RECURSIVE FUNCTS in AZ 141 .10

483

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00:482.40; T. "+" signs were "Functions" & could (perhaps) be different handed (recursively).
So to main setup is "Differentiate": d(A+B) → dA+dB becomes a recursive call.
.02 ≈LISP: If X = x^n OR sinx OR cosx OR α+β then n x^(n-1), cosx, -sinx, dir α + dir β, resp
McCarthy had a nasty way of doing this.
Use "CASE"? No — that's IF on n go to? Yes...

The info content of .02 is an ordered list of If, Then pairs.
So for Factorial(x) = fact(x): If x=1 → 1 else x·fact(x-1)
.04 If α=sinx then cosx else If α=cosx then -sinx else α = U+V then dirU+dirV. → 485.02
.10; 482.19 SN On Recursive Functs in AZ141:
In AZ141: How is defining f(x)=a if x=b? (i.e. f(b)=a). Well, there is this special 2-input function; f: inputs x, a, b: when x=a, output=b; undefined otherwise Or inputs x, a, b, f_g (f_g is an arbitrary funct. it can be recursive)
So: This funct has 4 inputs, one of which can be, itself or a funct of itself
.19 f is being defined. f's inputs are x, a, b, f(g(x)), or h(f(g(x)))
.20 So f is defined by a, b, h(·) & g(·). It differs from normal functions in that h & g are function names; may be any (arg. functs, — or, any funct,
.23 [that has 1 "dummy" input.]
→ Note these recursive functs cannot be very complicated; their PC ↓ very fast w. size.

D20/01 These dummy variables (.23) normally occur in defined non-recursive functs as well.
An apparent difference between rec. & non-rec functs. It is possible to create a "1 use" non-recursive funct (a composition of previously defined functs) w/o using dummy vars. —
.30 [redacted] My present method doesn't seem able to do this w. rec. functs.
To use a rec. funct., one has to define it first, Then use it.
If the g & h functs of .19 R have been previously defined as functs of 1 variable, the rec. funct defn. needs no new dummy vars.
Look at 482.14-.19: we need to find n ∋ g^n(x_0)=x; then f(x)=h^n(y_0) → 484.00
(We might save time (particularly if h(·) is expensive) if we first find n in g^n(x_0)=x →

D20/01
ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00:48340 : If we can't find n, we don't have to waste time finding $h^n(y_0)$.
However, in Lsrch, if we don't have time to find $h^n(y_0)$, we are "nowhere" — so we mite as well not have done the calculations! — so for Lsrch, this is no "shortcut".
So, in Lsrch, we will alternate betw. evaluating $g^n(x_0)$ & $h^n(y_0)$.

Still, the defn of rec function AZ(4)15 a "kludge" & I don't feel comfortable w. it.
It is too much of a "special case"

Actually, we can define rec. functs 2 somewhat different ways. In one, we use $g^n(x_0)=x$ to determine n, in another we use $g^{-n}(x)=x_0$ to determine n.
.10 Since computation time for g^{-1} & g can be quite diffrnt, this could make a lot of diffrnce!
Sometimes we may not know how to compute both $g(\ )$ & $g^{-1}(\ )$. — Also one may be multiple valued! If so, we would save lots of cc by using the other (inverse → funct, funct → inverse).

Hvr, for the evaln. of rec. functs, all we really need is the info content!
i.e. x_0, $f(x_0)(\equiv y_0)$, $g(x)$ and $h(x)$. If $g(x)$ and $h(x)$ are not normal functs of 1 var, etc, they will have to be suitably indexed/labeled —
.20 telling which args are the dummys & which are params calculated externally

Of some interest! I have considered only 2 ways to define functions:
Composition & Recursion: There is at least one other: $F_n(x) = g^{(n)}(x)$. } This can also be done by recursion
also $g(x) = f^n(x)$ & (n^{th} derivative of f(x)) (n < 0 for integrals)
Also functs defined by finite or infinite series.

On my p 482.14–.14 defn of a Rec funct: It may not fit "x!".
.28 $f(x+1) = (x+1) f(x)$ $f(x) = x f(x-1)$, $f(x) = x f(g(x))$ $= h(x, f(g(x)))$ ‖ $f(x-1) = (x-1) f(x-2)$, $f(g(x)) = g(x) f(g^2(x))$ → .30
In LISP; f(x) = If x=1 then 1 else h(x, f(g(x)))
30:.28 $f(x) = h(x, f(g(x)))$; $f(g(x)) = h(g(x), f(g(x), f(g^2x)))$ f
$f(x) = h(x, h(g(x), h(g(x), f(g^2(x)))))$

$f(x+1) = (x+1) f(x)$; $f(1)=1$; $f(2) = 2$ → $f(x) = h(x, f(g(x)))$

$h^{-1}(x, f(x)) = f(g(x))$ so $(h^{-1})^{(n)} = f(g^n(x))$ find n ∋ $g^n(x) = a$; then $f(a)=b$
h^{-1} defined by ...

.38 So apply h(x, ·) to   Best look at Kleene's defn. of recursive functs.
Hvr, it is clear that there will be no great trouble in seeing how much (485.00)

D21/01

ID

$16^2 = 256$ $225+30+1$ 41829

Oooo..

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

11, 31, 41, 61, 71, 91, 101, 131, 151, 181, 191

Info (pc) theor is

.00: 484.90 : Info (pc) Theory is in a nec. data.
So: Get back to this when we get back to Camby. Looks very non-central. 17^2 289

.02: 483.09 : I was trying to get differentiation rule in "Heuristically usable" format.

Consider foll: [redacted] "state of relevant knowledge":

TM knows dir (sinx) = cosx, dir (x^2) = 2x ; dir($f_1(x) + f_2(x)$) = dir $f_1(x)$ + dir $f_2(x)$. How did it learn this last?
Also, what form did this "knowledge" take? For t. first 2, it could have been given!

Q = dir(x^2); A = 2x; Q dir(Sin x), A = cos x. It has learned that x^2 & sin x are functions!

Q = case(function, x^2), [redacted] A=T.

Q = case(function sin(x)) : A=T.

D22/01 (SN) perhaps dir $3 \times \cos \to 3 \times \text{dir} \cos$ would be easier to learn at this pt.

3 is "no", "cos(x)" is function. so dir(no. $\times$ function) $\to$ no $\times$ (dir funct)

Look at them in Polish: dir($\times$(3(cos x))) $\to$ $\times$ 3(dir(cos x)) dir 3 no. funct $\to$ 3 $\times$ no. dir funct.

[redacted] Symbol, mul, no, funct

The conc. of "exchange" or "commute" could be Useful. It works nicely if there are no parens.

.19 (NB) In general, when TM sees a sub object, it classifys it if possl. Sometimes there will be ambiguous parsing of a Q, so > 1 method of classifying t. parts,
I parens are used, then mis parsing may be impossl.
At first I will teach TM several defns, & it will try to use these "classifics" in induction.
Later, it should be able to invent its own "classifics" (≡ "Terminology") for induction.
One main goal of this "Classifn." is t. "Recognition" part of Q [redacted] processing.
One early applicn of .19 could be ANL: $4 \times 5 \to$ mul(4, 5) is immediately parsed (recognized)
as $no_1 \times no_2 \to mul(no_1, no_2)$ | no_1 "operation" $no_2 \to$ mul (no_1, no_2) doesn't always work. O.K.
Writing $3 \times 4 \to 12$ is bad, tho. "$\times$" should be written like any other binary function.

So mul(a, dir f(x)) mul(a, dir f(x)) = dir mul(a, f(x))

On Notation "F(x)" means "t. value of f(x) for arg value x"
What I want is more like f( ) the function itself not any particular value of it.

AZ141 can [redacted] (& does) define such objects, using dummy variables.

$\sin(x) = F(\sin(x), x)$ — i.e. it is a function of t function sin( ) & t. arg, x,

Also Note: 1) In ANL: I want "eval(3+7)"
2) If a=b this means that eval(a) is identical to eval(b),
3) I can do Algebra 2 ways (a) t. way I've been trying to do: i.e. rigorous means: —
doing Alg. in rigorous form.
(b) normal, sloppy Algebra, learning as by humans; this may be easier to devise tsd's, but t concs used are related to R.W., & often obscure human needs/wants.

SPEC 486.00

D22/01

REV

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : Some very impt TM ideas:

.01 1) 414, ... eq. What TM is trying to Maximize. { ≡ (a priori pr of O^n · a post pr of O^n.) T. assoc. = error convergence theorem, & what it means for QA TSQ's (i.e. future QA's could be in a new direction so we can't extrapolate much! → (See 998.00) } also see (7) (9)

.03 2) The cc modification of 1): Note (12) (.25) ↓ (.03R)

3) The conc. notes a basis of TSQ construction (see (8) ↓ (s. functs as well as d-functs)

4) T. idea that if "all info is in PD" Lsearch is optimum: How P.D. must be modified during Lsearch, for this to be true (critical f.N.m Sol(89)).

5) AZ141 method of assigning pc's to functions based on composition & recursion (recently: 483.10 ff)

6) (Maybe) factoriz. of O^n (in 1)) into Recognition & $\vec{P}$ parts.

.10 (7) In 1) t. expression of Probability functs by 3 input units & realization that we could implement it via AZ141-type functions (i.e. LISP-like functs) ← ≈ Chaitin's idea of "request another bit" as assoc. w. concurrent 3 input units.

(8) That stochastic (as well as Determ.) functs can be found by TSQ's guided by Conc notes

.13 → (9) The difference betw. t. a prio of functs used m & t. a prio of trials (based on extended "context") of trials to Maxe. (1). (I may need to reclarify this in my mind). (1) ≡ (.01) ↑ This results, ultimately in what seems to be optimum update system. T. a prio of trials may use also Have info from General Science, Math, etc — that the trainer knows, possibly even "cheating"

.15

.16 (10) Lrng Nat. Lang. by first lrng Algebra (or some other domain) then, (according to 8/9 below) Nat. Lang. & TM's internal lang. — about that Domain.

↑ 11) TM can use logical reasoning! this can be taught or "programmed in" "wired in" (see 487.32, (9)

.20 Also 486.13 – .15; (486.16 – .22 R)

.21: (.03) ↑ 12) T. idea that a P.D. has also a cc param! Note For each CB, we can have a different P.D. — this is t. pc available for t. agt. w. cc < t. Given CB. — We can either have t. same CB for all agts, or have a different specified CB for each agt.

.22R → 483.12

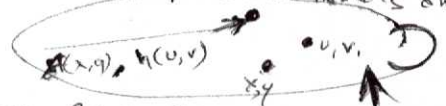
499.00 – .15 is a good discn.

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 (Spec 485.40): Advantage of Rigorous approach: Its probly better, simpler solus.
" " Sloppy " : It uses Human Heurs that may be more powerful than those of (.00).
For beginning: .00's may be simpler, — Tho getting Algebra into rigorous form may be diffc.

In the case of Sin(x) There is ambiguity: it could mean the value of sin for argt x or the function sin(x) w. dummy variable x. Usualy this is easy to resolve by "context". If dir sin(x) is asked for I would assume x to be a dummy; but we could get w. cos(x) — in which cos(x) can be a function & x can be a dummy or an actual argt.!
I think Joel Moses may have run up against problems like this in developing "Macsyma". Perhaps look at MapleV notation for problems to see how these
.10 alg. notations can be unambiguous. Tho it may well be that the techniques they use are "Kludgy" because they try to stay close to human ways of perceiving things —
.13 or just Historical ways. (.32)

.14: 484.38 on rec funct defns; This method at 482.14–.19 find [redacted] $n \ni g^{(n)}(a) = b$: then $f(x) = h^{(n)}(x)$
This can be implemented by simple Do loop. Hvr, Do loops can also use $\alpha > \beta$ as a stop criterion (not only $\alpha = \beta$) so they are more powerful. The equivalence of certain kinds of rec. functs (say prim rec functs) to certain kinds of Do loops may be [redacted] perhaps shown from this kind of argt. Maybe Roza Peter used proof of this kind.
.20 Anyway, I think most, (if not all) rec. function routines, can be defined by Do loops.
I'm thinking of rec functs defined by rules like $f(x, g(x)) = h(x, f(x, l(x)))$
or: value of $f(\cdot)$ at $g(x)$ and $h(x)$ determines f at [redacted] $r(x)$
• $f(r(x)) = h(f(g(x)), f(h(x)))$. So given f at 2 [redacted] pts of a [redacted] line, a value on a third pt. can be known always then. (How this defn way not be causal)



~~perhaps f(,) = f(x, y): f(g(x), h(y)) = r(f(x,y)) (no!)~~

.30 ~~f(g(x), h(y))~~ $f(g(x,y), h(u,v)) = r(f(x,y), f(u,v))$
Look at defn of Ackermann funct. — Then try to express it as nested "Do" loops.

.32 (.13) In much of heuristic reasoning, logical [redacted] Reasoning is used — In simulating these heurs, this logical reasoning should be used. We can assume TM knows how to do this, i.e. we can (presumably) teach it to him (or write it in); (see 486.13–.15 + 486.16–.22)R.
Any means we use to get TM to ✓ of 414.... is "legal".
.38 This (.32) suggests that Early work on A.I. — (Heuristic Reasoning, etc. etc.) could be useful if developed in a prob'listic way.

D 24.01
ID

Hybred Vigour.001

.00! [SN] **HYBRED VIGOUR** (GOR?) ! Explaind as (a) Cross of 2 function trees that give diffrnt solns of General problem. As opposd to (b) cross 2 solns that are very ~. In (a), possy of
.02 new, interesting trials. In (b) much less promise of "mix of "new ideas". (Unlikely that Earth formd this way! Since all planets are in Ecliptic)

Possb Hybred Vigour (for more likely production of Life) by Collision of Galaxys formd in vary different ways. (for Gunkel). Velekovsky's "Worlds in Collision"?

Also ∴ try to save children w. genetic diseases that seem ○ otherwise (O.K. But for "Bad" diseases that Involves several Genes: Difflt to prevent from Genetic Organ.)

Cross of Chimp (Gorilla) and Man? Seems likely this would have Occured.

Hybredy should occure at near optimum "coded" level of ∂crn of cands.

So mix is of "almost random" strings

.10

.11 So the idea of .00-.02 does seem V.G. Hybreds give larger σ², but they do give real possy of very superior offspring.

IMPT Z

.12 (486.22R): More detail on idea (a) (486.13 ff) [there is proby much previous Stoff written on this] ← IMPT — See 499.00-.15 for a good discn.

It is clear that the corpi for the 2 d.f.s are (usually) quite diffrnt. (Now) note that in the second case, the data can be "orderd" — that TM could recognize that more recent attempts to do Maxzn are better. Also, in the second case, TM (ideally) knows exactly what "Maxzn" means! It might be expressed as a recursive defn more easily implemented as a loop. I.E. "pick max of these trials the up to Σcc = cB." (Still, this is not the whole (complete) defn of "Maxzn") → 492.01

20

.21 ; 3 approaches to TSQ design:

.22 (1) (Formal): write examples (Q/A), "Or", solns in Formal Math. Do this for a fair section of Algebra. Then look at solns. See if CJS's are too large! If so, look for linking concs; Draw Concnet. — Try to find examples of the concs to bridge the large CJS's.

(2) (Informal) In English. Write examples (Q/A's) & solns (rules) for same corpus as (1). Make concnet, then try to formalize solns (Q/A) as in (1). Advantage of English is that it automatically gives "soft" solns, that suggest good formal solns.

(3) 487.32-.40 (487.28 → .40 in particular) Use of Old AI heuristic technique + trying. Should work.

30 One of the above ((2) is a super case of (1)) so should work!

.31 So (2 or 3) ≤ should work.

.32 [SN] Lenat's "AM" was run ~ 1973 28 yrs ago: $(1.5)^{28} \doteq 400K$ so we are now 400K times as fast!

However, even if we used a good such method, identifying a useful conjecture would be hard for a human — But if AM says its a V.G. conjecture and gives Reasons, this would narrow things down considerably!

Perhaps try to read Ken Haase in "Cyrano": perhaps improve it & run it on fast Machine. → 489.18

.36 Get it on the Net. I think his Thesis is on the net. (Difficulty finding it or even Ref to Ken Haas! Try Kenneth Haas.

I'm interested in the structural details of "AM": so I can try to get it to do its "thing" using probabilistic Srch. — Also have a goal-oriented part that looks for new heurs. [NB] I did regard AM as a kind of "Version" of S89.

Carano.

29 73 2^{18} 256K ~400K

449.00

D 26.01
ID

489

26
17
9 : 9pp.

ABC

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

:448.40 ; So try (488.21-[redacted].31) :

First: Prelim trials for notation: "$n_1 + n_2$" means trails w. n_1 & n_2 random integers
"ev(n_1+n_2)" is t. evaluation "n_1+n_2" — it will normally be a number (integer).

n_1+n_2 • : ev means Q is a str n_1 ; it is ev(n_1+n_2).

For most early stuff I just have to write "Q." "A will be "eval "Q"."

To start off, t. "numbers" & "non-numbers" are clearly (indexed / indicated) on t. symbol itself! So if TM is given t. Q "3+4;" "3" has an associated byte, that tells (among other things) that it is a number. "+" also has an assoc. byte, that tells it is a "non-no.".

T.M. [redacted] is taught that objects having t. property "number" & has t. property "no." & TM puts this in to in its "property list" (?). (The "property list" is deterministic: I'd want a pc for a properties that have been evaluated this pgm. If certain pc's are used by several objects, we should want to update them & store them in a comm'ly accessable Register—

T. Format Q: [redacted] T. Six digit chars are left justified. All characters are of significance, Perhaps for each Q string, there is an "end" indicator — so Q's form a prefix set.

→ A Built-in Human Heur! In Q, nearby symbols have more likely hood to be related.

:(488.36) Kenneth Haase: http://haase.www.media.mit.edu/~haase.
Contains his [redacted] thesis, Cyrano. Looked "Kenneth Haase on Google.
Thesis on [redacted] Web, is in HTML: Originally in Latex. Perhaps got Latex from Haase?
HTML file is many "Nodes". ~86 "nodes. ~20kby/node (in HTML).
HTML file is many "Nodes". have to be individually downloaded.

Perhaps Get Ken interested in Getting E ≒ "Cyrano" to work on QATM

Haase (in "work by others") mentions Wei-Min Shen
W. M. Shen: At his web p. are refs of interest! — But to read Haase on his ... work: it is ≒ reproduction of AIM I Paper.

→ Another "common sense" idea humans have, is idea of "counting" — T. Cardinality of a (finite) set.
That is, "card" of a set is indep. of order of [redacted] elements in set.

→ Haase's discussn. of limitations of "Cyrano, gives one some idea of what it does.
He says Cyr. does new things & uses of defns to mean new, "But does it do "inversion" or "Modulas" of previous decade. — It would seem that this "language" is very limited ...! — but isn't it missing full "Lisp"? At any rate, I should look at T. limitations of Cyrano, & make sure that QATM is not limited in those ways (or any other ways that Haase may suggest)
Also, I want to look at [redacted]KEENE, to be sure AZKI is truly "universal".

One way to do 488.24-.31: Just write a "naive" TSQ & see where TM gets "stuck":
Then try to (invent / discover) (concs / heurs) that TM would need to overcome those difficulties:
On "substitution": "a = b" iff in any expression equation remains true after subs(a, b, eq)
(or, instead use string of "subs" ("subs" can be further ref. to "a" : "eq") → 495.34

Another approach to TSQ's: use Polish or ≒ Lisp as range/ya of systems, so it's always clear how to parse Q. (But just how is this different from the previous stuff? TM's own heuristics are always in ≒ "Lisp". T. difficulty is that [redacted] inf. forgs., Q can have any form. → (spec. 490.00)

→ Haase's pgm may have gone beyond AIM: but look at Haase's comments.

D 2 3/81
ID.

490

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

(489.40) What is being proposed is that 1. Early TSQ be in "easily decodable form — easily parsed. So, given sum(mul 3 4(8), 1) TM would be able to do it easily,

A problem might be : solve(x, sum(3, x)) Q = Solve x, (sum(3, x) = 10) : to solve f. eq. for X.

In.01 Eval (sum, (mul 10, 8), 3) would be easy to do as a Q: w.o. eval, the meaning is unclear. In "Lisp" it is clear; but otherwise one would have to write "eval" or "Quote" before each expression. In Lisp, if "Quote" does not occur, Eval is assumed. One could have Quote be assumed unless eval is given — which is intuitively more reasonable, but probably involves more typing ("Since" Eval is much commoner than "Quote").

The I'm not sure "Quote" default would work! : Say I would Qu, Ev, S.

In Lisp, is QS, → S. ; In Q default, it would be Ev S. which amounts to same thing.

SN : Re: Infix ("normal") Alg notation: Main value is the apparentness of associativity for mul, subtr. Also assoc of sum sub strings & mul/div strings. Few functions associate; few commute — But use of + or × notation in infix helps in Pol. + & × both commute & associate, so a set of "+" terms form a "bag". In general, any string of + − can be rearranged to "all +" by changing sign of "−" terms — Similarly w. × & ÷ :

There may be tricks in Polish ! sum(sum(sum a b) c) d sum Sum Sum

or sum sum sum a b c d There may be simple rules to change order of symbols in a string.

sum (c sum a b)

sum d sum (c sum a b) Sum sym dc sum a b Sum Sum.

Well, say TM knows Polish notation. It is also familiar w. 1+ binary/ arity operations.

sub(0, x) & div(1, x) are commonly used binary ops & are given special names, as unary ops.

N.B whenever TM defines a new function, its "arity" appears in the "property list" of that function.

SN Do a quick try for linear Eq. solvn: It knows in Lisp to start!

Q : solve(sum(3, x)) : A -3 Solve (f(x)) means : "solve f(x)=0 ".

sum(3, x) = 0 : sum(x, 3) = 0 (= x; sub(sum x, 3), 3) = sub 0, 3 = -3

Example of " put as much that is known on RHS. ", simplify LHS than put more on RHS than was on LHS, etc.

Try to simplify LHS by dr. or sub.

Concept of "simplicity" (may be built into System) → Automatically : Not sure. Must solve of the probs use H.O. or 02 methods) — A heur is: "Simpler" (eqs/probs) are usually easier to solve.

More directly : Solve (sum(n1, X)) → A = sub(ϕ, n1) : sum(sub ϕ, n1) = neg(n1)

sum(n1, sub ϕ, n1) = ϕ. for any n1 : n1(sub ϕ, n1) = ϕ TM could run Pol. equivalent (sum n1 neg n1) = ϕ

D29/01
ID

GA.00

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00: 490.40: SN If GA can work "Hard probs", it should be able to work on SI. ~~[scribbled out]~~
Superficially this may seem impractical, because each "trial" ~~[struck]~~ for this Meta System, takes an enormous Am't. of time. Hvr.: Perhaps Not! There are certain Aspects of S.I. that get much faster F.B.: They are:

.05 1) Mutation & Cross methods as a funct of problem type.
.06 2) Initial population, & pop. Size " " " " .
.07 In 1): we have a bunch of Mut & Cross methods, & we try them out occasionally, for each problem. If they seem good, we try them some more: We can ~~[struck]~~ test them in. more ΔG/generation (or ΔG/part of Generation), we ~~[struck]~~ can also ~~test~~ Mut & Cross of t. Mut & Cross....
10 Sample sizes, (run sizes) for these Experiments ~~[struck]~~ should be determined by Sequential Analysis (Wald).

In 2) we can save parts (or even whole) populations (or models of populations) of all probs. t. GA has ever solved. We can then sample these populations at various Gore
.13 levels — & try them in t. present problem.

.16 → In both 1) & 2) we look for relationship betw. t. initial problem dcrn, & t. successful
.17 choice Ranges in 1) & 2).

In t. problem of .16 - .17, we have a single evolving population that continues as new problems are given to t. system.
20 .05, .06 & .16 - .17 are t. Main Methods of S.I. needed by GA.

~~[scribbled out]~~ A Main problem is not t. above, but method of Representation of t. problem (including Fitness function).
The "γ" (or other monotonic fnct of t. original fitness funct, can be continuously "tweaked" as in .07 - .13, but there are other aspects of Ft. Fn that are also impt. — Many times, t. fitness funct is really a vector, ~~[struck]~~ $x_1 \cdots x_n$, & we use $\sum_{i=1}^{n} f_i(x_i)$ as a all over Gore. We can then wt. t. f_i's i.e. we like, which is usually more complex than "tweaking one γ".

30 T. "representation" of t. problem is usually intimately linked w. t. defn of t. fitness funct.
Koza seems to have found a somewhat General method of representing problems as "Networks": (Mind full of G. kron & TAPM Strings). I don't yet see how it wd. deal w. Nat. Lang. input. Could we teach N.L. t. same way I plan to teach QATM Nat. Lang.? → Thom?

Anyway, T. way .05, .06, .16 - .17 work: we are always taking random samples of mut. cross, "initial populations", when we find ~~[struck]~~ any that are good, we try to correlate this info w. t. initial (Representation) problem dcrn, or w. t. (Representation) problem dcrn. selected by t. user.

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : TM could lrn "Facts"; "Genzns of Facts"; "Genzns. of Genzns...."

So initially, TM could learn a set of nos. & relations betw. them; & Genzn. these relations. At first it has a special name for each no. Later it could lrn binary notation, so it would know about numbers that it had never heard about before.

.04: 488.20 D30/01 : If I actually used the Pd. implied by the code for O^n, as a prompt for searching for O^{n+1}, (This would not use info about Q_{n+1} or A_{n+1}, so clearly it would be very "inefficient") — how bad would that be? If using rather large machine, I could get the system to a pt. (using suitable TSQ) to where it could usefully work on the problem of "updating" i.e. $Q = O^n$, Q_{n+1}, A_{n+1} is any other "contextual info"

.08 .09 $X = O^{n+1}$. I would have it MADE!

.10 In the TM system of .04ff, the update Algm. is simply Lsrch! The Big problem is devising the TSQ.

.13 So if I just wrote a naive TSQ for TM, how would .04 react? If some CJS's were

.14 "too large" than augment the TSQ or "wire in" (otherwise useful), needed concs. By looking at code of initial (unaugmented) TSQ, (by "code" I include info on how I think I discovered wrote it), one may be able

.16 to tell which concs are needed for acceptable CJS.

.17 .04–.16 seems like a reasonable approach to TSQ Design & (perhaps) to ATM design!

The Q.A's in math notation for O^n's in English, to start.

In .14–.16, if we need certain flexes to get CJS: we can hypothesize a TSQ that would give those flexes. These flexes are "Meta rules". In .13–.16 I wasn't

.20 Considering using flexes of this kind — but it would seem that they could be usefully employed!

If we do use flexes of this kind, I guess we are back to the SSG! If such flexes are not adequate for the "complete" TM, we use the SI trick of .04–.05 to get the Optimum TM!

.26 ABIQ! Using Lsrch to find a good O^{n+1} would seem to involve testing all previous Corpus — which would make Lsrch not practical. However, in normal induction by Humans, this is somehow avoided — Perhaps by Breaking up corpus into parts by the "Recency" Function. Anyway, In doing .04–.25, try to discover just how .26 is dealt w.

.30

.31: SN Use of Symbolic Integration as TSQ! If System Got to be very Creative, it would have to use this as a "free prompt" to suggest hints.

.33 SN In APL, the trig (& possibly hyp trig) functions are expressed as functions of 2 args: the first arg is an integer that gives "name" of func. The result is (I think) the identity or trig. take a very simple form.

$\tan = \frac{\sin}{\cos}$: $\tan^2+1 = \frac{1}{\cos^2} = \sec^2$; $\tan^2 = \frac{\cos^2}{\sin^2}$ $\cot^2+1 = \frac{1}{\sin^2}$ $\cot^2 = \csc^2$. $\sec^2 - \tan^2 = 1$

$\tan^2 + \sin^2\cos^2 = \sec^2 = \frac{1}{\cos^2}$. $\sin^2 + \sin^2\cos^2 + \cos^4 = 1$ $\sin^2\cos^2 + \cos^4 = \cos^2$

1/1/02

ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

.00 : Re: Mathical, Logical reasoning: TM will have to lrn most of this somehow, if lrng other things:— Via suitable TSQ's. In the Elementary Algebra TSQ, there is much Elementary Logical Reasoning that, if used, will ↑ pc of result enormously. So it's very desirable that TM know this. — How much should be wired in & how much lrnd, is unclear at present

Some "reasoning": Q: n_1+n_2 A evl (n_1+n_2) :

n_1 & n_2 are nos. A is a nw. T. only functions going from nos to nos are sum, sub, mul, div.

So we could actually try sum n_1, n_2 & sum n_2, n_1; We also tried $\binom{\text{div}}{\text{mul}\ \text{sub}\ \text{sum}}\binom{n_2}{n_1}\binom{0}{1}$ & $\binom{\text{div}}{\vdots}\binom{0}{1}\binom{n_2}{n_1}$.

So $\sum_{n_1,n_2}^{n_2,n_1}$ doesn't have an enormous pc. 2 out of 32 so $\frac{2}{32}$ pc $= \frac{1}{16}$. $4 \cdot 2 \cdot 2 + 4 \cdot 2 \cdot 2 = 32$

.10 to lrn, Inc $(n_1) \to \sum_{n_1,1}^{n_1}$; dec $(n_1) \to$ sub $(n_1, 1)$ ($\frac{1}{2}$ pc of Inc)

Now TM has to distinguish betw 4 + 2 Inc problems & dec problems.

To distinguish betw these 4 pairs of problems: T. presence of + − × ÷ Inc, dec. is adequate.

.13 Is TM ready for $n_1 + (n_2 \times n_3)$? I'd like TM to realize that $(n_2 \times n_3)$ can be subs by evl $n_2 \times n_3$, to obtain "simpler" expression. This can be done (for nested expressns): either way:

.16 1) (α) replace directly evaluatable expressns by their values. Recurse (α) 2) (β) Take largest parent expressn.

Its evalnis obtained by doing β. ∃ a recursive defn., using a stack: ①(.16) doesn't need a stack.

(SN) on "Logical reasoning": I assumed that Logical reasoning rvrs usually close to pc = 1 — so at zero cost — a rest of pc was not relatively large (very little srch needed). Hvr., it may well be that "log. rsng" does require much srch, so one doesn't really save on cc by using it to "narrow down possl. cands." Hvr., usually breaking a task into 2 parts like Pat's: a Logical reasoning part, to narrow things down, followed by a simple L srch (or exhaustive rote srch), is better.

One way to get the recursion in .13, is to have TM lrn many evalns. w.o. recursion. Then it looks at all of the solns. & finds a common recursive form for them. One thing that facilitates such discovery, is noticing Common Sub. expressns. For each of the simpler problems, there are many solns, & only a few facilitate recursion (Pat: much use of common sub. expressns, would narrow down the low level "solns. that were to be integrated w. recursion.

.30 So try .13 also $(n_1 + n_2) \times n_3$ mul (sum n_1, n_2), n_3

$n_1 + (n_2 \times n_3)$ Sum (n_1 (mul n_2, n_3))

.32 .34 .35 .36 (So the n_i in param, are combined; T. result is combined w. other n_j ← Pat: works in both cases.

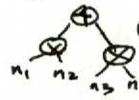
The traces of .33 — .34 are the same as those used in normal search for functions.

spec 494.00

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

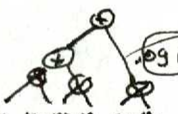
.00 : 493.40 : Unclear as to how to ~~formalize~~ 493.35–.36 : "The nos. betw. the parens are made the args. of a function". This is in "English": so it has the desired Ambiguity. Perhaps leave it that way, until after, I do the recursion (in "English"). In 493.35–.36 "The result is combined with other n_j". — This amounts to ~~a kind of~~ "Substitution".

.04 Try $(n_1 \times n_2) + (n_3 \times n_4)$ Since there are 4 vars, this can be a long srch!

.05  (.05L) A possible soln. to .04: Do 493.35–.36 — either way (n_1, n_2 first or n_3, n_4 first)

(N.B.) the n_i have to be betw. (,) in that order.

Try $((n_1 \times n_2) + (n_3 \times n_4)) + (n_5 \times n_6)$ — which is in the spirit of ".04". woops! nested parens needed.

.09  (.09L) How TM interprets the "nesting" is unclear; It could pair the parens by:

.10 (( )+( ))+( ) This is "wrong" of course, but by replacing 2 of the parens, it does get the right answer."

.12 So: What is the problem? To devise a Meta lang. to dscrb. functions & "features" of functions, So that (a) The non-recursive functs for ANL are easily lrnd.

.14 (b) " recursive " " " " " " .

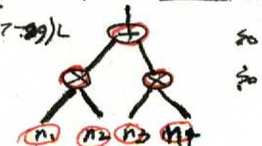
I also need a way to dscrb. the problems "Q" & their "features" & how they may be correlated w. the features of the lang. of .12–.14. Actually, the lang. the dscrb. "functions" may be already available. What I need is a way to dscrb. features/properties of functs & their Dervns ~~[crossed out]~~ (trees).

.20 This ANL approach ((infix → Polish) translation) discovery, seems diffclt. : Perhaps try a diffrnt. TSQ.

(.22) Note (.09L): uses (.05L) as a subgraph — in fact, usually problems of this sort use previous solns. as subgraphs. — so PC's are quite hy! (easy to get lots of cases (Examples)). — But still it has to look at the Q & figure out what Graph (function) to use; so I still need a good [redacted] lang. to

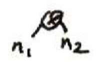
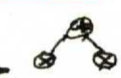
.25 dscrb. Q & its "features/properties".

Well, we could draw lines, making correspondences between symbols in Q & parts of the function Graph.

.27 (.27–.29)L  So any the Graph edges, & the parens or unassigned edges : 4 parens! Adding another may be appropriate. So each paren pair ⟷ a pair of edges. Perhaps regard the parens as "graph" instructions!

.29

.30

.31L $((n_1 \times n_2) + (n_3 \times n_4))$ $(n_1 \times n_2) \rightarrow$  $((\times) + (\times)) \rightarrow$ 

.32 Since we can get lots of examples of (Graph ⟷ Q) (.22) we should be able to make the correct method.

.34 Perhaps try writing Recursive Soln. that should stem from these Examples

.32–.34 Seems like a more or less "well defined problem" so leave it for a while.

→ Looking at (.27–.29)L & .31L; it seems that one can easily design an algm. to go from ② to ① (and ① to ②)

To go from ② to ①: first write $n_1\ n_2\ n_3\ n_4$; ...... ?

1/2/07
ID

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

00: 494.40! In 494.27 ff: One could use "rewrite rules".
The Graphical (494.27-.29)L can be written as a polish expression.

sum ((sum (n_1, n_2), sum(n_3, n_4)) v.s.
$((n_1 + n_2) + (n_3 + n_4))$

So $n_1 + n_2$ rewrite as sum $n_1 n_2$] rule a
$(\alpha) + (\beta)$ rewrite as sum α, β] rule b
Note that rules (a) & (b) are very similar.

Also, to go from a to b, it is probably possible to devise a CF "xltn Grammar"

I could just drop this problem for time being, since it is certainly solvable:
One possl. reason to work out now: I'd like ideas on how to (recognize / discover) recursive algms., & this one seems particularly simple.
Also if I want to continue this particular TSQ, it would be well to know just what [form] this particular "Recursive induction" takes. Actually, I can put it into any form I like as a special "A.H" part of the "Updating Algm".

So: say TM can now evaluate all expressions containing +, −, ×, ÷ . . .

A possible next lesson: To lrn that "inc(x)" means x+1. Also that inc(α) = α+1, where α is any evaluatable expression. We can teach this w.o. any explicit "Definition" facility.—
i.e. Q = inc(n_1) ; A = eval(n_1+1) : Once this is lrnd, how easy would it be for TM to realize that we can replace "inc(x)" by "x+1" in any expression?
It would be more likely if TM had already experience w. substitution; So if substitution was a "primitive", TM might do ANL much easier; & well as this stuff. (The other Alternative would be for TM to "discover" substitution "on its own")

Its difft for TM to lrn $(n_1+n_2) = (n_2+n_1)$ if n_1 & n_2 can only be actual nos. One could, perhaps teach this, if addition took a lot of time? So if we asked for Q: n_1+n_2 : A = eval (n_1+n_2) Then next; $Q_1 = n_2+n_1$ TM could save time if it remembered (n_1+n_2) & knew that commutation rule for addition.

Thus, the real point of TM knowing "the laws of Alg", is in dealing w. unknowns; The idea that one could manipulate expressions w. unks. (according to "Laws of Alg") & get new "True" expressions, & often solve eqns this way. To solve eq., TM could at first do xfms in a "Lynch order" Then later, lrn that certain operations & seqs of operations were more likely to help solve an eq.

.34 489.34 [SN] Recursive funcs: The functions that are definable using simple "if" statements, are computable via a stack. Its my impression that stack machines are limited as to what kinds of funcs they can compute . . . . So this simple "If" statement prodn will not cover all (partial) rec. funcs. Can All funct. be computed w. 1 stack? does it need 2 or 3 stacks?

1/3/02
ID

● HAMPTON RESEARCH ASSOCIATES ●
26 Boylston St., Cambridge, Mass. 02138

:00 : 495.40 : Perhaps try to take a bunch of (problem/in Auckuprobleum) types & formally order them for TM's acquisition order.

One over-all listing: (1) ANL : including ~~recursive~~ recursion (or equiv)

(2) solving linear eqns (3) solving some N.L. eqns by use of basic "laws of Alg".
("laws of Alg" ideally, is level by TM to be & "motivated".

(4) Solving some N.L. eqns by (invertible) substitution.

(5) Differentiation

(6) Integration

All things is symbolic (non-numerical eqns). A Big modification: Include numerical approxn for soln. of eqns.

:10

Another off shoot: Soln. of polys by factoring.

:12 A distinct Partial ordering: Start w. TM (understanding/knowing) ANL, but for literals & only. Then soln. of alg eqns, using literals only: first linear of various kinds, some N.L. than quadratic (w. $\sqrt{x}$ available), then cubic (w. $\sqrt[3]{x}$ available). Using these, TM would have discovered complex nos. ! ~~[illegible]~~ (we write above 2 $\sqrt{x}$ values, also 3 $\sqrt[3]{x}$ values.)

:16 See if it could do quartic: would it realize that $\sqrt{x}$ & $\sqrt[3]{\sqrt{x}}$? (i.e. 4 values). ?

So there are several ways to "Learn Algebra":

:20 1) ANL using integers, so answers are always exact integers (including division).
Then soln. of linear eqns.; some N.L. eqns (polynomials, some other eqns)

:23 2) Use of literals only for Algebra (.12–.16). [Some difficulty w. division by zero: often one can't know if a quantity is or isn't zero or not.!]
. ~~Later~~ Next, TM might learn about + × − ÷ operations & actual reals; (or just use integers as in (1) (.20).)

3) Understanding of "approxn", soln. of eqns in floating pt., successive Approxn. of sols. to eqns.

:29 In (2) division by zero is a difficulty. We could ask TM for a soln. or as how division as pass!

:30 We could tell TM, which constants/expressions were ϕ, or $\neq\phi$. TM could give soln. in form: If (α) (or expression) $\neq 0$ then $x=a$; if $\alpha=\phi$ then $x=b$. I.e. soln. of $\alpha=\phi$ (ordinarily $\alpha=0$) must not divide by ϕ – or if it is ~~not~~ we should have a new expression that can't be ϕ.

Say $(x-a)(x-b)=0$; If we divide both sides by $(x-a)$ then $x=b=a$, $x\neq a$.
If this is not possible, because $x-a=0$, then we can also solve & get $x=a$.

Say $(x-a)(x-a)=0$. So if $x-a\neq 0$ then $x=a$; If $(x-a)=0$ then $x=a$; so $x=a$.

Say $ax+bx=c$: If $a+b\neq 0$ then $x=\frac{c}{a+b}$; If $a+b=0$ we can't solve it.

497.00

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

[redacted]
00: 496.40: Actually, "division by zero" is a not very relevant problem: we can deal with it in an A.H. way: T. main problem is soln. of eqns., so I think there may be instructive situations re. "lrng".

In TSQ of 496.20, + "No division by ϕ" rule, is slightly simpler than it is in 496.23.

— This is because one knows 3+7 ≠ 0; but one never knows whether a+b ≠ 0.

In both cases, one doesn't know if 3+x ≠ 0 or a+x ≠ 0.

.06 Q: In 496.20! Its soln. to a linear Eq is an integer, is it always possl. to find its soln w.o.
..07 using ~~normal~~ division yielding non-integers? (.075) If it is not, then I'll have to have TM lrn about fractions. Here I suspect that t. answer to .06 — .075 is "yes": TM simply keeps nos in form
10 like $\frac{7}{3}$ & adds & subtracts them, ect. Its equivt. to knowing how to manipulate fractions, but TM need not know fractions "exist". Eg. TM knows $\frac{a}{b}+\frac{c}{d} = \frac{ad+bc}{bd}$
so b can't be zero since $\frac{1}{b}$ exists.

$\frac{a}{b} + \frac{c}{b} = \frac{a+c}{b}$ | since $\frac{a}{b}+\frac{c}{b} = a\cdot\frac{1}{b}+c\cdot\frac{1}{b} = \frac{1}{b}\cdot(a+c) = \frac{1\cdot a+c}{b} = \frac{a+c}{b}$

$\frac{3}{7}$ could be acceptable as a soln. to a problem! Whether TM should (want/need) to know $\frac{20}{30} = \frac{2}{3}$, is unclear at present.

The TSQ of ANL (493.00 thru 494.25) (w.r.t. final recursive function) would be an instructive TSQ for ME for TM to lrn.

20 REMEMBER t. Subgoal of 492.04–.16

.21 So 493.00 — 494.25: We observe (no op no) many times in "Q" forms, so we give it a name, α: & rewrite those expressns: We then observe (α op α) & call it β.

α & β are of form (α op α) or (no op no); so δ = (γ or γ) w. $\gamma \to \alpha$ or no; ~~$\gamma \to \alpha$ or no~~.

~~Maybe~~ we could then def. t. recursive rule $\gamma \to$ no, α, δ.

If we have many cases (494.22 asst. case), then this should be easy to spot.

.21 It gives us t. general form of Q, but not its corresponding A. We might modify .21 ff so its a Xlt grammar, so elements of t. "Q" expression generate Corresponding Elements of t. "A" expressn.

30 One way to devise t. Xltn grammar is to find separate Grammars for Q & A, then make correspondances betw. t. 2 Grammars.

(n1+n2) × (n3+n4) → $\alpha \times \beta$: Again ($\alpha \times \beta$ → mul(α', β')) is a recursive rule that could express t. whole Grammar.
mul (sum n1, n2 , sum n3, n4) → mul(α' β')

Good idea to Review present status: ① where am I at. tasks not for TM ② what about 492.04–.16? ③ Specifically what (13 area) present problems(s) & subgoal(s)? State bottleneck(s) clearly. If I can't solve one now, Go on to Next one!

→ (Spec 498.10)

1/4/02
IP

Convergence Pirm for QATM: Criterium: .00

HAMPTON RESEARCH ASSOCIATES Associates.
26 Boylston St., Cambridge, Mass. 02138

.00 : 486.03R : On t. convergence Pirm. for QATM: In 578T3, the seq to be predicted is selected a priori: so we expect about same error in t. near future. In QATM, t. Q's are selected to form a TSQ & if t. future Q's are radically different from Those of t. TSQ, we don't expect a formerly good error to continue much.

If t TSQ & future Q's are selected a priori, we expect better continuity into t. future.

.10 : (spec 497.40) 1) T. MainProblem now is writing TSQ's. T. partial TSQ's written Thus far:

a) S56 ("An Ind. Inf. Machn")

b) S86 deals ANL a bit. (one version).

.13 c) SAARB: I ANSL using Stack Machine: (Mech. detailed).

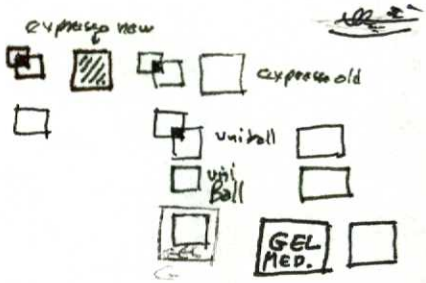
II Soln. of linear eqns & discovery of "Laws of Alg".

III linear → quadratic → cubic equ. viz invertable substitution Heuristic.

d) Recent ANL 494.04 — ≈ 495.33 : The "Exact Mechanics" of Recursion is (unclear/undecided), In general, details not worked out: mainly "In English".

e) Suggested other TSQ's: I: Develop (492.33 & APL) II: differentiation → formal integration. (492.31)

III Use AI, work on "Expert Systems" Get them to learn t. rules.

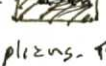


.20 : In Search I (.13): 2 drifts: ① I only used entire previous Solns to probs in Memy for new problems

② As no. of elements in memy ↑ (n elements); pc of many accesses was ∝ $\frac{1}{n}$ — which did not "scale" to large systems.

W.r.t. ②: I was using OSL, but I did not use it properly, so pc's of retrieved concs were not correct. — But more generally, I did not use context at all! All items in Memy had same pc.

A very simple context that would be "not bad", would be "recency" — pc = funct. of no. of problems back, it was created. — Also frequency of use (inf > 1 ☺). Context could be "How many problems ago was it's most recent use — which would ↑ much for frequently used concs. Say ↓ of pc w. n (n ≡ no. of problems ago) so $p.c. = f(n) \leq f(n) = 1$

.30 Then if there were k_n concs. that could have been referenced in the "nth ago" problem, Each such conc. gets  $pc = \frac{1}{n \cdot k_n}$.

As I find applicns. for various kinds of concs, this is "recency rule" [redacted] will change — perhaps elimination of "recency rule" completely!

Note: t. QA's are "ordered" & coding O^n can't take advantage of any apparent apparent order in t. QA's. But "Context" can take advantage of such order. → (499.20)

.37 Woops! T. foregoing analysis of recency seems wrong! — since t. pc I'm computing is to be used on O^n! — Tho maybe Not. (No, it's O.K. see) See 499.00 -.15 for a review of this problem → (499.00)

HAMPTON RESEARCH ASSOCIATES
26 Boylston St., Cambridge, Mass. 02138

— ABC

.00 : 498.40 : Say pc_1 is regular apriori : we maximize it in O^1 of a conc.

pc_2 is of a conc. is it pc assoc. w. its "context".

See [486.13; [redacted] → 486.22R]

→ 488.12, 492.04. The these refrences are helpful, I don't think they give an adequate idea as to how pc_2 differs from pc_1 : I think pc_1 is relatively well defined. It is regular apriori of O^{n+1}.

.05 I think the original (& perhaps "best" correct") defn of pc_2 had to do w. "context" in a most general

.06 sense. As corpus, it included everything that occurred in & was observed by TM since its birth. → Also into TRAINER May want to rerun

[While .05–.06 seems "not bad", I'm not sure it would be adequate for explaining to "another (not me) person"! (or to "Me" as of 1 month from now (or which was 1 yr. from now!))] → 500.19–.23, .34–.36 cleaning this up time

I.y. seems that pc_2 is related to ~~the~~ expected accuracy of predn.

.10 pc_2 is related to cc for finding code .... to its CJS; Lcost or whatever

Also, pc_1 & pc_2 differ not only in their corpi — but (perhaps) in their apriori ? Well, may be not! — both can use any truly apriori info — i.e. obtained before & indep of the corpus.

The pc_1 of the corpus is $\prod_i pc_1(Q_i, A_i)$

.15 The "Lcost" of the corpus is $\propto \sum_i pc_2(Q_i, A_i)$: This is the sum of the Lcosts of each of the problems.

Note 487.32 on logical reasoning in srch .... as relevant to pc_2 } But in different ways, since the corpi are different
But also Note that Logical Reasoning can also influence pc_2. } (pc_2 has pc_1's corpus plus a lot more.)

.20 : 498.37 On the "recency" heur/content: This may be fine for teaching TM in a warm, friendly envt. — (The infant TM), but for its later life, "recency" is usually not as useful, (tho it still can be useful.)

The original TSQ is usually designed so that "Recency" is a good "context index" — but for a TM doing R.W. research, the problems can occasionally be ordered by TM, (i.e. TM is designing his own TSQ) &. The TM may not be specifically designing the TSQ so that Recency is a good "context", I suppect that it will often be so — that it will tend to be so in most any TSQ that TM would want to design for himself.

18.30 -710 Bus A P# 39

30 Re: Recency for SAARBI (498.13) : this would really speed it up tremendously (I think!)

Also, [redacted] CJS would ↑ much more slowly w. size of corpus (I think). This "theor" would be v.g. for initial Trg. of TM, to get it to the $O^{n+1} = O^n(O_n, QA_{n+1})$ [redacted] point.

Recency may be crit. for infant TM [redacted] for pc_2 : But pc_1 would seem to be not v.p. — Not much better than classic SAARBI ! — which seems close to AZ141.

One trouble w. using pc_2 (rather than pc_1) to guide Lsrch, is that using pc_1 would generate much more bias toward by pc_1 — (which is very desirable). We might mix the 2 by doing trials in pc_2 order but do cutoff via $\frac{cc}{pc_1}$ (or vice versa) — (Does this mean anything?) In either case 500.00