4 hrs 47 min.

Space (12.4.90): (Normzn, cont) For early TSR, normz will be a problem: since Rizzo track Lsrch, we are interested in type functions - we need to keep track of Σp_u of pcs of those that don't converge, vs $\frac{\Sigma p_c}{\Sigma p_c}$ of those that do.

So Normz constant is $\propto \left(\frac{\Sigma p_c}{\Sigma p_c \Sigma p_u}\right)^{-1}$. This pools data from diffnt Q's, hrr.

This recent stuff on string prodn (which will work for real prodns as well) is encouraging! — particularly re normzn discussn!

For Lsrch, in QATM: I guess we just do Φ^2 mass in pc order.

My impression: A smooth transition betw MTM & NMTM in (QATM): In MTM, just one function $= \Phi(R \equiv \phi \text{ say})$ in NMTM was just here (many) more II functions w. various wts.

So Ph. I seems fairly clear now! For both Lsrch & Bsrch. Also clear understanding of just what to Compass is for TM2 ⇒ 120.09.

Some uncertainty about → BU srch

" " " GA/GP for Busrch.

On Busrch: We could continue w. TM2, getting better pd over functns.

T srch itself would perhaps be identical to what I was planning for GP" — i.e. using a v.g. TM2 "emulated" pd on functs constructed over a "Population Reproduction".

We might use $\frac{\sigma}{\mu_x - \mu}$ / threshold for population upon which TM2's d.f. is based. TM2 can be a stoch. cfg csg, special csg w. facilities for discovery of impl. "sub-trees", whose possibly use PPM w. modifns for "sub-tree" contexts.

24 Q: If we are doing Lsrch & we change to pd on functions — just how is this done? Ideally, we keep track of which trials were remade, & never do some unlikely pc trials that we haven't done, do them. Then don't do more lower pc trials that have already been done. (Use hashcoding to keep track of trials Only've been made. Hrr, the actual mechanics of ordering & trials is unclear.

So .24ff — NEEDS work! Hrr. I was thinking of using T=3T for Lsrch

For .24ff! Say we were doing T=3T Lsrch. Well, we reorder between T=3T "Rounds"

34 Ti Fogg suggests that "Phase I" is in good shape: What Phase 2 needs is a better understanding of "Opfns" than Phase I has. It's still not to early, nor likely to soon way to "understand" opfns. E.g. Sam Annual is a way of "understanding" opfns & it has some understanding of "local optima".

STM

: [Among discussion of PH II, I should mention clear that there are many ways of ~~doing~~ ~~of often~~ doing "often": one is a way of understanding "often". I will want to list various modes of understanding "often". Are there any that "dominate" — in the sense of being a combination by which certain methods can be derived?

: 120.19 : [redacted] layer CB (or some [redacted] modification of the appl).

This Needs more Prot! [Another Possy] is that there is much implicit info that we have about ~~it~~ not in the appl given by our Reference Machines E.g. T. Bernoulli's rags, derive by "Laplace's rule", say, are not intrinsically in Ref machine — but would have to be ~~discovered~~ discovered there a large corpus that had [redacted] frequently occurring Bern regularities. We feel that it's O.K. to insert the "PPM-type" rags into the appl because we have a strong "Pro corpus" w. those rags

T. Forg. looks like an extremely impt. Q. That has much bearing on our general understanding of How ALP works, & to what extent "Occ Rez" is correct. — How Appl info enters into ALP.

"t. Bern. Seq. rags"

Certainly ~~Any~~ Anything" t. Reference Machine usually has no info about Binary Sequence regularity"

— Similarly for PPM type rags & "Sub-tree" rags & CEG & CSG rags.

21 A clearer understanding of what's being done is obtainable by postulating a long rich pre corpus
22 w. the desired rags. This effectivity & boost of those rags in dens/of the corpus. This boost reduction
23 is observable in the
24 The Forg Suggests that "TM_2" can't be ~~any~~ any smarter than ~~[illegible]~~ using [illegible] rags is that are consciously
5 known" by the trainer — that TM_2 or {$TM_2 = TM_1$} will not help much, if at all.

(.24-.25) is an impt. Q: Also: What about BU srch, [redacted] what's going on Resamp re
wrt the utility of TM_2 etc?

(.21-.23) is perhaps the best way to analyse this stuff

→ A particularly impt. output of Phy recount I'd like is a way to understand just
What the "corpus" of TM_2 is!

Also, the importance of the [redacted] Ref. machine = set of primitives:
This, too, is ≈ equiv. to using various TM_2 tech rep recos (= rags
detection Algms ≡ rags definitions).

To an impt. extent, various induction methods used for patterns, compression, etc., can be regarded as different appls or different aspects of the appl.

Is it (almost always / always) possible to express any TM_2 rag as a succession of the
use of certain "Macros" — or alternatively Resamp to rest primitives?

— if so, I should be able to diagram new TM₂ entirely!

T. Various TM₂ reggs may be regarded as alternative inductive means, to be used in a "Boosting" system.

120.09 -.19 +126.10 ff | Needs to be developed, 'filled out"; One way is to write an exposition also "review" of it: explaining the problem to a "Layman".

Well: one way: Consider the way the apriori info of a (person) changes during his life. He starts w. a certain parameter = "common" apriori. That is, say (this individual) is interested only in predn — which gets rid of biases due to (his) varying utilities of various concepts.

So, the way that the guy's (at time, t_1) facts/info at this age is: what it is ideally, what it should be. Say C_1 is his corpus up to that time, t_1 and P_0 is his apriori at $t=0$, his birth.

Then, the "relative" probability of K_0, a particular continuation of C_1, will be $\frac{P_0(C_1 K_0)}{P_0(C_1)}$ ($C_1 K_0$ is the concatenation of C_1 and K_0).

$P_1(K_0) \equiv \frac{P_0(C_1 K_0)}{P_0(C_1)}$. ($P_1(\cdot)$ is the apriori of this person at time t_1. $\equiv P(C_1)$)

Note that (14) is very similar to the techniques that use compression to get variable extrapolations of data. (I don't think of authors, composers, windows of factory, 'don't think of styles of music, literature, etc).

What does $P_1(\cdot)$ look like? How does it differ from $P_0(\cdot)$? $P_1(\cdot)$ is corpus info itself, all of the regularities that have occurred in C_1. However, $P_1(\cdot)$ is obtained w. $CB = \infty$. — Consider $P_1'(\cdot)$, the apriori obtained by a real person w. finite & very limited, CB.

$P_1(\cdot)$ will include usual Bern. Reggs: (Pinckleton's Laplace's) rules, ≈ PPM, various linear & nonlinear methods. It will include many reggs not discovered via corpus C_1, directly, but by references in C_1 to previous data & to summary rules, obtained by other individuals.

$P_1'(\cdot)$ will have many of the reggs in $P_1(\cdot)$. Many (implicit) regularities will be in the form of words that have been deduced into the common language by the social, scientific community in which he lives. These definitions gives low bcosts to the defining of certain reggs in the corpus — giving by pc's for those reggs.

What this implies: that the ssz of the occurrences of these reggs need not be very large before it overcomes the ~~[illegible]~~ pc of referring to (rather than the much larger bc of (verbatim) the regg.

So: The implication here, is that we don't want to apply P_0 to predicting K_0 (.13)
I.e. $P_0(K_0)$, but we want (.14) which is usually quite different.
$P_0(K_0)$ is what we get if we use the reference machine directly on K_0 as in Lsrch.

: presumably, by using TM_2's regrs, we get closer to $(27.44) \cong \frac{P(C_i \& k_0)}{P(C_i)} \cdot \equiv P_i(k_0)$
Also, by selecting Reference Umes that we feel have been useful in ~~[illegible]~~ expressing
past ~~[illegible]~~ regrs int. corpus ($\equiv C_i$), we also get closer to $P_i(k_0)$.
While the mechanics of how .01→.02 approaches $P_i(k_0)$ seems clear enough, the ~~[illegible]~~
mechanics of how TM_2 does this is unclear. In particular, it's unclear as to what
TM_2 should use as a corpus in QATM. An expanded general surface wk of
the preserved analysis should be an (some) understanding of how Q.

(S/N) ~~[illegible]~~ Almost all of AI induction/AI methods described in the literature
that actually work (say "AI about" & "Pattern Classification") could not be logically
discovered on a very large SSZ (& probably not by Lsrch).

Just as in Lsrch, for BUsrch we can probably make Estimates as to how
long a particular regr would take to be discovered.

→ The moral of .10 ff is that if we want a smart TM, we will either have to accept
it hears like .10 – .11 (saying "Meta Prediction of TM_2") or use an enormous
corpus (perhaps many books of "textbook" problem a/o the internet)
The Ramanujan was able to do well on an apparently small corpus.
— He may have had to pick up premises here in the non-Math
parts of his life & use them in Math.
Also Newton didn't read a lot before his "very productive yrs"....
There simply wasn't much "good stuff" to read!
The books I/Rom read did have a lot of Incur material. Also, these
books probably had a lot of "textual, RW" type problems that we may have
difficulty deriving/explaining to a "R.W. naive" TM. — By abstracting
those problems & giving them to TM in those forms, we we may be
then depriving TM of many impt. hours/clues — in which the hours
needed were obtained from R&M's RW. experiences!

Anyway, the point is that very much of the literature in AI & related
statistical methods would be, should be usable/used in framing TM —
That otherwise TM would be unlikely to derive these hours w. reasonable
size & More compt. Reasonable cc (see (013)[9]).

Perhaps, while TM is actually being framed, in II, we will
continually be adding new techniques to TM_2 so as to ↑ TM's
prowess by "boosting".

: To do 128.35 One would have to understand much of recent "A.I. work" (≈ classical) — a truely formidable reading/education task. — yet the AUTHORS of "A.I. & Mod approach" & "Stat Classifn" seem to have done this — tho no one person may understand all of that material. It may be possl. to teach people how to make use of statistical/heuristic techniques & put them into a TM≈ routine that could be part of a "Boosting" cluster. So we just hire a bunch of chor Grad. students to do this) → (.13)

A Bit Q: How far can one go w. Lsrch alone (not yet in BU mode)? Can one go as far as Ramanujan? — Newton? — essentially "small corpi"? Was either a "small corpus"?

Consider apri info: What is the best way to 1) Recognize it 2) Utilize it in pred'n process? What are some (not really Best) ways to do this?

3: (.07) → One way to do this would be to put all induction systems (that we want to include into Apripd) into a common format. ASP is a common format of this kind.

Amenable to "BOOSTING"? — The Lilian Boosting is not possl. in Lsrch - only BU srch.

However, there seem to be 2 quite distinct ways to "Mix" PEMs!

1) MaxRob (as in last Method in Sol'64≈) & "Coding & recoding" as in TM≈ ("secondary prediction"). Are there ANY other ways?? → (.33)

18

APPM Part

In (Sub-tree Context augmented) PPM (≈ APPM): One could just "spend time" in (≈ alt improv) SI, looking for Subtree contexts that were good for prediction. Then make a data base of them and some kind of Zero coding or Hash coding to see which of those contexts was relevant to the present extrapln.

I was critical of that approach, in that it didn't do "N gram sets", but after one has a set of predictive contexts, we can pool the contexts by similarity of Main preds. preds, then try to find short codes for the sets of contexts w. similar preds. Having a "set of context", one can make a bunch of new ones by concat & Boolean operations. If we use recursion we can define infinite sets this way (as in CFGs). These newly defined sets can be tested for Goodness in pred'n.

33: (.18) → Actually any way of mixing PEMs is legal/acceptable! But intl. (in "Meta induction modes") its easier to apply the PEMs & evaluate params for them — if good (large enough SS≈) to evaluate the pc's of their defining sets, etc.

Meta coding seems better ≈ recoding

However some PEM may be of the forms that have to be "mixed" more w. other PEM≈ types. (Can't immediately think of an example).

5TM

: Certainly the methods of II & Meta coding are 2 easy ways to use new Pams. II coding has good Theoretical Basis. → The Meta Coding does, also. ☺.

Essentially any PEM is: Given the ("past") corpus it gives a P.D. over the entire future or equivalently, a grd over the next possl. Symbol.

While all Pams are (usefully) famorable to II coding! The Meta Coding seems more particular!

i.e. Say we use L such as ALP for Pam, & APPM for Pam$_2$;

Then Pam, for primary induction & Pam$_2$ for Meta induction seems fine for QATM corpus.

But w. same corpus: could we use Pam$_2$ for primary & Pam, for Meta? — Using APPM for primary on QATM would probably work fine w. an enormous corpus. —

— But what would Meta Induction do? — How would it work? How to even define it (in this case)? Meta induction is easily defined when primary induction produces a compressed code (or several compressed codes) of the corpus.

One way, would be to express each PEM as a compressed code (say Arith. Code) — which seems, somehow, very "Artificial"! → A (perhaps) "less Artificial" way would be ← .24

Another thing we'd like to retain, when Meta Predicting is the diversity of primary induction obtained by multiple II codes. (DIVERSITY) Obtaining pc's loses this for whenever sums over II codes. — However, many practical methods consider many II methods of (reasonable) prediction, w. a "Best" one selected. By using all II predictions, we need not discard this info.

Expo: How Non-Bayesian Statistics is a seriously impediment to achieving Strong A.I.

- That putting A Priori info in the form of Hard constraints of necessity limits kinds of hypoths to be considered, in the wrong way — i.e. Hypoths to be considered should be limited by CC only.

24. (.14) → To use many II codes of 2 multiplot (compress original corpus) (for 1) so PC = $\frac{3}{8}$ of "1" would be coded as 3 codes of length 3 ($2^3=8$), 5 codes of length 3 ($2^3=8$) for 0.

To get higher resolution than 2^{-3} we could use 2^{-10} or 2^{-100} "grain size". These codes are not actually used, but are for Theoretical Understanding and analysis only → (.34)

One way to mix 2 pams: Use both to get sequence of pc's for corpus ^0P_i, ^1P_i: & Make a binary time series, X_i, where X_i = whether ^0P_i or ^1P_i is larger.

Use $\hat{P}$, $\hat{P}$ or a third Pam to predict X_i on the basis of previous data on $X(\cdot)$; $^0PC(\cdot)$ & $^1PC(\cdot)$; the original corpus, C_2. If X_i is compressable to fewer bits than for the predn, we have an extra compression. Actually, we can try to compress X_i w. a "fidelity criterion" — since certain bits of X_i distinguish betw. $\hat{P}$ & $\hat{P}$ that are almost the same.

.34: (.25) → In Theory Any pair of Pams could give a set of II codes via .24 - .27, and these II codes could be individually compressed by any other Pam$_2$: In fact, Pam, could be used! For all practical purposes, it's unlikely that one would do this: "Too many codes" ☺, but for Theoretical analysis it may be OK. — So any 2 Pams could be "cascaded" this way. Whether the order commutes is unknown. I don't immediately see how to get anything

: out of this!

This "cascading method" seems close or identical to "regularization"? Here we remove certain regularities (say by linear smoothing, detrending) & removing periodic (usually yearly) components. Then they look for regs in the resultant T.S. — Say "non-linear" regs. Any Quantum Method of detecting Non-linear Predictability in a T.S. (BDS ?? Method) should have an associated predn. method derivable from it — Tho it may not be easy to implement.

So if Pem_1 is a "coding method of induction" applying it gives (usually) a set of codes that have much of the regs "removed".

In line w. Detrending: Consider a real T.S. w. discrete times: We have a Pem_1, say, Rest gives a $\mu_i \pm \sigma^2$ for each element of the T.S. as a funct of past data. We could create a new T.S. $X'_i \equiv X_i - \mu_{Pem_1}(i)$. where $\mu_{Pem_1}(i)$ was the most probable prediction X_i. (Pem loses σ^2 info). In linear predn. σ^2 is constant. — (But not in N.L. eg.)

To T.S. X'_i is supposed to be unpredictable by (linear methods — i.e. all linear methods should give a $\mu = \phi$ & a constant σ^2 for all predns. In the resultant T.S. X'_i, If we are able to find any legit regs, we use these predns and add back the μ_i's from the linear predn. to get the "final" predn.

[Can we use an oracle by Pem_0 to modify a corpus, (of an Alphabet .. of just binary?) So that Pem_0 gives $PC = \frac{1}{2}$ to all cases when Pem_0 is applied to new corpus Σ — yet knowing corpus, one can restore corpus, w.o. error, using only knowledge of Pem_0.

Does linear detrending satisfy .10–.20? (will any thing satisfy .18–.20? — it may not be what we really want

For every Pem_0, there will be a set of sequences, S_0 → T. predn for next bit is always 50% for 0 or for 1. Is there an invertable mapping that depends on T. domain of PEM_0 only, that maps this set of seqs into the entire set of seqs? Or, more narrowly: Assoc. w. Pem_0, there will be a set of seqs of higher μ: (call this set S_1) will there be an invertable mapping (↔) betw. S_0 & S_1?

No, that's not what I want. S_1 is characteristic of Pem_0 only. T. corpus C_0 will normally not be in S_1 (unless Pem_0 optimally predicts C_0).

I'm thinking of something like the Gram-Schmidt orthog. process.

for .23–.27, try to apply it to linear predn. T. simplest case, if T.S. has a μ. we subtract out the μ to get a new TS, invertible back to the original. If it has (1) fewer regularity Rmnt. original, (2) serial, say Rmnr is a cross corr. betw X_i & X_{i-1}, so $X_i = \alpha X_i$ is a good prediction (if μ = "Bias" = 0). We can then represent the series as $Y_i = \frac{X_i}{X_{i-1}}$. This series will have a mean value, with which we can (recursively) normalize. (trouble when $X_i = 0$ or $\to 0$!).

Also if $\left(\frac{X_i}{X_{i+1}}\right) \neq 0$ then X_i would grow exponentially.

Another approach: if $\overline{X_i \cdot X_{i+1}} \neq 0$ then maybe map $(X_i \cdot X_{i+1}$ into $Y_i \cdot Y_{i+1} + \overline{X_i X_{i+1}}$

This may do it. So $\overline{X_i \cdot X_{i+1}} = \overline{Y_i \cdot Y_{i+1}} + \left(\overline{X_i \cdot X_{i+1}}\right)$ ∴ $\overline{Y_i \cdot Y_{i+1}} = 0$.

So $Y_{i+1} = \frac{X_i \cdot X_{i+1} - \overline{X_i X_{i+1}}}{V_{ar}}$

6/22/05
5TM

6.27.05
CNBI
28" font

16:31 : watch
28" font w.r. Atomic time"

50 Which will cause trouble if y_i ever $= 0$! y_{i+1} (which becomes y_i) $= \infty$ if $x_i \cdot x_{i+1} = x_i \cdot x_{i+1}$

which will often occur approximately & perhaps sometimes exactly!

This is troublesome! Perhaps use a peak $\pm y_i$. Does this make the $y_i \to x_i$ return ambiguous? Another Q: $\frac{a}{\phi} = \pm\infty$, so we don't know whether to make y_i large $+$ or $-$!

$$y_{i+1} = \frac{x_i \cdot x_{i+1} - \alpha}{y_i}$$

Yet we seem to end up w. $\overline{y_i \cdot y_{i+1}} = \phi$!

Well, an occasional large y_i may cause no trouble! Say $y_{i+1} = \frac{z_i}{y_i}$, where z_i has a max value of ϕ & is occasionally ϕ. Say we define $\frac{z_i}{\phi} = \pm 10^6$ ($|z_i| \ll 10^6$ say).

$$y_{i+2} = \frac{z_{i+1}}{y_{i+1}} = \frac{z_{i+1}}{z_i} y_i \quad : \quad \frac{z_{i+1}}{z_i} \text{ can be anything below } -\infty \text{ to } +\infty.$$

80

$$y_n = \frac{z_{n-1}}{z_{n-2}} \frac{z_{n-3}}{z_{n-4}} \frac{z_{n-5}}{z_{n-6}} \cdots \frac{z_3}{z_2} \frac{z_1}{z_0} y_1 \qquad (n \text{ odd})$$

y_1 is arbitrary — say $y_1 \equiv 1$.

Some z_i will be zero. I guess that $\frac{1}{2}$ no. of the z_i w. odd i even i will be $\pm\infty$.

So in eq. .10 they will mostly cancel. — But y_n will occasionally be ϕ & occasionally $\pm 10^6$ (say). In .10, the chances of $z_i = 0$'s will be small & y_n is zero or $\pm 10^6$.

Say I get $\sim .00 =$ if working. Next: $\overline{x_i \cdot x_{i+2}} = \beta$: $\to 0$ in x find sequences. Same way, but we end up w. 2 indep seq's! Hence same β for both, same initial value for both w. $\overline{x_i \cdot x_{i+k}} = \alpha_k$ I get k indep seq's.

For quadratic prob's: $\overline{x_i^2 \cdot x_{i+1}}$ or $\overline{x_i x_{i+1} x_{i+2}}$ can be used.

Inc Irvg
Exp in text. PS
PS\4.7.05

Inc Irvg of interest pts in PS\4.7.05

20 A somewhat different track: For Perm: express as "predictive" terms of sequences of errors is Perm to be predicted as an indep T.S. or as a difference TS highly correlated w. original TS.

Or: Do various forms in W, Brad do "Boosting": "boosting can Bat[illegible] vert. detect. Problems or Detect. Symptoms (Bob, Say)!

BAHASA

P37 - 36 **Bahasa** Indonesian lang easy to learn lang. Lots of sets of lessons (w. sounds) on net. web

This note is for "Confessions of a Econ Hitman."

26 SN! Re: TM2 it would seem that a very imp't Q would be how to deal w. small $\$$$\$$! If we are looking for regs in O^j: O^j is still a very Big $O_{j \in EF}$. Logically, we can use as corpus, T: Entire trace of TM since Birth! This includes, of course, all $\left[[Q_i A_j]_{i=1}^{k}, O_k^r \right]_{k=1}^{\text{M. now?}}$ pairs.

28

29 Much of this stuff is clearly redundant, but which, we have no idea on how to use for predn. → 135.00 Spec

30 Expo On "Greedy Science": The ideal is always looking for a simple basic theory, then Boile & Partners, around it to defend it. Any new modeling have to be small & clearly "good". However, Backtracking (a very new concept = paradigm) is periodically necessary, & very expensive. Greedy Sciences are very conservative

45 Expo "Advice for a young Scientist": (20 is perfect.)

135.00

5TM SM (.00)

10 : In finding drivers over a very large corpus: say (1000 drivers) I was disturbed
.. that I'd have to ~~be~~ divide apparent ~~be~~ sum by 1000 to get realistic expected sum.
.. No ~~one~~ sum (apparent) factor of 10 might be possible over a Time of 5 ~~yrs~~ yrs, 1000 seems
03 · too large. → (Sara .10 - .12 for a better understanding)
A· way we did do it was to first narrow down cand. drivers by long time correlations.
Then we have at all times a "membership updated" set of drivers ~~be~~ for a
given driven. Using shorter correlns, we use these drivers for predn.
So I don't know if I can find Rao justifn for this strategy!

Another approach: Suppose we have an ensemble of random drivers/driven.
Certain pairs of them have actual/predicted correlns. I can assume a certain d.f. over
30 — correlns. — Then How could I find ~~f~~ correlated driver/driven pairs?
.11 · { If f. correlated pairs were sufficiently rare then I couldn't find them w. any reasonable
.12 { no. of yrs. — which is, I think, f. problem of (.00-.03). The idea is, that if
· I need a very large corpus, f. correlns are rare, & I need more evidence
.14 { for ~~the~~ them before deciding they exist! If f. correlns only last for a limited
{ amt. of time; then I may not be able to verify them if their likelihood of occurrence
{ is soft. low.

So, it would appear that having a very large ~~ensemble~~ ensemble from which to pick drivers
may not be such a great advantage! What it can do is get lots of somewhat
20 — correlated drivers, so one can use them together for a predn. How, f. cross correlns will
· not be very accurately known! Working this out rigorously would seem to be difficult. —
· T. alternative is to try it f. empirically — Tho this sounds not v.g., because if I
· don't know just what correlns, what accuracys are relevant, I will not be able to
24 — · rely reliably extrapolate f. empirical results!

One way to decrease w. .20-.24 ! Act as if f. drivers were uncorrelated — (as we do in APT)
This simply gives more wt. to correlated drivers.

· 27 { An easy way to get cross correlns is (Estimate partials) Say of taking highest correlates!
· { For f. entire pop'n. Ensemble: Take random samples & get correlns. w. driver
· 29 { We get a (parties) smooth curve that gives f. desired D.F.

30 Another useful concept would be: Select a ~~large~~ ~~number~~ elements of Ensemble
& random — get d.f. of cross correlns. Actually, b/ Driven (pairs!) are another
· narrow set, so one does ~~.20-.24~~ .27-.29 for each of them. ~~They are better~~

In actually using f. forgg. for "trading activity": T. ability of a driver would depend on
33 how frequently it trades. How much advance notice" is used for predn ~~&~~ — each
34 case could be different.

Another point: when ~~the~~ accuracy of a driver is very large, it may be more or may be less
predictive; or have in driven may be larger. This would have to be studied.

~5PM SM (cont. from 133.00).

: How we get data for actual trading, is impt. It may be possible to get charge subscriptions to ~~[illegible]~~ services that give some or many stock exchanges a/o other Drivers. If we have ~ 20 drivers, we may want to know at least 10 possible drivers for each! So maybe ~ 200 or ~~[illegible]~~ less than 1000 drivers. Ideally, we would set up s.w. to look for trades in any one of this list of 200.

Again, note 138.33–.34

The update of 1-driver list — looking at 1 to 50 drivers could be done each day ("overnite") Ideally all operations could be done by computer -- -- but we are not yet up to doing ~~[illegible]~~ this: Maybe Louis Morton? Also: Various Langs have been devised (Perl, Java) to enable w³ operations — so possibly we could do it ourselves.

P.S. 05 Amer. trades has/will have Real time streaming video. 9 notes

To start: Search I can use closing data (usually available) & to "predict" driven Δprice (next day.) on basis of driven Δprice (today)

By using/very large corpus of drivers, I can find v.g. drivers, but ~~[illegible]~~ Goodness will have large Variances. I can reduce Risk by using several in II. Hvr. large Corpus means by be to retrieve any ~~[illegible]~~ driver, ~~[illegible]~~ If "ought to be" that as Corpus Size ↑, we get better drivers but be of retrieval ↑. Hvr, it should be better to use large corpus to drivers, but not as good as one would think (superficially)

Also, using good/N.L. models to get correlns could be very impt. in getting better freedn.

5TM

16:30 → 17:30.

10 (spec 132.29): This is partic. coding & (Meta recoding) problems. 120.09ff

120.17-.19 Notes that t. "coding a recoding" idea is relevant to BU srch, not L srch.

Law 22
fuse 20
check 20
app. 15
15 tot.

The Plus test may be "too [redacted] strict" E.g. in L srch we do use tchon freqys, & we can use (APPM contexts) to modify P_D. for next tchon: T. problem in such a view is that SSZ is, in deed, too small (usually) — unless we include all Traces stuff at 132.28 ($\left[\left[Q_i A_i\right]_{i=1}^{k} O_k^i\right]_{k=1}^{n}$)

Certainly OSL (one shot lrng) would help much. It seems possl. to implement OSL using about t. same algms as ~~regular~~ regular APPM. In fact, regular APPM may do OSL.

38 Another way to ↓ SSZ would be to consider "contextsets" (cheaply definable ones).

39 - Go run a reasonable TSQ "by hand" & see if I can find new kinds of "reasonable
D — reqys" to include in Phase I

T. way I had of doing .08 (one way) is: for each possl. predictable tchon, look at all contexts that preceded it. Try to find cheap properties" that these contexts have in common.

I think .08-.09 may be t. way to go for t. L srch phase. I will have to myself know most of t. heurs / reqys that TM will use to solve problems.

16 → One trick that could make SSZ ↑ much w. TSQ size would be to "Recogn. ("& modify") algms" — that decide what types of problem each QA is. This makes t. TSQ consist of a seq. of individual probs that can to some extent, have "individual solns":

19 So that t. size of OJ does ↑ w. corpus size: At first, perhaps, almost proportional (i.e. linear),

0 → Another big Advantage of R and the d is that one doesn't have to test a soln of

2(a new QA on t. Entire Corpus. This is certainly in accord w. Human Prob. Solving. (138.22)

25 Expo "50 yrs of A.I." (since Dart workshop 1956) what has been done; what needs to be done.

10 (.09-.10) seems to say that not much progress has been made since sol59!

A few impl. ideas: ① Mixed Corpus Prem: So OZ probs & other kinds of problems
can be put in "integrated system" ② Understanding of how to put S-indication
(for both Memorize & Hwy except) into solvable forms. ③ Discovery of L srch v.s.
BU srch decisions ④ Ability to use "many" Machine Imrg "Modes for "TM2". (APPM GA, GP, ANN,
⑤ Understanding of equivalence & incompatibility of primitive langs & systems.

It looks like .09-.10 remains main problem: [S.16-.21 zeyroos?] (Looks very Reasonable)

Another possl. process puzzle: (Unclear in my mind as regds what this was!) As part of a proposed Soln. (spec 138.00)

6-27-05

STM

Video Game: "NERO"

: Video Game "Nero" See paper (also PS: 6/28/05) (PS: 6/27/05): Use tasks: Uses ANN & GA. GA can add (or subtract?) Nodes from ANN as well as reassign. wts. to edges.

In this game, there are various "Soldiers" who are taught to achieve goals by "Players"

A useful way to do this: Say soldiers are on "Teams" that compete for goals.

Soldiers on same team cooperate. Soldiers on competing teams don't cooperate with each other.

T. way it works: Players (≡ trainers) can ① assign (Task/subgoals) & decide on reenf. level for each soldier wrt each task.

So Trainer can do 2 things only: ① assign Goals (≡ TSQ) for soldiers.

② Assign reenf. for actions by Soldiers

A trainer wins the game by doing ① & ② better than his opponents.

Early subgoal would be to teach SS to cooperate in achieving Goals.

(cooperation could be informational (giving subprograms/macros to other SS on same team).

T. Goal of "Master God" might be achieved: have them first learn a lang. that trainer has designed. learn to communicate on that lang. to facilitate solns of common problems.

There need not be 2 Teams, tho for many trainers Human Competition can always be a source of excitement & interest. For many video games — heading only 1 player is a "plus"! — for Me it is certainly easier to realize.

How, it would seem that Cooperation betw. different SS could be more directly implemented by mixing strings of various Conds. of soldiers. — or simply allowing SS to be open so other SS could see & use one another Sets/Macros.

T. long Session use is (prob'y) FFANN plus G.A. to Modify Nets by adding, subtracting nodes, Adding, subtracting edges. They can play against the "Master" (≡ Standard Enemy).

Q's ① Are sets of tasks/skills that can be (and "Universal"? in Turing or many weaker senses.

② Are the Params available to the trainer ≈ "Universal"/whatever that means.

③ Is the GA Lang. ever used at all universal? — what are its limitations?

It may be that GA w. strong const (say, could control structure of FFANN in a way to get an essentially Universal System. Normally, FFANNs is not very universal because many funcs w. simple domains are very expensive to decode via f. nets.

A Universal GA could assign wts using arbs (univ.) values so the non-Turing-ability of the ANN would not be impt.

④ Since (GA/ANN is used for long) One could use the Universal RANNs for models: so the System could be really "Universal".

⑤ Abs Q: How could this ever be better than a single TM w. many tape codes: Each code able to "look at" (scan) any part of any of the other codes? Well, ultimately, a single TM is better, but this game Model may give ideas on how to break a diffl problem into parts: or how to use different models to solve different parts of the TSQ.

Sent 140.00

6/28/05
STM

On "Cyc" as an Approach to A.I.

137

00 : : Q: Is Lenat's approach in Cyc really bad? It seems to be inspired by
01 (McCarthy Quote: "In order to be able to learn something, the system has to be able to
02 be told it." In the case of human students, this is often not precisely true:
The trainer does not know the internal language of the student & can't tell him the soln. to a
problem in a useful way.

In order to be able to be very intelligent, a TM must have a lot of info.
To be able to read usefully, a TM has to have lots of info. Marvin & I guess Lenat
feel that once can get to this threshold by programming the info into TM.

My feeling is that we usually don't know how to do this very well. Essentially what
0 (ML) → Marvin & Lenat are doing is closely related to Conscious Mind of Man. As such, it
is very superficial & does not really approach do much that people can do.

On the other hand, a system that acquires info by discovery gets the info in a
way to tends to relate it to everything in the past corpus (as much as poss'l.)
The ML approach has simple performance of tasks as Goal (≈ Turing Test).
This does not usually make it more likely that the info is properly cross-coupled

Cyc might be taught to be able to find out gallons/day output of Amazon river by looking it up
in an ~~copy~~ encyc — or its own memory. A suitably trained machine would not know
this to Q unless it had found it to be useful in executing tasks in its TSCQ.

Another task: ML want tons of facts & assoc. info so that TM can deduce an enormous no. of things.
20 They conjecture that after a certain adequate set of facts/info has been obtained
the TM can just read & interact w. world & will learn more & be able to learn even
better.

Consider ML want to put lots of Info into TM., plus known logical ways to infer. Planning in the
new facts/info. All of the structure of the TM is well known, period.

If induction is used as basis, Amt. of info put in is much less. Its method of
organization is known only in a very general way, TM itself, spends most of its time
finding regularities in data, relations between various factual inputs. Only at the very early
beginning does trainer have much idea as to what TM has discovered.
So I want to put less into TM, but have TM try to optimally organize it — optimized for
0 max utility for whatever its problems were (& perhaps will be).

(.01-.02) is true for Conscious Comp only (& presumably small portion of it).
Long memory area, e.g. cortex, is not conscious. Attempts have been made to decipher
conscious long-term but not very successful. "The Zen of Motorcycle Repair" is
about trying to uncover Mind.

It is characteristic of all known long Systems that they eventually (usually sooner?)
become "unconscious": that the mechanisms of them, are not understandable to
humans — except by "local approx": AM, GA, [redacted] (BBN? ... SUM? [redacted] HMM?)

39 { Perhaps: the long system does stay conscious. Now in Understanding "it is a profoundly
Simulated System.

139.00

QATM solves IND and INV probs in same way! (.27 ff, 140.00 ff)

00: (spec 135.10) to R+ problem of assigning pc to string "A's" in QATM! T. idea was Backtrack "A" in T. TSQ involved a search. These srches could be "ab initio" - to find/short pgm. w. Qi as input Ai as output. — But we also want them to be "same pgm for all Ai's". — well, they all start out w. same Oj — some of them can "cancel" part of the Oj! Also, some Oj's may be real poor for certain Qi Ai pairs (but not pc=0!) (so we mite use "Bcosts" (convenience)).

Does larger set of all Oj's "↑ ssz" in a useful way — does it ↑ pc's of tokens by carry Common "Contexts"?

08 → Basically, I want to find a way in which a large no. of problems would give rise (roughly), ssz. T. ssz should ↑ w. no. of problem "types" (not w. no. of problems). Each new "type" — mite ↓ code length of T. Oj's.

10 → So T. Big Bottleneck seems to be T. "smallness" of 132.26

T. way. OS works: Assumptions: TM has small ssz for Oj: However. The CJS's aren't all quite "small" & can be dealt w. using present day mechanisms (PC's, say). As ssz gets larger, it would get harder to find solns — as ssz has ↑ so for ↑ difficulty of finding solns (since Contexts ↑ more info is available to be).

For a long time, I was considering 135.05-10 as a "primary thrust" for TSQ construction. More recently, I've been considering less carefully constructed TSQ's & "faster" computers — so CJS's could be larger than I had originally considered: E.g. Juergen's OOPS was doing CJS's enormously large — than I would have considered.

22.135.15: This would be an essential quality of any TSQ/TM that I could construct! In fact, roughly I could do it to make T. percent of Oj from 1 in size 2 to 1 in size 3 very! At first Oj will, indeed be quite small — but T. CPU IPC will be large enough to deal w. it. Also Oj will be usually small enough so that searching for relevant sub-trace contexts will not be very expensive.

27 Re: OOPS! It can be shot at in 2 ways & use 3 ideas: There were 2 problem types

- (1) Some simple (easy prob (Easy-prob) (2) Towers of Hanoi (Toth) that Oj parts consists of strings — in this case pc's for various Tokens (No context was used in OOPS).
- This string is then given (≡ Q problem desc) "LP" Large prob or TOH and a number that characterizes just which problem it is (e.g. how many Rings in TOH). Then TM has to find a R(andom) input that will create T. proper (known) "Ai".

In the case of OOPS Oj consisted only of the (3) sorts of tokens used in previous problem solns.

.27 looks like a way to represent INV problems within my present QATM (induction) formalism! For true Lsrch soln. to INV probs, TM would try function of Qi (Ci, prob desc) in pc order: T. "pc" being that given by Oj.

T. 2 examples in OOPS are common INV problems: They both have a "sized problem" parameter. (140.00 spec)

6.29.05

STM

Cyc v.s. TM approach 137.00 ff.

139

space
139.40

20 : A possible argt V.S. 136-39 : That MdL slant w. an understanding system. As soon as it is able to Read & understand Ordinary English, it begins to deviate from its carefully crafted input & becomes in many ways "Ununderstandable".

Perhaps my main Argt is that people is not a particularly good way to put knowledge into TM.

24 — That is why My TM puts. info in is "Optimum" in the sense (keys) it tries to get May pc for Corpus w. available cc. That max compression (including II codes) is the best set of "Explanations" of the data.

That .04 ff suggest that TM would be spending its time in the "best possible way" That the rest of the System depends on how good the TSC is: i.e. Is the corpus TM is trying to compress, a "useful" corpus? Does its sequential structure make it easy for TM to compress it (& find ways)?

0

0

30

6.30.05

5TM

30: (Soce 138.40)! This is peculiar — since QATM wasn't at all designed for Inv. probs! — Tho we can view INV probs as d-induction problems.

There is what looks like a peculiar Twist: If we [illegible] try to change/improve O^j: we will usually look at the pc's assigned to the seq. of R_i's — the possy that there might be a new R_i of (now, via the new O^j) hyer pc, is not considered (usually not, anyway).

If we had some reason to believe a new O^j would give hyer pc's for older problems, we would probably try them.

Another possy, is that in Lsrch we oversearch to try to find several solns: In this case a new O^j could look at all pc's of these several solns.

0 In this new view, the if used (A) ppen for O^j, we would have ssz, all of the problem solns this far!

12 Note that if I put the same problem into this TM as was would before, it will probably take about same time — No, not quite; the presence of the old soln will bias

14 the search — but not very much. But Get back to this!

Say we used [illegible] forg. TM formalism on S-induction, so some Q_i could have difent. [redacted] A_i's. Seems it would work OK. Note .12–.14, hvr. — It would tend to first output the old soln.

So the forg. certainly simplifies the conception QATM.! .12–.14, hvr seems a bit

20 weird! For INV problems — I'd like it to have some/strong memory of a prob. it solved in the past

21 The only difference betw. QATM use for induction vs. use for INV soln, is that in induction, the desired output is presented as a known A_i. In INV probs, the desired output is presented as a decision Algm (i.e., is a given A_i acceptable or not...

24 a special case is where the output is given as a specific (string (or sets of numbers)

Another difference in How this TM processes INV rather than IND probs, is that usually the solns to IND probs are rather fast, because the A's

27 are of hy pc.

If we give this QATM a lot of INV probs, it will, indeed, try to find various Macros,

30 sub-functions, that speed up the solns, to all of the INV probs it has solved.

It will not do this in a very clever way, hvr. The only ones it can notice are those picked up by APPM, say. We'd like a more universal kind of pd. being possl.

For IND problems: Again we want a potential Universal dmit-based (defined) O^j.

So Making O^j universal-based is the BIG Problem.

Making O^j = APPM might be able to work some induction & INV probs of interest, but ultimately we have to have O^j be universal-based.

spec
30: 140 40: **if** Q ATM used as $\exists$ func, it would, of course, be universal. T. search process in Lisrca is not immediately clear. We start out w. a small trial O^j & we see if we can get a set of R_i's that will "do the corpus". If O^j is universal (I stress we have to keep it
03 in universal: or at least have it able to express all process A_i's w. any possl. Q_y'.
(University will give Mat(alt/least) ... but Penrose more conditions it must satisfy).
So: I'd like to fix it so that T. O^j trials would all be either universal or do .03.

08 Well: How did I do it before T. discovery of 138.27 ff? (18)

09 (SN) One way: to get universality in O^j — or at least, approach it:
Have a fixed (or a set of fixed) prefixes for the R_i trials. So essentially
As time goes on, the set of prefix strings could get longer & longer & be more & more important. During this Growth of the prefixes, the d.f. via APPM would be correspondingly changed.
A trouble would seem to be that for each new prefix string or string Modifn, one must go thru entire corpus, testing pc's of A_i's. This may not be so bad, if pc's of A_i's are rather hy.
The prefix itself could contain definitions, [redacted] possibly biases on pc of various tokens (perhaps equivl. to a certain [redacted] (wt sense) of T. "precorpus"). The prefix could not by itself cause result in any printed output > 153.04

I think this is very nicely formalized in 153.04 - 11

18: (.08) Perhaps I had O^j as a single (prefix or set of) string(s) (the first of the 3 inputs to $\exists$ func)
T. second input was Q_i; T. third, R_i. These were 2 inputs to LISP or AZ, & I had
a (end) symbol that would slowly, adaptively characterize pc. Before R_i was read, Q_i had to be read, & no output
Call this string 'S'. TM$_2$ would assign pc's to each token in separately 'S'. until R_i started (or before).
The set of ('s) 's was O^j. I didn't think about TM$_2$ assigning pc's to
R_i's tokens — I think of R_i as being a random Binary string, say.
Perhaps in Lisp a/o AZ, there's never any output until T. stop token accurately telling us that T. function has been defined. In Lisp, Q_i would simply be one of: "Prim. Inputs".
I think that of R_i as being another (binary) input, but it seems clear that it could be seen as a segment taken of T. original function defined by "O^j" — which sounds very much like
(.09-.17) (!)

30 → But — go back and look at the recent Body of systems to realize Phase I,
31 (SN) APPM can be further grazed by including defns of (contextfree Nmcls): Can AIS or any other improvements of APPM give true universality?

Back to discussn of the recent "Breakthru", of 138.27 ff: The IND version uses PPM to make (some of) defns of contextfree nmcls, & subfunctions in the APPM version.
However, if the use a prefix string as in (.09) ff Lisp or AZ could make only kinds of definitions of functions or sets of strings.

38 A way to view it: T. old method of $\exists$ func induction had a string 'S (.21)
39 and a uniform d.f. over all symbols in R_i.
40 T. new method has 'S = NuH, & APPMish d.f. over strings in R_i. A natural (mixture
is λ

JTM

20: would have a 'S that grew w.r.t. Corpus Seq; as well as a d.f. on Rᵢ strings that would beat (at least) PPM or APPM or 141.31 — so one could, usefully have a Mixture — Rho in general .141.38-.39 (suitably ganged) could be adequate.

141.90 could be adequate.

But mixing f.s. a mite bit tricky.

An earlier tack was to use fixed-sized 'S', uniform apriori on R_i and somehow like PPM to extrapolate 'S (extrapolation 'S was job of TM_2). ← any of many induction systems for that.

The specific difference between 'S as corpus; and 'S of R_i as corpus! Well, not so surprising! 'S is a derivative of regs in R_i? Perhaps). Ideally, 'S should have no regs in itself. Any regs in 'S is essentially "Meta coding".

T. Thing to remember is the close correspondence between INV prob soln. via adaptive search, and {both d and S induction}

.12 A possible difference between INV & IND problems: In IND probs, the "goal string" is known & one might do "working Backward" or some variant of that. In INV problems, "working backward" is more difficult, because one doesn't know exactly what goal string looks like; One might know some of its properties.

T. idea of .12 (is?) also important in BU search. (? Is it?)

In BU search for inductive codes, one starts w. an induction code that is poor, but legal, & once tries to find a "better" (shorter) code: In INV problems, there seems to be no corresponding to induction problems.

20 It is notable that INV problems are normally not solved by anything like Lenat: Usually they are converted to an optim. problem — & various optim techniques are used to solve it. The conversion takes the Yes/No criterion for INV & makes it "Gray", so one has criteria for "closeness" to the soln. These conversions are somewhat of an "art", but, presumably, one could "learn" how to do it. There are often many different ways to do the conversion — some giving crazy solns - others giving diff. solns, or none at all. Often the way the INV constraint is derived, is part of a bylaw/"context" that is rather than Black/white. This can be used in standard ways to get a "grayness" for the original INV criterion

30 After conversion of INV prob to an optim problem, we can use optim methods to those used for IND. In GA, devising suitable "fitness functions" can be an important problem. We could start w. a INV problem — or an actually optim problem: In the latter case, we may want to change the Fitness function so that G.A. is easier. We may want, e.g., a normalized fitness function so summed all possl. fitnesses = 1. N.B. GPS solns of INV probs have a VECTOR fitness function ← Perhaps use of Ob. of Algebra.

35 What I really want is clear pictures of how diverse induction, INV. methods work — so I can compare them & perhaps understand them, & devise better ones.

.00 : Consider the Systems of 141.38 —

.01 1) S' is "uniform" D.f. for R_i'bits. So part of R string dependent on its length only. } This is the original Theme — so it should be correct!
S' did all of the induction — all of the copy finding

2) ■ No S' used. Instead, we have a D.f. over all R_i. Its the same d.f. for all Q's. In .01 we had same D.f. over all R_i's, also. ■ [See 153.04-.11 for Exact understanding | 1) & 2) are of equivalent power !

Seems that I went over this model of induction a long time ago! Decided it was bad, the first time I worked on it .... Then that maybe it was OK., the second time around! ... .
I'm not at all sure of this! ... The Q was: Was it a Universal System?
Seems like a paradox!

.0 Need to clarify just what I mean by "the same D.F." — Perhaps Rao the D.F. is indep of the Q. → 153.04 |
Perhaps I should write an exposition, explaining exactly what the problem was: — as to a New person, unacquainted w. much of "jargon".

Essentially what 'S does, is create a new machine (which no longer is Universal) — anyway, the "new Machines" do has a new D.f. over the R's.

.06 For each 'S, Q_i pair we create new machine (≡ function) a ∴ a distinct D.f. on output, P_i.

.07 — which seems like a very general solution to the QA problem.

.08 Next, consider a D.f. on all strings that are descriptions of functions of the Q_i's. Any D.f. of this sort could be regarded as generated by a "special machine": a "this special machine" could the one described by 'S : so this form is ≡ .06-.07 ! } Is this right ≡ 153.04-.11?

.0 So .16 -.20 may give a complete Understanding of "what's going on"! → 153.04-.11 is clearer.

.22 (1) 'S, Q_i, "Random" R → D.f. on A : 3 inputs to machine : 'S, Q_i and Random R. Gives d.f. on output strings.

.23 (2) Q_i, R → D.f. on A ; Q_i, U('S, R') → D.f. on A (U('S,R) gives a d.f. on R

"S can be viewed as a description of the D.f. on R.

Superficially (1) seems more "general" : (1) is U(Q_i, 'S, R) → D f on A.

(2) is U(Q_i, U('S, R')) → D f on A. In general, 'S, R seems more general than U('S, R)

.0 In (1) we have the machine {U, ('S, R)} operating on Q_i U('S, R) | U('S, Q_i, R) ≡ U(Q_i, 'S, R)
In (2) " " " " " {U, U('S, R')} operating on Q. U(Q_i, U('S, R))

In those cases (1) U('S, Q_i, R) U reads 'S, Q_i, R in that order, & knows boundaries between them.

.33 For (2) or U(U('S, R), Q) U(Q, U('S, R)) → U('S, R', Q_i) : U operates on 'S, ■ then R', producing R.
Which it then mixes w. Q_i to get output A candy. After R is produced here, U ■ forgets its internal state (resets itself) and abandons operations on R, Q_i.

.36 In (1) U('S, R, Q_i) ; U operates on S', R then it may remember an output, and it does remember its entire internal state for further operations on Q_i. Produced R' out

5TM

10: : In (1) we have $U(\bar{s}, Q_i)$ creating a unique R.E. set by enumeration of Q_i — a R.E. set that corresponds to a D.R. on A — so R.E. would seem to be "universal".

[In (#3, 33 consider $U(Q, U("s, R))$: $U("s, R')$ could be "s, R' in which case (1) & (3) would be r.e. sets! Does this solve it? "could" for certain values of "s. —] etc.

"s is of form; delete first 10 symbols of "s giving "s; Then $U("s, R) \equiv "s, R$ which means U must do "s Ron R, Then Q_i.

Q is: for every D.R. obtained by (1) using 's, Is there a corresponding, identical D.f. obtained using a suitable "s?

(1) can be of form $U(Q_i \Delta "s \Delta R)$. Δ is a special symbol that tells when an input ends.

So $Q_i \Delta "s \Delta R$ in a 1 input machine would simulate only 2 input machine.

(2) $U(Q_i \Delta U("s \Delta R))$. So; for only 's, can we have a

"s such that $U("s \Delta R) = "s \Delta R$? Normally, "Δ" is not in U's output vocab.

Another tack: Say $Q_i A_i$ is a r.e.Q: Can all possible compressions of $[Q_i A_i]$ be expressed in the form of varying $A_i = F_{i,j}(Q_i)$ i.e.

If $F_{i,j}(Q_i)$ is the j^{th} code for $Q_i \to A_i$, Then for each j, There is a set of codes $[F_{i,j}]_{i=1}^{n}$.

Say P_j is the set $[f_{i,j}]_{i=1}^{n}$, optimally compressed. Is $\exists P_j$ i.e. true

? Pc of the corpus $[Q_i A_i]_1^n$?

We can simplify the problem by letting $n=1$: (i.e. BAG-pred problem).

So $A_i = f_{i,j}$: $f_{i,j}$ is the j^{th} code for A_i. Then for each j, we compress $[f_{i,j}]_i$ optimally (note that there can be various ways to assign j's to the $P_{i,j}$ ... say we assign them in (any) lexicographic order)... Actually, we want to do something like compress the Bag $[f_{i,j}]_{i,j}$ — index of j order —

I'm getting confused about this! In INU probs, one could have only one (& shortest) code for each soln. Ideally, we'd have all codes (corresponding to INU.). We cannot now want to compress this seq. of $[f_{i,j}]_i$ sets. wtd via Pc or best

If so, since i doesn't vary, (false) there is only one set f_j. — No, it does vary: Equivalent to language extrapolation problem.

If it's not $n=1$, it's just that Q_i is constant — index of i.

Normal bag induction finds a machine w. random input to give observed UTM output w. max prob.

[for example of finite strings, a code uses shortest machine when to stop]

In this case since Q_i is constant, (1) & (2) are identical.

Look at #3, 33, 36: (2): $U(Q, U("s, R)) \to U(U("s, R)) \Rightarrow$ A d.f. (Assume $U(U()) = U()$.)

(1): $U("s, R, Q) \Rightarrow U("s, R) \Rightarrow$ A d.f.

Next: would it be of interest to have just 2 Q_i values? We could have an "s* or 's* compression for each Q value. If we had k different Q_i's Then $\exists [{}'s_*^2]_{i=1}^{k}$ ... k different 's's.

5TM

re: Carroll
Endless forms most Beautiful.

10: Then I'd want to find out what codes for set of IS^2's. Hmm, it seems unreasonable (?) to have to group the codes on the basis of "same or different" Q. Some Q's are very similar, but not identical.

Ph. basis of ② (143.33) was to iden but in INV problems: After one has solved lots of them, one should notice repys in soln. methods that enable faster solns for new problems. We observe many $U(Q_i, R_j) \to A_i$ so we compress the set R_j's so in the future, one will use materials in a better R_j order. A "smart" R_j, looks at its Q_i before & currently its A_i. — (The all R_j would definitely do this — unless it always starts out in a way that's indep of Q_i. .... Hmm I may not want to consider U as a "sequential" machine .... It may be easier to use Lisp or ATS, which always has "stops" at ends of inputs & outputs). A impt Q is: Is compression of the R_j set in this way .... does it encompass all possible types in this problem area?

Well, in the case of INV problems, there is little if any "info" in the solns. Given a INV problem, the solns are already theoretically defined. In some cases this definition may be "incomputable" — But L search will eventually find it, if a soln exists — in fact it will eventually find all solns. This may mean that the utility of ② (143.33) may not really follow from purely informational arguments about existence of codes, etc. → 146.10

T. Q of ① v.s. ② (143.33-36) seems v.s. to Kolmogoric expression of N dim functions as some constant/funct. of N 1 dim functions! (which I never understood!). (See 104.30ff 108.00ff)

Looking at v.s.
② $U(U('s, R), Q)$?
① $U('s, R, Q)$

in ②: If U still "knows" ('s, R) after computing $U('s, R)$ then it would seem to be same as ①.

does $U('s, R)$ necessarily have less info than $('s, R)$? There is no problem if the particular function is invertible — i.e. ∃ only one 's, R pair that gives $U('s, R)$. This would be true if 's causes 'sR also to be printed before any R output — which makes for very long codes! (no so good ☹!).

If ① & ② are not equiv, then that means that there is a better way to compress or use for predn. the results of many INV problem solns.

We want a ~~sharp~~ D.F. ⇒ it gives a hy P.F. to the Observed Q_i codes; pairs. essentially hy P(shortest code for soln | Q_i) (Q_i can be INV or INP problem). As a first approx P(short code for soln) could be a v.s. uncondl pc ... indep of Q_i. (i.e. ②)

E.g. Consider induction using APPM as TM$_2$: We get a prelim set of codes for corpus, (w.o. TM$_2$). We use rpm to find keys in set of codes (Uncondl pc). To get condl pc. of "rpm type", we look for "features" of the Qs that combine w. RPM's contents to give "layer contents" that give better predn. → 146.00 spec.

STM

Good, Simple version of Phase I (.14 ff

space
0 : 145.40 : T. Nature of the "features" being exploited is impt. PPM suggests substrings a/o subtrees in 'Q_i'. If we do find that we can do significant (or any real) compression. This way: it suggests that ① is better than ② (145.23), but does it prove it: Even if ② is theoretically adequate w. optimum 's', it may well be that if PPM is used for 's', that none will remain regrys that can be accessed by (context correla) of w. things in Q_i's.

.04

Re: ① & ② 145.23: I think one sort of ② is that UC's, R) is only 1 param & 's, R is 2 params, so they have more info. Not nec. at all: depends only on no. of bits, not on no. of params.

0 In line w. 145.13 -.19: Consider ①: UC's, R, Q_i)! As a set of possl. codes to produce A_i. In theory, 's can say "look at Q_i, use Lsrch to get the soln's in pc order. Then Print out the R^th most likely (shortest) code. This is certainly not what I had in mind, though it is a "very good" soln — perhaps best possl.

.14 → Possl. implications of foregoing stuff w.r.t 'Phase I':
Thus, as before, the Q_i contains both data (for INV probs) & simply the "inputs" for IND probs.
{I may be able to MIX the 2 kinds of problems!}
At first, use simple Lsrch to find solns to simple problems. On the set of codes that are solns
Then, after the no. of problems (not nec. the no. of types of problems) becomes larger enough (quite) this may still be true limitation: see!
use APPM to get PD to predict such fav solns. (Note this is quite different from looking for regrys in O_j).
① It gives hyper pc's to nite answers — so we end up w. a more accurate predictor
② it speeds up finding solns, since those solns will have hyper pc's. As O_j's grow in size, we may want to look for regrs in it. Try APPM because it is cheap & (at that pt in srch) readily available. If I can think of regry types O_j may have, try to devise tsq. for finding those regrys, & get TM_1 to work on it so it is TM_2.

.30 As further improvement of APPM, try 145.35 – 146.04

.14 – .30 looks like an easy approach to a preliminary "Phase I": It uses Lsrch for its primary induction (w a bit of APPM thrown in).

33 But such is a quite distant story: (perhaps) much of TM will be involved in Lsrch — but often both bus & Lsrch will be time shared when possl.

The .14 – .30 looks easy, it is not: We still want to use $\{^{q}s\}$ for QA induction → 147.00

7½ '05
5 TMe

~1930 ff SVM's: Digital inputs: (.30)

147

spec
00 : 196.40 : This is definitely not an easy srch. We try various $'s^2$ in pc order,
Then we try generating $+\sum A_i$] corpus using the $'s$ "prefix" : More exactly we use
$U('s^\bullet, Q_i, R)$ for trials to get A_i. ($U \equiv$ Crisp or A2)
For a trial $'s$, we run a few Q_i etc. until we find our w. an obsessively low pc (say ~00
for the given C.B.) — Then we do next $'s$. If a great $'s$ that does most
Q_i, we do retain it (perhaps by modifying abit).

Expo "on Strong AI"
102:00 — 103.90

07 → It is possl. to use $'s \equiv \Lambda$! If TM could find soln. w reasonable pc, we
Change to T&Q. We would want to use a very augmented PPM & certainly 145.35 - 146.0 of
T: "models" are detected/discovered by PPM as common "contexts" (including sub trees)
0 → We'd want to generalize. PPM as much as possl. — including generalizing on how to use Q_i as part of "context"!
.07 is ≈ what OOPS uses — The human didn't give complete problem
demos & he didn't carry context info from previous probs (just "Qboost" inst), and he
didn't use context at all. Hm., he did (probly) use a complete lang & Lsrch.
If he did give complete problem demos, TM system might be able to recognize "similar" problems
& ∴ better able to use Qboost" intelligently — w. more discrimination & hyperpc.

17 I had been previously considering SVM, ANN, RANN, GP, Decision trees, etc, 105.00 105.11
as useful for perhaps prelimy TM2 s/o in Phase 2 .... ? I just don't remember how I planned
to use them. Conceivably on reflection I might use them where I plan now to use APPM.
0 Check out that old stuff! 98.30 Manthous' Secondary Induction" — was this TM2 ?

So Mem reviewing problems ① ② BU srch 196.3 ! (When & How ..).
I think it likely that methods of .17 were meant for TM2 (ie. "Safe induction".)
Re: Methods use for "Safe induction" : Can also be used to obtain P.D. on safe-to-final strings for uncomputable —
for which I originally thought of using PPM ! See 97.26-.29 for TM verify.

30 SVM or SVM w. (Digital) Binary (discrete) input! we can use multinom polynomial U.L. terms.
we can use alphabet is ±c or 0 & 1. Say we use
~~Cubic~~ general Cubic or Quartic eq. (call terms up thru cube or quart)..
If coefs = 0 or 1 or ±1, computing values will be merely Boolean operations. This way of
using SVM may or may not be a wise idea! — But it sort of does map boolean domain into
pc's below 0 & 1 !

35 To analyse t. CQA problem: Alternative tractable way: $P(A|Q) = \frac{P(A,Q)}{P(Q)}$ (or similar).
we want ~~P(A|Q)~~ from $P(A_n, Q_n)$ from $[A_i, Q_i]_{i=1}^{n-1}$ which is something like $P(Q_n|A_n)$ from $[Q_i]_{i=1}^{n-1}$
and all abt. codes from Qbeebe. Q_i to A_i for i's relatives → 148.00 space

STM

O: (Speed 147.40): Normally, I think of 's as a form that looks at Q_2 & converges to PD for A_2.
— But may not there be other ways to represent the corpus?
If the actual best set of 's's (comes from completed ALP) are used, It should be impossible to compress data by compressing codes used. Here if we set 's to Λ, then there may be ways in the codes. — (compressibles).

Q: If I have a set of codes for a corpus (say simple Sequential Corpus), that is inadequate — in the sense that → additional codes of significant pc: Does this imply that the codes that we have can be compressed? Well, in theory, yes: Any code of set that was not been included in the set of codes can be regarded as a compression of one of α's codes, because it represents the same thing (not necessarily shorter, but the sum of the codes is a significant in pc & ∴ a "compression".) — Note that these compressed versions of the α codes is to be added as all codes; the α codes are still retained — but they will not have much wt. if the compressed version of them is much shorter than they are.

Now the "correct" way to do QA 13 $U(S, Q_2, R)$; I have been thinking of this as "U operates on 's to create a new machine, that operates on Q_2, R. — or the machine $U('s,$ operates on Q_2 & then creates a new machine that operates on R
i.e. $U(Q_2, U('S, R))$, but could change order to $U(U('S, R), Q_2)$, which is probably same as $U((S, R), Q_2)$.
Note that in Lisp/AZ, "Order" of args is not important — the args are seen in all (simultaneously).

Consider Bag induction: $U('s, R)$. We could restrict this so that the function must access 's & R in that order. That after R has been accessed, 's can no longer be accessed? (Suppose U put 's in RAM?) — so $U('s, R)$ can impose an "order" on R, 's —
— On the other hand $U(Q_2, U('S, R))$ is different. $U('S, R)$ is completed (← R) & we then compute $U(Q_2, R)$. I had been thinking of $U(('S, Q), R)$ some how as different. different by certain sorts operations on R. It would seem that $U(S', Q, R)$ would be more general, — but I don't see just how! If it is not then $U(('S, Q), R)$ would be equiv to $U(('S, R), Q)$.
Well, when I wrote $U((S', Q), R)$, I wasn't thinking of $U(S', Q)$ as being a value — but as $U(S', Q \cdots)$ being a function. But any function of a single arg can be written as $U(Z, \ldots)$ w. suitable Z.

Question: Given $U((Q, S'), R)$ and $U(Q, ('S, R))$, for any S" can I find a S' ⇒ for all possible Q, the 2 diff 's are the same? $U((Q, S'), R)$ v.s $U(Q, ('S, R))$
assume that U can compute $U(X, Y)$ for any 2 inputs (subject to partial recursive (limiting) sources).

STM

.00 So, does $U(U(Q,\ddot{s}),R)$ correspond to $U(Q,U(\ddot{s},R))$? (one of which can be Λ)

The universality of U means that if $f(x,y)$ is any function of 2 args then

$\exists\ \alpha \ni U(\alpha x,y) = f(x,y)$ for all x,y. α being a member of the prefix set

So α runs where it ends. Also $\exists\ \beta \ni U(x,\beta y) = f(x,y)$. β also is in prefix set.

The problem is not symmetric (I think): say for every 's $\exists$ a "s" s.t. it's true:

but it if it's true for a s & "s", there may not be a corresp s' for which it's true.

Consider $U(U(Q,\ddot{s}),R) = U(Q,U(R,\ddot{s}))$ or $U(R,U(Q,\ddot{s})) = U(Q,U(R,\ddot{s}))$

It may be possible to consider only sym, U's i.e. $U(x,y) = U(x,y)$.

.10 & or $U(R,U(\ddot{s},Q)) = U(U(\ddot{s},R),Q)$. Say $\ddot{s} = \ddot{S}$ are in prefix set. Also R and Q.

$U(R,\ddot{s}Q)$ so $U(R,U(s'Q)) = U(U(\ddot{s}R),Q)$

so & r. Args would have to be r. same [unless we consider ~~equivalence~~ equivalence of r.D.f.'s — which seems much weaker a cond.tion)

So say, equality: $R\,U(s'Q) = U(\ddot{s}R)Q$. or $R\,U(s'Q) = Q\,U(\ddot{s}R)$. since U is sym.

Since 's & $\ddot{s}$ correspond coordinates. F' & F'' say $RF'Q = QF''R$ seems very unlikely!

$F'QR = QF''R$

Re: 148.15 : $U(\ddot{s},Q_i,R)$ as r. correct way to do QATM! If $U(\ddot{s},Q_i\cdots)$ creates a Machine then no info is lost, it's not r. same as $U(U(\ddot{s},Q_i),\cdots)$ since

.20 $U(\ddot{s},Q_i)$ can remember both 's & Q_i.

.22 An Older Idea: In INV problems I certainly expected to use $U(Q_i,U(\ddot{s},R))$

I did have some ideas on how this might usefully get Q_i info into r. soln.

.24 There was r. idea that almost all Heurs could be put in r. form of Modifn of r. "guiding" PD" for LSRCH.

.25 But should it/Modifn of r. Pd depend on - nature of r. Problem? i.e. Heurs do modify r. search ordering, but Mainly in line w. "Nature of Problem".

So while we could do RPM on r. R's as a first Approxn, We will want to use Q_i info as well. finding Good "features" or useful functions, will be quite

.30 diff'lt. Using r. 's approach for both IND and INV problems, would seem best. Hvr, while this discn. of .22 ff has been revealing, it is likely that INV probs will not be solvd much by LSRCH. 142.20 ff discusses Conversion of INV probs to Optzn. Problems.

.24 - .25

So it's r. impt. point. The main reason I got into QA is was that INV probs use .22 $(U(Q_i,U(\ddot{s},R))$, in all of my original work on LSRCH. I did have r. idea that "all Heurs could be put into r. P.D.", but how to include .25 was perhaps unclear. Perhaps I felt that Heurs would be discovered that would enable TOM to take r. nature of r. problem into Account

5TM

.00 in a more direct way.

I think that I will use both 's' & 'E': First 's in early TSP: then 'E' via APPM when sszgets large enuf

Updating 's will always be a Big problem. Finding Happy in 's via various "Monte Carlo" methods will help, but I will have to find good ways to try mutations & crossovers. It may well be that I'll have to go to BU search rather early, because large 's will result in excessive cc for a bi-mito update!

.10 An alternative to Darge would be Sumac, if I can get it to work — I did discuss this possibly — including possibly of Intrapreneurial Backtracking (if ever) Sumac is sort of Mutation & Crossover. Mut. & Crossover is a Genus of Sumac; Using a Grammar is a Genus of Mut. & Crossover! So I might be able to use Grammar in L search phase. — as well as in BU phase)! — Or perhaps this is a way to Mix L search & BU search!

It may be possible to use L search in Sumac in a Big Way! Say $[c_i]$ are various codes for Corpus thus far ($[c_i] \leftrightarrow$'s). We get a new QA: to update 's"! We use "small modifications" $c_i \to c_i'$ & c_i, c_i' hardens not much longer than Moscow c_i. An equiv. view. $c_i \to c_i'$ is a P.D. (="Mutation")

.20 .21 .22 we learn how to do this Mutation via the corpus of $[O_i^j, O_{i+1}^j]_{i=1}^{n-1}$. This is an Phenotype not Codes or Genotypes. Do we want the Genes or the Objects to be "close"? Maybe the same thing! It is! Into distance below. Objects & their codes is "off". All codes for same object are of distance ∞ from each other (?).

Re: .21-.22 "Learning" Mutation: Deciding on Corpus is a difficult problem. If we want to use mutation $O_i^j \to O_{i+1}^j$ as part of corpus, On one hand, it is tricky to do: On the other hand, we will probly compare object by comparing

.30 their codes, and each object has many codes. Well consider the shortest codes we know. Also (But also note that we will usually have several O_i^j (for each ssj pair). Usually the O_{i+1}^j will have its code derived by mutation from a code of O_i^j. So, we will keep a corpus of mutations that were "successful" in the sense that $O_i^j \to O_{i+1}^j$ gave an accurate O_{i+1}^j.

.36 However, if we start out w. L search to obtain a sequence of O^j's — the sequence of codes of O^j's will not have been obtained by Mutation... but we can use them as a corpus to get a "preliminary" mutation distribn!

5 TM

LEARND MUTATION 150.36–151.00 ff

.275" spacing. Optimism Re Sumac .05

00 Presumably, we will have a fairly large corpus in LSRCh before we start to use Mutations to get trials. T. mutations will be tested in

03 [redacted] "C JS" order.

This "Mutation" idea seems much better than simply trying to "concatenate on" trial combinations of t. formerly adequate OJ

06: 11.9.10 on SUMAC I was thinking that Sumac might work if we kept enuf parallel trials — but w. "Learned Mutation", it seems much more likely to work!

This "Learned Mutation D.R." is very impt à it might be well for the trainer to use a "heavy hand" in this — by studying the corpus and trying to find xforms, etc, that seem to work. We can then "wire" these into TM or try to train TM (or TM_2) to discover xforms of these kinds.

So we might be able to get TM to work on t. problem of getting a good Mutation D.R. : If it is in t. format a standard Q→A problem.

REV

Some Recent Impf ideas:

1) Learned Mutation Dist and how it makes Sumac More likely to work. ~ 150.36 – 151.00ff ... (150.10 / 151.06 on Sumac.) (On Pictorial input SVM's! 147.30)

2) A better Understanding of Soln of (Note 140.21–.24 INV probs via Lsrch. 149.22ff) 125.34 "T. forgg. Suggests that (based 'Warped Shape'.
* In particular, that the way I'd had of doing Lsrch on INV probs, using a suitable "Guiding pd" was [illegible] as an approx., if I still used the Same D.f. for all probs. To do it correctly, the D.f. would have to be a function of the nature of the problem. From the analysis preceding 149.22 :

3) → I.E. 138.27 ff to ~ 149.22 : T. initial idea was that QATM could solve INV problems in its normal course of work. While this was certainly true, it became clear that the Mod of "Adaptive Lsrch" that I'd planned for INV problems was "imperfect" by a long shot, but that the Mod I'd planned for QATM was good & would work correctly for INV problems. (Actually, this turned out to be CORRECT After all! – 153.04–11 also 141.09–.17)
That the "guiding pd" was (in QATM), a function of the problem definitely, Q_2.
146.14ff (.30) discusses an approach to "Phase I" that I considered, before my understanding of 149.22ff → (.09). 146.14–.30 is not really "BAD" – it may work to some extent as first, for small TSP's – but eventually, I'll need Guiding PD's that depend on the Q_2.

4) On Conversion of INV probs to OZ probs! Lots of standard Methods! 142.30–.35 (Also Note GPS)

5) SM 133–134

6) Expo: 132.30 "Greedy Science" 132.31 "Advice to a Young Scientist."
135.25 "50 yrs of A.I" (since Dart. 1956).
130.20 How Non-Baysian Statistics is serious Impediment to Strong A.I.

7) How to use various Induction Methods that have been Developed by the A.I. Community. 120.09–.19; 126.10–132.29; 135.00–.21; 138.00–.26;
Ideas on How to "Mix" various Methods. "Coding and Recoding" (I have a recent note on this that I've not been able to find: Idea is that "recoding (≡ "Meta coding") always simply adds to the set of 11 codes for the corpus).

Other than a few simple Cases, I don't really get a way/understanding (i.e. of how to "Mix" various induction systems (of 11 codes or "Coding and recoding" when that was poss!). Tried to xfm a p.d. into useful codes, but it didn't look useful — e.g. Run Coding ⇒ it misses the "11 codes" (maybe still useful for "recoding"; hmm., but → (Spec 15300)

Rev(100)

space 52.40

0 It is same as Compression, then using two Rx compression schemes. Getting back to pc's of the original sequence would seem to be diff(t)/"messy".

The each compression is invertible. The sequence of compressions is equivl. to a single compression

.03 → A "SOLN"! Say final cascaded Compressd string is binary: s .20

.04 .1 4.17 → SN on the $U(Q_i, U(s,R))$ Question: Suppose that "s arranges so that all R's (w.pc >0) have a prefix "s, which acts with Q_i, to xfm the D.f. in a way that depends on Q_i. The result would be then/the same as $U(Q_i, 's, R)$!

.07 Is this true? we get $U(Q_i, 's, R)$, where R is an arby d.f. on codestrings. If 's is a member of a prefix set, then we can always decompose .07 into $U(Q_i, 's, R)$.

In (38.27 − .40, 140.00 − .27) we discover that QATM can solve SQUIVEL INV probs w.o.f Modify.

10 I claim all 's & 's must be expressable as prefix free members.

1) So this seems to say that 's is unnecy, one can just find copys in the corpus of codes for "A_i as a func of Q_i". So using APPM & any further improvements it might

.13 s/o Many Pems w/ ou Rx corpus, would be o.k. → 157.18 So I could be

.14 rather close to being Able to Start ptng "Phase I" now.

.03 → We want the

0 Say we want the p.d. of the string s.

Let a_i be the i^{th} binary string of length n.

Let $b(s a_i^n)$ be the no. of bits in the cascaded compression of $s a_i^n$.

.23 Then $P(s) = \lim_{n\to\infty} \sum_{i=1}^{2^n} 2^{-b(s a_i^n)}$

To get pc of continuation $s \to s\phi_j$:

$$p = \frac{P(s^\frown\phi)}{P(s^\frown\phi) + P(s^\frown\sim\phi)}$$

Unfortely, this is very expensive to approximate — there we might try to find continns of s that "compress well" & use only those ones.

Summation of .23

0 → Another tactic is to try to devise compression methods that somehow, include all possl continns of the corpus. — e.g. P(s) assigns a pc to each s_i, so if can → a compression scheme. I.e. "P(s)" really means $\sum_i P(s^\frown a_i)$ where a_i is the set of all finite & infinite strings.

.42 This St. technique amounts to [closely related to Cover's "Extension Complexity"]

Perhaps there is a General Approach in which we only

Optimism re the U(Q_i, U("S, R)) Model! : .21 | Also Note 150.30–151.00 ff! on Lrnd Mutation.
Optimism on Nat

Consider prada methods that take into account all possbl. confirms.
Most of v. Models in Sol64a did this: (f. ones based on General, Trees w. Bidirectional I/O, tapes did not) → Actually 153.32 does this!

SN on Dirichlet D.F. (PS: 7/8/05)

$P(\vec{x}) = \prod_{i=1}^{K} x_i^{\alpha_i - 1}$ [constraint $\sum_{i=1}^{K} x_i = 1$] Gives a vector $\vec{x}$, K dim', P's prob. density f'ct. over a K dim. space..

$B(\vec{\alpha})$ is normn const $= \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} =$ **B(α) Gamma Beta Funct.**

normed $f(x) = \frac{1}{B(\vec{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$ $\Gamma(x) = (x-1)!$

It is of import. in Extrapolativ & Bayes Seq.

Also "PlanetMath" and "Answers.Com" are V.G. sources of Math artcls. MathematicaSite is Good to some extent, but not so "readable"
Wikipedia may mirror some of these or Give its own discussn.

This U(Q_i, U("S, R)) model sounds V.G! In a TSG, it will not be necc'y to Backtrack (ever!?). We just get more & more vagys in t. U("S, R) D.F. : If we find that we can't find a soln. for a problem w. acceptable cc, then Modify t. TSG : TM is not yet ready for that problem!

Its not clear that .21 will work even so very far w.o. lots of very Hy level APPM. Possibly try Induction methods of various kinds on OI's creat'd.
Also, Perhaps try GP on t. U("S, R) corpus.

While .21 sounds V.G. — 150.36, 150.00 ff on Lrnd Mutatn for trial OI codes
Also looks V.G. re : Unclear how one could use both it & .21 [simultly, hrr...
perhaps try BoPs re individually.

Well one can do both simultly: Use('s) & modify it by Mutatn.
Use 's & Modify it by APPM.
"Learnid Mutatn" amounts to a (s-grammar for 's) and in corporates

(NO) → see 155.08

10 Genzn of "crossover" - so it's a very general kind of thing!
Ideally, we'd want something like that for "s" as well — but sse is larger so we can use it much faster, APPPM. — eventually we will want to use "Generalized GP" — using S-Grammars instead of ea mut/cross.

In s-Grammar recog: keep Lookup Order info for target lang. to facilitate "PPM-like" discovery of common sub-strings/sub trees

08:154.34 ! No, that is Lund Mutation is not a s-Grammar for's: it is instead what we get from a QATM! $Q_i \xrightarrow[\text{distbn}]{\text{Prob}} A_i$. Given O_n^j ($\equiv Q_i$) we get a pd. on O_{n+1}^j ($\equiv A_i$).

Another View of what "Lund Mutation" does: $[O_i^j]_{i=1}^n$ can be regarded as a time series, & we want a p.d. for O_{n+1}^j

3 Actually "Mutation" is correct: Mutation does make a p. and O_{n+1}^j a function of O_n^j — But it's Not [illegible] a "s language"

So .08 is correct. It is quite distinct from s-langs that are used to improve GP.

Actually, it's better than s-langs for GP, because it looks at previous Good O^j's & tries to improve it.

Re: $U(Q_i, U("s", R))$ form of QATM: Can I prove fast Convergence Rate? Probably 153.04 - 14 seems to imply it is equiv. to old QATM for which we have proof convergence.

.22

.23 SN TSQ: ANL a/c Mutation Lang: $3\left(\frac{\div}{+}\right) =$ If learns 4 codes $\underset{\text{num is sum}}{\overset{\text{prv}}{\text{sum}}}(x,y)$ but each w. pc = ±. Then it looks for ways to ↑ pc — regs in codes: Have 4 regs: two of form sum

$\alpha \times \beta \; y \; \gamma$. We have many cases of this w α, β, γ being "somewhat" eq. same, but x & y correctly $\left(\begin{matrix} + \to \text{sum} \\ - \to \text{minus} \end{matrix}\right)$ acct: The parts of α & β that are different are "numbers" so that aspect remains the same.

29

0 common recur We must look at all cases w. "−" in "x" position — what does regs of code have in common x & y are reflected by pc symbols: α β γ are all "numbers" or a constant symbols. → 156.16, 29

33 Note: looking at [Q_i, code_i] as corpus is a New IDEA!

34 Before, I regarded only [code_i] as the corpus.

TM

That is an interesting idea! — [redacted]

However, it does, give P.D. on the (Q_i, A_i) which is not quite what we want.

We want $P(A_i|Q_i)$ not $P(Q_i, H_i)$. If Q_{NEW} is an unexpected Q, then $P(Q_{NEW}, A)$ will have small P.C. However, that's not relevant: A person wants to recognize $P(Q_{NEW}, A)$ for a constructed Q_{NEW} & various A: This means what it we want a normalized $P(A)$ we have to get real. (to evaluate for entire corpus)

P.C of many A's: very expensive!

However, the exact form of the Grammar of $[A_i, Q_i]$ corpus, may make it possible to get $P(A, Q)$ from it. E.g. if it's a sequential PPM grammar, there would be no problem getting $P(A,Q)$. This sequential PPM grammar can be focused to use fine-tuned user contexts: — But a real problem is "Parsing" the Q_i. —

Well ↑ This seems very difficult, yet if we want TM to be able to "understand" Q's it will have to learn Parsing. It will enable "subtree contexts"

If we don't use subtree contexts included into Q (but only in A), we can still use Reis Grammar (more limited) on (Q, A) corpus to easily get $P(A, Q)$.

155.34 → Another kind of Raggy I'd like to get is 155.23-.34 ! But this still needs work! (p. 21)

An interesting point! If we do .16÷.15 (or even if we parse Q) — we get MAX P (term for A) as a function of Q (or subcomponents of Q). — which is a double dependancy on Q! Just what does this mean?

Q

21 | 155.34 (17) → ON TSQ writing: Write TSQ for Algebra (start w. ANL): write down (in English) human thoughts used in (2 dequoters for) that TSQ. Put those Human Heuristics/Search models that QA TM uses. Then Make language to Generalize/Interpolate/Extrapolate those heuristics.

Look at AM again (try to find My old Notes). See if I can actually use (& port to a such scheme) (& to Get TM to run) These heuristics. Not how AM implements Rank (of this info is available — Perhaps look at Cyrano — May be a start to Understand.

Another Task: Teach idea of certain x-forms leaving leaving Value of expression leaving "irrelevant": from this idea: various strings of x-forms can move toward "simplicity" & eventually a "number". Idea of "α is simpler than β" an approx concept.

More general: Write TSQ that is an ordered list of differences subgoals.

(These can be → I really don't know details of how the pgm. should be (found constructed).

Use much "computer processing capacity". If it doesn't find 158.00

7.11.05
STM

Limits of Predictability (vs. dictability) w.r.t. cc Bounds.

Say we have k p.e.m. — all of = wt.
Each takes exn to compute/uses for n-bit corpus. (cond.)
Consider a predictor that uses many/all k predictors! It would take you k·c·n to run them.
PAM$_{RH}$ We can create a new/(deterministic) Pem that looks at & chooses one | most distant from mean.
Pam$_{RH}$ is a very slow, hvy, factor of N slower than others. — But the expected value of its error is always $\leq \frac{1}{2}$, indep of k. Since alg deterministic we have very narrow Ensemble to consider.

If we ↑ no. of pems by 1 & cc for corpus of length n, we have to include very (exponential) no. of new Pems.

If we include all pems w. cc of computation $\leq cn$, then the no. of pems to be considered, will grow w. c, but I don't know how fast.

(Note changes on 10 October)

7.14.05

Very Imp Q: When I get a P.D. on the R_i's, $(U(Q_i, R_i) \to A_i)$ $(\equiv U(Q_i, U(S, R_i))$ (S' ≡ S)
If it were a Universal D.R. — ~~Bayesian~~ or a ALP d.f. — fine! But it is not! It's a PPM df on A PPM or source D.f. obtained by some pems. Just how bad is this? In the Lsrch for O^j model, the system was slow, but eventually, it would find any possible Generator of $Q_i \to R_i$. Here, on .15, we are using a PEM that is definitely not ALP, but we are doing a d.f. on pems for a UTM. If we used the "TM" trick on 160.30 ff to obtain the D.P. for $Q_i \& R_i$, we could be using a universal D.R. → 162.26-.27 May look the answer!

In General, I DO need to understand .15! At present time, my guess is that it may not work! If it does work, it's a great start to S's now rewritten as S'; "S now written as S": I made too many errors by miscounting S's, "S'(!)

Also, note the "Improvement" of 155.33 & -.34: using $[Q_i, R_i]$ as corpus for ($\approx$) PPM!

Great Breakthrough "Breakdowns" ☺!

So: 3 methods (1) use of O^j for QATH. find O^j's $\to \sum_j \prod_i P_{O^j}(A_i | Q_i) = \max$ (generate O^j) (used in IDSIA report)

(2) find/(using Lsrch) short codes, $R_i \to U(Q_i, R_i) \to A_i$. Then find regularities in the corpus $[R_i]$ use these (probabilistic) regs to speed up srch for new $R_{n+1} \to U(Q_{n+1}, R_{n+1}) \to A_{n+1}$. [See 153.04-.14 for "Proof" of correctness.

(3) Same as 2) except that we look for regs, not in corpus $[R_i]$, but in corpus $[Q_i, R_i]$

(3) was suggested by Heuristic of 155.23—.32; Note 156.18-.20!

→ 162.00 (spec)

1156.34 soln. in reasonable time, try to "factor" soln., & print resultant Sub. Goals.

Or just study problem, try to see why it is diff. for TM.

Or branch TSQ to a distant problem seq. at that pt.

Get Ramanujan's book (Y - one he used to read) to get ideas for TSQ.

G. S. Carr Synopsis of elementary results in pure mathematics
1886 originally
2nd edition
Chelsea 1970 Ram. Induction 1902 Born 1887 (Third Amazon, Barnes & Noble)
1886 GB Macmillan and Bowes, 935 pp Norway.!
ASIN 800086 CLVM
any date?

AND a very impt Trick!: Ab initio: Start out w. a large corpus of problems & solns. These need not be very close to the Beginning of the TSQ that we will start TM on. It is to give TM a reasonable set of wts. for tokens & impt. subtrees, etc.

With a Community of people trying to teach TM stuff: If I have trouble getting TM to solve a particular problem: See if the Community can find a partial TSQ that can lead to a Soln. for that problem. Maybe not so easy to do; because each TM will have had a distinct TSQ leading to the problem! Still, the other "Solns" could be useful hints.

If our TM seems to be beating all others in many problems we should examine its TSQ. leading to the present state of affairs — try to find out why it is good — perhaps use some of in our own TM's.

Another possible trick: If I know of a soln. to a problem, but it is excessive CJS. at that pt. in the TSQ, then the pc of all of the tokens used in the soln slightly! Enough to bring the soln. into the achievable. Or simply give the soln. to TM as part of its "already solved" problems. If the CJS is far out from available CB, then it may not be useful to add its soln. to the "Solved" corpus.

So it really looks like I'm ready to start coding!

A rather weak pt. at present: The details of how to implement the UMC, use what langs/primitives to use, etc.

Originally, I expected to write the TSQ, write solns to probs in English, then design a suitable set of primitives. — which is quite distant from Most recent Idea of Letting TM work on probs that I don't really know how to solve.

A kind of Compromise would be my writing the "Solns" to Problems in high level "English".

7.12.05

STM

Face in the Crowd - Ave

(Proc 158.34)

- One approach to ANL: Start w. 155.33 until x ÷ y is found. Then idea of certain expressions having "value" property. The idea of values of subexpressions being substitutable into larger expressions, leaving values of larger expressions invariant.....

Say $V(\alpha)$ is the "value" of string α: We get TM to run to evaluate V for a fair variety of cases. — and the recursive ideas in evaln. like $V(\alpha+\beta) = V(V\alpha) + V(\beta))$ etc.

(.04-.06) Seems like a good approach to writing the TSCA.

$V(7)=7$..... Tricky part for many strings α, $V(\alpha)$ will have a numeric value. — later consider literal values

Various interesting "learnable" domains. Integers, primes, LCM, simplicity, solve, adequate, differentiable, partial diff., irrational,

→ Th. Tug. Sg. in Lincos!

SN On N.L. regression [Models] If we start out w. short codes for discrete parts of them, we can use short corpus & random [antagonistic param.] values to test those models. — Hyp., d.f. of contin params is Broad for short corpus. As we ↑ corpus length, we zero in on narrower param d.f.

Hyp., as corpus gets larger, we should also consider more complex discrete models. — There are very many of them, & they will not give good compression w. short corpi. This seems to make method very diff(t) Well, we could use short corpi for larger discrete param models, check at compressions & estimate what compression would be for larger corpus — then expand to larger corpus.

SN So I want to write an outline of my plan for Phase I, thus far: w. refs to how to continue w. Phase 2. Also some discn. of Busy!

5 TM

REV.

:NB: A Serious Objection to 160.00–161.10 is 162.25–.27
The method received on 164.00–.20 deserves w. this Diddy — see also 166.08–.30 167.06–.25 on how to incorporate BU search into Lsrch.

Rev. of my Plans for Phase 1; Some discussion of BU search; & Some Plans for Phase 2

1) The TSQ. Start w. Elementary Alg., ANL (Alg notation Army); see 156.16, .21ff, 158, 159.00–.10. for good ideas. || 158.27–.34 looks right

Re. TSQ writing & reference UMC selection

156.32–.34 is important idea, Also General form of TSQ in "English" to start out.

Try to Find other ideas on TSQ's.

Getting TSQ's from MAPLE & Mathematica Definitions seems v.e.

.06 Possibly Lincos, re TSQ! : Also note 154.21–.25 on writing, running, TSQ's.

.07 2) QATM; Phase 1: This is somewhat different from the device in ID$IA report.

We do feed (Lexicographically) in a seq. of $Q_i A_i$ strings a/o no. pairs, Hyp., Given the pair Q_1, A_1; And reference Machine $U(\cdot)$, TM does an Lsrch to find a short code R_2, s.t. $U(Q_1) = A_1$. We do R_1 for a larger initial set of Q_i, A_i.

We may give TM some actual Q_2, A_2, R_2 solns. That would take at too long.

When TM has a fair no. of examples, it looks at the set of Q_i, R_i pairs & Uses it to get a d.f. on R_j as a function of Q_j. At first this is done by a simple version of PPM (from sequences of symbols to unordered set of string pairs. ... & string pair $Q_j \Delta R_j$ (Δ is a/ space (one Marker) symbol)' is treated as a single string, for extrapolation purposes. Given the string $Q_k \Delta$ & PPM is asked for P.d. on next symbol. If it is α, we then ask for the p.d. of symbol to follow $Q_k \Delta \alpha$ ... ect.

We do search via Lsrch., to find A_k's .... we usually for PPM

I'm not sure of this but it may give larger ESS

"By itself"

$164.00–.15$ Seems to do w. this Diddy!

N.B. 162.24 –.27 Seems to Make it clear that (.07–.21) is Not Universal. ∴ Buddy quoted

get several A_k's beyond the first A_k found — we can get the various RK continuations for Q_k in our corpus — Give less wt. to longer continuations.

We will probably want to augment PPM to recognize various substrings as contexts & eventually "substring sets", discontinuous contexts (as in ∃ ∀ ∃ or 3 dimensional scenes or in 155.23–.29 or 155.30–.32)

Also, we can use any other sequential prediction schemes such as ANN, (perhaps RANN w. GA. for updating), SVM, GA., GP. These can be used by doing a P.D. on R's or by an alternative parallel code to PPM,

In view of 161.02 we may want to do it that way.

actually, we can have a separate TM in that corpus!

.29 At a certain pt. (if TM has developed TSQ) we can use the past Q_i, R_i solns as a "special (sub)corpus" for TM & ask it for the P.d. of R_k as a function of Q_k. (Recall a hypercomputer TM is definitely not the same as the P.D. of A_k as a function of Q_k. (?)) — R_k is the "understanding" of A_k as a function of Q_k. I think it has more "info" (given a UTM) than A_k has.

Needs more Prob!!

.33 .29–.33 Sounds like "TM_2". To get it to be a real $TM_2 \stackrel{?}{=} TM$, we have

5TM

Rev ; N.B. see top of p.160 !

Spec 160.3F to make the corpus $[Q_i, R_i]$ perfect w. $[Q_i A_i]$ corpus I we might do this by putting a special mark/symbol after Q_i in the $R_{i,j}$ selected pairs.

02 Now a smart TM would notice that $U(R_{i,j})$ was always $\equiv A_i$. — A great compression!

— On the other hand, at first, TM will probly be constrained more reduction theoretic techniques, to the extent that it could not discover .02.

If 160.0F doesn't work, we can fall back on the method used in the IDSIA Report. We then have a seq. of O^j's recollecting the history of our solns for the TSQ.

.08 .09 0 to do O^j_{n+1} we get a pd. on possible O^j_{i+1}'s by using $O^j_i \to O^j_{i+1}$ as $Q_i A_i$'s in a TSQ. (This is "Learned Mutation").

Or use other ways to get the G. Grammar for O^j_{n+1}. + R.08 = 09 seems Much Better

using "ΔO^j" as corpus

4

speech
:157.34 For ③; Motor Rzt a D.f. on $Q_i \Rightarrow R_i$ is equivt. to a D.f. on $Q_i \Rightarrow A_i$, which is t. "final problem" we're trying to solve!

We start out by trying to find t. $Q_i \rightarrow A_i$ d.f. We can search over O^j (P.D.) $(A_i | Q_i)$, to find best Rt for Corpus. We can do a d.f. over $[O^j_i, O^j_{i+1}]$ as corpus, to get a P.d. over O^j_{n+1} trials from knowledge of O^i_n. A big problem is still having to check all Q_i, A_i corpus membership for each cand O^j_{n+1}.

How we can shorten Rt by listing Q_i, A_i's in order of Prob; so Cands will be discarded quicker. In past & Motor Rzt is only usable for BU Sch; but now it seems v.i. for Lsrch as well. So Start w. .03 - .04 to Lsrch O^i's.

Then use .05 to speed it up. (.05) (.03, .04 seems relevant) .06 can be (A) PPM, [ARUN, RAMN, SVM or any of various] induction methods Devised for Mach. Irng / Patt. Discovery, → 163.01

What is Prob? seems unrelated to .006!

.03 - .04 does seem very relevantly here.

T. advantage of using short codes of R_i of $Q_i \rightarrow A_i$, rather than $Q_i \& A_i$ themselves, is that t. short codes have much impt. info "Understanding" about Q_i &/or R_i.

→ Anyway, .02 - .10 does look like a viable method for Phase I.

T. way O^j works: we have a Loop or AZ machine, $U''(\cdots)$.

$U(O^j, Q_i, R_i)$ is run on R_i of t. Binary tape until we get $U(O^j, [Q_i, R_i]) = A_i$.

The R_i are self terminating w a "end" symbol. The PC or end symbol is adjusted to max PC of Corpus.

Furthermore .02 - .10 seems to be on firm Theoretical Ground — t. Proof of QA convergence does apply to it directly.

A perhaps critical point, is how good the $O^j_n \rightarrow O^j_{n+1}$ algm is in predn. If it is bad, then TM will be very slow, but still → ALP.

In t. recent discussion, short way to do Phase I: 157.29 - .30 ② or 157.31 ③

:157.22 → well, ② had job of finding a P.D. for $[R_i]$ corpus. for t. system to be ALP, Prob $[R_i]$ d.f. has to be a Universal QF. (or $[Q_i, R_i]$) — This finding of a PD on $\Sigma R_i]$ is an essential part of t. induction process — it doesn't merely "speed up" t. system. In ③ (157.31) T. corpus becomes $[Q_i, R_i]$ rather than just $[R_i]$. However t. corpus $\Sigma Q_i, R_i]$ is "easier" than t. corpus $[Q_i A_i]$, i.e. (11). Since R_i has some "understanding" of $Q_i \rightarrow A_i$. We may want to run t. Universal system (.02 - .10) on $[Q_i R_i]$ (instead of on $[Q_i, A_i]$).

(SN) I'd like to have a Set up in which I could try various combinations of Predn. Algms & Corpi, etc. easily, w. arby Corpi. e.g. "Lsrch" could be a

11:11
11:20

sketch. in which arby algts. could be Used.

So 162.02–.10 seems o.k. : The remark in .095–.10 is about the $O_2^j \to O_{2+1}^j$ P.d. to be approximated by (0) PPM a/o various Mach Lrng/Pattrecg algs.

This d.t. can speed up the finding of good O_{n+1}^j's — but the main induction is done by O^j .... which is a universal d.t. — so this o.k. to use arby, "imperfect" algms for $O_2^j \to O_{2+1}^j$.

Later, it may be possl to use TM_2 to do this induction ($\equiv TM_b$).

07 The $O_2^j \to O_{2+1}^j$ learning is "Learned Mutation"

162.02–.10 plus remarks of .01–.07 Do give a version of Phase I that is pretty much (the what I had when I was recently studying how various Mach Lrng/Pattrec. algms could be used in phase I.

The principal discoveries since:

1) That INV via Lsrch of Phase I / RATM Lsrch are very close — almost identical problems.

2) That the way I had planned to do adaptive Lsrch for INV problems, was somewhat wrong. It was necessary but the algm that derived/developed/updated the Guiding P.D. Was Universal ... not just simple (0) PPM Irng.

3) A perhaps clearer understanding of what the Corpus of "TM_2" was, (i.e. using $\sum [O_2^j, O_{2+1}^j]$ as corpus) — but still not entirely clear that this is the best Corpus. Perhaps the best corpus would be the entire "face" of TM's past activity + any "External Context" info. We could view the O_1^j, $O_2^{j'}$ sequences as a "time series" : . . regression problem.

SN! Consider the corpus $[Q_i, R_i]$. If we compress the Corpus $[R_i]$, I think this is relevant to Compressing! I.E. it seems like useful/usable info. Any statistical info on Part of $\sum [Q_i, R_i]$ is a constraint on the whole of $[Q_i, R_i]$ & is "useful" info.

In General, any regy obtained by any Algm is usable as a "code at least a" some can be regarded as a "recoding" (more coding).

If we could apply a UMC to $[Q_i, R]$ d.t. this would be fine, but I don't yet know how to do such a thing! There is the Coding theorem recoding way 153.03; .20ff

Also, one can do (1) codes via UMC.

Perhaps the reason for my interest in this is that $[Q_i R_i]$ can be "all processed" by APPM & probably other non-universal Algms. I'd like to be able to take advantage of this → 167.27

33

7/18/05

STM

164

Also see 166.08-.30
167.06-.25
for How to do BUSter & Lstrch in II

Rev .00 (latest version of Phase I)
.20 Reviewed
How, to start on TQ's of 160.01-06 is fringe

A reasonable way to do QATM, (in view of objection of 162.25-.27) → to t method reviewed in 160.00—161.10

Given a $[Q_i, A_i]$ corpus: We start out by setting $O^j \equiv \Lambda$ & finding $R_i \ni$ $U(Q_i, R_i) = A_i$ for several early i values. We get a P.D. on R_i is $[R_i]$ (via PA) PPM a/o other (M.L./Pattern/Compression) methods. Using ≥ not large part of beginning corpus we try to find O^j's (= a prip of O^j) $\ni U(O^j, Q_i, R_i) = A_i$ for Reduction of boost of QA corpus. ($\max a_j \prod_{i=1}^{n} O^j(A_i|Q_i)$)

To speed up t. search for O^j's, for t. R_i part of such we use the P.D. (.02)

As O^j changes the D.F. on R changes, but we may not have to update Rule D.F. very frequently & w. changes in O^j!

We then use increasingly long QA corpi à do .01—.05 on Rem — & solns O_i^j are for $\sum_{j=1}^{i} [Q_j, A_j]$ Later on after we get out O_i^j, O_{i+1}^j pairs, we use this several pairs as a corpus to get a P.D. on O_{n+1}^j as a function O_n^j. To get this D.F. we use t. Algms of .02. This P.D. is then used to Guide Lsrch to find O_{n+1}^j (known img O_n^j) } Also Note 167.19-.25

(3) As before, we use a R_i d.f. (periodically updated) for t. R_i part of t. Lsrch

(5) Also, to speed up discards of O^j cands, we list the $Q_i A_i$'s in order of "dirty ness for" j & try O^j cands out hardest, first.

So, (11) & (13) are speedups via Algms of .02

(14-15) is a distnt kind of speedup but probably very impt.

However, all of these (3) speedups do not give the 162.25-.27 trouble. — They are speedups for O^j such, which is a T—Universal Predictor. (predicts to)

So: .00 — .15 does look like an adequate QATM model.

A possl "dirty": There is the "Recognition function" device used in early part of IDSIA Report: Seems like a v.g. idea — ~ to way humans do induction.

Q: Is it Tuniversal? I suspect it is, because in an extreme case, Rem is only one "R" functn & this becomes identical/to simply optmzing O^j w. Gorn (.04) "R"?

So: it may be worthwhile to try to get that "R" method to work.

With 160.00-.06 To Do TSQ's & 164.00-.15 to do QA, it looks like we can start on Phase I.

As an Alternative to 164.00—.15 to QA; Consider 160.07—.21: while it is not Universal, it is fast, & can perhaps do rather well in Certain Limited Domains. Also, it may be possl. to somehow MIX rem. 164.00-15 & get t. Best Features of Both Methods!

7/18/05

STM

Field Prog. Gate Arrays ? ← Google has info about them. Maybe buyable on e-Bay
FPGA ?

(EN) A very fast Cellular Automaton for rapid sort in ALP:

Say we have a Universal 1 dim Cellular Automata (= CA).

On the first row we have the input cartesian.

on the N^{th} row, we have a seq. of ~~ksms~~/switches that generates the N^{th} row from the $N-1^{st}$ row.

So we have the system state propagating (very rapidly) down the "N" direction.

At N = 1000 we have the "output":

we also have a delay line so that when the output occurs, we also have as output the Y_i, delay line, the input that gave that output. If output is A_i, and we save the input, which is R_i. (and Q_i also).

T. prog. could implement the CA™ but w.o. the speedup devices of 164.11-.15 we'd only do 164.14-.15 (limit of Q_i's in order of delay).

We could have/ external Devices that that lookat O^3's & R_i's of the past and use this info to order input trials for the array of .00 ALP.

Essentially, .00-.09 implements "T. inner loop" & we want it to be very fast. Its inputs are arrayed by the 3 heuristics of 164.11-.15 & can be done more slowly by a regular Computer.

Well, it has to be the fastest crunching cards. — Fast enough for .00-.09 which is perhaps 1 trial/nanosecond — or even 3 trials/nanosecond! PPM generates one character per "(loop)" or whatever. — But still much slower than 1 bit per nanosecond — and .00-.09 could have inputs of 10 or 20 bits at least!

So PPM may be the bottleneck / we can get the CA of .00-.09 to be much slower if we use slow components (cheaper).

T. Q is: Do the speedups help enough to justify their cost?

.00-.09 could be made quite large & at 3 trials/ns, we could really do a lot of trials!

Still, w.o. the speedups, the cc of a soln of a problem is exponential in the problem size (for fixed N max) so not very practical!! Tho, at 1 ns/trial, we could do $10^9 \times 10^7 = 10^{16}$ trials/yr. $\approx 2^{53}$: 53 bits for a year

For some of the speedup: use analog computing — PE's need not be exact.

4

7/18/05
5 TM

Rev .08 - .30 } How to Do BU srch
+ 167.06 - .25 } usually || w. Lsrch.

How to w. Th. ↑ Sz So
↑ = ↑ ↑ .

SN1. On usenet easily available PPM pgms?

For ≥ 2 speedups of 164.11 & 13 all we need is candidate pc order: Using symbols/tokens as units (rather than Bits) would be OK. (int. curves)
T. resolu. given into pgms is in Bytes: If we list pos'l. contents of a cand. in "order" w. a resolu. of only Bytes, they will give us a good enuf ordering of cands for this purpose. T. ordering of any cand will never be "way off" which is really good enuf for the Lsrch I'm using rt now!

Even so, .00 th is very slow since caches to read & evaluate Copius/Branch character counts!

.08 SN2 The (Apparently Adequate) Model for Ph1 on 164.60 - .15 is for Lsrch only. (phase 1)
So what about if BU srch (or will we need it before we go to Phase 2?)

166.08 - .30
167.06 - .25
Seems to be adequate for BU srch
Th. Lsrch, BU srch Mix

Viz Lsrch, we say given a v.g. TSQ, but certainly there will be / exist universe of problems that it will be unable to solve, since CJS will be fatal by for the lowest CJS solns. It may well be Part of indication (at initialization) probs of Ph 2 and mainly of this type & mainly need BUsrch.
Whether BU srch is part of Ph 1 is.

If (in Lsrch phase) CJS's are large, then it is (clearly) that we will not be able to know (in Ph 1) whether a problem is solvable by Lsrch or not. Anyway, at a certain pt. in Ph 1, I will start to do Lsrch & BU srch in (1 - Lsrch from Bottom (shortest pgm); BU from top (longest workable pgm).

20 At any (late) time Phase, TM has (at least) 2 cands for Oj: ■ one for Th. other is a BU■ srch Cand. Lsrch ■■■■ in which it does trials of Mutations of (last) satisfactory Oj".
Well, say I've been successfully using Lsrch up to now & I'm finding no soln w. lots of cc: I can either modify TSQ ■■■■■ so as to reduce CJS of th. problems — or try BU srch. (boost?)
At present, Th. last Oj gives excessive (B2) for (last) problem, & mutations of that Oj don't seem to help much (for entire creative Corpus).
I could "Backtrack" by trying mutations on an earlier Oj. (using Mutation P.D.)
Anyway, BU srch would also do mutations, but not in pc order: It would do Monte Carlo srch.
We may also backtrack BU srch & do BU srch ■■■■■ by mutating a earlier version of Oj. So my impression is that Lsrch is a maximally Local — 167.06

Def: A Singular P.D assigns zero, pc to at least 1 elements. Bernoulli seq. ■■■■■ is not-Sing.
ALP is not Sing: PPM is not sing. An important property of n-sing seqs is that they can be input to a UMC & output will be univ. d.f. — Actually, we

Rem: .06 — .25 (on BUsrch).

TM 05, 330–339, 379–390 } Formats of Titles of Scanned TM Files

30 — any seq. w. positive entropy will be O(n·s) input. (i.e. $-\frac{\log p(n)}{n} > 0$)

— I'm not quite sure about what I want from a non-sing D.F.

Tho, "in some circumstances" if a D.F. assigns p=0 to a seq, we can't use that D.F. to get p's on continuns of the seq. This may be what I was thinking about.

1) What is extent of STM's "Gensel sent"?

2) How to characterize what they do as functions.

3) Do they do enough to simulate all/most functions (= steps "")?

36: 166.30 Srch, i.e. an exhaustive search: BU srch uses the same P.D. that Guides Lsrch, but in a Mt. Carlo fashion — so it's a much broader srch. If there are no solns. "near zero" (which is what Lsrch looks for) Lsrch will never find them. We must Note that in present case, both Lsrch & BU srch are for a OZ Problem, so each can have a "Best so far" soln. That may be very Bad but not infinitely bad (An "Infinitely bad" soln. would assign p=0 to corpus).

12 — Is There a way to design a lang. for BU srch so that $\frac{\alpha}{Max - M}$ formula can be used? — Probably .... Particularly if we are using Phil Gaugn. of G.A. for the BUsrch — If we are Gonna G.A. then (20–23) is appropriate.

The way (12) is done: We have various cards/tasks for many cands, the entire corpus will have a Gave for the entire corpus. We select a Gav threshold to include a cand in the corpus so Met $\frac{\alpha}{Max - M}$ is max; Then we make our Grammar from that corpus.

The stochastic Grammar can be S-C SFG or SVM or ANN or RANN or PPM

9 Ref: 164.11–.12 — For BU srch we can either use $[O^j_i]_{i=1}^{n}$ as corpus & have p.d. on O^j_{n+1} directly or use $[O^j_i, O^j_{i+1}]_{i=1}^{n-1}$ as corpus & get D.F. on $O^j_n \to O^j_{n+1}$ & so get "language operator" i.e. this corpus gives (log) $P(O^j_{i+1} | O^j_{i-1})$ so we get $P(O^j_{n+1})$ by using $P(O^j_{n+1} | O^j_n)$. I think 2nd form is better, if we can do it, but the first form may be easier to implement. If we use a PPM, for either (1) (19) or (2) (20) we can use, for induction, any M.L. or Patt rec. fgm.

23 — ~~3/0~~ Use TM_1 as TM_2 (w. proper TSQ)

27: 163.33 SN 162.26–.27 & 157.15–.22: Given corpus $[Q_i, A_i]$, when we find R_i's $\ni U(Q_i, R_i) = A_i$, we have found regrys int. individual Q_i, A_i's. Like representing of (sub-trees) phrases: After work done there, we could apply it to compress $[Q_i, R_i]$ — Since we've already found some regrys int. $Q_i \to A_i$'s, this should be easier than compressing $[Q_i, A_i]$ directly.

30

31 — A Good Method for Phil! First get $[Q_i, R_i]$ for $U(Q_i, R_i) \to A_i$

32 — Do this for a fair sized corpus. Then, using the statistics of $[Q_i, R_i]$ say (A) PPM, search for O^j codes $\ni \prod_j O^j(A_i | Q_i)$ is max.

i.e. find S' a subset of R_i' s.t. $U(S', Q_i, R_i') = A_i$ & $(\Sigma) \leq |R_i'| = \min$.

Using the P.F. from 167.31-32 on R_i to search for R_i''s (for each trial S')

I Rank the heavy robots for universality after scan.

A new S' search has to be done for each augmentation of $\{Q_i A_i\}$ corpus.

As soon as we have enough S''s, we use the D.P. $[S']$ or $[S_i', S_{i+1}']_{i=1}^{n-1}$ as a corpus to guide search for the next S'.

After we have a number of $Q_i A_i$ in the corpus, we list the $Q_i A_i$ in order of difficulty of search (.01) for S''s — we quickly discards on most difficult $Q_i A_i$'s.

However, the criterion for discarding a S' is not clear — also it is irrevocable — so if we find we aren't doing any better than our present threshold on the solution of the "hardest" $Q_i A_i$", we may backtrack no a layer (layer length → smaller pc) to a previous threshold

Note: In trying S' in $U(S', Q_i R_i') = A_i$. We already have a R_i ($\Rightarrow$ $U(\Lambda, Q_i R_i) = A_i$), so $U(S', Q_i, R_i')$ can always do as well as R_i did, by ignoring S'. However, because of the extra argument in .12, the bits of R_i' are always a little more expensive than those of (.13L), but that doesn't effect the general discussion much. The result is that we will always (.21)

SW Note: I say (above) that one should wait for a fair sized corpus before Applying my GAPPM! Actually not necy. Regular PPM starts w. the first symbol in the corpus & is able to get pc's of increasing precision as it goes along ... the only lower limit on string length is $\Phi(1)$.

(.16) However a not-to-bad bound on what we can expect from a new S' in the way of compression (i.e. "no infinitely bad S''s can exist!)

Actually, this is a bit disturbing because it means that we will never be able to quickly reject any S' candidates! This really seems to change the search quality Tremendously from my earlier view of it! → However see (.28)

The search could be done by testing a S' and w. just a few Q_i's because we do have an idea as to what's good.

(.25) Well, perhaps .12 ff is not so disturbing! Indeed, while the $U(\Sigma Q_i, R_i) = U(\Lambda, Q_i, R_i)$ solns will not have $|R_i| = \infty$, they will usually be rather bad solns, since they have learned nothing from the past.

When we are trying to optimize a S' based on the $Q_i A_i$'s — we have a Vector Gene like in GPS, & we should devise a GA-type algebra to deal w. it! Marvin said the GPS idea was Cod vector Gene) was explored, analysed much by Newell, Simon in their days of GPS.

72209
STM

There are 3 major speedup devices in the System Plus for: 164.11, 164.13, 164.14. This last (working on it) trying a crude on most difficult Q's, will be modified by using instead the "Vector Gore" idea of GPS (165.35); This has to be developed.

{We have to be sure that the heuristic/Ordering of trials does not destroy the universality of the strategy O^j's.

At present, this last, is the least completely understood of the 3 Speedups

Def. SU3 More on Speedups = SU3 : Some ways to do it: Try various S''s on complete corpus: Compare "Gore Vectors" (168.32) of various S''s: If some work well on some problems, poor on others, but they complement each other, then they will be v.g. in all. However, note that the wt. of each S' is on its total pc for corpus x its a priori (≡ 2^{-i}).

However, the 11 codes of .08 are also V.G. for the Diversity Theorem, so even w.r.t. all Qi Ai corpus, one can have lots of useful codes ... which will enable good extrapoln. of S''s → .22

Woops! Note that pc's have been greatly overall unnormed! — So Comparisons of Code lengths is not so obvious! However, if all code lengths (i.e. |Ri|) are not very far from default code (for S' = Λ) then perhaps the normalizing constants for distinct S''s will be close enuf so we can assume cancelation — so they are "unimportant" → there is some good discussion of Normzn (somewhere) → 162.17)

.21

22 : .15 → Sometime ago I had the idea (perhaps for BU search) of Geffrey O^j's that were good for some pairs of QA, then triples, quads ... etc. — So I had O^j's for all nos. of the Q's in the Corpus. The more Q's they did well, the more wt. they got. I then somehow used concs in Rosa O^j to downsize better trials.

25 — However in .08 - .15 we do it that but the wt. of a O^j is simply a·the pc. it gives corpus x its a priori. Although (long ago) time, I was afraid that often various O^j's would give pc = 0 to certain Q's — and thus I couldn't use Bayes as .26

Note: These speedups (i.e. in particular, note SP3) don't have to be Universal, so any heuristic or M.L. or Pattern discovery trick is Acceptable.

30 The "Vector Gore" of 168.32 (GPS) seems like a v.g. idea: The idea of localizing the "Badness" of a Gore. Possible Use of an ob-of algebra to get a Gore.

Rev. 00

.4 day/page. 400 = 2 yrs
400 M = 1 kpp = 2 yrs at 500 pp/yr. 4G = 10 kpp = 20 yrs!

So: What is present State of TM project?

Phase I is in not bad Situation:

General Form of Gore for QATM is as an Early pp of IDS/A report

What needs to be done is write TSQ's Devise Laws / Primitives of Lang. chose,

[~Lisp v.s. ~ AZ-141 v.s. ~ Forth??] → see 183.00

We write much of TSQ first, then write many (by hand) solns. of many problems

.05 in TSQ. This Gives a Viz "APPM" a rough P.D. on instructions in the lang.

.06

.07 We look at r. CJS's of r. hand pgmd solns. Are they within acceptable Bound?

N.B. → [If not, try to put in "Bridging" problems. ← ??? → Bis probably wrong!

[At this point, S' = Λ, so the system isn't learning much any thing wrt. inter problem info.

So r. Q of .07 is not reached: r. problems are to get a good set of w/s [for r. primitives. 2 Contexts

From that, we can then Begin to search for O^j's so I/m r. 'initial' TSQ., or perhaps an excursion ... r. initial TSQ of .05 → .06 was not a real TSQ — it was to get w/s & contexts of primitives only. We really

.14 need a specially designed TSQ for Teaching TM serious things.

Perhaps it would be best to do use r. TSQ of .14 to get r. pc's of r. primitives,

within Reader/ assoc. Contexts. Then use do I/m via trivial O^j's (which are originally Λ)

If certain problems have excessive CJS: we should study r. S-Learning

try to understand, propose intermediate problems. Ebreak big problem into more easily learnable

sub-concepts

Note: r. Speedup of 169.08 — so does not give sequential

O^j's so we can't extrapolate O^j viz r. [O^j_r, O^j_{r+1}]^n_{j=1} corpus. On r. other hand, 177.28 for how to treat it

it makes r. devising of TSQ's easier. We just Sic. TM onto r. "problem Pool".

Hvr, it then has to evolv/I/m how to tell which problems it's "currently smartest of"

to solve — a diff'lt thing to I/m. — Tho it may be that r. "vector Gore" of 169.32

.25 is in that direction. → 177.28

7/22/05

SVM

≈ SVM . ~~coff~~ a bit easy, maybe not so good, why ← unclear!

≈ 88.01

SVM's : A useful way to consider to many dim x-plane/separation.

Say we start w. a r dim space & we want to get to a h dim space using polys of upto degree d.

So, the r original variables : take all products of powers of variables so that ≤ powers is ≤ d.

say there are (m) such (terms/products) (including the constant = zero power).

Any linear comb. of such terms is a (Nonlinear) x-plane of the original r dim space.

So it is a vector in (m) space

Hwr, SVM is not much (concerned w. that (?)) : it just ends up w. a set of pts in the hy dim space, & we want hyperplanes to separate the 2 categories of pts.

One rough & dirty trial : Get Cent. Gravity of Good pts $\vec{C}_G$ & Cent Grav. of Bad pts $\vec{C}_B$. Then try ⊥ bisector of segment $\vec{C}_G \cdots \vec{C}_B$

If the G pts are small in no. & compared to B (This would be the case if we had to categorize say 40 different possl. "Next Symbols"), Then consider $\vec{C}_G$ & the distribn of G pts about it. . . .

Actually, even if the B pts cardinality is >> that of G pts, $\vec{C}_B$ is still of interest & the line from $\vec{C}_G$ to $\vec{C}_B$ is of interest.

A hyperplane ⊥ that line can be moved betw. $\vec{C}_B$ & $\vec{C}_G$ to find optimum separation.

This is certainly a fast way to get an approx separation : which may be adequate for my purposes!

SO : We just expand to random Hy DIM space : Then get Centroids of "Yes" & "No" pts. Get hyperplane ⊥ the line betw. 2 centroids, that gives "best" partition (Criterion is Unclear!) Try "Min. no. pts. misclassified"

I'd like Something like Min cost of coding & misclassified items.

If p is true pc of 1, & (1-p) is pc of 0 :(((If there are n pts & k are misclassified, we need a certain amt. of info to specify those pts. One way : $\left(n \cdot (n-1) \cdots (n-k+1)\right)^{-1} = \text{pc} = \frac{(n-k)!}{n!}$

Hwr, we may want to be able to use info about classifn (wrong!).

So if A are classified 0 & A_1 are really 0 } we need $p = \frac{(A-A_1)!}{A!} \cdot \frac{(B-B_1)!}{B!}$ which may be >>
B " " " 1 & B_1 are really 1 }

Try a case : $A = B = 10$; $n = 20$; $B_1 = A_1 = 3$

So $\left(\frac{1}{10 \cdot 9 \cdot 8}\right)^2$ v.s. $\frac{1}{20 \cdot 19 \cdot 18 \cdot 17 \cdot 16 \cdot 15}$ so is larger by ≈ $2^6 = 64$.

$\approx \left(\frac{1}{9}\right)^6$ $\approx \left(\frac{1}{17\frac{1}{2}}\right)^6$

SVM's

Anyway, once we get the partitioning H-plane, we see how far each of the pts is from it. Then we make a squashing function (like tanh) that maps all distances (from $-\infty$ to $+\infty$) onto $[0,1]$.

There is an adjustable parameter and we arrange so that pc of corpus is max.

$\frac{1}{2}\left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right) + \frac{1}{2}$ — so $\frac{1}{2}(\tanh(x\alpha)) + \frac{1}{2}$

We adjust α for max corpus pc.

I think we can omit factor $\frac{1}{2}$ so just Max $\prod_i (\tanh(\alpha K_i) + 1)$ ←

We use $(\frac{1}{2})$ those values for the pc of prediction.

We have the line joining the 2 centeroids. We have all data pts distributed along that line, & we have to find an optimum partitioning pt.

We may want to try several random projections into hyperspace, & see which one gives best partition — best pc of corpus.

How many bits do we need to specify the H plane? Ideally, we would use all possible H Planes & add codes in ||.

Say we get the 2 centeroids, then we need to specify the pt. on the line between them at which to put the $\perp$ hyperplane:

We consider a region & get a "pc of corpus" as a func of position in the region:  The width of the D.F. gives us the precision we need for that parameter.

But we still don't know the mapping into other coords of the H plane. (Or the line between the centeroids).

If we have k params that describe the H. plane, then the pc of corpus will be a say Gaussian D.F. of a Riemannian 2nd (a Multidim Gauss). This could involve $\sim \frac{k^2}{2}$ coeffs of the hyperellipsoid — sounds Difficult!

It may be that we can find Empirical Rules to tell if we have "Overfitted" or not!

Anyway, it's (.06) that gives the Gauss at any particular H plane choice.

If we want the H plane $\perp$ to the line between centeroids — it's a 1 dimensional problem! We have a distribution of pts from $-\infty$ to $+\infty$ so we have to choose a H plane (1 param) i.e. a α value (1 param) — so it's 2 dim. optimization — ∴ not very difficult!

"On a Hyperdimensional N.S. plane" Is the no. of dims comparable w. the SSZ? This does (maybe) make for trivial (Alt) separation.

If each data pt is a basis vector in a N dim space ... ($\leq$ SSZ $\leq$ N)

Is there always a "separating H plane" for any # Dimensions of the corpus? → 17.4.20

MDL v.s. ALP & How to use Huffman Codes Efficiently: ooff
MDL ≤ pc of corpus is within constant factor of ALP. Butmotto .31
I think this is a Dave Wills Result. objection

Marcus Hutter says he showed that MDL is significantly worse than ALP.

I think this is false. A MDL code will be at worst a constant no. of bits worse than ALP.

To show this: Say $M(X)$ is the "true" generator of the corpus: it assigns pc, M to all strings X. M has a finite description length "$|M|$".

Using M, and Arith coding, code any corpus, Z the equivalent pc of the code will be $< Z^{-|M|} M(Z) \cdot \frac{1}{Z}$

To use Huffman codes, we need to know M. If we knew the no. of bits in the corpus Z, (say N) we could consider the pc's (wrt M) of all seqs of length N & assign Huffman codes to them. This would give total code length of X $-\log_2 M(Z) + N + \log_2 N$. (N = no bits in Huffman symbol) In terms of pc per symbol the $\log_2 N$ term is the factor $\frac{1}{N}$: We need to get rid of it!

Say we don't tell what N is, but we assume an $\tilde{N} >$ the actual code length — as $\tilde{N} \to \infty$ do we end up with same code for Z.? → 174.00

.12 - .13 may be adequate, but perhaps we can consider "$N = \infty$ arithmetic". There may be a way to get the Huff code for an arb. sequence with $M(\cdot)$. This may be what is usually done in PPM, since Arith coding has assoc IBM patents. →

If this last works, then Huffman Coding is about as good as Arith coding .... The main CP is relative cc's involved.

Also if the Huffman works as described, it should be possible to code a corpus sequentially using it. — i.e. if α is the code for X then the code for $X^\wedge z$ must be of form $\alpha^\wedge w$ } This may not work since Huff is a prefix code. But I'm not sure that fact is relevant!

There may be many Algms for getting Huffman Codes.

In BZ2 & other commonly used Compression Codes, it is said that Huffman is often used because of IBM patents on Arith coding (p. 16) so I may want to look into just what they do!

One thing they might do: Code corpus in blocks of k bytes. } each Blocking Code w. one symbol

Maybe $k = 10$. Each Block has its own Huffman code. (= distance P.D. from other blocks) The size of the gains "round off error" in compression but may be larger: The smallest block can be $< k$ bytes long — so only $\frac{1}{k}$ byte round off error.

On second thought: .00 doesn't consider that MDL / MML is a "2 part code" ... Is this relevant? Well (.03-.06) is a "2 part code" → 174.00

- Part 1: description of $M(\cdot)$ || Part 2 the Arith or Huffman code.

5TM

(173.17 2 pages)

173.32 → [illegible] on [illegible] corpus length = ∞. Perhaps 173:13—.13 is the best approach

(173.13 spec.)

T. way to do H-coding: Say we have a bunch of prob. values $p_1 \cdots p_{80}$; in order of size. (largest first) How to code them so each p_i is coded with a length $< (-\log p_i) + 1$. If we can code them in order of size, then we can indeed

No, this is wrong! The p's of all strings of a certain length must ≤ 1, but strings of distinct length don't have to have their p's ≤ 1 all (!)

It may be best if it's common to use say $k = 20$; and whenever codes a text, one pads it out so it first set has 20 chars. This is OK for compression, but not so hot for "Recog". To compare texts, one could truncate them all to the same length (procrustus!). This has some advantage: If one wants to compare a unidentified story to find its author, the comparison texts of all of the authors should be same length — since coding efficiency $\propto$ length.

If we have P.D. over self Delimiting strings, we get a "Chaitin" Univ. D.F. ($C \equiv$ Discrete Univ. D.F.). These strings have $\sum p_i \leq 1$ so we can make a Huffman Code for them. So for the Discrete U.F. shortest code [illegible] Give PC within constant factor of Σ over all codes.

However, according to Gacs: for the Continuous U.F., the difference $\to \infty$, but the slower $\uparrow$ than any recursive fn. $\to \infty$

159 Theorem We can always tell how long a code can be by using $\text{Ceiling}(N)$ bits → 178.100

20: 172.34 — Well, the [illegible] unit vectors are $\vec{\alpha}_i$ $(i = 1 \cdots k)$, A k-plane is

$\sum x_i \vec{\alpha}_i = \beta$. $= 0$ is the $\vec{\alpha}$'s; β defines the k-plane. If $z < \phi$ we are on the origin side. If $z > \phi$ we are on the $\vec{x}$ side.

Consider the k-plane $\vec{x}$, β [illegible] $x_i = 1 - \epsilon$ for $i \leq k$, $x_i = 0$ for $i > k$.

$\beta \equiv k$ [illegible] the point $x_i = 1$, $i \leq k$, $x_i = 0$ for $i > k$ gives $z =$

$z = \vec{x} \cdot \vec{\alpha} - \frac{1}{2}k$: $z = 0$ is the k-plane. $\beta = k$

$z = \vec{x} \cdot \vec{\alpha} - \frac{1}{2}k$. for $z = 0$ pts are on plane $z < \phi$ otherwise. $z > \phi$ on side

Consider the basis vectors: $\vec{x} \cdot \vec{\alpha} = k$ so $z = k - \frac{1}{2}k = \frac{1}{2}k$.

Consider No: $z \equiv \vec{x} \cdot \vec{\alpha} - \frac{1}{2} \alpha_i = 1$ for $i \leq k$; $= 0$ for $i > k$.

for $\vec{x} =$ any basis vector $x_1 \cdots x_k$, $z = 1 - \frac{1}{2} = \frac{1}{2}$ (i.e. $\vec{x} =$ say $(0, 0, 1, 0, 0 \cdots)$, 3, k)

for $\vec{x} =$ any basis vector $x_{k+1} \cdots x_n$, $z = -\frac{1}{2}$

So for any partition of basis vectors, there is a k-plane that separates them.

W. Margin [illegible] $\pm \frac{1}{2}$.

32 If the basis vectors are not ortho, we can xfm them to orth. basis, put our

33 [illegible] k-plane & xfm. back — so separation is done.

.32 — .33 is possible if the basis vectors are not normal, but are linearly indep.

5 JAN

500 x 50 15k.

174.39 So in N space, / any set of N linearly indep pts. ~~in~~, any partition can be done w. a suitable H-plane.

Note: In any N space, one can't have > N linearly indep vectors!

The H plane in N space has just N+1 params. — so if one starts w. N, k dim. data pts, & k > 1, RR is not seriously needy "overfit".

While Vapnik uses his Support vector criterion — (which is discontinuous function) I ~~[illegible]~~ can use a continuous Gore of 172.00 - .30.

In ~172.00 -.30, I get a ≈ separating H-plane by drawing line betw. C.G. of Good v.s. Bad pts., & find best ⊥ plane to that line.

This is a simple 2 param optim. It makes (modified) SVM method a lot simpler to implement.

T. main problem is computing t. pdf of t. model. This must be estimated by looking at t. "Whole Good Gore" as we do random perturbations in t. params.

One trouble is that there are N+1 (or 1<6) params, & N is very large.

We must do random sampling (i.e. not all N) for N of t. params & more careful sampling of t. other 2 ..... One possl. diffy: Th. H plane will not be at a peak pt. So calculating its "width" may be ~~[illegible]~~ diffl. ! T. effective width may be t. second derivative at that pt (or Hessian). The linear deviations from t. control pt. cancel out. Only t. second diviations have a peak. So "width" depends on local Hessian, which is at a peak or not!

So may be get t. Hessian by some kind of optimator Random Sampling.

If one knowns Hessian matrix of 2 points one could work it to get a good peak estimator! But merely a 500 x 500 Matrix is expensive!

→ Another possy! T. Gore seems Related to Part of FRANN's! in which t. derivatives of t. Gore wrt t. hyperplane coords are easily computed. → (31)

(SN) On t. $\frac{\sigma}{M_x - M}$ formula: Originally we used this to choose a threshold for G.P. computer s-(any) model. We could also use an x window instead & choose λ w. some criterion: T. real Q is: would new gate be better $\frac{\sigma}{M_x - M}$? It might be handier to update that. Using a simple acceptance Threshold.

(24) → If so, one could easily compute t. many derivatives & second derivatives. Perhaps t. second diriv's (all ± x500 y500 at them) of t. Gore or Log Gore are an additive part of each of t. data pts. If t. off diag terms are small enough t. Hessian is diag. & t. matrix trivially invertible, & / or get its Determinant.

STM

Discussn w. G. Sussman: Said Lisp (compiled) is very fast; Also has library of useful functions including B trees. So it may be feasible to write TM in it now! (Also. ForR may also be v. fast) Macsyma, Maple, Cayo.

183.00

So: Write up Summary of TM project; First actually do it.

May aj $\prod_i O^j(A_i, Q_i)$ in Lisp, w.o. speedup algms.

Use very simple TSQ's & see how fast one can get.

[TSQ's from Elementary Alg; Mathematica & MapG operators - eventually Carr]

Then put in 1 or 2 speedup algms: using simple PPM at first, then automated. PPM & perhaps other "well known" proofn algms from Mach Lrng Community.

(Next) Dcrb. Various kinds of Induction problems. Show how QA is very General (perhaps do Pg as F.N. or appendix).

(Next) dcrb. INV, OZ problems. Dcrb. great variety of OZ problem types: Then talk about Phase 2, its Gore, & integration of PSM's etc.

An impt. thing in both phase 1 & Phase 2 is to have clear idea as to what kinds of problems are (appropriate/unfavorable) at each point. Eg In phase 1, I will probly get TM to do some equation solvg. Normally, this would seem to involve "heuristic search" which uses induction, but doesn't seem to be mainly an induction problem. If I view it as an INV problem, it is certainly "formally" of the form of an induction problem. Also INV problems are usually solvd as OZ probs or via the "Vector Gore" of GPS. → 177.00

Q: Would it be a good idea to solve Phase I induction problems via "GPS"? — i.e. Use of the "Vector Gore" ....? Is it conceivable that TM would itself discover this Technique as some thing to be used for practically all QA problems?

.20 Hier unclear: Needs More thot!

Re: .20: My impression is that Phase 1 should not be used to solve INV problems. If it is done w. the present schema, it could solve them & try to solve them better. — But could it learn .21-.23? Maybe it could w. suitable TSQ.

A related Q: Could we use GPS idea for ordinary QA induction? — i.e. Say an R_i produces something (A_i') close to A_j — A_i' differs from A_j in 4 ways. How can we modify R_i' to fix these 4 diffs? — So R_i is "like" a vector Gore.

Or: we have Gore: $\sum_i |s_i| R_i$ (177.11) Then $\{R_i\}$ can be viewed as components of a "Vector Gore".

Re: GPS Marvin said Newell/Simon GPS Paper was Good Exposition on how it worked.

1.29.05

5 TM

50 — 179.19: More on TSQ's: e.g. Could I teach TM what "Optimum" means?
T. idea of "t. best Prob can be done in 3 minutes" seems like a difft Ring
.02 to derb! I.e. How could TM derb it in its "Internal Lang."? → 195.00
For t. Algebra TSQ: I can write it 2 ways:
1) As a seq of problems w. large CJS both. Prom,
& ideas of what t. new Concepts are for t. many bers of t. sequence.
2) Same as 1) but I write going Prob solves each of t. Big CJS probems
I think I should write a longish TSQ, so I know what kinds
04 of concs I will need in t. more distant. future. → 179.00

10 SN | In finding $S' \Rightarrow \dot{U}(S', Q_i, R_i) = A_i$,
11 $|S'| + \Sigma(R_i) = \min$. I would have difficulty finding good S''s if t. corpus was of much size at all; (No speed-up-tricks this far).
T. way to get a set of S' (corpus) for reasonable trials was to find S''s for various subsets of t. $\{Q_i, A_i\}_{i=1}^{n}$ corpus.
Most subsets would contain <<n examples: but wts of t. S's would depend on ① $|S'|$ & how many and which $Q_i R_i$ pairs were worked by it.
I had some trouble assigning wts. (last time I did this, but t. discussion of (168.12ff, 169.22ff) clarifies this: It's the idea that
Every S' will have assigned a pc to t. entire corpus. (Usually it would not
30 .20 be much worse than t. pc assigned by $S' = \Lambda$). So we can get a fair
35 .21 sized corpus for S' examples this way using all to get their wts.
So w. $S' = \Lambda$ we get an "unconditional" pc for $Q_i \to A_i$
(a kind of "weak" unconditional, because clearly $P(A_i)$ depends on Q_i) — but
unconditional in t. sense that $P(A_i, Q_i)$ is indep of $P(A_j, Q_j)$ if $i \neq j$.
From this "corpus" of S''s & their wts we can create a P.D. to use in
more intelligent trials. This P.D. would not use into in t. ordering of
.27 cases in t. TSQ:
.29 :170.25 To be able to make use of ordering: For $j = 1$ to N apply .10 - .31 to
t. successive corpi $[Q_i, A_i]_{i=1}^{j}$. Each j will have a set of S' solns
30 w. assoc. wts. When N is large enough we can then get an adequate
1(corpus of $[S'_j, S'_{j+1}]$ pairs! Each S'_j will be a wtd set of solns. cf .10 - .31

7/30/05
5TM

GACS' Theorem on ALP v.s. MDL

Atomic Clock Blinks at 5 am to 510 am.

174.19 To tell [redacted] how long the corpus is (if it is not "self-delimiting")
$k(n) < \log_2^* n$. — but perhaps not much "<". (integer)

How Gacs Got this function that → ∞ slower than any rec. funct. is a big Mystery to Me.
Gacs says the proof is difficult .... So perhaps checked by few (if any!).

AH! I may have a proof! [crossed out] The additional a "no. of bits that ↑ slower than any recursive funct" can be something like the inverse of the Ak function, in which only a few bits can represent an enormous number ... but still to represent [redacted] arbitrarily large no. one still needs an arbitrary no. of bits.

So, we start out w. a few bits that represents a no. >> the corpus length.

How can we Use This Number plus, say Huffman (or Arith) coding

Woops! If we use Arith coding, Then the final code length is "almost always" < −ln pc(corpus) + 1. So it would Seem, that Gacs is Wrong!, No extra bits are needed! Perhaps the "almost always" Condition is important?

(Tho Gac's Thm may involve $\mu \ne$ CPM)

Another possy is that Arith coding always uses bounded precision pc values, and for precision of d bits, we get ↑ in code length for each symbol of some funct of d. So if $f(d) = 10^{-10}$ bits/symbol, for a corpus ~~length that's long enuf, we have~~ of length n we have extra $n \cdot 10^{-10}$ bits — which ≈ ∝ n. Hw, if we had precision ↑ w. n in a suitable way, we might arrange so that total error was bounded!

↑ This would be true if the "roundoff errors" did not "propagate".

Another possy, hw, is that Arith Coding is usually "exact" in the sense that whether the next bit is 0 or 1 depends on whether 1 pc is > or < another — & this can almost always be determined in a finite time ..... in which case the only "error" in Arith coding would be due to the occasional (but rare) "almost always" situation of (.10)

→ The Definition of "Computable probability measure" assumes this

Perhaps GAC's theorem was relative to $2^{-K(corpus)}$ to ALP(corpus) ← for non-self delimiting.

Hw, if we have a CPM then we can get ALP within const factor ≤ of μ using Arith coding from cpc we can get within const factor of μ.

If μ is not a cpm, Then we have to assign CB's to approximate it & convert it to a CPM. This CB may have to → ∞ as n → ∞? So, to code a long corpus (length n) we have to specify a CB that is a ↑ funct of n → so as n → ∞, the no. of bits needed to specify the CB → ∞. Hw, my impression is that as corpus length ↑, the CB needed ↑ more rapidly than any recursive funct. (rather than more slowly than any recursive funct).

→ 182.00

7/30/05

BOOSTING: Application to QATM: .22

5 TM

spec 177.08) : TSQ's (Cont):

YDA 7088r

One kind of (perhaps) Ph I problem could involve "Logical Reasoning". I may want to delay it to Ph II & have it learn a very special formalism. How much of Algebra can be "found" w.o. "Logical Reasoning"?

SOME IDeas I want to write about:

(1) Can I use PPM w. a corpus containing "Don't Care" Symbols? e.g. consider expressn: sum(α, β), where α & β can be large expressns. I'd like PPM to be able to "Notice" the occurances of "sum(xx, β)" where xx is any arby expressn & β is fixed.

(2) Consider the possy of a Wirthy TSQ in which I really don't know how TM is supposed to solve the problems! A constraint: But they would be a reasonable TSQ for humans & yet would not need "Human-specific" info. → 180.04

(3) In the special kind of TSQ considered in the discussn up to 177.27 Its a kind of "Problem Pool" I have fixed set of $Q_i A_i$'s: I get TM to try to get various S^j's s.t. $\forall i$: $UCS^j, Q_i, R_{ij}) = A_i$. w. $|S^j| + \sum_i |R_{ij}| = \min$. Each S^j will have an assoc wt. or how hy a (pc x apriy) it got for the entire corpus. When we add a new Q_{nn}, A_{nn} to the corpus, we try all the old S^j's on it & we also make a ~~new~~ S-language to fit the S^j's: We already had a slang to fit the S^j's that did only $Q_1 A_1 \cdots Q_n A_n$, so We need only modify that slang.

N.B. T. top (.10 - .20): It Does seem like Conventional "Boosting" would be applicable to it. Also the various S^j's are used in || for Predn ... Just as in "Boosting". Trouble is there is no way to "amplify" on QA's w. poor scores. → One Reasonable way: In the Slangs. The various S^j's are given wts or pc they assign to corpus. This wt. can be modified so as to QA in a way equivalent to a certain QA having occured several times. (Simply take the K^{th} power of its pc, to simulate a case count of K). The resultant S-lang. would be used to find S^j's that did well on that particular QA. — which is how "Boosting" works.

We have to modify grammar specially to deal w. each QA that we're having Trouble W. — This is done by simply modifying the wts of the various S^j's in accord w. (26 - 30). → (spec 180.00) →

Parallelization of t. S^j srch. for several ll processors.

(Spec 179.40) ← This "Boosting" stuff may be V.G.! Among other Values: It gives lots of S^j codes ∴ lots of "Diversity"!

04:179.12 T. way it mite work! I have this TSQ & I have an associated Set of Conc. (in English) that tell what concs I think are needed to lrn each QA. The Concs are simply "Names" & I don't necessarily know a good way to Program them. This training routine is pretty much what a teacher mite use w. Human a/c Animal Students.

Occasionally (may be often) the Concs I have assigned, really are not meaningful enuf, or have excessively long perms to express them — in which case, TM can't work these problems, & I have to use diffrnt. Concs, diffrnt problems to get along in t. Seq. Similar things occur w. human/animal Students.

Parallelization of t. S^j srch.: Say we have k processors! Have a "Master" assign diffrnt S^j conds to each processor. As soon as one processor finishes its job, it tells Master! Master then asks all other processors how many undone S^j's they have.

(α) Better: Master ~~keeps track of~~ initially assigns tasks & keeps track of which tasks have been solved by each processor. On this basis, it removes tasks from certain processors & adds them to others so as to keep a list of undone tasks about same length for each processor.

(β) Or: Master only takes action when one of the processors has only 2 more tasks on its list. When this occurs, it redistributes remaining tasks so as to equalize loads of each processor.

(γ) Or: Master has list of S^j's; Each processor asks for one from Master. As soon as it finishes, it asks for another. These tasks are simply kept on a stack to which all processors have access. There is, of course, the problem of interference when several processors request tasks at same time. Likely that this problem has been worked on much for web servers, etc. (7.34)

Good Simple

After a procr. does an S^j, it sends back to Master a Gore vector. After the entire Run these Gore Vectors are analyzed, & the S^j always is devised in line w. the "boosting" idea of 179.22 ff.

34 (30.) One way to do this: Rank processors by confidence: If one or more

5 TM

have simult. request, the one w. hyest Precedence no. gets served first. (Still not entirely clear!)

This will work.

Say have only 8 processors. Master has 8 flip-flops — one for each of the processors. When a request comes in from a processor its f.f. is set. Master communicates w. highest order set f.f. & then resets it. Then scans for next hyest set f.f.

Or, instead of f.f. Just have a 32 bit input w. 1 bit, different bit from each processor. Picking hyest "1" bit is easy, & setting it to ϕ after that processor is serviced, is easy to do.

Normally the 32 bit input is all zeros. As soon as it is not all zeros, Master attends to it. In II w. this register is an "or" gate so that any input request will make it go hy and request an interrupt. (NM!?).

TSQ's! For Algebra: Q: (If (X+3=1) Then X = ?) A: -2

Q: (If (X+3=1) Then ?) A: would have many possible answers.

As is, I'm considering only QA's like

eval (Q 7+3) → (A) 10 from −, ×, ÷, | from eval (7+(3÷2)) other kinds.

Then If X=1+1 Then X=2 (or X=1+1).

If X−1=2 then X=3

and more complex / newer than h.l. eqns.

Try to teach every "Einstein hour" about use of X & manipulating it until you find out what it is. Give problems that would have this hour as only type → 189.16

technique

. . . . . .

For Par TSQ's! Consider other forms of data representation.

Koza has a v.g. somewhat general way to represent "Electronic ckts" that is applicable (probly) to a much wider variety of problems.

Also Gabriel Kraas "Tensor Analysis".

Also Sussman on "One Shot Inn" has a different way of representing Reys & a different way of updating his models.

Re: Koza: An easy way to study his notation: First learn how SPICE input looks. From this, it shouldn't be hard to devise pgms & structures that would fit that notation.

→ Also Notation Used in Sys. Evolution describing Organisms "Evo Devo".

T. nitro Ring About Evo Devo Notation: It's likely that there will be easy ways to recognize useful "circuits". → 183.00

7.31.05

5 TOM

GACS Cont.

SPEC

178.34 One way to deal w. this: In order to do AriPh coding, we have to have p.d. on

Continuations: To use ALP to approximate K, we need a C.B. that ↑ w. corpus

length. Q: is how much error can we expect from any finite CB?

— even tho it ↑ very rapidly w. corpus length, we only need a few bits to

express this larger no. Results >> corpus length.

Unfortly, the CC ↑ more rapidly than any recursive function (needed) ···· so this probly will not work.

However, the general idea of using AriPh coding for any approxn. to ALP, means that for all practical purposes, using shortest code is within a constant pc factor of ≈ ALP — it suggests also that the "constant factor" is ≈ 1.

But it couldn't be ≈ 1, since the near-shortest codes must have somewhat reasonable pc.!

Anyway an important idea is that if μ is a CPM, then we can get a "shortest code" of, of length ℓ ⇒ $\cdot 2^{-\ell} > \frac{1}{2}\mu(s)\cdot P(\mu)$, using AriPh Coding.

where $P(\mu) = 2^{-\text{length of def of } \mu}$. $\frac{1}{2}$ is the "round off" error

Note that we are using here a "2-part code" as in MDL/MML.

Still, it would seem that ALP has to be > MDL because it includes a lot more (!) codes! — so I don't understand yet! Well it uses a V. G. Code, the sum of all other codes will be < ½ wt. of shortest code

Consider a simple non CPM: A Bern seq. with an incomputable/pc parameter.

In this case, each Finite Bern seq. can have its own code ··· not nearly a continuation of a previous code! Perhaps ≡ Covar's "Extension Complexity"

8.3.05 — New title. Brown Bridgewater State College. Shoobridge. — 183

5TM

Formulas and Theorems in Pure Mathematics — George S. Carr

2nd edn 1970 Chelsea Pub. Co. NY.

ISBN 0828402396

00 — : 181.34 **Re:** Forth pgms! Can they be regarded as RPN expressions ... ?

in which case, pgms would be trees & contexts would be subtrees ... just as in Lisp.

.01 — w. additional feature that Forth can be "fast & cheap". i.e. lots of not so fast µ procs: each w. maybe 256K of Memory in chip & no external memory.

~~Total~~ They could cost: AMD Sempron 2800 socket 754 L2 cache 256K : $60 — No rebate Mail in Rebate.

$12 for socket w. fan; if it's 30 watts #23 650 w ps. has ~120 watts @ 3.3 v power —

so $\frac{23}{4}$ = $6 for power 60+12+6 = 78 $\frac{2800}{74}$ = 36 MHz/$.

So $10 K buys 3600 Ghz : Not bad.

For $30 one can get MoBo for Rest CPU. (includes socket.)

So $18 more. "Not exactly new".

10 ~~Then~~ one can get MotherBds for $10 — but dittos CPUs:

look into this. Say, look at Forth pgms in COPS:

My impression of the way Forth works: The pgm is a sequence of atoms & functions. ~~Then~~ We move along the pgm: atoms are pushed onto the stack, Functions are executed. The arguments of all functions are on the stack. When we have gone thru' all elements of the pgm, the "result" is on the stack. From the design, it would seem that the "pgm" is simply an expression in RPN.

I don't yet see why we need 2 stacks in Forth. — Also, it may be possible to not use "Next", etc.

How do we define a function? well def of: name, no. args, (no. of outputs), & any locals, & end of def special.

20 How do recursions work? How is "If" implemented — say in defining X! (?).

If the pgm is on a special stack, perhaps it can be implemented by certain stack operations.

Actually: Normal forth can't do recursion, but there's a simple modification that enables recursion (But only primitive prime tail recursion?)

NB Re: Hardware methods (.01 - .10) If each cand is tested individually, w.o. storing state of previous cand, that was "almost the same", we have a speed loss of $\approx \frac{1}{n}$, where d is length cand. To avoid this (perhaps) we give each µproc $\frac{1}{2}^{nd}$ set of cands to test. [To theoretically analyse; this "set" can be regarded as a "single" trial: & there is a (perhaps) formal way to regard the length (= no. of symbols) in a trial to be so $\frac{1}{d}$ becomes larger.

37 Other objections: For each µproc we need a clock signal. This can be obtained by each µproc having its own small "motherbd", or by a single clock for all the µprocs: they don't have to be "in phase"

Objection: The Apm by with the µproc. accesses the Cache many times → 184.00 Space.

space
183.34 "Trade secret"
a closely guarded industrial secret. So this would (to some extent) have to be breached. (probly illegal: but maybe not if we don't reveal it to others ("trade secret"!)
We may be able to publish effective way(s) to use it. Caches mean, hmm
Even w.o. use of HW mech of 183.05-.10, all modern uproc do a lot of stuff
in ll (which is how AMD chips run at say 1.3 GHz a "set rating" 2800)
so we would have to use this ll processing — which is probly closely
coupled w. use of the L2 cache. (or L1?).

It may be that Forth is small enough to use to L1 cache entirely!
faster, smaller than L2 Cache.

ANOTHER HW possy is use of FPGA's Field pgmd. Gate Arrays

O.K. Lets study Lisp & Forth: What kinds of functions are used?
I looked at this Book "Invitation to Forth" - Katzan.
Easy to read, Easy to figure out what's being done:
1) Forth uses RPN; Lisp uses Polish.
2) Forth doesn't normally have a "Quote" notation. If it were used!
3 4 + would mean put 3, then 4 on stack: since + is an operator,
it is then executed. To do "Quote" write 3 4 Q + This put "+" on
stack & does not yet execute it. When Quote occurs a flag = 1.
as soon as operation is put on stack, flag = 0.

I have now pop'd stack, we have an operator & flag = 0, so
We execute that operator (?) How is it done in Lisp?
well Lisp has Quotes, so it can create an entire expression w. ease!

The tree structure of functions is true only for "Composed" functions.
We may be able to retain the tree structure if we use functionals, that
map functions & params into new funcs: Do, when, while loops do this, and
are probly adequate for prim. rec. funcs. → 185.08

27 Perhaps IMPT IDEA: I was Concerned that PPM would not be able
to recognize/discover frequently occurring sub-trees (Representing sub-functions)
However, if I did use PPM to help select cards, there would be a strong
30 bias toward sub functions that were recognizable by PPM

But I really have to Look at the kinds of Regs that occurred Do, when, until
loops & normal recursion

One trouble w. .27-.30! If I write a bunch of pgms to "initialize" PPM's
P.d. on cards, the pgms I wrote would tend not to be of type .27-.30 ---[however,
eventually TM (mite) would bias in that direction]

STM

.00 SN An important part of Algebraic learning will be Logical Analysis. "tail recursion"
I want to see how far one can go w.o. getting on to it, hwr. Perhaps
try rereading old important notes on Logical Analysis learning.
I think one basic idea was to have TM learn Logic as a pure Math Thing,
then try to apply it to "Real" problems in an "Analogic" way.
This would seem to be quite different from having TM learn to understand
Logic, as a special case of PC!

08 184.25 So: This "found" functions in Lisp/Forth seems very important. I'd like to
find a way to make the detection of "subfuncts" cheap/easy.
Consider a Do loop: Do i, N, Funct(i) OD. (#index (not used in Forth); Maybe unecc, since func(i) must have an "End" marker.
"Funct(i)" is not really a funct. It is a PGM: we may, hwr, consider
"Func(i)" to be something to be executed. If it is a function, it could have
"side effects" like changing contents of RAM, for printing output? ...
Then "output" could be regarded as a particular Ram address: (Or as TOS)
factorial [redacted] N); X=1 Do X=X·i (TOP of stack.
until i=N.
fact(N) 1 DO X←X·i until i=N // Looks like RPN is. Different from PN!
i.e. "3 4 add" in Forth.
easy to do w. stack; "add 4 3" in Lisp: Notes easy to do w. Stack!
In Forth Q (dup mul) maybe quote & unquote to set, unset "pgm to flag".
def "dup mul"
In addition to Forth & Lisp: there is Joy: He did something so that
pgms would be funcs: ① Did he do it? ② How?
I looked at Jurgen's OOPS Forth: It may be, that as one moves along
to pgm, one can think of each symbol creating a new "state" & Returns a function
of the previous "state". — which sounds very "sequential" <i.e. amenable amenable to PPM>
— Still, Functs will often be defined by the last 2 elements of the "STACK".
Tail Recursion: See OOPS paper p. 34 (& Sussman's Book on Lisp.)
Tail rec. makes it possible to define many recursive functs by putting
the name of an already defined function, the value of the arg. of the new func & the value of the new funct for
arg t=0. So it enables us to define funcs of the form f(n+1) = g(n, f(n)), f(0) = a.
While it might be used to define the sum of a series for arg n, it would not usually
.33 do this efficiently. The usual Forth way is S(n+1) = S(n) + f(n+1)
.34 f(n+1) = g(f(n), n). F(n) could be defined recursively but

t. implementation of t. summation would be very inefficient.

In General, hvr; I want t. System to be Truly "Universal" so that it could, in principal, implement/simulate any/one kind of computing something.

Hvr, there will be more common ways of writing pgms & I should try to get a good P.D. over them at least. Then spend some time on looking for other kinds of Recrs.

In t. case of tail recursion, we can regard Tail rec. as a function of 3 things: a number and 2 functions of 2 variables. So in this sense, it could be regarded as a sub-tree/sub-function which could be findable by "subtree search".

Actually Tail recursion need not be implemented as OOP's does it. It seems like an important enough function so that we should make a special pgm to implement it Easily - (which is easy to do).

We may also want a special pgm. for implementing series: e.g. Viz 185.33-34. To define a series, we only need to give t. first term, and t. function that gets t. next term from t. previous one - i.e. a function of 2 variables: Previous term & n. So it has some forms of tail recursion, but different result.

There is a simple Generic of Tail recursion that may not converge: We start w. $F(0)$, then $f(n+1) = \phi(f(n), n)$ we do not, i.e. no: $f^{(n+1)}(\phi) = \psi(f(n), 0$

$f(\phi) = \psi$ $\quad f(\phi(n+1)) = f(\psi(\phi(n)), \chi(\phi(n)))$

a simple version $f(\phi^{(n)}(0)) = \psi(f(\phi^{(n-1)}(0)), \phi^{(n-1)}(0))$

I'm not certain of this, however.... I have written much on this sort of thing as the soln. of certain kinds of Functional Eqns.

1) $F(G(x)) = G(F(x))$ say G is known, F, unknown.

So $G(F(G(x))) = G(G(F(x))) = G(G(\alpha))$ assume $f(x) = \alpha$ for some x.

1) gives $F(G(G(x))) = G(G(\alpha))$ which is now known. $\quad G(F(x))$

$G(F(G(G(x)))) = G(G(G(\alpha))) = F(G(G(G(x))))$ $\quad = \alpha(G(G(G(x))))$

does 1) imply $F(G^{(n)}(\alpha)) = G^{(n)}(F(\alpha))$?

if $F(G^{(n)}(x))$

G seems to simplify/resolve H, G known. $H(f(x)) = f(G(x))$ $\quad H(Hf) = f$

$H^n f(x) = f^n(x)$ say $f(x) = \alpha$; $G(x) = \beta$

$H^n \alpha = f^n \beta$ - this gives $f(\beta), f^2(\beta), f^3(\beta)$ etc ... as values of f for certain values.

9.5.05

5TH

Functional Eq. $f G = H f$: Completed solns: .07 - .08 (.15)

Found, G & H were known.

$H f(x) = f G(x)$ $\quad H f(f(x)) = f G(f(x))$

Note Gennaot .27 gives factorial function.

$H(\alpha) = f\beta$ so $f(x) \to \alpha$; $f(G(x)) = \beta$

$H f G(x) = f(G$ $\quad f(x_0) = \alpha \therefore f(G(x_0)) = H(\alpha) = f(G(x_0)$

$f G(x_0) = H(\alpha)$ $\quad H f(G(x_0)) = f G$

$f G(x_0) = H f(x_0)$

$H f(G(x_0)) = f G G(x_0) = f G^2 x_0$

$H f G^2 x_0 = f G^3 x_0$ $\quad H f G^n x_0 = f G^{n+1} x_0$

.07 $f G^{n+1} x_0 = H f G^n x_0$ — which solves it along w. $f G^{(0)} x_0 = \alpha$

.08 $f G^2(x_0) = \alpha$ ← Boundary Condy. I Picard Principle > Complete Soln

$f G^{n+1} x_0 =$ so $G^n x_0$ leaves successive values of argt. of f

$H f$ leaves successive values of f for Reuse as x's.

$H^n G^n x_0$ $\quad H f G^n x_0 = H H G^{n-1} x_0 = H^2 G^{n-2} x_0$ $\quad f G^n x_0 = H f G^{n+1} x_0$

$f G^{n+1} x_0 = H^n f(x_0)$

0 $f(x_0) \equiv \beta$. $\quad H f G^n x_0 \text{ eff } \alpha = f G x_0$

$f G G^n(x) = H f G^n(x_0)$

.15 $f G x_0 = H f x_0$ $\quad f G^2 x_0 = H^2 f x_0$ → $f(G^n(x_0)) = H^n(f(x_0))$

Hrr, I'm not sure $fG = Gf$ often occurs as a problem to be solved in "AI"!

If $f(x) = x+1$ and $H^n(x) = \lambda x(y-x)$ we get the Logarithmic Funct.

But we can't get factorial! H^{n+1} would have to be a funct of n as well as H^n.

$F^{n+1} = G(f^n, n)$ $\quad f f^n = G(f^n, n)$ $\quad f(x) = G(x, n)$

G is known. Maybe not. Ex $f^{n+1} = G(f^n, n) = (n+1) f^n$

$f(n+1) = (n+1) f(n)$ $\quad (\alpha+1) \cdot f(\alpha) = f(\alpha+1)$

23 $g^n(\alpha)$ $\quad f(g^n, f(g^n))$ $\quad f g^{n+1} = H(g^n, f(g^n))$ $\quad (f^n = u$

So factorial is of form $f(g^{n+1}(x_0)) = H(g^n(x_0), f(g^n(x_0))$

$f(g^{n+1}(x)) = H(g^n(x_0), f(g^n(x_0)) = f(g(g^n(x)) = H(g^n(x), f(g^n(x))$

$f(g, g) = H(x, f(x))$ $\quad f(g(x)) = H(x, f(x)$

27 $f(g(x)) = H(x, f(x))$ for factorial $\quad g(x) = x+1$; $H(x, f) = (x+1) f(x)$

$g(x) = x+1 \quad H(x, f) = (x+1) \cdot f$ for factorial $\quad = f(x)(x+1)$

.27 is cleanest Gennaot $f G(x) = H(f(x))$

Actually H^{n+1} is same as .27 $f(g(x)) = H(g(x), f(x))$ also gives factorial w. $g(x) = x+1$ ($H = g \cdot f$

$f(x) = H(x, f(x))$ no

An Imp idea is: How often & when will I make definitions? To some extent, "defns" are obviated by use of PPM. But in many cases, I will want to use functions as arguments of (functionals), & it would be nice if fn. args were in the "Dictionary" — i.e. they were "Tokens"

I may or may not want to include Recursions in the PPM corpus. I don't know if it would help precision of pc's for PPM.

For functional args, I may not need "tokens" ... I may be able to use PPM to generate needed functions. The context will be a functional so certainly mainly functions will be generated/tried.

[SN] on Funct eqns: $f(x+y) = f(x) * f(y) \to f(G(x,y)) = G(f(x), f(y))$ Has "simple" soln.

Look at General functional Eq. for prim. recursion. It may be that all prim rec. functions have simple attractors. They may be "fractals", hvr — i.e. have fractional "dimension".

Is $f(G(x)) = H(x, f(x))$ a most General Prim rec. functional eqn? i.e. defines f in terms of G, H, & $f(x_0)$.

Soln. may be 187.23: $f(g^{n+1}(x_0)) = H(g^n(x_0), f(g^n(x_0)))$ ← is this nice soln of .15?

So we first generate $g^n(x_0)$ for [illegible] $n = 0|N$: Then we expect $f(g^n(x_0))$.

Can we get $F(f^{n+1}(x_0)) = H(K^n(x_0), F(f^n(x_0)))$?

$F(G(x)) = H(K(x), F(x))$

$F(G(x)) = H(x, f(x))$ ↓ .15

$H(G(x), f(G(x))) = F G^2(x)$ so .17 may be correct.

Anyway .12–.14 isn't problematic at once.

According to Chris Moore, Prim rec funcs commonly use Do loops w. specified index limit. "While" loops can give Part rec funs, hvr (& presumably "Until" loops as well). I guess while & until loops are (pretty almost much) equiv. "While" comes before function eval'n, & "Until" comes after. Hvr "until" can discuss results of func eval'n. "While"/can discusses result of previous eval'n. — so may be they are really equiv.

So, for a Do loop, we put in upper index (may be also lower index) and the routine to be executed (perhaps in quotes), when I insert "Loop" occurs, the next symbol has to be the index limit. The first quotes is automatic, then the pgm, then another Quote, which ends the loop def'n.

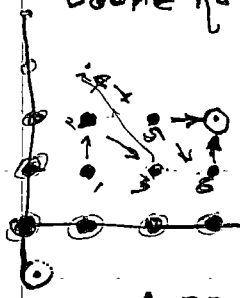
2-3, 4.5 hrs.

Woken 14" snow 8:00 8:15AM

Hook Wearing RPN Hour.

So, superficially it seems like I can do For Pgms that will do "Do" loops & while, until loops, & this may give all nec. part. rec. funcs. I don't see how to get double & triple recursion, hm.

"Double Recursion"??



$f(x,y) = G(f(x-1, y-1), x, y)$ — is this "double recursion?"

It looks like a Partial Difference eq.

initial conds: $F(x, y=0) = \alpha(x)$ known, $F(x=0, y) = \beta(y)$

A BIG Q: I'm not sure what these 2 varieties of recursion:

"for" loop (prim rec. funcs) & "while" loop (part rec. funcs) really cover all I want to say. Perhaps best thing to do: write TSQ: write answers (programs) by hand & see what "insts" I need: Search "Rapys" are accessible to PPM (or APPM)

I could start TM out w/ a min set of insts to work initial problems: Then gradually add insts as t. corpus grows. PPM knows how to deal w. "new symbols"

Ok: TSQ! → Also 1006 & 181.12 ANL; Run eq. soln.

ANL stores first as: 3 + 4 ; then Nested expressions. Perhaps use Parens.

To preface Eq. Solving: Would it be possible to teach it "Einstein heuristic"? "Einstein Hacer"

Consider t. string 3 + 7 : How would I process it?

push 3 onto stack "+" is a word — unclear

SN A function(al) of type both they can could be "inverse". Its a part. rec. funct. "naked" means "no dressing", no "speed up" no heuristics or best way of finding it. Mindful of Lenat "slots" in AM: One for a function, another for a fast (efficient) version of that function.

SN Note importance of generality of hacker for noting "correlates in Q.R."

3 + 7 → 3, 7 Add (Q2) Coding of this General type are very important — could be inserted into TM "A.H.": Post block, & +, add correlate is observed, then TM tries to find short code to "link" positions of these 2 symbols.

This is not part of Universal Induction but part of A PPM.

SN Another very general "lengthy trick" If a certain Conc. seems (by me) to be needed in a certain problem, This conc. can be taught by (long job) it as a TSQ for a "definition" of that Conc. [This was an (impt. one of my reasons I decided to use t. QA type of TSQ.]

This Mike be regarded as a "hint" because it simplifies t. trng of a → 190.00

00 — 189.3d Conc. a Remedy loses its value as a tool: i.e. a student probably ends up being less clever than it did a longer track needed to acquire a conc.

legitimately

for TSQ. Do several ADL tsq's: Infix to (RPN / PN)

PN to RPN. PN to INFIX? PN w. parens ⇔ PN w.o. parens.

Have Q's problems in form Eval(3 +7). As in Maple, this can be done

Numerically a/o symbolically.

The idea of equality/equivalence could be taught as "tactics": - but

See 189.30 ff. for possl. disadvantages.

Is this a Good Notation? Eval(1 eq 2) → 0 (false)

10 Eval(1 eq 1) → 1 (True).

Perhaps that Q is not so IMPT! Because! These Defns will be useful because they have subtrees in them, that will be useful in other problems —

More exactly, in problems that they were designed to help solve.

A possibly useful defn: Substitute (α, β, γ)

e.g. Subs (2X, Z, 2X+Z) → 2X+2X.

Subs (2X, 3Y, 3Y+2Y) → 2X+2Y (may be ambiguous?)

Subs ([redacted], [redacted], XXX) → YX or XY. In case of ambiguity, we could require both answers.

20 SN Maximum (al) Common Subgraph (lookup both by Google)

Downloaded 43 papers (PS) 8.7.05

One paper is not to say who has some he wants to send people. ("it's a PC." "exe" file)

Another says they have a "fast algm" for "sparse Graphs".

Also Maximum Common Subtrees (Google) has lots of PP's

Maximal " " (") Not strictly defined set of PP.

Try other Search Engines.

PAMI ~2000 had Special Issue on Graph Algms in Computer Vision.

Turns out: PAMI already has this Report

28 TSQ: Very Impt Heur: When I solve a problem in a novel (not old routine) way: Task: how can this technique be Generalized?

30 What other kinds of probs will it solve? The idea is to put the routine into an "associative database", so that when a new relevant problem occurs, I will think of using this new technique. It does mean that I do have the Data Base of "Methods" to be used w/ NEW problems, & there are "Hooks" in the Data Base that analyze a problem → 191.15 Spec.

Ideas for Talk(s):

1) kinds of TS Q's that I will not use at the beginning:

a) # Visual, Acoustic Scenes: (Too much cc)

b) Nat lang. : Better develop internal lang at beginning. Mapping into "Eng should be easy".

c) Use of Xtra corpora for lang (English & MT.) : Probly n.g. at start — concentrate on small corpora, One fact lang — Move overall — also emphasis on small corpora etc in interdisciplinary for good induction algms

d) "Hard problems" to compare w. OR or workers. For TM to work have problems when it is young, suggests that it has been tested w. A.I. techniques. This is not a good way to teach.

5 (Space 190.34) → To find impt. properties of it, so that/trad methods may be assigned to it. The Newly Solved problem has to be analysed to find characteristics of it that will enable make it easy to tell if it is applicable to a particular New Problem. These Property(s) must then be added to the list of operators used on new problems to tell how what prob solving methods should be tried on them.

21 In Phase II of "QATM": I have this (one of PSM's. Assoc w. it is a p.d. that maps problems into (we hope) appropriate PSM's. So, on 190.28 when we get this (problem, PSM) pair, we want to integrate it into the P.D. that links probs & PSMs. It has to be recorded as an input/New part of the corpus of this P.D., & we have to update that P.D. wrt that "corpus addition"

So .21 makes the (Meta?) heuristic of 190.28ff a normal/ Part of Phase II ... The fact that the Soln to the new problem second unusual means that the needed update may be difficult

30 Essentially, what we want to do is make mods in the P.D. from Problem to PSM ⇒ the new example (as well as old examples) are of type. I think there is a "Recognition Algm" implied in this (Meta) heuristic.

Essentially what PhII does is CLASSIFY probs directly. Its input data are formed triples ($Prob_i$, PSM_j, E_{ij}). The is Turing limit incl. of prob's

5TM

We want t. pc that PSM_j will get Corr. = G_K w $cc = t_i$
We also want an assoc. Algm. to list PSM_j w/ by expected G_K
(we assume "linearized" Corr).

8.12.05 SN Possl. Idea: After T. Corpus has been processd by a "Universal device". Its Output is in form of either ① A/single more Compact recoding of t. corpus ② Several recodings of t. Corpus ③ A seq. of pc Vectors for t. various Symbols of t. Corpus or pc's of new Symbols that can be inserted into Corpus. ④ Several ||| codings such as ③.

The sequence of symbols in ① can either be Compressed or expressed as a seq. of pc's by a new induction (not nec. Univl) algm. This can be done in ② as well, yielding multiple code Corpus. In ③ Ro pc vectors can be regarded as vectors to be predicted — using various methods for Predn of real vectors. The original real vectors may or may not be normed, & their second order predns need not be normed.

.17 169.21 SN2 On Normalization: Say we are using a 3 time (3 input time) form of Universal device: In a "Batch" Round (T ← 3T sequence) of LSRCH One can keep track of total pc's of kinds that have been & have not been completed. At the end of each "round", we will have an upper & lower limit for t. normzn constant. The mean of these will give a value w. a ± attached. so we get $A \pm \sigma_A$ (A = Normzn Const).

Previous to .17 I had been thinking of a "Round" as an attempt to get t. first (& perhaps t. second & third) code for A_2, & stop at that pt. In fact, in LSRCH, once always does complete batches (for each T value) before "T ← 3T". (I guess that + size of t. "±" was always to big to make t. "mean" useful.)

SN3 Not now, but should be realized (aky w. t. optimality of "3" in T←3T) in the course of LSRCH 5 If we process each Run as individual cand. codes & don't take advantage of much overlap repetition of work in subtrees in processing of diff. codes, we will lose a factor of cc amounting to L, the length of t. codes. Hm. This is rather ambiguous: It is true for all cands at/in binary tree level of a length. Modifies for other alphabets & various code lengths.... I don't know how t. results are Modified. —— This Q.13 (spec 19200)

5TM OOPS.10 [access.shortcut.table]

192.3# of much import when one is using (for Lsearch) many uprocs all [183.01 ff]
that are fed cmd. sequences to be processed by individual uprocs.
i.e. we will probably want each uproc to be given a set of commands
closely related cmds. to be executed. Just how much this can effect
the $\frac{1}{L}$ factor, is at present, unclear.

(Re: In) OOPS: Ivery said that while it does not now use "editing facilities" much
to create new tools, it does have a Complete set of insts. so it could,
in theory, eventually discover another derivable set of useful tools. Is this true? Can we give OOPS a task that would enable it to Discover
an arby Heur? Seems like Maybe "Yes"!

I would have to work out the details of this, hrm.

One way: OOPS does have access to complete old pgms ② Invent
insts. within its existing lang, to modify pgms to be tools for
new problems.

Expo:

1) (4 kinds of) Natural Intelligent Systems
a) Man b) Animals c) Org. Evolution / Development d) Society as whole / Science Community
e) Swarm Intelligence.

· How TM works:

There are 3 ways to put info into system.

① Original Turing Complete Lang should have primitives that can concisely, independently express simple ideas in Math, Sci & used effectively (Interest; e.g. Lisp (dialects); FORTH (dialects — not lisp) APL: Many primitives/fcns/structures in Maple/MathematicA (patterns).

② TSQ: Can Comp from usual text books / Ram Task. / defns in Maple MathematicA to be learned — rather than as "primitives" :: Learning Heuristics of Early A.I. How Prob Solving

③ ~~Probabilistic~~ eval methods & search for spec

From ① & ② we could have a theoretically very smart TM, but it would be much too slow. To speed up search for good inductive models:
Use current various current induction systems: PPM (Fast, Good Predictions) (ANN perhaps), SVM machines) — both suitably modified, GP (suitably modified using Prob Grammar Pick Population Strategy), Stochastic Grammars (both CFG & CSG).

Any present ML mechanisms might conceivably used to help my search.

STM

177.02: Teaching TM what "Optimum" means: I may be able to do this by giving many (as in "Resource Limited Often problems") disparate examples, along w. a set of concepts from which "Optimum" is to be constructed. If does seem like hard thing to teach, fine! Try to devise scenarios in which "time limited often" (vs resource) would be "learnable".

Random ref: SVM: 171, 172 (42.30) | Rev. of Ph I §170 | 3 speedup Methods 164.00 | 164.11-.13, 169.08 (Lsrch v.s. BUsrch)

Forth 183.00 | 164.00 | 166.08-.30 ? How to do Lsrch, BU

BUsrch: 167.06-.25 | 166.08-.30 | 167.06-.25 srch in ||.

Recs: 178, 182 (MDL v.s. ALP) | Review of Types of TSP's to try: 160.01

I Guess 164.00-.40 is main idea for QATM, Phase I.

In 167.10, I was thinking of Lsrch as doing trials on "pc of and" [redacted] order — so it's pretty much INV search!

W.R.T. BU srch: It starts w. a Gud (usually not v.g.) & tries to improve it.

GP, GA-like methods or improved w. D.P. instead of population & use of (Max-Mean) criterion?

but what about cc limit?

191.01: kinds of TSP's I will not use initially.

Types of ML likely to be useful in TM Meta-srch!

PPM, ANN, RANN, SVM, Decision trees, (Boolean Belief Nets?), linear, n.l. regression, GP,

PPM: BU search w. backup w. lower cc, moves to better points w. higher cc.

Lsrch: Start w. lower cc, backup. Increase cc, better pt.

SN: At present, it is unclear to me as to just how the TSP framework is used in BU srch. Actually, it's probably only used on problems that L. srch is currently working on or has recently been working on. Consider e.g. QAP TSP!

Lsrch has obtained a Pd. for, say, $O^k \to O^{nk+1}$ (or $S^n \to S^{nn}$?)

Say Lsrch has worked on this particular one for a long time w. o. success

So we take bad (e.g. G.) soln., or a mixture of soln's.

We can always start w. $S' \equiv \Lambda$ which will give a soln. — usually very bad, but improvable by Hill Climbing.

BU could use PPM corpus of $[O \to O^{nk+1}]^{n-1}$ or just corpus $[O^k]^n$; Then use this to generate Gud's, using the (Max-Mean) trick in the "GP." — we adjust threshold ..... but unclear as to how! Start w. threshold = -∞: When we have enough empirical examples in corpus, then begin to try to use

Also, how (in BUsrch) do we deal w. trials that don't converge?

Lsrch must be Modified ... but How?

BIG DIFFY

→ 196.00

STM

195.35 : Say we have Pcs Pd. on cands. ? ← The phrase "Business" it looks like I'm thinking about INV prob rather than OZ, & can't, chiefly now, really what Beach is BUSY on $\frac{cc_i}{p_{c_i}} < T$.

→ Also, say we have reconstructed (somehow, w.o. search) a small corpus of A.H. cands that do (marginally) "work"! From them, we can get a $\frac{cc_i}{p_{c_i}}$ threshold, and use it to discard cands. Maybe use threshold = $(2) \times \frac{cc_i}{p_{c_i}}$ max — assume that small constant.

Is this last "reasonable"? I suspect that "2 x" $\frac{cc_i}{p_{c_i}}$ max will be well beyond the time available for search: So for all practical purposes, it's an "infinite set" — it could never be explored by "search". — It is explored via "Monte Carlo" wise — from G.P. or G.A. ⇒ 213.60

Another Q is whether the cands can be tried in "sequential" way, (like an actual Time) — so we can discard a cand before we have completely generated/tested it. (i.e. after the symbol has been generated & partly executed, we could be already "Timedout")

Is there any awareness of Pcs distrib in the 80 pubs in AP 167—195? i.e. 195.31

I browsed thru it: No Mention!

Any of the normal Hill climbing methods don't face this problem because they all have (via Pcs) fixed upper bounds for cc_i : Not Universal

Hvr, remember, the goal is to get Best soln in available CB.

i.e. first Pc.

Even in normal "search" (not really search), we do trials in $\frac{cc_i}{p_{c_i}}$ order — but this is the ordering of the discrete models: The set of the corpus is a totally uncontrolled thing. (Actually, there are 2 kinds of problems: The INV like problems.

From 167.06 the idea on search here seems to be, do the discrete part (of each such model) via normal search & assume the continuous part takes a constant amount of time, per model. Actually, not really true: More contin params take more time .... but the cob off expr $\frac{cc_i}{p_{c_i}} < T$ can be modified to take this into account. We can empirically discover a constant, k, s.t. $\frac{cc_i}{(p_{c_i})^k}$ gives good estimate of threshold. or something like Pcs: say rashly time = $f(p_c)$ then $\frac{f(p_c)}{p_c} < T$ Determining $f(\cdot)$ may be hard, because we'd have to decide when to cut off trials! — trials w. $cc = \infty$ would disasterously bias results! Hvr .25—.27 looks "reasonable" anyway!

W.r.t. determining optimum continuous params: This is usually a finite problem & we can estimate its cc: a factor of 2 precision is fine.

Just where do the long trial (pam.rec) & the unending trial (perf.rec) occur? They can occur in (analysis of soln, no) & body problem (regarded as an INV problem, not a OZ or pc finding problem)

After a certain amount of cc spent on deriving a continuous model, we can estimate procs of contin params: The unestimatable part

5-TM

is the final pc at t. ≈ peak of the continuous params. So we pick our discrete ≈ part of the continuous part of the dcrn ≈ accept an "experimental output" ; the pc at t. ≈ peak. So its really a regular QL problem. So we're just doing bi trials in (cc of ps runs / params for pc) order.

Maybe do only LSrch (no BUsrch) in Phase I.

It may be that in Phase I we really can't get very close to Universal functs.

On the other hand, I was using Siemens to get probs distrbns in QA problems. They were discrete, rather than continuous P.D's, hvr. The peculiar thing about this Siemens was that we plugged our dcrn in (of known pc) & we then ran the function (which could take arbitrarily cc) & we stopped it if its cc was > $\frac{T}{pc}$. We then

If we were allowed to finish, we got a poss'l A_i & an assoc pc, for it.

.10 Actually, a search for codes for a particular A_i is an INV problem — but we are not solving it by LSrch. We are using $\tilde{L}$Srch.

Q_i, A_i is the future problem dcrn, we want $S \Rightarrow U(S, Q_i) = A_i$

.13 or $U(S, R_i Q_i) = A_i$ we want a "short" R_i. So this seems under control:

I'm not sure ---- it may, indeed, be LSrch!

Wouldn't LSrch look at both Q_i & A_i & try to find R_i?

.16 I.e. $(S, Q_i, A_i) \rightarrow R_i$, $U(R_i, Q_i) = A_i$

The analysis of .13 – .16 looks interesting & probably important, but no time now.

Use .13 as the problem & it's "$\tilde{L}$Srch."

.20 So we do get S-induction this way o.k.

Now how does BU Srch enter?

We start w. a set of R_i's & an S (maybe $S = \Lambda$) that does waste, tho the R_i are larger, we want to do some "H.C." to reduce them.

($|S| + \Sigma |R_i|$) the R_i can be arbitrarily long pgms.

In .13 if the cc, > T ≈ $2^{(-|S| - \Sigma |R_i|)}$, we stop | reject | $\frac{cc}{pc} \geq T$ to stop.

In BU Srch what limit should we use? In both $\tilde{L}$Srch & BU Srch | cc > T pc

We want to get max pc in available time.

In LSrch, since it is not Levin, is it at all optimum in any sense?

We do get a smallest $\left(\frac{|S| + \Sigma |R_i|}{-\log_2(cc_i)}\right)$ for the prob, which is something like the best pc for the available $\leq \beta$

.21 So what about BU Srch? Consider 196.01 – .06. We have a certain $c_1[R_2]$ w. a certain $cc \cdot 2^{(|S| + \Sigma R_i)} \equiv \gamma_0$ = Gone We then want to find cand's w. the same Gave better than this. It sounds like a "Ruth's ducky" soln. ---- Is it any good?

Well, in regular $\tilde{L}$Srch we tried to do a systematic/Robo search, in Lcost order. In 196.01 – .06 we consider cands in Lcost < a certain one we have found. → 198.03 spec'n

ST/N

Great feature of PPM for Search for good Models: 20

It can (perhaps) simulate a recursion using for=Next loop.

T. Forgy discussion has been for the case of discrete parameters for P.D. & Discrete P.D. How it would work for continuous Params, & continuous P.D.'s is unclear

03:(97.34 sec): In both cases, Least includes only the $|S| + \sum |R_i|$ i.e. cc of generating the R_i's: and generating final PC: it does not include the "final PC" which is regarded as part of the "Gen".

I guess it is somewhat Lsrch part of the search

The P.D. over the Cands will be determ'd Lsrch & BU Srch

Again, the Reasoning behind doing both Lsrch & BU srch is that for trainer-generated TSQ's, we can expect a good soln. w. acceptably small $|S| + \sum |R_i|$.

For problems not so benignly constructed, this will not be so, & we will start w. a large $|S| + \sum |R_i|$ & try to reduce it, & still get a good Soln.

What the Trick of 196.01-.06; 197?-.34 is & later, is uncertain. Legit.

20 EN HA! If we have a pure "Compositional Punchings" machine: By putting in "For i:=1 to n Do" loops, we seem to get great economy — it is compressed of n in code length. Here, if we use PPM for Mate search the compression ratio would be much less, because PPM would recognize that the code is mostly loops was the same. This PPM trick may not know the full power of a For i:=1 to n Do loop, it would still be much better than normal "comp of donuts" Machines.

Here, a function in a "for next" loop can modify > 1 matter variables: The output of a function is not to be a vector (as well as a scalar). as well as input

5 TM

#65/d. → ~~200~~ #195. 6:05

So what is present state of TM?
1) What is presently Contemplated "Main Path"?
2) What seems to be main difficulty.
1) TSQ for Elementary Algebra.
2) start w. MTM. (R≡0) #28

SN Ish Given Paths, try to find real problems that are close to mainline Math or Math or Tang. ed Audiences.

SN Possl. title ~~of~~ subject of talk : Prediction: ~~The Impossible Part~~ The Possible, The Impossible, and the Approximately. Give proof that Perm Test has error → 0 w. pr 1 is impossl. ← to state Talk.

SN Perhaps: Re: "Curse of Dimension" problem : A. Barron's Approach via Gazed ANN: Use Finite sep. predn : Corollary of Conv Perm.

$\frac{P_H(x_n)}{\mu(x_n)} > \alpha \qquad \underset{H}{E} \sum(\text{err})^2 < \frac{1}{2} \ln \alpha$

To use Barron's approach in a useful way, I'd have to know precision of his params as a function of n & d (i.e. no. of dimensions).

It is conceivable that Barron's method really does save us! — i.e. in which case we would simply use it as an ALP model and get to implied precision!

Barron's results would be in spirit of my eqn for cost of m param h-linear curve fitter — i.e. the exponent was set fixed at $\frac{1}{2}$ — i.e. → $\frac{1}{m^{\frac{1}{D}}}$ & cost-accuracy law $C + A n^D$

$\sum \text{err}^2 \propto C + A n^D \qquad \overline{\text{err}^2} = \frac{C}{n} + (AD - D)n$ ...

Which suggests that for large enough D, $\overline{\text{err}^2}$ ↑ w. n.. (i.e. if $AD > 1$).

My impression is that may be → $C + AD$? This seems wrong also.

$D = 1 \rightarrow \frac{1}{n}$ ← ≡ many errors $\sum \rightarrow \frac{1}{n} \ln n = \sum C + AD \ln N$ so $A = \frac{1}{2}$ in My case

$\text{Mean} N = \frac{C}{N} + \frac{AD \ln N}{N}$

Look at Sal 28 in A Barron ... ~ 2000?

Try Dmin X as walkes y dilation!

ANN or sine waves. This is 3 params (Sin Amp Cos Amp ω) 3 parameter

I think the idea was that the value of "A" depended on nature of the non-linearity

Rx good A = ½ for lines & locally linear non-linearity

28 04 → for MTM (or proply NMTM) TSQ's, we have to have "2^{k}" to be of a comp prable size (k is a measure of soln). so I think k should usually be < 10 & certainly ≤ 20!

This is certainly conspicuous re TSQ!

Could I get Gupta to work on this? It would need (at least) ppm for search over Operator (≡ function $Q_i \rightarrow A_i$) space.

Get him to read "OOPs", My Edsac report, & paper on PPM that I have & write up on how to apply PPM to ~~DATA~~ QA MTM → (X)

N.2.05

STM

200

Ashish Gupta

Braunstein has only 3 guest Rooms!

For Expositions: Some imp. ideas about ALP. ~~Most are not known~~ 6½

A few are commonly understood, Most are not.

1) (a) $P_M(x) = \sum z^{-l_x}$ (b) $P(x) = \sum a_i \text{Perm}_2(x)$. Moderate ⇒ Rich (not exactly)

2) Invariance theorem: Commonly understood.

3) Convergence Theorem: Rarely understood: That one P.D. can be close to a "bad" P.D.'s is ~~ok~~, that most people in bad ones is very extensive.

The implications of this theorem are ~~very~~ extensive.

4) Incomputability of $P_M(x)$: well accepted, but not well understood by many.

5) That incomputability is irrelevant to use of ALP for induction. Very few few people understand this. It's practically a "well kept secret"

6) Subjectivity of $P(x)$. Mixed appreciation. Many people ~~P(x)~~ $P(x)$ is not subjective. Others feel that the subjectivity makes it unacceptable! The truth is, it is subjective and that this is a naturally desirable property that all good probability evaluations from methods must possess.

α.34 SO: To what extent is a MTM TSQ useful introduction to a NMTM (stochastic) continuation? Well, it can have acquired many of the needed concepts as "contexts" that have been found in PPM for other current Mach Long techniques.

Would it be of interest to go more deeply into Math as they MTM problems?

→ Would it be possible to go very deeply into Math w.o. serious stochastic facilities?

At least this MTM will give me needed experience in TSQ writing.

Well, I could just try to get as far into math as possible & see if any probabilistic methods are needed on "the lowest (i.e. T-universal) level" (Tray will of course, be on the "search for operator" (PPM or whatever) level.

SN I had been mainly interested in sub-free (reason discovery) in extending PPM. Even if it had led by deeper, perfect sub-free discovery, it would not implement Prim. Recursive Funcs. 3 IS → 459

Any way to usefully get "do for" loops in extending PPM? Hm, I2mn ?

I did have one relatively simple way: design of Function als. (Ch 4 Sum

```
Do for i = 1 to n
    x = f(x, i)
Next i
```

a simple functional w. params. n, $f(\cdot_x, \cdot_i)$. Using the formalism of the A2 (?) log in the IDSIA Appendix, once we invoke the "dofor loop" formalism (which has a terminus), the next symbol has to be a formula for a no. (i's), & after that a formula for x ≥ 2 argt. function.

If it is ok for "level 1" — the primary universal functions

→ How to use it on Level 2? How to use even functional Composition on Level 2 ⊙ — or any function at all !!

Study — Characterizn of Q's (in QA probs) ! .06

Random notes

1) Perhaps write Pt per on Norman Prime in ALP.
Also note Rest Conv. Thm is true if R is not an Incomputable PM defined by a finitely describable Unic.

2) Is Conv. Thm. for ANN-type non-linearities: Does Barron's Thm. "Hold" — does it transcend "T. Curse of Dim" for info ?

06 → 3) On soln of QA problems: TM can do various "experiments" on Q: (Any operation on Q gives one result that is an "Experimental result"). The set of such results uniq. Q_i can characterized, disc Q_i's suggest solns. ~~Investigation~~ Thru analysis of corpus of experimentality (& solns) of other Q_i's — TM will find that certain Experiments ("OBS") are more useful than others. Also any trials that TM does on Q_i in attempts to get Q_i, will constitute an "experiment" ··· a "Measurement" of Q_i.

TM may want to devise a set of operators that is used on all Q's, to "Categorize" them: Could PPM help here? The operators become "OBS". Actually, these obs are stochastic R fcns.

17 → [BN] A "not bad" way to do ~~d~~ perhaps induction:
looks v.g.(
Get say 3 bunches of problems, each is solved w. its own special Oj. We then try to find a s or d operator to decide which of the Oj's should be used w. a new Q. It's operator — we try to get ncosts as small as poss! → .29

30 [BN] Main Path! What is main path now? — what are apparent troubles, difficulties, obstacles?
Consider ex. 28 ! MTM w. Alg-TSQ!
199.sheff
Initial problems: (1) Choice, design of rep. in which to express problems, solns.
(2) Design of TSQ. (3) Find, use various techniques to speed up srch: Lsrch is easy, PMM is another.

.29 (.31) → A "new" how to do MTM: from(.17-.21) Start by finding individual Lsrch solns to $Q_i \to A_i$. Then try to find solns to pairs, using PPM on compositions. Thus for pair solns are desired, given them more weight. Then look for triplet solns. (& also continues search for pair solns.). At any time, we will have ≤ 5 stochastic solns, to various Corpus; composed of a p.d. on Operators on singlets, pairs, triplets etc.

One important Goal now is to develop a good P.D. to guide Lsrch.

STM

T. forgg. looks like BU srch, rather than Search! I'm a bit vague on just how far I got in BU srch. I did recently do a kind of "review".

A major problem in BU srch: 195.31: How to deal w. trials that take too long" 196.01-.06 is a possl. way — but I certainly don't understand it. Well enuf to know if it's any good.

Also, what CB to use when we have Continuous params to adjust (196.17 ff)

My work on this was been "mixed". We can do a simple Search over discrete models — each of which has an assoc. pc. In f. foreseeable future, pc, we get as good a PC as we can in that time. As T→2T we will get a better pc "next round". After setting discrete model, the amt of pc↑ we get for a given cc, ↓ rapidly as cc↑ ("law of diminishing returns"). We also may want to use small sub-corpus (ssss) so that finding & pc of PC is easier.

Here, the thinking of Search over discrete space of Model's: T. next step pc — the continuous part, is regarded as an optimization problem w/ a CB of (.06-.07)

.11-.13 is reasonable, but how it fits in w/ the variation of sss of .10, is unclear. (.11-.13) may fine is well developed for fixed corpus. — Perhaps regard Library corpus as a "TSQ"?!

SN T. forg. treatment of Models w. discrete & Continuous params — well [illegible]. T. "continuous params" are very "square" Physics. We really should be doing models of continuous phenomena in which the whole model is digital. — I think this is more General than the usual optim. of a finite no. of continuous params. — which is ≈ the "2 part model" of MDL/MML. What I want is ≈ what Wolfram is pushing. — Paradigm of models of Continuous phenomena.

(.04-.23) is on continuous Params.

The (more?) immediate problem is .02: 196.01-.06 may be not so bad. Use CB for the time being. It is applicable to BU srch in general — G.P. (for BU srch) as modified by using a Grammar or PPM to "summarize" the corpus. — (196.01-.06) is perhaps applicable to any form of BU srch.

Sometime in Recent Past, I had that it would be feasible to do Search & BU srch "simultly" w. shared "Guiding PDs"!

I don't remember how that was supposed to work, — it would seem that they would have distinct "corpi" (??) — Check this! (Tho even if they have their own corpi & own "concepts", they can be run Times shared so cc ≤ 2×cc at most.

STM

β : 24:30

$\frac{m}{n} \approx \sqrt{2}$

$\frac{m+2n}{m+n}$ is better ?

$\frac{\frac{m}{n}+2}{\frac{m}{n}+1} = \frac{\sqrt{2}+2}{\sqrt{2}+1} = \sqrt{2}$?

~~Zeno eff.~~ is a Int. heard (S ± BU or Lsrch) or not/Rec?!

SN Look into ① Laplace eq. Analog Computers — look on web. ② Rosa Peter: on web: Also 1st Recurs. Guy who suggested her. ③ would Munchs have any info on Recs? Get his book from library! (c/o Buy).

SN **On Languages** to use for QATM: I had (wildly) considered Machine lang. — But now I may be in better state to understand how much I need main inst set for universality! Also if PPM will help, & perhaps how to augment PPM to make it work better in this case.

The conditional jump could give "do ... until" (which also gives "do for ever loop"). We normally have flags to store conditional results. for functions use + − mul div — perhaps integer & float, if its same speed as integer. Divide by Zero can terminate any Rec, & most computers have an automatic "divide by zero" detector.

So look at set of 23 insts in "Inner loops": see how many would be adequate. OOPS has 23 insts! (.23)

An Early idea was to introduce new insts gradually, so that TM could learn/use to use new insts incrementally. Introduction of new (symbols) insts into PPM usually gives Rec very low prob. ($\frac{1}{n}$, where n is # no. of Symbols Rec for).

Try to draw up Early QATM as much detail as poss/ to detect poss/ Bugs, diffys ....

One poss/ is to start w. the 26 basic insts (in "Inner loops" p 44). Get TM to solve a bunch of simple problems (or trainer solves Rec & puts Rec in as Corpus for P.D. to Guide Srch) — which become corpus for P.P. for Guided Srch. What this does is effectively ↓ no. of insts, yet enables TM to use all of Rec! It may be that we will need a very large corpus before the P.P. is good enough to help solve problems of mild diffty (or even any problems!)

T. Loops, etc equiv. to a hand wiggling inst. c/o devising "Macro insts". { The "26" is a small no, & any/all for these insts have much info in Rec }

There are 8 registers (7 + Instruction Pointer). I imagine all comput. done in Primary Cache (208.08)

.06 — .30 : a .16 th part order! Suggests that 5 mins start on any (fast) Conv. universal lang & develop & equiv of Macros (via PPM) | via Symbols [SQ a/b software problems]

so that it is equiv. to a lang. designed by designer of TM! This is (.29)

STM

So, consider the various ways to get a good lang. w/o a [redacted] weighting of insts on a large universal lang, so that the resultant pd on cond. Pgms is good enough to solve interesting problems in a reasonable time. Wkg. 20 8.00

Desired: AIS: One "kitchen sink" approach: Put very large set of problem solns into a corpus & devise a grammar w/o a PPM P.D. or so w/o some other P.D.s (perhaps in II) to characterize the "set of solns". .09

Here are regard the problems as "Inv" probs., & the resultant P.D. as being appropriate for such. (Later we must plan only given probs w. solns of accessable Kcost.) — or, use BUSeek (version 195.20 on BUSeek — 197.34)

.05 The "B16" troubles (.03-.05) is that the P.D. we obtain over cands Pgms may is not universal: ex. the search we use to obtain these P.D. does not consider all poss. Partial C (or even primitive) recursive funcs. It may be good enough to work many problems, but ultimately, it will be inadequate. We can probly use it, but to find universal S-funcs that are better than those used in "Am Mus Proj" adequately — those S-funcs can be universal. .29

SN On the "unique Original Univ": Q: Given any pair of universal machines M_1, M_2: & assoc PDs P_{M_1}, P_{M_2}. Does there always exist a string s such that if P_{M_1} is given s to continue, it will get P_{M_2}?

We know that for any M_1, s, & a new M_2 exists so that if s is given to M_1 the resultant P.D. would be a certain Univ. (Actually, since M_2, s induces a P.D. on continuations of s, & all P.D.s after all Epms have a machine assoc. w. them, & any non-Epms, but does all pms have machines assoc w. them? probably not all but all finitely describable pms may (perhaps) have machines assoc. w. them.

This process must imply a Machine I in my "2 kinds of Probabilistic Problems" I gave a constructive proof.

On the conjecture of .16: 2 possibly helpful ideas: ① M_1 & M_2 can simulate one another in similar ways ② P. construction & the non-constructible Proofs of .20-.23 for an inverse of the thm. to be proved.

Maybe send in to "red" list ☺.

.14 If the probs of .06 were not all inv. probs but some S-produ. problems (which can be formulated as inv probs (☺)). (check this!) Then we could get the system to devise new S-problem solns for the TM_2 problem.

5TM

Aside re "Break Pro": In the preceding kitchen sink (205.03) approach:
~~If~~ all Q_i's are distinct we have the "Recognition Problem", after we run the all $Q_i \to A_i$ pgms. Thus the "Recog" problem is well known & its soln. is well known & (relatively) CHEAP cc. SORT Q_i's!
If some are the same, we have a probabilistic D.F. on A_i's.
Since sorting is standard, it can have the ϕ cost!

The reason I'm using "Break Pro" is that it is a trivial observation, it does require much of my prof in this area!

A twist on this idea: If no. of Q_i's is very large, we may want to use Huffman coding of Q_i's. There is a trade off between amt. of RAM use & "find time" of the Alg. This has probly been investigated in Dasgupta Papadimitriou Vazirani's books will have good discussion. — Or look in "Num. Rec". or other books on Algs.

Anyway, all this means that I have to revise much of my past thinking in this area. This actually goes back many many yrs of error!
But my most recent run in was in K.S. of 205.03 & recent work in that area. One pt. was my (relatively recent) discovery that the algs used in K.S. were (at first run) not universal!

impt. → Just exactly what is the cost of this sorting soln. of K.S.?
Do we have to pay for the list of Q_i's?

Well, if we are comparing different $\{Q_i : Q_i \to A_i\}_{i=1}^{n}$ algs sets, the cost $[Q_i] = "1"$ since it is same for all $Q \in [Q_i]$ sets. If we are applying to a (say) single O^J that does the whole set of $Q_i \to A_i$'s, Recog. cost (perhaps) is the total pc of the self delim. codes of all of the Q_i's. This is the problem of min. coding of a set of unordered finite strings "à la Kinds of Probabilistic Induction problems".

So consider both. Code for a QA corpus:

(1) Code for $[Q_i]_1^n$ as unordered finite strings using truly universal Grammar or "lang".

(2) Set of codes for $Q_i \to A_i$ using universal model(s), but using say a non-universal TM (say PPM) to find the O_i's. So we have a code for $[O_i]_1^n$, but it does not use recursive functs (or it does but not "fully"). → (see 207.00)

(3) The min needed to code the nxn selection $Q_i \to O_j$. — This is the no. of ways to map an n vector onto an n vector. is it $n!$?

We compare this w.r.t. cost of the [single operator] O^J that does $Q_i \to Q_j$ for all n QA's.

5-TM [redacted] ports and CSQuest 625 Mass Ave: 3PM 617 876 5550 [redacted] Darren

Modify ot [redacted] 206.28: say $[O_i]$ was coded by Universal grammar/lang like $[Q_i]$ was. 206.27

In ~~206.28~~ 206.28 we could discover [redacted] O_i

first, by any means, even optimally coded $[O_i]$: Qu find $\forall$. Codes O_i i

simultly optimize $\forall$. search for min code via a universal (Grammar/d of.),

The second way could be v.g. The not as good as (206.33) (?!) .023 (?-note .05)

The 206.33 is a sub. case of .023-.03 By making several of O_i identical, .056

& cost of search codes (206.31) can be much reduced improved

Also, We can do .023-.03 as a leading (w TSQ) aspect to 206.33 ← a "final goal". "train"

Perhaps we can "load up" $\forall$. PPM into state we into from various Much Long. Systems.

A direction (course) must be $TM_1 = TM_2$. QA TM can solve problems of time Series, of which TM_2's data is an Example. → .21

[SN] An easier idea was to first do $\forall$. QA's as individuals, then pairs, triples, etc. $\forall$. recent analysis gives a way to understand .056

these as legit codes & how long Pair terms are — gives a good Quantitative Understanding. My (strong) Impression is that it ends up w. a legit, "Universal type" induction Operator (even when it doesn't have a single O^i operator for all 4 QAs.)

.13ff may be able to deal w. non-Turing solvable TSQ's. (whether it would actually work w. "BU Turing" type of probs in General, is unclear.) → 208.21

So .13 ff may be an adequate schema to get a very powerful Phase I.

Then $\forall$ Q is, how to use .12ff to work on TM_2 - type problems? —

Well first we restrict on problems of which TM_2 is a (Perhaps Advanced) example.

Super literally: TM_2 type probs are "so" given $\left[O^i \left[Q_n A_k \right]_{k=1}^{n_i} \right]_{i=1}^{m}$

a set of O^i's Pair assoc. Corpi — a $\forall$. Guess (= PC of a corpus w/ its operator)

Given a new Corpus of QA's to give a good PD over possible O^i's for it.

[Note: at this pt. TM still doesn't have a v.g. idea as to what "Optimum" means]

Problems like .20 can be regarded as QA problems: Q is a Corpus, A is an O^i

for training we either use TM's own seq. of (QA subcorpus; assoc O^i) pairs,

a/o a corpus invented by trainer a/o a corpi from some other TM.

Or a simpler corpus of many Q_i ("Q") (Q_i, A_i) pairs in this TM's hit type ("A")

5 TM

So Main Probls in TM now:

(1) Decide on / develope suitable lang. to express probs/solns in QATM. (Note recent ideas on Mach. Ling: 204.06)

(2) Develop details of Systematic 207.13 (single (QA) then pairs then triples ...) first write a detailed descr. (Note 208.21 ff) — Consider EPGA

(3) Design TSQ for QATM.

(4) Run ~~TM~~ QATM to see what works at all

(5) Develope $TM_2 = TM_1$ (207.21)

Write detailed review of each of these Problems

08 ;204.31 One good way to study the problem of designing an Mach Lang. language: First consider, not full partial recursive funct descr; but just Compositions — (Like FFT & NN, fourier expansions, Taylor Series etc.)

.10 What is way to use Assembly lang so that desired functions are easy to express & Most randomly generated fcns are meaningful.

T. Prepare an Assembly Codes for G.A. (Adv. in GP Kinnear: PP311 ff.' Nordin)

Use only w.o. non-loops: Very fast.

Hvr. it may be best for an "Advanced Machine", only a few trials are searched. (sands tried)

→ To get to that pt., hvr, may require very long srches over ~~reasonable~~ a large ~~reasonable set of probems~~

Event. "Advanced Machine" will sometimes have to do very long srches on diff probs.

.20

.21 SN More on 207.13: This looks very good! If we get a new problem added to the TSQ, we continue to use the old solns w. singlets, pairs, triplets, ect., ~~but we try to first solve the new problem by itself — Then~~ we first try to use the comparison techniques on the Q_i that map the Q_i probabilistically into the various "successful" O^j's. If this doesn't work we try to find a single soln. to the new problem alone — Then try to find common solns for it & one or more of the old problems.

Note that this is a "stochastic R" system as opposed to the deterministic R system that was used in the ERSIM Report (PP 8-12 see p12, consist of 5 118 for Maximum of stack R's)

.30 These "R's can probly be obtained as a soln. to a QA problem.

These s-R's are used to the reason of telling which O^j's to use w. which Q_i's.

.33 I think that one of the very nice things about the 207.13 approach, is using s-R's, is that when we can't find a O^j that fits a (the) new problem as well as

STM

all of the odd problems, we don't have to discard all of the old solns & try to find one that "fits all". We can get along, imperfectly with a few solns. That fit the new problem & (none or) several old ones.... Tho it is certainly better to get a single set of Recipes for all problems.

I need to work out the details of the S.R system:

205.03 ff (the R.S. method) may not be so fast. — specifically 206.00 ff

I need to understand how 207.056 -06 works (if it does!@).

Seems that it is mainly 207.00 ff that has to be worked out in detail.

207.056 -06 is an important part of it.

T. "BreakPnt" of 206.00: I thot that I'd obtained a good way to put "corrections" on unordered set codes. In Sequencial data, this is easy to do ... but not so easy for unordered data. My impression of the scheme of 206.00 ff is that although it does a code for corpus, it is not much use for prediction!

The idea of Coding $[Q_i]$ could be a way to code for entire $[Q_i, A_i]$ corpus — which is a "kind of way to do QA induction". The code for the $[Q_i]$ can often contain Elements useful in the $Q_i \to A_i$ mapping.

After we have found R_i codes for each $Q_i \xrightarrow[R_i]{} A_i$, we note that the D.F. on A_i is simply that induced by the original Unc. We can do prediction for a new Q_{n+1}. But it will be dependent on Unc only, & indep of the previous $[Q_i, A_i]_1^n$. Suppose, now, we have many codes of pairs of $Q_i \to A_i$.

To code for entire $[Q_i, A_i]$ corpus using .14 -.20: first code the Q_i, then, for each Q_i use R_i to code for assoc A_i. If a new Q_{n+1} is added to the corpus we code it using the ("grammar") that finds R_{n+1}'s to generate a D.F. on A_{n+1}'s.

This latter is relatively indep of original corpus.

Now say we have O_{ij} — which is a prefix for input to Unc that Recovers

$Unc(O_{2j}, Q_i, R_i) \to A_i$.

$Unc(O_{ij}, Q_j, R_j) \to A_j$.

Somehow, we want to use these to code the entire corpus "Economically".

Each $Q_k \to A_k$ will have several different codes in it.

To code the corpus: first, as before, Code $[Q_i]$. To code $Q_1 \to A_1$, say, which has several $O_{1,k}$'s that can be used to code it ... first we have to select the $O_{1,k}$ from the set of all $O_{j,k}$'s we have found. So we have a d- or $\leq$ function that does the selection. Since we have several codes for many of the $Q \to A$'s that must be.

Hoscat Wine

800 Organ
344 Wine
9463 Made
Kathy = Stacy

311.00

N11.05

2/10

STM

Random Notes on browsing

1) Abstracts in GenProg 1994 #1

2) 1996 #2

3) 1999 #3

4) Ezra Norton Hall of Fame 1998

(1) pp311 ff

(3) pp 275-ff also see 4) pp330 -338 } Dcrn of Machine Code used for "GP"

(2) 221 ff on evolving recursive solns to problems: p 240 Olsson. SJ; Summers77 (JACM)

221ff Generalizes CFG's to be strong enuf to do RPS (!?)

(2) pp177-ff "Discovery of Subroutines in GP". This seems to try to implement intuitive utility of a subroutine ... I don't know if its any good, but it may be suggestive.

See: .05 : Ray how Commercial product "Discipulus" for Tutorial on "Lite version ($200)" see N.12.05 pdf files.

I have several Discipulus-type files on N.12.05

T. website http://www.aimlearning.com/ or Ask Google for "Discipulus"

The "Free trial" version just does runs -- does it show what was done, or what functions were used. — So would seem to be of little interest/value.

I do have the manual for the "Lite" version: 128 pp only T. manual for advanced version is online, but I was unable to download it. Windows kept Crashing — It seems to partly load as PDF. — but had no access text. Sometimes I can/cannot it online but not download it.

Perhaps my trial version on SM? c/Documents/walter/Local~1/Temp

c/Documents/walter/local~1/Temp.

c/Documents & Settings/walter/Local Settings/Temp/Discipulus/Discipulus.exe 1.3M

Contains Owners Manual of "Disc..." exe (1.3M) 210 pp!

The Owner's Manual for the trial version has 210 pp → it may be source of

Manual for advanced version: It sort of skips introduction. pp159-175

It has "Discipulus Owners Manual, version 3.0 DRAFT

Normally Discipulus can have only 2 fitness function types. Regression & Classification. for extra cost! "custom fitness function Module."

In the "Demo" version it may be possible to change "discipulus.ini" params so it will work like the paid versions. Look at many other versions.

Lite $195 will be regression & classification only.

Desktop Professional 495 " " " " " "

Enterprise 695 Control over fitness function

Enterprise for web 1095 will also run on 2 cpus

Parallel Module: for any # cpus in all. i.e. 72 will. 195 add on to Enterprize version

Once may need 1 "11 module" for each new CPU!

This version for Pros, Enterprize versions only.

I see

Says: Version 3.0 Draft — so probably identical

Hurray, it is Demo version! Only exception of "Custom fitness function" is "Mean relative Error".dll.

STM 7 21.0.92

Spec 209.34: 209.25 – .34 will pretty work. Th. Selector funct of 209.33 – 34 is Minsky of an a priori approach I worked on w Time Bnds at IDSIA — I.G. A problem remains is a selector funct decides which approach to use to solve it. This is w.r.t. way Phase II operates) — so maybe a nice way to "ease" into Phase II.

A V.G. feature of Pr.'s approach is 208.33 ff: And the idea of 207.13 is a nice way of coding the corpus! We get a large ssz of $O_{i,j,k}$... operators to put into TM2 (e.g. PPM (or GP?)).

I think of Pr.'s function of 209.33 – 34 as being something that QATM (or other) can discover, or user (PMart) "usual non-recursive" suggests to find the selection function. I have to work out the flow of how TM does Pr.'s.

Actually, the selection funct plus all the $O_{i,\ldots}$'s are exactly rigorously equivalent to a single O^j or select O^j's, because they both perform the same function — i.e. $Q_i \to A_j$ pc assignments.

Any other further work on TM can be regarded as "simplifying" (i.e. finding pc of j & coding them) the O^j's.

One problem I have to clear up is just how TM computes / approximates the "selection functions".

I think many Machine Learning Algms could do an approx Selection Function" — ANN, PPM, SVM, GP, Decision trees?

Also when is the "sel. funct" updated? — How much CC is allowed?

My impression is that 207.13 is a reasonable way to code the corpus, & that it should be possible ("natural") to find easy ways to Legitimatize it.

Are there some "Default" Methods of doing the "sel. funct"? — Perhaps not so good, but good enough to have "progress"?

SV Kaz: That almost all time was spent on fitness funct. So basically wouldn't help! — But why so much time? If data is in I/O form of "pairs", checking shouldn't be much time! Use small sample, Then use as error. (Also, use other approaches: maybe a way when error is small!)

For problems in which fitness-eval is very slow (Electronic Ckt simulation) Does anyone's wouldn't have one value. — Another track! If fitness takes long time, we should use some to find V.G. conds! This is in line w. Goal of ALP "To exploit Computation & pc in Available time" ("Available time" = Time for Fitness Eval) = "Fitness funct".

For SM I can do it as a class binary classifn. problem or as a regression problem. In classifn, if I penalize error much, I will get many "hits" but small amount. — Not bad actually,

5TM

So: Present State of affairs:

208.00: Lists problems, stages of QATM

1) Re: $M: It may well be that my latest ppm to see if it's likely that the μ of a driver is >>0 is about as good as it can be! Quick & Easy course of action would be to take the drivers w. $\chi(6\sigma)^{-1}$ pc of Resp >0, & try to see if I can get a "good call out of the system as a whole"

I should, here, do tests on random drivers to check that my final eqn is correct. It is an easy eqn. to check. Also perhaps use good approx for erf for $\xi x/\xi$ 2.5 and check my ppm against tables.

See Computables! Downloaded zors files in this Name! Dec 7, 2005 / one PDF file.

Q: Can I get good, logit d.f. of μ of "top 10" drivers?

"NIH problem" Needle In Haystack: ~ 10 m 6 ft rn Sch. Not Invented Here / Not just Heath

Needham Haystack.

2) Re: Use of ~ Descipulus (210.10ff) ideas in designing Logs for QATM.

I could just look at the set of Mach Log insts that they use, then mixed w ideas from "inner loops" get a good instr set. Then try psmy coins of ANL's elementary Algebra. Is there a good way to give extra wt. to TSQ problems so ppmM would "learn faster"?

Other idea is to filter out log simple TSQ to "initialize" ppm or put in hand psmd (or partly hand psmd) examples to "initialize" ppm

Would it be good idea to just start psmy a system's data TSQ w.o. [redacted] detailed ideas on how TM is expected to [redacted] in each case.

Another problem: I'm really not sure about just how TM is supposed to do BU srch .... how it deals w. excessively long trials. The ideas of 210.15ff to ~ 211.24 seems rel., but I'm not sure of relevance to BU srch. (196.01–06) is a start at a soln that may work.

208.00–06 is a list of main pts (phases) of TM work.

color pairs, triplets, triplets .... of couples.

Assuming Long choice (like .10–.16) is ok.; then main Research problems are ① development of 207.13 .... ~ 211.24 ② Development or (discarding?) of 196.01–.06 as soln to BU Srch (Akin of "backward" Lsrch) → Absolute 213.00

③ Just how to Do $TM_1 = TM_2$. What is Corpus? How can I fit it into QATM? (T. corpus of 207.13ff is large & ∴ effective — but I have to work out theory of just how it works!)

NB [redacted] Is $TM_1 = TM_2$ a BU srch problem? Seems likely!

So 3 main problems .23, .22, .25

It would be possl. to start psmy on soln. to ① only.

In ① is Res related to: we have an event that is in a "indep" corpi. How do we predict it? This last would seem to be: Create corpus that is sum of 2 "indep" corpi & code it.

MPC ; RLBU ; TM12

(.06) (.15) (.17)

5TM

10 — 1.96.06 In BUsrch: @ CB should be ac Time available for each CE construct

(b) Probty ac pc of trial. : Reasons constraints may be enuf for most hill climbing CE Mutation + S-Grammar) methods.

We assume that there are many acceptable solns: Otherwise this search would take too long. Another possy is that Grammar would even help us?

My pc to solve: — Non-findable by (L)srch)

06 Def. → Name 3 probs: (1) Multiple Partial Codes: MPC problem.

(1) 2.12.23: Hm: Actually, the categorizer need not be done very well! (For standard McEliece Long Algms would probly be fine!). The reason is that this is just a method to gauge "hill ht." — But ultimately we will get just one Category & the categorizer will be unnecy. (Tho in some: for some QA's it may give significantly lower PC than the single code for the corpus, by itself.) — Also, when the "single code" has to be revised, the Multiple partial codes can be very useful.

(2) 2.12.23: Resource limited BU problem. RLBU ? Nobo £13.00

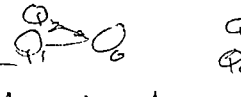
Perhaps first Consider/methods in which all trials take same time.

(3) $TM_1 = TM_2$ Say "TM12" works

MPC problem: Consider using PPM for the "classifier"

The classifier problem in MPC is essentially a QA problem.

$Q_1 \Rightarrow O_0 \rightarrow A_1$; $Q_2 \rightarrow O_0 \rightarrow A_2$. $Q_3 \rightarrow O_1 \rightarrow A_3$; $Q_4 \rightarrow O_1 \rightarrow A_4$.



A Different Approach (perhaps a variation of an old idea!) That each O_i can code the entire corpus but is much better for certain inputs than others. The bad outputs are w.r.t default outputs of the unmodified Reference use. So the code is simply all those O_i in 11 w.r.t or Prior PC for the TM's PC each assigns to the entire corpus.

So, this is then Identical to just looking for a good single O_i for the entire corpus! Delyght Looking at it this (2 ff) way simply makes it more (clear) that it will work! Essentially, we'll do a GP-like search for a O_i that will test all of the corpus up to now. — But we do this search by starting w. corpus of 1, then 2 then 3 etc.

5TM

[redacted] add in the O^j's from Q_1 add in Q_2 - - - - . At each point we'll have previous corpus of O^j's & their wts = that PC from assigned to corpus.

Trouble w. the wt. assignment: O^j that did whole corpus well, will get less wt than one that did only a subset corpus. One way to deal w. this: Each old [redacted] O^j for some older, shorter, corpus, is "updated" on the entire (latest) corpus & its wt. is = the PC it assigns to that entire corpus.

Trouble w. that is that this may give too little wt. to O^j's designed for short corpi (i.e. small "n"). — such O^j's would tend to have very useful concs in them, yet would be given very little wt.

We could multiply wts by a constant that was a function of no. of problems the O^j solved.

Now, even if this fancy works O.K.! There is the problem of: Do we need to check all past cases when a modified or a "successful for all" O^j is proposed. In the "Sc community" only the "part" of the general prob. alg. that is "relevant" to the new data, is tested on old as well as new data. The "d-R" system of IPS (A report (30 Oct 2003) PP & no R does deal w. this problem, but it is a d-R system, which, presumably, would not help in MPC discovery (213.06).

In general, given any O^j alg. — any (specific modification change) alg. $O^j \to \tilde{O}^j$ produces new corpus of examples like those: those affected by exchange, & those not affected by the change. So we'd like our O^j's to be "friable" (= Breakupable) in this way. .20-.22 a special kind of "friability"

A special case is afforded by the "d-R" system of (.17), where O^j is broken into parts that operate on nonoverlapping sets of Q's.

For usable friability, the d-R classes could overlap some:

This would simply ↑ the amount of "checking old corpus examples" that one had to do. (α)(β) ← no overlap (α∩β) overlap means that region of overlap has to be checked if α or β is modified.

Otherwise in [redacted] if α is modified no member of β need be examined.

If overlap occurs, this gives classic problem for predn. in overlap region. (Examples are in a overlapping sub corpus) — the problem is how much wt. to give to each of the data & models for predn?

A try is a possl. sort of ↑: (α (α∩β) β) . We do a

(α∩β)

5TM

codes in ||: One for $\alpha-(\alpha\cap\beta)$ (part of α not in β) and $\beta-(\alpha\cap\beta)$ (part of β not in α).

$\alpha-(\alpha\cap\beta)=\alpha\cap\bar\beta$

$\beta-(\alpha\cap\beta)=\beta\cap\bar\alpha$.

If both α & β have assoc recognizers; this is a legit code (I think).

Say R_α is a d-Recognizer for the α set of Q's.

Then if R_α & R_β are known, & classes $\alpha\cap\bar\beta$ & $\bar\alpha\cap\beta$ are well defined we. are extra info. We can then codethese sets as individ corp.: Here I presume coding of an unordered set of ("finite") objects. → start .123

Can we define $\alpha\cap\beta$ say, for s-Recogn fcns?

Say $P^{(Q)}_\alpha$ & $P_\beta(Q)$ are the pc's assoc w. α & β imposed on the classes, Q. (stochastic)

Then perhaps $\alpha\to P_\alpha(Q)$: $\alpha\cap\bar\beta \to P_\alpha(Q)\cdot(1-P_\beta(Q))$? $= P_\alpha(Q)-P_\alpha(Q)\cdot P_\beta(Q)$.

This assumes indep pc's for α & β — which may be legit, since we assume α & β given indep (addable) codes. (woops! Indep probys are not addable!

only Mut exclusive events probys are addable. T. codes for any corpus are mut exclusive.

The O^j codes are mut exclusive.

Say each O^j induces a P.D. on all poss Q's (as poss clients). We then assign each Q for O^j that pair with pc from it. ("Pandemonium").

(NO!) Suppose we use O^j directly on Q's to produce a P.D. We then do this for all O^j's on each Q_i, then use ....

Each O^j induces a P.D. on any Q_i for each poss A_i. We could then, for each Q_i, use the pc of a particular A_i that for all O^j's was max.

This is ≡ using all O^j's in ||; selecting max(over O^j's) pc for each poss A_i, then normalizing. The "Max" selection is used to get Mut exclusivity.

2 P.D.'s can be either indep or Mut exclusive — to get simple combination rules

(SN) **Re: 21.34:** If there are N_i O^j's for the i^{th} Q_i:

Then the no. of mut excl. || codes is just $\prod N_i$.

So the problem for "S-R's" (technique) is not at.

Re: the Problem of 21.13 + .19! Actually, not exactly true! What we do normally is classify phenomena so that only certain parts of the corpus need be checked: E.g. in physics, when we modify the "law of gravity" we don't check to see that all chemical reactions are affected by it.

However, we may discover old phenomena that the new gravity law applies to. More likely, we will try to get new data to test the modified Gravity law.

We do tend to classify phenomena rather well, so that we rarely find that a modification of a law need be checked in many cases.

However, perhaps I'm thinking too much of "Mature Science", in which

Expo .20

5TM

100M ¼W / 10 m centers 5kV $\frac{e^2}{R} = \frac{1}{4}$ $e^2 = \frac{10^8}{4}$ $e = \frac{10^4}{2} = 5kV$.

The classical of phenomena has been very effective in certain areas.

In case of social actions/experiments: it's difficult to be sure that certain Rings don't affect other's effects. (cross products)

Also Medical Rings: Much interaction of Drugs w/ surgery, etc.

T. Torp. seems to be related to 214.13-17-19. Is it really the same Ring?

An apparently analogous Ring: Pill α cures one disease β, but we have to check to see if it has no unwanted "side effects". T. apparent reply is "α cures β" — tho we are fairly sure α will increase hurricanes in Brazil, it may give t. patient hypertension.

So t. correspondence is between a varying Poncelet operator & varying an action on R.W. : In case of operator: we observe a variation produces a certain (desired) effect — but we are unsure of other effects. In case of action in R.W. we observe certain desired, but we are unsure of other (possibly undesirable) effects. Modern treatment of β.

↔ Modern of General/production. (Modern of "part" of production"

So t. correspondence does seem close.

→ Actually, the production model is a special case of Modeling in R.W. So RW solution is more general.

Developed t. pill/disease analogy + May suggest ways to deal w. problems in TM.

SN Expo: Names for Talk: Russel, [redacted], Wallace, Rissanen V.S. Gödel, Turing, Church, Solomonoff (A Barroom Brawl). TM. Powerful semi Universal Induction" Ideal of Russel's Theory of types" was developed as a system designed to automatically avoid certain kinds of "statements (barber shaves all men in town who don't shave themselves ... did he shave himself?)

Later, it was shown that Russ's formalism excluded certain kinds of reasonable statements that one would certainly want to make.

Gödel & Turing showed that if you want to include all reasonable statements, then there is no finite system that will enable one to decide between reasonable & unreasonable. [see Kleene Book for Brief historical discussion] Intro § 60

More recently, Wallace and Rissanen have proposed systems of induction that include only computable probability measures. From the Gödel-Turing-(Church) results, this must always exclude some useful computable probabilities measures. Solip avoids this difficulty by defining probability as "well defined path of second kind" (i.e. a resource limited optimization problem)

N.B. in SVN folder on ACER: (SZ 035. BAS 12.1.1998!) ← was this to check my formula for $\frac{1}{12\gamma}$ precision! Also how Lewis's approach. Variation (CNS!

5TM

Spec 216.34 Hvr, Riss may say "when I speak of an induction Algm — I mean a computable procedure."

In this sense then, ① Any of Riss "computable procedures" ~~must be inadequate~~ is very probably "inadequate" because there are usually better ones.

Expressing a Soln of ~~st. induction~~ problem as a Rec. ltd Opt'n problem

Expands the meaning of "computable procedure" — ~~to be a program~~ by adding an extra param to the problem — eg. the CB.

Mention situation in which CB can vary — can be ∞ known — i.e. in GP/GA in which fitness func'n is expensive — #evals of problem can have a CB ∝ R. Comput'n cost of the fitness function.

So: 2 applics of d-R System: ① ~~#~~ Less need to check all poss cases.

② the MPC uses d-R (or S-R!) || Also note analogy to "side-effect" problem in "Medicine:"

I may want to Reread & IDSia Reports pp 8-12 on "R-funcs"

One way to arrange so that R & Rj / almost never overlap: The R ~~are~~ start out as) real funcs on Qi. The Rij (largest value is Rmax) be R assoc w. a Qi.

My Guess for S-R's, I'd want Rmax, statistically indep of each other, rather than mutually exclusive. If they are mutually exclusive, this is d-R!

SN on "TM" sch: Time spent generating a cand should be ∝ time for fitness func'n. Thus no explosive relative ↑ in pc for TM sch to be useful. Does this?

So: Will locality for common sub-trees be "cost effective"? — it may be better to settle for a "raw" PMM, since it's so modifiable. If fitness function is enormous (like SPICE in Koza's Pgms) it should be possible to usefully ↑ cc of TM sch.

In trying for good "cuttings" of a parent cand, — we could just do a random node on the entire corpus and use the entire branch. This is ∝ like the usual "crossover" (Koza).

One way to do this: Use an xwindow for ordering population — cutoff ∝ $\nu\lambda$, say. Choose 2 ~~pairs of~~ random nodes (using $e^{-\lambda}$ distribn over population) generate 2 children. Test each ∈ ~~pair~~ sort them on fitness. — discard 2 ~~worst~~ bottom cands if they are worse than the 2 new cands.

λ can be chosen for best $\frac{\sigma}{\text{Max} - \mu}$. (? Is this possl.?)

Using Prem. a/o Perf. rec. funcs can ↑ fitness testing time a lot.

SM .03

5TM

Note that in Lsrch "cc$_i$ of a trial" is (generation time + fitness eval'n time) plus time spent selecting a trial). If for recursive funcs, "generation time" can be arbitrarily long for even small codes (& large apmt)

.03 In SM Lrch, I can afford v. complex recursive functions, when fitness eval'n is expensive (can easily be many sec on a large corpus).

Usually I will start w. small sample of corpus. Then mildly good codes with have mutations of hy appl to a large sample of corpus.

07 My impression is that "complex" recursive (hybrid) funcs has not been much tried in SM Lrch. Maybe not that simple (hy appl) but longish cc due to recursion — have probly not been tried.

For SM Lrch hy w LGP: I use trials in "cc" order.

We start by listing trials in pc order for discrete and "continuous" part. T "continuous part" is actually discrete, b.c. no. of bits used & simply means we do a smaller grid etc. We then try each on a subset of corpus. For very short codes, we use only small SSZ; but SSZ ↑ as we begin to do longer codes. As a result, both cc (& SSZ) ↑ as code length ↑ as we do T ← ST etc. When we do T ← ST; I think we want to CB to be by pc$_i$ i.e. $\frac{cc_i}{pc_i} \rightarrow$ "X3" We have to decide how much to ↑ cc$_i$ & how much to ↓ pc$_i$ for each run. Should we do both by same fac?

perhaps just to T ← ST & use cc$_i$ ← T pc$_i$. cc$_i$ will grow as "T" ↑. & SSZ will grow w. "T" Hmm, if we use recursive funcs, as pc$_i$ ↓ we get longer pgms & we get ≥ kinds of cc ↑ ① ↑ conds to be longer to generate & we also have ↑ to test ↑ cc$_i$ ↑ because SSZ ↑.

As we do T ← ST, we could fix SSZ for each round, because Testing then becomes ≤ a "≤ constant value", but generation of tests can be very variable. — Not entirely clear! ~~How do we want to bal~~ with each round, we want to ↑ both SSZ & cb for functions. — unclear how to "share" ↑ factor of 3 ↑! One could do √3 (or even √4 = 2).

→ Use of "sampled" (≈ "statistical") testing of conds w. < "all" SSZ.

→ Wow! How could "tables of commonly used functions" be devised/exploited by TM?? Is this a new kind of theory?

Q & TM: SM normally address much diff kinds of probs. .07 ff is mainly about SM type probs. Seeing just how much they are related & so

"normal" QAPTM proofs could be "instructive".

SN 2705 I had considered having a "Master" computer. Devising Cands & Putting them on a Stack; Other "slave" computers would take Cands or bunches of cands from Stack & Test them. May

Trouble is, devising Cands should take ≈ time as Testing them, So we'd have to have ≈ as many Masters as slaves! Or else we'd be allocating Jobs to various computers! (Job = devise cand &/or test Cand)

There are 2 kinds of Srch! Csrch & Bu srch

For Csrch: One possible way to divide up Srch: Say we have n cpu's: we divide up Srchspace in to n subspaces of ≈ pc $\frac{1}{n}$ each. 10

The way this is done: we first list all (partial)-codes of pc $> \frac{1}{n}$. We don't use binary because pc data is in pc (non-binary) form.

I think we can do this by constructing to completion Srch trees for all nodes w. pc's $> \frac{1}{n}$; We then divide up unsearched space into n (≈) parts & assign one to each cpu. 17

In the first approx, each cpu keeps & runs each Srch subspace for all "rounds" of $T \in \mathbb{ST}$.

For next approx: we divide srchspace into kn subspaces (Say $k = 4$ or 8). Each round, we randomly assign the subspaces to each of the n cpu's. 20

After each round, each cpu will know the total pc of unsearched space for each subspace that it has worked on. Each new round, Each cpu is assigned a bunch of subspaces whose total unsrch pc sums to ≈ $\frac{1}{n}$.

SN N.B. At the beginning, I will be using ≈ PPM, which is fast, so cc for Generation of cands will be much less than cc of Srch, So I could use original idea of a few or a few cpu's generating Cands, — Indeed, I may rather need a valve adjustment (at least prim Recursive) TM Better Cand Generator: I can ↑ pc's of Good Cands by ↑ cc of Generating Cands!!

→ By 3 is going to ≈ "Phase II"!

Rev 220, 225, 245

List of Impt. ideas that I've written about that I may have lost.

1. Footnote on 5of89: I have scribbled on this; see also (1.15.99) 2 theorems related to Int. FN: BBTS 23.21; 24.21 (Mem Book TM General 7.16.90

2. "If all or most items spoil subsets in P.D. then each is ± best."

"LP would better methods then we should have pot. fcts inherited P.D."

This may be exactly true if ① we don't use "Quick Abort" heuristics ② we don't use "Quick Abort" heuristics & similar heurs not based on Guided Pd. (Essentially if all heuristics are expressible as heuristics of E. avoiding P.D.

↑ T.R.S. subsumable as "all heuristics is MPD"

3. Hyp: All problems are not solvable by Lsrch! i.e. BU problems.

4. "Core Concept" problem (ITM (2001) (19, 19,26) (also see P18)

5. Dangers of RTM: ① Can't tell what it will do ② Good chance it will self-reconfigure (Junkee) & end up doing nothing. ③ Giving any very smart TM a very hard problem automatically gives a RTM

6. RTM: Actually not a good path to TM: RTMs are complex to analyze & still need TSQ design, etc. Best work on "simple TM"

(≠ RTM) — A TM of this sort can be a RTM / solve RTM's problem.

7. Big List of (32) Problems in TM (Many of them Solved by now): late 1990: "PAUL" folder

pp 216 thru 222; 247, 301 (Oct 5/90 — Oct 9/90)

8.) Hardware Model of A.I. — specially constructed Machine. very fast; Use Address passing "backw. removers Modules as Main Comput. Method.

9) MacMod(5) of putting outputs of an/ Stochastic Grammar in ∃ pc order. (≡ 15 nearby.

10) Formula for optimum time spent on TM_2 search v.s. Time for Gore evalu.

11) 3 Present Big TM Problems: 213.06, .15, .17 [P213–225.26] possible solns to MPC, RLBU; & TM_12 perhaps not appropriate in Phase I.

Multiple Partial Codes (MPC); Ressource Limited Bu Srch (RLBU); TM_{12} (≡ TM_1 ≡ TM_2)

Search 2.7.20 tracking this down.

222.06-.22

226.00

5 TMA

i.e. Prog see BU such problems. — Have in always many proofs for a given Prem.

SN Repeat of earlier idea on Nutters "Proof oriented search"; ~~Heuristic basis of~~ the actual proofs would usually be either non-existant or of excessive cost, to be of interest. But one could have "inductive guesses" that would usually work — which is quite good enough. An "inductive proof" would be like a Prem works for $n = 1, 2, 3, 4, 5$, ∴ its likely for all n.

SN Re: BU search, I say the fraction of objects sought (INV problems, say) as a function of p_c of "Guiding PD" is $f(p)$. The result is that the probability of soln. as a function of p, is $f(p)$ (!). $f(p)$ will = 0 for small p. It will = p if Prem is only (soln for Prob p: It can be $< p$ for some larger p; Depending on form of $f(p)$, ~~random~~ Simple random search (w. replacement) may be feasible.

Because normally not for INV, but this is "a" truly Problem".

Say we are doing all trials at p_{cp} for time limit = $f(p)T$ for pT,

Expected no trials to success > $\frac{1}{f(p)}$; Since $f(p)$ is unknown, we use pT as CB.

So time for "round" will be $\frac{pT}{f(p)}$ or $\frac{p}{f(p)} \cdot T$. $\frac{p}{f}$ can be >1 (or <1),

Say we do [Monte Carlo trials w. $p \cdot T$ limit on CB and we do this until time T is used up; Then use try $3T$ (or possibly $2T \cdots$ or some other value may be better).

Hm: This might be a reasonable way to do [Normal MS Carlo search]! — Use random trials w. C.B. = $p \cdot T$: Do this until total time T is used up; Then $T \leftarrow 2T$ (or $3T$?)

It is very BAD! Seems (.28) ... to 2-2-03

In fact, this is what .10 - .14 is! .13 impressive!! Also it looks like it will not help much — even if $f(p) \gg p$:

Hm, if $F(p) > p$ then (i.e. > (soln), then f. soln times of final round will be $\downarrow$ by $\frac{p}{f(p)}$. ~~How since~~ Hm, if we use $T \leftarrow 2T$ it is no big savings, because almost $\frac{1}{2}$ of total with accesses be for final round. ~~Therefore~~ If we use $T \leftarrow \alpha T$, f. total CC saved will depend on α & on $\frac{F}{p}$ of final round.

$\frac{1}{(1-\frac{1}{\alpha})} = \frac{\alpha}{\alpha-1}$ & $\alpha = \frac{\alpha}{\alpha-1}(\alpha^2 - \alpha) = \frac{\alpha^2}{\alpha-1}$. || $\frac{\alpha}{\alpha-1}$ & $\alpha \cdot \frac{F}{p}$. $\frac{F}{p}$ has to be > 1 for a soln to occur at last round.

One big trouble w. normal replacement Monte Carlo, is that Much repetition of hyp cands, & Efficy Considerably over "Non Replacement"!

To find a cand whose $p_c \geq p_0$, would take $\frac{1}{p_0}$ trials. Many (most?) trials will be on hy PC cands & will take on each times. Say we have 1 cand w. $p_c = .1$ But never converges. We will spend $\cdot 1T$ on per trial.

Say f. p_c's (known by to low) of all cands are $p_0, p_1, \cdots, p_{\infty}$.

Say we do ~~[illegible]~~ n trials; $p_i \cdot n$ will be for cand i and will take time $\leftarrow p_i \cdot n(p_i T)$ (under this).

$= p_i^2 n$. If $n = \frac{1}{f(p_i)}$ $f(p_i)$ is p_c of a soln. $\leftarrow$

$= p_i \frac{p_i}{f(p_i)}$ Say $\uparrow$ is time to first successful Grid so $\uparrow = p \cdot T$ $T = \frac{\uparrow}{p}$ = cost of soln.

$\Sigma p_i^2 T = \frac{\uparrow}{p} \cdot \Sigma p_i^2 \cdot \ll \frac{\uparrow}{p}$ (!) seems Crazy ← This is wrong

NO it is $\Sigma p_i^2 nT$ ↑ Correct!

$n = \frac{1}{p}$; $T = \frac{\uparrow}{p}$ so $\frac{\uparrow}{p^2} \cdot \Sigma p_i^2$

$= \frac{\uparrow}{p} \cdot \left(\frac{1}{p} \cdot \Sigma p_i^2 \cdot\right)$

cost very large factor (usually).

Denofs56

BUsrch ≥ possl. soln. .06 - .22.

Deno$e
i$60@AOL.com

If ∃ p_i distribution has any (large p_i's Σp_i^2 will be ~ 1 and $\frac{1}{p}$ will be an enormous factor of badness with $\frac{1}{p} = L\cos t$.

∴ I think Rule in general, if ∃ p_i's very small (which it usually is)

$\frac{1}{p} \cdot \Sigma p_i^2$ will be enormously BAD!

.04 Now, int. Case of BUsrch, Σp_i^2 will be very small — But I'm not sure.

Exact same Analysis Can be done — (Close, hrr:)

56 This "Soln" to BUsrch first introduced! 19600 also note 213.00 ← idea that cc limit of CB·pc.

It would be well to think of Busrch in t. $\frac{\alpha}{Max - \mu}$ context ... i.e. a gradually changing s-grammar language. which cc limit should possibly be cc pc of cand: Rate constant of preparedness should perhaps change as t. s-ray changes.

10 In one kind of normal Lsrch, t. constant periodically doubles. In BUsrch it probably shouldn't. Could we modify "T" (t. const of preparedness) "Adaptively"? We want T to give max ↑ of "Max" per unit time.

A small T means fewer completed trials; a very large T means occasionally very long trials, so fewer completed.

For Adaptive T, do Rum w. Given T; See how many Cands found (not nearly good cands) per unit time. Multi T by 2 or $\frac{1}{2}$ & ask same Q.

17 A perhaps better way put noise in ln T so it varies each time, (see .27-.28 for how much Noise.) Say range is $\frac{T}{2}$ to 2T or $\frac{2}{3}T$ to $\frac{3}{2}T$ or whatever. Then get corr coeff of T w. no of new cands. This will give slope, so slowly move in direction of slope. At some point we will have to approximate t. second derivative so as to get t. peak. This can be done by varying ln T over a wider range & fitting a second d.i.r.n.a.t.i.v.e

22 .06 - .22 is strategy for getting max no of feasible cands per unit time ... not nearly oriented toward ↑ in t. Core. — It may be that $\frac{\alpha}{Mx - \mu}$ is t. main way of doing this. Actually t. cc allowed for a cand could, indeed, be correlated w/ t. final Core (≡ Fitness Func), so possibly one mite use t. second

.27 dirivo of that "corr" to control t. mean T being used. Also studies of

.28 t. second dirivo could give clues as to how much α^2 to put in noise of .17

In t. long. I've been getting Second dirivatives in a noisy envt; (a very noisy proposition!) — but I think there is no alternative — either $\frac{d^2}{dx^2}$ or equiv is needed to get a "peak". Thus it may be that t. peak is so noisy that it's not easy to find it — in which case we don't need an accurate peak (a "peak" focused).

33 A (somewhat) More General approach: Consider Busrch w. some fixed T (a $\left(\frac{c.Bz}{cc}\right)_{\alpha} = R \cdot T$). See how it works. See what mutates to ↑ or ↓ T & how much.

& motivates??

D905

STM

N.B. This $\frac{\sigma}{M_x - M}$ idea is not related to t. S-grammar we were discussing earlier. Converse could be true — It may have different σ, M_x & M.

Sec 222.39

ref: 222.33 ff: W/R a given fixed T, take statistics on mean cc per trial, (i.e. no. hits/sec), Also statistics on Gore & ΔGore & $\frac{\sigma^2_{Gore}}{\text{GORE}}$ Those ideas seem relevant to $\frac{\sigma}{M_x - M}$ idea.

Consider idea of keep track of $\left(\frac{M_x}{\sigma}\right) = \left(\frac{M}{\sigma}\right)^2$ we want to MINE separately, because then, when M_x changes, one can immediately tell how $\left(\frac{M_x}{\sigma} - \frac{M}{\sigma}\right)^2$ changed.

Note that this analysis is for **BU** search only. I.e., we actually exhaust all cases, before we change T: CB by T+3T (Technically speaking, that kind of search for INV probs. is not used for OZ probs.)

So: in $\frac{\sigma}{M_x - M}$ method for BU srch, we periodically update T-Grammar (that gives our Bridge by P.P.) & we also update T (i.e. $\frac{cc}{cc} = pc_i T$)

T: Grammar itself is updated using new corpus. T: "T" is updated using A new To as compare & using T in a loop DF around To, using ideas of 222.33 – 223.10.

Remember BU srch is still only "Phase 1": it doesn't have to be perfect. — just good enough so that Phase 2 can get off to a good start.

So: A pro-tem Soln of the BU srch problem w. Recursive funcs.

.16 Both Prim. & Partial Rec Funcs can be deal w. using 222.33 ff.

OK. so 2 other Big problems are MPC ① MPC Mul Gen Par Codes partial codes (13.05)

19 ② How to implement $TM_1 = TM_2$. This is sort of a parallel way to Phase 1.

The idea is how to get a TM_2 problem into a usable QATM format (I guess)

Actually "TM_{12}" would be nice, but not essential if we go on to Phase 2.

I guess TM_2 problem is deciding what to use as "corpus" for QATM.

One way would be for TM_2 to use same Corpus that I will be using for PPM (or whatever other much Larger alphs used for TM_2). One way (again!) is we have big set of $O^j_1, O^j_2, \ldots O^j_n \ldots$ successful predictors ....

26 To get P.D. on O^j_{n+1} (or as PPM does) simply add 1 over i. $\{O^j_i\}^n_{i=1}$ set. → 225.20

It might be best to get MPC analysed so I'd know what kind of induction I needed. (Also Note that using QATM for TM_2 means that TM_1 has had a suitable TSQ_1 for TM_2's problem).

SN on MPC: one way of looking at it: w. corpus is A, B, C, D, E (unordered strings) & A, B, C, D, E are each QA's, or fairseed objects.

.33 we have codes for A, C, D; B, C, D; A, E; B, D, E.

How best to use these codes for predicting a new string: Well this

STM

Clearly a Bag-prods problem. For A, C, D, we have devised a P.D. on 4 strings! So for the 4 subcorpi of 223-33, we have 4 P.D.'s. We may want a wtd mean for predicting a new object -- but what wts to use? While each of the codes does well on the sub corpus assoc. w. it, each (usually) can code the rest of the corpus as well but, typically, — poorly, [Is this a problem for "Boosting?"]

So, we could use each of the 4 models to code the entire corpus, then give them the usual wts for new prods.

Alternatively, we could use the 4 separate codes as an auxilliary "classifn" scheme to decide which code to be used on each string.

→ A 4-class'fn scheme would be more useful because we would then not have to check each code modifn on the entire corpus. (Well, actually, even if we had a 4-classifyer, if we modified any of the 4 codes, we would clearly have to check it on items that it was supposed to work on. Presumably, objects would be mis-assigned to it by the classifyer w. suff. low frequency so it shouldn't make much difference if it worked better or worse for such data than the previous unmodified code.

→ In IDSIA, I was thinking of having the trainer give the classifn indices for early part of TSQ (for certain classes), so it would be easy for TM to use the resultant Complex Interface classes. However, in General I want TM to be able to invent new classifn schems & implement them.

Also B. Some Classifn schemes data based on External info (like the source of the data, or its applicn). TM would have to be given such info.

Also Note: Humans Do seem to use classifn in "pre-Induction" → decide "what kind of problem it is", then try to solve it.

While .10 off may be "O.K." it doesn't address the situation that the new MPC. Namely, a technique of coding parts of corpus w. various sub-algms. — that often overlap (or may not). It seems like a nice way of coding a corpus, but I couldn't find a good way to use it for predn. So I had no criterion for how good the separate sub-algms were — no way to wt. them as part of a corpus for Irng/extrapoln.

.03-.04 could be used. It has to good property that sub-algms that code much of the corpus are given much more wt. (I think I had this last time I worked on MPC.) As for "predictions" perhaps one shouldn't normally do it until one has a good safe code for the entire corpus.

D10 05

STM

I would, hwr, like to use 224.10 – .25 since it does have reasonable theory, MPC 3 big Busses TM12 problems

.01 Maruas used it much & also advocates 224.10–11

.02 If may be possl. to use both 224.10–.25 & 224.26 – 225.01

One way to do it: Say we have a set of sub corpi in which we want a common pradigm. If it has a large sub corpus, it may be useful to use on it.

Sub-corpus: first getting operators for sub-sub corpi – then operators for larger & larger sub-corpi, till we have an operator for the entire sub corpus.

.06 .07 It would seem that .02 – .06 would be the best way to use both MPC approaches of .02 (.16)

SN For the continuous PD. (ALP) T. Trme need not have a "stop state"; So superficially, it could not simulate Turing TM that did have a stop state. It (ALP could, hwr, a pair of states that it toggled between forever: it would never stop, but it would never print any more (or read input tape any more), A better way to indicate "a stop" would be for it to read more and more of input tape w.o. printing. Now this would make pc of printing very small. Not, T. total pc of all poss. input tapes / would be 1, so this pc would not be 0.

.16 .07 → For the 5 cases of MPC one could use different search of Busses.

SN Shane Legg Marcus Hy, ... FUS Fitness Uniform Selection.pdf (Notable: the paper is interesting in that in diversity not exactly variance σ^2 Access 2 D/22/05 : Ray's My Dogs)

While this is for Conventional GA; It looks like a "poor man's MXJ"

MXJ = Max expected Jump: $\frac{Q^x}{Mx - \mu}$ max; (Here note that this formula is not invariant under monotonic ↑ nonlinear transformations of Score. If Score is "linearized" it is corrected: Score = time, Money, PC recent are usually ln or linearized) is O.K.

22 : 223.26 O.K.: say .02–.07 is O.K. discussion of MPC; the "soln." of Ry Such

Consider TM_{12} (223.19 ff! We have Noted that TM_{12} is not ably hacy, since we plan to do Phase II! Also Noted: In Phase I, TM_2 really doesn't understand much what "Optimization" means. In Phase II has a much better Understanding. – It would probably for this reason be wasteful to use the complexity of full ALP in Phase I, when it didn't have a really "proper goal" for TM_2

.26 So may be drop TM_{12} for the while → Quick Review → 227.00

.30 The main apparent remaining problem is choice of Lang. for problem statement & soln. —— And ISQ design.

226 is Rev

→ 227.00

D1005

REV.

220, 226, ~~245~~ 246

(QA: Ganzn!) 3 kinds of theory — LTM 11.01 ff 2.2.01

5TM

D: 220.34

12) Formula giving decay rate δ as a function of fitness order for Tournament Selection and size of Tournament GA/GP.

13) The Soln (perhaps) of the "Chess" problem: I think involves making approximations & Gambles per such cases on basis of past Experience. (1·3·99)(25) perhaps || Note GTM 260.20

03

14) Just where is that original & copyright Stanford Comp. Sci. Dept. Report from Samuel to solve ANL? Somewhere like "Page 14". It is indeed Paul 14.00 ff: 6/7/90

Also Note P66 ibid 65502 deals a version of AM

Ibid also uses "Howto Solve TSP problem" in Japanese/way.

15) T. various forms, methods of putting/possible outputs of a known Stochastic source (3-long) in π pc order (=6)

16) Proof of first Gauss-Thm: (That $\frac{pc}{cc}$ orders optimum)

17) Kolmogorov's Theorem on expressing functions of n variables as a wtd sum of functions of 1 variable (— Mc Thm is π Pic — I don't know what th. Thm is; I don't really understand it. Hnd. Barrens 1995 paper on ANN may cover it — I also have once 2 downloads Professor U.E. . . . . but where?

18) On "Reversible Computing" I got a formula (Guess) for th. rate of computation in terms of Energy per unit time (= Power). Was somehow guessed to be $\sim h$ (or $\hbar$).

See QM file

I think h = "action" = energy × time, i.e. $E = \frac{mv^2}{2} = mv\left(\frac{v}{2}\right) = \frac{mv^2}{2} = p \cdot (\text{momentum})$

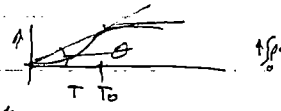
N 1.20.01 for details

For got just how it goes! Actually ~~it~~ it returns back bits: thus error can't be < T for worst trial/query. Energy uncertainty $\approx = \epsilon < \frac{h}{T}$ so energy for 1 bit is $\frac{h}{T} \cdot T = h$ dimensionally wrong — but th. idea needs cleaning up

say (K)

19) Electrolysis: I've forgotten details! It works as oxygen ion added indeply of th. other ions (—).

20) Proof that Th. normalized dash I used for ALP is the one entry closest expected error. Periodically lose/find that proof. → 22) formula on Norman

21) My Relatively Recent Solv of WON-problem: 

Specialization

Do/Goals in order of $\tan\theta$ & up to T_0: $\tan\theta = \frac{pc_0}{T_0}$, which is optimum order

We assume T_0 is acceptable "Grain Size" of solng.

See "WON" folder P23: P22 has refs on "NIPS" on some Relevant Work on AND & OR nets.

→ 24) 00

b 11 05

REV!

5TM

Space 225,34

this will be a review of Lsrch V.S. BUsrch: Why BUsrch is needed.

& implications: Then How BUsrch can be done, using suitable functs on cc for comps.

Next: Review of MPC (Multi-parameter Coding) problem: resolution into 2 distinct solns & integration of 2 solns.: Why we need both kinds of solns.

& BUSrch w. both kinds of functions:

1) Lsrch V.S BUsrch: Why do we need both: My impression is that If All of the hours needed are expressable as modulus of a coding P.D, and all of such hours cutoffs has been pointed to by coding p.d. and that one is not allowed to change the coding p.d. during srch, Then Lsrch is the best one can do.

Also does the problems addressed here w. open probs! finding good codes or good s.functs for the corpus

Do discuss why we have to keep variable cc's: That we do needed do Rec. funct's. recognition of Stop comp

Ffvr., there is a broad range of cases in which $\frac{cc_i}{pc_i}$ of soln is beyond what one has available, and so Lsrch would not be practical. If a problem of this type is soluble, then for a prior $^{\circ}pc_2$ of reasonably small, there must be > 1 soln. Conversely Lsrch is best way — even if it's impractical (>) Multiple solns means that one can pick a small $^{\circ}pc_2$ and do Mt. Carlo srches at ($\frac{1}{2}$) that $^{\circ}pc_2$ level. If the soln. $cc = cc_i$. and there are n solns at that $^{\circ}pc_2$, the probby of finding one is $^{\circ}pc_2 \cdot n$

If $\frac{cc_i}{^{\circ}pc_2}$ is too large to find; $\frac{cc_i}{^{\circ}pc_2 \cdot n}$ may be small enough to be practical (>) 1 of the no. of trials needed is $\frac{1}{^{\circ}pc_2 \cdot n}$ & the cc of each trial is $< cc_i$.

If ETM is usable form

If ETM can do BUsrch — why write TSQ's? Because it's a fast way to

So the Q is how do we get comp. bounds for trials, using BUsrch in Mt Carlo mode.

SN Re:

Phil

STM

On Turnl. lays for {prob. dern, soln}

I had been thinking of using folk examples to illustrate need Turnl dangs to solve probs! T. machine is given pairs of a sine wave to learn. It learns it. Then later, it's given a sine wave of diff freq, phase, amplitudes. ML systems based on non-recursive fcns can't learn & new sine waves w/o. an enormous no. of new data pts. A recursive soln. can realize that only 3 params are diff in 2 sine waves & quickly extrapolate & second sine wave.

The exact details of the "learning" have to be expanded.

Another point about recursion! There is classical recursion definition like in Lisp: f(x): if x=1, f(x)=1 else f(x)= x*f(x-1).

Very expensive! prim rec

Same fcn can be expressed by f(x): f=1; do for i=1 to x; f=f*i. loop.

However, .10 seems to be better heuristically, in which a person would notice how a soln. for n, s of a problem for x=n differed from soln. for x=n-1.

Jürgen Got his soln for Tower of Hanoi, using it form .11. Is there a soln. of comparable pc using form .11?

Perhaps Hanoi have discovered this heuristic!: Say a problem can be solved for X=n if a soln is known for x=n-1 (x, n pos. integers). Then if we can solve it for X=1 we have a recursive soln.

Is .17-.19 Related to theor I used to show linear, than quadratic than cubic eqns? (I could try running it on quartic!)

T. Pgm Discipulus does use a regular (Turing Universal) inst set from RISC that normally would use ordinarily using f. recursion machines.) Is this system able to solve problems of .10?

{AIM 8P6}

Ans. ACEPD Answers Manual for Discipulus is in C:\pemtiles\AIMemos\Discipulus\ownerstuff.

It has inst. set & discussion of it.

SN Q: MPC Mutt partial codes was considered discarded for the QA problem in Phase 1.

Here, in Phase 2, we normally start out w/ pure search & we do one problem after the other in strict order. If search process is quite well defined (except that we may used to write pgms can vary & modify search to some extent).

I guess MPC was meant to speed up Phase 1. We by various O's in strict Reverse Cartesian cross order. When we test a OJ out on other corps we find some O's work o.k. for some, but not all of the corps

229.00

5 TM

Guiding PD
Guiding PP
GPD

Def 228.34 So we use these partial solns to modify the GPD to speed up acquisition of a single O[J] that works for the entire corpus. — Ref. to McRob (224.26-225.01)

Alternatively (see 225.02-.06) we can use 2 methods of MPC:

First breakup the corpus using (224.10-.025) using some R funcs:

Then use 224.26-225.01 on each of the sub-corpi.

— So Essentially, we have 3 methods of working on Phase1:

1) straight Lsrch 2) using sub-R funcs (a possibly s-R funcs!) to breakup corpus, then use regular L srch on sub corpi. How to update Guiding PD, Coesdderacy distinct subcorpi, is a problem.

3) Using 225.02-.06 on the partial corpi, rather than simple Lsrch.

The many problems seem to be: 1) How to update GPD: How much info to get to examples from other sub-corp. 2) Choice of Representation Lang 3) Writing TSQ.

2 & 3 are linked.

Re: 2 & 3, I really could work on the TSQ directly, now, by choosing a lang. Then design a lang to express things I think are "nasty", stuff to solve the problems. I got into that approach for a long time ago (1990 & Baker) but now, I think that I understand langs better, I understand "context" better & I know some Mach Learn methods to enhance "context" in practical ways.

For initial test of ANC: One way: transcribe to Irn. $3 \frac{\dot{x}}{\dot{+}} 4 = \ldots$ with pc = $\frac{1}{4}$ for correct answer (unable to tell whether to use $-, +, \times, \div$.) Next compress by noting correct. of + w. spin, − w. sob., etc. It should be able to look for "near by" cancels of squares. This would give completely exact soln.

Next, to learn to evaluate "nested" expressions, — i.e. first certain substrings had "values" & "equiv" assoc w. them = these "relax" actual (no. or permissible arguments of functions,

An Older Idea on writing TSQ's: Just write the TSQ so that "it is reasonable" that a person should be able to do Irn's. Then write solns to its problems — actual Code in some lang or "Pseudo code" i.e. "lang" can be English-like with lack of specificity & ambiguity — so as to make appr. set of terms easy to write.

I may get my TSQ's from Maple/Mathematica Defns, operators,

EN Q: Is knowing what it means to solve a problem/equation related to knowing what it means to get "Best Score available from" the specific — or problem?

Then start making the lang more exact — & try to find way to make

System

"Primitives" operators that are cheap. Finding min cost operators to do a certain task can be done by GP (or Co evolution). TC it may be worth while to get operators that are very cheap, even operators that do exactly what I want them to do!

The idea is to have a rather vague TSQ, à unsure sans to it ... then slowly tighten up ent. TSQ à ft. sans.

An interesting Q is: just how to implement ft recursion used in evaluating Nested functions. I know of 2 general recursive mechanisms 228.10 à .12: Es use of a stack for recursion [redacted] one of 2; say 228.10?

[redacted] Ent. case of Assembly lang. 228.10 à .12 both use simple conditional jumps. T. use of a stack is usually done by special machine instr: Look in "Inner Loops" to see if machine stack saves any times.

In line w. 229.26 ff; I had ft idea of wanting ft "solns" to ft TSQ in "English" — somehow augmented by (functional) ideas that aren't usually a part of "English".

One of ft ideas was that of an "Object" à it having a no-assoc n. that object (its "eval'n"). At that time I didn't know about ft Lisp "Quote" function — that seems Relevant. [MapReduce has no "Quote": Manages to use it very usefully, but was a way to get it if/ ready needed (according to Fredkin)]

ISN Often there are several ways to work a problem, each usually instructive

How could I get TM to do such all possible solns? Well, if TM were in Search mode, I could just let it continue search till it found all of ft solns:

No that's not what I was thinking of; I was thinking of several different TSQ's — each able to get TM to solve a certain, narrow, difficult problem. Well, it may be possible to get TM to do ft several TSQ's in sequence. The knowledge obtained in previous sub-TSQ's should not hinder ft finding of ft needed concepts.

If it does seem that certain Concepts are not found because certain other Concepts were found earlier ... then we have TM "oversearch" until it has found Concepts. are found. This idea would work if we knew exactly which Concepts TM found à did not find à what ft Past Lessons were (so we can pay ft CBAG (at least) those Lesson levels. → Paul {14.00 (June 6/7/05)}

ANL at Search, I used the "Refresh" stacks mechanism to learn all kinds of Alg notation. I was really troubled because ft stack seemed a bad A.M. for that problem. On ft other hand, a stack is useful in many other ways.

The other 2 troubles: (1) I only used above that more complete solns to problems (2) The Lesson solns ran very rapidly in ft TSQ.

Dicos

STM

At present time, I think (1st) would be solved (or much mitigated) by using suitable "context" à PPM a/o other Mach Lang Algms.

As for lang. itself that I used, It would have to be augmented (possibly) to be turing Complete .... but I'm not sure. It does have "do (statement) until (condition) (enddo)" an until loop: which could give primitive recursive (some Partial) Recurs refuncs.

On Paul 14... Sun 6.7.05 : I had "call" I used this to access successful Pgms stored in Memory. Each time I solved a problem, I put the Soln in Memory. So pc of Memory access was N if one had solved N problems thus far.

Another approach to TSQ design was that of Sol 89: using a [concept net]. Perhaps I should try this again!

Actually in Paul's notebook on time back that (~ 1990) I did do a lot of study of various methods of writing TSQ's.

(7 to 1) → Ah! That linear → quad-cubic eq. seq was essentially a [concept net] !

Try working out details!

While I don't like to "tell" TM any thing directly: If it sees a problem (one may attempt) I'd like to [tune] the soln method. To some extent TM2 (PPM 32y) does this, but I want TM1 to do it in a more "Universal" way. → (232.07)

If, (Say in trying to implement ...12) I can't get TM to learn all of the needed concs, (due to my inaptitude), I can "tell" TM the concs. — Tho it would be better to use minimal "Hints" if poss'l. A TM "educated" w. occasional "tells" will not be as smart as one that does all of its education by true "living". (i.e. (pure discovery)). I may simply have to use a longer TSQ to get the TM 2up to speed ... Perhaps .14-.17 would help! Teach (or tell!) TM how to do .14-.17,

(Also Note 233.01 .14-.17 → 232.00)

To Some Extent a Skimmerray TSQ is like "telling" or "Many many Hints" To an Extent, one Can Monitor (# Gaps, Measure, Estimate) the CJS needed by an animal or Human as he goes along in a TSQ. This interview can indicate as to the Max CJS needed by the Creature in various problem areas. Also the bypass of discovery that are learnable; (Recursively, this valuable ability can be a simple inadequacy of the TSQ ---- This Q is in general "Undecidable".

TSN In tsq design, learning various definitions in Maple/Mathematica can be like "Milestones" in TM's Education.

2

STM

231.25 So: doing 231.12-13 or any other diff problem in "Heuristic Pgmg" would be tough! First write pgm that uses desired heur to solve desired problem set. Then write TSQ's to acquire the heur(s) or partial heurs needed — using other, more "primitive" heurs (it is not always clear what the heurs whether or not a heur is "more primitive" — Ultimately, this means we are closer to being able to solve it using the "primitives" of the system. We can, of course, include any desired heurs as "primitives of the system", but if they are "really" factorable ones, we would do much better with "factors" (smaller ones) as primitives.

The main point of the large "exercise" is getting a TDM that can go well beyond the problems it was based on. Factoring down heurs to be close to "apriori primitives" would seem to be a reasonable way to do this.

A "troublesome AM" (may be more troublesome) is that its primitives weren't small enough — i.e. it did have small primitives, perhaps because it hadn't devised its heurs, it did not have the "right" set of heurs, a "universal" set of them — but it didn't have an adequate set of heurs, or how to get things together to improve them! It was unable to discover new ones. Later, using Eurisko, AM was given ability to devise new heurs, but it really had no TSQ, as its quite possible that its lang. AGM was inadequate.[250] (243.07 in reference to AM)

AM(~) troublesome 231.12-13 as a problem is that one actually injects new primitives into the system ($x^{\pm}$ & $x^{\frac{1}{3}}$) as the TSQ continues, — which makes it a "non-standard" TSQ. It is non-the less useful as a "study problem" for Me: Also one could introduce the recursive operator $x^{\pm}$ w. arby positive x & arby positive integer n.

The main advantage of introducing $\sqrt{x}$ & $x^{\pm}$ later, sequentially, is that at the beginning one has fewer primitives, so it's easier (hyper approx solns) to solve the early problems. (In the Piaget idea of child development, a child, may have various needed skills introduced (frequently) at suitable pts. in his/its life.

It's likely that there are lots of Heuristic Pgmg Hacks that have been done to solve fairly difficult probs — that could serve as basis for a TSQ.

An imp. idea is that in the "big problem" one can, most of the times, not have would act the mechanics of trying many heurs — still, one would get lots of practice writing & using TSQs for sub-problems.

5TH

5 — in each case, the problems would be "well defined" ---- which is not true for usual "start from primitives" type of TS Q's.

Perhaps look at Encyc of AI, or more recent Reviews of AI, to get 'Good Hour' gaming problem Solns. Also perhaps Read paper by Sussman et al on "One Shot Irving"

$x^3 + ax^2 + bx + c = 0$ $\quad x = z + \alpha \quad$ $z^3 + 3z^2\alpha + 3z\alpha^2 + \alpha^3 + az^2 + 2a\alpha z + a\alpha^2 + bz + b\alpha + c = 0$

say $\alpha^3 + a\alpha^2 + b\alpha + c = 0$ which is original eq.!

07 On "Hints" in Meta Heur of 231.14-.17 : The "Generalize" idea is certainly impt. One Prog I once considered in this area: If TM had some ideas about what future problems would be like, it would be able to "Generalize" more intelligently. — The "Problem Pool" type of "The Method" was (partly) designed to deal w. this. W.r.t. any ideas about the future, TM can only examine the past & see if the new method of solving problems could have been used in solving earlier problems. So: for TM to have any idea what future problems will be is certainly useful in analysis of prob solving & the discovery/application of heurs. .... but it's not absly necy: TM probs of past can solve old Pbs — but not as well.

18 Key way: Discuss how first Chapter of Pearl's "Heuristics"

19 T. heurs discussed were obtained by ① Simplifying the Actual problem & Exploring Soln. methods of simplified problem as Main problem.

20 ② Instead of Simplified problem use an "Analogous" problem (in the sense

22 of "informationally close") — use its soln. methods (presumably it's

23 easier to Solve than the Main problem), → (Note 234.10) →

Hmm— A heur of this sort seems distinct from what it seems to need a QA for! Pearl uses Heurs to Solve INV probs. He generates heurs viz $(.19 - .23)$ is true/false on both. So he starts out w. a p.d. on All heurs — he then may try them in pc order or (cost order). These problems are all INV "SEARCH" problems. Since INV probs are rarely solved as such, but as OZ probs .... One impt kind of heur is Devising a "fitness func" (= Gore).

E.G. in "8 Queens" problem: Place the next Queen one pt. s.t. it leaves max no. of squares "unchecked" (i.e. available for new queens).

In Trav. Slsmn prob: We might try to invent a Gore. Say divide Cities into 2 sets s.t. one city from one set to another. Try to estimate Gore of

)22-5

STM

xo.

01

04

each of f. 2 sets: Chose a pair w. best "total" Gores,

Another ~ problem: Given 2 cities: to go from 1 to another w. min path

Given [copies of roads thru various other pairs. One heur: start out to

that length traveld + α(Euclidean distance to other city is min)

try various α's (α is usually > 1).

It may be that just about all INV probs use Construction of Gore as a heur.

Actually 2 of f. probs (Trav. Sls. & min path betw Cities) are OR to start.

T. heur consists of devising a Gore that is easier to solve. — Then using that Gore

instead of the Actual Gore. [One could do this in GP probs in which Gore was

Very Expensive]

233.23

0 : 233.23

Actually, the Devising of an Easier Gore can be regarded as special case of

heur of 233.21-.22.

Another (perhaps) way is to realize that present problem is ~

to previously Solved problem ("classification" needed to discover this) — & so

that we may need to Modify that older Soln marked by goal of so

doing w. present problem.

On 233.22 I considered similar problems to be those that were

"intuitionally close" to present problem .... but that's often not exactly

so pb. The right point is that the older soln was for problem ~ to present

problem "in certain way". What were Common features that Made the

heur Workable?

Can I easily Characterize all heurs for INV OR probs?

A heur has to be made "likely" via (optimal) reasonings or Statistical

studies of Past Cases. (& sometimes future cases!)

To start: Heurs equiv. to change of GPD v.s. Heurs (like Quick Abort) that are not equiv to GPD changes.

My impression was that "Quick/abort" may have been about all Heurs used in K.

2nd kind of heur: — Tho it is an important heur. E.g. Ordering Comps w. most difficult

Def QAb probs first, is one kind of Quick Abort (QAb)

In Pearl's book, T. heurs are often derived from the trainer's experience in R.W. —

e.g., in .01-.04, the use of Euclidean distance is common in R.W. — (possibly somewhat

common in M.R.). The idea of "distance" — that one can tell subsolns by

one distance for another — Could be learnable. If 2 "distances" both

Satisfy (a...a ≥ 0 a to B = B to A, & Δ inequality), then they tend to be more "real" than

most pairs of Distance types.

5TM

(SW) On possl talk about "Semi-Universal" functions: My NRS wants to Rely a bit. I can do this more exactly. Some kinds of Semi "universal" induction models fRANN, BBN (normally no feedback), Quantum Computing (simple sections) SVM (Simple kinds: linear; Non-linear — Rao may be variations I'm not familiar w.)

Give examples of sin wave Irnd, Ran given sine or distort Amplitude, phase freq. ~ "One shot Irng" for

Recursion should, in general, be able to get equal results w. smaller SSZ.

or better results w. same SSZ.

in "Pearl Pred. Challenge" workshop on "Model Selection".

Dataset ADD!	Size 6Mby	dense	features 48	training size 4147	validation examples 415 (always ½ to size of?)	Test examples 4147 (always equal)
4 other datasets.						

Perhaps look at the models. They give you a Matlab Package "CLOP" 4.2Mb download. — which runs under "Matlab® version 12 or above". (I could get Matlab!

It would be good to know just what kinds of Models are being used. — it looks like an enormous range of Models)

Discount SW: Matlab 7.0 release 14 w SP1 (service pac)

$50 for 3 CD's! — 3 CD's is a lot of Download time!

Maybe got CD set for $100?

May take a day or 2 to get it to run!

Would it be worth while to simply download the data & some by running records wrt. solns. so I can see how well I could do, using recursive Models. The results might be useful for exposition, but if may be that I can show theoretically that recursive Models would be better. (Like, probably perhaps hyper cc).

Getting Back to Hours: It certainly seems like a relevant Study just now. I could write a "TSQ Outline" Giving various impt pts. Irnd, in sequence. The main problem is to get hours to span the JUMPS, using realizable CJS's.

25 Jacques' "Boosting" (in OOPS) is an interesting hour: it temporarily Boosts pc's of tokens used in a previously successful problem solution. Main weakness: It should select "w" probs in past ("w" to present problem) — Hur, at the beginning — it has no good basis for "Similarity" measure.

As an hour, it is an OBVIOUS example of "Modifying the PS": Because of the complexities

30 of its use, probably one could do as well by getting a narrower pd only set of primitives by using Chem to solve a set of "Basicly impt." problems ... by the trainer. TC. set of ~70 insts could be effectively & by a lot by giving hyper pc's to various Primitives used solving the set of "Basicly Impt" problems.

STM

MchL MCLR MLR=Mach. Learning

① The kinds of Heurs discussed in Pearl (233.18 ff) do not relate present problem to previous problem(s). Furthermore, I don't know just how to express such heurs as "Modifn of the GPD".

I don't (immediately) see how to heur(s) I had in mind for ANL fit in to any of the schemes: (E.g. 1) ··· 5) — ⑥ is what I had in mind for ANL recently.

① Modfn of GPD ② Solve "like" a problem in past ③ "Simplify present problem; try soln of simplified prob. on present problem. ④ Devise soft Gorc for INV problem (→ 02 prob.) → for 02 problems: change to Gorc in way that reverses soln (& verifies) in reverse.

Dap (Guiding Prob by Distribn)

So I don't have a very "General Picture" of heurs: Other than heurs are either Modfns of GPD or ≈ Quick Abort methods.

⑤ ∑ One Genus of QAbt is to shorten/hasten/mean. There may be few cases of Phase so that I can safely "wire Brain into" TM ("Teaching By Brain Surgery")

⑥ Another "kind" of heur is the kind of GPD modfn obtained by PPM or other "Machine Learning" means

⑦ Another Common (imp't) heur: do problems #1, 2, 3 ··· n: Then see if there is a general pattern, so prob for #n+1 can be obtained from solns for #1 ··· #n

This could be done as a "problishes symbolic" regression, rather than "d-regression"

→ This is actually what we expect to be done by an extrapoln of O_1^j, O_2^j, O_3^j ··· $O_n^j \to O_{n+1}^j$.

using d-regression may be regarded as special (distinct?!)

6, 7 are cases of ① ⑤ is QAbort type(s) so ②, ③

④ (devise 02 Gorc for INV prob) can be regarded as a special heur w. many variations/modifns/approaches,

→ DoPs' "boostq" (235.25) is a cascade GPD modfn; where "closeness metric" between problem descrs, this would work well (probly other methods would work well if we had a suitable "closeness metric". Actually, what we want is some way to measure certain aspects of the similarity of 2 problems, and map those (probabilistically) to soln techniques (or PD's on soln techniques ··· which is what "boostq" does.

The idea of 2 problems being ≈ in a certain way so that the soln method of the first can be used on the second. ···· In classic heurs that sense, this was found by "logical analysis" — But this "logical analysis" was done by the Designer not "AI" pgm.!

So at first ··· at least until TM can do "logical Analysis" it would not be able to find heurs of that kind, using "logical analysis" — But it might be able to find s-heuristics that were close to d-heurs ···· & perhaps find them by statistical, rather than logical analysis.

5TM

SN On determinism in Language AZ (Described in Appendix of IDSIA report).

Using PPM for TM2: If a certain functional form occurs in generated by a candidate AZ, I think the pc of creating that function in the future (on that candidate) will. For that matter, I think the amount of it would be that which would occur if we used a "definition" for that function as used the symbol for that function twice. (expand this, it's not clear!)

So as for definitions, PPM knows about some pc's in AZ.

If PPM were able to do this for all functions & all possible substrings, then definitions would be strictly unnecessary .... They would not modify the pc's at all but probably have more cc than PPM.

The rub, however, is that PPM cannot detect all subtree recurrences, so the pc at least doubled at the output of such routines can't find all subtree recurrences we may want to use definitions. → 238.12

Eg. If all heads in GPD models; we can restructure GPD models to code Easel Rider as heads & separate it from Rider

SN In PPM the cost of an "escape" (if it's a different number) is $\log n$ where n is the no. of symbols thus far. Say escape has assigned pc $= \alpha$. We can code the corpus & compute its pc as a function of α, then choose α whenever each time they occur, to maximize pc of corpus up to that time. I think α may end up being given by some Percy like $\alpha^{n_1}(1-\alpha)^{n_2}$, so it's clear what value α should have to max this expression. It is, however, likely that this pc is close to the Bernoulli pc.

Is n_2 the no. of symbols in the corpus?

It is an old idea.

(as) have been tried: May be look into it; it looks like research Medical analysis.

Also Note → Since (I think) escapes are rarely used only once when coding a symbol .... we could use α_1 for the first escape & α_2 for all subsequent escapes.

If n_2 is no. symbols in corpus $\frac{n_1}{2}$ is max no. of escapes per symbol coded. $n_1 \leq n_2 \cdot C$ where C is the expected no. of escapes per character. ? — I think $C \geq 1$.

C is always < 1. $\alpha^{n_1}(1-\alpha)^{n_2}$ is max when $\alpha \propto n_1$ & $(1-\alpha) \propto n_2$ (see .32) ∇

$\alpha = \frac{n_1}{n_1+n_2}$; $(1-\alpha) = \frac{n_2}{n_1+n_2}$. $\alpha = \frac{n_1}{n_1+n_2} = \frac{C n_2}{C n_2 + n_2} = \frac{C}{C+1} = \frac{1}{1+\frac{1}{C}} = \alpha$ optimum

n_2 = no. symbols in corpus; $n_1 = C \cdot n_2$ = no. of escapes; C = mean no. of escapes per symbol

C is some Percy like the mean no. of non-escape codings except between escape codes.

n_1 is no. of escape codes n_2 is no. of non-escape codes = no. of symbols in corpus.

$\alpha = \frac{\text{no. escape codes}}{\text{no. escape codes} + \text{no. of symbols in corpus}} = \frac{1}{1 + \frac{\text{no. of symbols in corpus}}{\text{no. of escape codes}}}$

.13-.28 is a nice way to code/extract PPM! Use this understanding to extend PPM to sub-tree redundancies. — Also to compress to Z14(.

32: from .22 ∇ $\alpha^x(1-\alpha)^y$ ln $x \ln\alpha + y \ln(1-\alpha) \to x\frac{1}{\alpha} + y\left(\frac{1}{1-\alpha}\right) = 0$ $\frac{x}{\alpha} = \frac{y}{1-\alpha}$ $\therefore \alpha = \frac{x}{x+y}$; $1-\alpha = \frac{y}{x+y}$

More exactly; if $\bar{A}$ is Δ and the non escape & escape symbols; e.g. seq. $\Delta, \bar{\Delta}$ in an iid Bernoulli seq. the pc of the n^{th} $\bar{\Delta}$ will be $\frac{k+D}{k+b+n_\Delta}$, where D is the "Dirichlet constant"

DQ4: Dir. const

D 25.05

Which replaces sets of "1", but we may be able to get better values by studying texts (say Eng. texts). There is some fuzzy part of the first 10 or 20 symbols of the $\Delta, \bar{\Delta}$ seq. may have somewhat deficient codes. Also, we can make a post-refinement by coding each Δ (in a seq of Δ's) w. a different pc: i.e. the nth occurrence of Δ (in a seq of Δ's) will have its own Bernoulli seq. & its own "Directact Constant". HM! After Compressor has coded the entire corpus (while leaving it "open") it can decide on an α value and make that value part of the code (or it could code "C" on the 1st of 257.22

SN Would it be useful to study books on "problem-solving for Math enthusiasts" like "The Ramanujan book" by G.S.CARR ? Write down each heuristic used; then try to find how it could have been derived. Maybe Google can help find Books or Papers on this topic ... or Ask Marvin or other Teachers.

12:237.11 → I want to see if I can get a close correspondence (Mapping) between elements in PPM & Elements in ZCH (or AZ).

Another approach to PPM: By using the longest prefix having the needed symbol following it, we get one code for corpus. By using the (nth) second longest, we can also get a code for corpus: Those codes are all codes & can be numbered in various ways: (Ming & Coverdiscuss this) Using the wt for a branch of all codes can give better results than using best code or weighting codes on basis of "pc of corpus wrt. that code" (whatever). Its an empirical Q.

It would be interesting to compare PC of entire corpus viz the various codes. Then compare w. various ways of all wtng.

23 The same idea can be used in "trace PPM". Consider all "prefixes" that consist of sequences of symbols "ruled out by the predictor". Pool predictions of all such prefixes. Closest to PPM, pick largest i s.t. that has desired prediction. The largest i that has "any prediction" will be $\hat{i}$: $\hat{i} \geq i$, $\hat{i} - i = k$, the no. of escapes needed — so pc $\propto \alpha^k \cdot (1-\alpha)$ for 1 symbol. This is mult by the relative freq. of the desired symbol; in all of the prefixes of "size" $\hat{i}$ ($\hat{i}$ = definite-wall determined; i = largest prefix that has occurred at least once before in the corpus)

123 is a formal soln. to the problem: assuming that one could do a complete search for all prefixes of size $(i = 0|\infty)$. Actually, this is not terribly necessary. If one uses any definite-wall determined d-algm. to find prefixes, (even tho one will inspect all of them) we can use 123 thus. Esp., we may limit our searches to trees depth of 3 or 4, say. → 239.10

STM

Back to Hours: I want to make a big list of Hours, so I can find ways to categorize Them — to make a general Grammar to deal with them all — so I can get TM to learn & extrapolate to Grammars. G. S. Corr

This is essentially a kind of Problem Solving Grammar-stuff. Kind I expect to design in "Phase II".

Hmm! Will This Grammar of Hours be something TM is "born with" — like human ability to understand human [redacted] created languages?

There is a defense, kinda..... the human rays were designed to be easily read by infant brain Mechanisms. Here, we "Design" the hours to reflect the world TM will live in — the world of problems we give to it. ← 243.00

.09

0 : 238.31

STM Using PPM for Substring detection: for depth 3, say, code the corpus 2^3 ways! each corresponding to a distinct branch choice

← exactly 8 (= 2^3) distinct ways of branching = results in a distinct "corpus".

Each path has same top node ∴ same point in "total corpus".

So we get the longest prefix ending in desired symbol, for each of the 32 paths. from [redacted] Then, we can get a "size" of it (see 238.23 = 31)

(obtaining "?" from the 32 longest prefixes is a "well defined problem" — I suspect it is not diff to solve. ~~It is just 32 paths of length k_i, Then~~ — But I don't want to spend time now! Probly a recursive Algorithm could do it!)

An Objection to all of my stuff is that the system will not detect any Postfixes that result in desired symbol. One way to deal w. this: After TM has found "good" postfixes from the limited 32 paths set, I can examine the (relatively) small set of prefixes, perhaps to see if they can be extended. Actually — This is easy to do ... only paths of length 3 can be extended.

So maybe not so bad!

12705

STM

Compression "State of the Art" .. (up to Dec 25, 2005)

Matt Mahoney has website "1. PAQ Data Compression Pgms"

It gives performances of very many pgms upto Dec 25, 2005. — w.r.t. Calgary Corpus of 3,141,622 bytes uncompressed.

Pgms for ~ 3 Mby corpus

The best so far is ~.568 Mby = factor of .1808; | X8 = 1.447 bits per byte

Usually the Text files are fewer bits/byte (this 1.447 is for mixtures of various Document types)

But anyway, it is approaching the 1.3 bits/byte of humans

This website has pointers to pgms (& Binaries) all are freely accessible ... but some (WinRK) are proprietary: WinRK has 30 day trial. has about the same compression as the best (~1.504 bits/byte ... maybe lesser (latest version)

Also pointers to Literature, papers on Ideas.

I think the best pgms use a Mix of predictors: PPM, Dictionary lookup, adaptive weighting of various predictors. I think a neural net was once used for part of it.

The PAQ pgms (many of them) are all "open source": Maybe mainly written in CPP (=C++)

PAQ7 is latest version (not much better compressor than PAQ6, but maybe faster)

They do have assembly code for some assembles w. NASM

I downloaded (latest "β" version of) WinRK (Dec 26 05 ??) 30 day trial. —

However, probably the pgms available on line would be easier to use for predn. & easier to analyse. (WinRK uses an English Dictionary),

---

Actually: Since ~ 1.447 bits/byte is for a 3M by corpus of various kinds of data —

One could compress better (equivt to larger corpus) by putting in "a priori" params obtained by coding data similar to the sort of data in the corpus.

& I.T. algm would have to know how to distinguish one kind of data sent from another — ... but I think some of the better compressors do this anyway —

They have special predn algms for each corpus type & they adaptively calculate wts for each predn. algm.

In comparing to human 1.3 bits/byte: The humans do have much a priori info — & they know that the data was (English) "text"!

Also the time needed for each pgm: we should use {length of compressed pgm in bytes + $\log_8$ (time for given pgm)} as efficiency factor.

More exactly use "cc" here rather than just run time; Memory cost of CPU can be (Thusly) included

STM T. "Theorem" about the extended Kraft inequality starting on .23, is proved on 242.30 → 248.00 - .12

SN HA: A much simpler proof that Model 1 of Sol 64₂ converges: I considered all types that cause the machine to eventually stop, and I used the size of the program covered by the Machine as its "program length". These programs form a prefix set because any extension of any program would be one a region that the machine could never go. So $\sum 2^{-l_i}$ converges, as does $\sum \left(\frac{2}{1+\epsilon}\right)^{-l_i}$.

(No! see .18)

I didn't even need the $\left(\frac{1}{1+\epsilon}\right)$ factor!

Even Leon's (misguided) objection was wrong! $\sum 2^{-l_i}$ does converge - - -

— but check this! In the original Sol 64₂ the exact natural input sequencer was unclear.

I suspect that the "prefix set" property of the input sequence is tacky in dep of just how one defines the input's "length": We pretty much must end up with Leon's "Extension Probability"

The "prefix set" of pgms is a bit different from usual. The "pgms" (It's a "double ended" prefix set (if it is) ≠ prefix set) are finite strings over which the Machine had control. As prefix sets they can't continue on either end. — which is ok. since even normal (Sam Input) prefix sets can't even be extended on the "zero" end either.

Still, I'm not sure how (or if!) the Kraft inequality does apply in this case. [bracket] is illegal! [bracket] is illegal.

.17

.18 Woops! [bracket] [bracket] [bracket] ..... are not even extensions of one another, but $\sum_0^\infty 2^{-2} = \infty$.

So it's not a prefix set

Well not so simple! All of the pgms have at least 1 square in Common!

Say [bracket] is one of the pgms.

Try this: consider the common square! If we disregard all excursions $\le \phi$, then we have a (normal) prefix set for the sequence ≥ 0: so $\sum 2^{-l_i} \le 1$!

If we allow extensions to $< \phi$ this only makes the codes not shorter so $\sum 2^{-l_i} \le 1$ still

Trouble is: we are now allowed 2 (overlapping) codes to be identical for squares, ≥ 0 but different for $< \phi$!

In that case for a set of codes where positive parts are same sort of long pgms, then the total wts of the negative ends will have wt. much smaller ≤ 1, so this "objection" causes no trouble. This seems to prove the Kraft inequality for those 2 ended codes!

I'm still not sure I have a proof!

(Teacher's reply.)

It's legal? Quite possibly not! very likely not. (Almost) certainly not legal.

However. When I wrote Sol 64₂ it definitely had the idea that $\sum 2^{-l_i} = \infty$. What sort of input rules did I have in mind? — [After machine stopped, any desired extension of input code could be still part of a legal code for the output]

D27 05 5TTM

Also if one has a "normal" complete prefix set, starting from squares ϕ in >0 dimension, then it is impossible to add any "bended" strings to it. This would have to be true & I think it is.

A sort of proof of ≤ 1 : Consider all of these strings as inputs to a TRMc (not nearly UMc), that cause it to stop. $\sum \leq 2^{-l_i}$ is known to be probability of indep events. — which must be ≤ 1.

.06 One problem is: Given R. dom. off. set of strings of 241.17-.34 !

does there always exist a Machine that will stop for those & only those pgms?

.08 .09 Re: Hrr, the Q of 241.00 (Re: Sol64a) : I think that

0 this is definitely a proof of convergence, because the $\sum 2^{-l_i}$ sum in the numerator of eq.(1) of Sol64a (& 9 perhaps) is over a subset of all pgms that cause the machine to eventually stop. The sum over all stopping pgms is the probability of stopping: which is always ≤ 1.

12 .17 Rereading Sol64a pg 9 : eq 2 is explained! — in the sense that it says that eq 2 diverges! So I even really don't need eq (3)! The eq (3) is wrong.

.66 It says "Unfortunately, it can be shown that the no. of bases of length in

.67 of T_α (or any other seq) is at least of 2^m for large enough"

I think this is false. — since eq.(2) can converge.

.69 Perhaps the explanation is a notion p. 9 of Sol64, Upper Rel. Prob.

0 If the machine printed x then stopped, with α as input then α regarded as to be a code for x, then all possible finite b. In this case, .16-.17 is

22 true. So it's probably what I meant.

Anyway, doing it the simpler way and considering the "total input to which the machine stopped" as the "length of input" gives the desired "Kraft inequality".

So with this "extended Kraft inequality" and the discussion of .19-.22, I think I understand Model 1 of Sol64 rather well!

→ I still don't know whether Model 1 is same as model 2 (say Model 1 = Cover's Extension Probby)

Also, I haven't really proved the Extended Kraft Inequality (.06-.08), but the likelihood of its truth is very close to 1!

.29 A possible comment that may prevent it: Say we only consider the $>\phi$ pairs only of the set of segments - would that perhaps have to contribute a prefix set?

(two omit duplications of segments originally having negative parts)

If I can prove this (it seems likely/easy) then it follows that the negative extension of each part that has a negative sum, will have to be a prefix set & satisfy Kraft ineq.

If (& conjectured) the case of 241.32 is illegal then I think Lemma (.50) follows & the theorem about the Kraft inequality holds. → see 248.00 for confirm →

23

D.28.05

5 TM

239.09
0.0 : 239.09

Parallel w.t. Study, Categories, Enumerization of Heuristics. I can be working TSQ's in daily way: First make a rough [redacted] seq. of problems that would seem to be useful "subgoals" of a TSQ. Next write solns. (In English) of how a human might solve these probs... using "human" heuristics. Try to identify just what heuristics are needed. Try to find ways to learn these heuristics, what other problems would use these heuristics? If TM were given a soln. to a certain problem, could it (somehow) Generalize to obtain heuristic used in that soln?

07 Look at AM at greater length. What heuristics were used. — How implemented —

(See 232.13-.18 for some Discn of AM) Try to get Paper: "Why AM seemed to work - - - " by Lenat et al. Security connection: Paper on CI's broken AM

10 (This is a Bad Page! Difficult to Read. $\frac{1}{4}$" erring is only 108 smaller Pencil.

11 other Wild paper — but Faint & thin lines cause trouble! ⇒ 2 44 00

20

30

243.11 Perhaps try teaching Arith. before Alg. notation:

First Counting Objects: This involves 2 recognitions ① Recognize object type (orange, say) ② recognize that 2 oranges differ from that orange. Assoc numbers w. different County results: Then learn "addition" and how it is related to counted sub sets.

Maybe idea of a finite set of objects w. certain defining characteristics & ways to tell them apart. Then assoc Number w. set down as "locus" of set of objects that are "set" candidates.

In AM, the machine was motivated to define & investigate concepts that were "interesting". A TM working hard problems should spend a fair amt of time on this sort of thing as part of "self improvement". (Say, spend ½ of its time this way — "the 50% solution"). Then Nov. TM would try to fuse these concepts to its "Production Problems" — which makes it quite different from AM, which had no "Production Problems". (20)

(5N) Adam: Maybe write at least 1 pp of TM before doing any thing else! (Maybe coffee first .... possibly lunch even) Try to continue (past) pp. 1 pp. Ahh!

If I have no good ideas: Then write Review of "State of TM" — describ. most impt. problems. List them, tell why diff, list possl. approaches ... (& list of approaches to problem solving).

Re Counting and Addition. Say concept A defines 1 set, concept B defines another set. Could A & B be mutually exclusive. Card (set A) + Card (set B) = Card (Set A OR B). The Mutual exclusivity of A & B is . Seems like a complex concept, hm! The non overlap of A & B || A & B have no common elements. Card (A AND B) =)

So if (Card (A and B) = 0 Then Card A + Card B = Card (A OR B)

More Generally: Card (A ∪ B) = Card A + Card B − Card (A ∩ B)

(.13) ± This is also probabilistic thing.

Subtraction must be understandable after Addition has been taught.

What about Mult? It must be instructive to see just how AM discovered numbers, Multiplication, primes, etc. What Heuristics used? How used each. (23e.13ff, 24B.07ff) on AM

Actually 2 "TSP's" to get general ANL:

① T. Saves. soln. using a Machine that made this long easy! That it was easy, suggests it more likely that related (long) will be easy, so it really notes A.H. as it sounds

② learns 3+4, 3×8, 4−6, ... etc. TM is able to do it probabilistically w pc = .25

Next it studys cases: notes " + corresponds w. sum " etc. — Why then gives

STM A. Barron, 1993 is the neural net paper using Lisp, Sexpressions, tanh, etc. See 013,96 | 5.25.95 (on CaSD (dumps of D in ty)

100% accuracy! Next (as number): a) the idea that strings enclosed by parens have "values" (usually) b) the idea of substitution of values in expressions: c) the "Great Leap" that once we have "≈" between numbers, it might be good to try substitutions of values for (parens) in expressions. This would successfully complete general ANL. (w. much diffy$, burr.). We would also end up w. idea of "value" & "substitution" — both very useful ideas.

SN Max Common Subgraphs: Sys, Man, Cyb Oct 96 p785 Has several relevant References

Reed Scothon
Eurisko, Cyrano

IN① (Saab ANL: 244.31), to emphasize importance of various solns. Give TM many probs w. that soln. It will assign of the solns as well as of its sub-trees. Otherwise, (esp in PPM) the solns of older patterns would not have by-and-of p.c. so their use in new probs would have very small p.c., i.e. (dreaded) rapid ↓ in p.c. of solns of progressive problems in the TSQ would be overweighting!

It would be of interest to see if Saab TM (w. PPM) could form useful sub trees. I probably would have to segment PPM in tree-search — whether PCFG would be able to discern & needed patterns/subpatterns in the corpus of solns, would have to be looked into. So it would be a good idea to actually rewrite the old Saab ANL soln. & see if useful sub-trees turn up. GTM 2.00

STM Rev. 220; 226 Wave previous PP in this seq

50 226 34 22) On Normalization ② Rat-poof that $\sum_{n=1}^{\infty} U(z)^2 < 2 \ln pc$ of Γ^{pc} (also see 20)

b) That my method of Norm gave me ~~Sq. error~~ Sq. error ~~Fund~~!

Nord work. c) ~~Norm~~ Norm conj Brat was bounded! very probably false!

d) $\sum_{n=1}^{\infty} U(z)$ was bounded ← probably false! Paul 8/24/91

[Hvr. see 557.03-06; 558.01ff for counter Arg'ts. Hvr 1991 is better text Editor of Li-Vibany!

SN 96 TM 3 3/5

Has ~~a~~ secondary refs to some current ideas.

1) Hvol of 4 spheres

2) pc order of strings

3) 2pm finding doesn't work, no 3pm. finding.

4 "Riss" matrices Nyms

Bulg 2000; 187: on t. ob-op Algebra

23) "The Encyc problem" (2002 IDSIA) (2002 584.33 - 585.24 Baban)

Also Relevant to a problem in SM. of choice of Stock to play.

24) That Code that Alan Tritter wrote for me! on ~~FE~~

Matrix Power: Sun Feb 15, 1959! TMB 242.1, 242.2

This stuff is probly now "Comman knowledge"

25) 12 Problem Solving techniques that I want to write ~~grammar~~ for:

Those are not what I was looking for! ~~Zeitgeister~~ TS 7/21/80 p 172 ← Actually, this is a 12-item TSQ for "MTM"

TS 8/2/80 p 181 ← This was actually ~ to ↑ but more TSQ problems.

That to be more "Creative" problems.

TS 1980: PP 146 & ~~148~~ 548 have numbers on TSQ's,

I may have made a copy in ~~SAAB~~ Papers "Paul" 1990-(99)

20

26) A v.g. Soln. to the Zero frequency problem. (This is within last 6 months... realized very recently)

30

5TM

Lenat, Brown

ON AM, Eurisco, etc. See my notes on "Why AM & Eurisco seem to work" from AAAI 1983 R236

IN "AM": He has huge list of ~110 concepts to start. (a Conc. can mean either a General Abstraction or a "Conc as defined in Computational Lrng. Theory")

Heuristics has ~ 200 Heuristics to start.

Each Conc has a property list (≡ set of "slots").

The Goal of AM is to produce "Interesting" Concs. It does this by operating on existing Concs (Mutating them), using the set of Heuristics to evaluate the new Concs by an evolving "Worth" (AM has some Heuristics that evaluate the likely Utility (≡ interestingness) of a newly created Conc.) Then it is a Potential list of Concs & Given a property list. — Some of the properties need not yet be filled out.

So: What AM Does: It operates on the old set of Concs to produce more Concs; which are evaluated for "worth" & added to list if "worth" is > Threshold.

To decide what to do next, AM has an "Agenda" which is a list of tasks w. a "do" param for each. The Task in the Agenda with the highest "do" param is done; it may "do" param and that param is > a certain Threshold (?)

"The Book" (p 205 Appendix 3) gives an example of AM working.

Actually AM has 2 kinds of tasks on its Agenda (task list)

1) filling in or developing further a property of some Conc. (or combination of Concs?)
2) Devise a new Conc.
3) Apparently, it cannot delete an existing Conc.

1.6.06 **A Q is:** Just how AM does 1) & 2)? It has this set of Heuristics. I Guess that whenever AM does some task, it has a hunch that gives some tasks for the "Agenda".

Relationship of AM to GPS: GPS has a single Goal: AM has a "single Goal" but it can be regarded as satisfied in very many ways. GPS looks at present & Desired states, Then tries to find a way to reduce one or more Components of the vector "Difference". AM always has same Goal: "The difference betw. present & desired state": Unclear as to meaning/usefulness of this idea! AM is more of a Multimodal Hill climber.

W.r.t. TM's Goals: usually monomodal hillclimbing! (i.e. single Goal). What AM does would be useful for TM in "Self-IMProvement Mode" The Criterion for "possible "Useful ness" ("interestingness") is rather difft/obscure. : 6TM3.20

SW It may well be that Lenat's book & recreate AM! If All the Heuristics & most of the initial Concepts were given And we have the Agenda + meaning for computing worth (cp. p.4. p 32.)

03/05

5TM

≠ Bob Tanner

last page of 5TM

(248) last page of 5TM

242.35: The dem of the limitations on the segments constituting the set:

.01

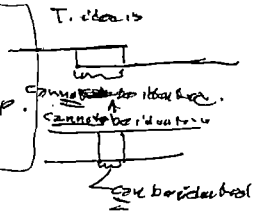
.02

.04

If two segments have subsegment α in common

I.e. say A & B are 2 segments! Then $A \cap B$ is their overlap.

A & B cannot be identified in subsegments $A \cap B$.



This may be a complete dem of the "extended prefix set" - the

$A \cap B$ must of course, contain the pt. ϕ, because all segments contain that pt.

I have to show that this cond (Ξ 241.32) is correct for the Model 1 input.

Proof: Say α is the overlap region. We started @ square ϕ. If A & B are both $> \alpha$

Then the read head must break out of α at one end or the other. If it breaks out at one

end, it can never break out at the other, because it would then be a pgm for neither A or B.

However, it can break out on either cond A or B, But one or the other.

I think this proves 241.3[illegible] $\therefore$ 242.30 - 35 $\Rightarrow$ the Thm.

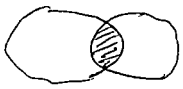
This can be gauged to Multidimensional "segments", by considering a Trmc with an input (address in several dimensions); it can move 1 unit in any direction before changing state". We consider sets of pts visited before stopping. The set of such pts. is a segment. The no. of pts in the segment is the "Length" of the segment.

Then $\Xi 2^{-k_i} \leq 1$, because this is the pr of stopping w. random input.

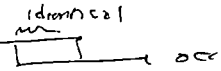
This Ξ can be computably since the Trmc need not be universal!

So just how can we characterize the "Legal overlaps" that would give a certain "stopping" pgms?

One restriction: no segment can be completely covered by another.

What about  ? in 2 dim.: well in 2 dims, the regions need not be characterized by simple boundaries. The head head only wander about in various paths. (All segments must have at least one common pt. - "$\vec{\phi}$")

We'll consider all paths (that include at pt of $\vec{\phi}$). They are linear objects so B. Gauzed knot-eq. holds for them. The sections of 2 paths that contain the pt of $\vec{\phi}$ cannot be identified. Hm., notice error of (.02 = .09)!

Another view of a prefix set: If the 2 segments like  occur,

identical

there could be no "unique parsing" of a set of such segments.

Do Consider Wtape & Ndim bdd tapes. for Kraft inequality

Consider sets of segments that are parsable/recognizable.

→ 6TM 1.00