Can't find any earlier stuff on TM in 1998! Could it be that I wrote nothing specifically on TM up to now?

4.6.98 TM: Lsrch: Optimality:

T. idea of "Lsrch is optimum for Inv probs if all info is in t. P.D." : T "proof" is — If one finds a better way to solve t. problem, then this "Better way" was not in t. P.D. so "all t. info was not in t. P.D."

[A similar argument can be made for OZ problems: If a better way to solve t. ~~prob~~ OZ problem has been found, than Lsrch w. a given P.D. — then clearly that P.D. doesn't contain that "better way" — ∴ so one must have had "info not in t. P.D.".]

ABcd.

So t. Q is: (1 can one put all poss. heurs (soln. methods) into t. P.D. w.o. signif. loss of info? (2 T. P.D. for OZ & Inv. probs seems diff'nt.

— What P.D. do I mean?

— Well its may be possl. to unify them.

for INV probs, we want a function that **maps** t. prob. dscn. into t. soln. strings:

INV. prob → M_v (Problem, n_j, Soln) → we want s_j ∋ $Prob_T(s_j)$ ∋ n_j gives shortest path & proof that it's shortest.

Trav. Salesman problem — Dscn of cities & dists. betw. them.

(I need to fill out details of this)!

But anyway, say I had separate P.D.'s for OZ & Inv probs: Since they are conditional p.d.'s I could have a single Unified P.D., if I prefaced each problem w. a symbol that told whether it was a Inv. or OZ problem. This enables one to share abs. that are useful in both Inv. & OZ probs.

If there is only 1 p.d. for OZ & Inv. probs: Then its likely that practically any Heur can be put into t. form of a modifn. of t. P.D.

,30 Sol 86: p478: T. P.D. is a conditional prob ~~relating~~: T. cond. is t. problem

[Inv. or OZ problem --- Inv. is M( ) and x: (to find p ∋ M(p)=x) OZ is M( ), τ; to find M(p) ∋ M(p) is Max given time limit, τ (real)].

So: for Inv. probs, M( ), x are t. conds: for OZ probs; M( ), τ are t. conds

[for "Anytime" OZ Problems; M( ) almost ~~completely~~ describes t. problem: We may ~~also~~ also need τ_0, a lower bound on t. available time] — So, for t. General p.d., t. "conditions" are

$$\begin{pmatrix} M_1(\ ), x \\ M_2(\ ), \tau \\ M_2(\ ), \tau_0 \end{pmatrix} \bigcap \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

[here t. 1, 2, 3 tell what kind of prob. is to be solved.]

or

$$\begin{pmatrix} M_1(\ ), x, 1 \\ M_2(\ ), \tau, 2 \\ M_2(\ ), \tau_0, 3 \end{pmatrix}$$

See 52.01–53.40 one way to express "cond'l. prob" is as a stochastic operator: "conditions" are input; t.d. is output.

$M_1(\ )$ has strings like x as output $M_2(\ )$ has reals as output.

So we can have a single proby distribn for all 3 kinds of problems. → 2.40 (only partly! spec.)

4.6.98 TM Lsrch optimality.

optimum Seq. Srch

I also want to make the mechanics of t. Lsrch clearer! In t. OSS report, I used t. prefix set α_i to assign pcs to t. trials, but any probability (or even a semimeasure) would be o.k. (All I need is $\sum p_i \leq 1$ & $p_i > 0$..)

Also, I want to give more detail on t. form of "heuristic info": how it's based on "traces" of previous work done by TM. Expand that F.N. in Sol 89 on this.

After writing more on t. GA problem (or SGA problem), I will want to integrate these approaches with Sol (86, 89)

In 1.30-.40 I showed how t. P.d.'s for t. 2 (or 3) kinds of P.D's could be integrated into one P.D.
This same sort of "condl. prob'ty" could be used to express other kinds of info that TM has.
— e.g. say TM has a corpus about t. R.W. or part of it. It could learn to make predns about that corpus thru a P.D. on it.

Similarly, w. various other kinds ("Modes") of training, we can prob'ly express them as P.D's

.19 and make them part of t. "GRAND" P.D. → see 4.30 to 12.30:

---

3 mos: ½ yr

.20 SN Did I not write anything on "TM" in 1998 - up to now? — It is possl. —

I spent lots of time on t. SOY problem as applied to SM FT.

97 SM 222.32 & t. following stuff was a main line of investigation — applicable to TM as well as SM.

I looked at all "Nite notes" from jan 98 to present time!, There is no real discussn of "TM" other than t. "SOY" problem, as applied to Fast Track.

6 yrs
× 200/5 = 40

50 → 200 By.
40 ! 6 yrs.
1.25/yr.

4.7.98 TM: Gen!

I want to outline what to TM path is a list of most impt problems.
I did a long run of TM work on this maybe a yr. ago or less: what was it about?
Perhaps Associated w. criticism of Jurgen's paper ("Lrng how to Lrn")

4.8.98: Some general meandering:

1) The 3 Theorems in Sol 86: I could write proofs for all of them: Th. last one in particular (Met (now using Tree Sharing Lsrch), t. method is probly within a factor of 2 of optimum, if we use t. "log soln". The proof of t. 3rd Thm is indirect: If one has a better way to solve t. problem, then one hasn't put all of one's info into t. p.d. ——— which violates t. "Theorem" statement of t. problem. ——— So t. "Theorem" is trivial. Its an explanation of a methodology, rather than a real derivation of a new fact from old ones.

2) Obstacles to tsq. construction: I could just use any elementary text in algebra for problems. If TM couldn't solve one, I either give more background problems for it to learn, or give "hints" (Modifns of t. problem that make it easier — or tell TM. t. soln. either by giving him very strong "hints" or by giving t. actual codes to be used. — as if TM had spent t. enormous needed time to Lsrch t. soln.

•21 (b) On "putting things in memory": I have simply added goals & probs to t. general memory pool, retrieval becomes very costly — (Not much >1 memory access is feasible per problem). ——— Anyway, we have to have a better way to access info in memory. By "association" or other index mechanisms that I & TM discover..... work in "Info Retrieval" is perhaps useful here - - - - but I will want some more very clever ways to retrieve things: by analogy (which is a rather broad grouping!) as well as simple associations. Recency could be an impt. bias. How do we humans categorize things so that retrieval from our memory is facilitated?.

Also EMPT: when we put info in memory, we must try to think of ways it is likely to be useful. & index it accordingly

Whenever we want TM to access its memory for something: try to think of ways that t. particular "correct" object is more likely — why would I think of that object much more quickly than most of t. other objects in t. memory?

If we have a soln. that was frequently used in t. past, what were t. surrounding conds. (or even global conds! like "it's a chemistry problem") ? → 17.0g

•36 On Analogies! When TM solves a problem o/o compresses some corpus! It can express results as a structure acting on a set of objects: The structure (= perhaps t. symmetry of t. objects), is stored in Memory. When a new problem w. similar structure occurs, then we expect analogous associated info! We want to make it easy for TM. to recognize that t. present situation is an instance of a previous situation. Then

Situation into ~~a str~~ one (or more) [str # appl. set]. Previous Strs. & appl. ~~sets~~ sets have been indexed in Rem for easy retrieval. When we put a prob. soln into "STORE" we also store w. it, the "structure" of the problem & other useful "factorizations" of it — to facilitate retrieval.

Another Source of Try. Sgts. is Kozza's Problems. Kozza's work is of interest in several other ways: One, is a test to see if my Lsrch formalisms can (at least) implement all of his tricks.

Another, & reverse, is that OZ problems can be automatically turned into a kind of TSQ using an INV kind of "Hill Climber" or — perhaps ~~other~~ using SGA.

((Is it in General true that any OZ problem solver is equivt. to a kind of TSQ? — even if we don't use the INV. approx. to solve it >

SN As a part of normal TM work: Write Jugen! (2hrs/d.) A good thing about studying Jurgen's writing him is that it forces me to examine my own methods & define them more exactly. → see 9.01

topM → This is a ~~idea~~ for getting the Sci Community working on TM!

.12 Find various Sections of TM: (i.e. initial state (= P.D.1) ~~desired~~ final state (P.D.2). This ~~can~~ will usually take the form of a/ particular problem solvd

Some "Sections": Developmental Euc, ~~&~~ a/o Projective Geometry: Algebra; Symbolic Integration (Find out about recent advances & how TM "combines"). Various sections of Meth (Bias toward parts of Math I'm familiar w). Ability to devise new formal langs, fit data to langs.

How to do development: Take standard elementary texts & the tsq's they imply. Then fill in the needed heurs. & other concepts. I should try to devise reasonable tsqls for each one. I put in, but not tabs. nasty! → 6.20

.24

.25 1.40 2.19 Also do some kinds of H.C probs! Ones that have been done w. GA. My ideas here are a bit vague, but I idea is to develop the OZ solving aspect of TM: Something like a P.D. ~~that is used~~ of the H.C. pgm that is a condl. proby of the problem given. → see 55.12

Input idea

Anyway: we publish a fair no. of papers that are examples of how to solve such problems (using Lsrch). Then we try to integrate these sub TSQ's into a single TM, that is then able to use all of the info obtained from all of the TSQ's. We could then issue problems to the Sci Community of the form of .12.

.36 Two ways to do this: (1) Just describe probs & ask for solns, to be added to TM. TM is available on the net, so anyone ~~can~~ will have available to him, the sum of all of the contributed TSQ's. Each month (say) we will put on the net, the latest ~~combined~~ integrated form of TM. → also see 5.04

Operation ordinary nasty.

Seed E

Bottom P2:

Back 3 yrs forward 15 yr.

Mat NOL

Operating

Losses.

pb 5½6

from ERS: /ss/p536.pdf

.01 (2) Another way: We give monetary rewards for solns. to problems — so the best soln. gets the reward. We could give all contries reward cc $\frac{pc}{cc}$ of their soln. — Total reward is divided up w. those wts. (Maybe normally small sums of money: except to grad. students)

.04 We publish winning solns w. names of authors & how much they won.

~~Prog to~~ .01 has some dirty in defn of cc: Tho we could define it wrt. a particular machine (say Pentium or Pentium II MMX).

I'd like to be able to do both .4.26 ① & .01 ② : ~~perhaps not~~ perhaps only have prizes for people who ask for them. (e.g. self-derbd "Grad students") (or prizes may be Bittleast Goodwins for non-grad stud., Money for Grad Stud.

SN Re: Lsrch always using "only" the single best soln: — (even after one ever solves" the probe in the hope of getting solns. of higher pc.) Graca suggests that in Backtracking one will normally try "second best" local maxima. Is there a way to use Ra/Id incorporate this into Lsrch?

It may be that this sort of thing ~~should~~ could be done by TM_2 ("compression of the P.D.") "Compression"

[On the other hand, may be I'm confusing this w. another problem: i.e. in Lsrch, we always use the cost of a soln. rather than pcost. If we had a way to assign pcosts to cmcs, then we could do Lsrch using these pcosts — & if we could make an approximate ordering of cmcs in pc order on (better yet in $\frac{pc}{cc}$ order).

Various cmcs could be labled w. their pc approxn. — we don't have to always use cost to estimate pc. For certain elementary cmcs, pcosts is perhaps o.k., but for any cmc. that we learn (or for ~~sub-routines~~ functions that we discover as subparts), we can have pc values not of the form 2^{-N} (or normed $2^{-N}/(2^{-N}+2^{-M})$)

$$= \frac{1}{1+2^{\pm N}} \cdot \left[\frac{1}{1+2^{1}} \quad \frac{1}{1+2^{0}} \quad \frac{1}{1+2^{-1}} \quad \frac{1}{1+2^{-2}} \;\middle|\; \frac{1}{1+1} \;;\; \frac{1}{1+\frac{1}{2}} \quad \frac{1}{1+\frac{1}{4}} \middle/ \left(\frac{1}{3} \quad \frac{\frac{1}{2}}{\frac{1}{6}} \quad \frac{\frac{2}{3}}{\frac{1}{6}} \quad \frac{\frac{4}{5}}{\frac{2}{5}} \right) \right]$$

After we find several/when solns to/each several distrit problems: we want to "compress" the P.D. so it's more likely that these solns would be found directly (or more directly w. less Lsrch — that the components of the solns. would have higher pc's than "normally").

Here, I think we want to derb each problem in many ways: for a human, ask o easy: where did the prob. come from, how was it derbd in human lang, what did the linguistic components suggest ~~etc~~: Analogs in other problem areas. This is the machine version of the Human heuristic: "state the problem exactly": i.e. for the machine: "State the problem vaguely" — implying various (analogous related) ideas, related & analogous problems.

4.12.98 TM

I could start out by fixing up those Ts. Q's that I wrote:

.02 1) Alg Notation Lrng;

.03 2) learning laws of Algebra.

3) learning to solve linear, then quadratic, then Cubic eqns easily using a single (impl.) heur.

4)

My main complaints about these TSQ's were

a) in (1) (.02) the lc of soln. began pretty large because pc of access to info in (Store Many) was too small! (see 3.21 for a promising approach)

b) T. only useful things we had in store were complete soln to previous problems.

I think this could be fixed! Say we had a (later) problem that did use subfunctions (or "parts") of solns. to earlier problems; we could then modify the entire system so it could put that sort of info in store & suitably index it.

See 12.01 —ap for How to do this (to some extent)

.20: 4.24: Along w. this: I could put in partially solved problems. These would be problems in which parts of the soln. had excessively hy cjs. We could "fix" this by adding a heur. to the "primitives" or by suitable augmenting of the TSQ so that heur would have a reasonable pc. Also, we might have "rules" in which a human had to explain/parse the text of the problem to TM, or even give TM gross "Hints".

In (.03) ... "learning laws of Alg" One thing I didn't do, that I should have: was to teach it to learn "if a=b & c=d then a+c=b+d" It could learn this for pure literals, then later, for mixtures of literals & nos., — so it could learn to solve eqns.

Suppose we gave it eg. $3x+7=3$ find x & it knew. We could give it many examples & it would find rules, perhaps by exhaustive search. — But eventually, after it had many examples, it would find that the general rules gave answers much faster. Similarly it could learn to solve many other types of eqns. Then, after it had all these soln methods in terms of literals in the eqns, it would try to find ways to obtain the rules for a new type of equation form, w. less lc. In this way, it could, perhaps, discover "laws of Algebra"!

But if it could learn these rules just as fast w.o. learning "laws of alg" — then "laws of alg" are really not necessary to understand how to solve these eqns.

4.12.98 TM

Another approach would be to Tell T.M. t. "laws of alg." & give it lots of probs in which it had to use them. Some of t. probs would be proofs of Thms running out of

.03 "t. laws of Algebra" - → 8.01

AH! I finally remembered how to get cubic! ← woopsy

Say $y^3 + ay^2 + by + c = 0$ Let $y = x + \alpha$

$x^3 + 3\alpha x^2 + 3\alpha^2 x + \alpha^3$
$+ a x^2 + 2a\alpha x + a\alpha^2$
$+ b x + b\alpha$
$+ c = 0$

$= k$

we can make it coeff of x^2 be 2 by suitable α $= -\frac{a}{3}$

So $x^3 + (3\alpha^2 + 2a\alpha + b)x + k = 0$

+k

e.g. can by

No!

$y = x + \frac{\alpha}{x}$:

$x^3 + 3x^2\frac{\alpha}{x} + 3x\alpha^2\frac{1}{x^2} + \frac{\alpha^3}{x^3}$

x^2 0 $\frac{1}{x^2}$

x^3 x $\frac{1}{x}$ $\frac{1}{x^3}$

$a x^2$ $2a\alpha$ $a\alpha^2\frac{1}{x^2}$ ← 2 (=0 ?)

$x^3 + c + \frac{\alpha^3}{x^3}$ bx $+ b\alpha\frac{1}{x}$ ← b

$+ c$

x^3 ↓

x : $3\alpha + b$

$\frac{1}{x}$: $3\alpha^2 + b\alpha$
$\alpha(3\alpha + b)$

So if can similarly get rid of x & $\frac{1}{x}$:

$x^3 + ax^2 + a\left(\frac{\alpha}{x}\right)^2 + \left(\frac{\alpha}{x}\right)^3 + c + 2a\alpha = 0$.

$(y^2 + \alpha y + \beta)(y + \gamma) =$ So if $3\alpha + b = 0$ we get this & 2 cu. fts.

$y^3 + \alpha y^2 + \beta y$
$+ \gamma y^2 + \alpha\gamma y + \beta\gamma$

$\beta\gamma = c$
$\alpha + \gamma = a$
$\beta + \alpha\gamma = b$

$x^6 + a x^5 + a x^4\alpha^2\frac{1}{x^3} + \alpha^3 + (c + 2a\alpha)x^3 = 0$

6 5 3 1 0

$a = 0$
$\beta\gamma = c$ $\beta\alpha = -c$
$\gamma = -\alpha$
$\beta - \alpha^2 = b$

$x^3 + (3\alpha + b)x + \frac{(3\alpha + b)\alpha}{x} + \frac{\alpha^3}{x^3} + c$

x^6 . x^4 . x^2 $+ cx^3$

a, b, or $c = 0$

Boyt-Loch in 1989–1991 "TM" or "Paul" notes. → Summary! t. subs. $y = x + \frac{\alpha}{x}$ is adequate.

$x^4 + ax^3 + bx^2 + cx + d = 0$

$(x^2 + \alpha x + \beta)(x^2 + \gamma x + \delta) = 0$

a b c d

$x^4 + \gamma x^3 + \delta x^2$
$+ \alpha x^3 + \alpha\gamma x^2 + \alpha\delta x$
$+ \beta x^2 + \beta\gamma x + \beta\delta$

first get rid of ay^2 ($\Leftrightarrow a = 0$)

Then $(3\alpha + b) = 0$

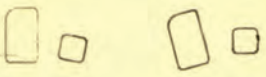
So $x^3 + c + \frac{\alpha^3}{x^3} = 0$

MERGING TMs'

.01: 7.03 [redacted]: Note that t. "laws of Alg." begin to become useful when one wants to solve eqns. w. unks in them!

[redacted] So one can discover new facts about "unknown" nos.

Another good way to organize, speed up work on TM: Take certain Expert Systems & have devise try seqvs. for TM so that he ends up being able to do (at least) what these "Expert Systems" do.



.08 One t. "Merging" of TM's: Say we have several different TMs that started w. t. same primitive abs. sets, but were given diffnt. TSQ's — so they are now quite different TM's. We know t. TSQ's: Can we merge these TM's by alternating items from t. various TSQ's into a single TM? Maybe: But it would seem to take lots of cc, since t. search space is much larger than that used by t. original, "Unmerged" TM's.

Another way would be to create a new TM that had the PD's of the older TM's: These PD's would (initially) be present unmodified, w. maybe equal wts: So if $P_1(\vec{x})$, $P_2(\vec{x})$, $P_3(\vec{x})$ were t. 3 (say) original p.d.'s, t. new one would be $\frac{1}{3}\left(\sum_{i=1}^{3} P_i(\vec{x})\right)$.

As this T.M. matures, it gets more & more "coupling" betw. t. 3 P.D.'s — until they are integrated .08 could be very impt. if we have a community of scientists working on various TM's, in parallel. Each group would have its own TM.

Anyway, t. first step (for me, as well as for t. community), is to publish many papers showing how to apply Lsrch to many problems: Also give examples of TSQ's.

→ Alternatively, w. each $P_i(\vec{x})$ we regard it as a ~~fail~~ "conditional" prob'y, and we have some way of recognizing when a problem is of "type i" ($i = 1$ to 3). At first, we can do this by telling TM which set of probs it is in. This would be equiv't. to no "merging" of t. 3 TM's. We could gradually give less info on what t. problem types are, & let TM figure it out. Eventually t. Final TM should find ways to Merge t. 3 P.D.'s

4·14·98 TM : Jargon

97TM 174 ff has t. beginning of a letter to Jurgen.

Some comments: (1) I'd like to make it clear that I do understand his positions & appreciate his positions.

(2) That I should give my own positions & reasons for it (which don't nec. make J's methods bad).

(3) Begin Letter on a rather hy Abstract level: Get down to details later.

(4) Note 97TM [redacted] 131.01 (on differences betw Me & Jurg. on Goals, Methods.

(5) He is concerned that my dividing up t. TM process into "prob. solving" & "Self improvement" is not clearly optimum. or well defined — I should discuss this.

(6) T. prob He's aiming for, is more General, more diff than what I'm trying to do (i.e. "Scientific Assistant") — But I expect t. solns of my problem will help tremendously in solving his.

(7) Actually, my main problem is not this letter to J., but some thing close to it; i.e. Making a general plan for my own. research on TM. 97TM 22.01 ff. was in this direction 22.01

(8) Caution: An initial effect of his paper has been my [redacted] questioning, revising my goals & expected path for TM.

(9) Perhaps start letter w. list of impt. ideas that were agreed on. Thus
(a) That dividing work up TM into "prob solving" & "self-improvement" is usually non-optimum; Its best to give machine a single "top Goal".
(b) It is better not to have to give TM a "Time horizon"? (I may want to say this, tho I understand his desire to do w.o. "time horizon")

AA
≤ 97TM 189
Actim
Algm

See special file of Jurgen:
1 2697, 8998, 8998) 3pp

4 2298 TM:

A Somewhat New Approach to "Trancy" TM:

"Its easier to lrn a new lang. if one already has an old one": TM starts knowing Machine lang, or some very simply structured lang. (not nearly a "complete" lang) ! Lisp, or Prof Punctuated lang. I invented, or Forth or whatever. (A Rise (reduced instr set Machine lang.) → 6502 ? or some other simple instr set.

We can start by telling TM various facts — about Numbers & other objects Plus lang. deals w. normally. (If its numbers, we might try to teach it the 2 text books Ramanujan used!). The facts could be about Number Theory a/o Algebra, to start.

"A/o: About relative positions of things in 1 dim, 2 dim, 3 dim, 4 dim (← Time (or 3 dim w. special, temporal) It could be a "Blocks world" type of knowledge. We may also have it know about Time, & how events, ("objects") can be ordered by "time" (≈ a 4th dimension).

Perhaps even dynamics of objects. — trajectorys, physics, etc. (perhaps look at Karl Simms work on Teaching Machines "physics" of R.W.

After lrng. about a world in its initial lang. — we have it lrn to understand equivalent statements, Q's in a simplified simplified English. (e.g., no pronouns, few ambiguitys, only 1 way to parse a sentence, eck) — Or, have it lrn some diffent. computer lang(s) .... Then simplified English.

After lrng. English, we have it deal w. R.W. situations close to the world it knows about, & we gradually expand this newer world, & we gradually make it lrng used to talk to it be closer to English. — Also as we continue, the "English" we use becomes noisier & noisier. — More & more errors in spelling, grammar, more pronouns, more ambiguity in parsing, ect.

.26 There are many potsh paths in {language, Universe of discourse} spaces that we could travel, & end up w. a machine understanding English & (part of) the world of Man.

Our main goal is to get it able to usefully cruise the web. It probably could get enuf info to be very useful w.o. being able to understand graphic material.

Instead of the web, perhaps use Brewsters Encarta "snapshots" of the web as a data base! In addition it could have many v. gd CD roms, like Britannica, McGraw-Hill Scientific Encyc, etc. (Encarta Britannica is < 1 GBy: I have 7.2 GBy HDD! So it should'n be hard to get enuf info on a H.D. for TM's later "rsng".

.35 I should explore (informally) many of the paths of .26 before doing a serious study. Perhaps show how other approaches to A.I. are "special cases" of it. E.g. my older (Sol 89) approach; CYC; etc; Blocks world (expanded to arbly complex physics)! Karl Sims's world for his "creatures" w. some laws of physics, time, Gravity...

Lenats "AM" a/o "Eurisko"?

5·10·98 TM
6·14·98

I would like to be able to "divide up" the TM problem so several groups of people could work on it independently in NL.

One idea involved the "merging" of TSQ's for different trained TM's.

Another: We train a TM to do induction (problem solving) in a certain area — but TM's knowledge in this area is incomplete: Many of the cases it uses have been "memorized" (or "learned") or obtained via Gross "hints". Devising TSQ's so that TM can replace each of these "unlearned" cases (or "inadequately learned cases" or "imperfectly understood cases") with the one that is properly trained via a proper TSQ.

6·14·98: Another exercise which I discussed in the past. Given 2 TM's, initially the same but given different TSQ's, so they are now quite different individuals — we can to some extent, combine the 2 TM's by raising a Third TM on their "learned" TSQs. So we combine 2 TMs by combining not the TM's but their TSQ's — TSQ's are usually easier to combine than TM's!

5.28.98 : TM:

How to teach a machine that can learn.

alm Ross, Wallace.

On Sept 9, 98: Gramm wants me to be [illegible] spkr at Computational Intelligence colloquium 1 day

Poss. talk: How 'The teaching for learning machine

How to teach / learning Machines.

Teaching Machines that learn. (NO)

Why it is a problem: Some approaches.

Discuss ways to break up problem so Sci Community can work on it in ||.

May be try to publish in "Machine Irng".

Teaching robots machines.

6.5.99: TM & expo: The teaching of machines and of humans.
How to teach machines and humans.

Title for London talk: How to teach The Machine — Let me count the ways.

Subtopic: How teaching Machines are different/similar to teaching Humans,

1) Teaching by neurosurgery, Logic (expert systems)
why / why not. (or when, when not). Success Chess, Symbolic Integration...

2) Teaching by example: How this can differ much between humans, machines
We can give machines much detail of example.
For humans, the "example" is much vaguer: Humans are forced to do much more induction

---

Q of difference of 2 sessions: that Inductive Inference & Machine Learning are essentially the same problem.

Relation to Human learning:
1) Good Models
2) We can use these models to make good teaching Machines — that make model of student's understanding & propose problems & hints & study ideas (things to read, things to read about, interesting facts ... — that are based on that model.

Main part of paper: listing & analysis of various ways to teach machines — One one extreme: Most Expert systems —
On the other extreme: Getting a machine to do data mining w. minimal a priori ideas about the corpus

Much work in ML is oriented toward problems arranged so that it's easy to tell if machine is lrng.

Division of this field into M.L. & Ind. inf. is rather difficult! Any practical method of induction amounts to machine lrng, but some ML methods translate only w. great difty. into induction (ordinary functions) 90.234 M Freq. 6283356

How to teach a machine — let me count the ways ..

Many techniques have been proposed and attempted, to bring a general purpose computer from an initial state of ignorance to a State of some intelligence, some competence in a variety of fields.
These Techniques have ranged between two extremes.
One extreme, most often used in expert Systems, laboriously attempts to extract knowledge from human experts and then tries to program this information into the machines.
Another extreme is often used in data mining, in which we

6.6.98

~~H.2~~
16 B

 give the machine some very general techniques for pattern discovery, and we set it loose on a warehouse full of data.

→ { What are these extremes of? }
{ Review Lenat's work: first AM as a kind of TNG environment

6.1.98 TM

I seem to have lost that stuff (2 pp) that I wrote in reply to Gunnerman about what I planned to talk about: title: "How to teach a machine" — found them; Professor now 11.1, 11.2, 11.3

Also, remarks about how ML is Ind Inf. were it same thing. ML was bigger. Usually Ind Inf research could be mapped into ML, but Gerry t. other way wasn't always as easy (a "way founder!").

Also brief discn. of Wallace Dowe paper — why I'll review it.

So: two source approaches: Expert systems on one extreme, AI (of some kinds) another. Extreme in amount of BIAS put in by pgmr.

Also extremes in "Brittleness"

"AM" was peculiar: lots of info at start, then very little (some from user response, perhaps)

C4c? Unclear .... I'm really not familiar w. it.

SW "Agents": How

Compare w. human infant: Lots of initial info: designed for an ensemble of universes that it's very likely to find itself in.

My main goal here is to find an optimum path for myself, To get to an usable TM in say, <10 yrs.

The mode of teaching of course, depends on your goal: Much of t. method of teaching to date has emphasized its attempt to have little Bias, so if TM did something clever, it was clearly to machine's cleverness & not t. pgmrs. I don't respect that constraint (I think Lenat does also) — I just want to get t. job done". As for true learning, I think I can tell by how/at beginning t. machine solves problems - as to whether it's learning in a useful way. (i.e. useful for future problems). [Later, when t. machine's behavior is too complex for me to analyze in detail, the problems solved by t. machine will be sufficient evidence of its competence.

contact Jack Sklansky — (7.18.98: I wrote him email several wks ago: but no reply)

Another possl. Goal: to bring machine as near positively to possl., to point at which it can usefully read documents prepared for humans so that it can autonomously, complete its education. Time/money

Idea of "path" in "competence" space

Idea of a Project for the "Scientific Community as a whole".

Idea of "Combining knowledge" of 2 machines that have been trained differently i.e. "know" different things. (This may be possl. w. certain kinds of things, but not w. all kinds)

6.12.98 TM

-88 3P
-22 2P

13

T. problem of Tryg t. machine is fairly diff't: — One approach to diff't probs is to make a nominally more difficult ~~Meta problem~~: "What's t. best way to train such a machine"?

~~What~~ How is ~~The~~ Machine training related to Inductive Inference?

(1) On a trivial level: Given a new problem: how does it bring all of its past experience to bear on t. Soln. of this new problem (2) If t. machine uses Lsrch, its searches for solns. are based on probability — ~~[scribbled out]~~ — or inductive inference

~~For~~ For some time, we have had machines capable of ~~discovering~~ solving any conceivable problem (problem, and that machine) ~~Pattern~~ in ~~a reasonable~~ in (which is for that ~~pattern~~) a reasonable amount of time.

The problem/focus, is no longer, how to make ~~or initially~~ a competent machine — our ~~the~~ problem is how to teach it to become a ~~useful~~ intelligent ~~creature~~ artifact.

ABC

How to teach a machine: For some time we have had ~~good mo~~ effective methods for machine learning. ~~Various kinds of problems~~/ of great variety and some difficulty have been solved by ~~these~~ machines using these ~~tech~~ techniques. These techniques have been used to solve a wide variety of problems ~~some of great dificulty~~.

How far can we go with these methods? A problem that ~~certainly pushes the~~ reaches well past what has been demonstrated.

~~The~~ One real milestone in A.I. research would be a machine that could read and understand human text on a wide variety of subjects. It would then be relatively easy to teach such a machine by giving it ~~more~~ readily available text that has been designed for humans.

~~Much work has~~ While much work has been done on Machine understanding of text, none of has yet ~~produced~~ gotten close enough to t. goal.

How to teach a machine. Gamma 2: 6/14/98

We now have many effective techniques for machine learning. They have been applied to a wide variety of problem areas.

We will discuss efficient ~~methods~~ ways to use these techniques to bring an initially naive machine to a ~~[illegible]~~ state of high intelligence

I don't know how this will fit into ~~your~~ proposed t. division ~~division~~ into "Machine learning" and "Inductive inference". Just about any ~~applicable~~ practical work on Inductive inference is usable for Machine learning.

Perhaps it would be best to decide on ~~the~~ division into sections after ~~the~~ all of the ~~subjects of the talks have been decided on~~. Speakers submit ~~titles~~ or subjects for their talks.

About ~~the~~ Wallace, Dowe paper you sent me to review: I can probably write a review of this relatively quickly. — ~~He~~ I've ~~then~~ discussed many of the ~~issues~~ ideas in one paper with both of the authors, at some length.

Most Sincerely,

Ray.

6.15.98 TM.

.01 There are probly many reasonable paths in language, {universe of discourse} space that would work: (10.25) Thusly
I don't think I can pick an "optimum" one: But I may be able to pick a not-bad one, in which I get max. leverage out of 4-5c; Community. — by ② interacting Team in doing part of work ① by taking old A.I. work & re-expressing Them in as partial [redacted] Tsgs. (10.35)

A recent idea has to be more directed toward language (rng [redacted], using a simpler universe of discourse, than, say, Math. In starting w. Math, I wanted to become familiar w. Eng. TM to do fairly hard probs; than Eng. 'language' could be a sort of Math problem.
→ Alternatively, I'd have TM. Become familiar w. a simpler world than Math (The Math can be very simple!), & then begin to learn langs moving toward English that discuss that simple world.

SN ∈ A motivation for discovering "laws of Algebra" for Humans, is that it enables more rapid calculation. i.e. if we know $3 \times 97 = 291$ [redacted] & we know commutativity of Mult, then we know $97 \times 3 = 291$ w. very little calcn. — Now most TM's can calculate 97×3 more rapidly than they could retrieve a recent calcn. of 3×97 from many, so no value for knowing commutativity of Multiplication... Nor, we can vary costs of various operations so it is cheaper to retrieve a recent calcn of 97×3 than to calculate 3×97.

We might also have TM do var. logarithms so it could do multiplication rapidly using tables & adding — which takes much less time than multiplication.

The Moral is: [redacted] In getting TM to learn via Human texts, it may be necessary for user to (pre-train) insert certain human-like limitations into TM!

6.29.98 TM! L Srch.

Notes for the talk on Lsrch:

Explain just how Lsrch can be optimum!

First: Prove it's "Universal" — i.e. it will solve any INV problem in K* optimum time.

to within a const. (prob'ty)$^{-1}$ of the t. soln. wrt. the reference UTM.

Problem is: if this soln. can be very small & the const is very large

~~Heuristics~~ Let's take another approach to search — how. — Examine Heuristic Srch.

Heuristics Order ~~searches by~~ ↓ search time by suitably ordering trials.

Any reorder of trials can be realized by suitable modulus of the P.D.

So for any particular Heur, I can modify the p.d., so that by using Lsrch wrt that p.d. I'd get same effort as the heur.

So We can solve problems in either of 2 ways:

① Heur search ② Put Heurs into P.D.'s & Do Lsrch.

Is there any advantage of ② ? — Yes! See Sol 96 for reasons.

So: work into the "Thm" that Lsrch is optimum for INV problems

Also in discussing Use of Lsrch for Genetic Algm. problems!

Note that M.C. search w. prob'ty p for object of prob'ty P_i is

expected no. of trials $= \sum \frac{p_i}{P_i} = N$ i.e. for an object of prob'ty P_i, it takes a total of $\frac{1}{P_i}$ trials.

So random search or rote deterministic search are just as bad.

Lsrch is much faster! G.A. searches are usually $\sum \frac{P_i}{P_i}$ searches — very inefficient.

M.C. search w. prob'ty of $\sqrt{P_i}$ is best (i.e. $\sum \sqrt{P_i}$ converges

but Lsrch is even better. (Its in order of P_i, if all cc_i's are the same.

$G = \sum \frac{P_i}{f_i} = \min$; $\sum f_i = 1$ $\frac{\partial G}{\partial f_i} = -\frac{P_i}{f_i^2} + \lambda = 0$ $\frac{P_i}{f_i^2} = \lambda$ $f_i^2 \propto P_i$ $\therefore f_i \propto \sqrt{P_i}$

·31 **PARADOX?** In Sol 78 in discussing Cover's "Extension Complexity")

I ~~used the~~ should Gac's Semimeasure norm on was not as good as "measure" norm on.

T). pc's differed by a factor that ↑ non-recursively slowly to ∞ as n → ∞.

This seems to differ from my more recent results & that Gac's norm is my norm

give about same results (Due for regular rather than "Extension" probability)

(But I didn't actually show this, but I may be able to do so, easily using that technique.)

Well, the norm/measure is unnormalized Semimeasure both ⇒ same limit, but norm/measure approaches it much faster. The rate at which measure & semi-measure approach each other is determined by the rates of convergence of the semi-measure to the correct values.

7.1.98 : TM:

In §197 (Bare) and perhaps in (Melbourne paper ~ 1996 : ISIS) I was berating Wallace & Riss' methods because they only considered a countable no. of hypoths. (for t. discrete part of t. spaces considered).
This is because the set of recursive functs is not effectively enumerable. Usually (if not always) they only consider Primitive Recursive functs! — which are effectively enumerable.

Hvr. when I make approxns to ALP (via RLP), I, too, must confine myself to effectively enumerable functions. In doing so, I make a definite choice as to which ones to consider (& not consider). This gives a real bias in t. results.

On t. other hand, Soyano selects a Ref. Univ (≡ P.D.). The models that are describable w. cc = C are a finite set., and all models are ordered as one ↑ C —
So t. hypoths seem to have a "natural order" if one takes cc into account.
It's only when one includes in t. set of legit. models, those w. cc = ∞, that t. list of models become non effectively enumerable. ACTUALLY $\frac{Pc}{cc}$ (≡ Lcolc⁻¹) is a v.G. ordering.

So it would seem that t. RLP formulation may, indeed, have a signif't advantage over anything else (Well beyond its advantages over systems that only consider prim. rec. functs).

Anyway, t. "natural order" of models induced by t. bounded cc requirement does, I think, give a larger, "better" set of models than using simply prim rec. functs, or other simple · essentially arbt ways to order t. total recursive functs (≡ epmc)

Does this "Natural order" also solve t. problem of "P.obty ratios"? for small cc, t. ratios can depend much on just what approxns one uses: but perhaps as cc → ∞, & one orders hypoths w.r.t. cc, t. ratios do approach some true limit.

7.10.98 TM ~~Gedanken~~ TSQ:

(1) In my earliest thy sqs., I was disturbed that t. only abss. used in a ~~program~~ were t. primitives plus t. (entire solutions) of previously solved problems.

Well, perhaps that's not so bad! If a problem subj. contains ~~abs~~ / sub abss. ("useful") that were not t. solns. of previous problems in t. TSQ., Then this latest problem is probly a very diffl one!

It may well be that finding useful sub abss in a corpus of problem solns. is something that does not occure often (like most heavys are not easily (or frequently) discovered).

Also, t. TSQ's I wrote were simple, essentially "skimmer sequences" - w. no fussy of error. (≈ "perfect" TSQ's). → (3)

(2) Another Major difficulty in this TSQ's was that the techniques did not "scale well" to larger problems: The "Memory ~~area~~" containing useful abss, became very large — so use of anything in t. memory ~~has~~ ~~to become complex~~ would become very expensive for longer TSQ's.

I think t. soln of this (sort) is to use Info Retrieval methods: When one puts something in Memory, it should be indexed so to tell what kinds of situations it is likely to be useful for. This is not an easy problem to solve: it may be that one can do it only after one has a large amt of corpus: So for problems early in t. TSQ, one has no IR, but it's not as necessary. For later probs it is necy, but one has enuf data to implement it.

(3) from (1) In my ~~search~~ search soln. to t. 11 (input/ 3n+2^n input) Mirror problem, again t. only new abss I used after a soln. was t. actual culture soln. of t. latest problem.

7/15/98 TM Gen:

ON Z141: At one pt., I decided that Z141 wasn't nearly so good for discovering finite state Grammers, for predicting sequences, but that it was useful, in a general heuristic way, for discovering (trial n-tups / useful n-tups).

Actually, I wanted to teach Roger as a general analysis of the process of constructing more complex objects from simple objects. This is certainly an impt. problem!

It may be that at the time I developed Z141, I didn't know as much about Coding — using prefix codes, and all. Anyway, the problem that Z141 addressed was: Given the Phenotype, to find good codes (Genotypes) for it. — This is harder than the problem in "GA." — in which we have lots of Genotypes & their assoc. "G" values. These "Genomes" can be trees expressing a Universal function lang.

To a first approx: Z141 assumes that the corpus was obtained by concat rules of the type

A → C·D
C → E·F
C → c
D → d
D → 2F·G

, where c & D are non-terminals (like A). c & d are terminals.

I think there are no loops.

To some extent, we could discover Non-terminals by noting unusually hi freqs of various n-tups. However, it's not clear how best to use info on unusually low freq. of n-tups.

The Grammar of .13–.18 could be used for the genomes of a universal lang. (.05–.10)

At this point: There are several related ideas:

1) The initial idea of Z.141: of discovering n-tups by (recursively) looking at frequencies of pairs of adjacent n-tups.

2) The idea of a Stochastic Grammar of type (.13–.18) I think it's a FSG. It could be the precursor of a universal lang of .05–.10: would this give a universal probability distribution? Perhaps it would: All I need to show is that any there is a common upper bound on the -log prob assigned to a sequence, and it's done by a UTM. — I bet this is true for all sequences.]

ABCDE $\int e^{-x^2} dx$:

3) To the extent that .13–.18 is a FSG, it could be discovered by "Hidden Markov model" techniques! (HMM is universal — they're the non-hidden Markov models of .13–.18 — & HMM have been successfully discovered. — So what about could these techniques be used for .13–.18?

4) I'm interested in G.A. of various kinds. How are they related to Lsrch? — How related to ALP? It would seem that G.A. creates a new population by moving "near by" in complexity space — i.e. it selects new cands that are conditional complexity that is small wrt original population. Mutation does this in a simple way. Crossover in a more complex, apparently more powerful way.

→ 79.20

Wrt. t. "Grand Picture" of t. TSQ problem: At least 2 approaches:

1) T: / Big Sub goal was to get TM able to do "self improvement" Well — as soon as possl. —

So I wanted to take a lot of optzn methods & make a genl. lang to express them.

See 7·24·95 TM General & 7.3.91 p 472 — Both a Mistaken.

Since then, I decided that I would "wire in" some fairly good & OR soln. methods — &

that using them, TM should be able to learn lots of things — including better OR methods.

2) The idea of / many possl. paths in {Language; Situation d urbd by t language} space.

{10.26, 14.01, . . } Here, the main goal, is to get TM to understand

English, as soon as possl., so it can begin reading Human Texts, & eventually,

T. Internet. (We could use a static version of t Internet! say one of Brewster Kahle's

"snapshots" of t. Web ~ 1 or 2 Tera Bytes)

Ability to Read human text would make it much easier to train TM further.

2) A quite difrnt track: Th. Ramanujan T.S.Q.! There are Three 2 Books he read —

which is about his entire formal education in Math. First get t. Man who knew ∞"

(Bio.) to get t. essence, of t book — Major in G Hardy's Math? Analysis? (No reference).

There are other Books on Ramanujan — One can has all his published work — many have R. annotated with.

This goal would be more in t. direction of showing that any approach to TM can give very "creative" results.

Perhaps Easier, or as "study problems" :@ Learning various tricks of Symbolic Integration.

[I'd have to learn what t. state of t Art is in this area]

Easier yet: Going thru an ordinary seq. of Hyschool, Freshman College Algebra books.

Or Euclid: The Theorems in t order of some text books. I want to be able to

get TM to do this as a "Human", w.o. nearly understanding fr complete logical basis of it.

How much "intuitive understanding" of Geometry is needed, is unclear.

or "Automatic" (39,01 R)

ABCDEFG

.30 → A difrnt ~~idea~~ related idea: After TM has knowledge of any field of Software Size,

it can begin to learn to understand questions, statements, etc. a text that field in

a lang. close to English. For its "Mathematical" training, statements & questions would

be in a very simple formal language. Later, TM would learn to translate Between ↔

English & that formal lang. -2520@9PM

However, we from write, in English, about things TM doesn't know about. How he deals

w. this, is unclear.

We could start w. knowledge of Math & English about Math. — Then have it learn

about "Physics" ("Blocks world" or Karl Sims world) — and gradually mix training about

various areas w. Understanding English about those areas. → 21.01

Q! How does this Relate to CYC?

7.20.98 TM

Reading Some old Notes (not so old ... 6/92)

There was t. idea of not having to write TSQ's by (a) T. problem pool start w. only a few Simple problems, then add harder ones as TM learns. (b) Competition between 2, or 3, or 4 TM's for "food" — as They Grow smarter, They become more of a challenge to each other. 3 or 4 or more TM's involves cooperation, sociology, etc.

(c) Have TM work problems that are "easy" because there is no noise. We then add noise to make harder problems. This is an easy way to construct T.S.Q.

More Recently, instead of TSQ's, I had t. idea of Evolving environments. — so any kinds of devices can be used by t. teacher. Hints, telling, giving heuristics (or partial Heuristics) ...

Karl Simms has simulated creatures that fight, w. ≈ r.w. physics, gravity, friction etc.

— A survival Contest. Could we also reward them w. "Food" (= Energy = reproduction) if they solve certain problems we give them — like "a cosmic God"

(SN) In use of Lsch to solve n+2^n input Mixer: If there is any noise in t. system $\frac{S}{N}$ ↓ rapidly as n ↑. One way to ↑ $\frac{S}{N}$ is to ↑ ssz, but this increases CC of Soln, rapidly. To what extent is normal G.A. soln. of this problem (Say Koza's soln) degraded by This Effect?

Another way to ↑ S/N is to do search for 2 or 3 kinds of addresses, rather than just 1. This involves more search, but t. Q is — do we in t. long run, solve t. problem faster?

For t. Paper, & for my own Use: Try to list "all" of t. ways to teach machines: The categorization so one can see just "what's going on" & what t. Alternatives are"

Maybe put t. various Methods on "file cards" or equivt.

Each "card" or equiv. will have a brief descn of t. method plus (perhaps) a list of references.

Some newish ways to teach machines

1) S.W. Agents: possibly not.
2) A life: Several different techniques
3) Simulations of various kinds of "Complex Systems".

Use of Loglan or Esperanto — Since they should be easier for Machines to lrn.

7·30·98 TM TSQ:

Related to (19.30 – .40)

Yet another Approach to TSQ! (Actually Now!)

We start out w. a TM that's smart enuf to interact usefully w. a human.

So we start out W. a min. amt. of "Common Sense".

At first, the human communicates w. TM. in a very simple lang. . . .

Then as they interact over time, the lang. gets more like English & becomes Richer.

The goals of the machines are: (1) To learn as much about the Human in his world as possl., in available time.

(2) Not to waste Human's Time (3) Not to "wear out" the human — but to keep him interested.

When the TM gets smart enuf we may have >1 "Teacher" — sequentially or in //.

Also, he will have learned to Read English Usefully.

So I guess this Human teacher could be a path from Initial "Trng by neuro surgery" to Trng. by reading English.

We want TM to be smart enuf so that its Q's asked of the teacher are of much info content. So TM thinks a lot before Q's. Q's do Expensive.

We may arrange so that TM can make patterns in sound & Video that may amuse the Teacher.

Perhaps the goal of the machine at first is merely to amuse the "Teacher". It could be like Karl Sims' Video & Pictures & Music.

Maybe stuff like Dave Burke's: In a public Area, it learns to amuse people it interacts with. Maybe charge 25¢ for 5 minutes of interaction! User wears stereo phones, Maybe stereo Goggles!

The maybe Emphasis on mainly Ascii interaction: So it will learn to read English. One could have other modalities of interaction parallel w. Ascii. So the Ascii bounds the occurances in the other Modalitys.

Perhaps TM has access to Data set and it must learn to Answer Q's about it. It could occasionally be given "correct" I/O pairs: Or when asked a Q — it could ask for Answer. — Then

7.30.98 TM TSQ. short run

would ↓ its Gone, but maybe ↑ in long run.

hs hs

For the teacher, Runner is a pre-processor for his Q's & A's: It tells the human if its ss were O.K. — that they used "known" words & "known" syntax. — i.e. known thus far. So teacher can edit what he tells TM. When a Q w. One unknown is given, TM may usually ask for answer. — Or maybe give it "free" answer — Teacher can judge when. (The TM can learn to do this "editing" feedback to user (= teacher).

Is there a way for Teacher to give "hints" (ways to ↓ the set of search)? Teacher can occasionally ↑ cc of TM's Q (if it's a Good Q as judged by Teacher.)

.20 Teacher doesn't have to be right every time. — very Impt!

Re: .20: I had this idea that MTM was quite different from NMTM.... Realizing that they are about the same, makes noise much less impt.!

— I was thinking about learning pgm's w. a teacher.... in which it would be terrible if the teacher made a mistake...! — No more worry about this: the Models normally aren't disturbed by noise!

7.31.98: HOW "S.W. Agents" fit into this:

1) One Advantage of "passive" training rather than Active training (w. teacher). — Passively taught TM's can be easily combined by combining their pgms! It's harder (or impossible) to do this w. Actively trained TM's.

2) Watch SW Agents: The user can periodically give them a score for their performance a/o ↑ difficulty of probs Given. Also the user can vary rewards, punishments for time taken, goodness of reply, etc.

3) Agent watches human and periodically says "I can do that": TM is then tested & may be given more response abilities. 4 5

52 50

.30 8/98: Try this: There is a small (to start off) corpus: perhaps in English (perhaps in Esperanto, or logician or a simplified English or Computer lang.). We start off by writing Q's in a simple lang. (≈ Machine lang.), & TM learns to answer them, using data in the corpus. At first, the Q's are word oriented. ("find ss containing certain words or ~~syllables~~ syllables... or...). Then as time goes on, we ask more & more complex Q's about the corpus. Then we relax implicit constraints on the corpus — by making it closer to English, by making it

Syntax have occasional errors

We can give TM examples of correct Q A's;

We can allow TM to ask Q's of trainer, but such Q's have ~~the~~ cc.

SN One Criticism of my TSQ's in Saarb., was that their use of Memory was Bad. That as time went on, Cost of retrieval would ↑ so it would become less as less Useful. (↑ model did not scale to larger problems).

.10 T (proposed) Soln. was IR techniques: When one put info into memy, one had to index it so that it would be easily retrieved in appropriate circumstances: This "ease of Retrieval" also meant it would have much lower PC in appropriate Situations than just "$\frac{1}{N}$" (N being no. of items in Memy).

However, proposal (.10) is also ↑ use of a/ other numerous heterogeneous techniques for IR, is a rather big project... forbidding!

(Tho, ... perhaps there is a standard "Gen!" soln.)

→ Anyway: Should/can search automatically devise IR techniques — except is "Universal" ≈ close to "Optimum" ? If so, how does it Automatically do this "IR" ? (both input & output for "IR"). → How this can be done (Conway): 33.11 ff

Well, (1) first, ↑ Machines I was using, were not Unix. (2) Even if they were, I'd have to either "Lead" them to IR techniques by a suitable type Envt. a/o use an Instruction set (≡ Appropd) that made IR ideas easy to discover (by PC / low cc).

→ Big Q: Say I got ↑ system of 22-30 ff running well: That it could do IR for ~~Math~~ English Requests. Even that it could learn enuf English syntax to analyse Eng SS to see if they helped answer to target Q.....

Would such a T.M. be able to read English books in, say Algebra, & learn Algebra — well enuf to be able to solve hard probs in Algebra — ~~Even~~ Do "creative" work in Algebra?

→ 26.01

8196 TM PSG Diary. Z141.026

In the Paris Paper (~1959) I had a theorem that seemed necessary for the technique to work, perhaps "Not Nearly": I can have certain categories of sets of strings: Σ (the acceptable set), the known set of terminals. N a finite set of (Non terminal)s: These are sets of strings (of which Σ & T are included), We try various rules of form $N_1 = N_2 \cdot N_3$ (or $N_1 = N_2 N_3 N_4$) to see if they work! Initially $N_1 = \Sigma$, , $(N_2)(N_3) = T$:

— A way to find rules that work well (not nearly 100%) — we have way of scoring a value of a rule that's not 100% correct.

SN There was this problem of how to compute pc of error correction in the induction of sets of finite strings. It may be that my recent Theorem real soln. (Pres report letter to Wallace plus the "proof of convergence" based on theory Conjecture) may help w. this. — I think I did find a soln. — sometime after Saarbrucken (or may be at Saarbrucken).

Actually, using ALP, we realize that we will not nearly get the "right grammar", but in general, the symbol a substring (probably) produced will be good.

-26 SN Impt: In Z141, I was concerned w. the fact of unusually low frequency 2-tups → I felt that the Grammar Discovery technique had to be able to use this info. NOT SO: the frequency of N-tups was merely a heuristic device! A measurement to help find good grammars, — but not nearly infallible. — The reason I knew it was infallible was that I constructed the wizard grammar, that was not a Bernoulli Grammar.

-31 As a general Methodology, the idea I had was good, i.e. Look at the present stock Grammar (or best found thus far). Do various N-tups occur w. expected frequencies? If not, modify the grammar so

-34 that the Expected frequs are closer to observed frequs.

The trouble is, the grammar I devised to express unusually low freq. 2-tups was somehow bad, or at least it was not used properly, — I think the way it would: first get frequs of symbols: look at frequs of pairs & make corresponding rules. But the resulting grammar was

Somehow constrained. If α = ab was defined, ab had to be coded as α so the freq. of b following a in the resultant code had to be zero. It was no longer a Bcan Grammar, so I couldn't use the same heuristic to define new ntups.

Hvr., the hour of 24.31–.34 is very Good! It gives a sequence of improvements in the Grammar. Unfortly, I suspect that it often → local extrema, so one must normally try several possl. "mutations" of the grammar at each "branch" in the path. — This ↑ cost of the grammar discovery, tremendously.

8.9.98: PSG discy!

Hot Bot. com
Larry Randell

.17 Using techniques of, say 24.31–.34 find ntups whose definitions compress the corpus. This amounts to a CFG w.o. loops.

If we had, for a corpus, a CFG with loops, & we did this ntup defn. to compress it — we would discover lots of ntups of interest and, if the corpus was large enuf, we'd discover a very large no. of grammar rules of the type A → C·D.

We then look at this grammar alone & see if we can compress it by introducing loops. It should be possl.

One (perhaps ~~a~~ a sufficient) way to suspect a loop:

.25 We have many Rules of the form $A_i \rightarrow X B_i$.

.26 We then suspect that $[B_i]$ are a ngmset. We look for other rules of form $C_i \rightarrow Y \cdot D_i$ & suspect $\{D_i\}$ is an ntpst. If $\{B_i\}$ & $[D_i]$ have many common elements, we suspect they are the same.

Or, just look at the elements of $\{B_i\}$: ~~Can we~~ make the defn of that set has a grammar (that we already have discovered): it's the sum of the rules that are able to generate its elements.

.29

.26–.29 may not be so interesting! More interesting would be if we find a ntps on the rt. side of rules & a ntpst on the left side & suspect they are the same. If so, we have a loop!

I.e. If we have many rules of form $A_i \rightarrow X B_i$

we may suspect $[A_i]$ &/o $[B_i]$ is an Ntpst. Say we find many ntpsts by this technique: Then we ~~try with loops that agree~~ compare ngmsets that occur on rt. of rules w Ngmsts that occur on left of rules. We make a M × N matrix that compares these Ntpsts in pairs (how many common (or % of common) elements).

8298 TMTSQ (GA & "Introns") (.05) | Sol 78 T3 (.25) | 24.35 attached to 31: on PSGdisky 26

planet around

.01:23.40 : ~ "Expo" (in Bay Area): Suppose we find a manuscript on/ α content.
In t. analysis of t. Doct., we would postulate a culture created it, & try to build up a picture of that culture from info in t. Doct.
Parallel w. this, we do "pure" analysis of t. Doct. itself.

.05 (SN) GA: looking at/ conventional (say Koza's) GA "solns" to problems: t. Density of "Introns" (non functional code) in them could give an idea as to how much more efficient (cc-wise) Lsrch is ← since it (usually) has no "introns". ~ 27.18

.08

→ So, t. former is "Domain knowledge", t. latter — maybe pure syntactic lrng.

(SN) On t. varieties of "Supervision" in lrng:
1) Asking Q's of "teacher" 2) Teacher looks at performance of T.M. up to now & [redacted] (selects / devises) a new Corpus on that basis. Teacher can also give sample subcorpi to TM to "test" to see if certain concs. have been acquired.

.25 (SN) on Sol 78 T3: Proof for Radix = ξ: To show Lemma is true: Its a "≥" inequality, over a certain region: find the lines on boundarys of t. region where t. inequality is "=". Then show t. first & second derivs are of proper sign in other regions of t. boundaries (& all other parts within t. boundaries).

→ On corpus sequence design: Try this: we have this small corpus. For certain concepts' discovery, this corpus is too small. Also, t. pc of these concepts is small wrt. t. "normal" initial inst. set (≡ apriori).
We introduce "Hints": these can be examples related to RW (≈ Domain knowledge) or any other devices to ↓ cc (or just (cost) of soln.
This "Hint" could put an ad-hoc bias in TM's p.d. — so

This starts "Gno. Domain knowledge" —.

L. Rendell, Ello
IJCAI 87
Guiding constructive induction for inc. lrng. from examples.
4 pp

email:
lrendell@ncsa.uiuc.edu
or
rendell@m.cs.uiuc.edu
ph. 217-244-0092
Is on NCSA computational Biology faculty.
U of Illinois at Urbana-Champaign.
Nat. Cent. for Supercomputing Applictns!

1996 (IT)
42 48-54
conc. lrng., using complexity Regularizn.

To avoid that, we have it recompute ~~[illegible]~~ its pd using ~~[illegible]~~ an augmented corpus.

I was worried aby "CYC" getting wrong biases by having ~~a~~ "knowledge" being "programmed in" — but Here we give "hints": T. info is tied into t. rest of the P.d. using t. augmented corpus. "Hints" should be weak as possl. — we want TM to put info into its P.D. in its own way.

So: How to start out (?): Maybe a "Blocks World": But with perhaps simpler language (to start) a "few Blocks". The path is then to put in more Blocks properties — with or w.o. TM's ability to "Experiment" in a simulated "BW" — richer lng w.r.t. "Blocks"! Then make lng. more like English: Make SS have random errors: Then introduce ~~more complex th.~~ new things into t. world.

Maybe Sim's world, ~~[illegible]~~ interacting Robots, but have language to talk to them — with rewards for correct or useful responses: T. ✓ to "teach" & talk to them in "world". — T. world needn't be nearly as complex as Simms' world is, hvr.

.18 : 26.68 (SN) Re: ≃ GA: perhaps try constructing "approxns" to simple functions like $(3x+y)/(2x-y)$ viz Lsrch: Compare w. GA (cc).

Also One way to ~~sl~~ solve things like this: Use digital srch (stoch lang) to generate $(ax+by)/(cx+dy)$: Then use ~~[illegible]~~ special N.L. H.C. method to find ~~[illegible]~~ optimum a b c d for the data set.

For Lsrch, we have to get reasonable good fit as soon as possl. — or else t. time limit ends (or we go very slowly in T.Share Lsrch). — So maybe do course lattice search of param. space, to start off.

S398 TM TSQ

.01 Say Grammar for entire large corpus were desired: To what extent would this have much info about the "Domain" of the corpus? — Well, by "Grammar" I mean as many logical regularities are found as poss'l. Seems likely that, were this done for a larger corpus, that TM would really "understand" the domain. — i.e. be able to answer Q's about it.

So: Perhaps: A large ~~text~~ text corpus: TM finds some regularities using "standard techniques"; Since corpus may be "very large", data mining ideas, methods may be relevant. The "TSQ" consists of what might be regarded as "Hints".

Narrower corpus, etc. ⇒ Q's.

Narrower

Kinds of "Hints"

(1) Special Augmentations of the corpus. (perhaps All that should be put into this format): (subcorpuses Q/A pairs: Plain Q's).

2) Some Analog I/O. — perhaps dealing same thing as some text I/O.

.19 3) Hard a/o Soft Partitions of the corpus: This could tell TM what the some impt ~~divisions~~ a priori divisions of the corpus are.

Many different types of partitionings of the corpus are poss'l. E.g. English, French, Grammar? — topics? "Topics of Math, Biology, Sociology", Hy freq. v.s. low freq. words.: child Early v.s. adult later learning. — perhaps an Ordering of the corpus like TSQ order.

Just How is TM to use these "Partitions"? → (43.36)

4) So (1) Augmentations (2) Partitions (orderings) of corpus are 2 impt. Categorys of "Hints".

5) Normally TM has a single Goal — to compress the corpus! "Hints" can take the form of other goals. I.e. various CS probs., INV probs. — using the P.D. obtained from the corpus.

In .01 ff., we have this "text world" (~ to "Blocks world"), & the Goal of TM is to compress (≡ "Understand") this world. The "hints" are oriented TOward that Goal.

Now, In fact, understanding "TW" is not really the top goal. The top goal is really that TM should be able to work any probs I give him very fast & very effectively. Hvr. / in understanding TW, TM gets the P.D. — that can be used for solving any problems.

Well not So clear that the Pd for R.W. & the Pd for problem Solving are the "Same": Actually, they need not be. Both are Conditional Prdby distribs, so we can have one Cond P.d. — One kind of condition involved compression, (Predn) in TW. Another kind of condition involves → 29.01

97.3

101
61
pdx

161.8 lbs
186 cm.
(5'.15 ½")

9 standard deviations EKG.

8398 TM — TSQ "TW" TW
Text World 29

.01:28.90] VIV probs — another involves OZ probs.

Try to discuss this approach w. Marvin, Selfridge, Jergin ?

"Textworld"
-92 Goals of TW project (NN 84.98 #(5)): ① Top Goal: Ability to solve arby, very hard probs, usually OZ, maybe some INV. ② Intermediate: To be able to understand English well enough to get some info by reading Human texts, — So Eng texts (usually for Humans) can be useful tsq.. ③ To learn to use other Modes of knowl acquisition; (Asking people a/o databases, Experimenting in Math, computers, in RW (Analog experiments in Electronics — in putting ckts together, on devising LSI ckts. — moving electrodes "by hand" in a vacuum envt. TM could easily devise new solid state "laws". → it could do in RW, what Koza's pgm did in "Spice"

The more it knows, the easier it is to Teach.

T. main Goal is to Get TM to pt. where its easy for use to devise tsq envts. for it.

We can start w. simple Mathematical lang. Have TM learn its ways; Then we slowly ↑ complexity of lang. — Move it toward English. — So we can use ≈ (Loglan) English Esperanto) to desc. Math'l facts. — Try to move toward English as its used in Math text books.

An impt idea that underlies writing TSQs easy is t. CJS idea — it tells which ideas are "close" to what other ideas. Perhaps one of t. very impt values of Lsrch is not so much its "Optimality", but t. fact that we can use CJS to get upper limit on its cc.

One of t. Attractive features of TW is that is seems possl. to express many "Tsq Envts" in this form. Thus "Compress t. corpus" can be an ongoing "Backgnd task" for any other problems. [I'm not so sure! "Improving t. P.D." is t. usual backgnd. — ∴ it is oriented toward solving OZ & Inv. problems.]

[One interesting common Tsq types: learning to answer Q's after being given many example Q, A's. — ideally, I want this to be treatable to a data base — So t. Q's are about info in t. data base — so t. [Q,A] set does not contain all of t. info.

AINT could be regarded as a [Q,A] extrapln. problem (c para for induction).

Theorem proving is an INV problem, but putting t. normal heuristics into a P.D. form is not trivial!

A Good kind of Prelimy problem for TW: Ask TM questions that it would know t. answer to if it understood t. Text. e.g. "Object X is Black" occurs in t. text. We ask: What color is object X" Other Q's could require more or less computation on t. part of TM.

T. main things I need to do for this talk!

.37 1) Read More on Lanst, Incr. lrng, G.A., so I can contrast what I propose w. what's been done. That 1992 paper of Lanst is v.G. — but try to see more recent stuff; try t. net for Lanst & Incr. lrng.

I do have lots of '89 '90 refs for incr. lrng: So look up t. authors or find refs to these papers in Citation index (Harvard Sci (lb) a/o MIT)

Blu But
Vio Very
Gray Good
White Wine

Y 4
G 5
B 6
Violet 7
Gray 8
White 9

B
BR
R
Or
Yel
Grapes

02896 TM 02 probs

-01 : To what extent is 117.01-.40 an adequate soln. of the 02 problem?
Look at 105.01 — .23 .... Actually total previous work is: 105.01-.25; 108.89-.40

Chris. Moore (?) wrote this paper showing how an n.l. analog dynamic system could simulate
.02 an arby TM. Unfortly, t. effective memory size of t. device was sharply limited by essential "analog noise".

Some new ideas on this (1) If there was any regularity in t. noise, then t. noise wasn't really as big as it looked, & system would have an effectively larger memory than I thot originally (still << 100 bits.).

.10 (2) Consider other n.l. analog systems w. feedback (recurrent Neural Nets). Probably they, too, could often simulate a TM. — again w. small amt of memory! — But if we have many such machines, each w. a small amt. of memy they could do a lot of computation! → (6)

(3) One idea I had was related to (2) — I was going to use atomic nuclei as computers, w. small amts of memy, then link enormous no. of them together to make a larger, fast machine.

(4) Hvr, say one had a bunch of "Noisy Computers": ~~each of them~~ could we usefully use them — say, for induction

---

(5) Another Tack: we can make very hi speed analog devices: Tunnel diodes, etc. It may be possl. to make very fast "Analog" computers of them. — Say like .10. They need not be 100 accurate. If they are ≈ universal, perhaps do Lsrch on them!

(6) Re (2): Recurrent ANN's can probably be UMC's! As such they have a "Halting problem". One way to deal w. this is to reduce feedback: which is like reducing memory size or "retention". Or just do Lsrch & cutoff on a trial when Time exceeds $2^k \cdot$ a.p.c. of trial.

{ Assoc. w. t. idea of Moore's machine having small Memy, because of noise induced by envt. (.02) A "similar" problem occurs in t. "decoherence" of Quantum computers when they interact w. t. envt. — Are these 2 ideas related? Are they t. "same"?

8998 TM "TW" "TW"? — Text World

Where is P34? = discovery → is a P50 discovery

I may want to sort of write up. talk now: Then work on details:

Also, perhaps write Jerusalem!

Make list of kinds of training I can patent re "TW" framework; Also Read that paper on learning from text books (in 1990 AI & Cont. (87-93) D. Cohen)

Make a list of the supposed "kinds of learning" that are in A.I. & ML. conferences,

.09 The Inept features of my approach!

1) A common (format, language, formalism) for many (perhaps all) types to log, environments. So learning in one modality or domain, can be used in all others. — Also facilitates incremental learning.

2) Subclasses of T "A very General method of induction that can fit all other forms of learning into a common formal, easy, formalism"

3) Possy of easy parallelism for Learn! Also possi. to divide up tasks better many sites — on the internet. Or, simply have many distant groups working on distant problems. If we use passive tsg's, then we can integrate like-TSg's of various groups, either by "advertising" (?) or by ≈ parallel, time-share. separately doing them. TSg's sequentially on the same TM.

We would want mainly v.g. TSg's: so most of time we should be spent (in a week, say) .20 by a group devising the TSg. — Then << a month spent by TM, learning it.

See how Tom Ray does his A.Life on the net. → 32.01

I had the idea that (seeking for common subtrees in the trees that represented past successful calls/functions — was a good idea. — But that it would not often be useful in "small problems". — A small problem being one in which its adequate to simply combine old useful abss. (& all "useful abss." have appeared as sols. to past probs). In "hard problems" / new intermediate abss occur (sometimes w. "1 shot learning") that are useful enough to warrant defining them even before the problem is completed. This is diff.: we have to get partial reenforcement both the problem is completed (as in chess, where winning a piece reenforces locally, & winning the game reenforces globally) & for a long time!

On the other hand, discovery of common subtrees could occur as Eh. Discovery of a heuristic — while analysing the traces of sols of many problems.

.32 I had the idea that if we recorded functions used, only in their "parsed forms" (i.e. forms in which they were executed), it would not be hard to find new parsings for the useful functions — So that the new parsings would be a more "compact" (compressed) "basis" for the useful functions.

So if .32 were true, we would only modify the conditional probys for function selection in "tree growing". To start out, After (+) is selected, we need 2 args & they are selected w. probys indep of the fact that they feed into (+). Later on we will have enough data so that for many functions, the different arguments will each have its own cond. probys for input functions.

These cond. probys are sort of equivalent to defining certain subtrees as being important (≡ hyp).

→ 32.01

.01:25.40 We then try loop rules that [illegible] assume various Rt, left NTpsts are identical.

02 The simple grammar of 24.31–.34, 25.17 may not be adequate; we may need more complex rules like $A_i \to X B_i$, $\to Y D_i$ } so each "Rt side" can have 1 or more "left sides".

04

So, consider some simple stoch. CFG's (with & w.o. loops): look at the frequency with which various ngms occur & see if my techniques would discover them.

(.02–.04) may not be really true: The single $A_i \to X B_i$ rules are the type used in Z(41)... It may be that rules of this type alone are adequate to form a grammar of any stack CFG (looped or not). W.o. loops, hvr., we need a larger corpus to justify the grammer — since it's a Bigger Grammar

↓ Quite Reasonable Approach.

→ 9.4.98! 1) Read stuff on J.J. Horning in Vol III of A.I. encyc. Also, get his paper(s).
He has a way of cutting down on search space for grammars w.o. excluding anything desirable.

2) Using ALP, I can do Hillclimbing in Grammar Space: starting either with ad hoc or promiscuous Grammar. The problem is to find good "Mutation" operators. Hvr. —
1. Mutation idea is too "Greedy"; try less Greedy: i.e. (2 or 3 mutation) trials. (Different amts of "Look Ahead")
Also, look at mutation operators (or Macro operators) that have been successful; try to extrapolate them in trivial ways a/c by "passing a grammar thru them".

8.9.98 TM "TW" **Expo: Talk.** 31 ison ps ... 32

← Saw 29.02 on Goals of TW project.

.01: 30.20 T. talk itself! Start out w. a general dern of what its supposd to do, & some reasons why its a good approach. [T. more it knows, t. easier it is to teach] ⌈ Lsrch for inv, oz probs. ⌉

Then give a brief history of my development of t. idea: talk about Lsrch "optimality" (explain just what it means), Then TSQ's I've written: what seemd wrong w. them — that They seemd to hard to write.

That big subgoal in long theory is Ability of Machine to lrn from Human text —

But I'd also like other modes of "teaching" t. machine to be possl. — A That t. approach I'm proposing seems to facilitate this.

(SN) Learning Systems that claim to be Universal:

1) **RLP** sol
2) MML wallc. (~~strong MML~~ → strict MML)
3) MDL Riss (predictive MDL, Stoch. complexity)
4) Chervonenkis/Vapnik {Support vector Method?
5) Soar (Rosenbloom/Newell) ["chunking"] — other refs.
6) **EBL** (≈/ universality Conjectured by ... DeJong?)

Discn of Rosenbloom's Macro Operators! (Machine lrng II p 280) 1983 actually Rosenbloom/Newell.

 Talk

.20 1) ~ Top Goal of Project: "Intelligent Assistant."

Derb. "Int. asst" "Expert systems" approach}: Why Bad: How Lrng fixes it.
Brittle, Diff to Get into from Experts!
Impossl to get from expert.
(≡ Unconscious Mind")

① It can go beyond "what it was explicitly taught"
① brittleness
② ... data input
③ Able to Model things Experts can't talk about. "Unconscious Mind".

2) My early work on this: **TSQs: First 3 TSQ's.** Dz probs, INV probs.
The Power of Lsrch: T. 2 Gambling house Thms.

3) Latest Approach: "Text world": Why its Good! Mention D. Cohen's experience.

a) Oriented toward "reading & Understanding **Text**." — (Cover and king et al. Pred. of text by humans (≡ compression).
(compression deals) implies predn. & "understanding" of some kind.
b) Also assoc. w. understanding & learning to solve problems.
c) Very many problems & training methods can be put into this format:
(Give several examples (some very general: others more narrow").

Some Subjects/topics More Detaild discussion!

ⓐ Maybe! What t. "optimality" of Lsrch for INV problems means.

ⓑ Why we need **ALP** (rather than MML or MDL) to solve predict probs.:
That **RLP** gives us models we wouldn't have thot of otherwise:
" Gives us a better shot at "uncons. mind" — By considering all acceptable models: If Ω computing ... we may have to modify our srch technique to to her advantage of it.
Give more detaild examples.
Some dissn. of Difference betw. ALP & MML, MDL.

By not ... considering t. Halting problem They are omitting many functional forms: In some cases: They throw out Baby w. Bathwater.

→ 39.01

81098 TM : LIT : Literature.

These are ideas obtained by reading papers of others:

Attempts to understand Explain Based Generalization. **EBL**

For DeJong: ML book #2 : p 571 ff!

My idea Pres for: A "scheme" is a canned procedural soln to a ~~for~~ class of problems.

A (perhaps unique) idea of this paper is: Say we see a single instance of the use of a particular "scheme" to solve a particular problem. — we have a "trace" of the soln.

How do we use this one instance + "background knowledge" to generalize this scheme into a more general rule that can be used in a greater variety of problems?

For me this may be close to (or a special case of) the IR problem. Something of interest has occurred. How do we best index this occurrence for easy relevant future retrieval?

This is much less of a commitment than DeJong makes: DeJong would actually use the one-shot "learned" rule. I would use the learning to retrieve the previous instance and combine it w. present info. to decide what to do. Essentially, DeJong discards the past instance after he makes his induced "scheme". I think there is this strong tendency among AI researchers (& perhaps most professionals as well!) to want to make a decision & then summarize data — then throw away the data.

Another "I.R." system is CBR (Case Based Reasoning): We have a present problem. We look at the past to find "similar" problems. We find the one that is "closest" to the present problem & modify it to it solves the present problem: This can be rather simple modifications, or by a "Analogical Kalman".

In EBL when we solve a problem, we Generalize it to solve anticipated future probs similar to that one. The Learn occurs at that time. Probably some "fitting" occurs when the Generalization has to be used, (?). If so, there is a continuum between CBR & EBL.

8.13.98 Perhaps actually start writing the talk. After it's quite clear as to what I will say, send Grumman/Lacy Description of Abstract. Perhaps make viewgraphs for talk & put them on Net.

SN on "Operationalization" ≈ "Explication" The idea is to put somewhat ambiguous, vague, English-Human statements into exact form so they are directly usable by a literal-minded Machine. In Cohen's paper (Vol II AAAI 1990 p 713.) he feels the Machine to do much of this. "Fuzzy" logic is another way to "operationalize" much of English text.

"Operm" is a way of implementing the human heuristic: "State the problem exactly" (which I usually assume has already been done for the machine — so the machine's job is to "state the problem vaguely" — i.e. find out what it's

S 1498 TM: Literature.

"close to", so it can know ~~whether~~ where to look for "similar" "similar" solved probls".

.02 Some rects of int: D.H. Fisher: Knowl acquisn via incremental conceptual clustering. "Mach Lrng" 2(2): 139–172, Sept 87
J. Gennari, Pat Langley, Doug Fisher: Models of incremental concept formation. "Art. ~~Intell~~ Intell", 40 (1-3): 11-61, 1990

Some comments on Cohen: ~~Hist~~ (33.35)
"Theories" are rules for bidding: "An overgeneral theory" is one that often works, but often doesn't! This is regarded as being due to it's not being narrow enough in terms of its use: e.g. ~~if~~ condition A is true, then Do B.
We have to narrow down ("specialize" = $\left(\frac{1}{\text{generalize}}\right)$) A (or specialize) the theory).

What "explanations" are: an explanation of a (correct) example is a demo that 1 or more of the rules would result in that example. I.e. given a hand of cards & a (correct) bid: show which rule(s) would cause that bid.

Multiple incons. explanations: I guess this means explanations that give bids to different ~~rules~~ examples.

My impression is that many of the "problems" in lrng theory are easy to solve, but the terminology used is a real Babel that makes it difficult to perceive similarities betw. problems that would make them much easier to solve.
Usually what happens: as soon as I begin to understand what the problem is, I ~~begin~~ to see a clear solution to it.
However, the Cohen art (~~33.35~~) has several impl. probs in "lrng from tasks" that I should understand.
2 kinds of diffys: (1) Underlined terms: They could be formally defined, but in terms so useless as to be operational.
(2) Some of the rules given were "over general"; they were fixed in most cases (for most users) but not all users.

In (1) we may have to use knowl from RA to get reasonable choices of defns: which are then tried on the examples + "rules" to see if they work: At least "RA" dependent heuristics would try minimally informative "modifns" of the defns that make them operational — then test them on the examples. 100% score on examples is not necessary. Some examples could be in error.
In (2) we want to minimally modify the rules to get a set that (a) have few (many) errors in the examples (b) cover as many of the examples as poss. (hopefully, "all", but not exactly "all")

81598 TM: Literature! [Incremental Complexity] [Conditional probby] (10)

I think by "Theory" Cohen means ∃ complete set of rules to decide,

81698 Well: EBL! A possl way to look at it from APL perspective:

An "explanation" of a data-set is a short code for it: Say a 2 part code, w. ∈ rule or law or set of rules/laws a ∈ code for the data in terms of those laws. (for a situation they describe, the code is of pc=1, pc=0: because it fits exactly. Usually it's like a Q.A. corpus, w. Q being "free".

Say we have some new data. An EBL approach: we want to code it using a rule & the code of the data in terms of the rule: (Consider Q, A, ... type of corpus):

This old EBL wants to make a new rule that is "close" to the old one — (small [Incremental Complexity]).

.10 SN "Incremental Complexity": defn: A → B is of small inc. comp. if we can obtain a code for B from a (short) code for A, using a "hy pc xfm": — "∃ set of hy pc xfms" is obtained by considering the corpus up to now.

.14 More exactly, the distribution of pc's of these xfms is —

Basicly, we want a set of pc's on these xfms such that some/all the pc of the entire corpus is maximized

these pc's (wts) could be assigned solely on basis of pc of entire corpus —

or they could be assigned so as to maxz. the pc of the corpus achievable w. available cc. (RLP).

.19 We could view the wts of the xfms as like a "TM2 problem" — which is part of the problem of getting "new good ideas" from "old good ideas" → 36.21

Perhaps a less EL. View: That we want to minz. total cost of coding:

.22 We build up a "vocabulary" of predn. operators & xfms of predn. operators & sub operators & primitive operations. (However, we also have to consider pc of corpus wrt the operator (S)).

This is an inept idea! It bears on the Q of how good "incremental" lrng is — & the best way to do it. One area of interest is Generalized Z14 models: In which we make models w. increasing goodness, based on (previous models, parts of previous models, & standard combination rules).

{ It seems to be very inept! Its certainly incomplete --- needs a lot more work --- but do get back to it! }

First try to decide, exactly, just what problems are being solved.

.32 Other problem: We have coded corpus C_0 w. (short) pgm P_0 (thus far).

We get incrementally to $C_0 \to C_1$ How best to search for code for C_1?

If P_0 is a sequential code (& it must be for ALP, sequential/predn), we just do

.36 random increments of P_0. If, as is usually true, P_0 is a 2 part code, we just do random continns of the second part (& first part is a self del. code

.38 & cannot be incremented in a simple way). However, at a certain pt., we decide that the increment to symbol of the incremented part of P_1 is >> that of C_0,

.40 So we begin looking for modifns of the first (& pert) part: which is a "finite object".

.01: So 35.40 begins to look more conventional — i.e. the measure of similarity of 2 finite objects. This is like 35.10–.14 & 35.22: Consider A → B; we look at the code of A: if it has "parts" in it (reused functions) we devise a stack lang. (could be a very simple lang.) that can produce A w. hyps; then any other hyp objects in the lang are "close to A": .28

.05 Remember, Hvr, the Top Goal is to get a good code for B. 35.36–38 is the usual "Default" code.

We might look at the new sections of 35.36–38 & try to find out just how they didn't fit the functional forms that worked in the past. This sort of thing occurs in Physics, .09 when the corpus augmentation consists of data in a "new part of the Universe" — say for extreme parameters. At this point, we may want to try modifying, not only the shortest code for Co, but other shortish codes, that are clearly much worse for the universe of Co.

Hvr, this "Physics" idea of .09 ff. does involve expansion into a "new part of the Universe". Just how to proceed is not clear. It seems reasonable to try to go back to more "fundamental concepts" (i.e. the generators of the best models for Co —— and generators for earlier parts of Co .... before we devised the latest Models for the "complete" Co.

.20

.21 Perhaps the TM2 approach of 35.15 is more correct! That we look at the "trace" of how the various models for Co evolved ... including models for just the earlier parts of Co (models used in the historical past). — Corresponds to Schmidhuber's "knocking things off the stack" in his "Learning how to Learn" pgm. (~ Dec93, Jan94)

.24 So this is a kind of Vindication of part of the system. (an impt. part, conceptually — tho I wouldn't actually discard the top sections of the stack! — I would keep them for "information" (Entropy?) std. down until ~ 8 45 P.

.28: (.05) → Also, the problem of 35.10 ff is related to what Genetic Algs is all about —— Whether he was pure mutations a/c crossover & how it fit the "Building Blocks" hypoth. is correct. Hvr, in G.A. most of the work is not quite right: i.e. they should be trying to model the variation of G w. structure of the cand. Instead, they seem to want algms that give G's that are "close" (or w. large "±", but measure of .28 G from known cands.)

One way to look at conventional G.A.: we want to make the space of cands more "Continuous", so we can use the many optim. techniques that have been devised for

---

So: several (perhaps related problems):

1) Revising theory in Physics in view of new data that does not fit old theory (.09 – .20)
2) Choosing a good cand in conventional G.A. — on basis of knowledge of G's of many other cands,
3) " " " " " " $\leq$ G.A. — (A more general Goal)
4) (Searching for augmentation (or new code) for a corpus where one has coded the old corpus O.K. 35.32 ff
5) → 22

8.17.98 TM

G.A. , SGA .09 ff, .21 ff

.01 (1:36.40 start) 5) We have a (TM) w. an assoc. p.d.: When solves 1 or more problems, How best to update the P.D.? (a) Consider case in which t. P.D. has only "simple hours" that don't look at t. "traces" of past prob. sitns. (b) consider t. case of a more general hour in which traces are considered.

(SN) on G.A. Given past rounds $[E_i, G_i]$ i Thm associated G_i's. We want a Grammar passing thru the G_i seq, but G_i is given wt. that is some monotone ↑ funct of G_i. $\sum$ slightly in t. direction of S.G.A. } closer to Conventional G.A.

.09 [Actually, much in t. direction of SGA! — at least my early formulation of it! — In fact, this might be a sort of Vindication of a certain class of G.A. method!]

.12 Well, not so easy! Say we ambush 2 rounds w. G values $G_1 \neq G_2$.

.13 We expect t. offspring to have G below $G_1 \& G_2$. In general, it doesn't. perhaps one pool of param. adjustment is that [.12 - .13] is apparently true. We devise a trainer → Round$_1$ ≠ Round$_2$ has G_1, G_2 resp. We don't even combine Rounds i | $i \geq 2$ directly, but., we just use t. branch laws. Then create G rounds $i \geq 2$ to create a new round, using t. same/struck params that Created rounds $i \geq 2$. Its much more like currently view rounds on t. basis of t. entire corpus of rounds, rather than "combining 2 rounds".

.20 10
55
15
36
5 1/4

.21 There may be some sort of "fixt pt. Thm" to effect this!
Say we have a set of params $\vec{\theta}$ that distr. a stoch. dram. Let $\vec{x}$ be a round. Then $p(\vec{x}, \vec{\theta})$ is t. probly of $\vec{x}$ wrt parms, $\vec{\theta}$, & tries somehow digitally describe structure. We also have $G(\vec{x})$ which is given empirically. So I want to

.24 find $\vec{\theta} \ni p(\vec{x}, \vec{\theta}) \approx G(\vec{x})$; ≈ could be that p & G give about t. same ordering" of t. $\vec{x}$ rounds! If t. rounds $\vec{x}_i$ are discrete, we can use as a distance below t. P & G functions on $\vec{x}$, the average distance below. to orderings of the $\vec{x}_i$. I.e. say $N_G \vec{x}_i$ is the integer telling t. ordering of $\vec{x}_i$ wrt G, similarly w. $N_p \vec{x}_i$; Then we want $\vec{\theta}$

$\Rightarrow \sum_i \left| N_a \vec{x}_i - N_p \vec{x}_i \right|^{\gamma}$ is minimal. [Atrim! Adjust γ, perhaps ↑ in γ as one gets closer to optimum.

.30 .21 it's t. original SGA! T. more general SGA devises a funct $F(\vec{x}, \vec{\theta}) \ni F(\vec{x}, \vec{\theta}) \approx G(\vec{x})$ (as .24), with t. additional constraint

.32 that for a given $\vec{\theta}$, it is easy to find $\vec{x}$'s w. large **F**.

T. formulation of .30 - .32 suggests a criticism of t. model of .21 - .24. I.e. I wanted to use a branching rewrite rule grammar, probabilized. ($\vec{\theta}$ #'ware probys). & get very likely $\vec{x}$'s I set all $\vec{\theta}$ components (which are naturally on $[0,1]$) to 0 or 1, depending on which was closer. This seems to group into equiv. classes, all param. values >.5; & all < .5 into another. If there are k dimensions of $\vec{\theta}$, there are just 2^k equiv. classes. — Well! Maybe not so bad! →

8.18.98 **TM** **GA, SGA.**

There are just 2^k possl. orders configns. of t. system — so many 2^k equiv. classes is "fitting". !

Let us compare mixing $\begin{pmatrix} x_{i,j} \\ cond_{i,j} \end{pmatrix}$ v. wts $\begin{pmatrix} G_{i,j} \\ W_{i,j} \end{pmatrix}$ & doing this again & again — (say an ∞ of times.) with the usual method of SGA: passing a stoch grammar thru t. data, using wt. G_i for cond, $\vec{X}_i$. It would seem that this G.A. "Mixing" is a kind of way of getting a stock grammar — Hvr, it is not clear from this Argument that it converges faster to t. correct $\vec{G}$ than just selecting $\vec{X}_i$ at random.

Well, it probly does do better than random, but t. big problem is to see just Why and How!

8.18.98: TM: T. talk iBalk Expo:

.001:5240 29.01

29.37 – 30.20 | 32.01 + .40

30.09 impr. | 29.01

essentially 29.01 — 30.40; 32.01 – .40

28.01 – .40 also

21.01 to 32.40 (v. pp24, 25 omitted apparently)

↑ IS All on This Stuff

29.01 ff is more narrowly related Bod. talk.

A poss. way to write talk:

1) first write a very short talk, ~ 10 min. Which gives main ideas, & proper emphasis of what's important.

2) Then just expand various sections &/or add sections: But be sure to explain pts of sty. Shorter talk also recorded in 4. longer talk

---

If 10 min talk is diff: try 5 min Then 2 min then 1 min.

Perhaps starts w. Abstract!

UPL file ↓ from Gamm 3 ~ July 7, 198

#95 AutoMathology. AUTOMATH Lang. Pgmg.

94
142
113
138 Selected Papers
149 on Automath
152 1994
153 Ed
167 P.P. Nederpelt
180 J.H. Geuvers
181
183 selected R.C. de Vrijer
papers on Automath. North-Holland 1994
185
186) verlo

N.G. de Bruijn.) Home Page: de Bruyn

www.mupad.de/automath/

Automath → MuPad developed 13 6 1994.
Univ Paderborn.

What "MuPad" is, is not clear: its an alg. manipulation (symb., exact). Automath is a pgm. written by de Bruijn. originally It / supposed to check proofs in a texts by Landau (not Law)

I could use it. Data is maybe some of the system to do a inductive theorem prover — since steps in this book were very closer.

39

The title of the talk: "How to Teach a Machine"

Very brief abstract: We now have many effective techniques for machine learning and they have been applied to a wide variety of problem areas. We will discuss efficient ways to use these techniques to bring an initially naive machine to a state of high intelligence.

→ 41.01

See pst Abstract Screen:

First some definitions: By "State of High Intelligence": That the machine will be able to work problems of interest to me, much faster than I could; — & that it can solve probs that it cannot solve. — That it would do this in many different areas of sci & technology. To construct a Machine of this type is one of the main Goals of A.I.

"Initially as a machine": That it initially has very little knowledge of the domains in which it works problems. It has built into it a standard techniques for discovery regularities in data.

41 →

8.18.98: TM : T. talk itself Expo:

1991:32AO 29.01

(29.37 – 30.20 ; 32.01 – .40

30.09 impt.

29.01

essentially 29.01 — 30.40 ; 32.01 – .40

28.01 – .40 also

21.01 to 32.40 (w. pp 24, 25 ommitted as pscdary)

↑ is All on this stuff

29.01 ff is more narrowly related to d. talk.

A poss. way to write talk:

1) first write a very short talk ~ 10 min. Which gives main ideas, & proper emphasis of what's important.

2) Then just expand various sections &/o add sections: But be sure t. main pts of t. shorter talk are retained in t. longer talk

If 10 min talk is dif't: try 5 min Then 2 min Then 1 min.

Perhaps starts w. Abstract!

UpLifts etc. from Gamma 3 ~ July 7, 198

#95 AutoMath lang. AUTOMATH Langs Pgms.

94

142

113

138 Selected papers

149 on AutoMath

152 1994

153 Ed

167 P.P. Nederpelt

180 J.H. Geuvers

181

(183) selected papers on Automath. R.C. de Vrijer North-Holland 1994

185

(186) verb

N.G. de Bruijn.) Home page:

de Bruyn

www.mupad.de/automath/

Automath → MuPad developed at Univ Paderborn. 13 6 19d.

What "MuPad" is, is not clear: its an alg. manipulation langs, systems.

Automath is a pgm. written by de Bruijn. It originally / supposed to check proofs in a texts by Landau (Edmund not Lev)

I could use t. data & maybe some of t. system to do a inductive theorem prover — since t. steps in this book were very close.

2.6 The title of the talk: "How to Teach a Machine"

Very brief abstract: We now have many effective techniques for machine learning and they have been applied to a wide variety of problem areas.

.29 We will discuss efficient ways to use these techniques to bring an

.30 initially naive machine to a state of high intelligence.

→ 41.01

So put Abstract on Screen:

First some definitions: ~~By the~~ "State of High Intelle": That the machine will be able to work ~~solve~~ problems of interest to me, much faster than I could: — & that it can ~~do~~ solve probs that I cannot solve. — That it would do this in ~~a wide variety of~~ many diff't areas of sci & technology. To construct a Machine of this type is one of the / main Goals of A.I.

~~By~~ "Initially naive machine": That it ~~would~~ initially has very little knowledge of the domains in which it works problems. It has built into it ~~machine learning~~ a standard technique for discovering regularitys in data.

P 40 on Sol 78 41.01 →

8.18.98 . TM : on Sol78" : ganzn of ganzn. (copy → sol78 files)

Sol78: comments: On t. proof to Thm. 3 for unordered sets of finite objects.

More 2

I had the idea that t. corpus could be $A_1 \vee A_2 \vee A_3 \cdots$.

where A_i are t. finite binary strings & "$\vee$" was a Reid symbol. A constant fast generation of t. seqn i is that after $\vee$, t. generating machine must return to t. same state.

I then had a generalization of this corpus, so that some of the A_i were not finite objects, but prefixes of finite or infinite objects. So one Q is: Do we have to tell T. M. when an object "ends" & if so, whether it is ~~a~~ t. prefix of a finite object or an infinite object?

8.19.98 : I could start by considering a corpus of finite objects: all of which have a prefix of a (possibly) infinite object. **Woops!** This could be a very impt. kind of problem: Say SM, HR, ... !!

229511
195218
37293

137!
4:01

For SM: One kind of possible soln: T. t. corpus is the set of finite histories of the different stocks. — Not merely all of same length. T. code consists of a self-del. code common to all of t. stocks, plus many codes for each of t. stocks w.r.t. t. common code.

Vector Graphics

Perhaps an easier way to think of it: There is only 1 t. s. — this a vector each day. This is then very simple if all stock histories are of same length. While it's not a general soln, this soln. can be used to check a proposed "gen." soln.

We could have ~~zeros~~ null components for vector, for times at which stock did not exist.

This vector idea is good for capturing t. fact that on each common date (day, time) for stocks have a common characteristics: Also that t. pasts (precedents) of some stocks predict others.

Seems to be Impt. idea: But needs work!

→ 52.10

52.20 −.35 13 2

Two-stock soln.

40 item Sol78

The ~~initially~~ proposed Method: Starting with the naive machine, we give it some simple problems to solve. Using some built-in ~~problem solving~~ general techniques, it ~~is~~ solves these simple problems. Having solved these simple problems, it is not so naive. It knows just how to solve ~~problems like~~ that kind of problem very rapidly. It may have learned some heuristic techniques by reviewing its own problem-solving experience.

We then give it some more, harder problems. Building on its previous experiences, it is able to solve them.

In a similar way, we give it progressively more difficult problems, that eventually bring it to a stage of ~~[illegible]~~ & next problem solving skill.

~~Then are~~ Two Critical Questions: 1) What is this General problem solving Method?

(2) How do you design this sequence of problems of increasing difficulty ⇒, so that the machine ends up in this ~~desired~~ state of high intelligence)

— The First problem is solved by Levin's search procedure. ~~Examples problem~~

One broad class of problems is ~~we call~~ inversion problems. We have a well defined function of, say binary strings. We want to find the input ~~that~~ that function that gives a certain output. The P and NP problems of Computational Complexity Theory are of this type.

.19 Levin's search procedure:

Give ~~[illegible]~~ x_i p_i best ordering of trials

w. ~~[illegible]~~ $\frac{c_i}{p_i}$ as times for soln.

If the apriori probability of a string x_i being a soln is p_i, and if takes time T_i to test that string, then the best ordering of way to do trials is $\frac{T_i}{p_i}$ order ↑ smallest first. The ~~[illegible]~~ if p_j and T_j are the probability and testing time of a solution, the expected time to find that soln is about T_j/p_j.

Levin's search procedure is a good approximately implementation of this sort.

Normally, "blind search" (even with Levin's improvements) is very inefficient — but it's possible to use heuristics of various kinds. They can be implemented by suitably modifying the probability distribution, p_i — ~~[illegible]~~, the

If you have incorporated all of the heuristics you know into that probability distribution, then Levin's search is at least as good as any other search using those heuristics.

Perhaps Give less detail & only results: Refer to papers for details.

→ Say More briefly: at .19. Levin's search procedure uses p_i the apriori probability of a string's being a solution, as well as T_i, the time to test that string to decide the search order that tests in an optimum way.

2489700
CBO 1030
Soulers
15
10:30 PM
Guilty
Goldreich.
10 Ingram
Boston
(253 comb. S.
comb.

If the probability distribution has been modified to express various heuristics, Lenat's procedures at least as good as any search guided by those heuristics.

It can also be used to solve a broad class of optimization problems — and since the discovery of heuristics can be expressed as an optimization problem — we can use search to discover heuristics.

SN The idea of "Tools" is cost of developing a tool: e.g. a screwdriver, a hyperbolic accelerator, a very precise weighing machine; the differential & Integral Calculus

Say you are I. Newton & are mainly interested in physics, rather than math. You take a few yrs off physics to develop the Calculus, because in the longer run, it will be your all over productivity.

However. If you have too many years to live and it's going to take 9½ yrs to develop the Calc. — you'd only have ½ yr to use it for physics. (But you could give the tool to other physicists — they would take it!)

A very good idea is usually/likely based on how to accept its use.

What we have, then, is a technique for solving a sequence of problems and also a method to discover heuristics to help solve those problems.

Perhaps Give Appendices at end of talk: More details on Lenat, it's "optimality"

Say 15 minutes to this point. The next Q is: How do we design a set of problems that will lead the machine to a state of high intelligence?

Explain how Concept Nets work; What they are; How used to understand prob. solns.; How used to design TSQ's.

3 things I've designed:

Troubles w. them ① Cost of retrieval from storage was too high.

② System didn't use sub cases of previous prob. solns.

→ Solns to these: diffs >

Re: 4. methods of Solving: I'm not sure that Good TSQs can't be written this way: I only know that ten relatively simple methods I've used have turned out to be far more difficult than expected.

2 troubles w. Solving:
1) Cost of retrieval from storage was too high
2) System didn't use sub routines often used in previous problems.

N (SN) Perhaps instead of Giving a quick ride thru the main ideas! Do that more weakly, & give a few pts. in more detail, so a pt. audience can really understand it.

RMM

SN IDEALLY: In Lenat! (partial) Success or failure at one pt of the search net, should change pc's at many other pts of the net!

Just how could this be implemented? Probly best done via time sharing! But maybe easier by T-2T method!

8.24.98 TM: Expo ~ Prof. Manor in Israel,

New Visas in Thy Envt. Design: as a TSG for the envt.

The idea that "says. ab. probs. must be Universal" but it wasn't the nearly as essentially for Humans to train.

What are some other ways to train Machines? List a few:

1) comparative lrng. (Jin w Alifa, G.A.) each organism is TSG for another

2) In G.A. each optim problem is solved as TSG of TM probs

3) Get TM to read English, learn use of Human text books for training it.

4)

⋮

.10 Remarks on Value of ~~enormous amount of~~ enormous nr. of papers on Mach lrng: Great Variety: Many (if not all) can be put in common format (advantages)

(a) Common lrng. systems can use cmps. obtained in "other kinds" of lrng.

(b) can be used to understand, criticize, improve.

.15 SN On the diffy of TM solves a problem using a lower L cost soln. than using the concept that the trainer wants used. What to do (1) run it sec'dly longer: it may pick up the desired conc.

(2) Use a distant prob. to get TM to acquire the conc. Try various probs: If it always turns out that there is an "easier" way to solve the problems, then the conc. you want is likely to be inessential. That the most of the TSG can be done using other, better, concs. If it really is essential to a new class of problems, it would seem strange that it's impossible to find ways in which this conc. can be induced. If worst comes to worst, find a way to program that conc.

Note: It should be possible for a human to look at TM's soln. to a problem & easily tell what concs were used in it. Solns will in general be "Simple": not like G.A. or ANN solns to problems

Perhaps do .10 first (value of other work)

I'd like to be able to take various existing ML pgms & integrate them into TM.

I don't yet see how to do this.

However way I try to read thru much ML literature & try to integrate the implied heuristics, into TM.

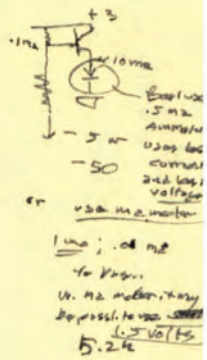
.33

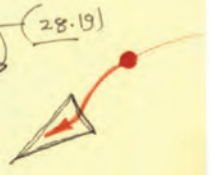
On "Long. Lrng": Early infant "lrng lrng" is on effectively small vocab. Even if the vocab used by Parents is large, there is only a small set of words that are perceived by infant as relevant to it. This "partitioning of the corpus" is a usable hour.

(28.19)

.36

(See 44.01-03)





→ 44.01

82598 [redacted] TM expo 44

.01; .40 1) Use warnan "early/long" to reduce search cc for useful models. The past experimental results are not very "sharp" — much anecdotal, small ssz, poor controls, ect. Bias in observation.
.03 but are fine for reducing size of such space.

2) for talk — do give examples — Try Berwick, [illegible] Book on Chomskys "Universal Morphology".

---

for talk: Main ideas: ① That lots of work has been done on ML: Much of it useful.

② ALP (& other Genl induction models) are able to integrate, understand, debug this work.

③ The main remaining problem is TSQ design.

.10 ③ So what can we do w. all this learning skill? ⓐ Expert systems that learn; that eventually acquire skills far beyond those of their designers. ⓑ A Generalized E-S. — T. Scientists Assistant. ⓒ Creation of truly "conscious" machine. ⓓ Human's companion, friend, advisor.

I'm mainly interested in this last.

④ What has to be done to bring it about.

So I gave maybe 20 min talk to Grace (on walk.) — Perhaps start w. digest of Goals of talk! 1) Motivation to read (More complex) paper 2) take home ideas to push, to an Audience, it would be — [illegible]

Much longer: It was an intro. Genl. ideas:

That we now have lots of work on ML, much of it useful — but using all kinds of diffrnt lrng techniques.

We have some General theory of lrng. — Mention Chomsky's objection (Chomsky 1980) — that could be used to integrate that work into a coherent whole (discuss advantages of this) Also discuss L Srch [Hand Genc. Nets]

(Also discuss 5 or 6 difrnt "General Theories")

What can we ultimately do w. ML.? — what would we like to do? (a) See .10.

How do we go about reaching this goal!

While I myself, am mainly interested in The Scientist's Assistant — The techniques for reaching this Goal are about the same as those needed to approach the other Goals.

Mainly, to problems, even w. the best of learning algms, we have to present material to the machine in some kind of order of difficulty.

Difrnce betw human infant & V.g lrng Algm: Difference of "Bias": prenatal knowledge. programming in "common sense" is not a good way to do it. (Most of what is regarded as "common sense" is learnt after child is born.) Chomsky has suggested that the nature of some prenatal biases for learning Languages.

One subgoal for a very smart Learning Machine, is the ability to read & (to large extent) "understand" printed text! Reading is far more useful than conventional programming. Criteria for Understanding!

1) compression → prediction 2) Ability to answer Q's about text.

We want several codes (not just "best"): sub optimal codes are useful for modifying when shortest code no longer works well. (A scientist will store several explanations of the world)

2) Discussion of other work on this problem: Work on story reading & understanding: (By children / By Adults)

a) Blocks world (a world to understand, to talk about) — early breakthru.

c) D. Cohen: Learning by reading text on Contract Bridge.

d) Studies of lang lrng by infants: Theories useful — Not as "facts" but as suggestions that effectively reduce size of search space. (.01)

82698 TM Expo: XFML Grammar, Chomsky Hierarch (.05)

That's a sort of Talky introduction. Now, I want to give some more exact ideas on how ~~I ...~~ I go about solving problems?

Some possibilities to expand (1) Conc. nets; Lsrch, optimality — 43" w cprom. 43" w.o. c prom

(2) Application to Grammar discovery.

.05 (SN) The effects of a "transformational" Grammar can perhaps be obtained by having a Meta Grammar construct a more complex cfg. The ~~Ra~~ end result total complexity of a corpus + dern of Grammar, would be about t. same. T. meta grammar need not be a CFG. (or C Dependent G)

(3) Perhaps describe how I deal w. problems in D. Cohen's Pgm: EBG, case based CBR, etc. I "Shot lrng." , "I,A", etc.

628 / 356 / 628 / 3356

(SN) "On Government and Binding." These are (Grammar) rules for deciding ~~which~~ what pronouns refer to & "who does what to who" in a sentence.

My impression: while there are rules that usually make it unambiguous: — & Sentences are often ambiguous about this — That one has to look at context & sometimes actually "try out" various assignments to see how they fit, in view of subsequent development in t. text.

I think Chomsky is not interested in this sort of thing. He feels that there is some "Real Grammar" ~~...~~ that almost always decides these things unambiguously, & that any ambiguity is rare & usually due to inept use of t. grammar. Perhaps it would be part of "Discourse Analysis". [So is out of analysis ... that of one Sentence.

I could describe ~~...~~ in more detail how "concept Trees" work: How Lsrch finds solns. within time $\approx \frac{t_0}{P_i}$. 2* or 1*. Also, just what t. problems are & how I expect to solve them.

Perhaps discuss EBL & CBR as methods of IR.

Maybe "one shot lrng" (MDL & MML may differ on this).

One way! is introduce up to "How to teach a machine" Part. Discuss "Blocks World" (not a lrng pgm)

Then talk about The series I've written: That [(external) Goal was to get machine to understand Algebra, then begin text. Q's, A's about Algebra.

~~...~~ Then discuss TSQ's I've written: What was wrong w. them (2 things)

How to fix those things: "IR"

"A better Approach": To start w. Text (Text

Oxbridge / Oxbridge / Oxbridge / Oxbridge / ABC / Previous

[* One indication of "Mature Science": When ~~...~~ Researchers are able to Build on previous work.

To ~~...~~ some extent, this has occurred in ~~...~~ Machine lrng ~~systems~~ ", ~~...~~ ~~In Machine Learning it has occurred to some extent~~, but expressing any of that ML work in a common format, with clear theorems on expected error could give a tremendous boost to this area.

[redacted] It gives us a way to compare learning techniques so we can integrate them.

It enables us to Analyse lrng techniques to criticize, to improve them, to understand them.

82698 TM Exp Ra≅ EBL EBL

(SN) Ra ≅ EBL: Suppose we solve a problem: t. prob, soln. pairs [crossed out]
(PR₁, SL₁) In EBL, we try to Generalize: Make less specific, Pr₁,
∋ SL1 is still a soln. (SL1 is, in general, a function of PR₁, so if we change
PR₁, t. Q is, is the same t. same function of PR1 ≡ a soln of PR1?

.05 In ALP: when a problem occures, PR₂, we want to know if "similar problems" have occurd in t. past. We have a "metric", so [crossed out] we look for
$PR_i \ni |PR_i - PR_2|$ is "small" this means that $PR_i = PR_2$ + a small no. of bits.

The reason we are interested: if PR_i, SL_i occured in t. past, Then let us define
$\alpha \equiv Pr_i, SL_i$: Then the pair Pr_i, SL_i; Pr_2, SL_i can be compactly coded
=> α, α + a short code. So the SLi is a more than average likely soln to Pr₂.

.14

.15 T. fuzzy. discn. is a genzn. of simple "best hot lines."
We observe AB (A is a long (bit) string). Then A occures later.
We define β = AB; s. that AB and then AB later have short codes.
Then when A occured, B is likely to follow.

In .05 – .14, $\alpha = (Pr_i, SL_i)$: α is transformable into PR_2, SL_i ($\equiv (Pr_i, SL_i)$)
with a short code, if Pr_i is xfmable into Pr_2 w. a short code (sim. short code).
So, when Pr_2 occures, we look for Pr_i in t. past ∋ $|Pr_2 - Pr_i|$ is small.

How is this done? well, when t. soln, (Pr_i, SL_i) is found, we put in our (file/index..) all Pr_j ∋ Pr_j is close to Pr_i. ~~when we~~ Then, when ~~we~~ Pr_2 looks into t. past, it finds a Pr_2 that was close to a Pr_j in the past & it knows that Pr_j is of interest.

Note the metric we use, will be domain dependent — but just how is this reflected in the code structure of t. metric?

This whole thing is rather unclear!, But I think there is something
Basicly right about it. One main difty is t. relation of Pr_i to SL_i.

.30 Perhaps: in t. past [crossed out] PL1, SL1 occures & SL1 = F(PL1).
or, say we have (PR1, SL1), which is xfmd by short code to (PR2, SL2)
∴ SL1 solves PR1, then SL2 is a good trial for soln. of Pr₂.
PR2, SL2 is an "Analogy" of PR1, SL1, if t. xfm from one to t. other is "cheap"
It will be "cheap" if it has been successfully used in t. past as an "analogy
.36 generator".

Not BAD

.30 – .36 is not BAD) — But it sounds "too good". i.e. problems are not so likely to be solvd by analogic xfmns. — I'm not so sure! If t. analogy is very cheap (i.e. very successful) then it could be very effective in solving new problems.

Grove 17
→ C's line 3
Conduct com Bad.

C's & AEB are not needed on (C:/)

82798 TM Expo

So there are several related ideas in ML learning: EBL, Case BR (Conceptual Clustering) | GA & SGA? Related to?
One Shot learning, IR (Info Retrieval)
(Genzn is ≈ same as Lrng)

EBL may not be what I'll do. Its more like one has solved a problem, & one is able to Genz. this soln., so one has solns. to many problems. T. Genzn. is done just after 1. soln, so t. indeed generality will be immediately available.

e.g. Tom Sawyer Gets his friend to paint fence by getting friend to perceive it as lots of fun, privilege etc.

Genzns.: Many possl. ① way to get/pay work done. By convincing friends its fun or by paying them in other ways etc. How to get others to not find work unpleasant. e.t.c. We want to also find ways to index t. Genzns. so the Genzns. can be found in t. future when they are needed.

(related to many explanations v.s. 1 best explan.)

(NB) As opposed to many EBL rschrs, I don't want only t. "best" Genzn.; I want as many Genzns as t. think would be useful in future probs.

So actually: perhaps I'm working on 2 slightly different things:

.12 — 1) Unification of Learning theory & practice. [Show how other work on lrng. could be done better w. ALP.]

.13 — 2) Text world & TSQ. → item ② is TSQ construction. "Text world" is ??? adding to stable.

I'm also trying to unify these 2 things

Studying 1) seems to give various techniques useful for 2).

Text world; Also involves study of lang lrng in children, (animals?). Theories by others. A.I. & ML work in this Area: story understanding. Text book understanding.

Perhaps put various papers & general paper on "Home Page".
e.g. "What to work on Next"
② Comments of Chom's Universal Grammar (≡ Grammar Grammar)
③ On Chomsky's "Minimalism"

(.12)(.13)

One aspect of unity of problems 1 & 2: In (.12) TM could learn to understand some domain: In 2) we could write things about that domain & discuss t. domain w. TM. ("T. more TM knows, t. easier it is to teach").

So, it getting TM to lrn reading: If we have other ways to teach it things — perhaps easier than reading/ — then use those ways about them

common cum is corrupted?

(SP) Re: Math Problems: see NN: (82798, 82898)

One kind of TSQ could be "proof checking": in which gaps betw. steps will be of varying size. Eventually, TM should be able to discover entire proofs w.o. being given intermediate steps.

The main problem here, is t. search for heuristics — which are usable in most of Math — not merely in t. part they were discovered in. (Note Landau's book: w. small steps (see ref. in \SS\ file on DeBruyn.htm).

Heuristics: I can get many of them by reading stuff on A.I. theorem proving: Then, after I know t. heurs, I can design problems or "tng. seqs" for TM to learn these heurs.

In a Math problem (or Proof), t. axioms or thms needed may be "given" or t. Mathn has good idea of what thms are relevant. This last is a kind of "IR" problem. — Av.g Mathn will have good, non-standard (analogic) ways to do "IR", to guess at which thms. (or parts of Math) are relevant.

.37 Also, t. problem of Making good conjectures — Conjectures that t. Math Community would find "interesting" is also a good problem in induction that TM could work on. Solving this, & being able to do "proofs" would make TM a v.g. Mathematician.

DeBruyn.htm
Examples of a text written in Automath. (1968) in Vol 133 of a 1994 Book
N. Holland Pub. Co.
Amazon?

K.L. Schubert: JACM 1974 21(3) 436-445 July On Pgm Minzn.!

Common. Sys Com NEB C.S. ZZ AEBPB Conf SSS

→ 49.01 Spec

82998 TM·Expo — recurrent Neural nets; GA. | Feedforward ANN universal function Approximators. 48

Near start: List of Universal Learning Systems:

ALP-RLP
MML
MDL
C.V. dim; SVM
SOAR
CYC, Eurisko, AM

Discuss: "Universality" meaning: usually very restricted. — Perhaps put into letter to Gaum.
2) How ALP-RLP, MML, MDL differ:
Emphasis of "The Best Model" in MML: why this is not so good! We want other Models (Theories) to make new ones! That the "bad" theories of the past have some good features. One should know about them & use them in [redacted] trials for new theories.

.07

[redacted] MDL tries to avoid a priori by choosing least controversial a priori. This discards useful data. That this data is subjective, it a necessity — Makes the statistician's work harder, but not impossible.

C.V. — dim, "Support Vector Machines" SVM: Ellen have been derided this morning. I'll not sufficiently familiar with them to give any deep criticism. My impression is that the V.C. dimension is a variety of coding and it probably corresponds [redacted] to a kind of code length for a concept.

The Soar & CYC appear to be less probabilistic than the rest. They [redacted] do use a great variety of induction techniques. [redacted] Analysing them using any of the previous methods could give useful understanding of both the system analysed and the system used for analysis.

.47 Other Useful Things to Analyse using Universal Induction Systems:

1) Meta Analysis — (4) Analysis of recurrent Neural Nets.
2) GA (& SGA) variety of GA: w.o. w/ recombin.
3) ~~Chaotic sequences and~~ prediction of chaotic sequences v.s. prediction of somewhat random sequences

Say: ALP is good for 2 things (1) Getting good probability values: Also getting good models for understanding things (2) Solving problems via Lsrch.

Discuss how it differs from MML, MDL. — Partly in how it deals w. Halting problem.

Studying work of others on ML. is useful because they may have lots of heuristic techniques that I want TM to learn, & I can "teach" TM these heuristics by giving it problems or some other kinds of training that elicit discovery of those heuristics.

.30 Heurs (or any kind of lrng) can be "programmed" into TM ("wired", "soldered")! This is a valid method of teaching, but should be used only as "last resort". (E.g. in the soln of "Peter Bergmann's problem" 43.15–.33) Perhaps explain (to self as well as Audience) why "real lrng" by TM is better than "soldering".

Still needs work — It is because I don't know TM's wts, concs well enough to always know l. best way for TM to solve a problem. In "real" lrng, TM fits a soln. in an optimum way, into t. concts concs & their wts that it has at that time. For me to "wire in" a soln. is very presumptuous on my part — that I know TM better than TM does!

.38 (Also see 49.12–.20)

.01 : 47.40 Hvr. — even w.o. t. conjecture stuff of 47.37, TM could solve classically difficult problems in a "V.G. Mathematician".

Actual Doing Actual proofs in Geometry, Algebra: I it's almost certain that these problems have been formalized (Geometry in particular — Algebra easier) so that it would not be hard to devise tests.

In doing proving theorems, if t. CJS is too large I'll have to "make limits". Sometimes: Suggested constructions: suggested subgoals ("can you show $\angle A = \angle D$?")

Learning how to conjecture auxiliary constructions.

Construction Probs in Math. (17 sided polygon!). Ability to do constructions

Might be useful in TM's conjecturing what constructions might be useful for a proof.

.12 : 48.30-.38 → or Tell TM how to solve it. Look at CJS. instead of TM doing the search, act as if TM did the search & found the big CJS soln. TM needs

(This Term (or concept) .... and reasoning by ~~more~~ finding it a very big CJS, it isn't able to use recombinant these the components of this soln. for problems of lower CJS.))?

This could leave a "big hole" in the try. of TM. However, if one does it many times, it may be that one finally "fills in" TM's ignorance. — After all, a human "student" is able to use many ideas that he doesn't understand — but normally

.19 .20 the more he understands the ideas, the better he is able to use them. better

See .36

Peter Bergmann

So: Is the Bergmann problem's "soldered in soln" related?

Well actually, "soldering in" may not always be so bad! : If I really have a "reasonably" accessible

C.J.S... Then its really not so bad — Its almost the same as having TM do the search.

A Big advantage of TM doing the search is that it checks on the working of the systems.

"Soldering in" a soln. in which we know the C.J.S. & the pc's of the components of the soln., is not so bad. What is Bad, is "wiring in" a soln. that seems to be of enormous C.J.S. ("wiring in") (So that TM is unlikely to be given cc's of that level for problem solns)

.30 So we are essentially "telling" T.M., & TM doesn't know "Why?"

"Knowing Why" means that suitable sub-concs. have been reinforced (& perhaps more important) conditional pc's have been reinforced: the conditional pc's are more like heuristics — They are much narrower in the implications they have.

.33 Give. — They are ≅ "IF solns". →.36

The only value to TM of that soln., is the soln itself. The updating of the relevant sub.concs. is not useful. (?).

For probs of excessive C.J.S., we might try a forward, backward chaining search to try to find a lower C.J.S soln. for TM. It still might be impractically large CJS, but still the reconfinement of probs & cond. probs of its factors would be many useful to TM than the corresponding elements of the "Gross" CJS soln. (?)))?

.36 .19 .33 → Whereas I would "Memorize" something in college, that I that was really impt. — I'd try

.37 to understand" it as well as poss.: The limits of its areas of applicn., How it could be generalized

.38 to other applicns. — But not understanding knowing a heuristic path to the "something", it's much harder to do .37-.38 — much less "yield."

→ 51.01

Discussn. w. Alice M.   I CBR! We have a present problem: We look into the (suitably indexed) past to find similar situations — Using various types of Metrics. Usually, we pick the "closest" case (nearest neighbor). Or They may pick several cases in past & try to mix them.
At any rate, one try to xfm the past cases into the present, (if they are not directly usable). using Analogy
Or try to combine several to be relevant to the present case.

Usually we do CBR when we are in a hurry: Mixing several cases is time consuming & so not so often done. [In general retrieving relevant case(s) is easiest part. Xfmg. them to be useful is hard — More time consuming.]

Memory based reasoning (Walz) Dave? seems to be ≃ CBR: It uses large data base (data Mining) & fast Machine (say Connexn Machine).

EBL^G: When problem is solved, we try to Genz. Soln. before indexing it, for use in future prob. solving (into retrieval).

In CBR, the "traces" of the previous soln. are retained, so we may more easily discover what is relevant to the present situation.

In both CBR & EBL: One has present problem:

One wants to retrieve past cases for induction to help w. present problem. for IR: EBL generalizes before indexing.

In EBL I think we use various indexing schemes to enable future retrieval.

> Collegues
> Use of friends for IR. Tell friends best ideas; Then later, discuss one's problems w. friends — They may remember my great ideas or their own developments of them!

Now, Having the present & several past relevant cases: How best to make present decision/solve present problem? RLP can deal w. this, in both EBL & CBR.

The problem of EBG is also one that RLP can help with. — We can suggest very many different Genzns, depending on expected future problems.

In General the "Expected future problems" Q. is diffc.: It may be that an older TM goes over its own past history & reindexes the past in view of a better idea of what future problems will occur. This idea of "Going over the past" occurs in a trivial way when TM updates various (conditional) probabilitys in its "Grand P.D.".

spec. 57.01 →

.01:49.40 : Math problems: In Geometry Humans have advantage over TM, in having "Geometric Intuition". What to do? Well, TM has advantage over Humans; It can make hy accuracy "typical" diagrams, to tell w. very hy proby, if a ~~subgoal~~ conjectured subgoal is true. Also, after TM has proved lots of theorems, its totality of heuristic knowledge, will amount to a kind of "Geometric Intuition".

What the forgg. does do, tho, is make TM's intuition much different from human's — So it becomes harder for human's to know how to train TM. → 54.22

.09

.10 O.K. Starting over w. Talk: | Sentence #1 of Abstract (39.28
We have these ~8 difrnt universal methods of Lrng; 48.02
[crossed out] Discuss what they are, what they are not
I will discuss ALP in greater detail because I'm most familiar w. it, and Certain of its properties are ~~of most~~ most relevant to the second ~~%~~ of the talk)
~~[illegible]~~
What (should / can) we do w. them!

12 I) Apply them to classic problems in induction. (48.17 13 ~ 1381)
[crossed out] Among them, analysis of ~~[illegible]~~ ML in existing literature. CBR, ERL.

H:\ AR32E301.EXE — This is Enternet Ext. ov (Vol 3.) — std on By ~5:15AM — 59.01

Also if one really understands the learning process, one should be able to ~~[illegible]~~ make a machine that can learn to...... ~(39.29)

I will now discuss part II: how to use these techniques....
How ML has been mainly non-incremental. ~~[illegible]~~ Best performance (Koza) very non-incremental.
How ~~[illegible]~~ non inc memory works. You want to design a good ~~[illegible]~~ microscope. So you start out with a human infant, educate him for 25 years, then give him the problem and voila! He is able to design this great Microscope. ~~[illegible]~~ Now, you want to design a telescope. You take this very capable engineer who designed the microscope, Wipe out all memory of the last 25 years. Then you start from scratch & re-educate him. In about 25 years he is ready ~~to design the telescope~~, ~~[illegible]~~ you give him the problem and voila! he is able to design the telescope.

Seems to me that we are making inefficient use of this man's talents.

We want to start with simple machine, ~~train~~ and it to learn to solve simple problems, then use the solutions and ~~[illegible]~~ solution techniques of these simple problems to solve more difficult problems.

If we really understand learning, the main problem is to construct this training seq. sequence that brings machine from initial ~~[illegible]~~ university state to ~~a state~~ an overall state of high intelligence

8318 TM EXPO: Conditional probs & ALP Gonzo: see (40.01–.40) for earlier work on this

:40.40

.01 Novel method on learning and the Pd.! Perhaps the Pd is expressed as stochastic operator: For each input, there is a Pd usually different output.

.03 Given a set of I/O pairs, to construct optimum stochastic operator.

Go over my soln(s) for the stochastic operator: There was a simplifn. if the

.05 input contained "no info" (i.e. it was random). Also, say the corpus is a (big) data

.06 set plus I/O pairs and maybe a few prefixes of infinite sequences!

.20–.35 may solve this problem.

Write up a Bibliography of work on this!! I do forget! — put this on my kind of index,

40.40

.10 Another kind of corpus: Say a finite set of prefixes of self del. codes of finite objects.

Well we can't tell if they are finite! This may be an infinite point. Say we

are given a bunch of prefixes of finite or infinite objects. To extrapolate

any of them. Some may be complete objects, but we have

no way to know this!

.20 A possible way to do .10: We have a univ. w. 2 inputs ① is self del.

machine desn.; ② = D_j is random seq. We want the product of the probabilities that

a prefix of the output will be each of the objects.

i.e. Say A_i is one of the objects. Then for random input (for ②

from Machine desn in ①) the probability that A_i will be a prefix of the output

is $P_{i'}$: Then the probability of the corpus with that machine desn.

is $\prod_{i'} P_{i'}$. for MDL j we want a D_j s.t. $\sum_j 2^{-|D_j|} \cdot \prod_i P_{ij}$ is max.

We then use that D_j for extrapolation of the old objects or any new (partial) object.

More exactly using ALP, the PC of the corpus is $\sum_j 2^{-|D_j|} \prod_i P_{ij}$

.30 P_{ij} is the probability assigned to A_i by D_j

(we sum over all D_j's : the D_j's form a prefix set.)

All the techniques of .20–.30 can be applied to the other kinds of corpi:

i.e. A_i can be any of the kinds of things (The various A_i's can be

different types): The prefix of an infinite string. A finite object.

The "finite object" can be any small or large. It can be a Q, A pair

.35 or it can be a large data set (as in .05 R).

essentially the same as .05–.06

So, the Q is: Is .20–.35 "a soln."? does it have desirable properties?

Note Sometimes prefixes can be parallel time series: in which case the

set of symbols for each common time point can be regarded as a vector . . .

and we combine them into a single vector times series:

How do we use the .20–.35 model to extrapolate? : Well, we just look for

"regularities" in the corpus. These regularities can be within an object or between many objects.

83/98 TM Expo

ALP Ganzn — More General type of Corpi.

A "big object" (containing, say, a data base) should have many copys in it. The Q, A's will have less interobject redundancy, but will have redundancy between that in the "Big object".

In proving 52.20–.35: we do have some difficulty: We may not be able to use the radix 3 corollary of Sol 78 Thm 3.

.05 A (maybe) distinct kind of proof! — That does not use that fact. — Assume the corpus was generated by some machine M that had a finite diam $\tilde{D}$. The treatment of 52.20–.35 will give a pc. for the corpus

.06 that is within ε a factor C of that given by M. Here C = sum of probys of

.07 all ways that the unc of 52.20–.35 can simulate M. Then the total expected error in

.08 proby of each of the bits of the entire corpus is bounded by $\frac{\varepsilon}{2}$ ln C.

.05 is the beginning of a theorem: First we show that the factor "C" is as stated;

Then, we may use the proof structure of Sol 78 Thm 3 to prove .07–.08.

There appears to be some difficulty in the approach of 52.20–.35; i.e. if we are given an entire finite object & we are told it is an entire object;

vs. being given part of an infinite object & told it is part of an infinite object

.16 In the first case, we are only allowed self-del. codes of the object: In the second case, any codes are OK. } The theorem is provable

{ If we are given a partial code of a finite object, then a prefix of a code for an infinite object is illegal ... but perhaps this is a finer distinction than the "finite object" can be arbitrarily large!

Hmm. '16 appears to be a problem: Inf.'s some finite objects do not code differently

To approach the problem: consider the case of "only finite objects allowed in the corpus"

It may well be that this corpus has essentially $\geq$ symbols!

On the other hand: consider (.05–.07) if we use the same restrictions (or lack of restrictions) for each object in the corpus before Derns $\tilde{D}$ & then after D_2, maybe we get the same C (.06) anyway! So maybe we don't need the radix 3 Thm!

(G) or We may be able to modify corr. of Sol 78 ETZ so it applys to this 3 case! — use radix 3! (G)

Anyway, in view of (.05 – .08) it would seem that the Thm of 52.20–.35 is a bookkeeping.

That "stop" states the self-del codes: For user to vary size "stop" state, the system must have an extra symbol. (Willis used extra symbols of this sort in his 1970 paper)

We may be able to modify the Corollary so (a) it is radix independent (b) it works for a great variety of devices that compute proby. (i.e. a great variety of "Corpus Mixing") (i.e. it can usefully mix a great variety of distinct kinds of corpi)

9.11.98 TMcKgo

.01: 51.12 After reviewing these General Lrng. Models! What common needs to be done?
In almost all of ML work there is a general

General unawareness of Genl. solns. to Lrng. problem. — which stood factor
over an enormous variety of different techniques used — each laboriously constructed for
largely intuitive bases — All of this work could be done much easier, more
accurately if the techniques were generated from General Induction Systems
of known precision. We now have the "Maxwell's equations"
for all of learning and discovery. Any correct work on lrng should
can be developed from these principals

To some extent, this has been done. In a few areas of Machine Learning
very general techniques have been applied. eg. decision trees,
linear & nonlinear regression.

.368
$x^* = \min$
$\sim .692201$
$= \frac{1}{\sqrt{2}}$
.36
.37±
.38
.367
.369
.3675 .3676
.6922
.00627
.3679

A briefer approach! Just do 51.10-12 i.e. "What we have all these
general induction methods (maybe durb how they differ (a bit))
But I still, we don't have we have many plans, that can lrn to solve
probs in specific starting w. certain basic info.

These are the tools we have available to us:

[I mite mention that "T. tools" have already been used to do certain lrng tasks but very little
on "integrated" lrn tasks of any magnitude.

.22: 51.09 — Math proofs: The steps in a math proofs often are quite large CIS—
Verifying that proof is often easy, if the steps are clear. It just amounts to
checking that the theorems combine properly to give the desired result.
However, after doing the search (or simulated "search") for the various steps in many proofs,
TM might learn how to reduce their cost. It might amount to lrng suitable heuristics.

.27 The induction problem would be: Given ordered pairs:

Looks Good!

(a) stuff proved thus far + theorems, postulates; ; Also next step to be proved.
(b) The actual proof. (i.e. = theorem used etc)

Say T.M. has given many such pairs: it could then induce relation of
(b) to (a), given (a) get a good Pd. for (b).

Then, after T.M. has learned to do these very simple proofs, I can use that same
(a),(b) pair trick to get it to learn slightly longer proofs: say, proofs taking
> 1 step. Also, I might get TM to learn proof methods by observing
complete proofs — Also observing "traces" of proofs being done by real
people or by PGMs that knew how to do some proofs.

An important idea here is that I can slowly increase the difficulty of the problems,
(This set of examples could result in CBR : It could pick nearest neighbor" 55.101 spec.

9.1.98 **TM** Expo Math probs

.01: (spec) 54.90 & use a rather complex analogic xfunn to decide that a neighbor is "close". Also use of a combination of many neighbors. Indeed, one mite regard this whole induction trick of 54.27 as a "genzn" of CBR.

Judy! Walk.

Another kind of problem of "Graded diffy" (able) that can be solvd in ~ ways:

Translations betw computer langs: e.g. Basic ↔ Fortran; Fortran ↔ C, C ↔ Mach lang. (of interest.)

Say we have some machine code output of C (or other compiler), to restore t. C source code.

~~Putting comm~~ disassembly of Mach code & putting in useful comments.

Disassembly of compiled Cobol & putting in comments.

Translation ~~between~~ di of pgms betw different Machine langs. Attempts to optimize for speed, memory, etc.

.12 So TM can learn in many different "domains": putting all knowledge into a common **P.D.** {remember probys are all conditional probys; so we can use "same" condl. p.d. for all varieties of problems — (See 52.01 ff. : There is one earlier more general discussn, but I haven't been able to find it : 1.40, 2.19 : 4.30 are some references. [52.01–.03 is t. basic idea.] [I think 52.01–.03 is it.] A recent idea was that a condl. p.D. can be expressd as a stochastic operator. The input is "t. condition": t. output is a p.d. over t. desired space. 52.01–53.90 is about stochastic operators **of this kind.**

My talk will discuss a few domains: (Proof checking → Thm proving: Genl math problems ~~Translatg Computer langs.~~ Language lrng; "Text word": Ability ~~...~~ answer Q's about Math (its domain of knowledge), ~~Other~~ Impt sub-goal: Getting TM to be able to read (for Humans) text books & get a fair amt. of info from them.

A very impt. idea is that t. ALP. (condl. proby, stochastic operator approach enables us .27 to optimally mix learnd info from Many domains → 58.01

In my olde approach to tsq design, every time TM solved a problem, it .29 would incorporate that soln (& perhaps its "traces") into the P.D. ... This was the only way it lrnd. Now, t. lrng of .29 is only one kind of example, that TM uses for a corpus to help "engineer" its **P.D.** We could use only external examples, so it isn't necy to have TM solve problems (in t. training phase) (while its learning to solve problems).. but ~~...~~ using (emphasizing) problems solvd by TM has advantages ① Tells us how good TM is lrng. — so we can modify t. TSQ if necy ② TM has t. hoarse traces of its own solns; & this can have much info not ordinarily included in "external examples". → spec. 56.31

Say we have a Big corpus of "facts", etc. plus a bunch of Q,A's.

Since the retrieval of A to Q depends on info in Big corpus, the only way we can get

.03 info transfer is to have a complete dcrm of the "Big corpus" (all of its info) in some machine (52.20) w. self delimiting dcrms $[D_j]$. We could then ask for a conditional p.d. (by algorithmic induction), so we put in Q and get a p.d. of the Answer for that Q.

In the more general T.M., we will have a conditional cond. p.d., i.e., a conditional:

One is that we are in Q-A mode. Two is just what Q was asked.

.11 It may be that we get almost all of the "Big corpus of .03" into the dcrm D_j only after we have seen Q's & A's to cover the entire "Big corpus".

So, often, if a Q is about a part of the Big corpus never asked about, it will be easy to (recode user's view) D_j to get any info about that Q into D_j. We have similar problems w. "The Short Irng". We will have to make special computational devices to deal w. this problem. (.31)

.18

Another set of problems of "erasable dirty" MT. from one computer lang. to another.

Start out w. simplest pairs of langs: that are very close: then eg. decrease lang closer to machine code.

Eventually try Basic → fortran; fortran → C : A useful problem (very high level) disassembly —

Machine code → Commented source code.

This might be a useful "learning problem" for reading English text and understanding it (→ translation into usable internal language). Being able to answer Q's about English text would involve learning. Compressing text would involve short internal codes for commonly occurring objects in the English text.

.28 (SN) HASH CODING is similar to ZATO CODING in that we always get at least what we want, but sometimes more. [Also the mathematical details are similar.]

.31: (54.16) 55.40: In Sol 86 I had the idea of recompressing the whole p.d. every time I solved a new problem (or in lines 55.29-.40 every time I was given a problem soln or useful "example."): Actually, this wouldn't work because of the "short (run)" effect of .11-.18; we would have to deal w. this in a special way. In general the OSL problem is usually mainly a IR problem: We only have 2 cases to compress! Coming from the larger corpus are then indexing techniques: the factoring of an "example" into a structure multiply parameters" to yield the example.

Is this possible for a Q,A TM: We feed the Q in — & the output is a p.d. of related Q's: (Not so good: It wouldn't help us retrieve stuff from "Big corpus" that hasn't been subject of previous Q's).

9398 TM expo: CBR: 4-9-87

52

.01'.50.40: Re: CBR; It's Generalization (as well as the way it is normally used in the current ML community) can be very useful for TM & for Me!

I.e. I can try to "index" various ideas I have; also books, objects, H.W. etc.

This "indexing" can be of both practical value for my failing memory & of theoretical interest for TM research!

Japan Wikkin ... AM: -.80%

It's the General IR problem. — having 2 Aspects: the pure P.C. & the CC.

The interesting thing is that ALP does give the form of the general soln.: it tells just how the data is to be treated, & gives the significance (meaning) of the resultant conditional pc.

We have always 2 problems: (1) inserting (indexing) the data (2) Retrieval of data for retrieval.

OZ problems: The idea of condl prob'ty & stochastic operator may be a real breakthru here!: The idea of having the same optzn. routine for all problems seemed unreasonable: I had some idea of using a listing of all optzn pgms & somehow finding which ones should be used for each problem.

How to deal w. practical aspect of 1 shot lrng: "OSL"

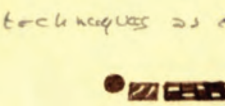
The idea Gave is: a stoch. operator on which I have the problem to be solved as input & a P.D. on all optzn techniques as output. This p.D. can then be used for Lsrch.

Dr Tomie. 354-2400

.30 DEF) Again, there is the OSL (1 shot lrng) problem. Actually, the problem is more general than "1 shot": In 1 shot lrng. The difference betw ssz = 1 (w.o. OSL) and ssz = 1 (+) is rather large; But even w. ssz = 2, we may not have enuf cases to warrant a definition, but ssz ≥ 3 may be enuf! So the usual way of making a pd. based on the past (w. definitions justified by their being useful in the past) will be often inadequate. We could always add 1 to the apparent ssz, but this would seem to be impractical for ssz = 1, since almost all of them would not be useful — i.e. they would not occur in the "Question"

So in General, I need some good ways to deal w. (≈) "OSL":

IR & "Indexing" is one "gl." way ....



One way to deal w. "OSL-type" problems: Say we have an unordered set of complete objects and one partially complete object. We want the P.D. for the rest of the partially complete object. We may also have a "Big Corpus" object.

We may not... conundrum AES & ... ...ing.

.36 → Say we make code for all of the complete objects! Then we have some "heuristic" routines for modifying (augmenting) the code wrt. the partial object — that automatically give pd's of various continuns.

Look at what's actually done in various practical OSL situations, see how they did it. (.36)

Ratio of probabilities: Definition (.23)

.01' 55:27 : Section of talk on TSQ's: TSQ's discussed:
Micro Theorem proving ~~Proof checking~~ larger Theorem proving: Gen'l math probs!
∠ Booth, Dutch Guy!, "AutoMath"
MT of computer langs: "Text world"

Other than Dern. of problems themselves Discussion of Methodology: just how proof "checking" is done.
Actually, it's "not" "checking"; but long to do small steps ("Micro proofs").—But proofs used as example source must have steps that are verifyable by TM.

We can start w. 1 step proofs, Run 2 step, Run 3 step ... etc.

Also ; simpler if t. ~~[illegible]~~ amount of available info to implement a step is small. — say w. only a few postulates & Thms.

For longer, ~~more com~~ proofs, usually a certain amt. of planning is required. We can give the machine examples of plans and of the traces of people or machines implementing these plans.

(SN) : T. Search for short codes is nominally an OZ problem. It can, hvr., be viewed as a INV problem, if t. machine tries code in Lcost order. If t. machine has an adequate P.d. to discover ~~t. code~~ a short code.

Perhaps Mention SM as source of TSQ's; Start w. early date (easy to find regys) Then as data continues regys get harder & harder to find.
Discuss Predictability of t. SM.

Befor 45:01 is 2½ talk (intro'n!)

---

comp sample paper talk: In talk at some point discuss Condl. probty using sets of finite objects.
~~Show~~ First discuss seq. extrapoln as a kind of condl. proby. — Then discuss Q, A's, Then Heuristics, | examples: (sets of Dinosaur bones) → extrapoln.

Davis & Lakat 1982
Compiled 1976
CYC started ~1984

Newton 1642–1727 84 Y.O.

---

23 A possl. way to order trials for ratio of probys $\frac{P_1}{P_2}$:

.24 Ordering is Lcost of $\frac{CC_1 + CC_2}{2^{-N_1} \cdot 2^{-N_2}}$: CC_i are times or CC's spent thus far.
N_i are lengths of codes (in bits) tried

No! CC_1 & CC_2 are amt. of CC on a particular trial
N_1 & N_2 are t. lengths of a particular trial.) N_1 ~~[illegible]~~ is conditional probability of t. codes ~~[illegible]~~ for ~~[illegible]~~ string 1, (say) given string 1 & one's previous experience in code finding.

Say we have a numerator trial CC_1, N_1 ~~[illegible]~~ p.c. of P_1

{ While its a 2 dimensional search, .24 may order t. trials.
{ My mind is not yet clear on this!

Perhaps discuss difference Betw. ALP & Computational Lrng Theory ;
~~[illegible]~~ # CLT is concerned w. Learnability of "concepts" — ~~Mainly whether [illegible]~~
~~they are learnable to a certain probability of~~ error, given a Sample size and
Roughly speaking PAC Learning asks if the ~~[illegible]~~ error probability in learning a concept is polynomial in Sample Size and Computation Cost.

ALP is concerned w. a ~~[illegible]~~ related problem; Given some finite amount of data, We ~~can~~ make ~~[illegible]~~ various statements about possible extrapolations of that data.
~~E~~ Given any finite amount of time ALP can give approximate probys for any of those extrapolations. Given more time, new probys will be obtained — and are expected to be better.

9.3.98 TM appl

~~Introduce~~ We don't particularly care if the expected errors poly nomial in sample size or computation time. We just want the best values we can get and an estimate of precision, so we can use the estimate. Very often the very practical ALP problem is much easier to solve than the theoretical PAC learning problem.

Proby best to deal w. even V.C. very briefly.

On learning Heurs: The Heuris must form a Concept net that has acceptable cj's for each lrnd Heur.

D: Windows
\ command
\ Ausi.

So: talk: Start w. 2 ideas: (1) Existence of Genl. Lrng Theory. (2) the TSQ problem.

(1) Discuss Genl. lrng Theory.

(2) Then discuss TNG Seqs in More detail

7.84
+.52

215: 984

-20 lo
-50 hy
~10-1
(+3)

.10 (1) Gen lrng Theories: (List on screen)

Sol
well
riss

V.C. ~~VCM~~ S.V.M
Recurrent ANN.
Soar.
Lenat AM. Eurisco, Cyc.

4.988
99.299
5.689

~~Maybe Mention Recurrent ANN~~: but analysis need. a lot of above technics.

CYC

Discuss Difference betw. Sol, Wal, riss: Emphasize how Sol ~~does~~ can deal w. very dift problems. E.g. Sociology, Economics, Geology. Also Machine ~~can~~ able to do things done by Subconscious Mind. — Discuss this later. Perhaps Mention Roger Penrose.

Maybe Mention Roger Penrose.

(2) The Training Sequence problem:

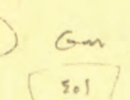
5:44 ±.30

Computer as "Idiot Savant." The smarter it is, the easier it is to teach.

.019

The computer is initially, ~~incredibly not~~ has an incredible lack of smartness: it's an "idiot savant".

Desc my early ideas on TSQ design: (Conc. (nets) ~~Trees~~ (almost a tree)) (how a Conc. was not a Comp lang. theory Conc.)



Lsearch: Gamb House thms: $\frac{cc}{pc}$ optimum Ordering: Time is $< \frac{cc}{pc}$.

5.79 ±.35

If all heurs are in P.D. then teach each w. time a factor of 2 of optimum.

So "TSQ's easy to write"!

Actually not so easy but I did write 3 in varying % of completeness:

AKG Notation lrng.
Linear eq. Solving & lrng rules of Algebra.
Solving linear, then quadratic, then cubic eqns.

3 problems w. these TSQ's (1) Hard to write (2) no use of sub solns, only solns.

(3) Didn't "scale": Many use it w. times so eventually it was unusable

Solns: to (2) larger probs (3) Use of "I.R." (used much in the ML work CBR, EBL ...)

4 Or: Other Modalities of lrng:

Some trouble w. extrapolating sets of examples: We want to find a short-coded machine ~~algm~~ machine that assigns by p.c. to ~~t.~~ examples. Or a ~~the~~ short-coded operator that takes ~~inputs~~ sets of examples into outputs. How do we find this operator? I guess by Lenat: ~~over~~ heuristics ~~i~~ components of heuristics. So we can have a training sequence of sets of examples or have ~~t.~~ machine update its ~~minimum~~ self for every new (k) k = 1, 2, . . . examples

perhaps we want ("Meta Hints"); that Enable TM to find ways to (modify/augment) short codes of ~~t.~~ machine that probabilistically maps from input to output of Examples.

The Interesting part as part of cond'l. p.d.'s → stoch operator, is that it makes ~~t.~~ problem clearer — makes it expressible as a well defined problem. It makes clearer how ~~t.~~ various parts of ~~t.~~ corpus are "allowed" to influence ~~t.~~ output

:eg

Anyway: Emphasize that ~~t.~~ Cond. prob'y Edge enables us to merge info, recpts, subrecpts from disparate sub-corpi — This is a characteristically "Human" characteristic.

Actually, their new operator form of ALP also enables us to do this, but, I think "cond'l. probys" are more commonly used: The single sequence/corpus in ~~t.~~ original ALP is certainly useful, but much human induction is on sets of finite objects.

So ~~t.~~ Q is: just how does TM devise these "short codes"? See (o1 -eg)

SN ~~T.~~ problem of devising a cond'l p.d. can be solved exactly by ~~t.~~ exact soln. of "the universal extrapolation problem" or, approximately (assuming no info in ~~t.~~ inputs) by finding a short-coded operator that maps I → O w. max p.c.

If we have a (large) common corpus, we can automatically include it w. each input to ~~t.~~ stoch operator.

Example; we have ~~t.~~ text of an encyclopedia: we want TM to answer Q's about the text: A training example consists of a triple:

(1) The ency text (2) A question about it (3) A possible answer to 2.

~~We give TM a large set of these examples~~

One way to train ~~t.~~ TM would be to give it a large set of such examples. It then tries to find a short code for an (machine e.g.) that can map from (1, 2) to 3 i.e. given 1, 2 to get 3. More exactly we want a stoch operator w. short code that assigns a high probability to ~~t.~~ known corpus.

~~Probably an easier~~ Another ~~to train~~ way ~~to solve t. same problem~~, Get TM able to do the same thing is to start with 1 or 2 examples; have TM solve them, then give a new example; have TM modify its old code to deal w. the new example; give it newer example etc. ——

Here if we careful design/many sets of examples and give TM a good set of ~~the~~ primitive operators, it will be able "track" the new examples.

We can make TM's work simpler by giving "Hints"; ~~The~~ "Hints"

clip 28, 25, 31 Kolmo...

.6587

.711

spec. → 62.01

9.5.98 : TM: on the probabilities of the integers !

One way to deal w. not this problem but a usually closely related problem:
We have a sequence of binary substrings & we want to mark between them. One way is to have a third symbol Δ.

Say the binary substrings are of max length k: & there are n of them in the corpus:
By using Δ as marker, we have k·n total of binary substrings, the n, Δ's: so
kn + n total no. of symbols, n of them are Δ so p.c. of next Δ is $\frac{n+1}{kn+n+3}$ ← symbol for "previous"

$\frac{n+1}{(k+1)n+3} \sim \frac{1}{k+1}$ for large n ($n \to \infty$).

This is O.K. if we have a large corpus already. If we don't, then
it's a bit strange ! i.e. $n \neq 1$; So we have 3 symbols: the binary strings of length k
the p.c. of one Δ on the end is $\frac{1}{k+1+3} = \frac{1}{k+4}$ •

T. Prop. model would be obtained by assuming uniform prior for all 3 symbols.
— using integration, we get "Laplace's rule"

spec .01:60.40. "Hints" are info that make it easier to solve a problem. In the present case, a "hint" would be information that effectively reduces the size of the space to be searched.

Another problem, that is formally similar to the Encyc QA problem is the problem of learning to prove Mathematical Theorems.

Here, the examples consist of [a triplett: the first part. the postulates and proved theorems available to be used in the proof. The second part, is the theorem to be proved.

The third part, is the proof.

We start out with as very simple, I step proofs. ["Proofs" would correspond to single steps in normal proofs.] and as TM becomes more skilled we give it progressively more difficult proofs. For each example, as in the encyclopedia, the machine tries to find a stock operator that has a short code, that gives max probability to the known set of examples.

1.27
1.86
.21
toke
soc.
today 2PM.

Here = $2^{-|D|} \cdot \prod_i P_i$ (for set of finite objects)

or $2^{-|D|} \prod_i P(O_i | I_i)$ ABCD

Present proofs first, then Encyc.

For Encyc. we start with a very minimal set of statements in the encyc, written in as very simple formal way. We gradually modify the way, and increase amount of info in the encyc, & make our Q's more complex. The important thing is knowing the state of the machine we have a measure of how much time it will take to solve each more problem. — So we can taylor our problem steps (the CTS) to be within acceptable bounds

Perhaps just Random → Solution of the optim problems using techniques similar to G.A. can also furnish TSQ's.

Closing remark: That Using Many Models of TM's of the same conditional complexity P. P. can give a broad scope of intelligence needed for really hard problems!

Another thing: That A.G.I. isn't restricted to kinds of functions people consciously use.

(If time, Discuss Roger Penrose)

I was worried about Not having an adequately good TSQ to start TM's "rise to the top" & My present thinking is that it doesn't matter Much; that one just teaches TM whatever one can — that ordering of topics is not critical! — we just "throw it all in" & TM is able to integrate it. (The "kitchen sink" approach) I'm still wary of "kitchen sink" however. TM may have to be fairly advanced before one can be so cavalier but when one needs it!

I ~~had been~~ Concernd w. my early (ε)TSQ's because I didn't see a ~~reasonable~~, easy, way to continue from! W. the present approach, We can continue in practically any direction!

T. ~~Things that~~ are being "lrnd" are Search heuristics, & they are valid for most any "Domain".

Well o.k. I see that (perhaps) TM could discover various such heuristics — (like (backward / forward) chaining, αβ, GPS, etc! But how does this fit into t. Universal Condl. P.D. ?

I may be Getting TM_1 & TM_2 mixd up!

.07 Def The Un. C.P.D. (UCPD) — Univl. Condl proby Distribn. can use itself as a setn. — Say to do a search going

Def from A to B, ~~[illegible]~~ & heur suggests subgoal "C", Then we go to the ~~C~~PD CPD to findout trials for going from A to C, Then, ~~[illegible]~~ from C to B..

10 TM_1's General ~~[illegible]~~ JOB type, is to find short codes for sets of objects: Usually this ~~[illegible]~~ results in a CPD — of t. form; for given input, t. output is a set of objects in rough order of PC, w. PC's Given. (Q: maybe $\frac{cc}{pc}$ order?). | Try, Printer ink "Pilot" pen!

ABCD

The problem of searching for short codes can be (recursively) expressed as a ~~[illegible]~~ CPD; i.e. given a certain corpus, what ~~[illegible]~~ is ~~[illegible]~~ < a good trial for a short code > ← output. ← (would we want this in ~ ~~least order~~?)

Given a (maybe Large) set of {(dern) Problem, (soln, trace)} pairs TM induces a CPD so that ~~[illegible]~~ for any (drn) problem, it will propose various [soln / trace]. (Not always in P.C. order; but P.C.'s will be assignd to each output Given)

In TSQ's for this proposd device: It can go from one Domain to another, in any order! If it isn't doing well in one domain, we have it work on another. — Later it can go back to formerly diflt. domains, that mite now be easier.

One impt domain, of course, is TM_2 mode: Generalizing, (finding better) heuristics. { Well, not so fast! In this CPD, ~~we~~ we just Generalize from past heurs: we don't rarely deriv new/Better heuers, Deriv of new heurs is a OZ problem.

.31 {Tho it mite deriv. new heurs by "diffusion" as G.A. search does ("Optimization by Diffusion") (Also Sim Anneal (does this?))

IN General, hvr., OZ probs are solvd by CPD; i.e. we put in t. problem: The outputs

.33 (w. assoc. probys) are various optzn. techniques.

Note that "finding short codes" is an OZ problem. — so TM uses [.21 – .33] on such probs.

Theorem proving is an INV prob. — Hvr., it is a "Problem", so it can use t. general C.P.D. to ~~[illegible]~~ order its Searches

Q, A wrt a corpus is pretty much an OZ problem — i.e. we want as good a code for t. corpus as possl. But. its soln. ~~can~~ will be part of ~~t~~ TM's General CPD.

9.6.98 TM Expo

O.n. : So Just how Does this TM work? It has a CPD to start w. some initial info. Initial info mite be a set of optzn. pgms, with an uncondl. proby on their use. ~~Tweqrmr~~ Actually, we may "sorbow in" a lot of info. — TM then just uses this info along w. new examples & other new info, to make a Better CPD.

← we may want to "factor" the optzn pgms a/o express them as a (uncondl.) stoch. grammar

It mite be possl. for this system to use CYC's data .... The whether this would be of any value is unclear: I know very little about CYC & what kinds of info it has & what form its it.

.10 Describe TM's "steady state", then derb an initial state (Tabula Rasa ☺?) and some initial startup training that will put it into a useful "steady state".

One Q, is t. form of t. CPD: ideally its a stoch generator; but it is created/modified by considerations of codelength of its dcrn. [Its input is a problem dcrn, its output

.14 will be trial solns w. assc. pc's: For INV probs, presumably its output is t. soln. only (?)

Maybe not: We could just ask for a SPD of solns (a priori) with an implied Lsrch to find solns. Hvr. in good Lsrch, t. pc's change as one works on t. problem.

Dropping for a while its response to "INV" probs.

(SW) There is some ambiguity in t. defn. of "CPD". In general, t. output will depend on but allowd CB. (RLP) Do we have to either specify a Time limit, or accept an "Anytime" soln ? Well, in t case of INV problems, no CB is needed: it just works untill it solves t. problem (if ever!)

.22 Start again on "Steady State" dcrn: At all times it has some Dcrn of a CPD which it uses for all inputs. Its "input" is a prob. dcrn. Its output will depend on its input dcrn (Naturally, since its a CPD! ☺). We will have to decide just what kinds of output we want from TM, for each input type: for starters:

.24 1) Input is a set of finite objects: we'd like one or more of t. following: Modifn of TM's CPD dcrn. so that, given a partial object "in that set", it can give a PD of possl. completions or partial completions — e.g. "what comes next in t. partial object?".

.33 2) Given t. input of 1) (.24): Modify of its internal CPD dcrn to minimize t. dcrn of t. entire corpus.

3) Input is a partial object: Output is a pd on possl. continuns. of that object.

Well, we could just have 2 modes of t. input: ① t. input is an augmentation of t. corpus: we want TM to modify t. CPD dcrn. to accomodate it. (Minz augmented corpus dcrn).

② we put in partial augmentn. of corpus (or of part of corpus): output is p.d. on possl. continuns.

This is a combn. of .29 & .33 : we can create either by mixing ① a/o ②.

9698 TM expo:

Another way to look at it: We input same set of objects & that augment the corpus (to a "sub corpus") plus partial object we want continuation of. We want modification of CPD down to a pd on the continuations of partial object.

If the partial object is null, we'd get a pd on the combined objects in the CPD corpus — but we can request not to do this step.

→ Note that to make an optimum "continuation" p.d. one must recede the entire CPD down.

.06 If we give it an OZ problem (w.or.w.o. CB constraint), its output is a p.d. over a set of optimum methods. (The nature of the problem suggests which methods are best... & one use

.08 this p.d. as a basis of search for OZ prob. solns.

Criticism of .06 - .08: As one runs the problem on optimum pgms, we may get into text suggest modifying the probs. [This may be helped by my soln. to the "what to work on Next" problem.]

It> "learning" of which optimum methods are best for what problems seems flawed. It will spend most time in its "favorite choices" — so that choices is, indeed, must likely to find the "best" soln: We will have to compensate for this bias by giving more reward (w.r.) for solns. by "non-favorite" optimum methods

Well, we can give TM optimum probs, & have many different optimum techniques work on the same probs: from the results, TM will get a kind of empirical idea of what optimum methods are best for what problems.

.19 This is what "well-conv": Ideally, TM should be able to analyse a problem, & by virtue of its "structure", & the "structures" of various optimum techniques, decides which technique is most for that problem. However, I'll have to "look into" just how this could be done by TM.

Maybe by watching the "traces" of a person (or properly trained machine) solving problems of this sort.

.23

I think what I want to do is list various kinds of problems: Then for each, decide what I'd like TM's response to be.

But the main idea (I think) is that we have a technique for always getting a CPD for a variety of problems & we have a way for it to learn to improve its performance.

All I really need is a very good idea of its "steady state" performance. 64.10 & .22 are some ideas.

I think the hardest part of TM's work is: Given a corpus w. a short code for it: Given an augmentation of the corpus: To modify the short code so it is "short" for the augmented corpus.

An imp. case is when the augmentation is small (e.g. one full object that is a possl. completion of a given partial object). We may want to devise special techniques for this case.

Another imp. kind of prob. is OZ probs & improving TM's skill in this area: .06 - .28 discusses this a bit: .19 - .23 is perhaps imp.

9.2.98 TM · Expo!

.01 Dem. of Ray's system for t. talk! (1) The CPD! Steady states It has a short code for a current (thus far) corpus. We augment this corpus w. some examples, plus a "partial object"! output TM then looks for modifns of its code in view of t. new augmentation of t. corpus & possible continuations of it.

Its output is a pd. of t. continuation of t. "partial object".

for talk, maybe don't mention "partial object" at first; t. CPD is for finite objects in view of t. "corpus up to now".

~~It uses this facility to improve itself in five ways:~~

.10 (2) At any particular time it has a large set of optzn prms in memory. It looks at every OZ prm & it is able to form a pd on these optzn methods — which would be the basis for that problem. A common known OZ problem is finding short codes for a corpus.

.14 (3) It is able to work on the problem of devising better optzn prms.

.15 This is a special case of its normal skill at solving OZ problems.

This is very ~ to Sol 89: but perhaps main difference is emphasis: Now, I'm "mixing" domains a lot. T. probs tend to be (at first) simple induction problems based on a set of examples.

I may be less afraid to "solder in" various impt. ideas — (inductors) line/optzn routines (tho eventually, I want to factor them & make a grammar for them),

I may be less afraid to (any ind. - Tng of TM) give it problems that I haven't myself found a soln for ("soln" in t. sense that TM could find that soln. in acceptable CJS.).

Perhaps a major difference: I'll be giving TM almost exclusively OZ problems. It will not use Lsrch much, except in a trivial way to select appropriate OZ methods.

So t. emphasis may be on starting w. v.g. optzn techniques — & not until much later try to get TM to improve them (.9/.14) & to some extent, (.10)

A possibly Major Difficulty in t. present approach! That "finding short codes" is not an ordinary OZ problem: As t. corpus expands, it enters new "Domains", the kinds of techniques used to find short codes will change; so (.14) (improving t. optzn methods)

.37 may be an essential part of t. steady state system. That w. a fixed set of optzn methods, I will only be able to do a limited

Does paper cause Queasy stomach? Or certain other foods?

Setup "domains" or "regions of Domains" — That TM will have an effective "epistemological cut-off" : for this reason, I may want to return to the old [Sol 86, 89 system] — using Creatively designed Concept nets.

Could I reach a kind of Compromise between the latest system & Sol 86,89? — Having some of the best features of Both?

.05 First look at how Bad the duty of [66.37 - 67.02] is. It may be possible to solve it, using a suitably Conc. net, Tag-seq. ext. — That / Regularities" forms a corpus

.10 Conc. net, à TM simply has to learn them in that form. Hmm, even so, if TM is

.11 "Sufficiently Advanced", I can design "TSQIKs" "intuitively" (as for a human student)

Ab Initio

.12 → So: One Q is: Can I get TM to be "suffly advanced" using Various

.13 "optim techniques" I know of? → Note 89.29 — It may well be that the finding of Regs in a corpus & needs a special optim. technique: That this "concave net" consideration of .05 - .11 is an accurate assessment of the problem.

The basic idea of .05 - .11 is that the regularities observable in a corpus, are

.20 based on concepts that have to be found as a conc. net is found, & that the same constraint on CJS applies & tells us which concs can follow which & how much

.22 cc we need to discover each new conc.

Looks like the discn. of .05 - .22 may be an adequate understanding/resolution of this most crypt. problem.

So 66.01 - 67.22 pretty well defines the new System & its critical problem(s).

It suggests a very Optimistic Evaln. of the system [i.e. suggests that it's likely to Actually Work!]

An impl. Mgt. system might do is to be able to look at a corpus & guess what kinds of Regs are likely to be init. On a gross level, this is a "simple" IR problem, having to do w. ways of indexing corpi. what "features" suggest "what types of Regs"? This "preanalysis Evaln." can be as deep as it can incorporate (& do part of) the final search that short circuits. Ideally, these 2 aspects of coding should be integrated.

9.7.98 TM Expo

59.07 – .40 is an outline of most of the talk! 66.01 – .15 is an outline of the end — this is a condensation of 59.07 – .40.

The talk itself: View Graphs: ①

Concept Net:
acyclic Directed Graph.
Partial ordering.

① General Long Recovery! 59.09

② Concept nets (picture)

$\frac{cc_j}{pc_j}$ optimum ordering!

$\frac{cc_j}{pc_j}$ ~ time for soln: ~ upper Bnd. (not average)

~~upper bnd on srch time.~~

~~it is within factor of 2 for any way, its ... the $\frac{cc_j}{pc_j}$ for ...~~

If pc_j's are normalized, then the certain Model Lsrch srch time will be very close to $\frac{cc_j}{pc_j}$

It was possible to express ~~search~~ Search heurs as modification of pd: So if you put all of the heurs you're going to use into the pd., Lsrch will do at least as well as heuristic srch using those heurs.

Between

10:05

Sort of flattened

SN: Uniform priors, Laplace's rule — if we want "Expected values" of proby: Also for Decision making! perhaps these "Expected values" are just what we need! ~~...~~ Critical objects are Proby * Utility. (or maybe expected (c')) are too rite!

③ Demo of latest ideas on ■ CoPD machine. See 66.01 – 67.40
But mainly 66.01 → .15

Also draw picture of examples of Q, A, pairs of ~~...~~ (theorems, available + possibly known), traces of solns, pairs. Note that it works only for "1 step proofs".

Then discuss diffys & how they will be dealt with.

Re ② Perhaps discuss ~~...~~ idea of building up more complex sequences on "simpler goals"; then draw Conc. net. idea of from cong.

Then idea of Lsrch & $\frac{cc_j}{pc_j}$. (Maybe discuss Time Share).

Discuss: This is simple demo: In addition, systems able to ~~...~~ in srch heuristics.

And $\frac{cc_j}{pc_j}$ is ~ time to soln or upper Bnd.
is within constant factor of Best possible way.

~~...~~

Idea of "Constant factor": Its often ~ 2 or 4 depending on how one does Lsrch.

Idea of heuristics in srch: How they → modification of pd. If all heurs are in P.D. then L srch will give at least as good (& usually better) than use of those heurs in "heuristic" srch. for soln.

-.7.98 10AM rcrd. (late by nights)

9.7.98 TM : VAPNIK, Vladamir (Volodya?).

Discn: He is not a Baysian: Tho he does use apriori info., he uses it in a difrnt way — to select a set of functions.

See (relatively recent notes on V.C. dimension & S.V. machines: Related to discussion with Poggio (q.v.)

An impt. param. is his "ε" (the zero penalty zone size): It is related to "Margin" — which, (in t. case of hyper surfaces separating classes of points), is the distance of t. h-surf. from t. nearest pts in t. classes being separated. (But what if t. h-surf cuts thru both classes?

So he has a 2 part Gen'raln. (1) t. V.C. dim (or equivt) That tells t. "complexity" of t. ~~hypothesis~~ concept class ~~[illegible]~~ & t. other is t. [redacted] observed error size.

He is impressed by Popper's falsifyability: His term "Shatters" is t. same as "Falsification".

He says that if t. ~~search~~ concept class is all classes, then there is no example that can falsify any concept class. So "degree of falsifyability" is closely related to V.C. dimension.

Lucient Teach nalg.
Visit:
Gmm. ~2pm Tues.

He says (& Poggio says) t. ~~SVM~~ S.V.M. technique is much faster & better than Neural nets, & may be beginning to replace them.

He has pgm. available on net. for this stuff. He does not need enormous machine to get usable results.

9.8.98 ~1(30 3
9.9.98 2:30A 2.5
5A
7:30A 2.5

He says that if the ~~true~~ true prediction function is not in the set considered, then Bayes assigns zero to that hypothesis. I think this is irrelevant: we never have a info to know t. correct hypoth. Instead, if t. SSZ is small a/o CB is small, we automatically get large Granularity (≈ Vapnik's "ε").

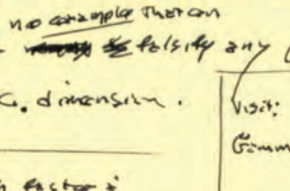
V. mentioned Barron's paper on Error Convergence. V. may not be aware of a Sol 78. He felt that Barron's result was somehow not Baysian. (?). [It is possl. that Cover is not a Baysian!]

He uses ~~[illegible]~~ Hyperplanes for separation of classes but t. coords can be n.l. functs.

like $x_1 \cdot x_2$; $x_1^2 x_2^3$; 1 ; $x_1^2 x_{17}^7$, ect.

.28 What does he do when "Margin" is < 0 i.e. classes inter penetrate!

T. "true" model! ← (But we have random noise in observation which gives)



.30 No. of Dims = No. of Data pts (≈ SSZ) [needed for separation] of his H. surfaces (→ see 69.2.14)

Perhaps his Apriori info is always of t. form: Certain functs are impossl. T. rest (which we know what they are) are of uniform a priori. — so he has to define a density function anyway! So he might as well do it over t. entire set.

V. says he gets better results than ANN & much faster: Well, ANN is normally done wrong anyway using Max Likelihood (Shen.) should improve it. But V. says he usually needs more support vectors than are usually used in ANN!

V. says that attempts to relate his stuff to "Lcost" have been Mixd! Some parts correspond in a simple way. Others do not correspond at all and can't be made to correspond ... they get essentially difrnt. answers.

9.8.98 VARNIK.

5 [illegible] Gamm want my c.v.:

2 sources:
1) That proposal I wrote for Rur become Sol 89
2) Stuff for Oxbridge Brochure.
3) more recent update of Bibliography

Maybe recent Lectures.

C.V.:

(1909)

$$\int \underbrace{K(u,v)} \varphi(u)\varphi(v)\, du\, dv \geq 0$$

← $K(u,v)$ said to be "Pos. Def."

$$K(X \cdot Y) = (X \cdot Y)^5$$

~~Display "5" because~~ why "5"

$$\int (X \cdot Y)^2 \varphi(X)\varphi(Y)\, dX\, dY > 0$$

$$e^{-\frac{(X-Y)^2}{\sigma}}$$

$$X^t \underline{A} X > 0$$

$$X = x_1, x_2 \quad \overset{z_1}{X_1} \quad \overset{z_2}{X_2} \quad \overset{z_3}{\sqrt{2}\, X_1 X_2} \quad \overset{z_4}{X_1^2} \quad \overset{z_5}{X_2^2}$$

$$Y = y_1\, y_2 \quad y_1 \quad y_2 \quad \sqrt{2}\, y_1 y_2 \quad y_1^2 \quad y_2^2$$

$$(X \cdot Y)$$

$\times\ K(\underline{X}, \underline{Y})$

$$\sum_i \alpha_i (X_i \cdot Z)$$

$$X = (x_1, x_2) \longrightarrow x_1,\ x_2,\ \sqrt{2}\, x_1 x_2,\ x_1^2,\ x_2^2$$

$$Y\ (y_1, y_2) \longrightarrow y_1,\ y_2,\ \sqrt{2}\, y_1 y_2\ \ y_1^2,\ y_2^2$$

$$\left(1 + (X \cdot Y)\right)^2 = x_1 y_1 + y_2 y_2 + 2 x_1 y_1 x_2 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2$$

http://svm.dcs.rhbnc.ac.uk.

* Address of Site w. SVM pgm.

9.9.98 Vaphik — see 73.07 for More on Vaphik

Vapnik suggested [redacted] that Pogio mite be able to get me some grad student at MIT.

in His SVM pgm, one is able to derive general way, t. separation ~~curves~~ h surfaces or the regressn function: (along w. an "ε" Value) & it will relatively quickly, find an optimum fit. — Among t. functions it can use for regressn: linear (& perhaps arby degree); splines of various types.

His SVM pgm seemed to run at o.k. speed on a 386. Main slowdown was graphic/output written maybe in Java (an interpretive Lang.).

He says that in ordinary ANN pgms, t. no. of Nodes they use is much less than the no. of "support vectors" he uses.

He says that t "support vectors" are always a subset of t. examples.

I had been mainly thinking of use of curves, splines ... to separate classes of pts (classifn. problem): But he also uses it for Regressn (curve fitting)!

1/4 So given a bunch of data pts: fit a curve using some error function w. any error < ε costing zero. We get a curve, then we find that only a certain subset of the data are "support vectors" & that ~~the~~ fitting them alone, gives same curve as entire data set.

Th. no. of support vectors ~~→~~ gives something like t. complexity of t. hypoth(?). (See 69.30)

The ε used corresponds to t "Margin" in 69.28

Re: Support vectors for curve fitting. There are various ~~tinarring~~ params describing Goodness of fit (≡ Gares): ① One is for RMS or some other norm; ~~②~~ is one w. V's "ε" of zero cost ③ no. of coefts or other measure of info complexity (≡ rcost) of t. model (maybe V.C. dimension).

④ By w cutoff of ∑ xtns usd to model t. function, or idea of A. Barron on fitting using ~~...~~ $\sum a_i f(b_i x)$ for model or $\sum a_i f(\vec{b}_i \cdot \vec{x})$, if $\vec{x}$ is a vector.

These various params give different properties of t. models usd for approxn. They can tell something, perhaps, about t. "True" model of t. data.

9.8.98 TM Expo

.01: 68.40 [redacted] View Graphs:

approximate Defn. of Lang;

① GENERAL LEARNING THEORIES. [How to ~~make data from Past~~ use data from Past for Prediction]

'03 Algorithmic Probability: Solomonoff — ALP ~~not~~ never used for Predn.

~~Wallace MML~~ MML Wallace

~~Rissa.~~ MDL Rissanen.

~~E.~~ V.C. Dimension, ~~Svpp~~ S.V.M. Vapnik, Chervonenkis.

~~Reinforcement~~

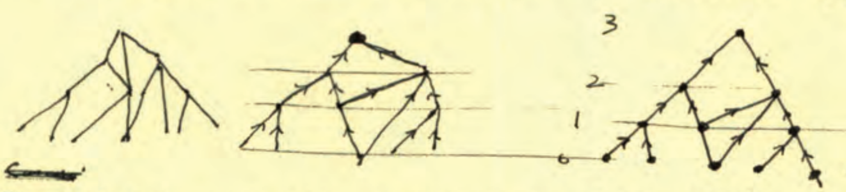
Genetic Algorithms · Holland.

Recurrent Neural Nets ....

SOAR ~~Rosenbloom~~, Newell, Rosenbloom.

......AM, Eurisco, Cyc : Lenat.

② TRAINING SEQUENCES and L SEARCH (Levin's Search procedure).



~~Concept~~
Concept net : acyclic Directed graph: Partial ordering.

Concepts: Solns of problems are ~~concepts~~

Macros that can be combined to give ~~other macros or other macros~~ or

~~Other macros / Solutions to~~ 1. Solutions to problems
or 2. other macros.

yield
~~yeild~~

~~[boxed]~~ How Systems Solves ~~[struck]~~ problems by search.

" " assigns probys to concepts.

" " uses probys for srch:

$L\,cost = \frac{CC_i}{P_i}$: Lcost of trials used to order trials

$\frac{CC_i}{P_i}$

~~[struck]~~ subtcost of search,

L cost of th soln. trial is goodestimate to total search time needed & find that soln.

This secret quality makes it ~~e~~ much easier to design Trg. sqns. — we can always estimate how long it take TM to solve a problem.

* Learning Algebraic Notation $3+7=10$; ; $((a+1)/\_)+1 = 2$ . . .

Learning to solve linear eqns and learning "Laws of Algebra" (associativity ~~[struck]~~, commutativity ~~[struck]~~ ~~[struck]~~ distributivity laws)

Learning to solve Learning Run Quadratic, Run Cubic.

~~Problems: ① TRAING Sequences~~

9.8.98 TM Expo

(3) Problems with Traing Sequence design:

1) ~~that~~ very difficult to write them.

2) ~~3~~ Didn't use subcincepts ~~of~~ of previous ~~prob~~ problems

3) Use of ~~the~~ common Memory to store ~~solutions of problem~~ useful concepts:

Too much in memory, probly acrec. w. retrieval was ∴ low. Lcost of sols $\left(\frac{cc}{P_i}\right)$ was too big.

4) The snippets of tng. seqs I wrote didn't fit together to form an integrated body of tng.

2) Due to ~~small~~ only easy problems solvd. [Hard problems involve certain techniques that ~~to~~ can make their ~~soln~~ quite soln technique entirely differnt from soln to small problems

3) Use of indexing for retrieval.

Books in library: No index; just search thru all books: If ~~index~~ if ~~we~~ we knew something ~~about~~ about a book and the collection ~~too~~ is indexed so we can ~~use~~ use that info for search. retrieval can be very rapid.

— [Cost of ~~a~~ a "tool"] ~~to~~ teeth, Hammers, electricly driven hammer . . .

So similar in learning concs. Index them when learnd. (In ~~many~~ many ~~possible~~ ways possible.)

---

(4) ~~Latest~~ Latest system: 66.0–67.40

1) ~~Inductive Inf. Machine~~ Learning Devices.

Devices: (a) Input is a corpus: output is ~~an~~ ~~most likely continuation of it~~. ~~corpus is Cont. Pr. pos~~ one or more most likely continuations of that corpus, with their probys.

Why this is a very useful device: Use for sequential extrapol.

SM, Horse races; other competitive sports. ~~Stocks~~ Encyclopedia! Corpus is ~~some~~ some encyclopedia: Plus many Q's ~~about the encyclopedia~~ their answers as obtaind from the Ency. Plus one new Q: — the output would be most likely answers.

In ~~math~~ proving theorems: Given a theorem and the postulates and thms usable in its proof! — in time T: ~~what~~ what are ~~the~~ ~~most probable~~ ~~proofs?~~ some good ways to try to prove the theorem?

~~(b) How it works.~~

Given postulates, theories of physics: ~~Both~~ Derive

(b) How does it work? Each of the General Lrng Systems is able to deal with some of these induction problems — Not necessarily all. ~~The~~ All of the ~~many~~ the learning methods I've mentioned have certain constraints on the kinds of models they may consider.

~~For very difficult problems in Sociology, Economics, Geology, whatever~~ ~~shows can be long. and~~

~~My~~ One reason for my Bias toward RLP: It ~~might~~ ~~models of low~~ ~~at~~ ~~for~~ omits ~~the~~ the consideration of models that have low a priori likelyhood that take too long to compute. We try to consider models in $\frac{cc}{P}$ order.

cc is computation cost; P is a priori probability

Subdevices: ~~inputs~~ ~~corp~~ ~~partial~~ input is corpus, code for corpus! + augmentation of corpus + cB output is ~~augmented~~ ~~a~~ short code for augmented corpus.

control / codes

If TM has several ~~Probys~~ ~~of~~ concepts, many things based on α. — Then α is found to be false! How does TM update? If α is a theorem in Math: TM has to do laboriously, what a Mathemn would have to do: Backtrack.

Otherwise, often times of heuristic concepts are never "wrong" — just "u give low pc & more or less" coding via condl ~~Proby~~ condl Probys.

→ Actually all it needs to do is give short codes for a corpus: which may include partial objects.

35 mins

~~The~~ method of solution using RLP:

.02 We have an algorithm that operates in the following way:

We insert a corpus of data: ~~sequences~~ sequence of numbers or vectors,

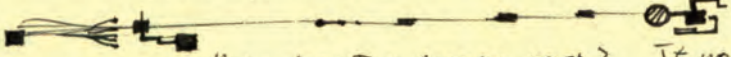
or ~~symbol string~~ a set of standard strings — each that represents an object.

The output of the algorithm is a (short) sequenced codes for the corpus. If we give the

machine more time, it usually finds shorter codes.

These short codes can then be randomly combined to give roughly correct codes for

the corpus and its probabilistic continuations.



How does this device work? It works pretty much like ~~recognition~~ the training sequences

I discussed before. We start with very simple induction problems. The machine uses

an L search over a set of ~~[illegible]~~ possible codes. (When asked for output, it presents the shortest

codes it found thus far. The longer we wait to ask for output, the ~~better the~~ shorter

the codes can be.

After it solves this problem, we ~~[illegible]~~ extend the length of the corpus and

ask for a short code for it. The machine starts with codes for ~~[illegible]~~ the early parts

of the corpus and tries to modify them, to code generate codes for

the ~~[illegible]~~ augmented corpus. We can continue this learning by giving

the machine an augmentation to its corpus to code, or an arbitrary new

augmentation

Corpus.

The ~~[illegible]~~ coding problems I've described will only be solvable in reasonable time, if

the regularities in the sequence of problems form an acceptable training

sequence — involving concepts that build on previously learned concepts.

In this sense, this present problem has all of the difficulties of ~~[illegible]~~ constructing

~~Training~~ TSQS in our previously described project.

There are two impl. differences.

1) Training examples can come from a variety of sources, to help learn various

kinds of regularity in data. The system affords a kind of ~~[illegible]~~ learning

device that is able to integrate learning from various domains

2) An ~~[illegible]~~, important problem in learning systems is ~~[illegible]~~ often necessary

to recognize when a situation is similar to ~~[illegible]~~ situations that have occurred in the past.

We can use inductive machine learning to help solve this problem.

[Perhaps Memory CBR and EBL] ~~Helps~~ solve the ~~[illegible]~~ problem of

using Memory effectively.

Perhaps Do

Discuss implic.

of measures)

Try

9949 12:15A
C15A ~TM Vapnik — (.07) ~23

Augmentation of talk:

1) Go into differences Betw. ALP, RLP MML, MDL. — who throws out what, also varying p[illegible].

Idea of Incomputability; unknown error size;

"killing t. ~~messenger~~" harbinger of bad news".

2) Discuss [illegible] "Reading and Understanding": Re "Textworld" project.

.07: 69.[illegible]–69.2.90

3) Re: Vapnik: [illegible] is simply a ~~n~~ class of loss functions. If its a good class, it should be useful for compact coding.

4) The "support vectors" are always ~~n~~ t. data set & t. loss function (for that subset) (function having min.) will be identical to that for t. ~~[illegible]~~ entire set. Usually t. "support vectors" will be at the edges of the classes to be separated.

I mentioned case where classes ever lapped in unaccustomed way (that t. "true classes" were separable, but t. true classes + noise overlapped): He said that in such cases we ~~[illegible]~~ do a ~~[illegible]~~ "best" fit & we have a certain no. of "error pts" & we are penalized for them.

5) Vapnik: For separating 2 sets of 2 dim. pts [x_i, y_i] I will use $f(x_i, y_i) > 1$ & $f(x_i, y_i) < -1$

f will be, say a 5th ° poly. He may find some pts that can't be categorized so;

So he has a no. of "error pts".

6) Perhaps just go thru t. talk once, & if there is time, go back & discuss certain pts. in more detail.

9998 ½TM
9998: 11PM 4 Amp Card. 1145P
91098 430A 842P 9pm 4amp Card.
1045P 1¾
91198 1230A 1¾
2??A 2¼
630A 3¾
845A 2¼
945A Same Approaches
5p 4 amp card

1) These Trig identitys I'd covered may be ... of expansions of known simple functions! Maybe not: it was a finite sequence. dd14.

2) T. problem of "Change of Representation"! Essentially dd14 problems. Clues are correspondences & analogies w. other "~" problems.

So Main problem, is writing TSQ's for finding of reggs / short codes.

Same Approaches: ① Make some kind of Algebra, to show Reggs as concepts; so concepts can be combined to form "Reggs." This is more like t. Reggs as t. language, & f.

Concepts are terminals & non-terminals.

So study various kinds of Reggs: See how they are related.

① $K(x, |x|)$: can be used to define ALP. But still, it may be necy to define "Minimal codes" ⓒ

② Re: Alan H. (utchinson?) Write him about defys of Z141: Also Maybe ... to Book by Van Heerden.
Explain ⓐ That One of formulas has typo ⓑ After correction, formulas are OK. for only first round of decodes. T. resultant code is no longer a Bern seq. So t. formula (method) can't be applied again —— It may be possl. to Modify technique so it works, I have done this, but not via Binary defns.
How best to use "$f_A \cdot f_B \Rightarrow f_{AB}$" / infer is unclear. Clearly there is some regy.

③ Write Rissanen about how he is wrong about $\binom{R}{A}LP$: about incompatibility & rate of d of error w. cc. Copy Sol 86 Section & relevant it. Also copy Section from Barc. Paper

④ Wallace's statement that error ~ $\frac{1}{n}$ is incorrect; $\propto n^{-k} \propto n^{-\frac{1}{k}}$ | $k n \rightarrow m i$ | $1.6 \rightarrow .62$
a) Numerator is length of first part' code ... b) I think it should be $\frac{1}{n^k}$ or $\frac{1}{n^{\frac{1}{k}}}$: see Sol 78

⑤ Maybe write both Riss / Wallace about Differences btwn. ALP, PLA, Riss, Wall.

⑥ Wallace: In physics, usually considering models other than t. shortest, will not make appreciable difference. t. "Best" Model will be far ahead of t. others. In most other sciences there ... can be significant differences [Wallace's "strict MML": His "Granularization" of t. space of Models, may related ~ to Vapnik's "ε approxn" & to V.C. dimension or "No. of Support Vectors"]

911 98 11PM
91298 2A 915A 3¼

⑦ Scientists lie when they talk about Sci method" & "How they invented things"
ⓐ Newton. dis... "no action at a distance" — but law of Gravity,
ⓑ Keplar's explann of his reasoning.
The thing is about Sci method is "Non falsifyable"

⑧ On ... Paradigms: linear, then renaissance of a small class of non-linear, then for nonlinear yet behavior, extension of N.I. to P. recursive. — for going past the cutting edge of scientific discovery. for t. [redacted] curious for the Bold (Not for the timid)

⑨ "Beyond the cutting Edge": A new vista in (Science, Prediction, The Predictive Sciences)

So a major problem is how to devise tsq's for learning how to discover short codes (= "regularitys).
On 66.32 — 67.40 is an earlier (optimized) model of the system: Finding "short codes" is regarded as a kind of oz problem & we have a great Variety of often pzms "on tap" in Memory. Also, we have "factored" this set of opz pzms into a language w. suitable (terminals / non-terminals).
I got the idea that finding short codes was likely to be a very special kind of oz problem — perhaps needing a very special optz technique.

One approach: watch me do regy detection: Give various Methods & factor them into a language (Grammar)

Glickstein Stine — Eye Brain Mind, a Neurophysiological Approach.

My personal approach to short codes: First: look at the problem: This will suggest which types of regy to look for.

University College London.

Martin Löf wrote about "Randomness tests". Perhaps one could "Hill Climb" on results of Randomness tests ....

Some "Randomness tests": for a sequence of reals: Try prediction as a function of previous corpus. Do various SM predn. schemes. Try "dimensionality" approaches in Chaos theory.
The BSD (or whatever) test for chaos — is also a test for regys of any kind.
Conventional fitting Neural nets to data. detects a great variety of patterns. Doing it nicely could be much better.

V.C. Dimension & Support Vector approach can detect regys. (see 69.01 ff on Vapnik etc.

(SN) "Partially Computed" (partial / partly) functions are b exactly what we do. Its much better than "Partially Computable" or Approxly p.r. or Practically p.r.: It "implies" or "allows" the idea that one can explicitly, optionally limit part of t compn. with "say, a "CB", in a most general sense.

(TMA) (SN 2) Impt! The recent idea of TSQ's for only learning to find short codes, gives an "adequate" TM, but it seems very hard to do. An earlier idea leading to that one, was the idea of a Condl prob'y learning machine but it learns things of any sort & put them in its C.P.D. (not just ideas on how to find short codes)

(TMB)

→ wasted 72.02-.40 ← Very likely (verbatim avoidability). Also 66.01—67.40

66.0 - 65.40

Now, it probably that these 2 ideas are equivl, that any kind of lrng. involves finding a short code. If so, then the methods of TMB could be a good (more easily designd) source of TSQ's than doing TMA directly.
We could simply Do TM_B & have TM_A be one of the "conditions" for TM_B's CPD.

This TM_B idea does seem nicer, & easier, more fun, to write: More immediate joyous Feedback than TM_A. But TM_B is a type of TM_A: Perhaps I somehow didn't realize that any induction problem could be made a part of the tsq of TM_A.

In fact, TM_B, (maybe?) must first decide what kind of (induction) problem it is being asked to solve. At first, we will include, with each problem, a label, telling what kind of problem it is. TM will learn to pool various problems types & separate others into sub-classes.
So later, the labels will not be critical, tho they can furnish TM impt. auxilliary info!

.01 There seems to be a serious Difference between TM_A & TM_B: e.g. TM_B can be given a set of I,O pairs: $[I_j, O_j]$ & given a new I_i, it gives a p.d. over O_i. It does this (usually) by constructing

.02 a (set of) short code(s) for [I_j, O_j] — or, more exactly, a short operator for $I_j \to O_j$ (A BIG Q here is Just How its Solves O.S.L. (One Shot Lrng) problem. As such, it is built up from a TSQ of less complex operators, that enable it to learn impf. "concepts".

.04 A "Gold Prob Machine" is a stoch operator, (like for soln. of .02 – .04.

A simple example of a sample of .02 – .04 is the "Alg. Induction Learning" TSQ. — & its soln.

I should write up in much more detail, just what TM_B does: examples of its operation & how it is able to implement TM_A. [similarly, I may want to detail out, how TM_A implements TM_B.].

.14 TM_B is a way to solve TM_A's problems: It amounts to a "change of Representation" or "reformulation" of a problem. How did I discover this? — In general, keep track of "reformulations" that I do, in attempts to see how I do them. A major sub goal for TM would be its ability to do such "reformulations". → NN1.10 (#12.)

The main (perhaps only) way that I had TM_B solve problems was by

→ Lsrch using a suitable (conditional) P.D. Does this amount to simply trying to

→ get short codes by Lsrch? (using suitable CPD).

91498 In line with this last, searching for short codes for a corpus: first one "identifies" where the corpus comes from so one can get Conditional Probs. Then one does a Lsrch on suitable "primitives" using the cond'l. probs assoc. w. that "kind" of corpus. The "primitives" used, are (conditional/dependent) on the "nature" of the corpus.

So one has this big set of possl. cncs. The ("name/index" / "nature") of the corpus, then induces a pd on those cncs. w.r.t. the problem of finding a short code for that corpus.

91298 (NN)

.01 (TM 74.49) M. Glickstein cerebellum, eye

.02 "Change of representation"

(10) to Alon : That Amerala's change of method of expressing problem

Eye Brain Mind a Neuroplys approach.

was a reformulation "looking at prob in new way" : A new method of coding.

(11) Memories of Nobel Prize winners!

a) Fermi: Moved toward door at end of lecture

a2 " : Fermi v.s. Yous as lecturers.

a3 Fermi's reply to Eappenal should R w. temperature

J. Nash's rebuttal.

91298 515 A (cont) 8:45A 3½ — previous cardura 5pm!

130–230PM Dizzy! 3PM cardura tmg

b) Newell (≤ Simon) ! From conv. with Newell : That Fredkin's got better/more ad hoc into machine! I that Simon must be insane! He knew about History, Philos. of Sci.

91398 1295A 615 5½ 1015 4 (.10) (12) (see TM 75.25 | TM 76.14) BALD-FACED

TM_B is a way to solve bald-faced problem of TMA. — So it's significant. "Change of Representation ∈ sec. (.02) is should be studied

91498 1201A 12:15A 2mg card 430A 4½ 530A 2mg card .17 830A 4H5 1247P 4½h. 5P

as such. Just what happened? How did this "reformulation" work? How did I think of it?

(13) Try to keep record of "inductive leaps" ↑ & make i see if I can get TM to make them w. acceptable CJS.

(14) SMFI! TRY present pgm. w. updating, but change time scale from 1 day to 22 work days (1 Mnth). The idea is that we don't have to work so much every day. Even if t. yield is much less, it may be worth it. — Say when we are on Vacation.

11:45P 2mg card 1598 1201A 445A

Also, we can have a much larger set of possl. driven (use all fid's), w.o. much trouble.

9A 3mg card

4 91698 2A (15) Soln. of "How Building Blocks Hypoth" works; If A & B are useful concs.

1230P 2mg card 1230P 430P 4 1130P: 2mg card 1130P:

then they have short codes: Hunting for uses of "A·B" would be finding a justifn of t. new defn, A C ≡ A·B — which would give a shortening of code.

This is a justifn of t. Z141 idea. (which was never clear to me)

So looking for unusually many occurrences of AB is somewhat like

91798 27 415A 4½.

L-srch! Theorem is t. Q of ! If ($f_{AB} \ll f_A \cdot f_B$) then clearly a way has been found, but how best to use it for induction?

28 815A 12mg card

see 2.20–2.40 for Avg. & Z141 → (3.04) (#22) → see 2.34–.40 $f_{AB} \ll f_A \cdot f_B$

(30) ——— (16) Simple proof of first Gamb. House Thrm: If trials for A & B are not in $\frac{f_A}{cc_A}, \frac{f_B}{cc_B}$ order, then exchanging their order will ↓ expected cc of srch. We need only prove this for A & B being adjacent in ordering & misordered.

(17) Awareness of "Sudaky Peaches" (Nectarines)

Because if the whole set is properly ordered, $LC_A < LC_B$ for all adjacent A, B. → 6.01

(18) Re: .27–.28 & Z141: If $f_{AB} \ll f_A \cdot f_B$ then α ≡ AB should be defined, & t. cc of corpus will ↓. (I think): Note that ordering of definitions must be correct. Also parsing of corpus must be correct — It may be that we can only get this ↓ in cc by a certain parsing, & that we :. only have one short code. Get to into exact details of pc computation —

Use examples, Maybe use computer to check examples & thereby check accuracy of symbolic calculs.

Also Note! T. ordering of defns is impt. "Greedy" is not necessarily best. Try various

18) (cont) If for large SSZ a steady state, $\forall$. P_C is $p_A^{p_A} \cdot p_B^{p_B}$ $\geq (p_A - p_C)^{p_A - p_C} \cdot (p_B - p_C)^{p_B - p_C} \cdot p_C^{p_C}$

i'l $p_C \neq p_A \cdot p_B$? We should have equality here ↗ i'f $p_C = p_A p_B$.

$p_A \to A$, $p_B \to B$, $p_C \to C$!

$C^C \cdot (B-C)^{B-C} \cdot (A-C)^{A-C} = (A-C)^A \cdot (A-C)^{-C} \cdot (B-C)^B \cdot (B-C)^{-C} \cdot C^C = (A-C)^A (B-C)^B \cdot \left(\frac{C}{(A-C)(B-C)}\right)^C$

i't $C = AB$ then $= (A-AB)^A \cdot (B-AB)^B$

$A^A \cdot (1-B)^A B^B (1-A)^B \left(\frac{AB}{AB(1-A)(1-B)}\right)^{AB}$

$(1-B)^A \cdot (1-A)^B \cdot (1-A)^{-AB} \cdot (1-B)^{-AB}$

$= (1-B)^{A-AB} \cdot (1-A)^{B-AB}$

$= \left((1-B)^{(1-B)}\right)^A \cdot \left((1-A)^{(1-A)}\right)^B$

i'f $1-B = A$ then RHS is

$A^{A^2} \cdot B^{B^2} \neq 1$

$.25^{(.25)^2} \cdot .25^{(.75)^2} = .779475$

$.5^{.5} \cdot .5^{.5} = .5$

v.s. $.25^{.25} \cdot .25^{.25} \cdot .75^{.25} = .25^{.75} = .3535534$

Hvr: If a regy exists, it must be possl. to recode t. corpus wt fewer regys, so that t. resultant code is more "random".

T. Moral seems to be that simply defining $C = A \cdot B$ does not capture all of t. dependency implied by $f_{AB} \neq f_A \cdot f_B$. The finite state machine with A cannot be followed by B does capture this! So that in t. resultant code we always have conditional probys. ← to 20

(19) Re $0 \times B$ & St. Petersburg problem: w. random data & no Bkg., yield will always be ϕ, but Vanc. can be $>>$ Net of data itself.

.16 (20) from (18) we want to code it into a Bernoulli seq. so we can look for regys & "Recode" it. f_A f_B f_C

.20 But: Say we had 3 symbols w. such freqs: 0, 1, 01. How would we detect it? — what regys in corpus? recall, t. frequency of 01 would be $f_0 \cdot f_1$. So perhaps if $f_{AB} \neq f_A \cdot f_B$ define $C = AB$, but how is frequency $\neq$ t. freq. of occurrence of AB! (Maybe good!) ! This coding method has very many ways of "parsing" t. seq. — i.e. one can decide on any of t. AB's as being C, in t. resultant AB's being "coincidence"! So I'd worry to code t. corpus in many (1 ways to obtain

.25 all of t. pc's.

On t. other hand, t. Z141 method gave only one parsing, but it was harder to detect regys in t. resultant sequence (i.e. "reading")

.20 - .26 Seems closer to every understanding of Pres ("Building Blocks" Concept.)

Ah So! ! If we find $f_A \cdot f_B \neq f_{AB}$ by test : First means that $C \equiv AB$ is a useful defn., but it doesn't really tell when to use it! It just gives an $\uparrow$ in pc of C, for coding t. corpus. So its of "strong Heuristic value" (1.20)(#15)

.34 Wow! How do we get $f_A f_B \neq f_{AB}$ by defining $C \equiv AB$? Well! If $f_{AB} \ll f_A \cdot f_B$ then $f_{AA} > f_A \cdot f_A$ &/o $f_{BB} > f_B f_B$. So we define $C = AB$ &/o $D = BB$. If there is a 3rd radix, this may not be so good (?!) (unclear, Gen.)

Probly best Pkg to do: Define $C = AB$, say, then look for trigrams of Unusual frequency if any are found, "they may occur w. suddenly low freq. so one can try parsing t. corpus's all poss'l ways w. them (there could be only 1 or 2 poss'l. parsings) → 90.01

9598 (NN)

3

21) I thot that GA was "Universal" that eventually it would solve any problem — Back 1:30p.

→ False! It suffers from "Local Maxima" ("Premature Convergence"). How Do GA practitioners

.03 → Deal w. local Maxs? (e.g. Koza) (Look at my recent Book on GA.). (Also Note 6.36 crossover of GA)

.04 1:28 22) from (15) (1.28) "Building Blocks" (cont.). In GA, the meaning of F_A, F_B, F_{AB} is unclear; Maybe we mean

frequency in the hy G part of the ensemble of phenotypes (candidates ≡ "cands").

Remember, it's "the finite set of objects" corpus. .. (or even [Ob_i , G_i] corpus. ← This is the (More exact) way to do it!)

OK: Lets Just Consider the [ob_i] corpus! (no G_i considered). We have a small Grammar for it, using F_A & F_B.

We note $F_{AB} > F_A \cdot F_B$ (significantly), so we find the new Grammar Component $C \equiv A \cdot B$. — which compresses the corpus

.09 further. (Noting $F_A \neq F_B \neq$ (significantly) lean of freq. of all symbols, is the first (Bernoulli) compressn.)

.04 – .09 is fine! Now, try to introduce the more exact corpus, [Ob_i, G_i].

By assigning more prob to cands. that occur in (trials / cands) of hy G, we may be approaching the SGA! (Anviety of)

Note that in SGA, we tried to devise a grammar in which the pc of a (cand / s / trial) (of the cand.) was $\propto G$: So G_i was like a wt. of Ob_i.

Is it nec'y to "linearize" G before using it in a SGA? Also, if we make $wt_i \propto G_i$,

What kind of proportionality should we use? It would seem to agree the log of difference. I did write a lot on this. I think [this was] closely related to another difficult Q in SGA — But seemed at first, quite independent. → 7.28

(SN) It is possl. that Cover is NOT a Bayesian! He was very happy w. models that assume no apriori.

.19

.20 There was the General Q in GA of [Any monot. funct of G has same max as G

.21 does: So how can Conventional GA decide what reproduction rate to use?] — They usually make reproduc. rate $\propto G$ (or maybe $G - G_0$) (initial value)

Hadley Lunch 21-4? Sup. ~ 8 to 12 / 9 to 10

They may chose the Const. of proport. w.r.t. G empirically: i.e. run it a while, see how it's doing, then ↑ or ↓ "∝ const." dynamically, slowly. They may not worry about (.20 – .21) at all!

Using classic SGA on .04 – .09 may be easy. Perhaps try it a bit. $wt_i \propto G_i$ or ($wt_i = k G_i$).

Suppose that the final grammar is expressable as a "Base set" on a small set of n-tups. What is its ultimate capability of modeling various p.d.'s on the set of all finite strings? We are mainly interested in good

.27 modeling in the hy P region, since this is where most of the wt. is.

[9 17 98] Re: Vapnik: There was this theorem that $\frac{1}{f(\vec{x})} = \vec{x}\,\vec{M}\,\vec{x}$ was possible — but there was some coord. system in which $f(\vec{x})$ was expressible as an "inner product"?

Maybe what was meant was: "inner product" was $\sum x_i^2 z_i^2$: where z_i were eigenvalues.

So this is thm I used in 1962 to prove the Metric was correct. Maybe Vapnik meant this in some more general sense — for a more general Metric.

(SN) Maybe ask Mitch about localization of site of long. in rabbit cerebellum.

(23) G.A. on the $\tilde{G}$ (approxn G funct for SGA)! Rms error or other simple metric

.33 Betw. G & $\tilde{G}$ is not what we want; Some better ideas: (a) Considering all $\sim \frac{N^2}{2}$ pairs

.34 of the corpus; a min no. of mis-ordering. (b) Say we pick top 10 $\tilde{G}$: cands wrt $\tilde{G}$:

We'd like to contain top G. If G is "noisy" (w. noise ∝ or in G), measure the

.36 "top of G" is less clearly defined — say, its the set of cands within σ of the apparent top.

→ we can't know top G, ordinarily.

→ 5.01

m.glickstein@ucl.ac.uk TM 80

91798 (JW) 0171 504 2888 ← gen. no. at dept Anatomy & Devel. Bio 4

263 7166 — home phone

(SN) On EARLY development of Ethics in Children:

This has mainly to do w. development of "ego" — i.e. what parts of t. world that t. child considers as impt for its welfare. At first, the parents are regarded as part of the envt. that takes care of t. child. ~~Therefore~~ These entities are never threatened (to theory of mind of t. child) so the child doesn't consider them as part of its ego. Later, as t. child begins to understand "how parents work", t. child might feel threatened if it perceived its parents were threatened. As the child grows older, other people may ~~also~~ develop parent-like properties, & t. "self" (t. part of t. ~~[illegible]~~ world that t. child feels is impt in its wellbeing) is expanded: siblings, family, extended family, teachers — may be included.

A moral problem might be: "shall I hurt one part of t. family (self) to help another"? It is similar to "Shall I harm one part of my body, to help another"? It's a very "selfish" view of the world, but it looks like concern for others.

Later in life, w. better understanding of t. world, the "ethics" can become much more complex. "Ego" can include a nation or the whole world — and other forms of ethical evalns. may evolve, that don't look as "selfish". A person may have innate (genetic) "goal" of concern w. welfare of t. larger "extended family" (≡ extended ego).

3½ hrs
3:15am
26° × 9/5
45
32
79
78.8
18 → 69
36
30
645;
78°F
100
26
28.8
52
30
82

91898
1201 P 2mg cod
120 P:
230 P 2.5
5 A 2.5
7:15 A 2.25
930 A 2mg. cod.
930 A.
91998
12:30 A 2mg, cod
12:30 A.
2:15 A 1¾ 4
6:15 A 4
10:45 A

(continued)

3.? (23) In Gathy $\tilde{G}$ one of the main goals of SGA was to use info in the low & part of the sample to get a Good fit for $\tilde{G}$ in hy & part of the sample [Efficient use of sample]

(.03) Hwr, the main goal seems to be that top $\tilde{G}$ cand. is most easily found! One criterion: That top $\tilde{G}$ cand is near from Pk. top wrt. $\tilde{G}$. we want Min k. Ase ss ↑, k will normally ↑.

This sounds not so Good! If we have many parms to adjust in creating $\tilde{G}$, It may be pathologically easy to find a set of parms that put top $\tilde{G}$ close to top G.

(.08) Another / Core for $\tilde{G}$: Best max $\tilde{G}$ should have hi errs by G as possl.

So: Cores for $\tilde{G}$ Rest for: 3.33, .34 ; 5.03, 5.08. || 5.03 may be best thus far. (maybe 5.08!)

Re: .03 May be not concentrate on getting top G! Getting close to top G may not be bad!

Also note, we will usually never know top G any way! so .08 is more retrievable.

.11 What we usually want (i.e. in most places of the calcn. of Max) is that $\tilde{G}$ be able to

.12 Suggest new data pts., so that fitting a $\tilde{G}$ curve thru the augmented corpus either

(a) does .11 – .12 a/o has $\tilde{G}$ of very by G.

Hmm! the Gorcs seem to miss the point: I have [config i] $[O_i, G_i]$ data set.

I want $\tilde{G}$ to get rid of/order any G_i for $\{O_i\}$ — particularly in the hy G region: Most of the previous Gorcs, involve new trials. $\tilde{G}(O_j)$ is available for any/newly O_j, but $G(O_j)$ is available only for old trials — or at hy cost if O_j is a new trial.

Hwr, the Gorcs can be made "operational" by limiting the data involved to the (new) set $[O_j, G_j, \tilde{G}(O_j)]$. [$= G(O_j)$]

So we can ask that $\tilde{G}(O_j)$ may give us ashy a G_j as possl.

Actually, I want ~~the best G of the k (say) pts of $\tilde{G}(O_j)$ over data set.~~

the $\tilde{G}$ ordering over Pk. hy $\tilde{G}$ pts of $[O_j]$ to be close to the G ordering over those same $[O_j]$ pts. — But really the known $[O_j\ G_j(O_j), \tilde{G}_j(O_j)]$ pts. (even Case of law $G(O_j)$) should be used! (Efficient use of the corpus).

.29 **E.g. if** G is linear in X (X is a scalar) so $G = a + bx$, but a & b are not known. $G(x)$ for any values of x, is useful. or say $G = a + bx + cx^2$, values of G for any x are useful! But some are more useful than others.

.32

.34 Essentially what we want is that $\tilde{G}$ ordering be close to G ordering over the known data set — So it is unlikely that as-yet-unexplored O_j regions of very by $\tilde{G}$, will also have very by G. We will perhaps bias wt. of orderings in by $G, \tilde{G}$ regions. → 8.01

IMPT I think there may be a fair amount of literature on "Ordering" in statistics. How to find it? Who to Ask? There may be refs in my book on "Social Statistics." #25 → 6.32

19 IMPT. The "number of misorderings" or sum of Gorc errs should be used to correct $\tilde{G}$ so it is a "perfect code" of G (i.e. has same ordering). (See 9.01 for a version of this idea ... but no soln.!)

~~24~~ / 1.30

.01 : 1.30 : (24) from (16) : Say we order them so low $\frac{cc_i}{pc_i}$ is first.

Say $[cc_i, pc_i]$ are a known set of cc, pc values. Let $i(n)$ be a particular ordering of the trials:
so $i(1)$ is first, $i(2)$ second, etc.

Then $\sum_{n=1}^{m} cc_{i(n)}$ is the total cc of search, if $i(m)$ is the correct trial.

.05 So $C_0 = \sum_{m=1}^{\infty} p_{i(m)} \sum_{n=1}^{m} cc_{i(n)}$ is expected search cc for the $i(n)$ ordering of trials.

Now say we change the $i(n)$ ordering by exchanging adjacent trials at the
k & k+1 position. So the ordering is $i(n)$ for $n < k$ & $n > k+1$.
for $n = k$, we have $i = i(k+1)$; for $n = k+1$ we have $i = i(k)$.

In .05, the sums w. & w.o. reversal are the same except for the 2 terms when m=k & m=k+1.

In the normal unreversed $i(n)$ case, the 2 terms are:

$$p_{i(k)} \sum_{n=1}^{k} cc_{i(n)} + p_{i(k+1)} \sum_{n=1}^{k+1} cc_{i(n)}$$

$$c_0 = p_{i(k)} \sum_{n=1}^{k-1} cc_{i(n)} + p_{i(k)} cc_{i(k)} + p_{i(k+1)} \sum_{n=1}^{k-1} cc_{i(n)} + p_{i(k+1)} \cdot cc_{i(k)} + p_{i(k+1)} cc_{i(k+1)}$$

In the reversed case the 2 terms are:

$$c_1 = p_{i(k+1)} \sum_{n=1}^{k-1} cc_{i(n)} + p_{i(k+1)} cc_{i(k+1)} + p_{i(k)} \sum_{n=1}^{k-1} cc_{i(n)} + p_{i(k)} cc_{i(k+1)} + p_{i(k)} cc_{i(k)}$$

The red boxed terms in c_0, c_1 cancel.

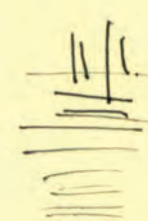
The terms that don't cancel are: $p_{i(k+1)} \cdot cc_{i(k)}$ & $p_{i(k)} \cdot cc_{i(k+1)}$

$$c_0 - c_1 = p_{i(k+1)} \cdot cc_{i(k)} - p_{i(k)} \cdot cc_{i(k+1)}$$

so $c_0 - c_1 > 0$ iff $p_{i(k+1)} \cdot cc_{i(k)} - p_{i(k)} \cdot cc_{i(k+1)} > 0$

so $c_0 > c_1$ iff $p_{i(k+1)} \cdot cc_{i(k)} > p_{i(k)} \cdot cc_{i(k+1)}$

$\equiv$ iff $\frac{cc_{i(k)}}{p_{i(k)}} > \frac{cc_{i(k+1)}}{p_{i(k+1)}}$ } i.e. the ordering is wrong.



I.e. if the ordering is wrong, we can ↓ c by reversing the k, k+1 elements in the ordering.

This Algebra came out a lot easier than it usually does!

.32 : ~~5.23~~ 5.40 (25) There was this business in normal G.A. of making no. of offsprings ∝ G or ΔG:
Say no. of offsprings = k ΔG: Then k is like γ in sharpening the bias toward the old-known G cands.
— i.e. "Like" using G^γ for reinforcement. I think k is usually chosen empirically:

.33 This is a genl. diffy in G.A. — That any monotonic ↑ funct. of G will have same peak & should
.36 be equally good for simple optm finding! But it's not so in GA, so GA must be wrong! (See 3.03 for Another criticism of GA)

In 5.34, we are concerned only w. ordering, so objection .35–.36 doesn't apply.

We can get the effect of "γ" by giving more or less wt., depending on the absolute ordering of the cand pairs.

Another thing to consider is the distance betw. pairs that are in wrong order.

9.20.98 (NN) 1983 7

.01 (26) TSq. for Analog tasks (in Cerebellum)! Start w. large ∈ (Vap units ∈), Then ↓ as try proceeds
[9.20.98] — to final ∈ = 0. Since V. has a pgm to fit curves of various types w. arby ∈, this might be easy to do.

12:30A 2 mg card. | For discretely decide task: slowly ↓ no. of errors made!
12:30A 3¢ | for "GPS-like" task (Vector error). Use slow ↓ in some norm of error vector
3:45A 3¢ | For k·2^k input Mixer, start w. k=1, Then 2, ect. — interestingly enuf, this is
7A | how it learns even if it starts w. k=3 (or k>>1): It suggests that w. only "and, or, not,"
12:30 2 mg card | ("If" optional ... "If = k ="), it could learn k=1, Then k=2 ect, even
if it were asked for k=3, initially — so I would boot k=2, even if "and, or, not" were present.

Greco: Back by ~2 AM sm

.08 (26½) So ~ 2 impt. Q's about GA (& SGA?)! 1) How no. of children depends on G of parents 2) Is mutation or Crossover better? I find it hard to believe that people seriously consider "Mutation only": in Mut. only Each individual has to find its way up the hill, himself. "Generalized Crossover" enables individuals to pool their experiences to devise optimum "child" cands.

.01 for expression of kinds of probs TMB can worken

G.A. w.o. Crossover is very close to Simulated Annealing. G.A. w. Crossover is like having all cands mate w. t. same cand (which furnishes mutations) (t. polygamy theory of evolution)

Suppose trials are extremely expensive (Human, animal large animal reproduction).
Then one should take as many of known (O_i, $G(O_i)$) pairs as possl. & use them to devise

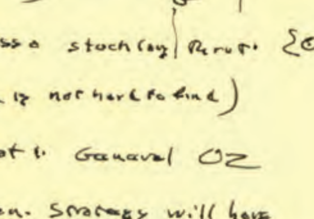
.17 O_j's w. either ⓐ hy expected G or ⓑ O_j's that are "very informative".

.18 (27) E wrt Goal ⓐ! Just pass a stoch lang thru t. [O_i $G(O_i)$] data set — or, more exactly, Get t. optimum Exgoln. operator for O_i input. (I have to think this thru, better!): There is t. "One Shot Lrng" Q: doesn't do OSL properly? OSL is very impt., very common. So for each O_j we get a P.D. for $G(O_j)$. We can call this P.D. $\tilde{G}_p$ — T. expected value of $\tilde{G}_p$ is $\tilde{G}$, say. So we could have a $\tilde{G}$ function of t. O_j, so $\tilde{G}(O_j)$ = expected value of O_j. Hvr., this may be missing t. point! If we do lots of O_j's w. large variances in their $\tilde{G}_p$'s we could get a very hy G that way! (If t. D.f.'s are sufficiently Uncorrelated)

.26 → T. forego system looks very much like SGA: i.e. pass a stoch lang thru t. [O_i, $\tilde{G}_i$] data set (w. additional constraint that O_j a $\tilde{G}(O_j)$ is a max is not hard to find)

.28 : ~3.19 (28) (SN) It may be that w.o. linearizn. of t. Gorc, that t. General OZ problem is ill (incompletely) defined. For a fixd (compu. Bnd CB) each optzn. strategy will have a d.f. on final G values. Unless we have linearized G we can't tell whether one D.f. is better than another. Hvr., in many cases, t. difference in choice given by various linearizns will be quite small. In some extreme cases hvr., it can probly be enormous.

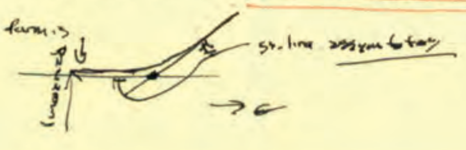
Example of very non-linearized Gorc! $(1-e^{-G})$ = linearized Gorc



So for G >> 1, it doesn't pay much to ↑ G.

One could do a rough linearizn of t. Gorc, (if possl.) before doing SGA.

Another kind of Gorc function



In this case, if $\tilde{G} \approx G$, for G > 0, but differs much from G if G < 0.

9.20.95 (NN) REVIEW: (.05)

.01: (28) (cont.) For G < 0, hvr, these are low G pts, & we don't care if $G \neq \tilde{G}$ in that region. T. only reason we mite want G to be close to $\tilde{G}$ in t. low G regions is that this would enable us to use low G data to fit t. $\tilde{G}$ curve better in (hopefully) t. hy G regions as well. ([redacted].5.34 — discusses explan. of "ordering").

.05 (29) Review of impt. ideas! * : Need work

(A) Understanding of proper soln. of Z(4): 2.20 — 2.40

(B) Development of EPrics in Children: 4.01 — .40.

.09 (C) 7.26 : An understanding of ≈SGA: How it may be a kind of optimum soln. of Optzn. problems (not exactly OZ. problems).

.11 (D) 7.01 A Bunch of kinds of lrng that can be [redacted] put into t. form of TSQ's! This list could be readily Expanded. [redacted] 7.01, .32 is better, hvr.

(E) 6.01 9.20 : Proof of t. first Gambling House thm ($\frac{c_i}{p_i}$ is best ordering).

* (F) [redacted] (5.39 / 5.34) : " " Second Gamb. House Thrm. Time to soln is $\leq \frac{c_j}{p_j}$ (j is soln). On "no. of Misorderings" as a Score for fitting a $\tilde{G} \approx G$ in ≈ SGA. Also note 5.39 and 9.01

.18 * (G) TM75.25–76.40 TM$_A$, TM$_B$ [redacted] A Clearer understanding of My New Method of Designing TSQ's or Tng. envts": (Also note 7.10 (D))

*{ (H) Mutation only v.s. Crossover only (or Crossover + Mutation) : A path to understanding : 7.08 – .17 Also general Q of How normal GA works (crossover): ↙

(H') "Building Blocks": [redacted] 1.24 – 1.29; 3.04 – 3.09: 3.09 – .27 How it works: Good Suggestions: BUT NEEDS WORK!

5.34 on misordering = Score.

1.24 (15) may be impt.

.3

9/21/98
9/22/98 .01:5.40 (30) Amusing ~~AIP~~ problem! (NOT a MATH Problem!) Given a sequence of n objects that are in the "wrong order". What's the least amount of info. needed to put it into rite order?
12:01A 2mg card
12:01A
2 95A 2½
5 45A 3
8 30A 2mg card.
"Cost" of a particular code is unclear! Say 2,3,4,5,6,1 is observed order. — We need 1 2 3 4 5 6. We could do it w. 5 exchanges: (6,1; 5,1; 4,1; 3,1; 2,1) or one "jump" "1 from one end to the other." — assuming others all moved.

9/22/98
1130P: 2mg card

Another approach: List all n! orderings. Tell which ordering it is in the list. There are n! orderings.
For large n, almost all orderings are "random" (I think).

9 25 98
2 15 AM 2mg card
2 15 AM
5 45 3½
2½
8 30
8 30 2mg card

Actually, G will "order" the O_i, but several O_i can have same ~~order~~ position! Similarly w. $\tilde{G}$.
A way to assign sort of "BAdness" to an ordering! Consider it to be created by "noise" of some kind (Gaussian, $e^{-|x|}$, etc). "Noise," hvr, is most likely to occur uniformly over the range/domain of G if G is "linearized".

9 26 98
~2A 2mg card.
4 45A
2½
7 00A
10 A 3⅕
3 P
3 P 2mg card
9 27 98
2 AM 2mg card
2 AM
5 45 3½
12 45 P 2mg card

That we are interested only in the ordering: Arg't: Say G that we are using really
$G = H(k)$. k is a simple funct. (short term), but H is a monotone funct based on random functs, so it has hy cost. So we'd have lots of trouble approximating G, but much less trouble simulating k (via "same ordering").

Note Also arg't. of 5.25-.32 on why pts. of low G can be very "informative" about the shape of G.

domain
domane.

$\frac{c_j}{p_j}$

$p_j \sum_{i=1}^{j} c_i$

$\sum c_i < \frac{c_j}{p_j}$

$\sum p_j c_i < c_j$

$\sum p_j c_i < c_j \sum p_j$

$p_j \sum c_i < c_j \sum p_i$

$\frac{\sum c_i}{\sum p_i} < \frac{c_j}{p_j}$

9 28 98
12 30A 2mg card
12 30A
.20 5A 4½
8 A 3
10 A 2mg card So
No dizzyness.
9 29 98
12 15A 2mg card
12 15A
3 30A 3½
6 15A 2½
11 A 2mg card
9 30 98
2 A 2mg card
2A
5 30A 3½
9 30A 4
11 45A 2½
1 PM 2mg card

Proof of Second Comb. Insp. Thm: If (j) 1st. soln is we do Trials in $\frac{c_i}{p_i}$ order, So $\frac{c_k}{p_k} \le \frac{c_{k+1}}{p_{k+1}}$ then Time to find soln. is $< \frac{c_j}{p_j}$!

Time to find soln is $T \equiv \sum_{i=1}^{j} c_i$: $\frac{c_i}{p_i} \le \frac{c_j}{p_j}$ if $i \le j$.

So $\frac{\sum_{i=1}^{j} c_i}{\sum_{i=1}^{j} p_i} < \frac{c_j}{p_j}$ $\therefore T = \sum_{i=1}^{j} c_i < \frac{c_j}{p_j} \cdot \sum_{j=1}^{j} p_i \le \frac{c_j}{p_j}$ (since this is ≤ 1.)

This proof suggests that T is usually $<< \frac{c_j}{p_j}$, because the inequalitys & $\sum_{i=1}^{j} p_i \le 1$ are usually strongly on the "<<" side.

.30 Building Blocks: from 1.24-1.29; 3.04-3.27: The Argument of 1.24-29 involves $f_{AB} >> f_A \cdot f_B$, which is used for code compression in ALP.
Hvr, the cond. for defn. of A·B is that G(A) "+" G(B)? G(AB) > G(A) "+" G(B) —
i.e. A & B are "synergistic" concs. &
I let G(A) act. mean the max G of cands containing conc. A.
Possibly before we can talk about — We must linearize G. This is also a way for detection of "Synergy".
So this "Building Blocks" idea is beginning to look rather distinct from Z141.

One way to look at "crossover" is that we do 7.18! pass a stoch operator from the set $[O_i, G_i] \to [0,1]$

BUILDING BLOCKS (.01)

.01:9.40 Then we use this stoch operator to try to get hy G cands: In crossover, we reduce the [redacted] SSZ from [the cardinality of $[O_i, G_i]$] to just 2! Using Koza's functional ~~mutation~~ crossover (of function trees), we do get a stoch lang. from any pair of cands. Since the [redacted] SSZ of 2 is very small, the nature of the induced stoch lang. depends heavily on the a priori. In devising a stoch lang. from a pair of cands, we do get hy σ^2 in the G prodn.

92198 (NN) 1M 86½

11:15P 2mg card.
11:15P
[82298]
3:45A 4½
8A 4½
8A 2mg card
1130P 2mg card.
1130P
[92398]
215A 2½
430A 2½
730A 3
1145A 4½
1PM 2mg card.
[92498]
12:15AM 2mg card
12:15A
3:45 3½
5:35A 1½
5:15A 2mg card.

[0198]
[12:45A 2mg card]
12:45A.
430 3¾
645 2½
915 3½
1140A 2½
[1140A 2mg card]
[0298]
[1AM 2mg card]
1AM
5A
645A 1¾
845A 2
11A 2mg card.
[11P 2mg card]
11:30 —
[0398]) 7!!
630
1045 4½
[1145A 2mg card]
443P 2mg card
[0498]
330A 3½
6A 2½
15A 1½
9A 2
[1030A 2mg card]
0598
[12:01A 2mg card]
12:01A ....
545A 5½
815 2½

A1) Re: OxB; Try starting TR (1/2/3) days later each quarter! See if phasing is critical in $\mu, \sigma, \frac{\mu}{\sigma}$.

A2) Machine Lrng & t. Goals of A.I. Where are we Going, and where can we Go. Title for talk at MIT. a/c alSachum.

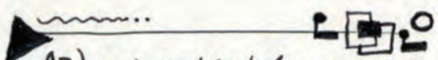
A3) VAPNIK: ⓐ for curve fitting w. n data pts: if we use ms norm & $\epsilon = 0$, t. no. of "support vectors" is n.

ⓑ If we do "separation of classes" w. hyper manifolds, and only incomplete separation is possl., we have a certain no. of "errors". I think V. gives = wt. to all error pts. This seems very wrong.

A4) Give a series of 3 or 4 talks at FAP! first similar to London talk — (as intro) then following lectures Giving More detail.

A5) T. World of Math: J.R. Newman: Vol 2: interesting stuff on Proby: also QM: By Shröd, Heis etc. Also C.S. Pierce.

A6) Arthur Koestler. | ⓐ Biog Stranger on the Square: Author & CyPher Koestler 1940–1956 | A.K. died 1983
ⓑ Arrow in the Blue: 1905 – 1931 (+Born) Joined comm party.
T. Invisible writing. 1932 – 1940 + (Epalog 1940–53)
ⓒ T. case of t. Midwife Toad)
t. Roots of Coincidence.
T. Challenge of Chance.



A7) what kind of sub-corpus is OZ data? [redacted]

A8?

Compass
Starstruck
Our Count. Good
They like Us
Bloo room
Handbag · Parenting

92298 TM v.s. Indexing input (.10) (.27) TM87 11

1) t. CPD can contain all kinds of "stuff" — including statements, "theorems", an "oracle", TM itself, etc. integrated.

T. way this works: comes e.g. for compressing internet (say) can be used in solving other problems. We do have to have a cond'l P.d. that suggests which comes are likely to be useful where. (t. IR problem).

(Very) Early in this work (perhaps just before) This Tip I had t. idea that partitioning of data/set could be used as a "Hint". Ordinarily t. data remains partitioned somewhat (at least, temporal ordering). It can also be partitioned by "topic" by t. trainer.

This last partitioning can be indicated by an index no. that comes w. each data chunk, e.g. each of t. chunks of .01-.02 would be assigned an index no. Sub-concept could also be given. Sub.(s sub...) indices by trainer. Later, TM itself may want to do some index assignment.

27 In opposite direction from (partitioning indexing) is "pooling" — an important part of t. induction process. Pooling finds, indexing, partitioning finds, but in a useful way — it finds size of enemy in which to hunt for usable sub. comes to combine.

"Indexing" is perhaps best defined as a hierarchical type of "partitioning".

32 As is, it appears that we can integrate into various kinds of "Grand CPD". One kind of input that might be very useful: Traces of TSMs or Problems worked by other Learning Machines. (Preferably machines similar to TM)

This would in ideality, TM would be able to integrate

.03 such info into its own Grand CPD. This is a v. g. way TM88
to integrate TM's from hard-won & difficult problems a/o in difficult problem areas.

.04 Its not clear as to how difficult TM would find this problem 92.32 good!

2) I want to write up (8.09) (7.19-.26 ?) (8.11) (8.18) in some detail: My new "Grand Plan" for TM: How far I've gotten; what's solved & what's just what's unsolved. Do it mainly in "English" Doesn't get too detailed yet.

This will be a review of The most recent "Grand Plan" for TM!

Impt refs: 8.09 (C) 7.18–.26 An Understanding of SGA: Toward a class of Good Solns to opt-zn problems (not exactly OZ probs)

8.11 (D) A Bunch of kinds of problems, inputs, trng seqs. This latest TM should be able to deal w. See 7.01 very expendable! 11.01 in particular 12.04 (≡ 87.01–88.04)

8.18 (G) Main PLAN: TM 75.25–76.40

TMA: [72.02–.40] (Expo) finds short codes for arby corpus types: sequential a/o unordered set of objects or partial objects

TMB [66.01–65.40] Conditional proby Ing. Machine: Is able to solve great variety of problems by pas using a (common) grand Conditional Proby Distribn (CPD). (Def CPD)

Also, pp. leading upto 66.01 are very relevant (good) 7.01 is also

---

Re: TMB: There is some confusion in my mind as to just what it is! — 66.01–65.40 was an early picture: I think I modify'd my idea (often) (may be w.o. writing much on it. modifn, before writing 87.01–88.04 (7.01 was minor).

So maybe an early idea of the [cond proby] CP Machine is: In steady state, it has within it, a CPD; This CPD takes form of: we put an augmentation of the corpus in it — including a "partial augmentn": It's able to give a P.d. for the contnmn. of the partial corpus. (a/o of minimum likely "or" "very likely" continn(s).)

(The "augmentn." a/o the "partial augmentn." can be null.)

At first, TM would be able to do this in carefully labled "Domains" w. no interdomain backr. domains. Each Domain would be completely separate from all others w. its own separate TSQ. Later, TM would try to integrate the P.D.'s of all of the Domains into a "Grand P.D." for all domains: At first each Domain was like a Sol89 machine w. its own carefully designd TSQ, But "Conditional Probys" for IR in "LTS Memory." — to reduce research IR from Memy.

Expanding this idea into 87.01–88.04 was an impt. "Breakthru", but needs much detailing. This last expanded considerably kinds of problems TM could work on a/o types of info it could use for inputs.

(SN) TM has contents of all time, either the original corpus or a very easily decoded compressn. of it a/o compressions using very hypo definitions: like words in english separated by spaces, Rest are in the dictionary, will be specially coded. 67.12 (67.12) 13 an impt. Q.

.29: 67.13: Later I began thinking about this "sufficiently advanced" state of TM: Presumably, at this point, it would be rather easy to teach TM things. One could design TSQ's in an intuitive way — as one does for Human Students. At an even more advanced stage, we could give it text books designed for Human Students. So one big Q is: could we design TM so that Ab mito it was at or fairly close to one of these levels of intelligence.

.01 (3.14C / 2.90): When parsing a corpus w. C ≡ A·B substitution: one can never be sure that a specific $C_0 \leftarrow A \cdot B$ (or even have evidence for sure) is correct, unless that C_0 has been made part of a larger ntup. defn. — otherwise, all

.03 possl. parsings (not involving / involving) t. substitution $C \leftarrow A \cdot B$ are of = pcost.

→ 109.01

T. essentials of t. stochastic Z141 lang. are: C.F rules, but no loops allowed. Since t. lang is finite, there are a finite no. of ntups defined. Each ntup has a frequency assoc. w. it, that can be calculated from t. pc's in t. grammar reg.

.07 $\Sigma \xrightarrow{p_A} A\Sigma$, $\xrightarrow{p_B} B\Sigma$, $\xrightarrow{p_C} C\Sigma$ — Bern seq.; $C \xrightarrow{1} AB$ | be perhaps t. whole grammar.

T. original corpus only has A & B in it: they occur w. frequencies $f_A + p_C$ & $f_B + p_C$, resp.

.12 Another possl. Grammar

$\Sigma \xrightarrow{p_A} A \cdot \Sigma$, $\xrightarrow{p_B} B \cdot \Sigma$, $\xrightarrow{p_C} C \cdot \Sigma$, $\xrightarrow{p_D} D \cdot \Sigma$ $\qquad C \xrightarrow{1} AB \qquad D \xrightarrow{1} CA$

observed freq. of A & B in t. original corpus (which contains only A & B) are:

$$f_A = p_D + p_C + p_A$$
$$f_B = p_C + p_B .$$

So we take this corpus containing A & B that seems to have dependencies betw A & B, & we make a Bern seq. of ntups each being a finite mix of A's & B's. T. probys in a Bern seq. are indep.

So: I idea of "Z141" is that t. Bern "grammar" of .12, with defns of C & D is a larger pc. of deriv of t. corpus, than (a) t. original corpus as a Bern seq, or (b) any other grammar of this type. | (b) is rather uncertain: we usually cannot try all possl. grammars of this type.... Unless we find a systematic way to do it in reasonable time.

After finding a rather large no. of defns of this type that ↑ pc of corpus: Can one find "looping rules" if it is, indeed, an infinite lang. w. loops? Well, t. grammars of .07 & .12 both divide into 2 parts conforming (a) Subsets of Σ w. various probys {contains a simple loop.} & (b) a part containing other subsets all of proby 1} No loops. Look at part (b) alone & see if we can find loops to define & thereby ↑ pc of derv.

Lets looks at 1 or more simple CFG's that have loops

$\Sigma \to a\Sigma$, $\to \Sigma b$, $\to b$ } probys give expected langs of strings of $\binom{b}{a}$'s. : lang = $a^{(n)} b^{(n)}$ — This lang also has grammar $\Sigma \to AB$, $A \to Aa$ (or aA) $\to A$, $B \to Bb$ $\to b$ (or bB)

Lets look at this lang: ab occurs readily: we will define $A_0 = aa$, $B_0 = bb$.

$B_1 \to B_0 b$, $B_2 \to B_1 b$, etc. $A_1 = A_0 a$, $A_2 = A_1 a$, etc.

so $B_{n+1} \to B_n b$; $A_{n+1} = A_n a$. | Both of these suggest $B = B$

.01 $A_0 \to aa$; $A_1 \to A_0 a$ .... $A_{n+1} \to A_n a$: $A \to aa$; $A_{n+1} \to A_n a$.

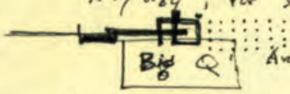
so $A \to Aa$, $\to aa$) or $A \to Aa$, $\to a$ } Better. Maybe we get this w. "folk" reasoning"!

th. A_n ("only left" objects) form a class because they are all of the form $A_{n+1} a$.

The A_{n+1} objects form a class because they are always combined as $A_{n+1} a$. 2:30P

so try $A = A' a$. But no the A_n & A_{n+1} classes are the same.

Anyway, for some reason or other a large set $A_{n+1} \to A_n a$, $A_0 \to aa$ suggests $A \to Aa$, $A \to a$.



Arises being given many ss of the large or just one long s from the lang?

In the original Z141, I was analysing a potentially infinite string. When I discuss "loops", I'm usually talking about a corpus that's a large set of finite strings (= ass). This last is perhaps closer to what I want to use Z141 for: i.e. Ftee coding by creating of new codes by combining / "chunking" old codes.....

.17 Even w. the single infinite string corpus my Z141 analysis is o.k. (i.e. expressing it as a fam seq.)

The idea of further compression by compressing the grammar alone is interesting.

.19 In general, this "2 step" method of getting a "loopd" grammar, is not efficient!

i.e. Many ngms will not initially be defined because their defns are too expensive.

Later, these defns would ↓ in cost thru "Looping": So the defn of ngms & the

detection of "looping" should, ideally, be done together, for max efficy

("efficy" means max compressn. for a given corpus) [≡ SSZ]

Hvr, try the inefficient "2 step" soln. of .17-.19 to get a feel for what's going on — as well as a possibly "default" soln. to the PSGd problem.

~~One catch~~ At least 1 problem is that we might inadvertently write the "first step" grammar

.25 as $A_0 \to aa$; $A_1 \to a A_0$, $A_2 \to A_1 a$ .... which would waste SSZ. We could here

various ~~xform~~ "free" equivalent xforms of the first-step grammar out of the .25 form into α 1 form

So to study the xform of "step gram" into ("first step grammar" / "Loopd Grammar"!) look at various CFL's

& see how various loop instruction sets give rise to various lang forms!

~~The~~ The Paris paper is of interest (s0160?). The idea of ~~Loops or~~ crossed loops

as "loop marker" pts: like $A\,\gamma_1\,B\,\gamma_2\,C$ $\gamma_{1,2}$ are "loopmakers".

34 so $A\,a^n\,B\,b^n C$ could be an (acceptable sentence as.). To detect such Grammar rules,

Def SOG the SOG (step one grammar) needs to have rules of type $A_n \to a\,A_{n+1}\,b$ or equivs.

("equive" means in this case $A_n = a\,A_{n+1}$, $B_n \to B_{n+1}\,b$. Hvr, and $[A_n] \neq [B_n]$

can be the same, is unclear. So perhaps in creating SOG's try to put them in

form .34 if possl.

In summary: ~~Design~~ Devising the SOG involves trying various parsings to see which ~~give the~~ parsings give the best "next level" compressn (i.e. most new ngm defns. at the next highest level). → 109.01

→ 97.03

9/24/98 TM

I want to rethink the list of "types of TM inputs" on 87.01 & .32 this gives good
.02 ideas on How TM can work, what kinds of TSQ's would be very useful! → .08
.03 7.01 deals w Analog TSQ's. How to design them. Also should some other kinds of TSQ's
that can be represented as open probs, of ∀ type verifiable by GA & probly by SGA.
It will be interesting to design TSQ's for parts of Robot arms that have various constraints.
Also, learning to walk — to (reach balls, throw balls) using robot arms. To play
ping pong, using various constraints on speed of paddle motion — constraints
on speed, reach, accuracy of optical pickup. Constraints of hysteresis
in servo action to move paddles, etc. (93.36 has a good approach to ping pong)

11:50
1:50
13:00
11:50
12:50

So essentially, we would use computer hardware! Or maybe (just on V.G. " not so bad")
.07 H.W. to start, a slow degradation of H.W. as a kind to TSQ. → 93.36, 94.05

87,88: very nice pp!

08:02 : Some kinds of inputs to TM. (1) "statements (usually, but not nearly "true") — they can be "facts", heuristics, "hints" of various kinds for specific problems. (shades of McC's "Advice Taker" I'm sure really forgotten details of what kinds of advice it took, & what form it had.) ← perhaps related to .32?

(2) "Wired in" info: I'm not sure just what this should mean. One possib: would be a computer s/w set of computer pgms. that TM would learn to use in solving various problems: say Mathematica or Maple, or various spread sheets. Mathematica & Maple contain various sections designed for various domains. TM would have to learn (a) how to get access to these various "Modes of problsolving" & (b) how to use each mode. Maple (e.g.) has online help, but TM would have to know how to read, to some extent, to use this info. In the case of Maple this "online info" contains very little reference to R_W, so it may be a good place for TM to learn English! Good, because TM would presumably have much knowledge of the domain, before trying to learn to understand English text about that Domain.

(3) Encyclopedia: This could be put in "ab initio" as a while away. (or the internet)
or it could be put in as a "text understanding TSQ", in which we started w. a simple corpus w. simple rules & slowly moved toward the ultimate unrestricted Encyc. text.
As for the Internet: In attempting to delay TM's discovery of R_W the internet could be put in, in Recorded Snapshot form (e.g. Brewster's "snapshots"). The idea here is that TM does not find it possib. manipulate the "snapshot" and could do so w. the real internet.

A perhaps unavoidable way that TM manipulates the environ. is thro' its answers to the user. We could reduce this effect by pre programming our sequence of req vests to TM (// how req vest: d is a funct of all previous actions/replies of TM)

32 (4) (87.32-88.04) ← perhaps .08? : Various kinds of info usable by TM as inputs!

(a) Traces of problems worked by this TM in the past, as well as traces of problems worked by other TM's (particularly as to how TM) or in general, traces of various probs. solns. given in various degrees of detail. Text books should be a (perhaps extreme) case of small amts of detail. Another would be proofs of Meta thms & solns. of Meta probs given in literature.

(5) Analog problems: 92.03, 93.36, 94.05...

.01 An approach to better understanding ("Detailing") TMs would be obtained by debating various ways that TM would deal w. each of its input types: The input types of 92.40 are "novel" i can be of much import!

So no more conventional input (≡ problem) types!

.04 (1) Extrapolate a set of [Q_i, A_i] w.r.t. W.O. OSL (one shot learning), I think all 3 TSP's I've "written" are of this type — But for MTM: I could do it for harder MTM probs & for NMTM probs.

(2) Simply find short codes for a corpus. I haven't really tried this for a large ordered corpus (≈ "regression") — Re .04 was done in Alg. Not. Lrng. as a kind of short code finding problem viz Lsrch. → 94.01

.10 A perhaps important idea: Fast storing for short codes is always best done by

.11 Lsrch, using Conditional probs learned on (temporary corpus) sections of the corpus → 94.01

.12 SN One way to solve IR problem for many access: for example: We solve the simple Alp. notation Lrng problem as we already did — w. somewhat larger cost of many access. When this occurs, the system is in good shape for looking for CPD's on many subsets of its access! The more the problem costs, the more benefit we get by finding a good CPD! Think of various (usually "soft") partitions of the many that could be helpful in retrieval. / Recency (temporal) is one.

(.12) is perhaps the first real use I've specified for CPD: Tho it is a bit vague, it's still looks very promising. The vague part has to do w. the kinds of partitioning/indexing TM will use for this problem. I'll perhaps design a language (instruction set) for implementing various indexing, partitioning techniques. (See 87.10 ff for discn. of "partitioning" vs. "indexing" vs. "coding".)

For partitioning a set we can use Boolian functions of subsets.

Hierarchical indexing is a kind of partitioning using AND only. We can then "OR" these / "AND" conjunctions. Negation is not normally used — but perhaps should be.

33 → What about Relational data bases & the "Join" operation? This is certainly an important kind of operation & a good way to organize data! (or whatever it is called!) Lots of Scientific Data takes form of flat files that can be "Joined" in useful ways! However, I should look up my writings on Reldata bases (in my Ektron files) — also, perhaps work of others on Relational D.B.'s! Perhaps ask Pat O'Neill for good Refs.

.36! 92.02! HA! The cannibalism simulation! Since analog. expts. in R.W. are expensive, TM. makes a model of R.W. using whatever "Hard" data is (≡ expensive data) is available. It's "H.C. problem. w. expensive trials" so much (cf tuning) is spent devising a v.g. model. → 94.05 spec

92498 TM ANALOG Lrng (.05)

.01! 93.11 93.09 Σ • Finding short codes for corpus. 93.10-.11 is a sort of ratcliff/obershall for "coding à recoding".
I'm not so happy w. my "soln" of x. "Alg. reflection Lrng" problem: T. machine used, had too much info in it — i.e., it was a "stack" machine, à it had to Lrn an awful lot. — But
.04 actually, maybe a stack machine is not so bad, after all! — Since CRL's are common (CE Rank!),
.05! 92.07 93.40 (SN) An interesting kind of analog Lrng!: TM (not to simulate motion of real people, for simulated actors): As input, have motion of real people.
Have outputs of machine be many humans à have them be Lrn (frequently) "scored" by humans. (Clearly, this is an example of expensive F.B.; so the remarks (à methods) at 93.36 -.40 are relevant.) → 96.20

~~A more difficult problem:~~ We assume in trying that TM has been given a way to take human (video, 2dm, say) figures, à it has been given parameters that can move the limbs, torso, head, etc. A more difficult problem: TM is given video motion pictures of humans moving about. Say persons A, B, C are shown doing various "plays": ~~TM machine must~~
A is shown doing tasks α, β, γ: B is shown doing tasks ε, δ, η.
We then ask (somehow) for TM to make a video of "A" doing task δ.
One application will be Blackmail: To get TM to make videos of the Blackmailees, doing illegals and embarrassing things. Sell it to CIA, FBI, Mafia! ☺. — See .37 for a possible way to do this.

The ultimate goal, however, would be to be able to give TM a description of a person doing a certain thing as well as perhaps some pictures or videos of the described actor — à TM would output a video of that actor doing those things. This would be a step toward having TM present Plays, from the Ascii text of the play.

Or TM sees a play done w. one set of actors à makes a video output of the same play using different actors.

The foregoing tasks for T.M. may in themselves not be of much interest as goals (or outputs), but writing the expert system for them would be of interest.

.37 One easy approximation of a lot of the above could be done by Morphing one person into another. Good Morphing involves intelligent selection of

.01 Corresponding pts. in M. ≥ objects being xfmd (Morphd). TM could learn how to do this.

Woops! ~~Accepting~~ Ordinary Morphing would not do it! We have ~~Person~~ Person A in position α & Person B in position β. To construct Person B in position α.

Say we had lots of pictures of B in various positions from various directions. Then we could morph B onto A's position α.

.12 Other problems like this. Perhaps would be to simulate a certain person given many "voice examples", saying an arby thing, or simply an arby song.

.12 might be a lot easier to do than this video of 94.05 Af. — yet the main issues of this TSQ's could be many N. .12 might be a TSQ. for me on how to do things Af ☺. Actually, this is not so easy. People have dialects, so may say certain words in certain ways (e.g. Matt's gf. : "special cases" "exceptions"). An "incomplete" set of words said by a person could result in a simulation in which he said a word in the wrong dialect! This problem would have to be carefully studied: perhaps by humans, perhaps by TM! ☺. (99) 6.12

.24

.25 Several Major (General) Problems of TMB! (RM) Rosenujin

1) How to solve TSQ's for various modes (Domains).
2) " " Deal w. IR in memory for each individual person
3) " " Integrated Domains from various common enemy & assoc IR problem (Note treatment of this problem in Case Base Reasoning & Explan. Based Learning.)
4) Treat "OSL problems" for each domain.

Probably other problems in interpretation (≡ "Data fusion") or Understanding

I have been acting as if the only thing RM had to do was "concept acquisition" i.e. acquisition of a concept ("knowing" of it) & [having it & cond. pc's for various conditions].

Conditions

This last can be very time consuming! — But it is an essential part of living.

I want to look at it more carefully! what are its mechanics, etc.

→ Also, consider the variety of cond. probs: they can arise in various ways & be correlated in various ways, Perhaps the simplest cond. prob is a sum of "flat" Many of concs. within a Domain] to first, an ~~un~~ non-flat, uncond prob, than a True cond. prob! cond. on when each conc. should be used.

Next, ability to use concs from other domains.

.39 → In all cases, the formalism of 96.01-.12 can (should) be used, to get optimum Cond. Prob distrib.

So in Theory (96.01-.12) .39 should "Solve" problem.

→ 98.21

.01 On t. "Gen'd" 2 kinds of ~~complete~~ problistic Induction "problem"
We have/ for corpus a set of finite objects & a set of ~~partial objs~~ partially complete objects. This latter may be parts of a finite ~~or~~ or an infinite object.

Example of such a corpus: (Q, A) problems in which we are given many complete Q, A's, also many Q's w.o. A's.

Guess at soln: A priori of this corpus is (→ See 99.01 for more careful discussion.)

.07 $$\prod_{i=1}^{k}\left(\sum 2^{-|P_{ij}|}\right)$$

Note: on 99.01 we discuss t. machines that could have created this corpus. Eq. .07 must be mult by t. a prio. of those machines, and summed over all possl. such machines.

There are k (complete + incomplete) objects.
P_{ij} is t. j^{th} "minimal" pgm for the i^{th} object.
We may want to have 3 symbols: 0, 1, (S Stop). There is no "stop" state. Once t. "S" is emitted, it continues to print "S" forever. → (99.01) ← cmnt.

.12

A kind of trick TM should be able to do! We give TM a diff't problem; then we show it how to work a "related" problem & TM is able to work diff't prob. An example would be solving of cubic or quartic eqns. or various kinds of symbolic integration. (≈ [redacted] "Transfer Lrng")

.20 : 94.04 Right now, I don't really remember the heuristics behind t. "Alg word Prob Lrng." TSQ. That I did. It did use a "stack machine" which is a useful. I'd like t. machine to work thru acquiring t. concept of "value" & parsing to obtain "values", substituting "values" & recursively, etc .... t. way humans do it. (sounds good: I did a few such of work on this idea in ~1990 (search).)
→ I could start out w. a TM that already "had" t. conc. of "Value" & go on from there. In a sense, this would be giving TM some intuitions into that humans may have or acquire early in life — or by transfer lrng from other Domains.

How would this conc. be represented in TM? One way: that assoc. w. certain substrings is a "Value" & that "values" have certain (useful) properties.

.30 → 103.18

Another kind of TSQ that I've considered, but not actually done work on would be to try to get TM to go thru a TSQ. based on t. order of info & examples & problems in an ordinary human [textbook on Algebra]. At first, I was worried that certain (perhaps essential) parts of t. TSQ implied by t. text Book would involve concs. from R.W., accessable w. diffy. to TM. Now, I think that I can deal w. such concs, by writing alternative sub TSQ's or using hints or ≈ "telling" TM about those concs. T. main interest of such a TSQ is that it actually does learn useful concs & guess them useful PC's
.40 (and useful cond'l PC's). Also that one can get useful lrng, even tho all steps are not really "lrng".

92598 TM Z141 (.03)

.01 [redacted] T. idea of 96.40 is a possl. "Breakthru" in my ideas about TSQS.

→ Platforms

.03 SN on Z141 : My colony may have become unsafe.

91.40 In "Fuzzy Grammar" we have f_A, f_B & f_C w. $C \doteq AB$:

Observed freqs of A & B are $F_A + R_C$, $f_B + f_C$. AB is observed w freq. $f_C + f_A \cdot f_B$.

Given $F_A + F_C$, $f_B + R_C$, $R_C + F_A \cdot f_B$; to solve for $f_A f_B f_C$

.06 α β γ

$A + C = \alpha$ $\quad C + AB = \gamma$ $\quad \therefore C = \gamma - AB$ $\quad A - AB = \alpha - \gamma$

$B + C = \beta$ $\quad B - AB = \beta - \gamma$

$\frac{A}{1-A} \; \frac{1-B}{B} = \frac{\alpha - \gamma}{\beta - \gamma}$

$X + Z = \alpha$ $\quad Z + XY = \gamma$

$Y + Z = \beta$ $\quad \therefore Z = \gamma - XY$

$A(1-B) = \alpha - \gamma$

$B(1-A) = \beta - \gamma$

$\Rightarrow A\left(1 - \frac{\beta - \gamma}{1-A}\right) = \alpha - \gamma$

$B = \frac{\beta - \gamma}{1-A}$

$A\left(1 - A - (\beta - \gamma)\right) = (\alpha - \gamma)(1 - A)$? quadratic in A.

$X + \gamma - XY = \alpha \quad X - XY = \alpha - \gamma$

$Y + \gamma - XY = \beta \quad Y - XY = \beta - \gamma$

Black King is below Black Bishop w. King.

In previous move, it was checked by motion of W. Rook.

$X(1-Y) = \alpha - \gamma$

$Y(1-X) = \beta - \gamma$

$Y = \frac{\beta - \gamma}{1 - X}$: $X\left(1 - \frac{\beta - \gamma}{1-X}\right) = \alpha - \gamma$: $X(1 - X - (\beta + \gamma)) = (\alpha - \gamma)(1 - X)$

where $X(1 - \beta + \gamma) - X^2 = \alpha - \gamma - (\alpha - \gamma)X$

$X^2 + X(-1 + \beta - \gamma - \alpha + \gamma) + \alpha - \gamma = 0$

$X^2 + X(\beta - \alpha - 1) + \alpha - \gamma = 0$.

$X = \frac{1 + \alpha - \beta \pm \sqrt{(\beta - \alpha - 1)^2 - 4(\alpha - \gamma)}}{2}$

woops! Additional eqn:

$X + Y + Z = 1$ } This is true if A & B are
$A + B + C = 1$. } true only for two of the symbols in T.

Corpus : if there are > 2 symbols,

we [redacted] can know $A + B + C$,

but it would be < 1.

So this may be over constrained ! 4 eqns, 3 vars !

[redacted] $\alpha + \beta = 1$

What is exact value of γ ? (.06) It is the no. of times we observe AB divided by (what ?),

(Also note : $A \neq B$; symmetry analysis doesn't hold if we are looking for "1" occurrences, say,

As in Z141 (or Sc 64 part 2) we treat this as a special case.

.28 Say the grammar is [redacted] 3 symbols A, B, C, w. freqs X, Y, Z ; $X + Y + Z = 1$.

We have this Boring corpus w. N symbols: (B) occurs (NZ) times, (A) NX times.

We plug in $C \to AB$. The corpus now contains $N + NZ$ symbols.

$N(X+Z)$ are A ; $N(Y+Z)$ are B, AB occurs $2N$ times as a result of $C \to AB$.

AB occurs $N(XY)$ times as a result of random adjacency of X & Y.

So in the entire corpus of $N + 2N$ symbols, A occurs $N(X+Z)$ times | B occurs $N(Y+Z)$ times,

AB occurs $NZ + NXY$ times.

freq. of A is $\frac{N(X+Z)}{N(1+Z)} = \frac{X+Z}{1+Z}$

" " B is $\frac{Y+Z}{1+Z}$

" " AB is $\frac{Z + XY}{1+Z}$ | i.e. reading of "freq of AB". This is ratio of no. of occurrences of "AB"

in corpus, to no of symbols $(NX + NY + 2NZ = N(1+Z))$ in corpus

.01 So given $\alpha\left(=\frac{x+z}{1+z}\right)$ $\beta\left(=\frac{y+z}{1+z}\right)$ $\gamma\left(=\frac{z+xy}{1+z}\right)$ and: $x+y+z=1$.

So given 4 equs. to find 3 vars! — seems to be something wrong!

This is certainly not a "deep problem", but it's annoying! 97.28ff seems to give .01 !

perhaps $\frac{y+z}{1+z}$ is a fn. of $\frac{x+z}{1+z}$? $\frac{x+z}{1+z}=\frac{1-y}{2-x-y}$.

$\frac{y+z}{1+z}+\frac{x+z}{1+z}=\frac{x+y+z+z}{1+z}=1$ $\frac{y+z}{1+z}=\frac{1-x}{2-x-y}$ | $x+y+z+z=1+z$

So that solves it: $\alpha+\beta=1$. $1+1-x-y$

$z=1-x-y$

$\gamma-\alpha=(.01)$: $\frac{xy-x}{1+z}=\frac{x(y-1)}{2-x-y}$

$\frac{1-y}{1+z}=\alpha$ ∴ $x=\frac{\alpha-\gamma}{\alpha}$?

$\alpha=\frac{1-y}{1+z}$ $\alpha-\gamma=\frac{x-xy}{1+z}=\frac{x(1-y)}{1+z}$ ∴ $x=\frac{\alpha-\gamma}{\alpha}=1-\frac{\gamma}{\alpha}$!

So $x=1-\frac{\gamma}{\alpha}$; $y=1-\frac{\gamma}{\alpha}$: $z=1-x-y=$ $y=1-\frac{\gamma}{\beta}$.

$z=1-1+\frac{\gamma}{\alpha}-1+\frac{\gamma}{\beta}=\frac{\gamma}{\alpha}+\frac{\gamma}{\beta}-1=\gamma\left(\frac{1}{\alpha}+\frac{1}{\beta}\right)-1$ $x=1-\frac{z+xy}{x+z}$

$\frac{x+z-z-xy}{x+z}=$

$\frac{x-xy}{x+z}=\frac{x(1-y)}{1-y}$

$=x$. → 109.01 on Z141

Machine Lrng is t. goals of A.I. where are we going & where can we go.

→ Name for TALK! "A General Purpose Learning Machine" title of paper.

.20 (SN) Expo: T. new TM will be called "RM" Renaissance Machine (Man, Person).
RR (" ES

After Dark Ages (1960–1980) t. emergence of the Renaissance (ML).

.21: 95.40! So, in theory, (96.01–.12) 95.40 should "solve" t. problem! [

.22 → A possl. remaining problems: Just how to get t. cpd's of h.c. sreats to use for OZ probs. How can 96.01–.12 be applied to this problem? And, even if it could: how does one get new h.c. methods and modify t. cpd of those h.c. methods w.r.t. a given) problem?

(It may be that we usually don't want to use Lsrch for OZ probs! That)!? using (SGA) is usually better & easier to implement.

There is one kind of OZ problem on which I normally don't do "Lsrch OZ": the finding short codes for a corpus. A common way is to look for concs that reduce code length, recode t. corpus using this conc., then repeat ad infinitum. I think this was t. method I use in all 3 TSQ's that I "wrote".

It seems that t. "Lsrch OZ" would not be a true optimum. — unless t. h.c. method choice was conditional on not only known $[Q_i, G_i]$ corpus, but on t. traces obtained generating the new Q_i and/or on t. traces of t. G evaln. (if this evaln. trace was available: sometimes it is partly available) | Note: p.875 99.36–38: what is this "expanding pc. t. goal of"??

BUT: Most recently, I had t. idea that one could go pretty far by using a Good fixed set of h.c. methods. That improving these methods was a "Heuristic" & was not very often done —

.01: 96.12: T. way we could have gotten t. corpus, so its pc was 96.07:
We initialize t. machine. We insert random input. Object 1 is given as output.
We reinitialize " " " " 2 " " " "
" " " 3 ect.
"Given as output" means: for finite objects, machine stops after printing object.
for prefixes of finite or infinite objects we stop t. machine when it prints that prefix, & reinitialize it for next object (or prefix).

The machine itself: It is a 2 input machine. For input #1, it takes random input and has "no output". At a certain point it stops and requests 2 random string for input #2 — which can have output — it can have a "stop symbol for output".
In fact, both inputs can have a stop symbol for output: hvr, on input #1, there is no output symbol.

Perhaps look at Chaitin's treatment of this 2 input machine in his (?) IT 1975 paper.
It may be clearer. Also, see if Wallace's 2 part input for a 1 input machine can be made to work.
Also, consider sense in which input #1 can describe any machine (so it is a "universal machine").

Note that Wallace uses "2 part machines" for prefixes of infinite objects. If I were to use Wallace's approach in t. forgoing, I would have a "3 part code" for prefixes of infinite objects:
The first part would be as in t. forgoing. The (second part) which describes a cpm would be Wallace's "first part",
& the (third part) (describes the corpus in terms of the cpm) would be W's second part.

In my formulation, I can have partial recursive functions (→ partially computed functs),
I think Wallace cannot. (or he simply doesn't try to get them).

Anyway, t. "first part" may describe a machine. Since this machine can be universal, it can have "halting problems".

T. total a priori prob of t. corpus of 96.01 is t. total of

.23 $$\sum_{\ell}\left[\text{a priori of machine }\ell \left(\prod_{i=1}^{K}\left(\sum 2^{-|P_{ij\ell}|}\right)\right)\right]$$ so $2^{-|\ell|}$ can go here.

$P_{ij\ell}$ is t. j^{th} "minimal pcm" for t. i^{th} "object" w.r.t. machine ℓ.
In .23 instead of "a priori of machine ℓ" I can use $2^{-(D_\ell)}$, where D_ℓ is t. descn of t. ℓ^{th} machine. → Actually, we can get $D_\ell = \ell$, since we will be describing all possl. machines.

.28 $$\sum_{\ell}{}^{'}\left[2^{-|\ell|}\left(\prod_{i=1}^{K}\left(\sum 2^{-|P_{ij\ell}|}\right)\right)\right]$$

ℓ is t. descn. of t. ℓ^{th} possl machine. If this input pcm, ℓ, does not cause t. machine to have an output & eventually stop, then we don't include it in t. sum: Hence the "'" in t. Σ. [Σ']

.35 So .23 is essentially what I want: .28 is an attempt to formalize it, that looks reasonable. → (04.01)

.36 (.28), hvr. seems incomplete: It doesn't deal w. OZ probs at all: (98.22): One diffy w.t. OZ tech method was that it wasn't clear as to what t. "guiding pc" was t. prob'ty of! I guess
.38 I'd like it to be t. (condl.) prob'ty that each h.c. method was t. best way of solving t. given problem (condition).
.39 Another view: we have the set of past [OZ pab's, HC method j, CC i_j, G_{ij}]
That gives t. history of various OZ problems, how hy G we've gotten in a given CC, in t. past.
From this, one has an idea of G to be expected after a given time on a new problem for each of t. HC methods. One then spends various amts of time on each of t. HC methods. →

100.01

.01:99.40: **Could** this be an applicn of t. "[WTWON] What to work on Next" soln? Certainly "not exactly"! — 40 x 6 = 240" = 20'

But perhaps I can use t. General technique of some how modify soln of WTWON. 800 ft²!

T. WTWON prob. is ~ to OZ in that t. HC methods are "in parallel". T. WTWON soln is interesting in that ① it is optimum ② it does jump from one cand to another — back & forth — which would seem unreasonable.

Perhaps a ~ soln. would be optimum for [illegible] OZ problems.

In WTWON, when we had several continuous distributions in ||, one kind of soln. was a time share in which t. share depended on t. slope. In a ~ way one could do time share on t. H.C. methods, but it's unclear as to what "slope" to use!

.07 (SN) In Normal OZ Lsrch; T. HC methods used are indip., so t. set of trials made by one method .08 is not available to t. other H.C. methods! [This is a tremendous waste!] [Tho, it may well be that most H.C. methods don't use much of t. past corpus — they may use 3 or 10 of t. most recent trials. Sim. Anneal 10 uses only t. last trial! → (.32) ← Hvr. See (.32)!

"Slope" mite be ↑ in G per unit time of "Best cand. thus far" as done by "HC method" (whatever that means!). Given a set of [O_i, G_i] pairs, each HC method should propose a "next trial", w. a cc assoc. w. that trial: Hvr, there are an ∞ of H.C.M's, so it's not practical to try them all. Def HCM

Note: t. form. may not apply to sim. anneal: t. trial depends on t. "Temperature" at that time.

(SN) Another Approach to optzn: Say t. Gores of all OZ probs are known: Then one can ask for a OZ method that will give best expected ΔG in t. available time.

Another possy! From t. corpus of 99.39: one can estimate which of a set of H.C.ms will be best for a gen. problem. — & this will vary as one continues on one or more of t. HCM's. The directly relevant sub corpus of 99.39 would be those in which the G_{ij} were max for the C_{ij} & HCm_j & OZ prob. — but doesn't this have [diffy. .07-.08]? (i.e. T. sub corpus has t. C_i, HCm_i, OZ p. Trial in which t. G was max; but this subcorpus need not contain t. global G attained.

Could **SGA** be a "Not Bad" soln. to t. General OZ problem? T. [diffy] appears to be in an "easily computable path" for t. model. We can restrict t. models we consider to be of this class. ●

Wrt. OZ problems: While finding short codes is an "OZ" problem, it does seem to be diff't from most, in that it seems "well-solved" by INV Lsrch! A possl. uncertainty in its being optimum — Tho t. heuristics needed, will derive from parts of t. corpus other than that being coded; (i.e. not searches for short codes) — T. relevant parts mite involve OZ probs of t. "regular" kind

Which would be Unfortunate, since it looks like I don't understand such probs very well!

32: (.10) → Well, t. method of [using many hcm's & have them share a common "corpus of known [Set of cands or known G]] is a new HCm, & should be included in t. Lsrch as such.

Perhaps one could Modify t. proof of t. "OZ Lsrch is optimum" Thm, so that t. pc's of each Hcm (Hill Climbing Method) were conditional upon t. set of known [O_i, G_i] thus far. HCM Def

It would seem that t. hmc's would then become interdependent on their order of trials.

→ 102.18
105.01

A review of t (RM system)'s ins & outs:

(1) See 95.25 for a prelim list of unsolved probs.

(2) See 92.02 ff & 93.04 ff are listings of a few classes of probs ("inputs") for RM

(3) I'd like General OZ probs to be solvable by t. system, but Regular Search for OZ probs seems inefficient (99.36 to 100.40 at best); & I'm not sure how to get t. PD Mat "guides" t. search OZ soln. & make it part of RM's "Grand CPD".

4) Just what form do these CPD's take? One possible form is a stoch operator. — two parts, one just trys to find Good stochastic operators; One that Maximally compresses t. corpus.

.09 (SN) I could just start off w. a TM that could work mainly INV problems plus short code discovery

It would be a useful TM, yet it couldn't solve General OZ probs — which is to say Most problems. It could solve OZ problems approximately, using INV. Search & GA

[redacted] there are probly other approx. methods it could use! (See 99.36 — 100.40 for some ideas, perhaps) Also, perhaps SGA could be implemented. → 102.20-2

.7 It can usefully solve QA probs.

T. "Analog problems" of 93.36 ff.... seem to all be OZ probs (except for a singleton "cerebellum" see 93.36 102.20-2)

T. "cerebellum" idea of 93.36 ... of modeling an analog I/O corpus of analog I/O pairs, seems to be a full-scale draft OZ problem. [No! — see 102.20-23!]

Def CRM → So perhaps draw up a more detailed dem of t. (.09) device: (Crippled Restricted) RM (CRM / cripRM)

This CRM can do lots of things; it will give me a better idea as to what t. main Remaining problems are! Meanwhile, I could work on various possible solns to t. OZ problem — many types of Approach.

SN on [redacted] HCM's (attempt to map OZ prob into short-code-finding prob & maybe OZ probs). OZ probs: Given (Black box) M(x) (string).

To find x → M(x) in Max, (in time ↑). Consider as a coding problem: [redacted]

for each Make a printout of M: so natl M by [redacted] a large expositive real & itemize (round off) → M'(x)

A code for x is that So M'(x) is an integer. For each value of M'(x) there will be a certain no. (usually finite) of strings, x: If we give Sucin. → probby we can code them call it Card(M'(x))

Using M'(x) & [redacted]. So, we can code (many) x's by [M'(x), n(x)] where n(x) is a no. below [redacted] ϕ ÷ card ∑ M'(x)]. Short codes for x may involve small card ∑ M'(x)], and actual (maximize maxima) can involve small Card [M'(x)], but other x can also have small card ∑ M'(x)].

If we make time limit on M(x).... (This doesn't seem to help.)

I'm thinking of t. SOM problem, in which a kind of optza problem was expressable as a short-code finding problem

(SN) It may be that true optimize of TM behavior (including all cr considerations) is essentially an OZ prob., & not always a "short-code finding" prob. An old problem that I (perhaps) never solved!

Given a postset of [Pr: (problem), Soln: [redacted], cc] to induce an optimum CPD for all problems, ⇒ t. [redacted] ≤ cc of soln - of all past problems is minimal. Well, by assigning [redacted] epsilon probby ≥ 1 t. known solns of all known problems, we get a ≥ min cc. This may correspond to "remembering solns" of problems. If we get a new prob. ≡ out of t. old solved ones, we should

.01 try its past solns first. However, for a new novel prob, this will not work — [redacted]

For a large class of problems, obtaining a CPD from the corpus [pre, soln] would be adequate — then search would be the best way to [redacted]. continue.

I think there are cases in which we want to bias the cpd in ways to reduce expected cc of soln, but they are not the same as simply finding the cpd of the soln. in view of the problem — rather they somehow involve the cc's more directly. "Quick [redacted] Abort" may be one such trick —

We find a way to quit a test early so as to ↓ cc of testing. Modifying cpd [redacted] of solns. [redacted] cannot implement this heur.

.09 Interaction w. world (say a reinforcement problem) is an OZ problem.

T. General TM2 problem (to optimize TM1) is an OZ prob.

[redacted] CIAP(G): we may be able to use fact (.09) to construct a fairly intelligent TM, that is not "concrete", & is unlikely to give problems assoc. w. "awareness of outside world".

18! LOT.40 100.10 Another view of OZ probs: that TM might learn to take OZ probs & "change their representation" so they are solvable by a standard set of hem's

.20 Many OZ probs: Given corpus [tiee probi, trial solnij, Gij], TM can form a good CPD for G of cand. solnij's & probi. finding them may not be easy, but [redacted]. This would be a useful

.22 function if eval'n of G is expensive; So this soln. could implement "T. Cerebellum" problems

.23 of 93.36; However, using the "cerebellum" for analog [I, O] is usually a OZ problem; In a pinch, 101.19 20.4

.24 we can use INV Lisch to approx. a soln (or other approx. methods). → 105.01

25 101.17 So startout using the {non OZ (corpi, probs)} of 92.02–93.04: see how far we get w. the analog probs of (9405–95.05?) say, using Inv Lisch for the OZ part (or some other approx of OZ).

So basically, this is the idea: (concerns): In earlier TSQ's, I started out w. a set of primitive (objects, abstr concs.) They all had some pc: & I would use all possible combinations of them in pc order (assuming same cc for each trial). When I found a conc that was a soln, I put it in storage. I would either use storage as a prim conc. and give it pc = pc then .. to all concs w. similar storage, or simply add to know concs to the set of prim concs (w = pc), & then

Go to next problem: loop to . There may have been some differences in pc's dependent on freq. reuse (essentially Bernoulli probs).

NOW what I want to do is make the pc's of the concs. distinct for each problem/application of these concs . . . .

. . . . a term "conditional pc".

If a conc. appears for the first time, as the soln of a problem; then the next use of that conc. will amount to ("OSL"): which is normally treated in a special way. My impression is that in OSL, if a conc accessed only once is dealt with. Then its pc of occurring again, is fairly low.

However, that the conc. was the soln of a prob. may give it considerably more pc.

Perhaps SN If we actually save TM solns to problems (along w. the problems), it will take much less time for TM to integrate the needed concs. (i.e. find relevant concepts), than if it had to also find the solns.

→ 103.01

0.1.98 TM   Unordered set of objects Thm: proof: for unordered set

.01: In the "proof" of the conv. Thm (Sol 78 Thm 3 √), I used the 3 symbols: — (the continuation of this idea seems n.g. .... to stop). — Hvr., the more General Thm of 99.33 : .28 could use this 3 symbol proof: The third (stop) symbol doesn't seem like an appropriate termination for the "partial objects" i.e. "prefixes of infinite seqs".

.05: 102.40: In general, at each point in a "trace", the "next move" is governed by condl. proby's involving The problem being solved, & the previous trace within that problem. Later on, the condl. proby will have more input conditions possl. — so it can draw info from other Domains.

— Hvr., we want to start out w. uncondl. pc's; then gradually add more conds, as ssz ↑ & ∴ as more conds become warranted. [To do this, we merely augment the corpus w. the new subcorpus of "conditions"]

SN 3 impt. deficiencies of current model of RM:

1) Ability to do OZ probs, but only in an "approx." way (non optimum)

2) (Possibly a result of ①) Not all types of heurs can be implemented, since only those that modify the cpd are possl. (e.g. "Quick Abort" types of heurs are impossl. to implement)

3) IR Lsch itself (either in T≤2T or Time shared Mode) is non-optimum since it doesn't take advantage of interactions ("cross-info") betw. Trials. — Ideally, this info should (at least) change the shape of the pc curve.

.18: 96.30: One way to implement this "Value" idea is to give examples in the form: [V(10+3), 13]   TM must then extrapolate the V( ) function to any strings! (Hvr, V will not be defined for most strings); Perhaps it should first learn some "type" ideas: that "10" is a "number" — that "V( )" is a "number".

To make it easier, label the sub/corpus "V" & have examples be like 10+3, 13

(This bears a close resemblance to the old corpus! 10+3 = 13, (!☺).

The difrnce is, that TM can know about the comma: that it separates the 2 parts of the sub/corpus in the "Q : A"

— So TM would know that it had to express the string "13" as a funct. of the string "10+3".

Having the idea of "Recursion" could be useful — or have it as an available primitive Functional in the universal "Functional Lang". In the present case: One way to do it: V = "look for the first section of the string in the domain of V: apply it & rewrite the string w. this value" Do V until it is no longer possl. ← this may be an adequate recursive defn.

i.e. V = G(V, A) ← a functional ∫ some functn. Recursive defns. when taken literally, are usually inefficient in execution. We can make special purpose functions to evaluate them efficiently, Ideally, these Functions should be very useful to express other impt. concs (functs) as well.

C S Pierce on proby PS(Monthly?) 1878

So maybe look at "APL" or "J" languages,

Another approach would be to start w. less clever "prim. concepts", & lots of corpus & lots of cc.

← Direction of Jargon.

.01: 99.35 [SN] On the formalism of 9923, .28: Wallace may actually want to do this as a 3 part code:

.02 1. Self del. code common to all objects & partial objects & / the seqs.

.03 2. for each of the (objects of all) in 1., a self del. code giving a P.D. (105.27: use of this part should be zero!)

03 98 .04 3. for " " " " " " ; a self del. code giving the prob of the complete object, ?20

1) A 2mg ... or a (2 / ...) self del. code for the partial object.

1115P

0498

215A 3. .05

4A 1¾

545A 1¾

615A 2mg ...

.08

.09 [SN] On "Lsrch" for a growing corpus: The idea I had was that one would start w. a short corpus, then using Lsrch, find some conc. (sub) defns. that would compress it. Then using the old primitives and new defns., use Lsrch to code an augmented corpus, & find new defns. loop to α

→ Some problems here: how much α is to be used for each corpus Augmentation?

[Note that in general, one would try to recode the entire corpus, not just the augmentn. — The larger corpus may have enuf ssz to find some new low level

.13 defns. to help recode earlier parts of the corpus] → 105.24

[SN] On functional langs: I will want various simple (prime) functions:

.20 Say x+y, --- etc., integers, maybe sin, cos, exp, ln. etc:

Then functionals: To implement: x,y,z are inputs
Do untill $G(x,y,z) > 0$
$(x,y,z) = \vec{F}(x,y,z)$
loop

the outputs x, y, z after $\vec{F}$ has been applied a certain no. of times (x and/or y or z can be a constant; $\vec{F}(x\,y\,z)$ may not allways change x or y or z)

So this function is defined by the word "untill" & the 2 fncs $G(x,y,z)$, $\vec{F}(x,y,z)$ and ">".

Do Untill (cond.) 20. ($G(x,y,z) \leq 0$) inputs x, y, z → $f(x,y,z)$. loop

To define n!: Input n: output x = n!

.28 do untill $y - n = 0$ start: y=0, x=1, ;
$y = y+1$
$x = x \cdot y$

So the function x! is defined by: initial y, x, n; $y_{n+1} = f_1(x_n, y_n)$; $x_{n+1} = f_2(x_n, y_n)$; $G(x,y) = 0$ is criterion for end of loop.

[y=0, x=1, n; $y_{n+1} = y_n+1$; $x_{n+1} = x_n \cdot y_{n+1}$; $y_n - n = 0$

This is one way to define n!: it seems to use many params

A less complex defn: 0! = 1; $x! = x \cdot (x-1)!$ standard recursive form.

} we can have a compiler that xfms this defn into the more "computable" defn. of .28

.01 | :102.24 / 100.40 | (SN) Interesting Q! Given a corpus $[O_i, G(O_i)]$ (scalar) a data set in which we want to find O_j ∋ $G(O_j)$ is as large as possl. w. t. given CB (compu. Bnd.) (≈ t. OZ problem).

Is it meaningful to ask for $P(O_j, CB)$ — t. proby that O_j is t. value for which $G(O_j)$ is max: [somehow CB tiesin].

Well, $[O_i, G(O_i)]$ implies a certain $P(O_j, G)$ p.d. for all O_j, G values.

.07 We can ask for t. proby that any one O_j has max G. This would seem to be a clearly defined [hard to compute] function if t. $P(O_j, G)$'s for t. various O_j's are indep. If they are not indep.

.09 it may still be defined but even harder to compute function of O_j.

If so, then this could perhaps tell how OZ problems can be added to t. corpus of 104.01 – 08 (i 99.23, 28).

Even if t. P.D. of .07 – .09 is very hard to compute, it is then perhaps meaningful to ask for t. best estimate we can get in 10 sec, say.

If th. forg. discn. is correct, this will help a lot in t. discn. of OZ probs in 100.07 ff. 100.07 – .40 suggests strongly that I don't really understand t. OZ problem: That I don't really know what t. C.P.D. for t. H.c.m. is. I have no defn. for it. —

105.01 ff suggests a defn., tho it's not at all clear how one goes about approximating it!

20 → Actually, t. discn of .01 ff isn't exactly rite for OZ: for OZ we have several H.c.m's H_i. We have $[H_i, O_{ij}, G(O_{ij})]_c$: $[O_{ij}]$ are t. set of O_j ∋ gen. by H_i in CB = c. Due to diffy 100.07 $[O_{ij}]$ is not clearly defined: but anyway, from that corpus we're supposed to get a C.P.D. for H_i getting t. best G in CB = c.

.23 { — All of this is w.r.t. problem Pr_k. So these are cundl. probys that H_i will be best for Pr_i in CB = C. → (108.09)

"IR" probs — on "IR" prob: 93.12, .33

abcdef

.24 | 104.18 [SN] Fort. Genzn of 99.23 – .24! I need t. analog of $\frac{P'}{P} > 2^{-K(n)}$. [n = total corpus size.] → (See 106.23)

To start off, use a generator of t. corpus w. a finite dcm: $\frac{P'}{P} > 2^{-K}$.

This amounts to a finite dcrn of parts 1) & 2) of 104.02 – .03.

.27 If we only have finite objects in t. corpus, this suggests that bcoz of 2) (104.03) should

.28 be zero! — On t. other hand, if we have elements of t. corpus that are prefixes

.29 of infinite seqs., then it seems bc of 104.03 should be > 0 !

For finite object corpus: clearly 104.03 must be zero, if t. stoch. lang. dcrn. is finite (unusual, but possl.)

.31 For corpus that is prefixes of ≥ 2 inf seqs, that have some, but not all, common concs., bc of 104.03 should be > 0!

Going back to 99.23 – .24: T. way we are to use these formulae for extrapolation;

Say we have t. prefix of some object (finite or infinite). We want t. value of pc of various contins of it. For each contin (which is a prefix of some object), we want t. total aprip of all objects (plus t. rest of t. corpus) that have that prefix.

Consider cases: 1) Corpus contains several finite objects & one or more prefix of objects, that [redacted] of objects that have those prefixes. (Those are "cylinders"). (Also consider any finite dcrn for corpus.) We want the total Σ pc of all corpi [redacted]

may be finite or infinite.

0.5.98 TM Mixed Corpus Problem

This brings up an interesting Mathical object: say the corpus consists of prefixes of two infinite sequences. We want the product of the pc's of 2 "cylinders" — This is supposed to be some kind of sum of many pc of many doubly infinite sequences — many of which have "zero" pc. So we want a measure or sum; measure over some space. (.29)

.04

I think the idea is that 99.23, .28/.31 are correct — but what about 105.28 - .29 | 105.31 seems to be the "Big Q".

In some stock ranges (w. ass ≡ finite strings), it is possible that the generator pauses for a while before generating certain of the strings; could these pauses be interpreted as |0 4.03 (≡ 2.).

Analysis of 105.31: the "1." component: commencements and depths of sin, cosine, ln, exp, etc. The "2." components are the individual parameters in terms of sin ... exp, plus various constants (params).

So sequence a): $X_{n+1} = \sin\left(\sum_{i=0}^{k} a_i X_{n-i} + \text{noise}_a\right)$. Noises a & b are uncorrelated,

b): $Y_{n+1} = \sin\left(\sum_{i=0}^{k} b_i Y_{n-i} + \text{noise}_b\right)$

(instead of "sin", use a func. w. clearly several params.

.17 One of descrb. trans 2 seqs as a single vector sequence, w. vector components, X & Y,

In the vector representations, the X & Y sequences need not be of same length. — The way this is done: the pc of the partial vector seq. is the measure of all vector sequences that have both the "x" prefix & the "y" prefix [the 2 prefixes being of distinct lengths]

This vector representation can clearly be extended to any finite no. of sequences.

Certain of the "vector components" could represent bits of finite (objects / strings) — so this could be a "Unified soln." to the "Mixed Corpus" problem of 96.01

.23: 105.24 The no. "n" in $\frac{p'}{p} > 2^{-K(n)}$ (105.24) is simply the no. of bits in the known corpus.

.24 : .04 : The way this is done: We have 2 sets of code strings of int. length. To get the product of their measures; Take the measure of the first set of strings, all which each of the strings is given wt = measure of second set of strings.

.25

.26

Equivalently, if $[s_i^1]$ & $[s_j^2]$ are the 2 sets of strings, we want the measure of $[s_i^1, s_j^2]$.

and I think this .24 - .26.

A way to criticize the "vector soln." of .17 ff: Say the corpus is only a finite set of strings. — Does the soln. correspond to our vaguest "rather reasonable" soln to the "unordered sets of objects" problem?

33 → (1) What kind of sub-corpus is OK data? Even if we only have one or more OK pairs in the corpus — this is still an "open problem".

34 (2) In 104.05-.08 (the Fibonacci) 1 & 2 can be combined w.r. medium of total info — so the alternative vector representation of (.17) plays now diffly — it's only an "Alternative Representation" which is always good!

The vector representation does combine 1 & 2. — so we only need 1 2 input machine.

However, I do want to write up a detailed description of the (preferred) soln to the "Mixed Corpus" problem

If it works, the next big breakthrough problem is (.33) (or Jorpus).

 Mixd corpus problem

A diffy w. 106.34 ff: T. "mitial Machine "1" is supposed to start on t. various sub-corpi: each w. its own now, uncondl. p.c. For inf. sequs., t. p.c. of t. entire seq. is "φ", but for finite objects, it is > 0.

Perhaps for each inf seq., we could have a "pc of starting" — as t.

To work on this problem: Consider just out t. system works: T. diff. Post'n Gives for predns.

In t. case of pure unorded objects for corpus: Consider a stoch CFG as a p.d. dern: We start w. dern. of t. grammar, then we have various problstc choices of Non-Terminals: Then, at last step, (perhaps) t. terminals (final ~~object~~ symbols) are generated.

It is not t. "sequencial" Generation of most inf. sequs: it seems more like Cover's "Extension (complexity ~~prob~~)" — not exactly: for cf langs, — say one wanted to p.d. of various possl. continns of a "partial object": One can, normally (I think) devise a cfg (or/set of cfg's) to generate t. "rest of t. object". | Arr Boston ~2 hrs So ~ Midnite London time ~6 pm Boston time.

Well, ~~consider t. folg~~ Say t. corpus consisted of A & B (2 finite objects) & C & D (2 inf. sequs). A possy is that C & D have t. same "first part", & that they are simply different "stochasitic variations" from t. same ensemble. | .40 8

Another possy is that they have completely different "first parts"'s. A Mixd possy is that they have different 'parts'''s. but some "shared" part i's. | 50

I think I have a clear idea as to what "INFO" is required for a "Mixd" corpus. — So t.
.20 Machinery of Generation of t. corpus should be (trivial / easy).

T. vector formalism of 106.17 seems to deal well w. t. diffy of infinite sequs being infinite (so one can't "do one seq. after t. other")

.23 Another view of t. "Mixd corpus": we have a machine that generates t. corpus in t. folg way: Random input, unidirectional I/o tapes, 1 wk. tape. (α) One pt. in input: As soon as one of t. corpus objects has been generated (it can be a finite object or prefix of a finite or inf. object) We reset t. corpus & loop to (α). Under these circumstances, there is a certain proby
.27 that our corpus will have been generated.
.23–.27 is wrong. We abort a trial as soon as its output is not consis w. any of t. corpus objects, and declare it a "failure". A trial, hvr, may get into an infinite (but unrecoverable / inscrutable loop) — in which case we say t. corpus was not generated = "failure". (t. total proby of all objects in t. corpus is certainly < 1 (!)).
.30 If we generate an object twice, but it appears only once in t. corpus — That trial is a "failure".

.23 – .30 may be o.k.! Is it consis w. t. "vector" view of 106.17? (t. "vector view" may be equivalent to an Orderd set of objects.

.23 – .30 is for a BAG corpus.

.23 – .30 Suggest a defn for t. K(n) of 106.23, 105.24: We have a particular Tmc., M and it generates t. corpus (using rules .23 – .30) w. proby P. Say, using another mc, M', t. proby of generating t. corpus is P'. Then $\frac{P'}{P} < 2^{-k}$: k being t. booster dern. of M wrt M'. How to deal w. "K(n)" is not yet clear. I should first get t. continuous params straight, befor working on K(n): Just how does t. dimensionality enter?

"Mixd Corpus" "T. conv. Corrolary"

.01: I'd like a formalism for Mxd. Corpus that enabld ≈ Sol 78 Thrm 3 (or ~) to be used to [redacted] find Convergence Rate..

Use 3 symbol formalism. One stop (actually reset) symbol. For inf. sequs., we still have "the reset" Symbol: Any extension of the desired prefix, followed by the reset symbol, is regarded as a legit

.04 code for the corpus. This is ≡ Cover's "Extension Probability".

.05 (SN) The Extensibility "Extension Proby" doesn't allow finite codes of infinite seqes (like $1^{(\infty)}$ has a simple finite code)

.06 It may give ≈ same proby values as ALP. In Sol 78, I had the impression. That these mite be ≈,

.07 So: .01 – .04 may solve the problem, including Conv. rate. but w. diffy of .05 – .06 (may be not so impt.) → (110.01)

.09: 105.23 So next Big Problem is [OZ probs].

Say we've worked a lot of OZ probs using OZ Lsrch, so we have a lot of data on various HCM's,

.11 Probs, ~~CC's~~ CC's & G's. We have the ~~data~~ corpus [redacted] [$prob_i$, HCM_j, cc_{ij}, G_{ij}]

12 ~~~~ Then, given a new problem ~~~~ $Prob_0$ & a $CB = cc_0$, to find pc that HCM_j gives Max G in $cc = cc_0$.

Actually, we're not so much interested in the G values but in the ordering of various Hcm's

For previous dozen of OZ ~~~~ See 99.36 – 100.40; 102.18 – .24; 105.01 – 23

A Serious Criticism of regular OZ Lsrch is 100.07 (doesn't use all known [O_i; $G(O_i)$] info ∋ (No correlations betw. HCMs.) This can be fixed in a non-A.H. way By 100.32 100.07 diffy ~~99.36ff~~

20 But even ~~~~ w.o. that Criticism! I still don't know how to get P (Hcm; give max G on Prob w $CB = cc_k$) probability

105.01 – .23 looks like a serious step toward "Soln". Hvr. note 105.20

105.01 – .23 suggests that from the corpus of (.11), we can obtain (.12) — Actually we don't need the P.D. obtainable for CB=cc, but a much less good one, obtaind w. $CB = cc_2$ – The what cc_2 to use is unclear: The problem of obtaining $\binom{.20}{.12}$ looks like a TM_2 problem.

What $\binom{.20}{.12}$ is: we want a stoch operator w. ~~~~ input ($Problem_i$ (OZ), $CB = cc_j$, HCM_k) output proby that HCM_k will give best G ~~~~ w. $CB = cc_j$ for $problem_i$. (or: using only first 2 inputs, output is a pd on Hcm's HCM_j or a MonteCarlo output w. proper probys for each Hcm_j name, "J".

→ A main idea is that w. a reasonably large Corpus, we should be able to usually tell which Hcm to use w. a given (OZ) problem & CB.

If we really did good induction, The method should be able to ("Create" extrapolate) new Hcm's ~~~~ to fit a new problem on the basis of the corpus of .11. (Also Note 100.07 has suggested soln. 100.32)

→ [0798] The [OZ lsrch] prob. is beginning to look like the optzn problem in SM!

The diffy of 100.07 is effectively solvd by (100.32). So OZ lsrch may still be "optimum",

→ (117.01)

.01: 98.20! So anyway we have to consider all possl parsings for th data C=AB; Also not only th most likely no. of "C" occurrences. (See 141.01 for a good way to understand this.)

Go.03

If we use C=AB for prediction, it may not be necy to explicitely consider all possl. parsings. — (except to know how many there are), so we can suitable fr PC of this set of codes).

.04! 91.40 For loop closure (in Z141→ CPG): In my idea about Functional Laws, I just did combinations of rules. That were lower inst. trees. Like

Hrr., to get recursion, some of th lower objects are functionals that create recursive functs. from "non-recursive" & recursive functs.

"Loop operators" are functionals of this sort. — So perhaps it would be possl. to discover loops (for a CPG diagy) using these Methods. !

.10 In fact it had better be possl to discr loops this way! Because my idea of using Universal functional laws for concl discovery, is critically dependent on this idea!

Hrr. While .10 may be true, we may need hours to make it practical: Such hours could do statistical tests whose results would suggest which loops to try. These tests would give Condl. probys to th various loop trials. (Nothing Magic .... just usual way of doing it!).

Wolff may have some nice heur methods for ngram data chain & loop closure.

Re: Wolff's th soln of th Z141 problem! So far, my impression is that th usual way I expect to find regys in data is by "synthesis" so to speak. I find simple regularities & try combns. of them & th primitives, in attempts to find more complex regularities.

Does this obviate th Z141 (cent) problem? Superficially - NO! —

Say we are looking for solns to a problem via SGA: We have a lot of cands w. assoc G_i. One way is to make grammars for sets of cands at various G levels. The Z141 grammar is th simplest kind. It tells us to look for combinations (subtrees) (≡ subfunctions) that occur unusually often in th corpus. Th analysis of inductn over unordered sets, is th "Mord corpus" problem, enables one to properly assign probys of defns. to these corpi. E.g. if A & B are functions and AB is a combinn. of them, Then we look th th freqys of A, B & AB in th entire corpus of cands. with th desired G range. The regys of G can overlap, in order to get SSZ for each corpus. When we get a Grammar for each corpus (for each G range) we can then try to make a general Grammar for all values of G. We may note that th probty of certain functs (&) as G↑ — so we can extrapolate to big G.

→ 141.01

0.10.98 TM Mixed corpus problem.

.01 : 108.07: A more detailed discn of 108.01–.04 w. ideas of how proof will work.

We start w. an uncondl. p.d.; It induces a pd. on all finite objects & all infinite strings.
In t. latter case, it gives a measure on the point set that represents infinite strings.
Cany pts on 0, 1 interval represents an inf. string (or 2 inf. strings for terminating binary nos) ($.1^{(\infty)}$ or $.01^{(\infty)}$)

.05 Each such measure region can be represented exactly by a Trmc

[ΣN T. proof of .05 must be easily reconstructed, using the ideas in Arithmetic Coding. It is probably identical to t proof I referenced (By Cover & ...) in 5078.]

.08 The way the Trmc works: We put random input into t. Trmc (uni dir I/O, Bidirec. work tapes)
T. probty that it will output a particular finite object & stop, is t. pc of that object
" " " " " " " " " string w. a prefix ≡ that of one of t. prefixes of such inf.
.10 string that is in t. corpus; is t. probty of that prefix.

Since t. corpus elements are indep, we reset t. machine after each trial (successful or not).

[We have a time limit T, to tell if a trial will give any thing in t. corpus as output [or not], since t. Trmc can take arb time w. its calcns. ..... T. term is for $T \to \infty$]

.18 Because of this independence, t. pc of t. corpus ≡ product of pc's of t. objects in t. corpus.

So, every Trmc induces a pc on t. corpus (possibly a pc. of ϕ).

.20 Th. resultant pc of t. corpus is $P_{corpus} \to \sum_i P_{M_i} \cdot P_i$: Here M_i is t. i^{th} poss'l. Trmc., P_{M_i} is t. pc of M_i wrt t. reference Umc.; P_i is t. probty induced on t. corpus by M_i.

From t. formula, it seems clear that if P_{M_j} is t. pc of t. Trmc that produced t. corpus (or, more exactly, t. sum of t. pc's of all Trmcs that would produce the corpus w. t. proper probys), Then for a particular reference Umc, P_{corpus} (from .20) $\geq P_{M_j} \cdot$ (true pc of corpus)

[T. way we implement .20: let $\tilde{i}$ be t. binary string that represents t. integer, i. Say we feed $\tilde{i}$ into our Umc. This Umc is now by definition M_i. ($\tilde{i}$ results in some outputs. Then if all objects in t. corpus have that output as prefix, then its poss'l that t. corpus has pc > ϕ: If 1 or more objects in corpus doesn't have that prefix, then pc of corpus wrt M_i is ϕ). We then use .08–.18 to evaluate P_i (of .20).

.34 $P_{M_i} = 2^{-|\tilde{i}|}$.

Consider (.08–.18) whether or not T. Trmc is Umc: is this distribn t. discrete or continuous Universal distribn? (our 2nd or 3rd run?) It seems that it is both:
Th. cont'd discrete distribns over both semi-measures so $\sum$ of each is ≤ 1; sum of both is ≤ 1. Actually, there are (at least) 2 kinds of Umc: One has a stop state, one has no stop state: Either machine type OK. for .08 → .18! If it has no stop state it produces ~~no output after T time.~~ one of t. finite objects, & then has no output for > T time, we consider it as having produced that object & stopped.

I'm not sure if we can get rid of t. need for a 3rd symbol alphabet for finite objects.
A toy: T. prob of 0 or 1 after t. end of t. objects is zero. Such a is a finite object (string) →

9.10.98 TM Mixd corpus Problem.

printing α

.01 This means that in 110.08–.18 we reject (= give pc = ϕ) to all codes that give ϕ or 1 after
.02 as output. → (.20)

.03 Idea for a somewhat simpler approach that ~~may~~ makes proof possl. w.o. 3 symbols:
Since generation of t. objects is independent, we can treat them via Sol78T3, individually.
Consider one object in t. corpus. Our system assigns pc = P_1' to it., t. true prby is P_1.

$\frac{P_1'}{P_1} = \alpha_1$: $\alpha_1 = e^{-k_1}$ so $E\Sigma$ err² for object$_1$ < $\frac{k_1}{2}$. } This is via Sol 78T3 corrolary (a general thrm). see (.13) (.13 calls this S78C3)

$E\Sigma$ err² for all objects < $\frac{1}{2}\Sigma k_i$; and we know a bound for Σk_i : so that would
since $\Sigma k_i = -\ln \frac{\Pi P_i'}{\Pi P_i}$ ← From 110.20–.34
.10 prove t. thrm. !

.11 Some troubles: ① I'm not comfortable about these "expected values" they seem to be
→ see 01 ID381.00 for some discssn.
.12 over different ensembles ② I don't know if Sol78T3 can be applied to finite objects → It does. See .14
Def .13 Sol78C3 (Corrolary of T3) may apply, just as long as $\frac{P_i'}{P_i} < \alpha_i$ for all bits in t. object.
.135 → ③ S78C3: for some i $\frac{P_i'}{P_i} > 1$ so $\alpha_i = e^{-k_i}$ & $k_i < 0$ so EΣ err² $\Sigma k_i < 0$!! which is imppssl. — Look into this (9/22/01) → See 111½.00 for some discn
.14 On .12 We can apply Sol78C3 to finite objects: If t. object is a string of length m, then it applies directly to t. first m bits of t. object, w.o. modifn. For bits past t. m^{th}, t. additional error is always exactly zero, so t. thrm. continues to hold exactly as before.
[9/16/01 — what about ①? (.11–.12) ? See 01ID381.00 for some discussn.
9/22/01 — Re ③ — Still troublesome!]
.17 I think (.03 – .10) holds OK.

So: T. proof of t. Mixd corpus Thrm. (as of now) : is 110.08 – .34 ; 111.03 – .17
.18 → .37

.20 (.02) On t. Universal discrete v.s. Universal Continuous distribution:
① t. Discrete distrbn: T. pc. of finite string x is $\sum_i 2^{-|P_i(x)|}$ $P_i(x)$ is t. i th pgm that inputs t. reference machine & has x as output, then stops.

② Continuous distribn ②ⓐ finite string, x, $\sum_i 2^{-|P_i(x)|}$ $P_i(x)$ is i th pgm that inputs ref machine ~~has x as input and has prefix x.~~ w. output that has prefix x. The output may or may not stop after x is printed. [This is my defn, using "minimal codes"].

②ⓑ The Lebesque measure of ~~all~~ t. codes of all infinite strings that have x as prefix. Maybe use UMC that has no "stop" state!] This is Li-Vitanyi's defn.
This excludes codes of all finite strings. — But check on this.

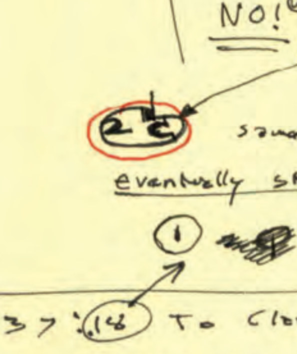
I think ~~① + ②ⓑ = ②ⓐ~~ : 2b includes only codes of infinite contns of x
NO! ~~① + ②ⓑ = ②ⓐ~~
②ⓒ + ②ⓑ = ②ⓐ

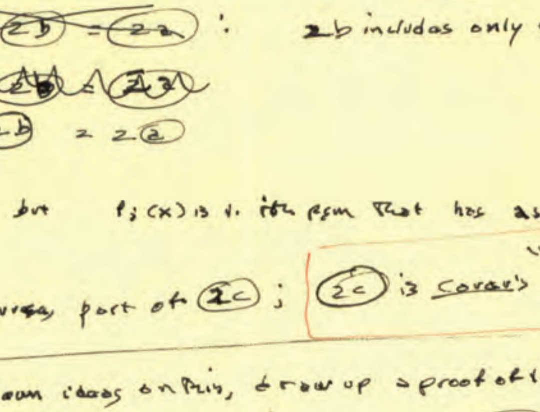
I'm not sure! I really don't know what their defn is.

②ⓒ same as ① but $P_i(x)$ is t. i th pgm that has as output a string w. x as prefix, & eventually stops.

① ~~is~~ is, of course, part of ②ⓒ ; ②ⓒ is Cover's "Extension Complexity"

.37 (.18) To clarify my own ideas on this, draw up a proof of t. ~~…~~ form. for unordered finite object corpus





see 111½.00
→ 115.01

9/23/01

Maybe not! p_i' & p' are the pc's [illegible] corpus up thru i'.

.00: 98TM111.135: On the $\Sigma(\text{err})^2$ bnd for BAG Induction; objection (3) for some i, $\frac{p_i'}{p_i} > 1$; $> e^{-h_i}$; $h_i < 0$ — so

$E\,\Sigma(\text{err}^2) < 0$ ≡ (if this) is true then $E\,\Sigma\,\text{err}^2 \le 0$ is also true, — which implies $E\,\Sigma(\text{err}^2) = 0$.

— No error! I'll have to look at proof of S78c3 to see what's wrong!

It is certainly not immediately apparent! Perhaps the Corrolary is false! (But See 8/19/02!)

8/19/02 Actually $\underset{p_i}{E}\left(p_i \ln \frac{p_i}{p_i'}\right)$ is what is of interest & it is always ≥0 (≡ Gibbs Thm).

Σ or something like this: → (Most likely in error): If Σp_i [redacted] $=1$, $p_i \ge 0$; $\Sigma p_i' = 1$, $p_i' \ge 0$ } then $\sum_{i=1}^{K} p_i \ln \frac{p_i}{p_i'} \ge 0$ } I mean is Gibbs Thm

So, the K-L distance is always ≥0.

This is a radix = K system

0.13.98 TM: ORG, EVOLN, GA: $\frac{30}{165}$ $\frac{6}{33}$ $\frac{2}{11}$

~~Org. evoln is quite difnt fr~~

T. ~~gen~~ etic Code for Development is quite difrnt ~~from~~ that of most computers: Each ~~site~~ locus on t. chromosome has several possl. Genes (Alleles?). Other Genes then orchestrate these Genes by turning them on & off at various times'.

I guess it is ~~mostly~~ mainly t. locus that is orchestrated.

In addition each gene may have clues in t. envt. that tell it when & just how to act. || D2299: The "control genes" are sensitive to envt.: they orchestrate t. "expression" of "structural genes". "expression" is t. no. of copys of a protein made at each time. Corresponds to RNA production (duplicates).

I tend to think of computer pgms as composed of chunks that are sequencially activated. This is quite difrnt. from t. parallel operation of chromosome codes

An argument for use of randomness in GA: Because t. G function is normally quite noisy (Inherent noise in t. G funct itself, then, for each reinforcement of a gene or subset of ~~[illegible]~~ genes, the ~~[illegible]~~ interaction of randomly chosen other genes.). For this reason, we want to try various genes many times w. various other ~~accompany~~ accompanying genes

For "chunking" criterion: say g_i is t. ~~[illegible]~~ allele at t. i^{th} locus. $G(g_i)$ is t. empirical (noisey) Gorc, for random values of all other loci. $\Delta G(g_i)$ is t. ↑ of ~~[illegible]~~ $G(g_i)$ over $G(\tilde{g}_i)$ which is the value of G for t. worst ~~gene~~ allele at that locus. We'd like to say that for 2 difent loci i, j; if t. loci are indip, then the G's of any confirmation of alleles at t. 2 locs are "addative". If super addative, they are "synergistic"

allele: one of t. various Genes that can occure at a particular locus.

2.1 Hvr, I'm not sure I understand exactly how "addativity" is defined; Say t. Gorc is "linearized" to start.

Say we have 2 sites: 1 & 2: at 1, the ΔG from min to max is X_1
" 2 " " " " " " X_2

If t. sites are indip, then t. ↑ from min to max at site 1 is indip of value at site 2.
~~So, [illegible]~~ So, if we ↑ G by getting max at site 1, we get X_1,
" ~~[illegible]~~ then ↑G " " " " " 2 " " ~~[illegible]~~ X_1+X_2

As is, $X_{1,2}$ are defined as "Δ wrt. all possl. configns of all other sites"!

.30 Since this is too much to calculate, t. evoln. is done by "sampling".

Suppose our "linearized" Gorc is like so: We ↑ X_1 at one site, then all all other sites t. ΔG is sub linear: This ~~[illegible]~~ can be an artifact of ~~[illegible]~~ t. Gorc! i.e. Say G_1 is a Gorc for ~~[illegible]~~ which t. sites are all addative (indip)!

.35 Then we have ~~[illegible]~~ $G \equiv \ln(G_1)$ which must be sublinear for all site pairs!
.36 If we had $G = \exp(G)$ then G would be superlinear for all site pairs!

T. above suggests (strongly) that my ideas on "synergy" need ~~some~~ more careful thot!

.38 In case of .36 we might want to Bunch all of the ky G alleles into one "Macro ~~[illegible]~~ Allele" — this would, indeed, maximize G. ; Hvr, in .35, the sites seem anti-synergistic, yet t. same strategy ~~as~~ of .38 would be optimum. !

01398 TM Org Evoln, GA.

Lets look at this "indep" sites thing again: say all $X_i = 1$. If we consider n sites, & all possible combns we get a Binomial (dfn. (?) $\approx$ Gaussn. about $\sum X_i = \frac{n}{2}$, max dev $\pm \sqrt{n}$ or $n^{1/2} \approx n$.

Now, if we take $\left(\begin{matrix}\ln(\sum X_i) \\ \text{or } \exp(\sum X_i)\end{matrix}\right)$ it will be fairly linear for ± 1 of right if n is large.

$\exp(\sum X_i)$ will not be, but is unlikely that our Gsrc would be that non-linear:

probly $\exp(K \sum X_i)$ w. small K or $\exp(\frac{\sum X_i}{n})$, is more likely.

Anyway, I guess it's unlikely that fit. Gsrc will be non linear enuf so that fit. digits of 112.35-.36 are of much import. — BUT I'm Quite uneasy About this argument!

Perhaps a better way to look at the problem in Genetic Genetics!

That we have for n vector each site is a vector component, & each component has Allele values.

We want quick & easy approxns. to fit. Gsrc.

Taking pairs of sites in to account is maybe using a Quadratic Approxn of fit. Gsrc!, & Quadratic form or exp(Quad form). Since fit. vector components are not continuous objects, but have discrete (Allele) values, & Quad form is not easy to define.

.14 One way of doing this approxn of fit. Gsrc is suggested by 112.21 ff: We start by assigning an individual ΔG to each allele at each site as 112.21-.30! We then examine pairs of Alleles at different sites & see if they don't add linearly; if not, we note the Δ or $\%$ in the joint G v.s. the sum of their G's. After finding pairs of alleles that have by ΔG, we try combinations of them w. other Alleles at good discovered pairs in hopes of finding even more non-linearity of the types.

Note that in 12.35-36, if one did do hill climbing in a simple max slope & greedy way, one would find the peak — in either 12.35 or 12.36. In 12.35 using .1 steps, one would declare victory after a mere 11 steps; but the result would be the same. In 12.35, one would not declare victory after 2 steps or any "steps" but would end up w. same results with peaks.

1:35 PM 7 9 CPU & MMX

SN on T. "Building Block" model/hypothesis for GA.: In my "In Progress in AI," in 1965, I had the idea that if one had crossover and breaking & rearranging of chromosomes one could have "chunking"! My impression is that (except for koza) they just do crossover, so very little "chunking" occurs! They may use other models of "crossover" so they do get "chunking"

Having many Chromosomes is another method of "chunking": Do chromosomes ever get "mixed"? (e.g. front $\frac{1}{2}$ of 1 chromsm, concat. w. back $\frac{1}{2}$ of another).

N.20.98 Much has been developed since 01398!

Soln of Mixd comps problem

" " where to get P.D. for OZ probs:

See 130.01-40 for Resource

Anyway, in view of (130.01-.40): I can view GA probs in 2 ~~ways~~ ways (at least!)

.37 1) The prob of GA is an OZ prob., & I know how best to solve OZ probs (130.01-.40)

2) As part of .37 We have this corpus of ($\underset{\text{string}}{\text{cand}_j}$, G_j) pairs: We make a model for the data set (a stoch Grammar), and we try to pick a cand_j w. large expected G_j.

— This is essentially what SGA does, but SGA also tries to get models in which choosing cand_j for highest expected G_j, is easy to do.

→ 132.01

0.17.98 : TM   Note on Sol64a.

On use of <u>General</u> univl. machine (no restrictions on I/O):

<u>D</u>efn. of ALP is unclear: Some possl. defns:

1) "Output" defined to be when machine stops.

2) "Input" is all input tape squares scanned befor stopping.

Now output defn: we want to know rel proby of sa v.s. sb (s is long, a,b short) ~~[illegible]~~ an input outputs "sa" if it <u>ever</u> writes (at least) "sa" on its output tape. Well what if it writes sa, erases it & writes it again? — Say it writes "sa" erases it then writes "sb"? Also, it could write an ∞ of strings ~~[illegible]~~ — so Y. poss. if <u>non-convergence</u> is <u>very</u> <u>high</u>.

There are several Models for induction Suggested. (~~See~~ ~~Another~~ Seen My letter to Cover for ~~[illegible]~~ analysis of Some.

A new one: Umc w. Unidirectional ~~output~~ input. Output tape is bidirectional, but it can't erase.

A certain input results in an acceptable code ~~if t. output~~ as soon as t. output produces a string w. t. corpus as prefix. T. output tape can have <u>blanks</u> in it.

11.40

.01 11.40: A critical Q is t. defn. of t. p.d. being approximated by ALP.

We have as "output" a set of finite objects, some of which are partial objects: (prefixes of finite objects or if an infinite string).

Several ways to define t. p.d.: We can define it as assoc. w. some UTM machine in all cases.

For any corpus: w. random input, what is probby that

a) T. output will be that corpus element.

.05 b) T. corpus will be a prefix of t. output string. ← least restrictive.

c) " " " " is a proper prefix of t. output (i.e. t. output is at least 1 bit longer than t. corpus element.

d) for finite complete objects, t. output stops after giving that object: for prefixes of finite objects, t. output has that prefix but eventually stops

for prefixes of infinite objects, t. output has that prefix, but never stops printing.
it may or may not eventually stop printing.

Consider various sub-corpus cases: in all cases, output has that sub-corpus as prefix.

A corpus: a) prefix of some infinite string: [illegible]

1) output can be any finite or inf. string. (.08) (Sum of 2) and 3))

2) output must eventually stop (Extension Complexity)

3) output must be an infinite string ("cylinder" probability)

B sub corpus is / complete finite object
output has that prefix

1) must stop immediately after printing that object

2) " " eventually after printing that object.

3) may or may not eventually stop.

C sub corpus is prefix of finite object
output must have that prefix

1) must eventually stop.

2) may or may not eventually stop.

.29 Which of above 3 possibly is true "correct" depends on one's knowledge of t. corpus. e.g. in B if one knows t. object ends at that pt., 1 must be true.

In A if one knows t. object is an inf. string, 3 must be true.

In C if one knows it is part of a finite object, 2 must be true.

.29 Seems to solve t. problem!

Next, Given t. "true" p.d. of the Mixed corpus, and our state of knowledge (.11, .29) at each of the corpus elements, we are able to show 1 a machine associated with that p.d. is that state of knowl. "I." state of knowl." enables us to tell which codes to this machine to accept.

In t. UMC (ALP) approxn, we use t. same rules for accepting the codes. So we end up w. a "count factor" that is t. pc of t. "machine that represents t. p.d.". — And from this, we can use 7&13 to prove the upper bound on error, using 11.03

92898! Jeremy!

Not a good Beginning.

I have read your "learning how to learn" paper some time ago, but until now haven't found time to write you about it.

Though I found several questionable features of the program, it's likely that these bugs can be fixed (if, indeed, they are bugs!). As I see it, The main question is: — Can one get a machine to learn how to learn, rather in what will ultimately be the best possible way, by using a relatively simple reinforcement routine? If so, is this a practical way to get very intelligent behavior? — How long would it take, and how difficult would it be to devise a training sequence for such a machine?

Apart from these remarks, I found the ideas to be stimulating — they forced me to express more clearly, my own goals and the techniques I've been using and expect to be using to achieve them. (confusing to parse)

→ 116.01

02898 TM: ∞ versions.

.01 | 15 AD Looking at his "Learn how to learn" can this be formulated as an OZ problem? — The rules about how ER ↑, & t. ultimate gore (whatever it may be ... eg. J's ultimate Goal) can be made part of t. problem.

.04 Hvr, we may also view it as a "sequencial OZ problem"! A sequence of OZ problems, in which TM is supposed to learn a better & better P.D. as it continues!

This is perhaps what I was aiming at in 108.09 (& preceding work) on "OZ probs".)

Say I have a set of OZ problems: $[M_i(\cdot), T_i]$ — Then there is some Algm, A( ) which takes as args $\vec{M}_i \in T_i$ & ($\vec{M}_i, T_i$) & outputs a string, X_i that's v.g. (i.e. $\Rightarrow M_i(X_i)$ is very high) $G(\vec{A}) \equiv \sum_i M_i(A(M_i, T_i))$ (Assume Gaussian "LINEARIZED")

for every Algm, A $\exists$ $G(\vec{A})$! we want an $\vec{A} \ni G(\vec{A})$ is max.

— BUT: How Much time Do we have to find A? DO we Have time= $\sum T_i$?

.12 Hvr. The main idea here, is that J's problem of "Learning how to learn", is really a "ordinary" lrng. problem. If there are better ways to lrn than Lsrch OZ, then they are within a constant factor of OZ.

— Hvr, I still don't understand t. OZ problem" (108.09 & preceding work) → 117.31; → 127.01 → 128.01 Looks like a quite a quote Soln. to t. OZ problem!

.20 t. "sequencial set of OZ probs" of .04 may be a key.

While J's "Lrng. problem" would ordinarily be an ordinary OZ problem & it is is not for 2 reasons: (1) It got F.B. (partial values) during its trial.

(2) It is only allowed 1 trial.

Hvr, 1 way such problems are solved ("normally"): TM makes a cheap Model of t. G function & he hill climbs repeatedly — (in private) — "core ballum" — cheaply — Then makes "e.g." eXternal trials.

D298 Jargon $: One main idea of "Lrng how to lrn" is minimal "Bias": That one t. System Lrns in a way conditioned mainly by its TSOP.

However, The language used by t. system. (instruction set) is as strong a Bias in J's system as it is in ALP.

A critical system is "Where is t. Summary of Experience stored". J. stores it in t. Stack, in t. individual stack Components

T. idea of .12 - .20 is attractive (now that I've "solved" t. OZ problem!). J's system may be a reasonable approxn. of t. Genl. Soln.; & if so, can be criticized, improved, and used, on Real Basis.

149.37 – 150.15
153.01

02898 TM **OZ probs** : [.01 – .31] Seems to solve the problem, conceptually: practical approxn. to this has yet to be detaild. 117

.01: 108.40 A direct look at the OZ problem syem: we have this data set of 99.39; 24.5.5117

Def 24.5 → 17 (= 1.7)

.02 [OZprob$_i$, HCmethod$_j$, cc$_{ij}$, G_{ij}] : Given the cc value T, (of Gtzn Technique & → Data set of 99.3) 21.5 → 17

we can, In the Manner of (SGA), get a d.f. for the G obtainable w. an arbitrary HCmethod, on (OZproblem$_i$) using cc=T. From this d.f., we should be able to Get the proby of each OZmethod giving the best G in time T.

N1699: I Don't See How SGA is Relevant!

These 3 d.f.s are the pd of G values for 3 HCmethods w. cc=T on OZprob$_i$ (= OZP$_i$).

.10 What is the probly that each will be max? → Actually, we can jump to .31 from here, but .10–.25 is conceptually of interest & may be good way to do Approxn.

Say the d.f.s are indep.

~~Pretty~~ $P_{1,2,3}$ are the 3 pds $\int_{-\infty}^{+\infty} P_i(G)\,dG = 1$.

Proby that "1" will be max is $\int_{-\infty}^{+\infty} \left(P_1(G_1) \left(\int_{-\infty}^{G} P_2(G_2)\,dG_2 \right) \cdot \int_{-\infty}^{G} P_3(G_3)\,dG_3 \right) dG_1$

Say $F_i(G) \equiv \int_{-\infty}^{G} P_i(G)\,dG$; So $\int_{-\infty}^{+\infty} P_1(G) F_2(G) F_3(G)\,dG$.

Say $H(G) = F_1 \cdot F_2 \cdot F_3$ then $\int_{-\infty}^{+\infty} \frac{P_1(G)}{F_1(G)} H(G)\,dG$

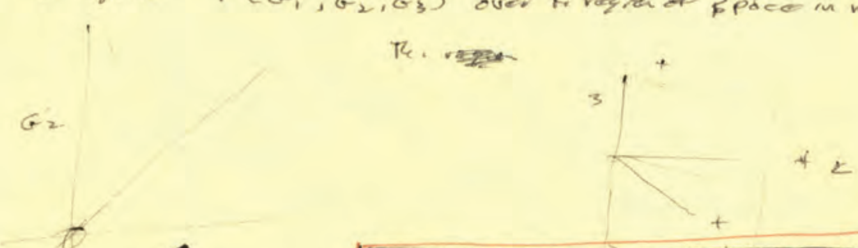
.21 So we get $\frac{P_i}{F_i}$ for each i: then dot mult it by $H(G)$.

If we assume independence, this solves the problem.

.25 If " " dependence, I think the solution is still about the same form.

Another way to think about it (for dependent or indep $P_i(G)$)

Integrate $P(G_1, G_2, G_3)$ over the region of space in which $G_1 > G_2$ and $G_1 > G_3$ —



The region $P_1 > P_2$ is bounded by a plane thru the "3" axis

Region $P_1 > P_3$ is bounded by plane thru 2 axis.

We could divide up the space into 6 (=3!) regions of $G_1\,G_2\,G_3$ ordering.

.31 We want [for the same cc=T for all 3 G_i's]

$$\iiint_{\substack{G_1 > G_2 \\ G_1 > G_3}} P(G_1, G_2, G_3)\, d\vec{\sigma}$$

volume element of the space

Nota Bene The proby distribns of .09 as based on the corpus of .02 — for the problem to be "interesting":

A finite C.B. (RPL) is necy. If CB=∞ for comp of these p.d.'s then ~~they become~~ the probys become δ functs — since in ∞ amt of time, we could compute exactly how much G would be obtained by each HCmethod on the OZ probs. — so the S of .31 would be trivial! → 127.01

.01 : On Search for regys, ect. All regys are searched for, using only previously dcvrd techniques. (plus "primative" techniques).

Presumably — at each stage, the search techniques are "Universal", in the sense that if they had a large enuf

.05 CB, they would eventually find any findable regy.

(SN) I'm not sure the "Universality" condition is a necy property of each search. Perhaps it is only necy for very dif(f). probs!

(SN) Perhaps Have only one Topic on a Page. If I have a new idea — put it on a Separate Page

More Pages, but Certainly << TWICE as Many PP. !!!

.15 → A possl. reason for insisting on ($\tilde{\sim}$) .01 — .05 : we want to be Sure that TM has ability to dcvr. all of the search techniques it uses. If it is then more likely to be able to discover new

.20 search techniques.

It would seem that normally, for each new problem, RM would (usually) first search for "solns" to that specific problem i.e. Trials codes that left p.c.'s of other prob. solns. invariant — but tried to ↑ pc of ~~soln to~~ trial for present problem. This corresponds to the "TM_1" phase. Next (or later), it may look for to do mod.ths that effect > 1 problem (or set of/job problems of a same macro-problem) This corresponds to "TM_2" phase. — or "Updating".

Suppose the Corpus is an ordered list of problems; Some INV problems, but many OZ problems. RM can go at his own pace. The goal Might be: To get Max total "G" + some Weighted Sum of G's for **INV. probs.** — Or, just Max total G for OZ probs, since/the probs have to be worked in specified order, all Inv probs up to the last OZ prob. worked on, have to be solvd.

[classical OZ probs: CB Given for each.] Another formulation would give a certain amt. of time for the total/of. all optzn. probs given. There may be several ways to formulate the global goal

03098 TM

To xform an OZ prob into an INV prob. We want Min (cc time to get G_0) a certain "G_0". While this clearly makes them into INV probs, we seem to lose the H.C. info used in most OZ solns. — i.e. In Lsrch for INV probs, info in past trials is discarded — other than the facts that they had been tried & they had failed!

If we give TM traces of all of its previous trials, then a problem can never be subject to > 1 trial! — since the set of past traces (The History) is always different for each trial. (Shades of Jurgen's "you only live once"!)

N.B. My "Genl Soln" to the "Mxd Corpus Problem" was for a pure prediction problem only. 1170-.40 supposedly tells how to apply the resultant p.d. to the Soln. of OZ probs.

Actually this "Genl. soln." is just what we need for the "Updating" part of the TM routine.

On the General Search & Discovery Process:

Concept "Aquisitions" ② Discovery of Concepts Via Lsrch.

(b) Eveln. of PC's of concs: a uncondl pc (?) (is this useful?)

β Condl.: 1) Within "problem areas" (i.e. This specific problem, Geometry, math, Science...) 2) Between different "problem Areas"

Go thru ALG (A) Notation (N) Learning (L) TSQ: See just how the 3 is Relevant.

One Thing of Import is "IR": It is both a problem for ↓ cc of search & a problem for ↑ pc of "searched-for" object. I was simply putting solns. of probs into "storage". What is justfn. of this, & how should condl pc's be assigned properly? (Both in accord w. ideas of ALP & w. heuristic ideas about Search).

In ANL(1) the system I used was pretty much like having a sequencial predn. problem, and after each "example" (= micro corpus), we "summarized" the corpus implied p.d. due to the data up to that point. From this "summary" we are able to give a p.d. on the future of the sequence.

→ [There is the problem of "One Shot Learning" — & ALP has to deal w. this in a special way]

Note that in ANL I was doing "Operator induction" —

.01 Which assumes the inputs are uncorrelated & have no Info-content. I.e. it doesn't take advantage of any regularities in the inputs.

.03 P.c. of a stock operator O_i would be $P(O_i) \cdot \prod_j O_i(Q_j, A_j)$

$P(O_i)$ is the prop. of operator O_i. $O_i(Q_j, A_j)$ is the probby (as given by O_i) that input Q_j will result in output A_j

.06 The eqn. soln. of the operator induction problem (when sums .03 over all i) — $P(O_i)$ is given by the lengths of decrs of O_i (summing over all decrs).

We can obtain .06 w. a 2 input Umc. — one input to decrb. .10 the O_i's, & the other input for the O_j's; & the output will be the A_j's.

N698 The ideas/problems:

.14 The idea of the Alg. Notatn. Lang TSQ was this! We start w. a short Corpus.

.16 We have a set of concs so we can use simple search on those concs, w. uniform P.D., to do trials & find a soln. to the problem, which amounts to a short code for the corpus. It implies a difrnt p.d. on codes for the future. This implied p.d. amounts to a "summary" of the past.

.18 Using this P.D. we search for soln's to the next problem set. Loop to .16

.19 Now: to what extent was ANL completely decbd by .14 – .18?

Given a set of examples, I would create a short code for it using "operator induction". For some reason I then put C in the set of instructions to be used in secondary trials for the next batch of problems. — Why?

Well, say TM learns how to deal w. "+" problems. — makes useable operator! Then w corpus of + & — he learns to do "—" probs also! But he then has 2 operators: Each is v. like the + time! Still a big compression! →

SN Using random nos. w. many digits (for examples) is equiv. to using larger CSZ w smaller nos. — the total compression for the v. like operator is the same in both cases. The method of using these 2 operators — & correcting for errors . . . look info this for "operator induction".

→ Another kind of "Lang" that I may want to analyse from the pt of view of .14 – .18 is soln. of GA problems by search.

→ Back to .14ff: In "unordered finite objects" corpus: Say the objects are large, but they have little or no mutual info: To generate the corpus: The grammar consists of a list of indep decrs of each of the objects. We choose each decrn w. = probby.

Notcom
547
3773
Charlotte
528
3356
Alex
100 step sum

Say $A_T(x)$ & $B_T(X)$ are 2 proby distribns. — They are each finite CB (CB=T) approxns to t. ALP of the string x.

$A(x) \equiv \lim_{T\to\infty} A_T(x)$; $B(x) \equiv \lim_{T\to\infty} B_T(x)$:

$\lim_{T\to\infty} \frac{A_T(x)}{B_T(x)} = \frac{A(x)}{B(x)}$. (We can be sure $B_T(x) > 0$.)

Now: what does it mean to say $\frac{A(x)}{B(x)}$ is "not enumerable"?
In what sense is $\frac{A(x)}{B(x)}$ "less computable" than A(x) or B(x)?
(Say both A(x) & B(x) are incomputable.)

It may be that while A(x) & B(x) are both incomputable, we can at least get a lower bound for each, which ↑ as T↑ — so t. bound & our approxns. are Monotonic in T.

Th. successive approxns to $\frac{A(x)}{B(x)}$ are not monotonic in T., so it would seem that our uncertainty of it would be usually "worse" than our **uncertainty** of A(x) or B(x).

On t. other hand, the non-enumerability of the recursive functions, does seem to make the "summing over all cpm's" an essentially meaningless statement.

Perhaps see Li-Vit for explann. of "Enumerability" 23

1.215/d → 3.07/yr.

.28 (They do say, and I think I said in SOL78, that there is no normalized proby measure that dominates all normalized proby measures. — The say universal semimeasure dominates all finitely describable semimeasures.

T. implication of .28 is that t. ratio of t. normzn. constants of 2 universal semimeasures can → 0 (or ∞) as t. length of t. corpus ↑.
Maybe t. truth of .28 depends on what defn. of "normzn". is used.!))
It may have been proved for a normzn system Different from Mine!
I think what they mean by "normzn" is simply $\sum P_i = 1$. I think t. Thm is that there is no universal, finitely describable, "normalized" proby measure. — This would include all methods of Normzn.

W Nov 25, 1PM

I had the idea that one could try to Model the discovery of various Sci Concepts by seeing if the CJS for the proposed models were reasonable. One initial difficulty was that it was hard to tell what input info each scientist had: — Tho from his writings one could often get a minimal set — a "lower bound" on Input info.

.06 Another/even more serious Difficulty. Lsrch is based on a Prob Distribn. These are Conditional P.D.'s - based on many features of a problem. Many of these ~~P.D.'s or~~ condl. pc's used in Lsrch of a human, would be computed "unconsciously": Tho these condl. pc's are quite impt., we will very often have no conscious access to them — the scientist doing the search will not have conscious access & the "Historian of Science" will have even poorer access.

—. Yet knowledge of these (unconscious) condl. pc's appears to be Critical in tracing possl. paths of Lsrch & in evaluating C.J.S.'s.

The colloquial (non-rigorous) analysis of the His. of Sci, is even less able to deal w. the problem of .06! So what Good is His. of Sci?

Well, Hof S. can try to find possl. modes of Conscious evaln. of condl. pc's to be used in Lsrch. The idea is that we may in this way, discover Some impt. tricks (=Heuristics for search/discovery) that were used by V.G. Scientists of the past. — but not all tricks, or perhaps not even many V.g. ones!

Finding models for past discovery can help & train present day Scientists — even if the models are not "true" — if they are "good" in a short rode sense, then they are of interest & they are potentially useful.

Main Problems

Suite

.01 1) T. OZ problem: T. "soln" of (117.01 — .31) is o.k. as "a conceptual model" The details of how best to Approximate this, have to be worked out → Are t ideas of "Information Based Complexity" ( ) relevant? (see 127–128 for pretty Good Approach

2) 118.01 — 120.10: This is an over-all view of how TM operates: The idea was to take Alg Notation Lrng (i ~ TSQ's) & try to give a better theoretical understanding of them — & (perhaps) improve them... Then try to fit them into the Mxd corpus scheme.

Write more frequent, more detaild reviews
Mixd corpus problem ends 115.40

.11 3) "SMA" problem: Stock Mkt Advisors problem: What is t. expected (unbiasd)

.12 estimate of "yield" of a sm strat. or "Action Algorithm". ? I don't remember just how far I got in solving this problem: There were at least 2 (relatively recent) indip investigations. One or both in "SM" file: perhaps within t. last year.

At present, I'm not sure it is very critical in t. TM goal. — It seems most relevant when one has large data bases, & when one needs much accuracy in pc estimates. — But I don't know. For TM applicn., I was thinking of evaluating TM's overall "Action Algm" and trying to Optimize it. In any optimization problem over statistically fluctuating (noisy) variables, one has this problem of of "SOY" (spurious optimization Yield). At least One of the approaches attempts to estimate t SOY for various Statistical Situations. Th. problem is particularly (Bad / Sever) when t. variables one picks t. "best of" have noise levels that are ≥ th. differences betw. their means

How impt. is J's idea of: "You only live Once" & we can never repeat a trial.

Is "SOY" relevant to (.01) (117.01 — .31) ← in particular.? in .01, we're not (Yet!) so much concernd w. t. magnitude of error: all we want is a p.d. over t. choices, of what t. "best" choice is. ← Th This is also what we want in Straight SM work, th. need for quntification arises when we compare the peaks obtained from somewhat different data sets. Presumably, if we had a joint p.d. for both data sets together, we would not need to know SOY. Hvr, it may be that in SM (& maybe in TM?) we often have to compare t. "Best choices" betw 2 data sets & we don't have t. joint distribu. funct.

A way to solve t. Soy problem: Say we have a joint p.d. of G_i $(i=1/k)$ of k variables: We have way to select t. variable (i) of max expected G. So we just add a $k+1^{st}$ variable, x, of known value & zero σ, to t. set. for each value of x, we pick an i that has max expected value of t. k+1 vars. for a certain G_0, when $x > G_0$, x will be always chosen. for $x < G_0$, x will never be chosen. G_0 is t. expected G of t. max choice.

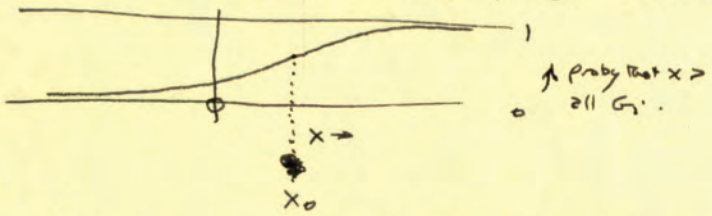
Try it w. k = 1, 2, 3: See what happends: Does it give correct answers? — We probly know correct answers for k = 1 & 2.

For k = 1 it works, but there is no SOY, so not so interesting!

For k = 2 (& for k = n) What is t. probty that all G_i $(i = 1/k)$ are < x?

N 698 TM:

Say we have an X ● ⇒ this probty = .5. I.E. that X t. value of t. peak of t. set of vars. $G_i(z=1/k)$;



.05 One, perhaps impt thing that some of t. Soy work did: It assumed a model for t. noisy variables, so that e.g. t. variables were generated by having their means be normally distributed, from each had a σ_i about that mean. — t. σ_i's could be assumed same, or may could be all diff'nt or be related to t. means. What one runs "assume" depends on $$$ & on how much ↑ in pc was got using t. "assumptions" as a model for t. data.

This (.05) "model" idea is to P.d.'s it gives, seems related to t. practical

02 prob (em of 123.01 (≡ 117.01–.21) — This stuff is very theoretical: The SMA problem gives p.d.'s that one can apply this to — and it suggests a model (class).

TM Regression (linear & Non-linear).

Assume we start w. info about the past of the seq. We are coding from point T_0 into its future. So we code w.r.t. a known pack.

Say the part of the future contains n points. If we agree on linear regressn., the "structure" of the model is the self-del. code for k, the no. of params.

Say σ_k^2 is the observed error in the best k param fit to the n data pts.

So: if $p(k)$ is the self-del prob of the integer k. Then the p.c. of a data seq $\geq k$ coding it codes is $p(k) \cdot \left(\propto \frac{\sigma_k^2 \cdot (n+k)}{(n-k)}\right)^{-n}$

$\overline{\Sigma^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^2$

P.C. of the "errors" is $\propto \prod_{i=1}^{n} \frac{1}{\sigma} e^{\frac{-x_i^2}{2\sigma^2}} = \sigma^{-n} \cdot e^{\frac{-\sum x_i^2}{2\sigma^2}} = \sigma^{-n} e^{\frac{n\overline{\Sigma^2}}{2\sigma^2}}$

$= \left(\sigma^{-1} \cdot e^{\frac{\overline{\Sigma^2}}{2\sigma^2}}\right)^n$ = max when $\sigma^2 = \overline{\Sigma^2}$ so $= \left(\sqrt{\frac{1}{\overline{\Sigma^2}}} \cdot e^{-\frac{1}{2}}\right)^n$

Say $\sigma_k = \sqrt{\overline{\Sigma^2}}$ | [redacted] · P.C. of "error" = $\frac{1}{\sqrt{e}} \cdot \frac{1}{\sigma_k^2}$

So we want to "chose k" so $p(k)/(\Sigma)^n$ is min.

No! error $= \sigma^{-n} e^{-\frac{1}{2}\cdot n} = (\sigma\cdot\sqrt{e})^{-n}$; so select $n \Rightarrow \frac{p(k)}{(\sigma\sqrt{e})^n}$ is max.

or more exactly $\frac{p(k)}{\left(\sqrt{\frac{e\cdot(n+k)}{(n-k)}}\right)^k \cdot \sigma_k}$ Select n to

.22 maximize! $= p(k)\left(\frac{e\cdot\sigma_k^2\cdot(n+k)}{(n-k)}\right)^{-\frac{n}{2}} = p(k)\left(\frac{n-k}{(n+k)\sigma_n^2\cdot e}\right)^{\frac{n}{2}}$

$p(k) \geq c\, 2^{-(\log k + \log\log k + \log\log\log k + \cdots)}$ use logs to base 2: Take only terms >0 in the sum

the constant c (normalizer) is .3490329 3 . . . . $p(k) = \frac{c}{k} \cdot \frac{1}{\log k} \cdot \frac{1}{\log\log k} \cdots$ only use factors < 1.

k	p(k)
1	.35
2	.175
3	.0735
4	.04375
5	.025
6	⋮

For more accuracy, probably if the maximum shows in k: Use a wtd sum of predictions averaged over k from 0 to ∞ w. wts = .22.

T. Porgy. works w. non-linear regressn. to the extent that the system is "locally linear" — "to a distance of interest." — This distance $\uparrow$ as $n\uparrow$. So it's probably "locally linear" if n is large enough.

We also assume that the function to be optimized is "smooth": In SM strategies these are discontinuous, this is a source of trouble!

N1398 & TM

[Tho this Arg't. seems Valid for "Mxd. Corpus" pure sequential prodn.] t. most General kind of problem

.01 & 120.40 : A possl. Good way! Consider we get "chunks" of corpus bo codes. After each "chunk" we "up date t. P.D." — what this means! We construct a new Unic. Rule summarizes t. corpus up to the end of t. most recent chunk.

.04 By "summarizes" I mean that t. best sequences can do is w. random input!

I suspect that there are many ways that can be called equivalent of .01 - .04

→ a) To what extent was "putting of t. latest soln. of t. AND problem into storage" an example of .01 - .04?

→ b) To what extent was "my soln. of t. Nuker problem via Isich" (instead of GA) an example of .01 - .04?

Main Pts:
118
119
120
126

Try to give several examples of .01 - .04 equivalences:

1. We have several (many copys / codes) for t. corpus. T. final summary is a wtd. sum of those codes = a wtd mean of their products. (Prods ≡ P.D. for future).
2. Sometimes A [Rarely several] codes for the corpus each of several sub. corpi: T. equivt. code for the total corpus is cart. product of those codes (if they are all indep).
3. Just how do various heuristic types fit in? {e.g. "Quick abort"}

e.g. How are "Info. Retrieval" p.d.s for obtaining more relevant "Conc. trials", implemented in this schema?

.20 SN In General, when one has solved a new problem, this does modify t. P.D. Some, by itself, but t. updating of t. compact P.D. implied by one or more new problem solns. can take much more time & be a more diffl problem than the original problem & soln.! Perhaps to start off, t. "Naive Machine" should have as good "updating algms. as we can devise. [The updating problems can be considered to be part of t. corpus, so, presumably, t. updating algms. will get automatically better as TM Matures. — So is this like my old $TM_1 = TM_2$ where we used t. "50% soln" to decide what fraction of time to use in improving t. General system (≡ updating algms) v.s. direct soln. of problems

? →

.29

So: .20 - .29 seems to be t. crux of t. "Updating problem" & is perhaps t. most difficult (& least understood/worked out by me) part of RM. Essentially t. "TM_2 problem".

129.01
129.01

OZ probs

.01 ; 117.40 : In 117.01-.40, we end up w. a p.d. over t. various H.C. Methods, but t. method is rather indirect : it involves a long chain of calcs. There should be a more direct way to get this d.f. or approximate it.

A trial approxn : Say we have the data set 117.02 : we want to "Explain it" by using various codes for t. entire data set. Actually, what we want is a "Summary

.06 d.f." → given any [OZ prob$_i$, #OT$_j$, CC$_k$, G$_k$] as input, it will have a Optizn Technique stoch operator w. t. proper probby of that combination.

SN In General, The "Summary d.f.'s" Are very impt. — They may be manly what we calculate in TM (126.01-.04)

.13 → From .06, by integral 117.31, we obtain t. desird p.d. on OZ$_i$, OT$_j$, CC$_k$

.14 T. Sof 117.31 may be (easily) not so easy ! (see .24) evaluated by Mt Carlo. ! I'm not sure, but since we only need approxt. values, this may be adequate. T. idea is that each time we evaluate P(G$_1$, G$_2$, G$_3$; ....) we do get useful info. for each such pt., all we have to do is figure out ..... No!

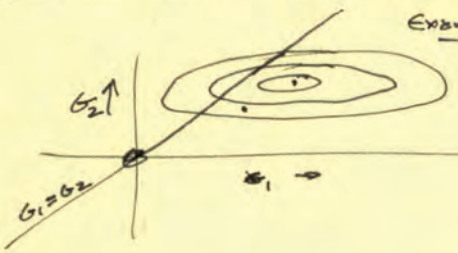
One way : for each # i, we pick random pts in t. subspace G$_i$ > (all other G$_j$'s). We add the P values to t. "i" register :

Another way : we pick pts as random in t. Space : At each pt. chosen, there is one G$_i$ that is max : we then increment the "i register". At the end of many trials, the i registers will give a profile of t. relative probby of G$_i$ being max.

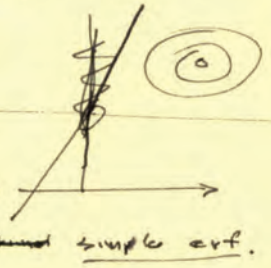
.24 Trouble is, t. real density in most of t. space will be very low, so t. above method will tend to be very inaccurate. Would it help to find t. peak region in t. space? Maybe ! Then by picking pts in t. peak region one might get much of t. total wt.

If P(G$_1$, G$_2$ ..) is expressable (in dip probes) as $\prod$ P(G$_i$), then we can find t. peak of each P(G$_i$) and width of peak. — so a multivariate function, w. known axii. This may not be so hard to deal w. One knows t. peak, & how rapidly it falls from peak, in each direction. T. Monte Carlo approach might be easy then ! → 128.01

.30



Exact soln : for 2 dims, its easy : If t. coords are such that t. distribn is isotropic, then its easy to compute t. % on one side of a line. Just rotate coords and it becomes simple erf.



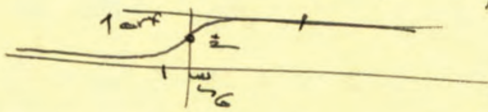
From > 2 dims, it seems much more difficult!

N·13·98 OZ probs

.01 127.30: If there aren't too many G_i's: one can find the center of the d.f., then find the distances from it to each of the hyperplanes that help define $\begin{pmatrix} G_2 \\ G_1 \\ G_3 \\ \vdots \end{pmatrix}$ = max of all G_i. From that info, one may be able to estimate the wts in each region. — Remember, we don't need much accuracy!

T. technique of 117.01–.21 may be fine if we assume indep. & normality. Since we don't need ~~very~~ very much accuracy, $\int_{-\infty}^{\sigma} f_2(\sigma) d\sigma \stackrel{?}{\approx} \mathrm{erf}(\alpha\sigma)$

can be approximated fairly well. ↙ T. middle section is easy to approx. & the tails are probly easy also — but diff't. approxns.



[We may not need to go out far on the tails because we would use info in that region infrequently]

.17 (SN) See [130.25R] for a complaint about this "soln" to OZ.

Main Path: General Structure of TM!
118
119
120
126

See 92.02 ff & 93.04 for some probs for TM to solve. Analog. probs ...

.33 Notes of 6/23/01

.01 126.40: I want to make a "list" of things I want TM to be able to do, & to as much extent as possi., How TM does these things. The problem solns. will use one another.

① (Basic Operation): L srch: Given a P.D. & a problem (INV problem or OZ problem) it performs L srch to find soln. In case of INV, we may or may not want it to continue t. search after it's found 1 soln. In t. case of OZ, it can continue in "Anytime" mode to continue t. search in "available cc".

A critical component is t. P.D. In t. case of INV probs, this is obtaind from t. corpus of past INV. probs & their solns. In t case of OZ probs, we use t. corpus of past [probs & solns], and t. techniques of 117, 127 & 128 to obtain a suitable P.D.

.18 ② (Basic Operation) Updating: This consists of "improving t. P.D.": In one sense, it is an "Anytime Problem" (an OZ prob. w. unspecified cc) w. best compressn as Goal.

Superficially, there would seem to be a problem of just "what part of t. corpus to compress." The idea being that certain parts are more likely to be relevant in t. probs. of t. (near) future. [Very True! Hvr., this is not a real criticism: All it says is that we need more info from t user, to optimize this pd. Info on what parts of corpus are impt. & info on "Time Horizon" of solns. → See (.33)]

T. simple criterion of "↓ total corpus b cost" sounds not. v.g. If there are long time series w. lots of data in them, then T.M. would satisfy this criterion by working mainly on them ~~mainly~~. [Tho maybe not: Since these are OZ probs., one has to have some sort of "linearized Utility funct", to decide how much cc. to spend on each.]

[In general, t. idea of "improving t. P.D." is undefined. How good a P.D. is depends on one's CB. (→ with large amt of CB, many P.D.'s are extremely good! i.e. ALG Proby = ALP) Hvr., P.D.'s ~~may~~ can ~~generally~~ be "partially orderd"

Also t. "goodness" of a P.D. could depend not necly on giving a hypo. for t. corpus, but [redacted] a measure of how good it was in guiding rapid solns. to prob (rapid, Hy & solns of OZ probs; inv.) problems. ([redacted] e.g. Quick abort)

D29.99 "Improving t. P.D." is a task that needs to be more narrowly defined — by having t. "User" give more params.

.33 6/23/01 No! T. USR doesn't have to tell TM what part of t. Corpus to Code. T. USR could give this info & it would help T.M., but normally it is not & need not be available. In Lugano (say April, 2001) I ran into this problem again — Decided that T.M would have to "learn" what part of t. corpus to code, (after it was given t cc limit (= Computation Bound = CB). In general, T. problem of induction, is not to just find max & pc codes for t. corpus in available cc, but it also involves learning how much of t. available cc to use on each part of t. corpus — i.e. how much each part is relevant to t. particular prob ~~rather~~ one is computing.

4/6/05 — See STM 54.14 for some "reasonable remarks" ☺ on this stuff!

What role to olde "field theories" (that were once good) Plex in (RM / Humans) [redacted] Srch, Discovery Exactly how are they used? (Quant. Discr).

90: 6:15 P sum cc

1980: 85 P. 1905 — One = "Quick abort" is small cc for wrong solns.

N1598 TM: **Review**

Some Critical ideas in TM progress!

1) At the begging in I had t. idea that Lsrch could solve all INV probs in a close to optimum way!
"Optimum" w.r.t. t. data & hours [redacted] & CB available at t. time.
I had t. idea that OZ probs could also be solvd in this way, but most recently, it became clear that I didn't know how to improve t. P.D. usd, on t. basis of new experience. → (see .21 for soln)

2) New Developments: (a) SOL78T3: t. convergence thm for sequencial data.
This explaind how t. convergence occured — that it was t. p.c. of t. individual symbols (via ALP) that converged rapidly to t. probys via t. "true" generating sequence.

(b) That t. ideas on meaning of convergence and proof could be applied to a corpus of finite objects. (for applicn. to unorderd set of finite objects, see "Computer Journal" paper 1998 or 1999 for meaning of conv. & applicn. to finite objects and a set of prefixes of inf. objects)

(c) That t. previous ideas could be easily extended to a corpus (Mixd corpus Thm) See 106-108; 110-111, 115.
consisting of several (or φ) finite object and several (or φ) prefixes of infinite sequences. (see 106-108, 110, 111, 115.)

(d) That "updating" t. P.D. for, say sequencial predn., consisted of obtaining a c.P.M. that "summarized" t. regys in t. corpus thus far. There were various means that one could do this in an approximate way, as an "OZ" problem. (see 120.14-.18) Hvr. t. "One Shot Lrng" problem, ("OSL")
.20 was a diffy that usually (if not always) had to be dealt w. in a special way.

.21 (e) I finally found a formal soln. (and good ideas on how to do approxn) to t. problem of obtaining a P.D. for OZ srch, based on a corpus of solns of OZ problems plus any other (Mxd corpus) data that ou mite have. Th. soln is in 2 parts: (I) The first part is to obtain a P.D. for t. G of any O.T.$_i$ (optzn. technique) applied to
25 a [redacted] / particular OZ problem for t. Given C.B. — based mainly, usually on previous experience w. various
26 $\{OT_i, CB_j, G_k, Prob\}$ quadruples of t. past. [Hvr. Still, selecting t. peak of several randomvars is not always t. best way to go! Say w. 2 normal vars. at same mean & var = 1, 1 of means 2 & var 0: t. first 2 is less better but are not.]
(OT$_i$: optzn technique; CB$_j$: Compn. bnd?; Prob: Rem.)
(II) Integrating t. space of t. previous P.D. to obtain t. proby that any particular O.T.
28 is Best for t. Prob of interest for t. stated CB. (117.31) So OZ is "solved"
108.09 — .40; 117.01 — .40; 127.01 — .40; 128.01 — .13; (130.25R) Lmp. Criticism.

Did I solve this problem yet?

3) Using the techniques and developments (a) thru (e), it becomes possl. to solve any solvable INV or OZ problem (starting w. a given initial P.D.): Then "update" t. P.D. so it summarizes t. past Apriori info as well as t. problem just solvd.
This "updating" process is a kind of OZ problem (But see 129.18 — .30 for some difficulties / inadequacies of this viewof this process.)

General Remarks :::

Looking at the review of 130.01-.40: I think at least completely that our parts are:

1) How to solve problems: (Inv or OE): By this I mean Not simply <srch — but just how the P.D. enables various kinds of hours.

2) Also, in implementing hours, just how does the P.D. work? What CB is used to obtain this or other P.D. values? (Note that the P.C. output of a P.D. depends on CB used.)

10:50 start.
<10 min.
drift.

11:25
9
30
31

Perhaps best work on a problem, or problems in hours, to see how this works. — In general, it would seem to take some cc to implement a heuristic: could we somehow add this to the cc of the basic involved?

Well, a single hour, may involve several trials — so how would this work?

3) I should write up the Review more carefully. → 130

** 4) Is there some way to automatically include the cc of the P.D. in the <srch process?

* 5) On OE prob: look at (130.25 R): I'm not sure the SOV "solv." solves this problem!

6) W.R. the TM only working INV problems: The first set of problems uses an uncond'l appln. The second set uses a conditional applnd: The "conditions" bring the problem to be solved. So we need induction on finite set of finite objects.

7) Very Impt. idea on TSQ's: using ideas of ~~Perceptron~~ MCT, I should be able to write TSQ's much easier! One easy way would be to follow conventional Math texts, filling in w. "telling" (when RW unavl. seems necy), using "hints" etc.

Try to get TM's "concepts" ≈ to those that I think I have.

Another source of TSQ's: Problems solved by other ML researchers.: Remember that present System is able to go from any particular state of knowledge to any other particular state: States are given as the Conditions in "Condl. Prob'y."

Also Note: 146.30: How to solve $1+3x=7$, using ideas that one pretends some unknown x is one pretending various facts about x.

8) An Impt. set of probs. in main line of TM.: Give examples of Many types of (Hours) probs & their solns. How are these probs & solns. put into the "Summary Machine?". What are different ways of summarizing?

9) continue from (7) on TSQ's: Another source: Look at classic works in A.I. on Prob. Solving (various A.I. "encyclopedias"). Understand how they give ways to solve problems: Make TSQ to learn those techniques.

N20.98 TM GA, SGA,

.01 : 113.40 : Now, to what extent is "Conventional" G.A. ≃ passing a "Stock Grammar" thru
.02 t. data set ≡ [cond_j, G_j] . ?

"Updating"?

I think t "Great Breakthru" of 130.01-.40, is t. idea that making t. stock Grammar is t. "Updating" process. = The "Summarization" process. T. main weak pt of this technique is that it doesn't deal w. OSL (one shot learning)

The [redacted] Grammar idea then supports the idea of "Building Blocks."

"Building blocks" are cncs. that are used "often" in t. dcrn of t. hy G set of conds. "Often" means defining them enables a more compact encoding of t. corpus (≡ [cond_j G_j] .).

1.3.99 : OK.: First realize that Normal GA & SGA are not similar to Search OZ solns.

Next: In "passing a grammar" thru t. data (.01-.02) we must note that it is a conditional Grammar—
.11 for cond. being t. dcrn. of t. "often problem". This means that any other "often prob" method
.12 can by on/t. system in t. past should help construct t. condl. grammar for t. present often. prob.".

Well, I'd like .11-.12 to be true — i.e. so each GA or SGA such could learn from previous (S)GA searches: But in fact, they do not. It is a weakness of both GA & SGA. {6/23/01 At this time, did I have any good ideas on how to "Update" a conditional Grammar?}

20 20
30 10
15 15
65 45

Now: [In a super-good mutation-only GA, [redacted] t. system would keep a record
(a) of t. (trace / path) of each search, & make an updated p.d. based on this past history, &
make trials to both "inform" & to "eat hy G".

(b) In a (mutn + crossover) GA system, ideally, each trial is made using info about all of the past 11 trials.

β is good if one has to do things in 11, but if β is done serially, it would seem to have no advantage over α. In both cases, one uses all info from past trials.

In fact, if we do β serially, α can exactly simulate β & still be a "mutation only" system!
.27 Looked at this way, t. only difference between mut. only & Mut + "crossover" is that "mtcr" is parallizable & pure mut. is not. (pure mut. is equiv to ≃ sim. anneal).

≃ SGA is about as good as one can do [if one doesn't use .11-.12 info] — say one doesn't restrict SGA Grammars to ones in which hy G pts are easy to find.

.27 is true for idealized search; In practice, "mutation only" GA, only looks at info about last 2 or 3 ancestors! "Crossover" "+ mut" does look at many more (past 11, serial) trials, but t. means of using this info is usually rather narrow.

However: Consider t. early Koza system (Crossover of function trees). To what extent can it be approximated by a kind of stock Grammar? — or an "Evolving" stock Grammar? To t. extent that any gen. population, w. [mut &/o crossover rules] implies a p.d. on next generation, — we have, by def'n, an evolving stock Grammar.

N 2098: TM : Chaotic Models for TM ! | A not so Great review! Sci Amer: Dec 98 pp 45, [illegible] 47

Sinha, Ditto PR Letters > Sept 98!

I haven't read this art. yet! — some ideas!

According to Chris Morris model of dynamic processes, certain continuous functions can act as UMCs:

$\to F(X) \to$ delay — I forget what t. I/O is, hvr. The info storage is in t. value of t. function (or argt.) so $X_{n+1} = F(X_n)$.

Any way I had t. idea that we have 2 input continuous functions, we could somehow get lots of computation! If we had BW. = ω ; $\tau \equiv \frac{1}{\omega}$, noise (rms) = b ; Possible values of $F(X) = M_X$

Then if T is total length of delay line, it could store $\frac{T}{\tau} \cdot \frac{M_X}{b}$ diffrnt states ($\equiv \omega T \cdot \frac{M_X}{b}$.)

F(x) itself can only store $\frac{M_X}{b}$ states, but with t. delay line we can mult this by ωT.

By having many F( ) units interacting, we can ↑ total memory.

with $\to F(x) \to$ delay, if $T = \tau$ ($\omega T = 1$) then we have 1 computation going on.

if $\omega T > 1$ we can have ωT indep calcns going on in t. system.

If $H(x, y)$ is a funct. of 2 args (like $H = y \cdot x(1-x)$) (logistic Equ.)

we can have these functions interact.

I had t. idea that we could do induction in such a system! When t. "output" was within ϵ of the corpus (say RMS error), we would have an equivalent code length related to ϵ & t. corpus length. — & we could give rel wts. to various approximate codes.

Hvr., I don't know where t. input code or t. output!

So I don't know what t. "summary machine" is. I'd like to be able to get "progressive summaries" as t. corpus ↑ (≈ T.S.Q.)

.23

New tech: Say we have a bunch of interacting function Generators & delay lines:

We can "start" t. system by setting say 10 of the functions at $(\frac{M_X}{b})^{10}$ diffrnt states, & all t. rest at ϕ. We let t. system run for a fixd time: t. "output" is

.26 defined to be a certain set of function outputs. (maybe all of them).

More Generally: A "Computer" is a set of active elements connected in a complex way. T. start state of t. system is its "pgm": Its output can be defined in various ways.

1) As .23 -.26

2) when computer "says" t. output is ready: say on certain registers a/o where t. system is in its state space, is an "output": Presumably there are large equiv. classes of states.

E.g. Getting output from a certain set of regs, is same as saying many states are "equiv" (if they have t. same output regs.).

I'd like to be able to characterize various kinds of "computing systems." What are good ways to each for I & O?

In an ordinary UMC (no under I/O), for induction, t. I/O has to be very carfully defined (see the letter to [illegible]) → 136.01

</H3>
<STRONG>CHAOS-BASED COMPUTING</STRONG>, a fundamentally new way to perform computations by exploiting the ubiquitous phenomenon of chaos, has been demonstrated in a simulation by researchers in India and the United States (Bill Ditto, Georgia Tech, 404-894-5216). Compared to digital computation, the chaos-based technique might come closer to how the brain performs computation, and might be superior in certain tasks such as pattern recognition. The computer consists of an interconnected grid of "chaotic elements," systems such as ammonia lasers which can generate unpredictable signals even though their behavior is governed by known mathematical equations. To encode specific numbers into each element, the researchers make specific signal patterns correspond to a number and ask each element to open its connection to the rest of the grid when it generates that pattern. Sending its signal out to the grid can trigger activity in neighboring elements. To carry out specific operations such as addition, the researchers connect the elements in a certain way. An unpredictable but deterministic avalanche of activity among the elements ultimately settles down to produce an unvarying signal that corresponds to the desired answer. Having demonstrated their technique in a computer simulation, the researchers are planning to test this idea with chaotic ammonia lasers and hybrid networks of nerve cells and silicon chips. (Sinha and Ditto, Physical Review Letters, 7 September 1998.)

<p>
<STRONG>TRIPLE PHOTOIONIZATION OF LITHIUM</STRONG>, a rare process in which a single photon removes all three electrons simultaneously from nature's third-lightest atom, has been detected for the first time by a Japan-US collaboration (Ivan Sellin, University of Tennessee, 423-974-2738). Studying this process further promises deep insights into the interactions that can occur between a trio of electrons and therefore a more sophisticated understanding of the interplay between charged particles in many environments such as stars. At the Photon Factory in Japan, an intense beam of extreme-ultraviolet (EUV) photons broadsided a beam of lithium atoms; a detector then recorded the rare process by collecting Li3+ ions. In the most simplified picture of the process, an EUV photon deposits virtually all its energy into a single electron; the electron immediately shares enough energy with the other two so that they could all escape the Li atom. The three-electron interactions are relatively easy to extract from the data since the photon vanishes after striking the atom, and the heavier lithium nucleus acts merely as a sluggish "spectator." Researchers have observed triple photoionization of heavier atoms, such as neon, but such processes are typically more complicated events involving internal rearrangements of other electrons in the atom. (R. Wehlitz et al., Physical Review Letters, 31 August 1998.)

<p>
<STRONG>BROWNIAN MOTION IS CHAOTIC</STRONG>. In case one needed any more persuasion that chaos is all around us, a Brussels-MarylandUtah collaboration has for the first time demonstrated evidence for chaotic behavior in fluids at the microscopic level. The data consists of repeated viewings of a 2.5-micron particle suspended in water. A plot of the particle's position as a function of time is translated into a form which provides information about how the particle's position at one time correlates with the position at a later time. This analysis proved to bear all the hallmarks of chaotic behavior. The chief symptom of chaos is the tendency for particles that initially follow nearby trajectories to diverge quickly from one another. (P. Gaspard et al., Nature, 27 August 1998.)

On "Summary Machines": For predn. of a sequencial corpus! Say we have several "short codes": The state of the machine after it has just accepted the last bit of such a "short code", is a "summary" machine. A wtd mean of ~~these summ~~ the summ machines associ w. each of the short codes, is a better "summary machine"

A summary machine has the property that one can feed it random input, & the d.d. on its subsequent outputs will be the p.d. of continn of the original corpus. It means that search on random input is the best way to try for codes of longer sequences.

A possl. flaw in the foregg. Arg't.: (a) OSL would not be implemented, since the ~~set~~ set of short codes would not have any idea about Raggys. That would occure w. $SSZ = 2$, when the corpus was augmented by one more "sample".

(b) (related / similar to (a)) If a ragy appeard once, each time the corpus was augmented & "updated" ("updated" ≡ "summarized"), then perhaps the ragy would never be noticed!

(c) Consider binary Sayg: If the "summary" is just the rel. frequs of 0. & 1, then all other info about Raggys in that corpus will be lost. Also, any ragys of new than rel frequency, will not be able to use that part of the corpus as part of their sample — because their new ragy will not appear in the decon of that part of the corpus.

.21 Counter arg't: That ~~the summary~~ any summary machine has within it the code for the corpus, so it has all info about the corpus in it!

— Well, is .21 true?: The summary Machine has the state (Head state + work tape) of the machine after it has printed out the corpus. Hrr.; the corpus is usually not recoverable from that state.

So, in General, the "update" is almost never a complete dcm of the corpus! So usually we will want several "summaries", including the uncoded corpus.

.27 By the way, how did I do "3 summary Machines" for unusual data? — Perhaps it was simply the (unconditional) stock Ragy. That generated the observed set of objects. — Same "summarizing machine" in the "Mad corpus" problem. (!). — Then it seems like a Discrete Universal Df.

.30 ↳ for continn. of the objects seems to be finite, we have ~~the~~ probs. that are the sum of an (usually infinite) no. of (rational) binary fractions. — so the sums can be irrational.

.33 On OSL: In OSL, a summary code, that would be useful for OSL, codes the previous (not one shot data) corpus in a new-optimum way: since it defines an object that has occurd only once.

.36 **Q:** Does .33 ff generalize to ragys other than Bernoulli: esp. type Ragys? → [37.03]

Q: in Lsrch do we also miss out (miss out) if we use a pd that would not implement OSL?

[37.01]

N 22.98 TM: Levin CHAOTIC H.W. for TM

01: 133.40 I t. ~~[illegible]~~ letter, I was able to define input O.F., but for outputs I decided to have it stop: which is very constraining (i.e. "Extension Complexity") & not so good for "Summarizing". It would be good if I could get t. General Unc to do Re "~~[illegible]~~" Universal Continuous D.F., "as opposed to t. discrete d.f.

-02: See (35.27-.30 for discussion t. Discrete r.s. Continuous D.F. in t. "Mxd Corpus problem".

D12.98: I got copy of Sinha, Ditto paper: also 1992 paper by some Guys on some more General Forms in this Area. None refer to Chris Moore.

N2298 TM Summarizing (P.D.'s Machines) ✱ Kinput Muxer (36)

One (apparent) goal. soln to Barn, OSL problem was that idea of considering all possl. Kerem definitions in getting t. pc of all possl. continns. of t. corpus.

$\sum_i \frac{2^i b^{n-i}}{i!(n-2)!}\, n!$

.03 135.36 : HA! ☺ This may be a justifctn. of retaining old Theorems that didn't work so Good as t. up-to-now "Best". Here, also, we retain them because they may be able to deal w. as yet unseen inaccuracies in t. future corpus.

regy.

So: Simple "Summary Machines" are not able to do OSL à (as) other impt. regy discovery methods. Still, "Summary machines" are easy to make (compute) à may be able to do useful stuff, tho "Imperfectly".

.10 Consider a "simple" sequencial predn (/codng) problem. We code a short bit of corpus w. Lsrch: use retain best m. pgms. To code t. next chunk of corpus, we use m defnt. "Summary Machines" for each, we do a simple Lsrch for t. continn.: we retain t. top M pgms, gram and Loop to α → (see 140.01-.04 for an impt Trim. on this.

sum. → SUM, sums.

After we've coded a chunk, we have these M best codes for t. entire corpus!

When we do Lsrch starting on a summarizing machine of code length l_i,

.17 we do Lsrch w. an initial pc $\alpha\, 2^{-l_i}$.

.10-.17 is very easy to pgm! (If we used a simple kind? kind of trunc-/like unc) would it be easy if I used my (somewhat complex) "Functional Imp".?

If one uses very large m, it could be same OSL: (see 135.33): Very large m," means I could after defining things that only occurred once — This this seems like grossly-inefficient way to do it!

.27 → Look at my soln. of t. Alg notation Lrng Problem (ANL) — See if it can be put into t. format .10-.17;

.29 → Do same thing w. my "GA-like" soln of t. 11 input Muxer problem.

Q: Just what is "Updating"! It is t. modifn. of t. P.D. so it will be better for future (Problems/corpus coding). How is "updating" done in .27, .29?

Two Aspects of t. P.D. (1) Unconditional: It gives t. pc of each finite object — either a finite string or t. finite prefix of an infinite string. (2) Conditional: Condition is usually t. (known) prefix of a finite or infinite string; we want t. P.D. of t. continn. of that prefix.

.35 Hmm! .29 is interesting! For successive runs, t. corpus is t. same, → So this is really different from what I had in mind for "corpus lrng" or "pat sring." → 140.01

.36 (SN) In t. 11 input muxer, I had t. idea that as t. "signal amplitude" would ↓. — well, it does ↓, per sample, but t. SSE ↑ rapidly w. n — so for t. total corpus, t. "signal" (i.e. t. amt. of compression obtained) may be consistent or ↑ w. n. — Hur as n ↑, t. cc per bit ↑ enormously!

N3098 TM : MCT Mixd corpus Thm. (convergence proofs - & theorem itself.) Gossl, IMPT Ganzy .17

I (recently) was a bit confused about the definition of "convergence" & meaning of "error". This Q. comes up in the simple corpus w. pure set of finite objects.

T. Q. was: Say we are computing the error int pc of the nth bit of the kth object.

What is the set of finite objects that is used in its data?

Answer: "none": "Since" the p.c.s of the objects are indep. of ordering.

We have a p.d. that induces a p.d. on every object or partial object. T. successive p.c.'s of successive bits of an object are dependent only on the previous bits of that object.

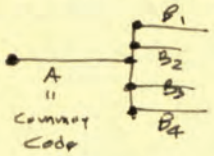
T. forp. is true for Mxd corpuses in general.

T. basic P.d. (for ALP) is the wtd. sum of all p.d., w. wts of each P.D. & pc of entire corpus - for that P.D.

17 (Diggs) A possibly IMPT. Ganzy or MCT!

Consider not full MTC, but just finite set of finite objects".

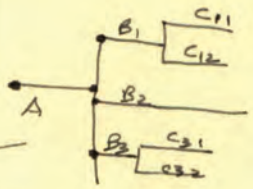
Normally in MCT, these finite objects all have some common code, & then each object has its own additional code:



Bi = individual additional codes.

.20

could we not have: We have 5 object items, w. differing amounts of common code.



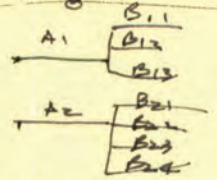
A B1 C11 is code for 1 object
A B1 C12 " " " "
A B2
A B3 C31
A B3 C32

for example: 2 time series could have common segments: Say they were both stock prices on the same set of days. The "usual" way of dealing w. this is to have a single Vector time series. — But .17 may be a more general, alternative way.

T. Ganzy. to full MCT seems clear.

In general, one could take any set or subset of things in the Mxd corpus, & make a tree showing which codes for which "thing" was common to the code for which other "things".

.32 Another simple case:



Essentially 2 indep corpi

697C
Dash Jet

In .32 (R, L) we don't begin printing until we start reading B_{ij}

In .20 (R, L) we print only when B_i or C_{ij} appears,

so in .20 R A, B_i are insts on how to construct a machine.

spec.
→143.01
144.01

D.2.98 TM: LOGIC!

A tentative idea! That "deductive logic" need only be consi, in order to be usable! In all almost all. impt. cases, we will use inductive logic (except for MoM).

.03 By logic I mean: Given a set of strings that are "true"; a formula to tell if certain other strings are "true". I guess to be of interest, the system must be able to tell if certain strings are "false" (= "not true") — i.e. one can't use the formula to show them True. [The ("not true" ≡ not derivable by the system) ≡ false ≡] is not obviously nasty. Must we use 2 valued logic?] — Perhaps at least: True, false, undecidable.

I think logic is usually concerned not w. T/F of strings, but only w. T./F of WWF's (well formed formulas). If wwf's are a CFL, we can probably assign integers to every wff. [1 to 1 (↔) Perhaps this is what Gödel did ..

It would seem that the most impt. idea in a wwf formalism would be which strings are the negations of which.. Other than that, a consistency, any implication formula would be o.k. see .03

In order be "sure" of deductive logic's conclusions, one must be sure the logic is consi. In general this is impossl. One might know that a logic is "consi of depth < n". So, all demonstrations of depth 4 are consi. Infinite recursion can be of finite depth, hvr. — We are only concerned w use of theorems — in general for proofs for humans — these are of finite depth.

.01: spec 137.35) What we did (& what GA does) is to find a (ragy compressing) of the corpus, then use the resulting compression implied summarizing p.d., to look for more compressions (= ragys) in the same corpus. Since any ragy is another parallel code for the corpus, one still uses the old

.04 search conds, but one has some newer parallel ones of hyer priority. This is an impr. trim "on 137.10 ..."

(SN) Is this related to "Sequencial Testing" ?

In .01-.04 : Say one finds a new ragy in the rst subcorpus : Would it not be well to recode the entire corpus wrt this new ragy ? Well, Yes & No ! If the "Summarizing Machine" did a complete job, there would be no point in looking at the previous, uncoded, corpus — or touching it in any way ! However, if (as is usually the case) the summary is incomplete, the new ragy could very well ↓ rc of previous corpus considerably (?).

D1598 A Main Problem : Look at various Lrng methods (≥ 141, ANL, ...) See just how they do "Summarizing". — Maybe also any "GA" (as I solved "InputMuxer").

I Guess "Summarizing" & "Updating" are almost the same. Except maybe for OSL !

In better good code. ~ GA. "InputMuxer" problem : We seem to update as soon as we detect somewhat ≈ the "Coding & Decoding" idea.

Impt., Very General idea .04 - .09

.01 | 09.40 Anice soln. to Defining $C \equiv A^{\wedge}B$: I was worried because $C \equiv A \cdot B$ implied that one would have to convert all AB pair to C for the defn. to work, & in the real world, a fair fraction of the A+B pairs would be random adjacencies of A & B.

.04 A V.G., "correct" way to deal w. the problem: When we define $C \equiv A \cdot B$, We do this not for 1 code string, but for all 11 code strings; So the total ↑ in pc for the entire set of 11 codes, is what we are concerned w..

.09 [I'm not certain what, if any, this idea is really new, but it seems clear that I did not apply it to the Z 141 problem!]

$N \equiv N_A + N_B$

.10 Anyway if A has occurred N_A times, B, N_B times; C, N_C times: [$N_C \leq N_A$ / $N_C \leq N_B$]

Then if we code the string using n_A A's : n_B B's : n_C C's :, the pc

the code is $\left(\frac{n_A}{N}\right)^{n_A} \cdot \left(\frac{n_B}{N}\right)^{n_A} \cdot \left(\frac{n_C}{N}\right)^{n_C} \cdot C_C$ | C_C is the pc of defining C.

$n_A + n_B + n_C = N$ also

If C_C were constant for all N_A, N_B values (it is not) then the total pc of the corpus using this defn would be

.20 $n_C = N_C$ upper limit ... $n_C = 0$ $\left[\left(\frac{N_A - n_C}{N}\right)^{N_A - n_C} \cdot \left(\frac{N_B - n_C}{N}\right)^{N_B - n_C} \cdot \left(\frac{n_C}{N}\right)^{n_C} \cdot C_C\right]$

If C_C were constant (it is not), we could take it out of the Σ.

· If .20 is significantly > $\left(\frac{N_A}{N}\right)^{N_A} \cdot \left(\frac{N_B}{N}\right)^{N_B}$ then $C \equiv A \cdot B$ is a useful defn.

The "summand" in .20 has a peak for a certain C_C. (98.01 to 98.20 depending on the pair) the peak %. (probably may)

on "C_C" This is usually easy to do! $N_A \to N_A + 1$, $N_B \to N_B + 1$, $N_C \to N_C + 1$:

T. formula of .20 is inexact, anyway! → it should use factorials: like $\frac{N_A}{N!}$ etc. I think Sol 64 b explains it in detail — also how to define $D = A \cdot C$, say. — it's pc of definition.

.29 I had first that defining D would be a made easier if I first had narrowed down which how many C's to define, & the parsings: I'm now not so sure it makes much difference; — i.e. on can define C & D! Then one has many many parsings. Usually one is mainly interested in prediction, [so only the parsings involving parts of the augmented corpus are critical; N.B.

.33 One sums over all other parsings. → 142.01

It should be possible to compute the probty of the next bit being A, or being B, on the basis of N_A, N_B & N_C only — summing over all possible parsings.

Also: Can one ask: what is the probty of a particular adjacent A·B pair being really a C? Yes! To find the answer: take the total wt. of all parsings of the data (including the parsing that does not define C) that parse as "A, B"; compare the total wt. of parsings that code it as "C"

Great Break thru: Serial or ll codes either or both are legit for paying for "definitions" (.07)

11 min.

.01: 141.33 Re 141.29 ff on Defining "D = BC, say: It is probly possl. to define t. "state" of t. system after C has been defined, so one has all t. params needed to define D = BC, & to determine pc's of continus of corpus involving D.. — These params may be approximate. — i.e. They may give just approxt. results.

In fact, it may be possl. to characterize t. state of t. system of 141.10 w. a few params & make pc of a C = A.B a function of these params.

.07 : 141.04 -.09 Maybe a GREAT BREAKTHRU for me! either I never realized it or I forgot it! Anyway: T. idea is that BOTH serial & Parallel codes can contribute to paying t. cost of coding a definition. T. idea is that t. definition should ↑ t. pc. of t. entire set of codes. A very simple example is t. coding of a Bernoulli symbol that has proby α by an equivt code symbol" of pc = exactly α (no need for "Arith. Coding".). All symbols are coded by a "chunk" of K bits (K → ∞ in a/d proof). A alphabet item of pc = α is coded by $(2^k \cdot \alpha)$ of t. 2^k chunks of length k.

(SN) T. Mixd Corpus Theorem seems related to this "Great Bkthru in Z141", But it may be a counter example! — On second thot, they seem much difrnt. In Z141, we have lots of codes in ll! In Mx Corp Thrm, we have t. "2 part codes": part 1 is t. "definition" of a. P.d. or Machine — corresponding to t. defn. of a & k tup in Z141; Hvr. in M x C Thrm: We take t. product of pc's of various objects & partial objects. In Z141 we add t. ll p.c.'s: So they really look much difrnt! (No Counterexample!)

Def MCT

1) T. "summarizn. machine" is always an approxn & "OSL" problem is one extreme example.

2) However we summarize "Z" t. state of system w. "C" defined so D can be defined easily

3) In general "Summaries" all summaries are approxns & lose inf. info. T. only way to do it w.o. approxn is to use an exact non-loss code. Usually we code w. 2 parts: Part 1 followed by a pure random sequence

11:11 → 11:19
20 2.75
27 No NN Vermont Conn. .22
N.Y. R.I.

• 13.95"
~23'
4.33 L ey/sec

D998 TM EB(L/G) (Explainn. based (Genzn) Lrng)!

~~[illegible]~~ Previous ref exists in 1998

This may be t. idea: There is some domain (area of expertise) where t. system has lots of domain theory. (≡ Good models, that often work).

It is Given a problem — which it solves, using its models: Hvr, it takes a fair amt. of cc to solve this problem. (say lots of "reasoning", search, etc).

So It "generalizes" this soln., so a soln. will be available in t. future at lower cc.

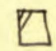
I guess t. purpose here, of t. genzn, is to save cc, rather than to do Induction.

The "Genzn" is done by examining t. problem & its soln. How can we relax/certain specs. of t. problem & still have t. same soln. trace "be valid" — so we can easily decide what t. soln. is.? For INV. probs. we can always tell "easily" if a proposed soln. is correct. For OZ probs: well we can get t. Genz. — By defn. of "INV" problem

---

An example: A proof that a Rt △ has certain properties; looking at proof, it is clear that we can remove t. "Rt" constraint & proof still works. — So we genz t. thrm to be true of all △.

probably phone call to Hy Sroore



I think that an impt. part of EBC is that t. "explanation" of what's observed helps tell how t. observation (problem soln. regularity...) can be genzd.

New invastors → $1MMM.

Fund: Renaissance Rassmus $3 B — Hy Sroore at Potter. or Palr. $62M now in the 2 ppls of Hy Sroore

Some Refs! AAAI 1990 conf vol II

.20 pp 803 contrasts EBL & SBL — Similarity.

.22 821 EBL & "chunking" — Mentions Wolf's induction ≈ 2 R's: looks reasonably!

Proc 6th workshop ML; 1989; p 2–4 Pat Langley: This is somewhat clear.

In .20 EBL is said to need small SSZ because of all t. constraints in its models;

→ SBL (Similarity) has few constraints so it needs larger SSZ.

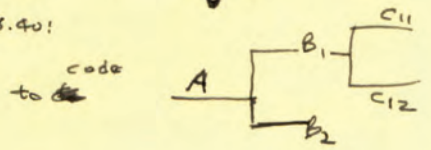
T. paper, .20 Tries to mix t. 2 ideas by making a system w. constraints stronger than SBL & weaker than EBL.

In .22 Langley refers to Wolff's work on (≈ Z141) lrng as "starting w. SBL & then moving toward EBL as t. domain theory was built up." So putting Wolf's work, (or my most recent work on Z141 (upto 140.90) in Mxd corpus lrng" form might clarify EBL, SBL & "Mxd corpus lrng." The idea is that as pc's approach 0 & 1, we approach "ebl".

— Conjecture: That while certain pc → 0 or 1 (after a large SSZ) other pc's in t. "like" problem area do not. (These could be/t. decision to try a linear eq. approxn.: Then solving t. linear equ. would use pc's of ≈ 1 & 0.)

D1698 $ TM : MCT

.01 : 138.40



we need ~ |A| + (|B1| + |C11| + |C12| +
+ |B2| + punctuation
to code t. entire corpus.

to code w.o. this graph we'd need (|A| + (B1^C11|) + (B1^C11|) } using ordinary codings of
+ (B2| unordered sets.

and ~~|A + B1|~~ |A + B1 + C11| + (A + B1 + C12| + (A + B2| to code t. corpus w.o. using common codings.

Could we get t. same results w.o. t. tree structure? i.e. within A, we have B1. C11 & C12 both "refer to" B1 (within C) before we start C11 or C12.

W. o. t. tree structure we could get more general sharing of dcrn resources that we could get w. t. tree structure.

A possl. advantage of t. tree str. (when we can use it) is that it reduces t. cost of referring to sections of relevant codes. W.o. t. tree structure, we need to tell what parts of A are relevant to our dcrn. Using t. tree str., this is not necy. or costs less.

.20 SN In ~~a~~ Soln. of OZ probs via Lsrch, we need not have a condl. pd: A uncondl. pd. is fine! Woops ~~False~~! — we do need a condl. pd. — That looks at t. H.C. problem & decides which H.C. method is best for it. I thot that since t. H.C. method "looks at t. problem", t. condl. p.d. would be unnecy. — But NOT SO!

D1898 TM

UNSOLVD PROBS:

(1) Sol78 Thm 3 : for radix >2 : inconsi results obtaind !

(2) SOY Problem ! 2 d.f.s of mean 0, var 1 , v.s. 1 d.f. of mean .1 is rare ∅ .

.07 (3) ~~Distribution~~ Prob by distribns. w. cc : / apparently Related to Chaos problem, Oz problem is.
cc = f(t) (f(t) is ↓ funct), Jargon's machine (fig. 37 - 150.08) ; see (TM '99) 2.25 for possl. Breakthru !

(4) Summarization Problem : What are Good ways to summarize a "current P.B." (is R↑ related to (3) (.07)
How to balance hy pc of t-entries corpus v.s. cc of obtaining pc's. .?

6888 TM: Genl. Path

Note pp 359 & 358 are in inverted order

01: 357.40: 11) An impt. Q: [redacted] Can I get TM to work on TM_2 ~~the~~ by-products in a natural way? — i.e. apply info on pgm. optimization problems to its own program optimization probs. w.o. having to "tell" it to do so?

12) ~~Related to~~ 11) Would TM give hy pc's to methods that were of small cc? (in Big Pachat)

See 358.01 (Ideas) for some discussion. Would it try to revise old prob. solns. 358.13 is reasonable!

so as to find methods of lower cc, hyer pc? (one very best soln. is a numerical A.H. soln.... but it will have very low pc ... probably too low to be within cc of TM ~~+ precision should be always kept~~ so that A.H solns. are essentially impossl.)

13) I'd like TM able to "Memorize" complex solns. to certain problems — they later be able to (usefully) try to "factor" to solns. to give them more pc, & to be able to usefully use the "factored" new concs.

.18 14) There is the possy that the search strategy can remain constant. That I don't even need Z141! This idea is related to 357.01, 357.24, 50% soln. (1·16·91 ... I don't get this!)

6.7.88 TM IDEAS

7588 Index 3.05

.01;247.40 ① I was concerned about TM not automatically developing methods to work probs that lead small cc à give hy pc's to Prem.

I think this is automatic! [redacted] Small cc methods will be more likely to solve problems. One can try out more of them per unit time. They will end up being "popular" à get assigned hy pc's. [This idea needs more careful thought, but..... but be sure to write down & reasoning on this.... it's a bit hazy in my mind at present!]

Is this an example (pos or neg.?!): TM find multiplication to be a useful operation, but it does take a lot of time (he does it by repeated addition). Would he be motivated to try to find a faster way to do it? Would he (a pri) have any idea that this particular "common sub. of operations" might be done much more rapidly?

.12 → Related Q: TM solves a prob. w. cc = 1 min; w. cc = 5 Min it obtains much faster soln. but w. much lower pc.

Hvr., This soln. at reasonable pc level is now available for "SSZ=1" learning for a new problem like

.14 ↑. first one! Say ↑. 2nd problem was very similar to ↑. first, so it used essentially ↑. same cc (symbolic) soln.

[redacted] Seems to me must whether or not Z141 is up to it, that I should be able to get ↑ ≈

[redacted] CBI to react properly to ↑. problems of .12 – .13. (Re Z141 does seem to be the right approach!)

---

18 → 1.18.91 Example of "face" soln. of very low pc: Given an eqn. w. low pc. coeffs.

T. no. that solves it is a low pc., low cc soln.

6988 TM: RADTS

(cont.)

.01: 360.909) That would be useful elsewhere. T. considerations of 360.30 (Re: GPS) suggest that there will be lots of "monster learning" in other domains!

10) A big Q is: what order of concept & exposition should be used. Should ① ≈ GPS be learned along w. elementary algebra, or should ② all algebra & a lot of other things be learned partly by "memorization": — from have GPS be discovered as an "interlocking principle". → (.19)

Very Impt.

Old view was ②: perhaps very difficult to write the appropriate heavy. is ①: see .19 item 13

11) One way to learn GPS is that one has at their disposal, this idea of "difference" betw. two things & one's Goal: perhaps memorize this/our idea use it as a primitive.

12) This seems to be departing from the "Ruf & dirty" idea of 357.08, .28 — lack of criticality of the trg. seq. → (.19)

(.19) → 13) From (10) & (12) I guess that I'd want to start w. lots of "memorization". Learn to do lots of tasks/ that could be learned by GPS. Then after those tasks are learned, have it discover GPS to reduce each of the pc's of all previous solns. To do this would for me, be v.g., because it would involve some "hyper order learning", perhaps. It would seem to involve "factoring" the original solns. — a having all these solns. have a "common factor", i.e. GPS could be a common way to solve them.

14) An interesting Q: to what extent would TM have to have experience w. problems like "discovering GPS", before it could discover GPS? Again, I think the problem may be transferring into from one level (TM$_1$) to another level (TM$_2$).

15) "Memorization" can mean: (a problem is solved by TM) something is learned, but the cc [redacted] given for the task was not large enuf for TM to discover the [redacted] useful "factors" of the cases that were adequate for the soln>.

cc? (not pc?) [I probly means "CB" "Comprn. find → cc]

16) Actually, GPS doesn't have to be discovered all at once. Parts of GPS can be discovered & be useful. E.g. Certain Ob-ops are found to be useful. Later, as part of GPS, the Ob becomes an observed difference, & the assoc. op is the operator that reduces that difference. At first, TM discovers this ob-op in a particular domain of probs, then later TM discovers this sort of thing is common to many domains of probs, so it can get rid of all pc by using this common factor in the code.

362.1.01

6.10.88 TM: 586: Time sharing w. 1 CPU, (20MHz) w. ≥ 1 CPU.

.01 : A possl. way to do Time sharing (Czech) w. many // machines: First, if we only had 1 machine, time share it by spending a fixed time quantum size δ on each task, indep of its pc — but w. frequency which one spends w. task is ∝ pc of that task. We arrange so that δ >> t. time needed to switch betw. tasks, so there is ≈0 cc lost thru "administration of T.s".

[598 Here, a problem is that this gives us an effective cutoff of low pc conds. There are a very large no. of "low pc conds". Cutoff depends on δ. So we wouldn't try conds w. their pc. T < δ: so as T ↑, we ↑ no. of conds tried.]

Say t. time quantum is δ: each task has a counter. Every δ interval, each task has its counter incremented. The "threshold" for each task is (pc of that task)$^{-1}$ (number)

Each task has an task id. number.

After CPU has worked on task no. i, it looks at task i+1 to see if its counter is > its threshold. If it is, (if it is not) it decrements t. counter by t. threshold value, & works on that task for time δ — then goes to next task & looks at (counter − threshold).

it goes to next task & looks at (counter − threshold), etc.

One nice trick (fast method) is that if a task has been "overdue" for a while, before its turn, then this overdue amt. makes it worked on earlier next time.

Hw, how much time is spent updating counters? — there are ways of them.

.20 To do this time shared (w. > 1 machine) we could have a bunch of CPU's. Each does a task in // w. others. When done, it looks for t. next task it could find that's "overdue" (i.e. (Threshold) > (counter)). Hw. all CPU's finish δ tasks simultly.

T. problem here is how memory is shared betw. CPU's & how t. state of a task (after a CPU has just finished its δ unit) is transfer'd to t. next CPU to work on t. task

.28 → One not-so-efficient approach: Assign each CPU a set of indep tasks. Do this ∋ t. Σ pc_i is ≈ same for all CPU's. Each CPU has its own indep RAM.

.30 → A trouble in .202 & .28: Sometimes, when a CPU will look for a task to work on but find there are none overdue! Another possy is that we have unknown capacity of CPUs, so as time goes on, t. "overdue" level of many tasks gets larger & larger! One way to deal w. this: The threshold of tasks = $\frac{2}{pc_i}$. The value of 2 is being constantly corrected so that .30 & .31 occur infrequently: I.e. either kind of event pushes 2 in a direction to reduce its likelyhood of that kind of event.

→ to make .28 more efficient: when a CPU is spending too much time on its tasks (since then Σ pc_i is ≈ as large as other CPU's) then one or more tasks may be transferred to it from other CPU's that are spending too little time on their tasks. We probly don't have to consider transferring unless t. Σ pc_i for 2 CPU's differs by a factor of > 2, say.

See also 1 page note! D2785 on use of disc for Czech,

01 140.40 T. Summarizer Machine is "any" usable form of (conditional) P.D. As a ganzed P.D., we can get pc values of ↑ accuracy from it, but at greater cc. In general, t. relationship of cc to pc precision is "unknowable", but we can probably approximate it w. finite CB.

T. main idea is, we want best Lsrch results. — i.e. good pc & small cc.

We could try ~~use~~ varying amounts of cc (a scanning ~~and~~ optimizing sense) & use th. variation in times to solve INV probs, to select best cc to use.

.07 The soln. to t. "Summarizer Machine" feat I had in "2 kinds..." seems ok. except

.08 for OSL. We may not need t. "OSL" part of t. true P.D. for Lsrch(;) . → (52.01

So we have 2 applns of t. "Grand P.D.": (1) for Lsrch on problems (2) for output to RW — to make decisions, eval. I guess I want to optimize it for Lsrch. so if Lsrch doesn't need OSL, then it should be easy! (Well: I'm not so sure Lsrch doesn't need t. OSL part!)

17 [A Major problem in "Summarizer": We want to "improve t. P.D.": we want ↑ pc for t. entire corpus, but we are also concerned w. cc of t. pc evalns. T. "goodness" of a PD has 2 components: pc of corpus & cc of outputs): So we can certainly formally order t. P.D.'s: — but we need a "linear ordering" to guide "updating" (= "improvement") of t. pc for t. entire corpus

This Q. of just how best to do t. "updating" seems to be a major problem/bottleneck in t. entire project.

Actually, what we are really interested in is hyper low cc for extensions of t. (general) corpus. Perhaps Low Least $\frac{cc}{pc}$ would be what we want.

Consider t. actual applications of t. "updating": We have a Big Mixed corpus! (Neglecting OSC) we find 10 short size items of t. corpus. We are then given a "partial object". We are required to find (likely) continuations of that "partial object" w.r.t. t. Mixed corpus.

.27 ~~General~~ A Prize we aim for! Given t. 10 short Mixed corpus items, to devise a stoch operator, ⇒ given any new partial object, t. output is (in some form)

.29 a p.d. of continuations of that partial object.

An "update" is a state of TM that makes .27–.29 relatively easy to do.

If, in .27–.29, t. new partial object is a Big Question, it will be messy to spend much cc analysing that Q.

.34 Say C is t. (Mixed) corpus; B a partial object (≡ a segment of C). To get .36 to Best 10 codes for CB; we might first use sequential codes of B, then "coding à recoding" of CB.

T. General "Mixed corpus" updating problem looks pretty much t. same whether (.34–.36) it's a Mixed corpus or a sequential corpus. For each augmentation of t. Corpus, we want to get t. best set of codes (or equiv) for t. augmented corpus, in t. available time.

→ (99) 3.01

6.13.88 TM: Rodts.

dynamics associates

$$\begin{pmatrix} A \supset B \rightarrow \sim B \supset \sim A \\ \sim P \supset Q \rightarrow \sim Q \supset \sim \sim A \end{pmatrix}$$

B→~B means "toggling" of B's "~"
so B→~B implies
~B→B.

2

"Human Problem Solving"

.01:362.1.40] However, on the whole, the data in HPS seems interesting and may prove very useful! I can probly make much (more complex) better models of brain human S's than NiS did, because of the use of CJS!

In the data of HPS pp 593-6, there is improvement in prob solving ability as the exp. continues. So this is a key area of interest.

So this may be a very feasible way to do the exps! HPS is one source of both problems & data on how humans solve them.

Also, I could use Pat Langley's probs (& data ?) on discovery of scientific laws.

In the case of HPS examples: I watch this human solve the problem & I postulate a certain # prims or concepts that would give such a soln. Note that normally, both prob. solving & updating are occurring during Phase/human solns & so I will have to use a TM model that Time shares on updating & solving along w. the regular work on tools.

147.01
→ 9&TM: ~~146.01~~

6.13.88 TM: RADTS.

dynamics associates

.01,362.40 Perhaps a main problem is to find a set of parallel problems, that are nominally in diferent domains, but have similar structures, so I can try "Transfer learning" as soon as possl. I'd like to try this "Meta learning" as soon as possl. Ideally, w.o. use of a TM$_2$ formalism.

N.B.S. "Human prob.Slvg" may have some useful ideas. I have avoided reading their stuff because of their very bad methodological errors — but it's likely that I can get usable problems from them.

A possl. source of N probs: A list of 11 problems they used for "GPS & Generality":

Another possl. is A. probs used on Humans in "Human prob.Slvg":

I think they did a lot w. "Cryptarithmetic"

Def The "Missionary & Cannibals" prob. (MCP)

One way a human might solve it: play around w. atoms, make up "interacting" terms. Keep eyes open for relevant theorems or conjs.

This is probly true for all 11 problems! — so good possy of transfer!

Another possy is to get the TM 11 probs (of all 11 types) to work on "simultaneously" using Time Sharing.

In Hum. prob Solving: 3 Domains!

1) Cryptarithm. 260pp	2) Logic 152pp	3) Chess. 125pp	"Theory" 537pp

Also, do elementary Alg. learning (How to solve linear eqns).

My present Idea is to have TM learn to solve each kind of problem in a "general" way as possl. — (by using largest possl chunks in A. Exp. Seq). Then, by having TM note similarities in diferent probs, he devises better ways to have discovered A. old solns, & to discover solns to new probs

? —> Tic Tac Toe as an elementary problem (?)

Def In (GPS) (Human Prob.Slvng), there are short passages on learning, that could be of interest. (See index) pp593-6 are of interest, because they give a temporal order for acquisition of impt. learning concepts! (Perhaps unfortly) N.B.S have GPS in mind as a model, so it probly biases them gothereby of info

6888 TM: **RADTS** Rul i dirty tny. Seq.

.01 ·393.40 **1)** Say / top Goal is soln. of linear Equs.: let us work backwards:

first define top goal: T. equ. to be solvd is a linear eq. but it is in a form such that various standard operations have to be done to get it into t. form ax+b=0. T. eq. is gn in PRN or RPN or conventional alg. notation (e.g. $(3x+7)(2x-3)=5$) — hvr; see (.19)

.05 **2)** So TM. Wants to "simplify" t. equ! — get it into a form that he knows he can solve. TM has idea that a simpler looking eq. is usually closer to soln. than a complex looking one; A TM has some natural measures of complexity: e.g. no. of terms — (perhaps certain terms give gn. more-complexity than others).

.12 3) In simplifying TM will start w. primitive operations — then learn pairs ... (n-tuples) of operations that are u.g. These "operations" are in t. "ob. op" algebra — i.e. they involve operations that are contingent on observations. We can have varieties of: sequs of obs; sequs of ops; sequs of ob, op mixtures.

.19 4) Somehow we have to convey to TM, that t. problem is to find x ∋ that equ. is true. — We can use floating pt., & have an external error criterion, so t. eq. doesn't have to be solvd exactly. — so "almost always", we can assume infinite precision.

.22 (TM can "watch" t. eq. that checks its cand. solns.)

5) After learning to solve equs. w. numerical coifs., it should be able to do it for literal coifs.

6) Some info we expect TM to have in order to solve (.01)

a) Some possibly useful info: that certain xfmns of t. equ. don't lose info, so they are equivt. to it, & one can save t. new equ. instead of any of t. old xfmns. — (tho t. original may be special).

b) Some operations — like discarding old equivt. xfmns — can save RAM (& ↓ size of srch space: ∴ ↑ pc of soln)

c)

7) It may be that t. hours of .05 — .18 (6),(3)) aren't normally required until TM is quite "Mature": that t. solving of .01 usually uses a lot of "Memorization" — & that after these memorized things aren't factored until later, when TM requires t. hours of .05 — .18.

.30 8) A way to acquire t. hours of .05 — .18! Discover ≅ GPS: find various "differences" & assoc ops that reduce them (see op. ob algebra of (.12)) Attempts to get set of "orth" distances & a/o set of "orth" operators that reduce t. differences individually. With this GPS model, TM will be able to work other GPS problems (like symbolic integration, Missionary & Cannibals, Tower of Hanoi — (But, presumably, much better than GPS).

9) Why (.30) is very encouraging: I was afraid that learning to do simple Algebra wouldn't have enuf assoc. probs. to be "interesting" in t. sense of learning impt. heurs

The system as of now: (≈ 5 86):

.01 1) The initial ~~set of cons~~ primitive set of concs need be universal, but that's all. We can put all t. info we like into t. prim. set. If we don't have enuf or its wrong emphasis, we can correct this later in t. Trg seq. (or, just add to t. primitive set.)

2) The search algm. is fixed. We can do T ← 2T or timesharing in various ways, but this is not very impt. difrnce. Hvr., see 359.18

.09 3) Trg seq: Also not very critical. If we make mistakes these can be corrected by later training. → ([illegible] (.28))

4) One Goal is to get machine ~~[illegible]~~ smart enuf so it can usefully work on under sets of problems (using time sharing). (See 247.17 — 40 for some vaguely related discn. — also see (.20))!

5) I think that as one adds probs to t. T.S., TM keeps working on old probs, but there is some cut off criterion for branches based on total time worked on it ~~[illegible]~~ present pc.

.20 6) It may be that if, even w. a rather primitive TM, (→ perhaps even in t. infant TM, use a crossing multiples!) one always gives it about 5 problems to work on simultly — that one doesn't have to be too careful in devising trg seqs — just so long as one gives it lots of time to work on early probs (even if they are **easy** probs, a large cB will keep t. machine working on possl. improvements).

.24 7) I don't know just how necy t. TM, 2TM, thing is. If it is, I would simply have T.M. timeshare 1/3 of its time on TM_2 type probs — so ~~[illegible]~~ "updating" would be going on continuously.

.28 8) Because of t. lack of criticality of t. Trg. seq., I could just take (any old / some) algebra problem, then work backward in a "conceptual pathway" to ~~[illegible]~~ a suitable primitive set of concs. Try this for several problems & see at what points the "conceptual pathways" joint in their way to t. primitives [1998: Thereby giving a strong set of near primitive concs.]

9) The trg seq. should be first done in English! Perhaps first see if human could follow it. Then get cjs's for a machine: Then insert needed concepts & examples to get cJs down to acceptable sizes.

6888 10) A ~~[illegible]~~ possl. Bottleneck that I haven't that about too much (tho there is a Ref: 111.01) is Making up probs that use certain concs. in their solns. Also how many examples of each prob. to give. Using enuf signift. figs. in numbers in probs can elim. possy of A.H. solns. Perhaps enuf signift. figs can give effectively large sez!

359.19 on Back of this sheet
360.13 ~~[illegible] 359.15~~

D2095 Ruth RADTS (Ruth's dirty TS) [Rel. IMPORT of Various Sub Corpi (11)]

.01: 88TM 362.2.90: Ibid 357.01 – 362.2.90 is an examination of a maximally simple (as of that time) TM & its TSQ. At that time two problems that I was concerned w. — First I feel I now understand much better: (1) $TM_1 = TM_2$ structure, & system (2) How ideas in one area of Technology get used in another area. Michael Reed 1988 dissertation was involved w. these Q's:

insert plot. — last point — extreme bottom. — zippo.

The $TM_1 = TM_2$ problem is treated by regarding updating as a ordinary OZ problem. The problem of relative wts. of Strings to OZ probs (& perhaps mixtures w. INV probs) does not occur in the original form. The all ever Goal is simply to find best redos for the entire corpus of Inv. & OZ probs.

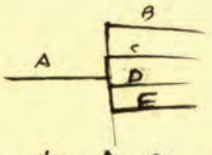
.11 There is a "rub," however... Certain Probs are more important Than others, so I (think) I'd rather ↑ pc's in certain parts of the Corpus rather than others. Could this be regarded as a special case of the General Problem of What parts of R.W. to integrate into one's Corpus? — a/c the problem "what direction should MathR go in?"

A very long Time: Various or a large Sub corpus: — These could give more ↓ in boost/cc than short sections of compact Sub corpus

If I don't want an excessively general I.M., but one that can help me solve what I Rm/ are important probs, — Then I can give TM these wts. → It may be possible for TM to assign these wts as a result of an induction problem: At first I shall

.20 In General ("TM_2" (T. updating proc.) will have the problem of deciding how much (relative) time to spend on improving the pc's of various parts of the corpus. Say "A" is the common code, & B, C, D, E are redos for separate finite objects or parts of finite or infinite objects. I can ↑ pc of entire corpus by working on any of the B thru E problems, or work on them jointly,

TM wts of many probs — Often eventually TM sees pattern in the ass-ignments set



.26 to ↑ pc of A.

In General is .20 – .26 a variety of "What to work on Next" problem? Its a generalization of the WTWON problem: Instead of a yes/no soln, we have a "soft" Gen. ~~Probability~~ of ↑ of total pc.

.01

.30 However another idea from [ibid]: In learning to solve eqns., I originally did this by treating an $x \rightarrow 3x+7=2$. The idea that one could pretend one knew what x was, & manipulate expressions containing it, was a major heuristic breakthru breakthru. Then, learning a growing set of x forms, for manipulation added to heuristic power.

.32

Now .30 is much different from the TSQ's that I've written for Algebra. I'd like to be able to get TM to do .30 ff, because it's close to how I think about problems & is (on some level) easier to teach TM. For .30, I am (apparently) able to do the TSQ "in English" & so I should be able to put it in formal form.

Mem I found 10:10A Dorris 492 5081 2395 Mass Ave

Would the form of .30–.32 be EBG?

Some kinds of Induction / updating:

Tylenol on A-sp N-G.

1) Numerical T.S.: Some leading into ideas! ~~that start w.~~ sequen. exact rules, $X(t)$ can be found at t as (near by) previous values of $X(\cdot)$, four types TS's:

$X(t) = f(t, X(t-1), \ldots X(t-k))$ starting trials w. small k.

To find $f(\cdots)$ one might step thru t-space & all possl. funcs, but t. pc of Soln is pretty too small.

2) If one adds noise to 1), t. soln. is t. same w. additional pc of error corrections,

3) Soln of Equs: Solve $x = 2$; $x = 3+1$; $x = 3*(1+7)$ .. etc,

— is this same as A-NL?

" " $X+1 = 2$! This could be solved by direct search if X is an integer. — in fact, most of integer solns.

SN Advantage of several examples of low precision v.s. 1 example of hi precision! Search is faster, but "sharpness" of soln. is muddled — so several low precision examples is better TSQ than a single hi precision example (usually!)

.20 is true for direct numerical search. It is not true if search is over functions/forms, in which case t. search time is same in both cases. (usually, we'd rather that t. search be over functions/forms)

D2288 TM TSQ

Some Impt kinds of Things I want TM to be able to do!

1. It watches a problem being solved — say a specific linear eq. . From this, it's able to Genz. & solve any linear eq.

2. It watches an eq. solved by "invertable substitution" ; (say a linear eq.) from this it can Genz. to quadratic eqns (once ↑ is known as legal)
Then " " " Cubic " (" ↑ " " " ").

3. Given $\sqrt{-1} = i$, when asked what $(3+7i)*3$ is — what would it guess?
$(3*7i)*(4+2i) = ?$ ect : — would it be able to generalize complex nos. ?
— Square roots; complex primes ect. Complex solns of quadratic eqns. ?
Complex cube roots? $(a+bi)^3 = a^3 + 3a^2bi + 3ab^2i^2 + b^3i^3$
$= a^3 + 3a^2bi - 3ab^2 - b^3i$
$= a^3 - 3ab^2 + (3a^2b - b^3)i$.
$a(a^2-3b^2) = a^3 - 3ab^2 = x$ } x, y known.
$b(3a^2-b^2) = 3a^2b - b^3 = y$ }
$ax + by = ab(4a^2 - 4b^2) = 4ab(a-b)(a+b)$
Given a geometric interpretation of complex nos. in a plane! After seeing powers, could it then do roots, ect ? — could it lead it to discover various properties of complex nos ?

Re $Ae^{-i\theta}$ representation.

4. on (☆) EPL : When it solves a prob. or sees one solved! It might be able to generalize suitably if it has ideas about what probs will occur in t. future.
Say it has a "Problem pool" — problems are always being added (by Me) to t. pool.
TM looks at t. problems, tries to categorized them as well as poss'l., so it has ideas on how to solve them. Then. when it solves a new problem, it will be able to know how to Genz. it's soln. (There are many ways to genz. ... so ... it may get ideas from t. unsolved prob. pool. (i.e. in t. directions of one or more of t. unsolved probs).
Also, wrt. solved probs; — it can genz a soln to a new prob. if it solves older solved probs. in different (& perhaps, but not nec'ly, better) ways.

In General, t. "Problem pool" can be worked on w.o. actually solving t. probs : just look for regys. in t. problems, ways to compress t. pool.
Actually, this compression is usually a key part of prob. solving ! The problem statement may be regarded as "a partial object" : prob + soln = "whole object".
Since partial objects are always part of a corpus that needs to be compressed — So we do want to find similarities, regularities, analogies & things better. t. various problems.

Gary 5-7-15-a

"Hints" ! "Hints" ! I kind of Hint to solve prob A, is a prob B that is easier but uses ideas that can be Genz.d to solve A

More impt. Things I want TM to be able to do:

In symbolic Integration; TM should be able to see a particular trick being used in a particular problem: Then generalize so it can use this trick *much* more generally. There are an enormous no. of tricks ~~that~~ of this sort that have been incorporated into pgms like Macsyma, Mathematica, Maple — There *may* be books on this — but also, much info *may* be proprietary.

.08 T. idea of "invertable substitution" can be ~~xx~~ used for both eqn. solving & symbolic Integration.

Some Q's:

1) T. "Grand P.d." Can be in several forms or mixes of them:

a) We input a string (condition): Output is seq. of (output strings) in RUff pc order; ~~xx~~.
~~3)~~ We may have to put

b) Output could be a "most likely" string or number, & a σ^2. — ~~xx~~ (i.e. normal (dist.) or ~~output of~~ description of some other P.d. T. P.d. ~~can be~~ could be Multi modal — in which Case, ordering t. outputs by pc order could be diff.

c) Output could be a ~~xxxxxx~~ Monte Carlo set of outputs w. t. proper probys:

— Not so desirable, but it could arise if t. P.D. is a stochastic Grammar — say a *c.f.G.*

.18 There is t. recurring problem (solved *many times*), of putting outputs of some kinds of/ simple stoch. Grammars in $\sqrt{}$ RUff pc order. — In simplest form, we have a ~~xx~~ ~~xxxx~~ finite set of symbols, each w. its own pc. To list ~~xxxx~~ all finite seqs of these symbols in pc order. I think I solved *this* once, most recently,

.21

d) Q: Could input to "Cond'l p.d." operator be a *p.d. on strings*? — It certainly *could* be: but t. Q is: does this (often/ever) ~~xxx~~ occur in TM?

e) P.d.s have 2 applicns: (1 Use to make decisions): 2 outputs of TM (II) to Control an Lsrch (used internally by TM)

2) Work out Example of TM's in action on a problem of some diffy (like some of t. diff. Genz. probs of 148.01 – 149.08) — or *147.30 ~~-40~~*

3) In ↑(1) I probly *need* to work out methods of converting from one P.D. type to another (like (.18 – .21)).

4) (≈SN) In "Operator induction": one form: we are given $[I_i, O_i]$ a set of I-O pairs: to induce. ~~xx~~ (new) O_n given new I_n. This is a normal "*Finite object set induction problem*." If we are allowed to furnish new Inputs, ("Experiments") t. problem seems much different. Each experiment has a known (or Unknown until done) cc — (*may* be / *zero* for some studies) This is "Learning w. a teacher" It is also ~~x~~ same as or close to problem of Scientists wanting to learn about RW. — so t. prob. is *not* "well defined".

.37 A related problem is Jurgen's "Learning how to learn": Essentially an *optzn* problem. Given a / known sequence of ~~Input, G~~ Input, G, pairs, to find ~~xx~~ v.g. Inputs
(Ans, see 150.07 for probable *Soln.*)

Spec 150.01

.01: 149.40 (4) cont. Soc

But t. problem is actually more difficult! G_i is for $(I_0, I_1 \cdots I_i)$, so we can never repeat a trial situation (which J. makes a big pt. of).

T. main Q here is not whether J is rite or wrong (his rite) but whether I, personally, am interested in solving that problem. It is t. problem of a living organism trying to survive & maximally reproduce, in a hostile envt. Certainly an "interesting" problem & perhaps impt. — but I may be able to build what is, for me, a useful TM, w.o. solving that problem.

.07 Actually, .01 may be solvable as an OZ problem! (hmm its of the "unsolved type" $\max_{x,t} (M(x) \cdot F(t))$. ← Try this as an "anytime" problem — using TS as in Time shared soln. of OZ. Still — This doesn't say when to stop!)

.08 One has t. set of input G pairs: $(I_0, G_0), (I_0, I_1, G_1), \ldots, (I_0, I_1, I_2 \cdots I_i, G_i)$:
From this, one can view it as regular OZ problem. One is always restricted in one's input trials (since each input must include all previous inputs), — But I think t. standard Lsrch Theorem would work. T. Theorem gives a P.D. on all functions that look at t. problem & available trial data, & devise a suitable Hill Climbing Routine.

.15 A big. Q is how this P.D. "learns" from previous Organisms' trial lives.

---

5) Another very broad Problem: How do I take ideas like 147.30 (solving eqns. — t. "Einstein way") & put into a form useful for TM, so suitable Lsrch can be done, etc. ?

One way: In my own soln. of t. problem, write down what seem to be impt. ideas (not all necy, but likely to include t. necy, & /o sufficient) used in solving t. problem:

.20 So: Ideas: ① assoc. w. some (maybe all) strings is a (real) number (exactly or approxly) ("numerical value" of string).

Def teq ② (String a) = (String b) means that strings a & b have same "numerical value" : a = b is an "true equation". = (T.eq.) Tq

③ It is possl. to change teqs to produce new teqs: Also combine several Teqs to produce 1 or more Teqs. (See ⑤ for some ways)

④ If "a" is a string that has a value, then a = a is a Teq.

⑤ if a = b . & c = d are Teq's then a+c = b+d, a − b = b−d; a*c = b*d are Teq's.
and if c = φ is not a Teq, then d = φ is not a teq. and a/c = b/d.

⑥ if A substring has a value, then eqs containing that substring will still be true if we substitute t. value for t. substring. More generally, if a = b, then b = a & we can substitute a for b in any teq & it will still be a teq.

⑦ if a string is a number then t. value of t. string is that number.

There may be other impt ideas, but ① thru ⑦ can prolly be reduced to a smaller set of ideas that will gen can generate ① – ⑦.

From t. ideas ① – ⑦ as a kind of corpus, we could ask TM Q's about strings & numbers, to see how smart he is. So we would quiz TM carefully in this area, before going on to problems like solving eqns.

Another Q is: what kind of knowledge TM has — is it "factual" (= true/false) or Probalistic? If t. knowledge was obtained by (any (induction)) it must be problistic.
— But by using many examples or nos. w. many random bits (much "m8") on can get very close to "certainty" — as in ANL

So, we can perhaps get TM to learn certain of its "tricks" of 150.20-40 by a few cases of hy precision.

Probly best way to do this: Write out details of trng in English, then progressively formalize. Compare w similar process for ANL.

While 150.20-40 can probly be much simplified, shortened, & improved, I should be able to do the formalzn. of almost any English procedure — not really of basic English procedures.

Intact, this is the primy problem: to be able to formalize any really well defined English routine.. Suitable

So consider various kinds of English statements, Q's, definitions, ecf.

Answer Wallace Re: Thesis.

Definitions: For TM are more (as for Humans) are much more than agreements to substitute one string of words for another! They are a statement that of "Thing defined" is of importance & its even a hyer than normal pc: Also we look for its occurrences in text, goes to update its pc. So if a "rt. d" has been defined we look for "rt ds" in (almost) all texts. We mainly look for them in Geometry & Math texts.

→ Thus if we find them in other texts, it is of unusually hy interest!! So maybe look for them in all texts.

Fortunately, looking for certain string types is easily parallelizable — so adaptable to human & into progs., but also to hy capacity Machines!

On the use of "numbers": If I just use integers I will be limited to classical number theory: Which is interesting — but I'm not at all knowledgable in that area. If I allow reals (which I am familiar w), there are various difficulties of approxn, Round off error, etc.

However, I may be able to get TM fairly far before these difficulties become serious; At Least Time, I may be able to explain to TM what the difficulties are & how to get around them!

D2698 TM
:145.02

OSL One Shot Learning

152

.01 (Much previous work/note): Usually, the way it occurs (?): We have summarized corpus, C upto now (w.o. OSL); we get a new partial object, A—, so corpus is now CA—. We want a "summary" for CA (or "updater" of summary for C): The way we usually do this: We look for previous occurrences of A or part of A in previous corpus, C. (This is if we are doing just ZZ141, simple ktup. data).

So, in this forum, the Technique of "2 kinds of —" in designing a Summary Machine. Is probly correct — we design summary machine for CA, not C. — On the other hand, in updating code for C, → CA, we do not only use random contins of the best codes for C. We augment these codes w. "OSL" codes for A. These are derived in special ways: depending on the start codes for C alone. How this is to be done, is clear for ZC4C, but not in general clear.

The Q. about whether OSL can be done by MML, is unclear: If MML find a min code for CA, it may, indeed, be able to do OSL. It still would have many pc's for ktups that occured only once in CA.

An impt. idea: For any long sequence, no matter how random it is, the continuations of that seq. are not all of ≈ probty. If the original seq. is n bits long, the 2n bit repetition of the original seq. has much > pc than just 2^{-2n}.

[Hwr, the larger n is, the less regularity we can expect in a fixed length k bits extension of the original n bit seq. (? — is this true? Say "replace first/last k bits"). — if k >> the no. of bits in the inst "repeat", then this k bit seq. is of >> pc than 2^{-k}.)

Actually, "A" can be quite large: Say it's a new Scientific Paper, or a big batch
.25 [of data: One could code it in small increments, but I suspect that this is non-optimum.
.26 [One can code the whole thing, using copys in C, & then recode it using new found copys in CA, loop to α

This seems to be the way I often read a diff't paper: I go thru it several times — each time picking up more copys, "understanding", based on ideas from previous runs over the sub-corpus.

.29 [At certain pts, ~~I may want~~ I am reminded of ideas in C (the previous corpus) &
.30 [So I may go to reference other works.

So there are 2 major updating methods: .25 incremental, .26: coding & recoding.
Also .29 - .30 seems impt, tho I'm not sure as to how to implement it!

One way: As I work, I develope categories of ideas, & areas in which they are applicable -
[I once wrote about a kind of Algebra for this: {problem areas} descriptors for,
{& in methods} descriptors for: A some sort of statistical/ computational correlation of usefulness
linking the 2 sets of descriptors.

spec
154.01

"o!" 149.37–150.15 ; spec → 116.40 : On 150.07, I thot J's problem wasn't "unsolved OZ type $M(x).F(*)$"

It may not be. If the Reenforcement comes at known (periodic) times, then it's a standard OZ prob. w. fixed CB. This is assuming J's "Greedy ~~a~~ Gore"... which is prob'ly not so good. I really don't know how to best ~~...~~, deal w. ∞ horizon, — Tho it may be usually irrelevant: That distant futures (predns, evalns, expectations) may be so imprecise, that they are given very little wt.

OSL & "SCIENTIFIC Method" (.20)

.01 ! 152.40 : A common type of short A, to update is a Question. If has to be understood in terms of previous Questions &, ultimately, in terms of their Answers. (partial object)

A simple kind of Q to trn. to answer: We have a data in Memy: So we ask "what is content of address A_i" T.M. then trns how to write answer.

If Resist. that data set TM is drawing on, the phrase of & "what is the content of address" is irrelevant. only "A_i" is impt. Later, when we have other refs to addresses, the phrase of will become impt. Indistinguishably one type of Q from another.

Back to 150.20 ff: One impt. idea is various classes of strings.

A string can be a no.; an eq.; or T (true equation.) ; or ≈T (approximately true Eq); a logical statement.

So me strings have themselves; some are true or false.

Some are Questions: others are statements, or "answers" (can a number for several strings be an "answer"? Probably always one can find a Q → a certain string is the correct answer.

→ But I do want to see if I can deal w. things close to an "English" representation of 150.20 ff.

Another idea I may need is "local variable", "local statement" — These are true only with a specified sub corpus.

.20 12599 IMPT Practical Aspect of OSL: Say in Science, One has several poss. theories to explain "Observations up to Now." One of them is much better than the others, in terms of pc for the known corpus, so in MML, we would tend to accept use only the "best" theory. .23 However, when New data arrives (from some previously untouched part of the scientific universe), the "best" theory no longer fits very well. To make new theories to fit the new data, we would, using Wallace's MML, use only the best theory plus for its sub-cases; its component. In fact what is usually done, is to take not only the best theory & its sub-cases, but much worse than the best theories & their sub cases, in order to try to get a new theory to fit both old & new data.

This ↑ is equiv. to what we do in O.S.L. Instead of using only codes that were good up to just before the new data came in, we use codes that were fairly bad, but which would code the new data very well. I.e., if the new subcorpus is "α", we consider codes which define α — which only occured once in the previous corpus & would ∴ be poor codes for that previous corpus, but good codes for the newly augmented corpus.

So if α is the "new data" of .23, then we try to find codes for the "old corpus + α". Such codes could be rather poor for "the old corpus alone". — (just as in OSL).

→(99) 11.01

D28 98: TM: TSQ (.07R, .12ff)

Health Care Assocs:

Dr. Weil

Cr. Brookline, Longwood. Shapiro Clinical Center. (CSRI?)

① Rash: Legs, Arms, chest (some). Back of neck. & Raw in places where it itches.

② Cough: Left lung hurts. Better now (130PM)

③ Cold.

④ Colonoscopy appointment.

(5) Ask about Medication after J 1 99 "low cog dynam?"

12 The idea of learning functions (≡ operator induction).

(1) Quantities
(2) Eqns (Teq's)
(3) old Teqns → new Teqs
(4) a = a
(5) = ($\dot{x}$ / −) =, = =.
(6) Substitution is legal
(7) If string $\in$ No., then Val(S) = S.

Solving eqns can be discrimin!

Solve (X, 3X=7) (maybe same as Maple!). For a "hint", will consider!

3 (Solve (X, 3X=7)) = 7 is a Teq.

Num(3) → 3   Num(3+5) → 8   ① I do want a real system!

Also for each problem, I'll need "local variables" — "local facts".

Local Universe". For various functions, one could have an extra [struck] argument telling where the universe is

Perhaps just start out w. some simple induction probls — using an incomplete system — This is just to get the "feel" of the ideas. Later I can modify the system to deal w a greater variety of problems.

Primitive functs that we start w:

1) Num(s) = (0,1) [redacted] depending on whether s is "number or not. Perhaps numbers will have to be labeled: ordinarily Ray will be a string of bytes. When Humans look at a no., they can always (?) tell it's a no. By "No.", I mean 2.71 or 13, not "π" or "e". (2.753E−21 also.)

Lenore Blum had a way of dealing w. nos. & doing Calculus, was running into lots of diffics ("Non-Standard Analysis")

— Tho I should be able to get TM to deal w. Math even tho it's not exactly "easy" — i.e. like a human / person does w. math.

Perhaps Go thru that elementary Algebra book (for high school) — see if I can simply present examples & get TM to do adequate induction.

Dolan Jones. Comm. # 208 652.73 recd. on D28! 929, 11208.

Winterizn. Nov 5.

2 Wks Ray after [test] 23 Nov 98 Ray will call Back.

Dermat

For Jan 8 2 PM 2nd floor Dermatology Dr. Worth

Blood test 2 vials,

Got both blood tests from Gardino. Got dates of them [redacted] / about blood test was 8 28 98

Temp 98.3°F

D28 98: TM

156

last page of 1998

Book "Intermediate Algebra" "≡IA"

Starts out w. student able to add, subtract (ε zero positive nos.) only. (no neg results possl.) So I can start w. Arith unit that can only do that. 4+5=9, 5-4=1, 4-5 = meaningless. (they be only integers).

I gather that t. book is mean for not very advanced undergrads. — a first course in "Algebra"

Some q's on pp (69—80) look difclt; T. ideas of "some", "every"

Knowing what an "integer" is. (Integers ≡ Q mod 1 — But this defn. not used) (is not rel.).

Infinite or finite sets: TM could induce summary of 0,1,2,3.... or 0,1,2...7, 9,10...13,

Defn. of an infinite set: could be criterion for inclusion e.g. "integers" (≡ φ mod 1).

PP19-21 Bowman's Shockwave Rider!

D2998 A perhaps difrnt. approach: (1) Make "Lrng Functions" t. main activity of TM: (2) Also lrng Definitions — 155.12

.17 (But int. form of functions: so [scribbled out] Equation (3x+7=2) = 1 = True:

(.18) (3) But start in middle of book (or at a pt. where interesting concs. are being introduced, usd.) & work backward to find functs. & defns. that are needed — to construct needed Conc. Net.

(.18) sounds most reasonable — otherwise there is a lot of stuff on numbers that will not be used to solve most problems.

A possibly attractive thing about t. Alg. Book, is that while much stuff is not needed, t. thing. seq. used in t. book is perhaps close to that occuring in R.W. — in that much stuff is lrnd. that is not relevant to most problems — but might be, eventually. T. approach of (.18) is a more "minimal" system — More "brittle", perhaps.

When we have solved a new problem, we have (1) Augmentation of Corpus (2) Augmentation of concs. usd to deriv. corpus — new concs. a/o updating of pc (frequencies) of old concs.

(SN) on "Definitions": we could teach TM many impt. Definitions in

.24 functional form. (a (.17)). Then see if it could recognize instances of any of t. defns. in a corpus of "Text". This ability to recognize defind objects is probly quite important in prob. solving. Much content of science is in t. form of definitions. T. "sci laws" are then (≈ eqns.) relating things defined.

A list of "kinds of lrng" for TM.

1) Function lrng (operator induction). Very broad class. Maybe broad enough for all lrng!
2) Definitions: (a) in functional form (e.g. .17) (b) find examples of t. defnd thing in this corpus. (.24 ff)
3) Concepts like "solve", "simplify": idea of "closer to a soln" (a) scalar (b) vector (GPS). ...

→ 99TM1.01

D298 TM : Index of some impt threads:

1) Jurgen S'. "Irry how to Irn": 9, 115½, 116

2) Mixed Corps Problem. MCP : 107.01ff This was solved on 115.40 : Comments : 126.01 : 138.01 144.01 : (130, 131) Review

3) History of Science : 122

4) "OZ problem" 99.36, 100.08, 102.18, 105.01, 108.05, (117.01), 123.01, 127.01, 128.01

5) Summarizing (P.D.'s. Machines) : 135.01, 137.01, 140.01, 146.01

6) Z141 : 18,
79.20 - 40; 90.01, 91.01, 97.03, 98.0, 109.01

7) GA, SGA, Org evoln: 112, 113, 132

8) OSL (one shot Lrng) :
146.07, 152.01, 154.01

9) Unsolvd probs (list of 4) 145.01

122
Has

117.01
108.09
105.01
102.18
100.01 123.0
99.36 137-

ABC