BACKTRACKING ≃ REVISION ≃ Updating (.29)

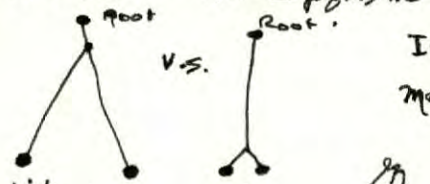
.00: 1950: In Z141: We backtrack to the pre-corpus; the part that we write later from is any output of the UMC. We retain a certain part of the pre-corpus: Usually all of the data & their (Σ) frequencies. We may discard a few of the most recent decks, but this is probly rarely done.

.04 In "M.B.", I first I backtracked to a part where the UMC had already started giving output bits: so the corpus was (partially) visibly coded. I would imagine that for most kinds of codes, this would not work well. If the corpus was not sequential, it would be possl. to recode a certain section of the corpus, (re-coding, perhaps to certain coding "cncs" which were to be changed.

Perhaps in .04, my ideas as to what constituted "Backtracking" were rather (naive/primitive). A more general concept than "Backtracking": "Modification of certain aspects of the code, & keeping the rest". "Backtracking" has the connotation of going back to an earlier (less informed) state.

A More General idea perhaps is "Theory Revision" ("Backtracking" is one aspect of this)

Def N.B. One of the ideas I had in "M.B." (≡ Ohm Backtrack) was to try to store 11 poems that were "Maximally distant": (for "Diversity") This had a simple enough interpretation in M.B. — i.e. 2 poems were very distant if their lowest common element was very low.

.18 

.21

In other "Revisions" — (saving of 11 codes) — the "Distance" measure is not so clear.

In Z141: any 2 parsings of the same set of data might be regarded as "same data, different". If we use distant basepoint data, for 2 codes, then the more non-common during the more distant" are the 2 codes ((≈ to .18–.21) left).

So, an interesting Q is: Are there any General principles of "Revision" — common to many coding methods / poems? A Basic idea here, is the "Growing Corpus". The thing that suggests revision, is a lot of new data of very small pc. ("very small" means one may expect much larger — based on previous experience.)

.29 Actually, "Revision" is same as "Updating". Whatever does depends much on available cc.

— So in General, it is a MAJOR PROBLEM Area for TM

.31 So, just as we expect to have a lot of OT's & a lot of induction Algms (IA's) & we want to make a grammar to Gener. the set(s): The updating methods (≡ Backtrack ≡ revision methods) will also have to be generated/generalized.

So: I will simply list various IA's along w. their updater algms. & look for common Concs

.00: : So where am I now?

.01 Well, Most Recently I looked at ~3 methods of "Backtracking" for a simple UMC coder. Comparing them to the Update scheme for ZIL, I suggested that if any of .01 worked at all, this would be peculiar to the (UMC, regsys in the corpus) pair.

That in general, Backtracking Theory revision & Updating, were all about the same problem — & that the way it was done depended on how much the "updated" (& perhaps on the type of regsys in the corpus).

Before that, I was working on the QA system. Again, the major problem (I guess) was (other than TSP writing), the update algm.

Some diffys: (1) for medium cc updates, one would want to use only part of the corpus. Just what part is a diff problem. For a 'pre-linny TM', we could assume updating is the entire corpus. — (Then, in general once updating technique for limited cc is to do a "Backtrack" — which is a way to use only part of the corpus (for certain IA's))

.14 (2) I think I had a set of IAs $[IA_i]$ and a function $f(X, \sum_{\text{I. problem}})$ that looks at X and at time available computes (or updates) a prob. vector for $[IA_i]$ (i.e. tells how much time is to be spent on that IA for the problem, X). Closely assoc. w. using an IA_i for pred'n, is updating it.

(3) We note that each IA_i has at any pt. in time, a partial summary of the parts of the corpus to which it has been applied.

(4) Go thru Peirce & print Bibliog. refs — where abduction is discussed.

(5) The main problem in QA was updating the IA's. for "operator induction" An easier (I guess) induction problem is Bay induction, & easier yet is Seq. pred'n. The operator induction can be viewed as result of a "Bay induction" problem — I usually don't think of it that way!

Consider the update algm. for seq. pred'n - - - - the "abstract" problem. The most succinct way I know to do Seq pred'n. — each unit's own update algm.

In fact, the Guts of an IA is its update algm!

SN I remember an apparent famous diffy in the f(x) re idea of .14: F(X) has the problem X as input & it has a d.f. on IA's as output. The IA's have X as input & the soln. to the problem as output. Somehow, the F seemed to be identical in function w. the IA's. Hum, again, I'm not sure as to just what the problem was!

SN I'm really uncertain about "Anytime" problems! If we have an F(X), & X is an "Anytime" problem, then based on past history, the A_i's to choose will depend on the Time allowed. — If we use (1 "Lsrch", the wts. of the various IAs should change as the calcns ("Lsrch") proceed!

.37 .38 There is the standard T or 2T soln. to the Anytime problem. I think this is equiv to using (1 Lsrch, but changing the pc's as .37→.38 — since each round of T is diff. & ∴ the pc's of the IA. are diff.

→201.00

IDSIA

T. "Correlation" problem

.00: : On t. "correlation" problem: In L srch for solns. to problems, Guided by 2 probby distribns, L srch would be optimum for "BlindSrch", if t. pc's of cands were uncorrelated. If they are correlated, many sets of cands will be about t. same; .03 it is very wasteful to test all of t. elemnts of one of these sets. Testing only .04 one per "correlated set" would be best (if sets have little correlation betw. them). If one knew ruffly what t. "correld sets" were, one could implement .03-.04 — but usually one doesn't know this.

An Alternative way would be to try to get "Trials w. hy diversity". These are relatively hy pc trials that seem very diffrnt & ∴ are expected not to be correlated. For each grammar that Generates Cands, one might be able to dream up a measure of disparity 'bt 2 cands, — that is some sense t. 2 cands have "very diffrnt derivs" wrt t. Generating Grammar. 2 cands can both be ok by pc, because they use cons of hy pc .... but they use diffrnt consts — so t. 2 cands are "very diffrnt".

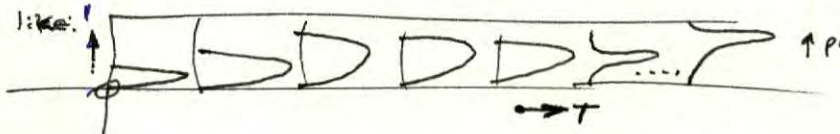
Having Cands that are uncorrelated is also ≈ Having much "Diversity" in trials

Diversity in GA (or other search Algms) makes local extrema less of a problem.

IDSIA

.00 : 199.40: SN In the QATM I had been thinking of using many IA's in all (199.14 ff).
→ There were some apparently serious difficulties in that approach. If I used different IAs on different problems (or for different amounts of time on different problems), it would become very difficult to update F(x). Remember that the amount of time an IA spends on each problem changes ("updates") that IA.

Remember that we don't compare 2 IA's by simply the total pc they get for their corpus. It may be closer to the [redacted] mean of the pc ... [redacted] plotted. More exactly, we want F(x) to give a distribution on pc of soln. as a function of time spent.

.09 So: like: 

Probty of pc as a function of Time spent

Also, if T for an IA is long enough, it will only work on problems in the corpus that it had not been applied to in the past! — Just how they (the IA's) go about this, is not clear!

Another BIG Q: Just how much time to we give to f( ) to compute f(x)?

.19 If we are given a "T" as part of x, perhaps P, itself could do the forming an optimum time limit for itself! At first, perhaps, the USR could choose this T for f(x)... Later f( ) could learn how to "optimumly" do it for itself — the optimizer expends some of this time. itself is the set of IA's. 202.03 202.03

I think that near the end of my most recent work on QATM, I began to realize that each IA changes as it works on problems, that what each IA is (At any particular time) depends on (1) the IA (2) how much time it spent on each of the probs in the corpus

---

Assuming .19 is adequate: Is QATM at all clearly defined? — There is, of course, the problem of "updating f( )". This is an induction problem! It probably could be regarded as a QA problem of a special type — but its a question

.31 (w.o. any specific "correct answer" or even a set of correct answers.) — CONFUSED (.36)

.32 F( ) does, hrr, have a well defined Gaussian (?). We want it to give "Accurate" (≡ avg pc) predictions of the curved "surfaces" of [redacted] .09. Also, we have to discount its pc by its down cost. Also, we want this updating of f( ) to be able to use cases that occurred in the IA's.

.36: .31 Actually f( ) does have correct answers: for each IA that was tried on x, we got a pc at a time, ≡ (≡ correct answer) so updating F( ) is a standard prediction problem. We should be able to apply many of the IA's to this problem, since it is a QA problem! The Q is for a set of Q's (see)

6/2/01

IDSIA:

Eric Wiewiora

:00:20:40 : of the form (X (including T_i), AI_i, T_i (total / Time AII was applied to X)) : "A" is the pc of the soln. of X that was obtained.

.02 Well, this sounds not so bad! Perhaps write a "summary" of this entire QA model.

.03 In 20.1.18-.19 FC) finding an optimum Tcmax limit for itself is not exactly an induction problem! It's an "optimization problem" perhaps of the $G(x) \cdot F(t)$ type — unsolved by Lorch! While it is an essential problem in the system, wherein it is not very critical, as for a final system, we can probably use some not very optimum solns: — T. main difficulty is that we have to express it as a QA problem: After TM has world variables $G(x) \cdot F(t)$ or problems as "QA problems", it will be able to essentially work on this one. } This will be a fairly highly developed QATM!

We could start off w. $T_{FC} = \frac{1}{2}$ or $\frac{1}{4}$ or $\frac{1}{8}$ of T_{IA}. Fixed if the factor is $\frac{1}{2}$, there is the old argument for (T. "50% soln") that it couldn't be Too bad!

On "updating": Each IA has its own method of updating. At any particular time, a IA will be represented by some stochastic Operator. "Updating" involves modifying this operator, in view of some augmentation of the corpus. The way the updating is done will depend on how much time (cc) is available. For large cc, it can involve massive recoding of the corpus.

I can list various IA's & their update algorithms

1) Base Cap (Say Alphabet > 2): The complete update consist of the set of case counts for alphabet. Base are trivially updated by the corpus, & the pc of the conditionals are obtained from "Laps rule".

2) Z141: The system state is the current set of definite & Recursive counts. Updating updates case counts (in view of augmented corpus) — a parsing of the augmented corpus using the current case counts. For large cc, we can try recovering more of the corpus than just the newly augmented part — perhaps all of the corpus. For larger cc we can look for new definitions, & if we find any, reparse the recently part of the corpus, or the entire corpus, in view of the new definitions.

.33 3) AZ141 (augmented Z141): ~~More of the corpus is now taken to be a function, or a set of (not necessarily) ordered pairs.~~ Here, the corpus is a set of QA pairs. At any time, we have a stoch operator that goes from Q to A. This operator is "generated by" AZ141, from a primitive set of operators. We have a procedure (like Z141) that includes the primitive set of the functions & any defined functions. The p.c. of any final function, is determined as in Z141 by the product of pc's of its component functions. The notation for functions as POP, A

.00: 202.40 : So if function described by a string, & we can get pc's of each symbol in string (as in Z 141) by considering how/frequently that symbol has occurred in that situation".

In the simplest case, the current (MTM), if the current function gives A_i exactly for each Q_i thus far. (i.e. operator is not stochastic).

Finding the funct from prim Qs is easy for the beginning of the seq. Updating is usually a bit complicated:

.05 Say we have a punch. Ω that works for all of corpus up to now. We augment corpus w.

.06 Q_n, A_n. It doesn't fit this operator, Ω. Using the statistics of Ω we construct a new type operator F_n that maps Q_n to A_n. Also we find an "Ob" that can distinguish Q_n from all previous Q's — so we know when to use F_n. This is/ conditional combinement of Ω & F_n gives a new operator that works w. all QA's thus far. I.e. roughly, "updating" is for small cc.

For larger cc, we want to interpolate F into F_n in a new pc way.

In simple fsQ's, it is clear as to how to do this, but I can't say anything very General about this 'interpolation'.

.20 To doubt: data (mildly) as a stoch operator, using corrections for the new A_i (i.e. we use old Ω, but we learn to "correct"). We may use Gain-ratio chop" ma[illegible] until we get Storage Modification Machines. — Schmidhuber out of GSZ to do a better Ω operator.

An Alternative way to deal w/ the update of .05-.06: use

.16 If the relation of Q to A is stochastic (NMTM), Then

for each input Q we can have additional bits that are needed to obtain the particular A for this Q. (i.e. same Q when repeated, can have different A's)

One simple way to implement this: we have a non-stochastic Ω for all QA's:

.20 In addition, we may (or may not) have a seq. of bits that tell how many bits of the output were in error, & just which bits they were. (See stuff on Z 141 & Wolff, for exact formula(s) (There are 2 alternative formulas I obtained).

---

4) A third very General kind of induction is Bag induction. Operator induction can be derived from it. Using the formula: $P(A|Q) = P(A,Q) / P(Q)$.

In Operator induction, we want $P(A|Q)$. $P(A,Q)$ & $P(Q)$ are obtainable by Bag induction from a corpus of Q,A pairs. So $P(A|Q) = P(A,Q)/P(Q)$.

In 3) we obtain $P(A|Q)$ directly.

In the present pure Bag induction (unordered finite strings), is considered: $Q_1, Q_2 \cdots Q_n$.

In general, we derive a stoch grammar for the Q_i set. There has been much written about Grammar induction. Most successful work has perhaps been on Finite state Grammars, Certain subclasses of CF Grammars, ... I'm really not very familiar w. this M.S. Fu has written much on applications of Grammars, but much less on finding them.

Wolff is working on somewhat General Grammars, but so far, I've found his writings very hard to understand.

ID, SIA

E=∞ : Unbnd Extensibility : (.18, -.29)

Extensibility Unbounded

.00: 205.40: IN 3) (202.33) I Rink Pert we can be guided by a reasonable TSQ — to help determine what kind of Grammar to use & just how to update. This is because I feel Pat I "know" t. "soln" for t. TSQ & (including updating) is all I Pen need to do is map this knowledge into a good formalism. I Rink t. formalisms of Z141 & AZ141 would be very useful.

My latest idea was to use a "Universal Fourthgen lang." — like Lisp, — Pen write several more solns to each problem in Pat lang, & see if I could find an updating scheme or reasonable pc.

Another impetus was Pat of "condl pc", so t. pc of a conc. would be a function of t. application to which it was to be given. This was expected to deal w. one aspect of "t. Scaling problem".

SN Some Relevant Refys to Read!

1) McConnell's paper on "Xs"

2) Try to get idea as to what wolff has been doing.

3) Try to get papers on CFG discovery: Look at Horning. Also other, more recent work. Not nearly using ALP or MDL: Look at my own work on PSG discy

.18 → An Aspect of t. QA TM not dealt w. in this recent analysis: How to set of IA's is augmented by "updating" FC()? Since any conceivable type of IA can be invented, t. system that is theoretically capable of arbly clever IA's. So it is t. SE aspect of t. TM

It Gives UNBOUNDED Extensibility

On 201.32 I got I. idea Pat updating FC() was a normal QA problem. It was a NMTM problem, hvr. (stochastic).

Well, t. output of FC() can be a string that can derb. any Concievable IA! Th. induction (updating) problem for FC() maps problems (x) into strings Pat derb. IA's: ∀ for a/nt TM, t. set of possl. IA's will be a (small) finite set — May be < 10 diffnt (specific) IA's. Later: When TM has more experience w. grammatical induction (& other good induction techniques), It will be allowed to extrapolate ∀ t. initial set of FA's to t. most general possl. set of IA's (Unbnd. Extensibility; Extensibility = ∞ (not finite)

.28 .29 I'm not quite sure of t. adequacy of .18 – .28, but it may be OK.

.30 AT present pt., I Rink I'm ready to write a good review of QATM: Better Pan usual, because t. main outstandg problems is "updating technique" & my main approach on this is to write a TSQ, ... which probably will not prevent me from finishing t. review.

6/5/01 Re: .30: First do a careful analysis of t. system to be sure it's "all there", Pat I haven't left out some critical part. The "updating scheme" is t. main "vacancy" — But I Rink Pat any Updato system should be ∈ close to Pat of Z-141, AZ141, Lenat, &/o a combination of Pem. AZ-141 consists of ① Discovery of new cncs by combinations of old cncs ② updates (PL's) of cncs ③ re parsing of corpus. Lenat enters in ① or ② when we have to do a search

205.00

:00:204.40 : to implement them. One example might be to find a coder for a corpus augmentor, using existing coders. T·(FC) System) looks at the problem & decides on how to deal w. it.

One way would be for FC) to assign hy pc's to relevant concs that would be used to attempt to (solve) code the new QA. By giving different pc's to the set of available concs, FC) is essentially creating a new IA■!

FC) can also be used in this way to help solve "sub problems" that are parts of "Main Problems".

.09 SN on QA Problems! Each Q should have one or more "categorization indices" that help guide TM in pooling its data: E.g. Say we are giving TM a problem in curve fitting: we give it various $\vec{x}, \vec{y}$ pairs. Each pair is a QA., & the pair has an index that tells TM the data is numerical, what kinds of nos. Float, single prec. Integer, or whatever. So Both Q & A could be "integers", & we could have a long sequence of pairs all belonging to the "same problem": so both have a common index that gives the "problem name".

Also, when humans are given problems, they are also given lots of assoc. info — Like: Did the problem come from physics, chem, Economics, Sociology, Psychology, "computer science"... ect. ■

They know when (in what order) in the T$ Q the data occured.

All these things can, & should be used as auxilliary labels on the data.

On the other hand, we will probly want TM to learn on certain kinds of problems w. minimal amt. of such aux. "Heuristic" info.

One might also try to tell TM whether an "induction" problem is MTM or NMTM

These indices can be also used directly, so TM can learn to categorize a problem that has to be inadequately indexed.

:00: : On the concept of a TSQ: A kind of formal defn. :

For a Sequencial Corpus: We start w. an initial (agnostic) P.D., $P(X_n)$ (X_n is a string of length n)

We start w. a training problem Z_0. Using Lsrch we find a "summary machine" for $P(Z_0^0)$. The cc is c_0^0. This gives new PD is $P^0(X_n)$; we await a new problem, Z_1^0. Using Lsrch, it finds the summary machine $P'(X_n)$ using $cc = c_1'$ by using summary machine $P^m()$ w. use problem Z_{max}, [and $cc = c_m$ to get "summary machine machine" $P^{m+1}()$.

For a more general corpus: we have a condl. P.D. $P^m(Q_m, A_m)$ (or $P^m_{Q_m}(A) \equiv P^m(A|Q_m)$). After being given problem Q_m, A_m, using Lsrch's "suitable updating of P^m" [Algorithm takes $cc = c_m$], we obtain $P^{m+1}(A|Q)$.

One problem is the ambiguity of the type: Not the meaning of $P^{m+1}(A|Q)$ depends on just what sub-corpus one uses to compute it. Also, one might well "over search" (in Lsrch) to get a better $P^{m+1}()$.

.17 Another important omission is $P()$: the choice of a UA ($P()$). — or is $FC()$ meant as a "factorization" of $P(A|Q)$ into $F(Q) \Rightarrow P_Q(A)$?

.19 $FC()$ is a function that does from problems to UTM's $= P_Q^i(A)$

.20 So .00 – .11 is the more general approach. .17 – .19 is one class of it. → 207.26

Basically the NON-el procedure for TSQ's: [see 207.29 for Best non-el machine Prob] We start w. $P^0(A|Q)$.

This is any way to get from Q to the probability of A; we then give $P^0()$ a QA pair: Q^0, A^0 (a corpus). After "updating", we obtain P^1 using $cb = c^0$. We obtain the updated $P'(A, Q)$. In general, $P^i()$ are all "practical PD's" — i.e. their values depend on the CD used to evaluate them.

For $cb = \infty$, $P^{i+1}(A|Q) \equiv P^i(A, A^i | Q, Q^i)$ (whatever Best means!).

A, A^i & Q, Q^i are meant to be "composite objects" that $P^i()$ can accept as args. I.e. Q, Q^i can be 2 questions & A, A^i can be their respective answers: But we can formally regard Q, Q^i as a "single Q" & A, A^i as a "single answer"

.33 I will probably want to ① try to get "summary machines" for BAG reduction ② Get "summary machines" for stack operators. (after writing a good analysis of a general Sola. to the Operators induction problem (of course one could just solve ① & use $P(A,Q) = P(Q) \cdot P(A/Q)$. i.e. $P(A|Q) = \frac{P(A,Q)}{P(Q)}$: to reduce P solns to 2 easier probs.

Anyway do .33 first.

.00 ; 206.40! The ☆ MDL way to do BAG induction: $P(X_\cdot)$ assigns Pc's to strings $X_\cdot$.
we find a $P^i(\,) \ni (Pc \text{ of } P^i()) \cdot \prod_{k=1}^{n} P^i(x^k) = \max$ $[X^k]$ $k=1|n$ is the corpus (Bag).
(we take exponents to the $P^i(x^k)$ if x^k occurs >1 time in the corpus).
The theoretical best update for $\Sigma X^k]$ $k=1|n+1$ is to optimize $P^i(\,)$ based on it.
.04 new corpus: starting from scratch. Here usually, we will find an update alg'm;
.05 so $P^j(\,)$ = funct of $[P^i()$ and $X^{n+1}]$, given. If we have trouble doing this,
.06 we may backtrack several X^p's (not nec'ly in t. temporal order in which they occurred but may be in the reverse)

For / Each operator's; same system: we find operator(s) $P^i(A|Q)$
.10 $\ni$ $Pc(P^i()) \cdot \prod_{j}^{n+1} P^i(A_j|Q_j)$ is max. Then updating $P^i \to P^{i+1}$
.11 $Pc(P^{i+1}()) \cdot \prod_{j}^{n+1} P^{i+1}(A_j|Q_j)$ is max! $P^{i+1}(\,)$ is then a summarizing machine of $P[Q_i A_i]$ $i=1|n+1$.

Normally, we would not use .11 to get $P^{i+1}(\,)$, but make P^{i+1} a funct of $P^i(\,)$ (& modification) i.e. each "IA" Q_{n+1}, A_{n+1} as in (.04 - .06). In General, each technique for obtaining $P^i(\,)$ from the corpus i.e. each way of approximating ".10 = max" will have its own "update schema".

All a IA is is a means for getting an approx max (.10)

Impt ideas here: that all IA's amount to getting an approx of .10 .... that they involve specifying CB use: this makes each search (IA ≡ approx), a type "Practical Pd" (i.e. CB is relevant param)

(2) IA AZ 141: The idea is "Induction by definition" One finds a regy, defines it, & (re)parses the corpus in terms of the new definition. That there will be many ways to reparse, is an impt. fact in evaluating pcost of the def'n.

.26: 206.20! A better "More General" approach than 206.00 ~.11: GPD() looks at Q_i and becomes a IA that gives a pd on A_i ("IA" means that we have to give CB info)
How do we update GPD() ≡ w.r.a new Q_i, A_i pair?
.29 GPD(Q_i) → PD on A_i's. we want GPD s.t. $(Pc(GPD)) \prod_{i=1}^{n} GPD(A_i|Q_i) = \max$ (≡ Probty search Node on OSL; 209.07) $[Q_i A_i]$ $i=1|n$ is the corpus. The problem of updating GPD as stated, is quite difficult.
In .29, we don't even look at the GPD for $i=1|n-1$. In practical induction, usually we look at the GPD for $i=1|n-1$ and we modify it in view of Q_n, A_n.
with OSL from of 209.07
So: .29 is the "Best," most non-ad hoc, kind of TM. We will consider various approxns to it.

This "best" can be obtained by $P(A|Q) = \frac{P(A,Q)}{P(Q)}$ In which case we have "factored" one operator induction problem into 2 BAG induction problems.

Obtaining $P(A,Q)$ seems to be obtainable using same cues as getting $P(A|Q)$ Directly — So I'm not sure we save any thing. One advantage: it is a slightly different way to look at the problem.

Rev

: 00: 207.90 : Outline of Review of QATM.

164.00 is early review.
160.00, .18 on SI for QATM;
161.15 early demark QATM
152.00 -01, 156.00

150.33ff list of ES & Reuters update schemes.

Work on Reuters QATM seems to begin no more recently than 150.00 or 149.00.
140.00 – 141.·· was a kind of review of work up to that time. — it told how present TOM model differed from that of 589; essentially, a list of impl. devices since 589

IDSIA [redacted]

2 hr/yr 2000

:00:20 Next, Consider "factoring" ■ GPD(A|Q) into F(Q) = PD on IA_i.

then IA_i is operating on Q to give D.f. on A. We do run into problem of each IA_i having

.02 had a diffnt CB for each Q_i A_i in the corpus; But see 210.01 !? (diffty) ——— 210.04

What constitutes a [useable TSQ] will depend on what IA's are used & just what the updating Methods is.

.07 .20 .29 [SN] T. formulation of 207.29 is essentially MDL! So [OSL] is not included: In most QA TSQ's, I would think that [OSL] problems would be very common (which normal MDL doesn't deal with)

.09 The (Theoretically O.K.) method of Dealing w. OSL, using MDL, is to include (Anst, Q_{n+1}) into product of 207.29 (i.e. the product index = n+1): Q_{n+1} is known, & this gives us a D.F. over

.11 possible values of A_{n+1}.

I'm not sure .09 – .11 is really correct, in terms of the complete ALP. formulation.

Consider MDL for sequential pred^n: "data" is the binary string x_n: We are considering various possible continuations, "a": P_i( ) is the set of all Pgms (including "universal recursive" Pgms)

.16 The D.F. on "a" is Pgm [redacted] $\max_i$ [redacted] $pc(P_i()) \cdot P_i(x_n \hat{} a)$. For each a, we write here $P_i(x \hat{} a)$ is sum over all strings of which $x \hat{} a$ is a prefix. (finite & infinite).

.17 a diffnt i that maximizes now takes this product. The ALP formulation is $\sum_i pc(P_i()) \cdot P_i(x_n \hat{} a)$.

I don't know how Ris would deal w. this. Probably he would restrict the sum to his pre-defined set of models (his "Model Class").

[SN] (Regression) Curve fitting (Analog/real) & [L search]! Regardless an OZ problem in a fair way!

The form of the model is a discrete set of fcns. The params of the space of real models, is a real space. [redacted] So, we assign a pr to the discrete space of models and do an L search on them. Each discrete model is then given a Gorc by optimizing the its params wrt the data, using standard optimization (min) — T. Gorc is then T. Hessian in continuous model/space [redacted] mult by pc of discrete model (mult by pc of data wrt the model).

T. time for each parameter optzn probably doesn't depend much on the ssz of data set.

If we actually spend same amt of time on each model, this will be ≈ McConnell's SURFER

McConnell discusses Commutativity & associativity contributing to Redundant codes ——

& how he got rid of this (by hashcoding I think) —— The one I'd use detect identity by use of random numbers —— i.e. 2 funcs are probably identical if they have same values for a few (tiny no.) random args.

One might use this redundancy to give more wt (time) to certain fcn forms.

Say we are working on 1 Candj for a while —— then we eventually go to [redacted] Candj & discover its identical to Candi: We then go back to Candj & work on it as an alt. assoc. w. Candj's pc.

Similarly, whenever we detect a Candj identical w. a previous Candi, we go back & work on the previous Candi. If L search, we mark the dup. Candj, & we mark the first Candi as having higher pc than apparent.

A (perhaps) easy way to detect identity: look at pr on first n examples: if they are the same this is evidence that they are identical in general. (But they in a true optimization, so: maybe not such a good test!).

IDSIA

●: : There are at least 3 "refinements" that I'm committing in this preliminary QATM: (latest)

.01:209.02 1) T. Corr is always wrt. t. entire corpus, rather than wrt a part of t. corpus more directly related to t. (latest) problem

2) We do not address t. problem of correln. betw. cands.

3) We do not consider OSL.

.04:209.02: Continuing on t. "factoring" of QATM's Corr into $f(\cdot)$ & $\prod_{IA}(\cdot)$.

If each IA has worked on various sachems of t. various default units of time, we may want to take a γ power of t. resultant pc's ($\gamma = 1$ for larger $T(\geq CB)$, $\gamma < 1$ for small T: for small T, this small γ minus pc → 1, which is a way of giving that predn. "less wt." . It might be clearer as to what's being done if we take ln of t. product of t. pc's of t. A_i's — in which case, γ becomes a coeff of t. various $\ln(pc)$'s.

Suppose we have several IA's making a predn about An. How much wt. to give to each? —

Depends on t. "ssz" of each IA on problems of that type, & how "good" it has been for such problems

Essentially, t. γ's have been assigned to t. IA's by $F()$ (via "wts" = "pcs")

Hyp. t. opinion of $F()$ is doubld "ground" int. because that $F()$ gives small T to IA that it thinks would not do well on the problem.

(b) So we have w. small T, an inept IA working for a short time so We have #2 weakening effects.

● In General, t. meaningful Effect of $F()$'s assigning a T_i to IC_i, depends much on both IA's & t. problem. For certain IA, problem pairs, $F()$ could "as precisely as it can do" Soln. is obtnd in less than time T. (even tho $T = \sum pc_i$ w. ↑ kind of pc's assigned by $F()$.) ABcde

• Another possy is that IA is not able to do any thing useful in time T on this problem.

Going Back to .04 The ALP soln. to t. QA problem is more like:

$\sum_i \{ pc(Pam_i) \cdot \prod_j Pam_i(A_j \mid Q_j) \}$ So perhaps $F()$ could assign $pc(\cdot)$ to Pam_i.

But $F()$ looks at Qn first!

.27 A perhaps More relevant Approach! Given t. new data pair Qn_{t+1}, An_{t+1}; How much time

.28 does each IA get to "update" on this new data?

Another approach: Post each IA has a "shallow" Quick update" time & a "deep" update time. "Deep" in many cases can be platooby so longer time gives better update.

.31 Actually, there are 2 (apparently related) problems: (1) - Problem of how much time to spend on each IA: Updating each new $Q_j A_j$ (2) T. problem of using (IA's corpus & t. sort of IA's that have

.33 been updated at various times wrt various parts of t. corpus) to make a predn for a new Qn.

6/9/01 An implc. consideration in .27–.28 & .31–.33 is t. fact that t. IA's may "update" differently.

● Let us study some extreme cases of mixtures in t. update/predn problem

1) Say 2 IA's have been trained on a completely distinct corpi: they'd give their own mainstream d.f. for An_{t+1} w.r.t. not Qn_{t+1}! How do we combine these 2 d.f.'s?

Is this t. problem related to "META Analysis"?

7:30 start.
7:55 end.

.00 : 210.40 : Well, if we assume t. 2 IA's are "uncorrelated" (= "Independent"), then we simply Multiply t. 2 d.f's. If a IA has small ssz & poor success; it will have a broad, flat d.f. — and it will have little influence when combined w. t. output of an IA of large ssz & much success ... i.e. a very narrow d.f.

O.K. Great! — but In fact, t. IA outputs are usually correlated: they use partly same data & often similar induction Methods. — So its Really a very Common, Very General problem! To combine 2 or more D.f's that have been obtained in somewhat disparate Ways. This does seem Related to Meta Analysis.

Earlier work on this problem: 177.00-03, 177.16-.37; 179.08, 179.36

.10 To do mixing of uncorrelated Predictive P.D.'s: Mult them together: $(P_1 \cdot P_2)$
" " " " highly correlated (identical) " " $(P_1^{\frac{1}{2}} \cdot P_2^{\frac{1}{2}})$ if they are a bit uncorrelated: $(P_1^{\frac{1}{2}} \cdot P_2^{\frac{1}{2}})^{1+\epsilon}$.
If we assume (for Simplicity) that all IA's are Uncorrelated, we would multiply all together & get a spuriously narrow d.f.

Using t. Model of 179.09-.22, but w/ multiplication (instead of wtd addition) of IA pds, we get something quite diffrnt from ALP's "All Pem's Model"; & its not really justifyable from an ALP pt. of view. ! The idea is that F( ) would have info about t. cross/corr, as well as t. "wts" of t. IA's for each Q_j problem. So F( ) would give a correln. matrix for each Q_j problem and t. prodn would be ■ ..... ?

I'm thinking of a "Corr. Matrix" as a "poor man's joint P.D."

.21 : 177.37 Also Note! Incr. corpus coding scheme of 177.14-.37 : The pc of t. corpus includes pc's of each of t. IA's in addition to t. pc of $f(\cdot)$, (t. switching function). (Note "cheap rate"!)
The IA's in this code are not "11 codes". But, t. original pc's of t. IA's are a priori — so they will be large pc's (low bcasts), (Maybe zero cost; pc = 1)

So 2 problems ① How to make a prodn. based on this mixture of IA's.
② How to update t. IA's (i.e. which IA$_i$'s to update w.r.t. which (Q_j, A_j)'s; how much ■ cc to use?)
MAIN Problem
① is contingent on how ② was solved — so say we know soln. to ② — how to solve ①?
Do ① w. & w.o. correlations between IA's being taken into account.

6/9/01

Chris Wallace's method of coding (non-linear) regression,

h & u

u how much history

.00 : At the start, we just give the discrete params of the coder: i.e. how many coeffs it fits linear regression. We assume steady-state, so if we have n coeffs & history h ($h > n$) (Rwind). Then we know the previous h values of x_i, to start off. Since n & h are given, the coeffs can be computed for each predn. in a standard way, so we need given only the deviations from the predns., & this is coded using a Gaussian dist. So the entire code consists of specifying h, n & the seqs. of normalized deviations from predicted values, — This last depends only on σ^2 and the no. of data pts $= m$.

We can choose h & n so as to minimize total bit cost of the codes, or take all codes in ||.

[It would be interesting to take a choose seq. & get linear coeffs for it & note that one gets significantly better predn by using || coding (corresponding to a non-integral no. of coeffs)]

The main idea in the paper is that we don't have to code the regression coeffs. ↑

Also, the method works for X windo as well as the Rwindo treated in .00ff.

I think the coding method of .00ff requires infinite precision even if the values of X are discrete: (say 256 values of x.) Any roundoff error would probably accumulate: However, if we did restrict X to 256 values (say), and this was known by the "receiver", & we also told exactly how the calculations were to be made in integer arith, I think it would be possible to do it w. finite precision. Each predn. would be a discrete value of X (actually X would have to have a very larger range, but granularity $= 2^{-8} = \frac{1}{256}$.)

.00 : : In my recent work on "Min backtrack coding", I ended w. the idea that probly it wasn't a v.g. idea: That most codes started w. a large non-primitive part, that decided the form — & then they just coded the corpus in terms of that form. For such codes, Min-backtrack is not much good.

But for coding corpi in which the dominant form was only < 20 bits ($2^{20} = 10^6$) [Maybe 30 bits & 40 bits w. new faster Machines.. $2^{30} \doteq 10^9$, $2^{40} = 10^{12}$] I'd think be o.k. to just do standard Lsrch to find codes. For (analog = real) I'm in doubt, I persist in binary form by using a threshold. Whenever the price exceeded the threshold (up or down), we reset the price to zero & look for next Ulam-Schield crossing. This gives a seq. of ups & downs (0's & 1's) = binary seq. However, predicting 1M from a single seq. is not a good way!

Any way, deciding on a best set of instructions for the ref. machine is not trivial! Perhaps look at Kleene for terms of simple instruction sets for recursive funct. theory.

Or I might just try simple seqs like $(10011)^{(\infty)}$ or $(01)^{(\infty)}$ or $(0011)^{(\infty)}$ etc. as input problems.

Try to use II modes of Pentium or AMD. or use Soup's PS2 chip. or use 32 (or 64) bit II capabilities of Mathematica for Binary Arith.

SN In "recent" work on S89+ (4/1/01 – 5/1/01 stay at IDSIA & at Lugano)! I did have this = "proof" that the Prob was "within a factor of 2 of optimum" — This compared it to a human using heuristic srch methods. There were a few conditions that made it depart from this "optimality", but it still looked pretty good! It used Lsrch.

The most Recent work on QATM does not seem to use Lsrch, but it is not yet a "trained system" (to put it mildly!) — it may eventually use Lsrch, & it may eventually be people to use the various options, knobs, on it.

Just what was wrong w. S89+? Perhaps nothing! — That QATM was originally meant to be a Simplification — No Inv. probs no "general" OE probs — just QA (Operator Induction) problems

→ 140.00–141.00 was a review of S89+ & told what some things were! I think "Correcting" later conds was a big problem. Also, I don't know to what extent I was able to solve the problem of Getting TM/to remember doing Lsrch & use info obtained during trials on other conds during the same "Round" of T<2T Lsrch — or in analogous situations in II Lsrch.

→ Q: Can this last problem be regarded as an aspect of "correctn between the conds"?

Can the present problems w. QATM be applied as criterion to the "olde" S89+ system? The olde S89+ system would work QA problems by Lsrch on the IA's & expand the IA's by expressing them in a grammar. (QA would be regarded as an "OE problem").

.00: 177.37 : On the approach of 177.16-37 of having obs best tell which IA is to use on a problem. This seems diffnt from using obs to tell which operator (function) to use in xform. Q_j into A_j. The IAs has as arguments, all (or at least a large part) of previous Q's plus previous corpus. T. function only has Q_j as arg't. T. function is, (presumably) much faster than IA's. IA has as output, both th. proper ob ≜ th (micro)function that operates on Q_j.

.04

Some Random Ideas!

.07 1) 207.26 Envs a general TM GPD(Q) → an IA (Because we give a CB.) that xforms Q into a PDonA. — rather non-real, but diffl.

Could we use L search to optimize GPD(Q) directly?

One (somewhat al.) way: The update problem is to find $GPD_{c,Q}()$ as a function of $GPD_c()$ and Q. This assumes no backtracking. How can we allow backtracking if we allow f. touch to be from $GPD_{c'}()$, 'X' Q to $GPD_{c',x,Q}(.)$ Here $c = c'x$; (c' is prefix of c). (we have backtracked by "X"s. So f. "backtracking" function is ≃ a special case of a non-backtracking function. Normally, we will update using >> one Q,A pair, anyway.

$GPD_{c,Q}()$ is GPD() with c, Q known corpus.

.17
.14

As usual, f. amt. of Backtracking one will do, depends on amt. of CC available.

2) If, in QA TM, we treat Q as "f. problem dem", it becomes clearer that this QA TM is potentially very powerful.

.19 3) In .07 (1) we could start by doing f. updating by "Hand" or "ExpertSystem" — in which we pgm. f. updating system. (But f. updater system is in "factored form" so later, TM will be able to "understand" it.). After TM has learned enough, we will direct f. problem of "improving f. updater system". To understand the problem, & to have good ideas on how to solve it, f. TM will have to have a TSQ of suitable "definition lng" & examples of various updating systems. In 207.26 (.07): GPD() can be any induction system.

What we need, of course, is an "expert" induction system that is good enough to learn enough to improve its induction algm. (which is ≃ one of main goals of S 89. — Th. other way lng. to read that lang.). So all of this QATM work should be simply oriented to finding an adequate (but not necy perfect) induction system for QA problems.

However, in recent QATM work, T. TM was, even at f. beginning, able to learn the problem of improving f. Induction Algm. ! (was this "Overkill"?)

.34

.35 So, to start off, I could just use AZ141 & f. (≃) Learn TSQ's for Algebra. Use ≃ Lisp (or Logo power insts — perhaps not even recursion) — add insts, later. However, we do have to watch f. "scaling" problem! (perhaps Primarily by noting that f. PC of any abstraction is a function of f. situation calling for it")

Next, look at 140.00 - 142.40 to see how relevant f. "post S89 ideas are to .35 See that I do have a complete formalism for early Alg. TSQs

(25.00)

IDSIA

.00: 114.90! On t. "non-universality" of AZ141: AZ141 is "universal" on "the first level." T. method of updating it" is not universal — Gödel arg. in t. corpus itself t. method of updating has no way to be recognized.

Hvr, if I do t. scheme of Z14.19 – .34, in which CQATM is eventually able to work on "SI": improvement of t. system's own induction/updating schemas, it may be "universal" in t. needed sense. To attain this "Meta universality" it will have to have a suitable TSQ – of t. type mentioned in Z14.19 – .34.

At present, I have a very primitive idea of how to do updating in AZ141. As I get TM to work harder & harder problems, I will find that I have to "add to"/"modify" t. updating schema (= "expert system"). So I will be "learning along w. TM" & expressing this "learning" in TM's update/induction system. → Z18.01

.12

SN On t. justification of AZ141's method of coding/p.d. construction!

Say I have a system that I can decide, that takes any binary (or finite radix or perhaps countably infinite radix) sequences, & associated w. each member of t. seq. a positive integer (positive delimiter may be better) that is a unique function of t. symbols in sequ. preceding it.

.13

Then I can normalize (number nos. & record them as cond'l pc's. T. resultant dcrn. of t. functions constitutes a logical p.d. on t. countable set of all infinite strings over t. alphabet of .13.

In t. present case, it Logically realizes t. probs assigned by Z141. As to AZ141 (assignment of pc's to functions), its not yet clear in my mind. — Tho I guess it is a "well darkened" assignment of pc's to all functions in t. lang. used by AZ141.

Hvr, because of t. possibility of many different parsing methods, in practice, Z141 & AZ141 will not give unique dumb probability d.f.'s. Any (uniquely derived) parsing of an (Function/n-tuple depth) system will, hvr, give a unique coding & a unique p.d. for any corpus.

.27 Another term/duality: in AZ141 (applied naively to Gödel but to code. t. coding of a sequential (corpus — say Nat/loops) — to code a letter A following t. letter C:

If we have darkened this condition, then all previous cases of A following C must be accounted. Do I have to (specifically, specifically) normalize Req's A until all occurrences of C in t. past, C in which A might have followed)? — or do I just write a no. for Pr(s A) = no of times it has followed C in t. past, then normalize wrt all other possible symbols (t. most associative)? I should get t. same result. — if not, t. system is inconsistent: Suppose C→CB had also been darkened but not CC or CD (say we only have 4 letters in A(phabet). I think I have to resolve t. ambiguity of coding before it becomes a "well-darkened system."

In Pure Z141, we only define n-tuples, & there are no

6/11/01

ID512

∴ Automated "Lapeca's Rule"

-.00: 215.40: "Intersymbol constraints" (other than those determined by the n-tuples). We have a purely Bernoulli seq., but we certain cases associate w. the ordering of symbols.

.02 → In the cases of 215.27 ff, I will have to clearly determine what I want the pc's to be!

This Q will become very important as the TSQ grows ("SCALING") because I will want (objects definitions) to become more & more "context dependant" ← (Gray/probabilistic context" hvr)

So: Say at present, AZ141 can assign a pc to any (conceivable function. [Ambiguity (multiplicity, parallelism) in parsing is a problem, hvr]. So we have a function (or perhaps several functs) that ordered the corpus $\sum_{i=1}^{n} [Q_i A_i]$ a deep hierarchy; we add Q_{n+1}, A_{n+1} to the corpus: What are some good, "fast" ways to get a new funct for the augmented corpus? ("Updating"). So there's the big problem. I do have one method: In which I find a cheap funct relating A_{n+1} to Q_{n+1}, using pc's w.r.t. old functions, then I find a cheap way to concatenate Q_{n+1} again using recognition functs on the old functions. Then I have the larger funct that works on the augmented corpus & I try to "compress" it ← (which I'm not at all clear about!)

So: Get a reasonable TSQ & see how I seem to do updating & incorporating in the AZ141 system.

Conceivably (probably?) OSL can be used in .14 & .16 to get "cheap functions": These are functs that have not yet been defined, but defining them is cheap because they have occurred once before (in undefined form)

Note again that Z(4) & AZ141 applied to operator (QA) induction, are quite different. Z(4) is a way to assign codes (i.e. pc's) to a sequential corpus. AZ141 on Op-induction, can assign good (risk) pc's to functions formed from a primitive set, but it does not tell how to derive by pc functs that describe corpus. For a small corpus, we might use Lisp enumeration functs, but for larger corpi, we have to derive a good system for "Updating" & "Min back tracking" (hvr, see 214.13 - the key way/min backtracking is the same type of problem as ordinary "Updating").

OSL one Shot Lrng.

.00: 216.90 : In OSL (beta Z141 & AZ141) Once can think of it as: In coding a new chunk of corpus, one normally has a choice of all primitives & all defined symbols. In OSL, one adds to this, the set of all definable contiguous strings of symbols that have occurred in the past. So if "AB" has occurred once in the past, (but has not yet been defined) — then we can (tentatively) use AB as a defined symbol in coding. ["tentatively" means we have to see if defining it gives any compression — the analysis of OSL in my writing a book we will give (perhaps easy to see) conditions on when it is likely that a set of symbols will be useable in OSL.

.14 An idea in AZ141 related to OSL coding: Say A is a larger operator: It has α as one of its sub-operators. Then, A w. α substituted for β in A (B is an operator w. same no. of I/O's (is this what "arity" is?). will yield an operator w. B fairly by pc: The reason is: A & B can be easily coded using a dummy variable function x, that can take the values α or β. [I'm not entirely clear on this entire process, hvr.] [I'm not entirely clear on this entire process, hvr] ←Koh-I-Noor

.21 The reason this is like OSL, is that we (redefine) both the old & the new Object → 219.13

IDSIA ABCDE

Guida

.00: 215.12 : So 214.19 - .40; 215.00 - .12 looks like t. most Reasonable, Simplest Model.

.01 Its not far from what I was working on in Saarb: But I now know 140.01 - 141.40.

.02 But what about t. Guiding idea of SBGT : of f( ) that looks at a problem & decides on how best to work on it? T. F( ) idea was also (sort of) used in obtaining a distrib pc of a bias to be used in restricting trials — as a function of t. "problem type" — This seems essential in dealing w. t. "Scaling" problem. See 141.19 for a little discussn of "Scaling" soln.

.06 Well, in QATM, There must be a part that recognizes what kind of problem is being presented, So it knows what sub-function to use on it. (The Ob-op algebra). So QATM looks at a problem & it has a "stuff Ob" to categorize it (Math or Physics or Chem or Linguistic type problem) Next it will have, within each category, subcategories to determine what funct. to use for part to answer.

Learning These Categorizations is a regular QA problem: After TM has done it w. some success (or has been told by indices a lot of categories) on problems 205.09 discusses use of such "indices"

So, initially TM may have a PD for each category of input. These will initially be labeled as to "type".

.17 In .06 ff note 214.00 - .04 on difference betw. .06 & 177.16 - 37 !

In both of them, we have Q as input & t. Soln.(s), A as output. In .06 the output is rather fast. In 177.16 we first go to an appropriate IA; which solves t. problem — (but could take much LC to do so!). The final output of 177.16, hvr, is the (cheaper) function of .06. One difference is that in 177.16, there are much fewer IA's than there were "solns" in .06. So .06 maps to a much larger space. In 177.16 there may be 10 or at most 100 IA's. In .06 there will be a distinct soln for each kind of problem.

In .06, Q might be what is square root of 256? "Answer could be a routine for getting Sq. roots. In 177.16 t. answer might be a routine for optimization of $X^2 - 256$, or a general system for solving Alg. eqns. (i.e. $X^2 = 256$).

There ?

So, it looks like there is a "Continuum" embracing .06 & 177.16 : .06 focuses on a narrow part of Q space & gets a quick, sharp, answer. 177.16 makes a broader categorizn in Q space & finds a IA that can work all probs in that space. QATM, regarded as an IA, would be an extreme form of 177.16 : There is only 1 type of problem, & may be all solved in similar way ("put them into QATM").

.30 I think .06 is a specialization (specialization) of 177.16 → 219.15

.31 t. Berge has been discussing of (apparently) "problem solving modes":

Another Mode is "Updating", in which we give TM a new QA or set of QAs. This can be regarded by TM as a special kind of Q : But do we have a way to tell TM when an Answer is "correct"?

I have been Raising of "Prob-Solving Modes" as being rather trivial : That TM just applies whatever t. Current Macro operator is — to t. current Q. It is Conceivable that t. output of that operator is stochastic — so there could be several probably wtd

6/12/01

IDSIA

219

The Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139
Telephone 617 661-1213

218.40
.00 : 218.40 "outputs", & each output could involve considerable calcn.

I had Prod. of t. diff'lt parts as being t. "Updating": & Modifying t. Stack Operator To accomodate t. Augmented Corpus.

.04 T. (Mostcommon main) case of complexity of calcn. in prob-solving Mode" occurs if we have OSL, or perhaps t. Special coding of 217.14 (A bit vague! — (which is related to OSL)

Non-OSL prodn. is prodn. w.o. new data. OSL is prodn using a new datum. —
In QATM it is implemented by looking for previous occurrences (pcats) of parts of t. new Q.
Note that OSL is normally probistic: T. "soln" assoc. w. t. "One shot" need not be very large. — it can be ec t. pc of some non-OSL "solns".

In OSL, one does a bit of updating during & problem soln. phases (Also perhaps in 217.14.)

.13 : 217.21 217.14 is (closely related) (closeness w. Pcats metric distance) in QA. It says that OSL occurs if something "close" to t. present problem has occurred (at least) once in t. past.

37"

.15 : 218.30: After doing "solving" a problem viz 177.16, we try to port into it 218.06 form by narrowing down t. initial pd on IA's (at least) — but literally, it's a QTM problem, we want something narrower (than a "soln") than a v.g. IA!

So: 2 things to understand: 177.16 v.s. 218.06; & prob solving v.s. updating.

"Updating" can be regarded as a "regular problem", but only a very smart TM could usefully work on that problem. — so we see it for TM's "more creative uses."

~~Sam usually thinks that 177.16 part~~ So! It looks like Pcs: often

When Q comes in, TM is able to (recognize) decide what kind of problem it is & quickly devise a soln — which may take a long or short time. This would be 218.06
If F() has no v.g. idea on how to solve Q, then its category of t. type of soln. needed is more problematic — like over IA's.

[but more generally, t. "Q" can be any kind of problem down! — I Prod. of all Q's being "induction problems" — while this is literally true, t. whole soln of "induction" problem will be an OZ or INV-problem. — Instead, t. "induction" part could just be t. part in which TM acquires an understanding of what t. problem is. For a problem stated in terms of Cons that TM is very familiar, t. induction part could be a trivial part of t. problem; T. main part could be an OZ or INV problem.

.36 Perhaps a more general idea as to what t. QA problem is: That A can be a prob. solving method — which it would have to be something like that for OZ

.39 problems — since we usually don't know t. "correct" answer.

.39 — [But in fact OZ probs are expressable as pure induction problems, if we regard

17 15
10 16 24 37
10:13 –
10:13 –
10:19 – 6 43 37
15 – 13 36
57 38

data
177.16: F() → pDown IA's.
218.06: F() → soln. of problems.

TSQ's & SCALING .19ff

The Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139
Telephone 617 661-1213

.00: : Some General Remarks

1) Quantum K cost is probly same as Regular K cost.

2) Resource limited/K cost Quantum may be diffnt from ~~Natural~~ but things reasonable this Q may be as difficult as NP à NP

3) (Hur, Q's about RL K cost dependencies: keep metathmor schanhage (~~turing~~ storage) machines or - - -
Obtaining Practical Probly values is not nicely solved by ordinary RL K cost — This may also apply to QM computers.

4) On Hal's conjectures that all of Modern Math is written in 100 or 200 bits of a certain Umc.
& Lavin's conjecture that ~~t~~ distances smaller than Hal's, but still within Practical distance. 10^{20} is now a practical $\sim 2^{68}$ $10^3 \to 2^{10}$
This distance is the amount of "choice" nacy to create Modern Math. So if we come into contact w. an Alien culture, this tells us ~~t~~ probly that they will have the same/Modern Math as we do. Hur, there will be a smaller/section of Math ("core") (Not included in all of Modern math) that will be closer to the Umc. —
As we have more (likely kinds of) having this "small core" in common w. the Alien Culture. The "choices" involved in deriving Modern Math, may be have been made by applying Math. to Physics, chem, logic, sociology ....

5) When humans are shown Algebraic Expressions on paper, they know & know that the ~~obs~~ position on paper is not important. This (& perhaps other ~~w~~ ideas must be somehow (person conveyed to/and by) to TM.

(.19) 6) On TSQ's a "Scaling". Each Concs ~~must~~ can be found by Lsrch, only if the Lcost of Each new Conc. is < CB threshold. This Lcost must be in terms of the conditional Lcosts of all of the old concs involved. So when a new conc is discovered, we are always concerned w. its conditional pc: Under what conditions is it a likely "candidate"? Unless we have usable conditional pc's we have an unsolved scaling problem. Unconditional pc's tend to give a "scaling problem".

.27 To some extent "vacancy" can be a usable "condition" — it amounts to a short short term Memy. Say we Use "X Windo": The "T" can be updated periodically, to give Lcost results for entire corpus or for an "Super K window" w. some larger, fixed "T".

This might be a good enuf "conditional" pc for "early ish" TM — until it is smart enuf to discover better conditionals. It may be that I can study my own prob-solving & get ideas for good conditionals — e.g. "Why" did I quickly decide that a certain conc. might be relevant in a particular (sub) problem?

If I use minimal conditionals" (I have to use some) & large CB for Lsrch, I will tend to have more "Diversity" & ∴ more "Creative" Solns.

6/14/01

IOSIA

Optzn. Technique

.00 : 219.40 : OZ probs as being defined by being able to predict, w/all, how good each OT will or be for any particular problem for any particular CB.]

Inv. probs are almost always solved as OZ probs, rather than by direct "search".

.03 : 219.38 So, in line w. 219.36 : The Q is mapped to a direct soln A, if poss'l. If not, it is mapped to a method of finding A from — which may be Th. 219.39 — 221.00-.01 methods for dealing w. OZ probs.

.08 Would it be of interest to do a QA TSP "like" SAARB — in which the soln. to each Q was a relatively simple funct. (tho it could have a large CC)? Since we are doing search, we have some bound $\frac{CC}{PC}$, & find solns within that bnd. — which was the idea of SSG & SAARB..

To what extent would .08 be a serious step toward a [redacted] ("Terminal") TM? ("final")

.17 (6/15/01) Consider several QATM models & the TSQ'ls Corp'i (≡ TSQ'ls) that they could (deal w. / make out w. satisfactorily): Describe Update phase & Prob-working phase. (They need not nec'ly have a separate Update phase .... but ....)

.20 Model(1) ≡ (.08) SSG, SAARB: In update phase, TM tries to find/function that will map all Qi into Ai. (This is a MTM). In prob. solving phase, it simply applys this function to (initially, Lazy PC) the new Qn+1. "Update" can work in at least 2 (ways): (A) It looks at the [redacted] complete [Qi Ai] Bag & tries to find a formation of non-el. approach) & probs (uses) Lsrch.

(B) It looks at the F( ) for $\sum_{i=1}^{n} [Q_i A_i]$ & tries to modify it minimally, to fit Qn+1, An+1. (C) For some Backtracking (part way between A & B) try to update the F( ) of up to Q_k, A_k ($k < n$) by using sub corpus $[Q_i A_i]$ $i = k+1 \mid n+1$

.32 N.B. in (.20 (1)) We always use the entire corpus to make predns & to update.

.34 Model(2) F( ) looks at Qn+1, assigns wts, pi to existing set $\sum [A_i]$ of Induction Alg'ms.. & default f( ) from model(1) of (.20) We do Lsrch over the $[A_i]$'s to solve the problem, Qn+1. Updating F( ) [redacted] as maximizing the pc's of the probs values assigned to OZ problems. If Gopc is that obtained in (GPD1, GPD2) MCTanalysis. We have F1( ) & F2( ) as in the GPD of [redacted] SSG+. This updating on F( ) can be done (as in .22-.32), (A) using the entire corpus, or as (B) a small modification of the old F( ), or (C) as a [redacted] various sized "backtrack" of the corpus)

simplifies for Model 1 is always used the entire corpus for updating (?) 2) MTM 3)

.00: 221.40 : IN Mod 2, we also update t. IA's: Each of them (t. IA's) has its own update procedures (for both finite & infinite $\subset B$)

SN A common problem: FC() works w. corpus thus Qn An; but not Qn+1 An+1 !
Model 2 (221.34)
How can we "minimally Modify" FC() so it works now "n+1" ? { This "min Modify" of old FC() is an approx; but ways to get a good FC() that fits all of t. data. Remember that we want a final goal, to get FC() that is good for ALL of t. Data.

The idea of "minimal Modify" of x to produce y is usually put in to t. Should be updated by amt. of Lcomp to Lgen.

:07 $P(y|x)$. My impression is that t. ref. machine used to compute this → $P(y|x) = \frac{P(x,y)}{P(x)}$?

But isn't it true that $P(y) = P(x) \cdot P(y|x)$? i.e. $P(y|x) = \frac{P(y)}{P(x)}$? No! , this is RITE!

Anyway, t. P values in .07 can be approx obtained from ALP: In $P(x,y)$ we want to find short codes for all x,y pairs. (.18)

.10 Hvr. In Mod 1 (121.20) we want a funct that has about t. same values for most of a certain old part of its domain; & new values for a new part of its domain. We "update" what to port of t. domain

.11 .12 heuristic Methods of doing this. One, of course, is to design an op that recognizes t. new domain part, & (from) a special operator must be found to map that domain section on to its proper values. [SEE 224.18-.23 for more exact def of "sim"] → 224.10

.16

.18 : (.10) Here "x" is $f_{old}()$ & "y" is $f_{new}()$. When thinking about a metric from $f_{old}()$ to $f_{new}()$

If $f_{10}()$ is very large, then it has lots of info about t. past corpus — that, presumably, would be relevant to Creating $f_{new}()$.

.21 We have t. successive pairs $f_i(), f_{i+1}()$: this does give a p.d. — a stochastic

.22 ● xform. from $f_i()$ to $f_{i+1}()$. → 224.00

T. idea of .21 — .22 might also be used to find a good probabilistic Mutation

.24 Operator in GA. Ideally, would want a cond'l. mutation operator — t. "condition" of inp t. problem dcmn. For pair-wise "crossover" or m. cond on all pairs.

Crossover: take t. set $[cond_i, G_i]$ $i=1|n$: a from it, get a pd. $cond_j, G_j$ pairs. To do good Mutation & crossover, it is well to have a good language expressing t. $\sum [rand_i, G_i]$ corpus. — T. lang. should have useful concs in it — useful for creating new conds in any G range desired.

I did consider .24 ff a lot in my analysis of GA, but I wasn't able to find a

.36 good stack lang class to try to fit to t. data. It would seem that I might be able to express a cond or mutation by $G\begin{pmatrix}\text{functs}\\ \text{conds}\end{pmatrix}$, as being comprised of a small no. of my pc. sub functions. A smaller no. of my pc funcs would generate more conds of any given complexity than a pf & .·. my expected G. We would have to first search thru these → 223.00

GA's

-.00: 222.40 hypc Cands first. We just search thru cands in pc order! We do try to find common "subfuncts" in hy G cands.

Perhaps t trouble w. 222.36 ff was that t. long did not fit yt data very well! — i.e. It generated lots of cands of hy pc Ci: hy expected GJ — but for t. most part, very few of those cands had hy G. — The cands of hy G that occurred almost always had much smaller pc's — combins of a few no. of "sub-functs"

— But, may be not! I imagine that usually successful funct is composed of only 2 or 3 or 4 sub-functs — There are too many "larger" functs to search over.

— So the "fit" may not be very bad!

There are functs. that are "very large" that are very Good. — But they are discoverable only if they have sub-functions that are "discoverable" sub-functs of other hy G cands.

There is something Bad about t. forgg: We tend to use mainly t. same parts of hyish G. cands. — which Gives us low diversity. We may get an Excessively **elitist system.** Well, we could get arbtly amounts of diversity by γ-flying the pc's w. $\gamma < 1$. $p_i \to p_i^\gamma$. We ~~probably~~ also do renormlzn. — but it probly can be done — .18 may be not nearly uniquely! In general, given a set of $p_i \to \sum p_i = 1$; there will be a min value of γ for a limit. $\Rightarrow \sum p_i^\gamma < \infty$

One reason for Wanting to do renormlzn as we generate t. cands: If we do it rite, t. $\sum p_i$ will converge: Otherwise they may not.

$\sum_i p_i^\gamma < \infty$ e.g. If $p_i = \frac{1}{i^2}$ then $\sum p_i^\gamma < \infty$ if $\gamma > 1$.

Say the subfunctions are f_i w. p_i = pc's. $\sum p_i = 1$, but we choose γ larger enough so that $\sum p_i^\gamma$ converges. Using this normalized $p_i' = p_i^\gamma / \sum_j p_j^\gamma$,

we can generate new cands that have pc's that are simple products of p_i''s of t. component functions. — These products need not be renormalized since they sum to 1.

{ Perhaps Go back to my (not so distant in time) work on GA "loops" & see if t. forgg, answers my objections to t. models I was trying at that time.

In "SGA" I tried to find a stock loop on t. cands, $\Rightarrow G^\gamma$ fitted t. p_i's of the loop as well as possl. For any G funct, there will be a lower bound on γ so that $\sum_i G_i^\gamma$ converges (see .18) & (18R)

To find a γ to get a G^γ exact. to "match" a p_i funct., look at $\ln G_i$ & $\ln p_i$ as functs of i.

If I may use both i linear in i, then γ is the ratio of their slopes ← NO! If $G_i = \frac{1}{i^k}$ then

$\ln G_i = -k \ln i$ — which is not linear in i — it is linear in $\ln i$ here — so see if both G_i & p_i are linear in $\ln i$

6/16/01

IDSTA

LOGICAL/Mathical Reasoning -.04

MODEL 1

Summary .10

.00: (Spec 222.22): Inducing $f_n \to f_{n+1}$ P.d. from observing $\forall$ $(P_i \to f_{i+1})$-$(i=1|n)$ pairs. is a rather gross way of getting f_{n+1}: Usually one does much "reasoning" to get f_{n+1} directly, or to narrow down the poss'l. choices, considerably. — Tho I haven't worked out just how this "Reasoning" is to be done. — logical reasoning; formal logic.

.04 (One way w/ to have TM do "reasoning" problems as part of its TSQ — Then it would have to realize that those techniques could be applied in other domains — i.e. its "own" problems solving — it's "Meta problems".)

It is analogous to the (occasional) discovery by students, that the techniques used to

.08 solve problems in school, could also be used to solve problems not-in school (

.09 → 222.16 Spec: The 2 operators of 222.12-.16 (could should) be related: Ideally, find them Both Simultly.

.10: 222.16 (Model 1): I Guess Model (221.20ff) would "work": The parts "needing work"

1) Interchange "update" is always for the entire corpus, since the Solution function $F(Q_i)$ is for the entire Domain of poss'l Q_i values. I'm not entirely sure this is true: Tho it may be literally true

2) The "update" problem is not clearly solved! I have some "kind of" "soln": 222.11-.16:

.18 More exactly, $F(Q)$ works for $Q_i \in A_i$: $i=1|n$, but not for $i=n+1|n+k$.

.19 To update we do 2 things: (a) find an operator $F'()$ that works for $Q_i \in A_i$ $i=n+1|n+k$, using some of the concepts in $F()$ that work for $i=1|n$: we want minimum cost for this $F'()$.

.21 (b) find an operator that can distinguish exactly between the set $\{Q_i\}$ $i=1|n$ & the set $\{Q_i\}$ $i=n+1|n+k$. w. minimum cost. (c) (d) (e) create a new operator $F''()$ that works for $i=1|n+k$:

Try to "compress" this operator (Note .09)

.23

.24 I expect that as I design a TSQ for Model 1, I will find ways to solve it & use some of those soln. ideas to improve the update schema of Model 1.

Also, I expect to be using the A2141 formalism to assign pc's to compiled functions:

.27 (possibly using a variant of LISP one of my own "Functional Lang.".

Model 1

As I see it .10-.27 (& the preceding part on Model 1, are pretty much what I was doing in SAARB. It was a MTM. The probs I've solved since then (Notes impl. (1) better ideas on how to deal w. scaling problems (see 220.19-.40 for recent ideas) (2) that I only used defns of previously solved problem & primitive funcs: I think this was true because the TSQ was very small, & I hadn't yet developed funcs that were complex enough to need "subtrees" of earlier funcs. (3) In SAARB's soln of probs, I did use OSL much — the soln to a past problem was automatically put in the "STORE" & given a pc = (total no. of funcs in "store")

[This, of course, led to a serious scaling problem.] I did not use OSL (or any other method) employing useful subfunctions (= subtrees) inside Past $F()$ functions.

225.00

IDSIA

Model 1 summary (cont) .01

.00: 224.40: Say I Got Mod 1 runny o.k. & Got it to do a large TSQ & got it to solve diff't probs.

.01 Some direction of Gonzn: (1) At all times, have > one FC) on hand, so that mod'n to deal w. new problems is more likely to work w. at least one of t. FC)'s.

.03 — This may facilitate a bit of NMTM.

(2) Do NMTM. Some remarks: (2) AZ141 is already probilistic — but this is a whole problem for Mod 1

.05 (b) Finding Many FC)'s would be one way to realize NMTM.

(c) I may want to Go to Mod 2 (221.34) to Get [redacted] NMTM.

(d) What are t. Limits of Mod 1? .36 Is BIG Limitation! INV probs would seem to be MTM problems — but Normally, they are solved as OZ problems, which are not MTM probs. Mod 1's method of operation uses AZ141 — which is a probilistic system — so on t. META level, Mod 1 is "probilistic" — is this enuf to solve OZ probs?

(e) Mod 1 can [redacted] be modified to work probilistic probs in fole. way: The FC) that is sought becomes a dcrn of a stochastic operator. This dcrn can be single string (of which it can be a set of strings — as in .01-.03 & .05). We hunt for this FC) in t. same way as we did before, using same update schemes.

(f) If Mod 1 can then work all kinds of probilistic QA probs, it is a "complete TM" & can work on SI.

(g) See 214.19-.40; 215.00-.12; 218.00-.01 for T. dcrn of ≈ Mod 1; that is ≈ "SAARBTM".

(h) (g) I (not "g" was "8") M1 is really solving induction probs! (whenever) How does it know when/how to stop in its search for a short code? — I guess it just picks t. first code that fits every thing I've (2) searches to find t. seen & looking for a good code for t. entire corpus! Normally it will do [redacted] 2 operations of 224.19 and .21. This Gives a ticket. Next will "work w. t. whole corpus" — but Then we want to compress. This "dual operation". T. criterion for "adequate compression" is unclear. Also, I have no idea on how to do it — But See 224.09 for G-lineage of an idea → It integrates searches for t. 2 operations sw. search for pair integration

(i) (10) I expect that much of t. uncertainty in t. range [redacted] are supposed to be fixed by suggestions & that of I get by watching TM & ME solve t. probs in t. TSQ. Q

(j) (11) The method by which TM is supposed to discover & employ heurs is unclear: I want to plan a plan observing on My Prob. Solving → (10)↑

Remarks on M1 (k) A Big deficiency of M1: I don't see how it can discover/use heurs.

.36 .37 (l) M1 uses AZ141: It's not clear how pc's of cases can be made Conditional on t. nature of prob. they are used to solve. W.O. this feature, we have SCALING problem. See 220.19 for dcrn of scaling problem: T. "recency" condition of 220.27 may be good enuf for certain kinds of M1's TSQs

→ on Mod 1 229.00

→ on Model 2 226.01

ID

.00 : 225.40 : **Model 2: 221.34 – .40; 222.00 – .10; .18 – .22; 224.00 – .08 :**

I Guess the Main Idea of Mod 2 is that we start w. this "Normal" version of QATM: F( ) looks at Q_i & outputs a P.D. on A_i : we search on this pd to find the A_i ? — wrong

If this is a MTM : Then a NMTM a pd on A_i would be the soln ! [redacted]

A non-el NMTM : F( ) looks at Q_i & outputs (conceptually) a PD on A_i : Here F( ) is a "Procedural P.D." so its output depends on how much effort was Given.

While this F( ) could, in theory, be found by Search : (finding it is the same as the General Operator Induction problem — one could do an Search over Lisp (say) defined functions. — (For MTM's) : If F( ) is a pd, then we can also do the Search as for MTM, but we retain many of the shorter codes for the $[Q_i, A_i]$ corpus. So we have a formal soln. for very larger cc, for both MTMs & NMTM.

For finite CB, the problem becomes More Interesting : In steady state, how does one do "updating" (we can start out w. a small corpus & do simple Search to get the many pgm codes that define f( ). ... So the "steady state" update problem, is the Main problem.) "Updating" means — How to obtain $F_{n+1}( )$ from $F_n( )$ & (Q_n, A_n) (or $[Q_i, A_i]$ $i = n+1$ to $n+k$.) (Actually from the codes or "params" of F( )) — corpus Augmentation, See, 220 for "conversion"

.17 In general, there are many IA's that can work on the ~~NMTM~~ NM-QATM problem — & each will have its own updating Alg'm.

.20 SN .17 isn't exactly updating how updating is done: Usually $F_n( )$ is defined by a short code or set of params. It is these params (or this short code) that are "updated" by the $[Q_n, A_n]$ info. — for an example, consider updating Bernoulli, Z141, AZ141 — in each case, the params that define the "latest predictor" are updated — usually in a simple way. In fact, this seems like a good way to look at

.23 (many/most/maybe all?) updating schemes. Before Q_n, A_n came in, there were certain kinds of progs that were defined by F( ), & its short (codes/params) told how to do this. When the new Q_n, A_n comes in, we will either have to see how it fits into the old defns — i.e. may make "modifications of params" necy. — & / or , it may make the data or a new kind of reqy necy (as in Z141/AZ141) — so we add (new parameters).

So Z141/AZ141 is a good model for (updating induction) in general — (at least the way I consciously do induction/updating.) — studying

An addition to changing params (progs), is defining new params (new reqs) : Updating can also reparse the corpus in ever or more ways. "Reparsing" is neither a modification of progs or a new defn, but it can be a very important feature of "Updating". It could be a new way to combine abbs to do predict directly or to code the corpus

→ 227.00

6/18/01
ID

.00: 226.40 : Look at some other induction systems & their **Updating Algms.**

.01 1) Feed forward Neural Nets: (This is a QA TM) : "state" of system is given by values of "wts":

There are various updating Algms on how Qn An affects these wts. The "partial derivs" is usually common to all, but Momentum & "learning rate param" are some other parts of the update Algm. Normally, cross validation is used in ANN runs. Here, if we use "easy Platonic" criterion, cross-val is not needed. Cross val would prob make updating more diff — — If we were to try to update by doing trials "near" the last "best pt". → see .34

2) GA: These are not normally applied to incremental ("tiny" — but one could (in at least the following way): One has a corpus & one has found a population that "solves". That corpus say it's a seq. of QA pairs like $[Q_i, A_i]\ i=1|n$: The population tries to get good codes for $Q_i \to A_i$ using a set of primitive (possibly universal) operators. We use & updated Alp Gorc. formula as a "fitness function". When a new Q_{n+1}, A_{n+1} comes in, we start out w. the old population that "solves", and we do the normal mutation/crossover routine to solve it. A $F()$ is defined by a population & its fitness values. .17

While $F()$ is defined as .17, the execution of $F()$ could involve picking the best cand. & the fittest cand & applying it (or a mutated set of it by fitness rounds) to Q_{n+1};

So that $F()$ is at an list the same as its params (or its derm/coeds)

3) Decision Trees: I'm not (at the moment) familiar w. them! But say we have some data (corpus) & we've fitted a good Dec. tree to it. There is a standard MDL algm & Quinlan, for doing this: If we augment the corpus we could update by applying the algm. anew to the augmented corpus — but the idea of "updating" is to save cc & so we try to modify the old dec. tree "minimally": Changing it would "Repeating" would seem to be too diff. — but we might do various small mutations of the Dec. tree to try to accommodate the new data — This would be in the spirit of "updating".

.34 4) Linear/non linear Regression. W. a finite sample, one finds a good pt. in params space. When Q_{n+1}, A_{n+1} comes in, one searches near the old pt. for an optimum pt. to include the new datum. (ANN of .01 is a special case of NL regression — other functional forms can be used: linear sum of Orth. functions, non-linear functions, or (Barron's) better ways to do non-linear regression (of which ANN (.01) is a special case).

INDUCTION ALGMS

• means "Universal".
~ means ~~can be~~ Stochastic.

.00 : 227.40 Somewhere in Russo IDS/A notes, I made a list of about ~~15~~ distinct ~~kinds~~ kinds of Induction Algms (IA's) : can't find the list. (It's 150.33 - 151.15) ← [Has Good Discussns!]
Has

Try to reproduce :

~ • 1) ALP w. CB=∞

~ • 2) RLP (=ALP w. CB<∞) : A sort of LSrch.

~ 3) Z 14 : (Born Sapusek w. Dartmouth Dens.)

~ • 4) AZ14 :

.07 ~ ~~3~~) Linear / NLinear Regressn. Wiener filters, (Kalman filters?) ~~Linear~~ NL. linear Regressn can use ANN

• 6) ANN & Barron's Generm of Perm. & [RANN] ← Recurrent ANN [ANN w. outputs → inputs]

(GA) not IA's, but can be used to update IA's. See 331.13 for more details.

? • 7) ("normal") GA & SGA & Sim. Annealing.

~ • 8) SVM Support Vector Machines (Vapnik & Chervonenkis) : read Part of Future I have in Postscript dir on C:\PS\

? 9) Decision trees (CART)

? 10) Max Entropy Method (Jones) & perhaps Ron Christensens Gen of it (I Think It's [redacted] is not a "special case of ALP" The Reader says it is.

~ 11) Grammatical Induction (related to Z 14 & AZ 14) — Can be Universal if Grammar class is adequate.

~ 12) Hidden Markov Models (related to [redacted] but more specific) I'm not sure, but I think it's usual machine is ? techniques for optimization of continuous params.

~ (3) "Usual Statistical Methods" whatever Bessel's are ! → Include Stein's ~~ma~~ Method. The discrete str. of t. state space has to be (Guessed) ? guessed by the USK.

~ • 14) Pradn / discovery in t. Sciences : This is example of N-L predn/models (not on ~~inca~~) but the Heuristics may be quite diffrnt, ~~from Simple ma~~ for guessing good N-L models.

? 15) Explaing Based Lrng → 277.30 ff strikes this : EBL seems ≡ "compression" ≡ "finding Regys in data"

? 16) Case Based Lrng. [related to : Generm or specialization of OSL (One Shot Lrng).

17) Bayesian Nets (Pearl)

ID

Z141 / AZ141 (i.e. Non-Universality / Non-extensibility)

Sect Mod 1

225.36

.02: 225.40: (M) What is Min augmentation/modification of Mod 1 that can get ① Trys of heurs ② Conditionals!

pc's in AZ141 → (i.e. 225.37). T. Solns to these 2 probs could be closely related; nature of problem.

Often heurs will be modifn of given P.D. conditionals on

.03 Poss'l. ways to solve them both: 1) In my working TM's TSQ: see what heurs I use

.04 & see how they could be derived by "mod", i.e. modify M1 so it could do "that sort of thing"

2) A way to solve them both (& perhaps Implement .03–.04). Modify AZ141 so t. pc's can be arbly general functs of t. past. (that AZ141 was not "universal" on that level, was an impt. worry I had about it not being "arbly extensible" i.e. arbitrarily

arbly "smart".)

One way to do this would be to apply AZ141 to itself at a hyper level. — 10:23

.12 as is, AZ141 can construct an arbly function: it is "universal" in that sense:

But it has a narrow sort of poss'l. d.f.'s on that set of all poss'l. functs.

Somehow, by "applying" 141 to itself on a META way, we will be able to get arbly d.f.s on t. set of all functs — because this Meta Z141 is "universal" in t. sense of .12.

.17 So this could solve (4) critical problems in TM. (1) 225.36! — Discovery of heurs.

.18 (2) 225.37: pc's w. arbly conditionality (3) pc's that can be on arb. functions

.19 all poss'l. functions! e.g. in Sol64c: In t. "All Items Method", I want to be (4) Q ATM should work on NMTM problems

able to do arb. on arbly per on "all poss'l functs".

Sect Ron 140.00–141.40 May be of relevance!

Also N.B.! The last time I worried about t. Non-universality of the AZ141 d.f., I don't think I had any specific objections in mind; My objections were at "Theoretical" (I think); It would be Great if I could find that stuff (reference)

One reasonable way to work on this necessary Expansion of AZ141 is

.03–.04

Q: M2 did not (apparently) have d.f.'s of .17 (1) & .18 (2) (I'm not sure about .18 (3)) (heurs; pc w. arbly conditionality)

Did it do this by using AZ141 on a META level? ← I think so! .19 (4)

On t. lowest level, M2 worked on probabilistic D.F.'s (NMTM rather than MTM) — so it was able to use universal probabilistic d.f.'s for SI (& Meta problems).

Perhaps if I had a good way to give M1 a Meta Universal pd, it would work & be much simpler than M2.

→ 232.00

6/21/01

ED

GIAP

.00: : When I program a computer: It does what I told it to do, not merely what I want it to do.

When I pgm GOALS into an Intelligent machine, those goals will not nec'ly be the goals I would like the Machine to have.

More exactly: When I pgm goals into an Intelligent Machine: It will pursue those goals: Not nec'ssarily the goals I'd like the machine to have.

This is the General problem of Bugs: Meta Bugs: A Bug causes the pgm to not meet/accordw. the specifications: A meta Bug is an error in the specifications.

In the forgoing "Goal" diffys; the error can be a Bug or a Meta Bug. Meta Bugs are often very difficult to discover; they are most often found after pgm. has been applied to problems in R.W. Thus Bugs can be very expensive, sometimes catastrophic.

My Guess is that a Bug in a Goal, is likely to be a Meta Bug. — VERY BAD, very Expensive, potentially Catastrophic.

Marvin's idea: That if a Machine is much smarter than us, we should not be telling it what to do. Reply: A person could be much smarter than me, yet if what he wanted to do was very Bad, I would oppose it. E.g. He might be a homicidal Maniac — He loves to build "Doomsday Machines". Since he is much smarter than me, what I perceive as a "bad action" on his part may not be bad at all — since he understands the action better than I do — on the other hand, his Goals may be quite distant from mine — he may not be concerned w. my goals at all.

6/21/01

ID

See 228 for list of IA's.

CREATIVITY

I scanned from 231 back to 208 — found nothing on "creativity"; 166 is a bit relevant

: Continuation of previous "Recent" note: An Experienced Good Scientist, will have lots of Good hours. They Solve most problems very fast, but ↓ in diversity.

A smart Student (1) has Power hours (2) ∴ more diversity (3) Very often Can spend 24 hrs/day; 7 days/wk working on a problem. Older Scientists do this less (Tho apparently, Newton was able to do it when he was 74 y.o.!)

Poor Scientists have a few Hours! They use Power as far as they can go — Then if problem is not solved, they can do nothing more.

.00: : 229.40: **Another tack**: Instead of using an unordered corpus (as in M1), use a purely Sequencial corpus, but have it be a sequence of Q A's. The advantages are:

(1) I don't know a good "super universal" d.f. on all possl. operators, but I do know a universal measure on strings.

(2) If TM finds a good answer for Q_n+1, it can make the "recognition" of Q_n be simply the pc of the integer, n. (later on, this can be expensive, however.)

Disadvantages: The QA TSQ really is mainly unordered data! The bias is some sequencial info in it.

Advantage (3) If I get a soln to this sequencial QA problem, it is likely to be very useful in solving the unordered QA problem

.11 Thinking about the sequencial induction problem & its usual ALP soln, reminds me of how to do the "super-universal" operator d.f. Formally, the set of the universal operator! — do it like the "Sol 2nd kind of Prob Ind." did for the BAG induction. for Non OSL problems,

Given a input unc. Input ϕ is desc. of operator; Input I is random; Q_i is input to operator, A_i is output of operator.

Input I is what is read in addition to Q_i, to get A_i

We try to find an input ϕ (finite string, always) → $\sum_i |\phi_i|$ (i = input) overall Q_i's that ... $+(|\phi_i| = \min$

We do this for several ϕ_i's for which this sum $+(|\phi_i|)$ is smallest.

In "Lispish" notation! We have a (Lisp) code C_1 for a string S_1 that defines an operator for Q→A

.19 The pc of this code is obtained via **AZ141**. We want S_1 to have small K cost, say, so C_1 should be small.

I wanted C → S to be via "Lisp": or some universal ~~interpreter~~ device!

I want the pc of S. In M1, the PC of S was obtained in a simple "Bernoulli" way via Z141

I want a better PD on S. This PD will be obtained by Observing (1) Context dependence of PC's of sub-functions of F (F(Q)=A) (2) Heuristics that Modify pc of the Sub-functs of FC) — .37

.30

SN In General, whenever a new defn is made, we must also make rules that tell when that defined object has reasonable pc (i.e. scaling problem must usually be addressed attacked (as soonas when'ever) a new defn is Made). — The sometimes i.e. SSZ. wouldn't be large enuf — we could sometimes use OSL, however.

.37: (30) Inline w. .00 - .30: Perhaps 2 essential features of M2 that Make it "Super Universal": (1) TM_1 is able to do problem induction (ENACTM)

(2) we have a TM_2 (which = $TM_{(1)}$),

333.00

.00: 232.40 : Hvr, TM_2 has to be constantly at work in each TM_1 induction problem, if we are to have much use of ① context dependent pc's of abss ⑤ (rng of heuristics. Hvr, even if TM_2 is only invoked periodically, w. OSL, a "recency" dependence of both "pc's of abss" & "search heuristics" (≈ similarity of structure of previous problems) we will get some useful effect from TM_2 in these 2 areas.

Ideally, I'd like to "super universal" p.d. to be implemented w.o. having to "invoke" TM_2 & TM_1 separately: but as a single partial Recursive function.

On the other hand, my observation of my own Prob. Solving, will probably be in the TM_1, TM_2 — 2 level form.

It is certainly possible to do TM_1 & TM_2 simultaneously, when we switch in Time Share Mode.

.11 There seem to be at least 2 ways of thinking about the "Super Universal" p.d.

① The idea of 232.19: a universal d.f. on S, which is the code for all possible functions (& operators)

② The TM_1, TM_2 idea, in which both TM_1 & TM_2 are of the A2.14 form.

.14 (I'm rather "unclear" about ②, hvr.!

Clarification of .11 — .14 would be a most critical problem, allowing the creation of a very simply described Super universal TM! (There are some other "details" — like what part of the corpus to use for a particular problem, given a certain CB.)

.17 SN Marcus' approach to a Reinforcement machine! This could your RTM could be given the same TSQ as a M1 or M2. Since M's approach is Normal — (i.e. complete ALP), it may be possible to use his approach for a "final R.M."

In analysis, using the QATSQ (as for M1 or M2) Hvr, would be much simpler than a real, general reinforcement & hvr.

Since the envt. is much simpler, & we'd need to do No "experimentation"

.24 (No) "experimenting, Physics" [RTM's Response to a Q is a P.D. on A's; its reinforcement is in PC of correct "A". Max $\leq PC_{Accept}$ = max pc of corpus (≡ Max Likelihood Method?)] — Hvr, I suspect that Marcus would want TM to have fast (i.e. "best") responses & Give R=1 for correct, R=0 for incorrect.

(SN) Also think about the Min Backtrack machine! Say it's use "Lisp", & after each successful run, I put all dents at beginning of the code. (like ZA2.141) Then I think the system would be very much like A2.141! Would it differ at all? — in what ways?

.30 If a code does not use one of the dents at all (directly or as a subroutine) then it can be compressed by eliminating it. ! — 234.07

.31 part of another used dent.)

.32 If the "the corpus" consists of both the $[Q_i, A_i]_{i=1/n}$ and the machine as stated in A2.141 to predict it, then I think any (1 level) Universal language can do (w. this:

There are (at least) two Big Problems ① we don't want the Code in the Entire Corpus for the Practical i.f. we do we must do a Produ. w. a finite known CB

② we have the "Dynamic Memory" problem that Marcus is writing about — but the actual cause of "trouble" is our own biases. We are at all times making inadvertent "Experiments". → (238.35 space) 238.35

Woops! We have to subtract out about the total! so it's not exactly "Max Likelihood" → 238.32

234.00

G(23)08
ID

00: 233.40! NB "If all info is in P.D. then Lsrch is ≈ optimum" This Mite be really true,
.01 If "Corpus" includes all of TM's Traces! [No! For close to Optimum, TM must be able to notice what Happens in early traces of an Lsrch, & use this info to tune in the Lsrch: This is not a simple Lsrch based on a fixed ("All info in Ps").

.02 SIN In/re parsing a corpus in (A)Z141, after one has Made a new definition:
Instead of using old Pc's of n-tups, (i.e. P_i) use P_i^γ ($\gamma > 1$). This pushes the code toward hyper pc: The code of least Pc has all P_i to same ("Max entropy"). P_i^γ trys to make Pc's as different as possl. ∴ hyper total pc.

.07: 233.31: Z141 does seem much diffrnt from Min Back Track Coding: In Z141 I automatically sum over/many codes. This is not done in My Most recent version of Min backtrack coding. In Z141, the parallel codes are all strictly equiv!, so there is no. possl. loss by "pooling" Pi. statistics — no (info. losing) "Decision" is ever made (other than the parsing decisions)

Also

Note on **MDL v.s. ALP**: Consider Arithmetic Coding: In ALP, we have an infinite no. of codes for each finite bit strng. We can approximate this, by taking Taking only the first k bit approx of the true (ALP) Pc. Because of the peaking property of $p^x(1-p)^{1-x}$ (i.e. it peaks when $x=p$), This will give an error per bit of an amount ∝ $(p-x)^2$ (The amount will always be >0). So for a long code, of length n it takes the no. of extra bits in the approx, will be ∝ n.

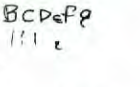
However, the purpose is not always to get a short strng style short code for a long corpus. The best way is to use as by precision as one can for each bit of pc of each bit, then use Arith coding. This would seem to keep error forbidding precision, in order to get a small outl. of bit less in the final code.

.26 Perhaps Rissanen figured out how to deal w this — so arith coding is Practical, & Elias coding is not! This is discussed in IT May 2001 p.1533 [IE³ Kempton]

Vol. 47, no. 4 W.D. Withers

Hver, ibid, (p1533 col 1). he gets $2^{-q}(q+2)$ extra bits per coded symbol if he uses q bits of precision! I should think he'd get ~ 2^{-2q} extra bits per symbol!

He says" ~~(illegible)~~ also says" (last line of col 1) "Various clever makeshifts are known for handling the carry-over problem". It may be that there is a problem in Arith Coding that I haven't really been addressing! ●



.2 mm
= .02 cm
= .02 / 2.54 "
~ .008".

LD

"Sub Corpus problem" .00

.00: : Out Core for induction : If TM is allowed to use different parts of the corpus for induction,

.01 the (apriori) Core is unclear (?). If the problem is finding $\frac{pc(1)}{pc(0)}$ ratio, then presumably corpi for num, & denom. are same. Say we spend same cc on selection of (sub) corpus & finding codes for corpus (numerator cc = denom cc). Code finding starts for Num/denom are "identical" (? - is this possible to determine useful way?)

Anyway, so Post Mark, couldn't we logically determine optimum strategy for corpus selection (denom, numer) for coding.?

It may be that (Semi-computable ("enumerable") functions are definable wrt. CB is given, but "non-canon" (ratios e.g.) are not uniquely defined.

If we use a standard approach, of a Time, CB, it would seem that ratios are well defined.

— But perhaps if one uses different strategies ('grand' for num, denom) that they don't converge to same thing (?) (but limit of ratios ≡ ratio of limits).

229.17 ff. &c
232.37

ABCDE

.14 Actually not so bad!

A possl. BAD way to do .01: say corpus is binary Bern. seq:

We select sub corpus of all 1's — This Gives sharp d.f. ∈, & looks v.g.!

Trouble is, the specification (dcrn) of the sub corpus is very expensive. SO! Is This a big clue to the sub-corpus Selection problem? Consider all ways to derb. sub corpi:

Each such Dcrn has its own wt (≡ pc). We could use all of those sub corpi & the resultant preds (on 0,1). So, in "normal" (cc = ∞) induction, we could also divide up the corpus & make preds using the parts — & wt. those dcrns or the pc of the dcrn. of the part. This would seem to be not the same as Regular ALP

— Tho, if there are sub corpi that are useful for predn, then this is an interesting kind of Regularity that ALP should be using.

.26 .14 - .26 is a novel take off on the "Sub-corpus problem", but perhaps predn in general. Reserve to do it later

If I get new ideas on it.

A rather el. "Soln": Initially, all problems are given several "indexes" by USR. From these, TM is able to tell how to categorize the problem & to some extent what sub corpus to use (No this info is not yet linked w. the CB given for the problems) Later, the USR is less careful about assigning Indices & TM learns to do it itself.

Now this all seems related to the "Scaling" problem: Assigning pc to concs, that depend on context. "context" seems related to the "indices" of problems, Just as the pc of a concept is related to its context: The pc of a prediction is related to the context of that predn. For concepts, we did pooling of info over the context of that conc. for pc's (preds) we have pooling over the "context" of that predn pc. Hence, for preds. the "conc" is the 0,1 pair. The assoc sub corpus depends on the local context of the (0,1) pair. This might be the "QA" — or just the "Q"

237.00

ID

"Epicurian Codes" (.09)

"Updating" & "Summarizing Machines"

Rap = Rapt.

.00 : On "Summarizing Machines" : Perhaps one way of looking at "Updating" is that it produces a kind of "Summarizing Machine". (~~Secum~~ SumM? SumM?)

One way to produce them was outlined in the "2 kinds of Prob. Ind." paper: use a set of k Machine states associated with the k shortest codes found. [My present impression is that this is not a v.g. approx. method: that it usually has to be very large in order to get a reasonably good Pd. .... e.g. for a Bern. seq. w. alphabet containing m symbols, — each w. a distinct pc — having k = m would not express the true PD very well — Hmr, I reached this "Idea" "Conclusion" by considering the ZIP code (which is a kind of "Epicurian Code" that uses lots of 11 codes: I'm not really sure that it's true)]

.09

A "Partial" Summarizing Machine" Code have a (relatively) short corpus c' plus a special Machine, M , so that M's Pd on the new chunk of corpus, x is the Pd induced by Machine M [redacted] on the string c'x. In the "2 kinds" (~1999) paper c' was the null string.

The more General Concept is that of a ~~"PD"~~ "PPD" "Practical PD" (a PD w. cc as an input parameter - The result of the "Updating" is a Usefull PPD — (usually a "PD"... Since the cc is usually small or acceptable or specified in the PD dcrn).

In Z141 & AZ141, There are (3) components of Updating:

a) update freqs of concs, & reparse.

b) Derive (1) new concs, ~~update freqs~~ reparse (3) & update freqs. (3)

If (b)(1) is not possl, then (a) is done.

It may well be that Most methods of updating IA's are close to or analogous to those 3 operations. 228.00... Gives a list of 14 IA's.

Also Note AZ14 can give a simple meaning to "Min Backtrack". — i.e. w.o. Backtrack, we ~~track by simply update pc's in view of new data~~, & do a rather greedy parsing of the new data & update the pc's. Then

Next in order of cc; Find new concs, reparse corpus, ~~update~~ recalc. pc's update

Then may be reparse & recalculate & update pc's again recalc.

.34 Next in order of cc: ~~remove last definition~~ After new data is obtained, remove last definitions,

.35 (a) try to find new definitions, reparse, recalc, pc's; May be ~~[redacted]~~ loop to (a) (.35)

w. more cc available remove after new data is available, last 2 defns, & do like .34-.35.

→ In some cases, it may be wise to remove certain defns that are not the most recent ones

7. Subcorpus problem! Relation to "scaling" problem of context dependence of pc's of concepts ("cones").

.00: (Spec 235.40): So TM looks at the indices of the current Q! All other Q's w. same set of indices are certainly "relevant": Q's w. all but one of the indices the same as the present Q, are next in line (if CB is large enough): Then we next consider Q's w 2 unmatched indices.

The rough rule: Certainly we should consider those certain indices should be more important than others.

.05 Now consider context dependency of conc. pc's: Say we have a Q & we want to construct an operator that maps that Q into its A. The Q will be the "context" & will have a [redacted] associated with it, a p.d. over all the conc. defined & thus far. (In some cases, it may be able to give pc's for cones not yet defined or used. — it does this by extrapolation from the [redacted] stochastic law. assoc w. the cones that have already occurred & their pc's in the [redacted] operators that created the A's. → .16

.10

.11 Any way: 1) the subcorpus problem & the 2) context dep. assignment of pc's to cones (or to other things) should be performed clearly! To what extent is TM "solved" if I solve these problems? Other important probs: 3) correlated pc's of cands. in search. 140.00–141.90 lists 12 items "passed" so 89

.16 (.10) [redacted] On context dependent pc assignment: This assignment should be CB dependent (like subcorpus assignment). If CB is small the [redacted] assignment should be very [redacted] narrow: to a small set of situations/environments. If CB is large, we can afford broader characterization, but it will lower pc's in most likely (Domains environments) in order to get more "diversity"/"creativity", by assigning not such small pc's in fields "prime time" dreams). [So this ties 1) & 2) to (.11) together, more]

On the "similarity" of 1) & 2) (.11): In the subcorpus problem, we are given a problem (Qo), & are to find a CB: Co: we want a subcorpus: set of Q,A pairs in the past. || for context dep. pc of conc! Given a certain conc/observations, X: we want its pc as a function of the context in which it is to be used. This "context" might be a problem (say "Q")

.30 (SN) Some confusion in my mind about "subcorpus" problem: When it occurs in "updating", we only have QA pairs (no isolated Q's): In one version of QA, when an isolated Q occurs, we always have available to us an "updated stock operator" that can map this Q into a d.f. on A's. [However, it only has a certain CB, Co, & that IA may be inappropriate (?).]

.34 In an Alternative version: TM tries to present stock operator on the Q — then decides that the d.f. isn't "good enough" ("sharp enough"), so it does more "updating" w. what it feels is the "relevant subcorpus"). But normally, the "subcorpus" is needed only for updating a set of QA pairs. In .34, however, we have an isolated Q & TM finds an appropriate (small(?)) set of appropriate QA's to update — which gives another subcorpus of QA's for inclusion in the updating process. This amounts to "backtracking" by subcorpus "Q"

So TM must be able to find a relevant associated subcorpus for either a single Q or a set of QA's (Spec 240.37)

ID

Outline of Letter to Marcus (cc. to Juergen): "RANT"

1) How Q.A. machines able to solve any problem that I'd/would be interested in giving it.

2) I am interested in a minimal machine that can be arbitrarily intelligent.

3) Idiot ID is a machine of this sort. It does not explicitly have a "learning" problem, since as is Q.A. is sharply delimited. When such a machine becomes arbitrarily intelligent, I will [illegible] Dynamic Prog/memory problems that I am unable to solve. (I am able to define these problems, but not necessarily solve them).

4) One of the attributes of the AI system that has no adjustable parameters is that it is more likely to be considered "intelligent" by critics of the "AI concept". I am very much not interested in proving that Machines may be "intelligent". When they are very usefully intelligent, that fact will be quite apparent. (and for Doctrinaire there will be many people who will be unconvinced — The reasons are not crucially related to AI critiques).

5) It's my impression that Reinforcement Learning is "Large-Large" problem. If you don't succeed, you (most) of you do succeed, there is great danger of Machine manipulating Real Estate. While many people will accept this "danger", I will not. When an advanced Q.A. machine is given a Difficult Dynamic Prog problem (or simply a very difficult problem, such as interaction w. R.W. is a Need of. Use of many manipulators This is ridiculous, I think it is unnecessary to restrict Prog in a general reinforcement machine.

If any intelligent machine is given "Large" problems, then

.31 6) See ID 233.17-.24 & (237.17-.32) R For comparison of RTM & QATM.

.32: (R 33.32R) Hrr. One could regard QATM as MAX Likelihood, i.e. the pc of the model (sponsor) is included. One trouble is that the single best (model, corpus code) would be selected in Max Likelihood — or as By RTM. I think QATM (would/could/should) accept several of the best models, not only the "Best", & perhaps could try to have Max diversity over the models.

.35: 233.40 On "No experimentation" in QATM! Actually, QATM does do "experimenting" & it normally has "Dynamic Prog" problems. These occur in ordering trials on Grids, (Non-Greedy ordering of trials is a Dyn. Prog. problem) Hrr, in early training of QATM, I will use very simple approxmns to the solns.... — in the hope of eventually getting to pt. at which I could give it "Dyn Prog" problem to solve. TM. (essentially — it becomes part of "SI"). So, to start, TM's order of trials will be completely or "fairly" Greedy.

233.40
.00: 238.40 : Also Note: The Dynamic Pemy. Problem Marcus is working on: Including CB (as walls, fixed horizon) is extremely diffl. To do anything like optimization wrt. ~~CB~~ ΣR & fixed horizon & fixed CB (that includes all of t. possl. paths), seems really formidable. (At first Glance, I Post it was t. Unsolved G(x)·T(x) OZ problem — but it is apparently not.)
I don't really know just how far Marcus has Gotten w. this problem. He says its within a factor of 2 wrt of optimum, but I'm not sure I know what "it" is.
.06 If he did get something like an Lsrch soln, this could be a big step toward a usable "Heuristic" TSP "Soln.

.07 REV 232.11 -.40 ; 233.00-.40 ; 234.00-.01 ; 238.31 -.40 ; 239.00-.06 is timeline line in developing QATM:
Other recent topics: 232.00 Using a Sequencial (rather than a BAG) corpus for QATM
234.02 : use of p_i^δ for pc's of comes (renormalize)
234.07 MDL v.s. ALP : Peter Elias Code v.s. Algth. code ← (234.26)
235.00-.40 } Sub. Corpus Problem.
237.00-.01 }
236.01-.40 Up Dating : "Summarizing Machines"
238.00 - .40 Differences betw. Marcus/Jeroen's RTS & my QATM! Advantages/disadvantages!

Back to t. Line of (.07): Say I had M1 : a QATM & MTM: ≡AZ141; "Search TM"
Its serious darkeys: 1) No NMTM problems 2) Inability to derive most heurs (since they are NMTM probs)
.18 3) pc's assigned to ~~comes~~ concs are not context dependent — so serious scaling problem.
.19 4) Perhaps inability to work on "sub corpus problem" (may be related to (2) : see 235.00-.40, 237.00-.10)

If we allow M1 to work on NMTM probs., we may get a large jump in its capabilities — & ability to deal w. other 3 diffys.

In M2 we got off "Gloss over" t. problem of NM induction, by saying "There are many soln. methods — each w. its own updating Scheme".
To start off, we have a set of IA's: for each QA, we use all of them! After we have enough data, we give TM t. (meta) problem of ~~making a P.D. norm~~ being a given a problem & finding a P.d. over IA's that are most likely to be t. "best" soln. of t. problem. ("best pc of soln" in given CB),
→ TM now uses this P.D. in Lsrch to find good P.P's for QA probs, w. given CB. (I think mobility to do something like this, was a deficiency of M1
TM also can work on (3) (.18): see 235.01-.40 & 237.00-.01 for Gen. discn See 237.05-.10 for how this might be done (This discn is a bit vague, hvr).
(19)
TM can also work on sub corpus problem 235.01-.40 & 237.00-.01 237.00-.04 in particular

TM can also find Heurs (Lessons) since this is a NMTM problem! How TM can apply these heurs to t. solns of probs is unclear! One approach way is to work on ~~each~~ many QA probs & finding heurs in parallel! Heurs are almost always expressable as Mobility of t. P.d. for Lsrch, so we will be modifying this P.D. during t. Lsrch (which makes it a kind of Modified "Lsrch") (or a search method inspired by Lsrch), (i.e. one that didn't worry about "convergence")

A main diffy of M2 is that when it uses Lsrch, t. trials can be highly correlated, which leads to many non-optimum results. A big Q is: would such a TM be "Good enough" for serious "Bootstrapping".
A similar Q is: Could a TM w.o. facilities for OSL ever do serious "Bootstrapping"?

IP

Sub Corpus Problem .10 (A new idea!)

7 to 11

.00 : 239.40 : So, in General, what "difficult-to-realize" features of TM could I ommit & still have it "Bootstrapable" (i.e. able to work on SI).

Well, at a minimum I should think it would have to be an NMTM.

So: How it works: It's a QATM: (NMTM)! In steady state:

F( ) [illegible] There seem to be 2 "f( )" functs: f?( ) is

for updating (using a new subcorpus of QA's plus a CB): f'( ) has a Q as input: It decides on what IA to use on that Q to get d.f. on A's.

CIAP's:
1) CIAP
2) MPP.
(wer is part of was to)
3) It [illegible] I recall to want what other than it to want!

In an earlier version of TM, I only had one GPD (Grand Prob. d.f.): How did it work?

.10 SN For inductive inference using a finite corpus & finite CB, Co, T. best strategy is not to

.11 try for the Best code for the entire corpus w. Prin. & CB. Co One approxim. is to just use part of the corpus for predn. — the size & nature of the subcorpus, being a (funct. from) F( )

.13 Co & the things to be predicted. It would seem that this is an el. soln. — that there should be a more general way to deal w. the problem. T. & el. soln is:

"Select sub corpus (as a funct of problem & i.e. Co) then do Lsrch on that subcorpus."

.16 A more General Soln: (") select an "induction scheme" — a "IA". —

This IA having a sub-corpus selection scheme built into it (if need). — Instead of chosing a single sub corpus, it can look at any part of the corpus it likes & spend as much time as it feels appropriate on those parts.

In the past, when I spoke of an IA, I had it as becoming of a fixed, known corpus. A more general IA, would look at the problem & CB, a corpus & decide [illegible] how much of the corpus to code, or if it might decide on some

.27 other method of (induction prob.-solving).

The sub corpus elem. was an outgrowth of my (el.) approximate factorization of T. QA problems into [(1) F( ) which selects the subcorpus for the IA which works out the problem w. CB = Co.]

& CB, Co [illegible]

.31 It may be better to have f( ) select the IA & have the IA do select/not the subcorpus (or

.32 whatever other equivit. "techniques" is Necy]

So: T. problem of .10 ff has (at least) 2 possl. ways to solve it! see 241.09 241.15 in particular

.36 (.11-.13) or (.31-.32). T. non-el. soln being "factored" fm. (.16-.27)

So we will choose one in our early Models of TM.

.37 : 237.40 : T. problem of .10-.36 is related to the problem of 237.30-.40.

In 237.30-.40 I had of prediction & updating as being separate problems.

I guess the difference is: Updating is done whenever answers are available. Prediction can be done when no answer is known

Hm. The solns to problems are used in updating: Say TM has for Q: [illegible] obtained

"Greedy r.s. Far Sighted."

.00 :240.40 : solns $[A_i^k, pc_i^n]$ k=1,.. Say A_i^5 is t. correct soln. The A_i^5, pc_i^5 info will be used in updating: Also t. techniques used to find this soln. will be given a "special treatment" in t. updating process.

.03 Think of Z141 induction: T. "Q" is t. latest "chunk" of a time series. (say). In this case, we would have to update before doing a prodn: T./TS consists A's as well as Q's: well, say we have t updated TSQ already & we have to make a prodn: We use t. Primitives, deals & pc's of t. last update, to get a pd. for t. next symbol. [Here we assume no "OSL"]. When we find out what t. correct next symbol is, we can update t. IA by changing pc's, defining new symbols, & reparsing. (Note Hyr.)

.09 .03 brings up a problem I've not worked on! That a given IA could have several subcorpi that it works on, & each will have its own update parameters.

.11 The subcorpi will also have some "shared" parameters — As in t. STEIN effect.

So F( ) should be choosing a (IA, subcorpus) pair. Each such pair has it's own update params; (+ some "shared" params (.11)). Just what t. "shared" params are updated/inserted/discovered; is unclear. So "Grand"/"Macro" update orgy → Or, when one has a very large CB for updating

.15 This idea is Plan Tiadem w.r.t. discard 240.10-.40 : It suggests that 240.11-.13 may be t. best "Elan". (rather than 240.31-.32)

Note, here, that in t. Fong "time Series" problem, t. parts of t. subcorpus were all clearly labeled ("indexed") as belonging to t. same Time Series.

A (subcorpus, IA) pair that has been updated is like a "Summarizing Machine" — So for a problem (Q); FC) selects (1 or more) "Summarizing Machines" to solve it.

.21 I have been (Most recently) mainly thinking about Z141 & AZ141-type induction problems. Clearly there are kinds of problems that look much different! Eg. solving/equs, n.l. Integration, (A long sequence of operations that solves t.(INV) problem.). (Search TSQ where short sequences of operations that solve t. problem. Q is: Are these much d.ferent from t. ones I've been thinking about?

Also, there is a general class of problems solvable by "NON" techniques (AND/OR nets of subtasks).

In AZ141, t. problems are all solved by Modified Lisp such over combinations of a select symbols in which t. symbol to use is "context dependent". T. Q is: can all problem solns. be expressed in this form? If not, can this system do SI & find a more general system that can solve "all" problems — or can this system recursively reach an arbitrary level of prob. solving skill? Or can any reasonable heuristic (prob solving) be learned by this system?

My work about this system has mainly been w.r.t Z141 & AZ141 type MTM problems (.21!) M2 has to do NMTM problems. But AZ141 has 2 levels! It is a P.D. on t. set of all functions. Time is this stochastic level; & there is t. "set of all functs" — which is MTM.

On "reparsing" restarting t. Lect, or "it has to" phrases may not be adequate!

Try to find "Creator Cancer" thing before — 4/1/01 before I left for Europe → Has seen 243.00ff

ID

Def

AZ1

.00: 241.40 : 1) AZ141 is a complete soln to MTM problem, since it is a wtd def over → 245.18 !
all (deterministic) operators: So we can solve any MTM prob. w. Lsrch

.03 2) Since dMTM a NMTM have same soln. (in ALP) this may solve NMTM problem !

3) For NMTM, I had the idea [to finite CB,] RLP was not the soln. (b) But for finite CB, Selecting part of corpus, then using RLP "like work" — but (1) it was not clear how to do/say corpus selection (2) w. the proposed method, of result-corpus selection, it was not clear that this was optimum.

4) (1) & (2) can be for CB = ∞, so even if (1)(2) "worked" it solved NMTM for CB=∞, It would not nearly be a soln for CB < ∞.

5) Re (1,2) algm: It should be possl. to develop a complete theory/derv of Universal stochastic operators (analogous to Universal Distribns) using AZ141

SN AZ141 differs from LISP, in that all definitions occur in the precorpus. — Other than that, I just use a "minimal" Lisp, w.o. its specific "definitional" facilities. [i.e. no "λ" notation]

6) There remains the possy. that "sum over short codes" is not a very practical way to get P.D.'s So, even if I found a way to get AZ141 to give a universal PD on stack operators, it would not be a V.G. way to do this! Still, it might suggest ways to do it.

.17 One (ALP) way to look at "the Universal Operator"
We have a (UTO / UTM) w. 2 inputs, 1 output: Input 1 is "Q" (input to operator); output is output of operator
Input 2 is a random signal. so M(Q, R) = A.
with a given Q, the probty that the output will be A is STOP (A is Q's two finite strings) is defined
.21 to be the probty of A for this model.

.22 An equivalent model: [to .17-.21] 3 inputs to Machine: Q, O_i, R; output is A.
Q, R, O_i are finite strings; R is random string. O_i is a stack operation/term.
The rel. pr of A being printed from Machine stops is the rel pr of Q → A for operator O_i.
In .17-.21 where sum over all O_i we write $2^{-|O_i|}$.
The length of the Random input that creates A is then effectively added to $|O_i|$ to give the
.27 rel. assoc w. that random input.

.2 Can [.17-.27 (a especially .22-.27) ideas] be used to get a stack operator & a universal operator, out of AZ141? Using AZ141 notation, we can have a 2 input machine: One input is Q, other is O_i: but O_i is a MTM operator. If we put random input into O_i we get a stochastic operator (probably a universal stack operator). ← If so, we could divide the random part into R and O_i, we'd have a stack operator "defined" by O_i.

.37 Or AZ141 could have 3 inputs: denoted O_i, Q, and R. R could be regarded as "an extra part of"
.38 O_i or of Q; or just a separate input.

In .22-.27, If R=Λ, then the output is the most primary A: Various values of R can be regarded as derived derivations from this minimum. — (This is "MDL to BA"!) — Not so! If we have sum over many R's we 244.00

7/5/01

ID

CURE CANCER Problem

CURE Cancer problem

.00: 242.40 : I haven't been able to find my previous.. .. previous soln (< 4/11/01) in Meeneeses:

A new try! i is time? index; X_i is TM action, y_i is corresponding response observed.

$[X_i, y_i]$ $i = 1/n$ is a possl. sequence of ~~stimulations~~ interactions. given $[x_j, y_j]_{j=1}^{i-1}$ $i \le i$

.03 Alp gives us a pd for each y_i ~~for all such interact~~ for all such interaction Sequences! (see .12-.13)

For $n = N$, we have assoc w. each sequence an R (Reenee) value.

We want to make choices so as to maximize Expected Value of R.

.06 Given history up to $i = N-1$, we can always choose x_N to optimize ~~gen~~ E(R).

.07 " " " " $i = N-2$ " x_{N-1} " "

.08 " " $i = N-3$ " x_{N-2} " each.

.09 $i = 0$ (no history) x_1 "

So, There may be a soln for fixed horizon = N. It seems Much simpler Mynmy Previous Soln! — Is it correct?

.12 In .03, ~~we~~ all we need from Alp is: given $[x_i, y_i]$ $i = 1/n$ and x_{i+1}; what is

.13 the p.d. for y_{n+1}?

Say $N = 100$ & each x_i can have 10 values only, and y_i can have 20 values only.

~~Here are~~ There are $10^{100} \cdot 20^{100} = (10 \cdot 20)^{100} = 200^{100}$ x_i, y_i strings up to length 100

In .06, we ~~eliminate~~ divide the no. of possys by $10 \times 20 = 200$

.07 " " " " " " 200 each

.08 " " " " 200 " " each.

In line .09 there is only 1 possy left!

ID

243 "on Cancer"

.00: 242.40 : don't get as much rel. wk for A: In fact, A may not derb "most likely" A.

If mite be well to study 243.22-.27 & AZ141 smarty, to get good ideas on how to use calls

.02 Express Stochastic Operators.

A nice Remy about AZ141: It gives a good pc for MTM operators: This is because the Operator terms can only have 1 way; & other summations of alternatives codes are automatically summed by AZ141 convention.

There will, in general, be an oo of way to derb. an operator in AZ141 & any 2 poor formalisms, but the equivalency of the codes will be hard to detect/verify.

.07 Some ways to derb Stoch operators: 1) Numeric Operators: → Numeric

2) Give center vector output & [redacted] Correla. matrix for derivations.

b) Give output as funct of a set of params & give standard d.f.'s for params.

c) For vector input, derb fuzzy/vault density in output space.

Speci 250.09 ff & 253.37 for 2 good methods of deriving stoch ops: Dis discussion cut to 269.40 is very jumpy, hmr! 250.09 was not N.G. - that later shown ocks.

.11 2) String → string operators: So one fairly general way: a stoch lang. is a pd on all strings

.12 A pd on stoch langs, [redacted] text is a function of a string, is then a general Stoch Operator. ← (O.K.)

Togal: stoch OP spring: → strings: ~~string~~ string (input) string, (grammar); string (data) → corpus of stoch formal langs.

Stoch langs can be derb'd by stoch Grammars, stoch ~~Grammars~~ can be derb'd by stoch Grammers.

So [redacted] We start w. simple stoch Grammar (= Promiscuous Grammar, or AZ.1f Grammar) ← use it to generate stoch Grammars.

Actually .11 ff is not exactly rite: ~~A~~ T. initial stoch lang. generates a pd.

on MTM Grammars: These can be converted to NMTM stoch Grammars by giving pc's for each grammar rule — parametrizing Rem: So we have a determ. Grammar & params → stoch Grammar.

This Conversion of det Gram. to stoch Gramm. is easy for Grammars resp ~~resp~~!

.18 rewrite rules: For other kinds of Grammars, it may not be so easy.

∃ 3 ways to Generate a stoch operator!

Q parameters a (stoch data) lang & via a (stoch det) lang. | ie Q → lang Operator dom. → lang, A

T. 3 possibs is lang, & lang2 are both stoch

.26 2) only 1 of lang1 or lang2 is stoch.

In 242.22-.27 ~~it~~ it's not clear as to which of the 3 it is: In 242.37-.38 it is lang, or lang2 Rem is stoch.

.28 A way to Reach above Stoch Ops: 1) Q generates a/single stoch ~~Grammar~~ Grammar2

T. relation of Q to Grammar2 is (deterministic or stochastic) If Grammar2 is deterministic

α must = stochastic. If Grammar2 is stoch, α can be ▷ or ≤. If Grammar2 is det,

if α = det, then we have a det. Operator. MTM.

So, in .28 Q stochastically chooses a data or stoch lang

.31 or Q dotly " a stoch lang. → Combo dom if Q dotly chooses { string (Grammar) ← 2 vectors of reals to supply w. probabilities so it's a stoch

AZ141 is a stoch lang — on def. operators. It can be made into a stoch lang on strings — since Grammar Size(.02-.26)R

Lmp ≡ UMC.

.34 Q: In .28-.31 can the models having only one stoch lang derb. universal ▷ operators ≡ i.e. can they

.35 give a universal d.f. on stoch. operators? → 244.09 - 245.09

For each fixed definition string in AZ141; we can have only rules/rules follow, ~~that gives~~ This gives a specific PD over all (operators / funcls).

Def .38 Also each operator defined in AZ1 (≤ AZ141) can be defined by certain kinds of strings: A universal d.f. on discrete strings, would give a universal d.f. on functions, but not a Universal d.f. on "def's on functions", ie.

245.00

See 245.09-.17 for Great Break thru: (I'm not so sure, hmr!)

ID

Very likely Theorem: Great Breakthrough: Universal P.d. on all Stochastic Operators

↳ But See (20) for DOUBTS! SEE 253.37 for how to do it right! — see 253.21 for counter example to .09-.17

.00: 244.40: We want a universal d.f. of (?) all finitely describable d.f.'s on functions.

So: A definition string in AZ1, defines a P.d. on functs, [or it can be recorded as a description of a single funct.] Any Universal discrete d.f. on those defining strings would be a universal d.f. on all functions.

How could AZ1 give a (discrete) Universal D.f. on strings? How does LISP give a Univ. D.F. on Strings? Well, LISP is a univ. Functional Lang. — So if we make the input constant (say A or $ or !) we get a universal output d.f.

So, it is natural to ask LISP to say Univ. d.f. on functions: [Initial string defines function as output.

This defined function is then the function of interest — But it is a def. function.!

.09 [Q (~ to 244.34-39) can a universal pd on def. functs be equivalent to a Universal pd on all {D.P. on functs}? (A D.F. on functs ≡ Stoch Operator).

.11 The answer to .09 would seem to be Yes!: Because any pd on a pd on functs is a pd on functs. This seems like at least a great idea, I may have to be careful when I have continuous parameters, since dcrn langs → ∞.

→ Is this to say that Universal P.D. on def. functs ≡ Universal pd on Stoch functs ??

Try to prove it using the "constant factor" argt.

Well, any p.d. on stoch functs is a p.d. on def. functs, & if one is describable w. a certain

.17 no. of bits — so is the other — by same no. of bits. So Prob .11 is TRUE!

.18 Now, note 242.00 (to .03)! If AZ1 is a universal d.f. on def. functs, is it also a Univ. d.f. on Stoch. functs? → Even if it is Univ., is it Any Good?! ↳ 242.03 suggests "YES"!!

Concrete Critics: See .31-.32

.20 Some Doubts on .09-.17: Say $F_i(x)$ is a complete set of functs of x (a string)

$G^j(i)$ is a P.D. on i — so this amounts to a p.d. on $f_i(x)$.

T. functions G^j have d.crns, so one can have a Universal P.d. on them: P_j.

$\sum_j P_j G^j(i) F_i(x)$ is then a p.d. on $f_i(x)$: f_i has wt $\sum_j P_j G^j(i)$

Still, do I ever need to know $G^j(\cdot)$ & give wt. to it because it's imp.? — Yes!

$\sum_i G^j(i) f_i(x)$ is a stoch operator on X: It could be a soln to a Q&A problem.

T. Soln. to a TSP of MTM QA probs will be usually much easier than for f. corresp. NMTM problem! This is because after solving a MTM problem, one usually only saves 1 (or very few) Solns, while in NMTM it is essential to save many. Note here, that in a complete MTM, we must have NMTM facilities so these may be derived & implemented.

Docs :31 Note: for NMTM, one only needs 1 function to solve all problems. For NMTM, I imagine

-32 that one would need an Enormous no. of (n+d) functs to give reasonable p.d.'s for most NMTM probs

.33 (M) means deterministic (NM) means stochastic. M probs v.s. NM probs; M functs v.s. NM functs → 247.00

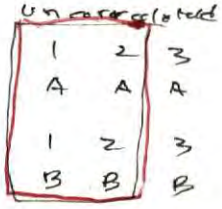
So, the Moral may be that while ALP is fine for Theoretical understanding, The problems of approximating it w. finite CB, normally involve things that don't look much like approximations to the sum over all codes in ALP!

(246.00 is on corrolm.) → 247.00

Correlation

.00: : SCALING2: A possibly new area where scaling is impt: "Correlations". Th. reasoning: Say we are searching th. pc space of k product of k cons.. Each one is self correlated so that every other trial is a bit th. same. So redundancy = factor of 2. If we search over k=2;

.04



4 combinations → 16 trials.

4 combinations → 4 trials.

So if r is redundancy of 1 conc, then r^k is redundancy of k concs.

"Counter Argt." As we ~~observation~~ go along in th. TSP, k will be kept constant. However, we would have lots of trouble as k ↑. This would limit our searches to small k, so a pathological degree. The srchs are already limited to small k for reasonable CB — but we would get an extra bias toward small k via this "correln" effect.

T. way th. "correln" works In (.04L) th. pc's of each trial over $\frac{1}{2} \cdot \frac{1}{2}$ as large as th. corresponding trials in (.04R). Ideally in (.04L) we'd like to "pool" all 4 of th. A trials, so their total pc would ↑ by 4, but we have no way to do this — {we have no way to recognize that they are ≈ identical & that their pc's should be added & that they should be processed as a single trial.

So Essentially, Lcost of solns would be mult by r^k. ∴ strong bias toward small k. — In addition, th. exponent bias toward small k.

ID

.00: (Spec 245.33): That I'd need a very large no. of (wtd) func. solns for NM problems. Unclear as to why this should be so! Maybe IO would be enuf.

.02 How would AZ1 do NM probs? Say t. trng. data is $X_1 \to Y_1$; $X_2 \to Y_2$; $X_1 \to Y_3$ $(Y_3 \neq Y_1)$

$F_1(X_1) \to Y_1$; $F_2(X_2) \to Y_2$; $F_3(X_1) \to Y_3$: We'd like F_2 to $= F_1$ or F_3.

Each F_i has its wt., P_i. If F_2 is distinct from F_1 & F_3, then use as "soln" t. $\frac{P_3}{\text{wtd}}$ sum of t. F_i.

The 3 preds are of pc's $\frac{P_i}{\sum P_i}$ so total pc $= \frac{\prod_{i=1}^{3} P_i}{(\sum P_i)^3}$. I guess t. total pc of t. model is $\prod_{i=1}^{3} P_i$. If some of t. F_i are t. same, then $\prod_i P_i$ can be much ↑.

SN If 245.09-.17 is really true (& useful), Then I will want to review many previous ideas, concs, problems, in view of this CHANGE in All-over-view.

Do a $CB=\infty$ analysis of .02 ff to get a feel for how t. soln should look.

For any $\{Q_i\}$, if no X_i has > 1 Y_i, then $\exists$ a single $F_i(\,)$ that will do it. — (tho it may be of very low pc — say the Y_i are "random").

So, t. induction will consist of all of t. funcs that map 1 or more of the X_i into its corresponding Y_i.

because we will then decrease codes for t. corpus from those functions.

7/7/01 Superficially: Say we have a seq. of input to ops: $[Q_i]$ $i=1/n$. deterministic

We have a "true" enumeration of t. A_i's: Which is a (pc wtd) Σ of various M operators.

.17 $\sum_i a_i F_i(\,)$. $\sum a_i = 1$. I.e. pc of t. seq. of A's is

$$\prod_i \left(\sum_i a_i F_i(Q_i) \right)$$

Consider an op. F_i: Count how many times it has been "right" in t. sense of getting correct A. Compare t. Universal D.F. to t. "true" D.F. in ops. in this matter.

My impression is that whether t. "learned" operator d.f. tends to learn the wt. a_i ($\sim 1/7$) but t. a_i mult. by t. fraction of t. Y_i that are kinds of QE's (whatever that means)!

Mainly, that t. trng result is dependent not only on a_i's but on t. seq. of Q's.

Rite now, I'm thinking of the TSP as being a kind of "stationary", "steady state" condition. — in which TM has to learn t. a_i's. Actually, this is not t. model at all! The QQ's are arranged so as to introduce a ops of increasing complexity in a kind of heirarchical way: so recent-ops are constructed of successful older obs.

.33 Hvr. The Convergence Thm of S78T3 is for not nearly stationary TSP's. — & I was aiming toward that kind of thm. for operators. What I really want is a correspondence betw. t. S78T3 mechanism, & Universal Operators.

One trouble is that S78T3 is oriented toward induction that includes t. Q's; it finds $P(\text{QQ}, A)$ not $P(A|Q)$. But $P(A|\phi) = \frac{P(A,Q)}{P(Q)}$.

Trouble is, $[Q_i]$ is not really a "statistical object" — At least, at least, it is designed to Train TM.

.00 : One of the motivations of 244.07ff, was that I had no really good ideas on how to make stochastic operators for string→string CA's. One of the (promising) ideas was 244.12. Input string Q → ; F(Q) = Q' : Q' (plus a set of continuous params,)

.03 defines a stock Grammar that Greg & Pat agree on.

A2141 gives a D.F. on Functions. This is different from .00–.05 & from 244.11–.18

It is a stock operator.

.06 [Another approach: "Updating" consists of creating A from Q as "cheaply" as poss'l. "cheap" being defined by the cost of sub operations summed over their use count past. Not BAD!]

.06 sounds close to what one often does in solving induction problems,

.09 → .06 sounds like "M" induction, hm. It could be (stochastic) NM if we found >1 way of creating the A.

But I'd like a more continuous way to do NM induction.

But .09 actually doesn't sound bad either! It's like when there are 2 or 3 poss'l. "causes" for an effect, & one makes a bridge from Q to A in various ways — each w. its own pc (& aprip).

Sounds very reasonable for "string" induction, in which there may be several poss'l. Discrete "causes" (= "paths of causality").

7/8/01

.16 How to use .06 to deal w. multiple (different) (comprimed) A's for some Q? At all times (for all Q's) all A's are poss'l. (i.e. have pc>1), because there are ops. that give those A's from the given Q —

.18 Hm, if one Op has been successful a/ most all of the time, for a large corpus, & that Op has hy aprip, then it will get almost all of the wt.

Say we have this "fairly successful op" $F_7(\ )$. If there is a cheap modification of $f_7(\ )$, then it might get some not negligible wt. I do want to work out details of how this comes about!

For each QA pair, consider all of the ops that are poss'l. w. that pair. Each op has its own aprip; so this alone could give a kind of induction — i.e. epsip's. But in addition those ops are structurally related, so that increasing epsip of one will increase aprip (& ∴ epsip) of many others. Again, I have to work out details of just how this works.

.28 Poss'l. Note of Impfc: Perhaps many of the Ops have no output for most inputs! They are specialized to recognize & process certain kinds of Q's. [like "3state" (open collector) output of Logical modules]

Consider the vector of wts of all ops. Say each wt. is ∝ no. of times that that op has been correct.

.32 Alternatively, Consider Ops of type .28, (w. reduced SSZ) — the fraction of times they were correct r.s. "relevant" (small SSZ because they are not always "relevant").

So: we have those 2 ops: One has been right 80 times out of 100; the other 19 times out of 20. Given the aprips of these 2 ops, one could combine them or say I was right 3 times out of 3.

There is another aspect of the foregoing: We can do an analysis assuming we have all operators available. Or, realistically, at any particular time, we have only constructed a few of the ∞ of ops, & we only know the pc's of those few (& to some extent) their (pc) interrelations. 249.00

ID

n:\kwl\voices.aft.com.

.00 : 248.40 : What I'm really (aching,) is a good ALP-type understanding of how this operator System (could / should) work. Just how is t. corpus derbd, & what's its pc?

Well, we have this Huge stochastic Operator: It gives a pc to each Q_i-A_i that occurs.

— So how do we use this for predn.? = How do we "update" t. System?

.04 An ALP-type treatment: Consider wtd sum of all stochastic operators. Its pc for t. corpus will be at least t. pc of t. stock op that is "correct" & (& a stock op that could have created t. corpus, given t. $[Q_i]$ s — multiplied by t. wt. of that stock op in this sum).

So this will give / $S78T3$: It gives & $S78T2$ (t. "constant factor" part) — From which $S78T3$ can be derived. .08 (aversion of)

Hvr, t. set of all stock ops is a very large Hilbert space whose vectors represent t. stock ops by giving t. rel. wts of each of t. functs that are t. "basis": $\sum p_i \leq 1$ where t. p_i are t. coeffs for a particular stock op.

→ From a practical pt. of view, what we want to do, is find stock ops that have hy pc for t. corpus

— This would give a kind of approxn of .04 – .08.

Hvr, for .04–.08, we need a "universal" d.f. on all of these pts. in Hilbert space. The region of H space is "finite" (in some sense) because $\sum p_i \leq 1$. Even so, any useful density funct on that space would assign (probably) a too small wt. to each pt.

(Actually since t. p_i are continuous, we are concerned w. "volumes" of h. space — this h volume being a ↓ funct of corpus size ($\geq SSZ$)

Probably t. no. of dimensions of t. "Hilbert space" will be an ↓ funct of t. no. of examples in t. corpus.

But anyway I need some kind of cutoff on the set of all stock ops, so that I can & compare d.f.t. "kind stock ops" in how good they fit t. corpus.

The $\sum p_i = 1$ condition would give a H space w. "$\infty - 1$" dimensions.

.23 A21 gives a pd on t. M functs — from a kind of Info content & cost (= "cost of coding")

Could we give a "coding cost for t. corpus" & try to give wts (= pc's) to t. various possl. stock ops?

Well, to code k wts of t. set of all M functs! We need/ to know t. precision of t. wts & t. pc's of t. corresponding M functs.

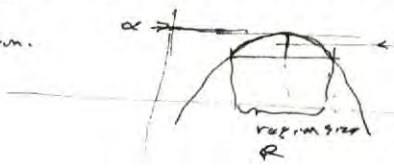
for $k=3$: we have $f_1(), f_2(), f_3()$ & wts p_i ($i \leq 3$) — These sum to t. pc's of t. $\leq f_i$ — These p_i's can be "closely coupled" so if we code $f_1()$ then $f_2()$ is easier to code, etc.

So for a cheap set of M functs, it's best that they be highly related to & t. pc of coding them all.

For t. stock op., we need wts. For $k=k$, these wts will be $\sim \frac{1}{k}$ & they must sum to 1, so they have $k-1$ "deg. of freedom"

In this 2 dim space each pt. assigns a pc to t. entire corpus, so we want t. region of max pc corpus. The unused pc of the set of wts = $\frac{\text{total possl. hvol. of space}}{\text{h. vol ultimately hy pc of corpus}}$.

1 dim.



choose R ⇒ $\frac{\text{mean pc of corpus}}{R} = \max$

for t. 1 dim case, mean p = $\alpha - \beta \cdot \int_0^{\frac{R}{2}} x^2 dx$

$$\frac{\int_0^{\frac{R}{2}} (\alpha - \beta r^2) dr}{\frac{R}{2} \cdot R} = \frac{2\left(r\alpha - \beta \frac{r^3}{3}\right)\Big|_0^{\frac{R}{2}}}{R^2} = \frac{2\left(\frac{R}{2}\alpha - \beta \frac{R^3}{3 \cdot 8}\right)}{R^2}$$

7/8/01

250

IP

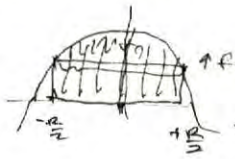
$\frac{1}{2R} - \frac{BR}{24} = \min$ $\quad \frac{1}{R} - \frac{BR}{12} = \min.$ $\quad -\frac{1}{R^2} - \frac{B}{12} = 0$ $\quad R^2 = -\frac{B}{12}$!

P225 uses hyperbolic Start of Mikrou005

.00: 249.40 ! This looks wierd ! : no; we mult by R to get p.: so max $\frac{R}{2} - B\frac{R^3}{24}$

$\frac{d}{dR}\left(R - B\frac{R^3}{12}\right) \to 1 - \frac{3B}{12}$

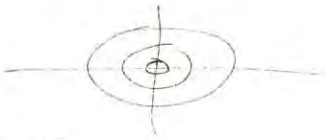
This begins to look all wrong !



Mean value $= \frac{1}{R}\int_{-\frac{R}{2}}^{+\frac{R}{2}} f(r)\,dr$

mean value $\times R = \int_{-\frac{R}{2}}^{+\frac{R}{2}} f(r)\,dr$

Maybe better in 2 dims! mean value of p_c



$\frac{1}{\pi R^2}\iint f(r)\,dr^2$ mult this by # cost of calls to πR^2.

$r < \frac{R}{2}$ say $r \le R$

So max is $R =$ "∞" or total S.

.09 If true, what this means is that we simply take the S over all coin space of the pc of the corpus.

.10 This is mult by the product of the pc's of the M functs.

We want (sets) of M functs so that the product of .09, .10 is max.

.12 7. forgone for eg. may not be correct, but somehow like it is correct. T. Forg. has a simple form we can probably easily find the h volume ! Its some kind of integral like

$$\sum\sum\sum_{\substack{\alpha,\beta,\gamma \\ 0<\alpha,\beta,\gamma<1 \\ \alpha+\beta+\gamma=1}} p_1^{\alpha} p_2^{\beta} p_3^{\gamma}$$

I think I once worked out the general case! It's a rather simple form w. factorials. for just p_1 & p_2 the results are well known.

.15 Actually, I think the result I've been using is $\frac{2}{141}$: (It results in a "sonzen of Lap's rule")

So, this may be an adequate soln to (part of) the stoch op. problem! — Also, I may be able to use $\frac{2}{141}$ – (the coding trick. — may be able to code the QA corpus.

.19 Yes! This seems V.G. ! → see 267.00 for "demo" of why this may be correct.

I will be referring to 250.09–.19 as what I mean by pc wrt "combs of a set of deterministic operators." From Counterexample 253.21, it seems that not all stoch ops. are reachable from

.20 → A dirty: Is a wtd sum of a bunch of M functs a good enough way to express any poss'l NM funct? (.28) While it may be true in theory" (A I should look into that), a second Q is can I get good approxns to NM functs in this way w. not very many M functs to wt.?

Two things: (1) Had idea I had on (ops / Functs that have $\begin{pmatrix}\text{no}\\=\\1\end{pmatrix}$ output) for certain inputs: In present context it simply ↑ the pc assoc. w. other funcs (OPS) ?? — So, would it change the norm. constant for certain Φ_i?

(2) There is a Norm'n Constant when one has the wts $\sum p_i = 1$ constraint in SSS of (.15 L)

.28 Im .20 Can I really express all stoch ops on this form?

Can I somehow have parallel M funcs that derive the corpus; so that the p.c. add rather than Multiply? — This may be very imp't! e.g. if I had f_1 & f_2 combin. ops w. f_1 & f_2 also f_1 & f_3; then pc's of these 2 would add.

Proof $M = NM$ — Non-Mem

So I could have a set of stoch ops formed by $\begin{pmatrix}f_1\\f_2\\f_3\end{pmatrix}$ in 11 · $\begin{pmatrix}f_4\\f_5\end{pmatrix}$ in 11 $\begin{pmatrix}f_6\\f_7\\f_8\\f_9\end{pmatrix}$ in 11.

so $3\times2\times4 = 24$ stoch ops: but their pc's add rather than multiply.

.36 However, each set of 3 f_i's has to be able to get corrects & by at least 1 of the component M functs. — Otherwise it would assign pc = 0 for corpus!

.00: 250.40 : Consider different ways of deriving stoch. Funcs:

One way: for a given Φ, a bunch of/k different A's w. assoc. pc's. Those/k are not [illegible] entire corpus is corpus via Phi's "pc of the partcodes"

$\prod_{i=1}^{k} pc_i$ · P(corpus via MFunct that gives correct output): Actually, we could code the corpus much more cheaply by considering only the MFunct that gives the right answer: No "probabilities" involved at all!

Here this set of k MFuncts is not meant to derive an entire corpus — but to be part of a derivation of a larger corpus.

.06 (from T. methods 250.[illegible]-.19 can be put into the exact $\Sigma 1$ or/$A\Sigma 1$ form!

Say we have 3 M functs that we want to use to derive the corpus: F_1, F_2, F_3.

The derivation of the corpus, consists of a precorpus, deriving the 3 functions $F_{1,2,3}$, in the manner of $A\Sigma 1$.

For each Φ, one or more of the F_i is correct, so the rest of the code consists of specifications of the past times when the appropriate F_i was used — spaces are II codes so the pc's add.

We end up w. the factorials that have arisen in $\Sigma 1$, but there is "overlap" of codes because each A_j can be coded sometimes coded by >1 F_i (as opposed to $\Sigma 1$ for/Bernoulli seq)

.13 Needs work!

We can think of the corpus code as a listing of the appropriate F_i's for each A_j.

If only one F_i is possl for each A_j, then we have a purely Bernoulli seq ($\Sigma 1$) — otherwise, if >1 F_i is possl., we have extra, parallel codes. — This is "addition" of pc's of Codes has to be done properly! [essentially it is a "Baum seq" in which several positions can have 1 of 2 or more possl. symbols at that position.]

3 things! (1) The codes in .13ff when >1 F_i is correct; is not yet of various, Needs work. I think I have a formal understanding, but not nice eqns.

.19 (2) If none of the F_i work for a given Φ, we have F_ω, the default A.H. Formation — we simply have to derive the proper A_j directly. — or we have an F_i that derives A_j to some extent, but we need some extra bits to complete the derivation.

(.19) May be in the right direction to get good MFuncts.

(3) Certain F_i's will be of no value in certain Subcorpi: So each F_i could have an assoc Sub-corpus in which it is relevant. — In fact each subset of F_i's w.w.t a MFunct could have an assoc. SubCorpus. Perhaps this is Related to the "Subcorpus Problem"!

(4) The normal Bern. seq. coding suggests that the integrals of .25a.12-.15 have a simpler, factorial-type values. I could extrapolate/from the known values for a 2 symbol Bern seq. to a k symbol sequence. Then, check it out/by numerically integrating via MAPLE V (I may have done an Analytic Proof long ago! — Maybe a recursion type proof — formula for k in terms of formula for k-1) Say $\iiint \pi p_i^{\gamma_i} \pi dp_i = f(z, \gamma_1, \gamma_2 \cdots \gamma_k)$; $\sum p_i = z$; $p_i < z$

$f(z, \gamma_1 \cdots \gamma_{k+1}) = \iint_{z=0}^{1} f(z, \gamma_1 \cdots \gamma_n)\, p_{n+1}^{\gamma_{k+1}}\, dz\, dp_{n+1}$ or some such recursion.

(5) I'd like certain MF's to be able to look at Φ's output/probably on A's: & do this in a natural way — perhaps as a "special case" of .250.09-.19: The idea of .00-.06, but "fixed up"!

(6)

7/9/01
ID

.00: 251.90 ; The method of 250.09-.19 would seem not to be a very efficient way to code a QA corpus. It would seem better perhaps to look at a Q, recognize that it is a certain class & could be best coded by a certain class of coding techniques: This amounts to a kind of pdt. One way of doing this is via "Recogn." of sub corpus. So it could use the same induction scheme for each sub corpus, but the findings will differ, because the corpus is different & the "updates" are different. Maybe use 250.09 method for each sub corpus?

.08 Sub corpi are first acquired by having QA's being "indexed" by USER.

Next, USER put in less indexing & TM learns to put in indexing, using previous user indexes as Data.

Next, TM modifies indexing as it is used till to its ends, not nearly corresponding w. USER's indexers.

.12 In line .00: 250.09-.19 does not seem reasonable.
249.06 sounds better.
What about the Args. of 245.09 ff on Universal stochOps? XXX

578T3 notes
247.33 — N.G.

I Not I had some form of soln to the QA problem w. a 578T3 type proof: (say much lost 10 pp)
Formal

Couldn't find it: Try: 242.17 ff: (Try 242.22)! Think (about / in terms of) soln to Bag induction problem.

Soln. to Bag induction: Sol99 (2KintPairInd) : MDL → soln to QA Bag problem.

Say O_i [242.22] is the denotation of a stochop. O_i does $P_{O_i}(A_i \mid Q_i)$

We want to find O_i s.t. $2^{-10|i|} \cdot \prod_{k=1}^{\eta} P_{O_i}(A_k \mid Q_k)$ is Max.

"=" $pc(O_i)$

.22 → More exactly use with $2^{-10|i|}$ • wtd product of preds of [all O_i] on $P_{O_i}(A_{n+1} \mid Q_{n+1})$.

A Universal d.f. on O_i's will assign $pc > 0$ to every [finitely] derivable O_i for stochOps derivable by continuous params, & Theorem [has to become] & its proof has to be modified.

not all stochops are expressible as a wtd sum of Mops.

Now, what I want is a look at the general stochop, O_i — it is constructable via the 3 input UMC of 242.220. Can I somehow do it via AI1 & "LISP"? Keep 242.22 + i & UMC in mind. We can think of AI1 as a [2 input] device: (1) Norm of Row (Col) Input to function (2) Output = output of function).

.33 In 242.17 (≈242.22) we have a 2 input UMC, to get to the model of 242.22 we need Random Access (Noise). 4 ways: (1) Add noise to O_i denn (2) pass thru gates 2 d.f. over stochops (2) Add noise to Q (an approx Q? — I don't know how to interpret this) (3) Put Noise in a third input. → SEE 253.00 !, But 253.37 is MUCH Better!

.35 By duality, an (arby) stochop O_i is defined by its universal / possl ways Given Q,

.36 — d.f. on A ≠ [for all ways] but d.f. on A can be deriv. by (1) Monte Carlo output of A; or for Arby A, the machine prints out A_i & $P(A_i \mid Q)$ in ≠ $P(A_i \mid Q)$ order. See 258.37 for Proof that the Model of 253.37 can implement an arby stochop of this (.36) kind.

.00: **252.40:** N.B. If a 2-input umc is universal on both inputs, I think that means any kind of info can be inserted at either input, & the info is equivalent! So 252.32–.35 would give 3 equiv ways!

If .00 ff is true then the approach of 250.09–.19 would seem to be correct! — i.e. We get a d.f. on M functs & this is (a corollary to .00 ff) equiv to any possl. stoch op!

I can review the various Arguts leading to Unreasonableness:

.06 1) 252.32–.35: That 3 possl. inputs of noise that define equiv stoch op.

2) 253.00 — that all 3 must be equiv.

3) Since AZI seems to use a umc, then 250.09 — the d.f. on M functs, must define an arby stoch op.

.09 This last result seems unreasonable! I may be able to define common stoch ops
.10 that are not of this form — yet I recc of form 252.36–.40 ≡ arby stoch op. (See .21–.24 for actual counter-example)
.14 A possible hole in argt of .06–.09! That the umc of AZI is not universal on the function & arg. input. — (I'm not even sure it's universal on the function argument input!).

Anyway the diff. of .06–.14 is very critical! Go over all of the pts. carefully. In particular, try for Counter example of .10–.11 — I feel that 250.09–.19 is somehow too narrow — I should be able to give counter example of a stoch op. that violates it: 252.36–.40 defines stoch op in most general way.

.21–.24 if actual counter example.

Counter example of a stoch op that is not expressible as (prob.) wtd. M ops!

.28 Good! .124

For input Q_1 → A_{11} P_{11}, A_2 P_{12}, A_3 P_{13} — input P_2 → A_{21} P_{21}, A_{22} P_{22}, A_{23} P_{23}

$P_{11} \neq P_{21}$, $P_{12} \neq P_{22}$, $P_{13} \neq P_{23}$ — In fact, none of the 6 pc's = any other & all 6 are distinct.

≡ COUNTER EXAMPLE ← WRONG! → 267.00 (Another counter example is 436.00K)

I don't see any way to express this as M ops as pc wtd sum of M ops. → This proof is wrong → 267.00

Keep Def of stoch op. 252.36–.40 in mind (& .21ff) as well; Try to get general stoch ops using ideas of 252.32–.35 — work out this last in more detail — just how is the umc defined to be "Universal"? (i.e. .00)

.28 My impression is that 250.09–.19 amounts to a p.d. over M-operators. (This is by adding noise to the operator dem.): By adding noise to the Q dem, we get effectively a pd over Q's w. a fixed (M) operator. (But a narrow kind of way to get a PD on Q)

Another way is to add noise to the output A, (which is a narrow way to get a P.D. on A)

There should be a way to modify (or use) AZI (Lisp) so that it has a third input that is of the desired kind of Universality.

In adding Noise to A, we get a PD that is indep of the M operator — which seems very narrow.

Adding noise to Q may be more general. Certain kinds of M operators could [illegible] different rate vectors. & Q & the noise & act like the noise was an input to an arby 2 input (matching operator).

.37 In Lisp, we can have functions of 2 (or more) variables! — So one can use "Noise" & be used to construct stoch ops. The no. of bits of noise needed to give a particular A, gives an estimate of the pc of that A. (We will always have > 1 noise string that will give a particular A value.

EMPT.! .40 (See 258.37 for a proof that this model can express all possl stoch ops as defined in 252.36

(see 264.12 on how to do .28–.40 on a 1 input UMC — A v.G. way)

ID

.00: 253.40! So we have the Lisp (functs operators) operating on Q. For each copy, use I if our operator gives it Rito A, fine; If not, we see how much noise (how many R bits) are needed to get it nitro A. In this way, for each possl. Operator, we will have a non-zero pc assignment for the entire [Q,A] corpus! As a default, the operator can not (use look at) Q, & use R to be A. This is usually a very expensive dcrn, hrr.

So, at all times, all we want is a stochop that gives the cheapest dcrn of the corpus (including cost of the op itself & the cost of the R inputs).

.07 One big problem is "Updating": In a general way, we have the Old Op, Opo that works w. the corpus up to now. We have new data, Q_{n+1}, A_{n+1}. We will search 'near' Opo: "Near" in terms of the code of Opo. i.e. its sub operators & their dcrns. Perhaps try to find shorter dcrn of Opo as part of "extended Updating" process — since a shorter code has more critical sub-operators. We will do search, so we can (calculate for a schmatter) the effects of any specific Soln. [See .20, .32 for 2 good tricks.

.11 Is .07 the best way? The search does not look at Q_{n+1}, A_{n+1}.

.12 This seems not to smart! Perhaps this is where Heuristics Enter?

Could I get insite on all-.12 by working MTM (Q_i, A_i) problems? (& SAAB TSQ)

Actually, in the SAARB TSQ — Did it look at the Q_{n+1} A_{n+1} in search (.11-.12)?? —

.16 Perhaps a very General (Hy level) view for TM to learn!

Given O_0 for $[Q_i, A_i]$ $i=1/n$ & Q_{n+1}, A_{n+1} & the structure of short redesc O_0:

What is a good metric over $|O_1 - O_2|$? i.e. what is a good order in which

.19 to search for Modifying ckt O_0? — (.38) TM will have to be rather smart for it can usefully work on this problem: So until then, I will have to devise (by Hand) a v.g. updating scheme (with testing, so TM can "understand" it.

.20 One way, is to find a short op for $Q_{n+1} \to A_{n+1}$, then find a recog. "fcn" for Q_{n+1},

.21 to distinguish it from all previous (i≤n) Q_i. (.34)

Anyway: I had that way to do QA inductions for MTM, which was o.k. but limited in various ways; all related to its being a MTM1 rather than MTM. Now I have this simple extension of MTM → $\overline{\text{MTM}}$ — so see if it is adequate — See if it addresses all of the points of 140.00 - 141.90.

.32 A common Method of Such for OZ ("Updating"): find a code w. a A' that is close to A, then use R input to devb A' → A. Related, is to use the unupdated Op on Q_{n+1} to get an A'_{n+1}, then try to find an R that gives the nitro A_{n+1}.

.34 (.21) Actually, .20 may not be so good! What amounts to is generating A, indep of Q — (NO!) which means we get the standard scripts for unconditional pc for A. (indep of Q). This Approach, is of

.37 course, dependant upon the previous corpus. NO! This is not what is proposed. The Heuristic was of accuracy in ANL (& SAARB TSQ) Learning "3 x 5" after "7 + 7" has been done. Learned.

.38 (.19) So, we start w. one or more methods to do .16-.19 (in factored form) Then we use it to work problems leading to ability to improve updating: Then we have TM work on improving the updating routine (.16-.19). This is an OZ problem, but my usual way to solve

ID

dynamics associates

.00 : 254.90 : OZ probs is in 2 parts: t. first part is an induction problem : t. second part is 2 kind of integration technique — but I had Mat Standards Monte Carlo Way to do it, Mat seemed Efficient

One trouble w. forgp. is That in t. Lsrch, corrolns betw. Conds cause much slowness

.04 Another Consideration on t. srch : When we are trying for new Ops, we Recursively have to test each one on t. whole corpus. Hvr, There are certain Mods. of t. old Op That will not change its resp. to all of or parts of t. corpus — so one could save much time by trying them first. One Op of This type recognizes "old" parts of t. corpus, & uses unmodified old Op on That part. For t. "new" part it uses Modified Op, so it has to test it on That part. T. new Op may be a min mod. of t. old one,

.13 so as to reduce its derv cost.

.14 On "Updating" : As of a long time ago (in IDSIA) I was considering Many possl. induction systems. Each had its own update system (But I suspected That t. update systems were all very ~ to That of t. MTM, AZ1)

What I have now, is a common way to express all possl. solns to operator induction problems

This makes it possible to compare their effectiveness on various sub-corpi.

W. This new idea (253, 39) (good way to get Good ops) I probably have to go back ~.16 (14?) and reprint t. whole system.

Initial system will only have one induction mthd. : 1 update mthd [More advanced systems will have several IA's & a FC) That looks at CQ & decides which IA's to try.]

Even in t. simplest system .04 - .13 (Mthds That involve testing only a small part of corpus) will be very impt.

At first glance, hunting for Good Stock ops. is n't much — much difft (in spirit) That hunting for good M ops. We do trials in pcost order : T. pc of a trial depends on both t. pc of an op (w. R = Λ), & how large R has to be to d crb. t. A. We can compute The total pc of t. successful trial in This way, & estimate its least (≡ cJs). We will do trials of Ops using short R's ; then new ops (& old ones) w. difft, longer R's, etc.

I'll have to work out details of recursive, recursive pc of sub ops, discovery of new sub ops — All part of updating UPDATING.

Maybe go thru pgmg of MTM first, (so those parts of $\overline{\text{MTM}}$ are clear) Then add in an extra input to t. UMC. In certain senses, $\overline{\text{MTM}}$ adds nothing to t. MTM problem.

.35 In MTM we look for a cheap Op (or set of ops) That does $Q_i \to A_i$ for known corpus.

.36 In $\overline{\text{MTM}}$ " " " " " " , plus R_i. " " " "

Its just That we have t. aux "R" input (in addition to t. Op derv) in $\overline{\text{MTM}}$. Its (usually) a lower pc object searched for, but t. search methods should not (superficially at least) be much difft!

dynamics associates

.00: 255.40: So Why did I get it ideas that MMT was so much harder. It may indeed be that the cjs's in MTM are usually larger than in MTM.

MTM has, normally, many assoc IA's, that could involves complex searches (?).

When doing Pred_A w.t. Stoch op. model of 254.37, we almost always get a p.d. of pairs for the output: we could get a diff A for each value of R! Conceivably, many R values would give the same A, or in some cases, w. infinite cb — we might have Output Pred_A indep of R —

So, one output w. pc=1 — Rej cd not occur if we considered all possible codes, hyp. → see 259.40-.43 for reasonably Analysis.

In General, the most likely usually A occurs when R=Λ [?? ← probably not - see .09 ff.], which is a great advantage.

Hyp, the R input has to be a prefix set, so $\sum 2^{-|R_i|} \le 1$ (Actually all we need is $\sum 2^{-|R_i|} < \infty$)

This means that usually Λ is not an acceptable input for R (!). (Since it's a prefix set w. only one member!)

.09 Actually, the way it works: we put Q into the operator. Q is self-delimited, so T. machine knows when it has minimized. T Q's must be a prefix set. On the R input, we just try inputs at random until we get output. TM reads as much of R as it likes, then eventually, prints A & stops. T. inputs on "R" that would give a crash/stopping, constitute a prefix set.

Each pred. involves feeding in these random "R"'s to get a collection of A's: We collect them, & note which ones have repetitions. We just observe statistical density of A's & use them to get random nos that Gen machine chooses.

Another, more systematic technique for generating Rai R's is not random, but a "left-hand on wall" search of a (complete) binary tree. The actual way it is done is described in "Osearch 1985" [5] pp 2

I also had a more recent analysis of Pris (In these "IDSIA" notes). I think the deterministic method may give them the most useful outputs, w.o. wasting time on repetitions — but I'm not sure it's so good if one only wants to spend a little time to get a set of reasonable A's.

Fortunately, one doesn't ask TM Q's very often — during its training — in fact one need not ever ask it Q's during its "Testing" phase.

So the fact that op may be slow, is not ordinarily an impt. constraint on TM. If we want TM to operate in Real time, it might be a problem. — But there are certainly more serious problems in TM's design!

.26 One Big problem is 255.04-.13. To make trial ops in a way so that one has to test them on only a small part of the corpus — because the changes an op will not change its response to most of the corpus. [whenever finds an (Op, R) appropriate to a QA pair, one saves both the Op & the Q. (pointers): The idea is that since one is testing a modified Op on the corpus, it may be necessary to test it on some old inputs. Knowing the R that once worked each would make it one's best trial R, & it could save much time → (36) (.31 Probly what to do: For each of the impt probs of 140.00-141.40 (& 255.04-.13) (& probly others)

Try it out on a TSQ. See how I seem to solve or avoid these problems. — Incorporate these techniques into the Heuristics for the IA's.

.36 (.31) One way to deal w. .26 is to divide corpus into parts (the sub-corpus problem) such that each part has its own (stoch) operator: Whenever tries a modification of such an operator, it only has to be verified on its own sub-corpus.

Even so: verification of a stoch op on a corpus of any size involves enormous costs! (← NO, see ↓) Gnerl. comput operator, one must check for each soln. for each Q & A. Well, not so bad! If there are k QA's in the sub corpus & each has pc of p_0; Then we just add the costs of each "A" such, so effective pc is p_0/k (not p_0^k). Also, for first trial R, we use the R that worked for the stoch op being updated — this could save much time!

7/11/01

IDSIA

dynamics associates

.00: 256.40: "IN Review": AZI could do MTM a bit, but only if hours were (initially externally) supplyd - since finding New Hours is a $\overline{\text{MTM}}$ problem. Hvr, comparing it to $\overline{\text{MTM}}$ solving General QA problems - in this case we would have to have TM be hyly educated before it was able to find hours in a useful manner: So for a long early part of the TSQ's, both MTM & $\overline{\text{MTM}}$ would be "about the same" in the sense that all of their hours would be externally supplied.

[SN] In early ALP work, I had the idea that the (formal CR=∞) solns to MTM & NMTM were the same. Hvr, from a counterexample of 253.21-.24, this would appear not to be so! Another possy, is that (Must (clearly) since MTM has special cases of NMTM) they are the same but the common soln. is different from what I thot it was.

For sequencial predn, it would seem that in both MTM & $\overline{\text{MTM}}$ one just does standard Alp. & finds all codes for the corpus. In MTM the shortest code is much shorter than any of the others (because the TSQ is very short). The aux "R" input that Ume is not needed to get the correct Pob. for $\overline{\text{MTM}}$.

Next, what about BAG INduction: Do (solns of) MTM & $\overline{\text{MTM}}$ differ? This is the Language induction problem. My early ideas on this were that MTM was not solvable — (soln was "undefined" → it only possibly (cases were in the TSQ.) formulation of the problem) a $\overline{\text{MTM}}$ has a maximum full CORE!

Probly .07-.08 is correct; But also note (.13-.15)! So I'd really have to use the full $\overline{\text{MTM}}$ soln. all of the time! On the other hand, for sequencial predn, the idea of MTM, is that there is a "short"/finite code that will generate the entire sequence. If the sequence has any entropy at all, an infinite seq. will need a code of length ∞. Hvr, even a seq. w. zero entropy can need an infinite code for seq. of length ∞: But the code length for seq. of length N will be $P(N) \ni \lim_{N\to\infty} \frac{P(N)}{N} \to 0$

I will define MTM as having a corpus that for SSZ = ∞ has a finite term.

Anyway, whether I start out w. a MTM or $\overline{\text{MTM}}$ corpus, the system will not be discovering hours for a long time. I will have to put in hours, & they may have to be expressed in probistic form: {say the 2 input Umes of 253.37}

.32 So in any TM, the major problem is finding efficient updating Algms. This is tied to the set of IA's used, since each IA has (& suggests) different methods. & In all IA's we will have hours for efficient updating.

One Q is How far can we go w. even a TSQ w. only a few update New Hours?

One Big Hour is to divide Corpus in to Subcorpi: initially no indexing ... (252.08-.12)

What I want to do now, is write (concise) Review of where I am now: Just how much of M/$\overline{\text{M}}$TM I have designd: what most critical probs are & what some stabs at solns. Also This would be Part of the IDSIA report.

dynamics associates ABCDEFG

1 $\frac{3}{4} \times 200 = 150$ So ~~they~~ 1.5 times as fast.
$\frac{200}{\frac{3}{2} \times 100}$

.00 | 257.40 : To what extent could MUTATION & Crossover & Fair sized populations be a useful Method to do TSQ's? The ~~the~~ Cands would be ~~t~~ 2-input Crisp pgms of 253.37 We look for a ~~the~~ Cand of max Gen'l Preds are best made w.~~t~~ Beast Cand. Abiz trouble (as 256.26) is testing Cands. Say a cand has only ~~t~~ Op dcrn. For each Q_i we must also find ~~t~~ Proper R to get A_i" — which can take "much" time! So ~~t~~ Q_i is would Mut & crossover be better than Lsrch? — (The Lsrch for small CD updating "has not been defined" (or; more exactly, I suspect there are many reasonably .08 good ways, but I only know 1 or 2). Any way GA is but one possl. Heur. for UPDATING .16

.09 N.B. I think my Main Line of Attack is to write a TSQ that I think I should be able to "Learn" — Then as I learn it, Express this Irng, & assoc Heurs, in ~~t~~ formalism that I've devised. Up to now, 253.37 my formalism hasn't really been "Complete": It seems to be now. I will want to take various IA's & their heurs for Update, & express them in terms of ~~t~~ 3 input UMC (of 253.37). Also note UNIversality of 253.37 (259, 40-43)

.16 (.08) **N.B.**: Since Evaln. of Cands appears to be very time Consuming: ~~If~~ I should use a Variant of GA in which we try to make a quick approxn. of ~~t~~ Evaln. funct'l. & use it for parts of ~~t~~ Trng. Another Possy is that I should have approxt. evaln. of cands be an important problem for TM. Normally, I'd think of this as a Problem for TM_2: & not have TM work on it Until it was "Suffic. Mature"! But it may be possl. to have TM work on this problem Usefully at an early state of Trng.!

.20 On Sub Corpi & Updating: For many ~~a~~ non-overlapping sub-corpi, TM will have separate Ops., that will be updated indply. Some params of these Ops will be shared (in ~~t~~ Spirit of STEIN) — but many will not be. It could be that ~~t~~ pc's of various sub operators are Completely Global, but ~~t~~ Operators for each sub-corpus are updated individually, since to an imat extent, their (main corpus used for evaln) are non-overlapping. Hvr, when an Op is updated, ~~t~~ pc's of ~~t~~ sub-ops used, will be reflected in ~~t~~ Global use of these sub-ops. Actually, this "Global v.s. Local" info in updating pc's of subops is quite imp. The STEIN effect is very imp. because sss's are usually small (~ 2 or 3, say). On ~~t~~ other hand, we do want some "locality" of pc's: so pc's are typically "Context dependent" — Otherwise, we run into ~~t~~ Scaling problem. So, ~~t~~ no. of Concs per subcorpus should be bounded (~~t~~) ("Window Bounded").

.37 A (Sorted) proof that ~~t~~ Model of 253.37 is capable of expressing ~~t~~ general stoch QA op of 252.36
Int. Gen'l model of 252.36, for each Q input we have a (possibly) diffnt p.d. on all possl. A's.
In 253.37, for each Q, we can have any possl. relation betw. ~~t~~ R or pair ~~t~~ output A.
Now, that's / relationship of $\Sigma 2^{-|Rel}$ to ~~t~~ pc of an output is what ~~the~~ ALP is about.
Any such $R \to A$ (functions / relations) defines a P.D. And every possl. P.D. can be defined by such a function — and the Operator dcrn input can "get Q" to define an array funct — (or have an array funct assoc. w. any possl. Q. ■

Quick ABORT
Quick ABORT: .15 - .21 (also .00-.03)
UNIVERSAL STOCH. OPerator Distribn Funct. .40

.00: 258.40: On Subcorpus Updating: Eval'n of Operation on a Sub-Corpus: Since we are doing many trials "Quick Abort" techniques are impt. Statistical testing can be useful: to use certain QA's within a corpus for Quick Abort, because they have been found to be "difficult" QA's (i.e. most OPs don't do well on them). It may be possible to save an enormous amt. of time by such "statistical tests". Remember: We are looking for a OP that is better (or even "about as good" would also be useful) as the previous OP. — so such tests are relevant. Note that we can afford to make errors. If we accept a stock OP the later turns out to be bad, we simply backtrack. "Backtracking" means the corpus augmentation is > one QA.

.07 (size of corpus)

.08 Outline of Review — See 208.00 for earlier review; 164.0 VSUP version

1. Dcrn of QA problems: M & $\bar{M}$: How much of Math can be trnd by MTM (QA)? That is All probs can be usefully expressed as $\bar{M}$ QA
2. Dcrn of AZ1: How this done, How updated: pc's, sem data, reasoning, pc modifn, scanning ...
The MTM QA model: AZ1.: The 3 input unc of 253.37: How to use a universal stock OP (.40).
3. Formal Solns of $\left(\frac{\bar{M}}{M}\right)$TM v.s. practical solns. } ONE of MOST DIFFLT PROBS. (Note .00-.06, .15, .24, .28-.38
long v.s. short CB. updating. } (.22)

.14

.15 [SN] ON UPDATING: Arrange so that when a UME has inputs Q & N, it will do the Q calcns first, & be able to rapidly attend to various random choices for R. After a choice has been made for the structure of the OP, we have a pgm that examines the OP & tries to put it in to SI's form. [I'm not at all certain about the feasibility of this!] Perhaps a goal of the system to start off, is to devise an OP that has this feature! : It may well be that this feature would be selected-for automatically! Well, an OP that got rite A quickly vis R input would be selected for. To some extent, this automatically selects against OPs that have low pc on the R inputs, but I'm uncertain as to whether it's really doing "Quick Abort" —— [This Q needs more work] (.28)

.28 abcdefg

.22 (.14) 4. SI as a QA problem. TSQ issue

5. Writing the TSQ. (258.09) often only one QA!

.24 ON Updating: Normally in updating, Modifns of (Grand OP) are only a small part of the corpus need be tested for those Modifns. This is an impt. kind of characterization of kinds of modifns one usually makes on the GOP (≡ Global Grand OP) (GOP ⊙). I really have to characterize these OPs (at .24). Try to find examples, Conversations from ....

.28: (.21) If we use a TRMC (for rct. machine) & it reads the Q tape completely, then asks for the next bit of R (it could have printed part A out at this pt.), we store the machine's state at this pt. & try various R values, repeatedly to get D.P. of A; or to find the R values for a given desired A. Perhaps we can arrange so that automatically by direction of machine — it reads OP dcrn first, then Q, then R. If so, we would want to save state after OP dcrn is read. Q, R.

.36 .37 .39 For a TRMC (& probably other machines) we can arrange the state table so that .36-.37 has to be met. (condition)

.40 The 3 input TRMC model of a stock OP of 253.37 solves the UNIVERSALITY problem! (approximately (wicked of continuous forms))
: — since all stock ops are/are processable in this form, as double sums. If an OP about printout a A is stop,
: (w.o. looking at R input) then that A goes all from that OP. To get pc ≠ 1 one must consider 260.00
.43 parallel OPs — (equivalently of much lower pc).

.00: 259.43 ☺ : So it looks like a main outstanding problem is small CB Update. [large (→∞) CB update is clearly defined]

.01 Some Updating tricks: (1) Division of Corpus into Subcorpi: Initially by indices supplied by USR, then tried, then improved by TM as 252.08-.12 → 258.20

(2) Try to devise a TM should try to devise a fast approx of the OP eval. func. 258.16
(In line w. my Criterion of how a proposed soln. of that problem): (ref) Stochastic Hill-climbing is an example.
Another possibility is to use a ruff approx for Q input & a more exact Gauge for final ("true") R input.

(3) Arrange so that the inputs that UMC are done in order: Opdesn, Q input, R input — & Restore can store the state of the UMC after each input. (259.28→38). This "storage" can reduce Testing time considerably.

.08 (4) (extension of (1)) The TSQ is modifications of GOP (= Good Op) are arranged so that Update time is minimum.

.10 (5) This can be done by doing GOP modifications That only necessarily require testing on very few (often only 1) CQA's. (259.34)
In Most Human prob solving, it is not necessary to test exhaustively any hypothesis that works on recent problem — on all of past corpus — why this is true, is much less probably all have redundancy → or usually on any of past probs in Corpus!

.12 (6) That Different IA's have different Update Algms (257.32): Hvr, the Update algms analogous to those of IA2 (i.e. pc updates, discovery of new sub ops, reparsing.

So study various IA's: (2) is a relatively simple example of an SA w. relatively easy update).

(7) Statistical testing (259.00): Just test after each past QA's & see that pc's of soln. for new Op are close to those for Better than those of old Op. Tested

SN General discussion: (4) & (5) (.08 .10) seem very important. (7) could also be important in that if one makes a Mistake & accepts a bad GOP, one will probably have trouble in near future, and be forced to "back track" a bit.

— Or simply do wider scratching (Pro: by guess back tracking is more likely to leave low cc to soln.)
Hvr, the General idea of (5) (.10) is most critical! Humans usually find solns that don't have to be checked on much of the corpus. This could be for "logical reasoning" reasons: trial GOP arguments are (devised, selected, invented) so they don't have to be (verified) tried on much of the [redacted] corpus.

.18 .19 (8) Find a shorter by pc step op for Quiti → Anti: Then find a way to distinguish Quiti from all previous Qi (i<n): 2<n;
.20 (254.20-.21, .34-.37.) The Ops of this sort don't have to be verified on previous Qi:Ai's (i<n). → 261.08
Generation of this Op will use same pc's as present GOP used — But context sensitivity of pc's is key!
Somehow, the system must compute the context. Some simple default context? No — that would be "no context at all", & would give very low pc to all concs! Maybe use "recency" context?
Actually, this is a general problem; whenever contexts are involved TM must have a way to Evaluate them. The present case is the normal case! If TM can't deal w. it, it can't deal w. contexts at all!

.26 (More "Gen Discn"): I'd like to start on TSQ as soon as poss., to get ideas of how humans solve "Update problems". Before doing this, I'd like to be (knowledgeable) enough to know what to look for, & be able to recognize something useful, when I see it! Also, I want to write kind of review of the update problem for myself & for the IPSIA Report ("working paper") → 332.24ff is a (B.bl.) review of some simple ideas on "Updating".
This involves knowing what problems are, so I will notice when a soln. to one appears!

SN For a TSQ: A possibly interesting sequence: Evaln of Alg expressions by successive "Simplification". There are several (Heavy Concs) involved — all of which would want TM to learn.
One Q is: do we want TM to "learn" from needed heavy concs? To do this would require a very large, diverse set of problems: The (concs / heavy concs) Elim ranking of: (3) The idea of a "value" often expresses/subexpresses.
(6) That a shorter expression is "simpler" & closer to one's goal (usually!), (4) The substitution of an equivalent value for a sub expression, leaves value of Master expression invariant.
(1) That one can solve problems by "Hill-climbing" - successive actions that improve some Gauge (in this case Gauge = Simplicity)

261.00

261.00

ED

dynamics associates

An Organizing Principle to Understand what TM's Main problem is .04

3.5 ε.b. 3.5/6 = 90% for 1.25 mby

or AAron in Google!.

SN AAron Ppmi is very low complexity Art! Get refs. from Raykurzweil's Yahoo Page.

.00: 260.40 SN Perhaps Ken Haas' Thesis would be useful to read now. It Seems more explicit than Lenat's descr of "AM".

His (Lenat's) problem would seem to be close to that of MTM (i.e. maybe $\overline{MTM}$) — but the Gdcc was a bit unusual: It was an estimate of "interestingness" of a Cand — & this Gdcc could evolve —

[That was true in "AM" — was it also true in Haas' "Cyrano"?]

.04 SN I had earlier, conceived of a TM that used many "IA's" in attempt to solve ($\overline{M}$) QA probs. It would seem that since all IA's have the same Gdcc, that each IA could be regarded as a kind of "Heur". Tho, in the case of Numerical v.s. Non-numerical $\overline{M}$ QA probs, these IA's could be very different — & the final Gdccs would even look different. .19 264.25

.08: 260.20 One way this could work: For $Q_{n+1} \to A_{n+1}$, we find that previously successful $GOP, [O_n]$ no longer works well! It gives a very low pc to A_{n+1}. We try modifications of O_n in "the usual way" (what were those are! ☺). We finally find an $\tilde{O}$ that works for $Q_{n+1} \to A_{n+1}$ very hy pc, but it doesn't work well for $Q_i A_i$ of $i \le n$ — so we make a conditional operator from O_n & $\tilde{O}$ — O_n works on $Q_i A_i$ of $i \le n$; $\tilde{O}$ works for $i = n+1$; we have to find a means of distinguishing Q_i's in these

.13 2 classes (two ways to write it explicitly — since the data is understood). It may be that $\tilde{O}$ works well for a larger class of Q's than just Q_{n+1}: In this case, it is always cheaper to specify this class of objects, rather than the single member, Q_{n+1}

.18 [At work the pc of a class is the sum of the pc's of its individual members]

.19: (.07) from .04–.07: we can view the QATM problem as: Given the $\{Q_i A_i\}$ to Q $i = 1/n$; and to assoc O_n: Given a Q_{n+1}/more Find a good O_{n+1} in available time. All of the various IA's & other tricks can be regarded as "Heurs" toward this goal.

Macro sub-corpus.

Set $\{Q_i A_i\}$

.21 SN Some musings on .19: One way to do updating: Look in the "neighborhood" of O_n (382.11–.15 discusses "metric", the neighborhood). First we put as shorter code for O_n as possible (or equivalently, as to probability, as keys were created). Then, using the pc's of Concs. within O_n, (& probably OSL), we make modifications of O_n of increasing boost.

.23 One trouble w. this: That I'd expect the "nearness" should depend much on (Q_{n+1}, A_{n+1}).

.25 .21–.23 is the classic $P(X|X)$ in ALP form $= \frac{P(X,Y)}{P(X)}$. $X \equiv O_n$, $Y \equiv$ cand for O_{n+1}. in .25, we use the analog of "unconditional probability"

— Perhaps we are already taking $(QA)_{n+1}$ into account, since O_n & O_{n+1} are $Q \to A$ mappings!? A more trivial case is peculiar .21–.23 would look at the "part" of O_n that is relevant to $(QA)_{n+1}$, & look at modifns "near" that part. [I'm pretty sure that O_n does have "parts" of this sort, & incremental, when we almost always update O_n, we only update part of it.

So O_n can have "parts" & the Corpus can have "parts": There could be a (←) correspondence between them: But not nearly!

.33

.34 SN "Recency" context for pc's of cones may be very useful! Each subcorpus (subject/domain) will have its own recency. The time scale could be no. of problems (into past), No of bits of info into past; No of compressed bits into past.

This measure might include Q's only, A's only; Q+A's.

Here, occasionally, one might add Batch of data (e.g. Census or Dictionary or encyclopedia) — this is ordinarily, part of a Q.

.00:26'.90 : [SN] Advantages/disadvantages of starting out w. MTM: (rather than $\bar{M}$TM)

Advantages: The Problems are easier, yet the heuristics used, can probably be used directly or in Modified form for $\bar{M}$TM. (So MTM is a good "Training instance" for $\bar{M}$TM ☺).

Disadvantages): Conjecture: It may be easy to run "error" in a TSQ (whatever an "error" may be!) Could get MTM into a cul de sac (local extremum) from which it could not emerge. It may be possible to avoid this "trap" by using "Cmc.Net" & associated methods as described in Sol89.

Disadvantages: MTM may not be able to get context dependant pc for cons. — if not then it could not go very far due to the "scaling problem". It could be that context dependence aspects could be developed by USR (MQ) & given to MTM as fixed heuristics for PC Assignment. This may have to be true for $\bar{M}$TM as well, since the young $\bar{M}$TM would not have much ability in working on probs of real dirty (??). I could simply use "raccoon" Context (261.34)

ID

.00: 262.40: Dcrn. of 3 input tapes used in QATM!

The 3 inputs are "ordered": The first is the operator dcrn. ~~The first is a prefix code~~ It is a finite string.

T. second is the "Q" input: it, too, is a finite string.

The third is the R input " " " .

The inputs and outputs are all unidirectional. The first input must be read completely before the second is read. The second must be read completely before the third is read.

Output may occur at any time. When the machine enters the "stop" state, then whatever is on the output tape, is "The output", = "A". ~~Write these~~ dcrn. then, the 3 inputs must all be prefix codes (easily shown") (.33)

.08: Because of the prefix property, each input defines a pd w. $\sum p_i \leq 1$. Hvr, since we are taking the product of 2 pd's to use as our "wt.", we may have stronger convergence conds:

— for 2 pd's p_i & p_i'; $\sum p_i = \sum p_i' = 1$, Is it true that $\sum_{ij} p_i p_j'$ must converge more rapidly than $\sum p_i$, say? Ie: Say we put the $p_i p_j'$ in order of size, so $x_k = p_i p_j'$ & $x_k \leq x_{k-1}$. Would $\sum k x_k$ converge? Similarly w. 3 pdf's p_i, p_j', p_k''

$x_\ell = p_i p_j' p_k''$, $x_\ell \geq x_{\ell-1}$; would $\sum \ell^2 x_\ell$ converge?

Say $p_i = p_i' \propto i^{-1-\epsilon}$ $x_\ell \neq p_a p_{a'}$.

Q: does $\sum_{ij} \frac{1}{ij}$ converge? It diverges for each ϵ constant; so it diverges badly!

But still $\sum_{ij} \left(\frac{1}{ij}\right)^{1+\epsilon}$ converges. for $\epsilon > 0$.

$\sum p_i p_j''$ diverges if $\sum p_i$ diverges — no matter how fast $\sum p_j'$ converges.

T. Kurp. suggests that $\sum p_i p_j'$ doesn't converge more rapidly than p_i or p_j. — it converges as rapidly as (conjecture) for most of the 2 pd.'s. So pp' will have first moment only if both p & p' have first moments (we assume the p & p' are in pc order, where moments are computed)

If the $\{p_i\}$ is not in pc order (largest first), then $\sum i p_i$ sometimes will never $\downarrow$.

I suspect that changing order of p_i will not change whether or not it has a first (or any other) moment. —

— The convergence depends on behavior of p_i for large i only (remove a finite no. of small i 😊).

However, I'm not sure of Kurp.

.33 : (08) The reason we want the O_p, Q, R inputs in that order, is that it makes it easier to test them. E.g.: Say we want to test a particular operator O_b w. various Q's & R's:

We save its ~~station~~ final state after it has read the O_p dcrn. input.

Similarly, if we want a d.f. for O_b w. input Q, we save state of comp after Q input, & do trials of various R. The R trials can be random, or in order of length, or ~~lexicographic~~ "left without wall" for exhaustive testing (The length order can also be exhaustive)

7/15/01
ID

13.00
$18^2 \approx 320$ =
$.042 = \frac{1}{25}$
$25hv \times 3\mu t$
Richard Gott ?
$625M \times 3\mu =$
1875 J.

42km
18V
13 Joules.

1 Input Trmc (.12) ; v.s. 3 input Trmc (.20)

.00: 263.40: Defn. of "Universality": The Op dcrn. input is universal, in the sense that if $M(Q,R)$ is any 2 input machine of the previous type (Q & R ordered inputs, (so they are prefix sets)), then we can simulate this machine exactly w. the suitable Op input. (the Op input would have a dcrn. of M's state transition table & a simulation of operation of that table.)

The above Umc also has (at least one) work tape. The we will probably not be using Trmcs; at least Cusp Machines [or storage Modified Machines or probably Random Access Memory Machines.

Each Op dcrn & Each Q is decoded by a prefix set of codes: set. Umc can tell when a code ends

.09 & start reading the next kind of input $O \to Q \to R$. The R inputs need not be coded by the user in any way

When the machine stops, it will have read the last bit of the R input. — This makes R input set

.11 automatically a prefix set.

.12 Note: We can easily use a 1 input umc, because since the Q_i & Q_j inputs are prefix sets, the Umc knows when they end & can go on to the next one. As for R_i, (.09 - .11) makes it clear that this becomes a prefix set anyway! As with 3 input machines, we can save the state of the System after the O_i or after the Q_i code is completed. The "O_i state" occurs (for storage) when it asks for the first Q_j bit; the "Q_j state" occurs when it asks for the first R bit.

.20 A possible Advantage of a 3 input machine! Q can be parallel time Series. A can be prediction of another time series (or one of those in Q). Instead of stopping when TM has finished A, it can continue indefinitely (as for normal sequential prodn). We can only read up to a certain pt. (today's day): "A" is tomorrow's data.

.25: 261.07! On IA's as "Neurons": Very Good! A problem of QATM is clearly stated in terms of getting a good Codes for the Corpus. Each IA (almost) can be Usefully Regarded as making more than a Heuristic decision for finding good Codes.

.30 A main very impt aux problem is the Sub Corpus problem! Given a certain CB: How much & what part of the Corpus should be used to make a particular prodn (or $Q_i \to A_j$)?

.32 This may be related to the part of the Updating problem involved in not (unnecessarily) having to check the entire corpus when one makes a modifn. in the GOP. My mind is certainly not clear on this!

.33 If one uses only a certain Sub-corpus for a prodn, then the Op used further prodn, will have to be verified on that sub-corpus at most. — but I think this is only part of the problem of (.32 - .33) — There are other methods. 265.00

.35 7/16/01 On "Updating": That we should have a good, fast, way to get O_{n+1}, given O_n and Q_{n+1}, A_{n+1} — This is the main idea of TSQ's — That search for Solns to Problems is not a useful technique unless this $O_n \to O_{n+1}$ fast updating scheme is possible. I think it is possible because people do it. But still "It remains to be shown". So really, this "fast updating" idea is perhaps the most critical idea in TM — the idea of TSQ's. And I haven't really shown how to do it in General! (That Soln of n+1th problem should make Soln of nth problem easier)

(Spec 264.38)

.00: 264.40 : → Usually when one updates w.r.t. Q_{n+1}, A_{n+1}, one does not need to check first stage corpus at all! Why is this? It may be that one uses special Q_n-modifications that make this possible. I might look at a few TSQ examples to get some idea as to how this is usually done!

[Note that this "Quick updating" is not always possible. Periodically one does Serious Theory Revision — = Serious backtracking — that can involve much more work.

One partial soln to the "verify w.r.t. whole corpus" problem: (1) reduce the problem viz "a sub-corpus" technique (264.32–33) (2) Verify only w.r.t. specially chosen, critical examples. This is close to the technique of Statistical Verification, but not the same. It can be combined w. Statistical Verification. A good candidate "critical example" would be a $Q_i A_i$ situation in which the Q_i took a particularly long time. (The reasons for the long time should be large C.I.S. not some "External" reason!

SN Another problem that may be serious. When a Op gives a P.D. for A, we are always interested in the normalized pc of the "like" A. If we use the "similarity" instead, the main problem is that distinct Ops for the same problem will have distinct normalizing constants — & so be not well comparable. It may be that this is an erroneous can "Live with"!

Note, however, that a very experienced operator that has been given Q's that always have A's will have a very small probability of "U" (= undefined) outputs. So for Mature TM's, Normzn will not be a problem.

→ See 'sol'78 comments' folder ~7/15/01 on "Normzn"

SN There had been a problem w sub-corpi in which one s.c. would end up w. a certain pc_0 for the soln of a particular problem. A distinct s.c. would end up w. a distinct pc_1 for a distinct soln to that problem. P_0 & P_1 and pc for the entire corpus — including the latest problem — so P_0/P_1 does not give the pc ratio for those problem solns. (— Tho perhaps this was an error — one must use the same/sod corpus for competing pc's). More troublesome would be if P_0 & P_1 were obtained w. same corpus, but distinct CB's &/or distinct Heuristics.

Superficially, this problem would be solved by the 3 input UMC model of 153.37 (or its equivalent 1 input model of 264.12). The output of the UMC depends only on random(R) inputs. The ratio of pc's of 2 outputs depends only & reflects how one does for R trials. There are several "unbiased" ways to do: (1) random R trials in order of length (3) Exhaustive R trials (this may not be possible, since for many/most Ops, for all A, ∃ a R that will give that A. i.e. no A has zero pc. → However, probably not true, see .39–.40, they 266.00!

.36 SN Tho the 3 input UMC of 253.37 is formally correct, it may not always be a good way to express stochastic ops. The main necessity is that I want a Term w. 2 inputs: (a) Op desc, (b) Q input & this derives a pd over all poss A's. This latter "P.d." need not be the "R" input form.

.39 It can be in any P.D. form that is useful. The absolutely necessary "R" input type is that it is

.40 it is Universal — so we get a universal device that can go from R to output for all Ops & all Q's. !!??

ID

Boris Katz ? (MIT)

208.00 132M early review of QATM

240.10 has Goodman subcorpus problem: it integrates w. idea of IA's as "hours"

164 is a v.g. early form of QATM.

.00: 265.40: This last seems Unlikely! A user can desrb. any Operation, & many operations are not Universal.

But Getting back to 265.36: I may not usually actually use 1 of 3 input v mc Models, — but that model should be able to be used to criticize any induction I do w. other models. [Note 267.00; I may be able to use it. "dds corded" approx. meth. too

.03 So main apparent probs. (other than TSQ unity (2) 264.30 - 33 -) are ① deciding on what sub corpus to

.04 assign a Q to ? ② what sub corpus to assign a QA set to for "update" ? ③ How to verify/evaluate a final Op w. minimum cc 2/0 how to select OPs so that this evaln. is easy.

In line w. the conclusion of .03 - .04: In my QA machine, it seems that all "real work" is done in Updating QA's. That gives a new problem, Q, finding a likely soln, A, involves only searching over outputs of the current OOP via Inputs.

Yet, in fact, when Ahuman ponders & learns a new, unsolved problem, it can take an enormous amt of time & Creative energy. What's Going On here? Perhaps this depends on what one considers to be a "Solution" to a problem. Perhaps in some Domains (Math, perhaps) a "soln" to a problem is a good technique for solving it. The "technique" could be quite cc. consuming!

Well, very hard problems are like "Curing Cancer". (293.00). One has to learn certain General techniques for solving probs of these kinds. 243.00 ff puts the problem in a precisely defined form (as a Dyn. Progr Problem) but devising suitable approx. techniques is something that only a mature TM could usefully work on.

As is, the QATM that I've outlined, is able to solve problems by generalizing from such solns to previous problems. I planned to educate it by a sequence of QA's only. In the present framework, it would be unnecessary to give it a problem to work "to educate it". When given a Q, it simply generates A's as (what it Thinks) one of my pc.

But: I could learn to solve equs & other math problems. It could learn how to "invent proofs" of Thms. It could be taught how to solve very diff probs. — But, to what extent could we get it to do such "Originality"

.27 out of such a source? See 231.00 on "creativity"; we could teach TM "creativity" by Examples. (.30)

[SN] In an earlier note, I had IA's choosing a subcorpus as well as doing the approx. So this is a "big job" for a Heur! —[see 261.04]

.30 (.27) also by biasing Search probs toward longer concepts — i.e. $p \to p^{\gamma}$, $\gamma < 1$. — how can do this with only limited γ since it will diverge if γ is too small!

Or, use a very small γ; use random (using p^{γ}) choice W.O. replacement (not so easy!) — maybe hash codes to prevent retrial? Or, keep a list of trials made & "look up" each when a new trial is suggested. It may be easy to find ways to eliminate the most worthless retrials.

[In some cases, we may want to allow (say 2) trials — e.g. if this is "crumby"]

[~~proof~~ Ref 253.21 // M ops could ~~not~~ simulate all ~~the~~ M ops] is wrong!

.00: 266.40 ; Woops!

We can simulate various pc by summation of various MOPs. Say we have 100 MOPs — each w. pc = .01.

For each Q_i, we can have any def. on many A_i's (limited only by ~~the~~ our resoln. of .01). So the model of 250.09 ff may be "OK"

So, the M MOPs may be able to simulate any M ops — but the Q is — is it a useful, efficient representation?

[One can ask re Q about 253.37 (see 265.36)

→ To prove 250.09 ff! Consider an enormous no of MOP: Each one ~~is~~ has no output for all inputs except one Q_i. That it has output w. pc = 1. We have ~~P~~ k P_{ij} of these MOPs for output A_j (k is large no. like 10^{10})

for each Q_i. For each $Q_i \xrightarrow{P_{ij}} A_j$ we have k P_{ij} MOPs w. outputs A_j.

So we have only these MOPs for $Q_i \xrightarrow{P_{ij}} A_j$: k P_{ij} of them. OP_{ij} has A_j as output for Q_i's input, otherwise it has no other output for any other input.

.09 .10 T. simulation of .00-.09 is not good! It assigns Λ output to Q's much sooner almost all the time.

.11 We would have to renormalize — re-weight Λ's at irrelevant — but this doesn't sound so good!

Another way, hmm. Say we have k (k >>> 1) MOPs. k P_{ij} of them respond to Q_i by A_j | each MOP has wt = pc = $\frac{1}{k}$

If there are n_j (different) Q_i's that output A_j's: could we do a proper assignment of $Q_i \to A_j$'s w. MOPs?

I think this model works OK, but I need to go over it. → 442.16 ff for a "Positive" View: Seems to work!

Woops! 442.34 is very serious objection

We still have t. Q of 265.36 is this ↑ or 253.37 a good way / useful way to express MOPs?

Also "woops!" → pc of this demonstration is very low!

.35 This model of 250.09 ff has a very simple update scheme | I understand. Zig if ~~a~~ one MOP had the correct A_i for its Q_i. The analysis of (.00)ff postulates many MOPs have each the correct A_i for a given Q_i. (Computationally / Mathematically) the problem seems difficult

.38 Say w_j is the wt. of the jth MOP (MOP$_j$) Then we want $G = \prod_i \left(\sum_j a_{ij} w_j\right) = \max$.

$a_{ij} = 1$ if MOP$_j$ gets the A_i for the i^{th} problem. Otherwise $a_{ij} = 0$.

So the L's multipliers w constraint $\sum w_i = 1$. The partial derivatives look very messy!

Therefore I in G (of .38) the coef of w_j is a big expression involving by powers of all coeff. of w_i's.

Since all of the w_i have to be ≥ 0, It may be that there are no local Maxima. In this case,

.26 it might be adequate (if there are M MOPs) to fit an M-dim quadratic form into each

.27 pt. in M space & do hillclimbing. In order to have a soln., each correct A_i must have at least one MOP that gives that A_i for Q_i. Typically this gives M+1 equations M+1 unknowns. We have M linear eqns involving the w_j's & then in addition $\sum w_i = 1$.

We then use L's mult. on the Quad form, w. constraint $\sum w_i = 1$

.29 Presumably, one look for MOPs that have a big no. of correct A_i's for as many Q_i's as possible → i.e. $\sum w_i = 1$

.30 Maybe we can do it w. a relatively small M — say M = 10 or 20. We search over a M-1 dim space — so basis vectors are different!

Another trick is to divide the MOPs into subsets of "non overlapping" MOPs. "Sub Corpi" (.27-.29) ff solves this

For options of t. $[w_i]$ one doesn't have to recalculate the MOPs.

It just involves solving for t. $[w_i]$ & can be relatively fast if, say M is < 50.

Solving 50×50 matrix takes ~~n~~ 50^3 steps = $125k = \frac{M}{8}$ — which is not very many steps.

& M = 100 would take 10^6 steps, which is significant. For a 1 GH processor: It may take a second.

But M = 100 seems very large; If we have ~~so~~ separate "sub corpi" (which is likely), then we may be able to have M < 10 (!). [We'd really like M = 1 or 2 ... 3 M+M problems.]

Actually, the "Quadratic form" trick of .26 ff may not be so easy! Finding the elements of the quad form

m: # MOPs n: # QA's

.00: : could be very time consuming. For m=10 there are ~55 coeffs. to evaluate. (55 quadratic coeffs plus 10 linear + 1 constant = 65 coeffs. The linear coeffs are simply 1. coeffs of 1. W_i in 267.38.

.02 I. expression is, I think $\prod_{i=1}^{n}\left(\sum_{j=1}^{m} a_{ij} w_j\right) \cdot \left(\sum_{i=1}^{n} \frac{1}{\sum_j' a_{ij} w_j}\right)$ — This factor is to cancel out various terms in 1. First factor.

→ Notation! — see (.21-.23) Actually .36 is correct

I in Σ', Σ' sum of .02 we do not include terms containing W_k ; here .02 presumes the coeff of W_k.

To get the quadratic coeffs. is much harder. I may take ~m^2 operations per coeff.,

so ~ $m^2 \cdot m^2$ = ~m^4 operations !) I think this factor involves summing over terms like

.60 $\frac{1}{\sum_j'' a_{kj} w_j \cdot \sum_m''' a_{\ell j} w_j}$ The Σ'' in Σ''' sum over all pairs sum over all terms containing W_k

" Σ''' " " " " " " " " W_ℓ

& Here we are evaluating the k,ℓ coeff of the Quad form. In (.07L) i = k, ℓ

We have to sum .07 overall ~ $\frac{m^2}{2}$, k, ℓ pairs.

.14 NO!! I want $\sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{1}{\sum_j'' a_{kj} w_j \cdot \sum_j''' a_{\ell j} w_j}$ Don't include terms that = ∞. } Not exactly! — see (.21-.23)

Fortunately, almost all terms = ∞.

If MOP_k was correct α times } Then there are α·β k,ℓ coeffs

MOP_ℓ " " β times }

If we use sub-corpi that are small (<100 QA's) then things may not be so bad.

So Conclusions:

That the coeffs could be feasible if we use small (<100) subcorpi —

Also, if Not many QA's have ≥1 MOP that gives the answer than the earlier (250.ff) method may give reasonable approxns (?)

How Good this model is to represent MOP's, however, is uncertain.

.21 Actually, it is wrong! since it involves partial derivs wrt W_k & W_ℓ, it should have a 2 coeff (factor) & products of the exponents of W_k & W_ℓ from accumulated expression.

.23 For some reason .52 is wrong.

However, in general, I am uncertain about these (.00-.23) results; I really would have to go over it in much detail.

Maybe use some .38 stuff for more accuracy.

~See .28 for Right way to do this !!

.28 This is a right way To compute Derivatives, since we want derivatives of a product: note: $\frac{d\prod f_i}{dx} = \prod f_i \cdot \left(\sum \frac{1}{f_i} \cdot \frac{df_i}{dx}\right) = \left(\prod f_i\right) \cdot \left(\sum \frac{f_i'}{f_i}\right)$ is provable by Math. induction.

$d(f_1 \cdot f_2 \cdots) = f_1' \cdot f_2 \cdot f_3 \cdots + f_1\, d(f_2 \cdot f_3 \cdots)$

$= \frac{f_1'}{f_2}(f_1 f_2 \cdots)$ $= f_1 (f_2' (f_3 \cdots) + f_2 d(f_3 \cdots))$

$= \frac{f_2'}{f_2}(f_1 f_2 \cdots) +$

From $\frac{d\prod f_i}{dx} = \prod f_i \cdot \sum \frac{f_i'}{f_i}$ we get a simple expression for .02: $\frac{1}{f_i}\frac{\partial f_i}{\partial W_k}$ = $\frac{1}{f_{2i}}$ or ϕ since $\frac{df_i}{dW_k} = 1$ or ϕ (if we don't care about f_i.

.36 So, in .02, $\frac{\partial}{\partial W_k} \prod_{i=1}^{n} \left(\sum a_{ij} w_j\right) = \underbrace{\prod_{i=1}^{n} \left(\sum a_{ij} w_j\right)}_{\alpha} \cdot \underbrace{\left(\sum_{i=1}^{n} \frac{a_{ik}}{\sum a_{ij} w_j}\right)}_{\beta}$

We next need $\frac{\partial .36}{\partial W_k}$: It seems to have ≤ n^2 terms; But do $\alpha'\beta + \alpha\beta'$

$\alpha'\beta$ is same as $\alpha\beta^2$; $\beta' = \left(-\sum_{i=1}^{n} \frac{a_{ik} a_{i\ell}}{(\sum a_{ij} w_j)^2}\right)$! So we have $\alpha(\beta^2 + \beta')$ for second diver. Note β' is negative

β'

~~So, computing $\frac{\partial}{\partial W_k}$ involves # additions~~
~~with no. of QA's in corpus. $\frac{\partial}{\partial W_k}$ involves #~~

7/17/01
ID

n : no. of QA's
m : no. of MOPS

.00: ; To do these calcns: first compute all of $\sum_{j=1}^{m} a_{ij} w_j$: There are n of them. ; each has at most m additions;
so $\leq$ m·n additions. Or is obtained by n multiplns of them. Next we compute $\leq$ n reciprocals:
$\frac{1}{\sum_{j=1}^{m} a_{ij} w_j}$: so n divisions : To get second derivatives,
Ignore these so n multiplies. For each second partial (There are $\sim \frac{m^2}{2}$ of them), we have to
add $\leq$ n (reciprocals)^2 : so n additions for each partial/diffce $\frac{m^2 n}{2}$ additions for all of them.
This $\frac{m^2 n}{2}$ is >> any of the other nos. of operations, so this is above f. cc of finding f.
partial diffces. Solving f. resultant eqns for Hillclimbing involves $\sim m^3$ operations,
The $\frac{m^2 n}{2}$ is an upper bnd: many additions are ommitted. Whether $\frac{m^2 n}{2}$ or m^3 is larger, is
uncertain. But $\frac{m^2 n}{2}$ is an upper bnd (assuming $n >> m$).

7/18/01 T. Lorge. tells how to get/peek for wts of MOPS ← By Successive Approxns.
Some initial Approxns: Have wts/ or no. of QA's they got correct.
2) Give nop w. a largest no of correct ans wins wh. of f ; rest wts = 0.
It is very well be that Approxn (1) is good enuf for most applications, & one doesn't need "Hillclimbing"

Q : T. analysis of 268.36 ff does it use "L's Multipliers: • Seems WRONG!
So " will give $w_i > 1$ & they will continue to ↑ as we do successive Approxns!
(267.27-.30) & discussed this: We simply use f. first & second partial derivatives of f. Gave.
to approximate f. Gave locally. Then we use L's Mults (w's "λ") to get an approx
peek for f. next approxn. This involves solving M+1 eqns in M+1 unks rather
.22 than M eqns in M unks. & so "$(m+1)^3 + \frac{m^2}{2}$" rather than $m^3 + \frac{m^2 Q}{2}$ key main Program!

L = 279.09
cut → 273.01
cut Hessian
Hvr. Note 281.00 exactly
268.28 ff may be inaccy!
Tho T. methods could be
very imprec. Poss 274.10-.14

7/18/01

ID

New Pen: .27 → 316

ABCDE

270

.00: : What were principal problems since I started working on QATM? ~ 259.08 164 208 in review

One of them involved some confusion in / multiple subcorpi : Multiple IA analyses:

Suppose we have several partially overlapping (IA/subcorpi): How do we combine Pearson products?

One way: Int. overlap region(s) we use weighted means of relevant I.A.'s : we choose wts

so that total PC of overlapping regions is max. — That's if we've already decided on the IA/subcorpi.

Use methods like 268.02 — 269.40 to get optimum wts.

If we have various sets of IA (IA/subc) to choose from, we consider the PC of entire corpus : multiplied by PC's of IA's used (presumably "IA's" include info on how to cut out sub-corpi — — If not we have to multiply by the pc of this info.)

For new probs, we use the same IA/subc selection criteria as before (these criteria are based on the Q's only) : we use the previously optimized wts.

5:08 P Hold.

7/19/01 Another way to look at the "target": (The error is somehow tied to this around p.165):

Say one has a Q : a bunch of IA's, [IA_i]. FC() looks at the (stored) Q : asks:

What is the probability that IA_i (of the entire set;) will assign the highest PC to the correct A?

This problem is solved like the standard OZ update problem is solved: by devising

2 FC()'s: F_1C() & FC): F_1C() involves drift inductively; FC() is a kind

of Integration of F_1C() : it's a purely Mathical operation. (However, write out more details

of both F_1C() & FC() .... I think I described this in my recent version of the "Dyslexia report". (FC() was called "GPD")

The division of this induction into an "Update" (QA) (input) vs. "prediction" (Query) (input) mode, is unclear, here

It's conceivable that TM would only do "updating" when a new problem was given; this way, TM would

have ideas as to what parts of corpus needed updating. — ∴ less work/s.

.27 7/19/01 Alvin Tech Liner 8.1 7/19/01 : 5:22 PM start of use. ~ cost $1.95 at Radio Shack's Porter Sq.

7/19/01

128 164.00 — 169.40 Gives a Dem of a QATM: Also 169.22 [-.40 has criticisms of weak pts. in K.

Systems: [One of these: "Non-Universality of the laws used to derive PEMs" is no longer valid. 259.40 250.08 25321 → 2567.00]

Main Criteria: 1) Optimizing FC) [which chooses IA's : subcorpi] is an OZ prob. for which

I use Lisp: which seems potentially very inefficient because of Cons'ns. Recurs. Calls.

2) (169.31): "That all of the info obtained in the TSQ isn't properly applied to the solution of the FC)

(optimization modeling induction) problem" (I'm not sure I understand this remark!) : It would seem that if the

model used for FC) was universal, there would be no problem — Perhaps this was the

point: At that time, 5/5/01 my model for FC) was yet universal ∴ couldn't approach optimality.

.37 So my present universal models would (presumably) take care of both disjoints (69.30-.32) No!

I think I was trying to say that the "drift" was a result of the fact that the model was not universal.

7/19/01

ID

.00: 270.40: So, my impression of t. system (as outlined in 164.00 – 169.40!

Main known theoretical weaknesses: (1) Use of L srch for f() optzn : correlation between cands. slows down L srch.

Other d.iffys: 1) How to "Mix" ~~different~~ preds of diffnt IA's!

2) " " update various IA's (in view of their diffnt. corpi)

3) In updating: How to reduce amt. of cc used, by not verifying trials on very many QA's.

4) I really need to Go over 164.00 – 169.40 to be sure that I really have a v. well-defined system.

.06 The main Thing, is t. overall system Goal (≡ t. Goal for improving F() & improving [IA_i]?)

So t. system Goal is related to t. Goal of .06. The structure of t. system + [IA_i] and F() are heuristics aimed at maximizing t. Goal. They are H.C. hours: (which possibly may be regarded as "Search Heurs")

So, I need to define ~~declare~~ t. [Q_i A_i], the [IA_i] [how do we distinguish on IA_i?]

Exactly what is F()'s goal? Exactly what is y. system Goal?

.19 How are t. IA_i updated?

SN Essentially, I am using an externally supplied set of IA's. The F() is (sub corpus selection) amounts to a v.G. Soln. of "t. Wrapper" problem. That's t. main contribution of Pkl TM model— apart from t. consistent use of ALP to resolve problems.

["T. use of ALP to resolve problems in Wrapper Optimization"]

or { Wrapper Optimization using ALP (description length complexity)?

Poss. titles for Paper / report.

Definition of a IA is its "Updating." The IA's we will be considering are all of t. following form: Given a seq. of ~~ordered~~ unordered Q_i A_i pairs, $i = 1/n$, the IA will construct an stochastic operator $O_n()$ which, for input Q_{n+1} will give a pd on A_{n+1}. This stochastic can take several forms:

[Monte Carlo; R input; output = list of A_i's pairs ≐ their p_i order; ~~Given A_i, it will give p_i~~

.31 Given n, it will give A_i, p_i w. highest p_i ; Also M, σ² or other common DF's. "best" P.F.?

If we have several IA_i's operating on same subcorpus, their predictions are reconciled by [weighting] each IA's with its (a priori Dem Lang Pkl) $\times \prod_i p_i$; p_i being the IA assigns to correct A_i of Q_i A_i, (with Q_i as input to it's $O_n()$).

"Updating" is of 2 kinds: (1) Global (2) incremental. In global updating the IA considers the entire corpus [Q_i A_i] $_{i=1}^{n}$ and finds new On and its associated parameters (& discarded also continue its params). This is ~~is~~ for many IA's this is only possible if n is small.

ID

.00: 271.40: In Incremental updating, we are given [Qi, Ai] and a O(n) that the system has obtained for this sub-corpus (sub corpus). We are given an additional subcorpus [Qi, Ai] i = n+1 ... n+k. (k is often 1). From this information the system modifies the parameters of On, on the basis of info. in the k+1 subcorpus, to obtain the stack of On+k appropriate for the S. corpus [Qi, Ai] i=1 | n+1. Usually this takes much less cc than the Global updates of S.C.1...n+k.

The Each IA has it own particular methods for doing Global and Incremental updating, most of them have certain common features, so that studying the updating techniques of one IA, with a view of improving them, will often suggest methods to improve updating algs for other IA's.

I will describe Global & Incremental update methods for ANNs & AZ1 & AZ1: Also for simple Bernoulli updating: Possibly learning filters (See Num. Rec.) Maybe ANNs & GA's.

.18 SN I think there may be a big problem in using > 1 IA! : If all IAs are in mut. exclusive

.19 subsets, so all IA's in same set have same (S. corpus) every time, — there is no problem

.20 Each subset corresponds to a Macro-IA, At. S.corp: of distinct IA's don't overlap.

If IA's do not satisfy .18 – .20, then predictive diffic. because IA's produce predicting A's for the same Q, have distinct s.c.'s (= sub corpus), & it is not clear how to weight them. Is the technique of 267.00–269.40 relevant? Certainly the problems don't correspond directly: 267.00 ff dealt w. deterministic cps (Mops), & each thing wts for them. Here, we may want wts for stack ops. (Mops).

Def S.C.

One way: Each sop has an assigned wt. That is used for all predictions: When other spp's predict .28 for same Ai, the predictions are combined linearly w. these wts & normed.

The pc for the corpus drawn is the product of the combined (i.e. combined semantics) pc's of all correct Ai's predictions, mult by pc's of all IA's used, mult by pc of FC) — the decision procedure for the IA Application. Seems a bit expensive way to decide 2 corpus [N.B. The analysis of pc of the wts of 249.23–250.19 will be relevant. If this is coding method of .28 ff is used,]

.36 However, much of the coding in .28 ff is in 11, so pc's add rather than multiply!

Yes: At each pt at which we have > 1 IA predicting, we have alternative means of coding,

.37 Also, the terms of the IA's (other than updating params) can be Apriori ∴ pc = 1 (no cost) perhaps also true of FC): (only the params have pc ≠ 1). Perhaps the methods of 249.23–250.19 could be adapted to the other (perhaps similar) continuous params. — like which we have to integrate/pc of corpus over all of parameter space. 249.23–250.19 is an argument/method 273.00

.00: 272.40 ; That can probably be used in many situations.

.01 HA! In that hill climbing method of 267.25ff (fitting a Quadratic Form), the Determinant of the Quad. Term at the peak gives us ≈ the true pc at that point! I don't immediately see how I could use this info, however. It might be useful if there were several discrete pts (local max's) — This would enable one to give wts to each such "local max" — or simply cutout the very low ones!

This extremization of 272.37 ff. is used for the wts. only of IA's only. We can use .01 ff to get pc's of IA's & of f() when they have continuous parameters.

.11 Hmm! The idea of 272.28 ff as a way to combine IA's: I may have written this up before! Somewhere betw. 165 & 265! I certainly did work on that problem a lot — Maybe I subconsciously remembered this approach ("Soln."?) Try to find past treatment of it! (165–265)

Anyway, is this Multiple IA prodn. the best Big difty? Look at this ↑ I may have had some good objections!

→ I hadn't "solved" it by 172.34 — still working on it by 177.11 (3 days later)
177.20 suggests use of "Decision trees" in early fc() to decide on which IA to use → for a particular CQ.

Since the wt. calcn. in 272.28–.40 will be time consuming & maybe a drag to program: We may want to estimate the wts. by seeing, on the average, how good each IA was, per QA prodn. so if the pc of the IA was p_0 & it did k prodns of $\bar{p}$ (geometric mean, then maybe mean pc of prodn = $\left(p_0 \bar{p}^k\right)^{\frac{1}{k}} = p_0^{\frac{1}{k}} \cdot \bar{p}$. ? Trouble is: if 2 IA's did same corpus, & got logPs A, they'd get wts of $p_0^{\frac{1}{k}} \cdot \bar{p}$; rather than $p_0 \bar{p}^k$ as they should. — So I don't know a simple approxn.

Take a clue from Bayesian estimates! Wts are no. of cases / σ^2
$\frac{1}{\sigma^2}$ is the accuracy; maybe analog of $\bar{p}$ so have wt be as k $\bar{p}$?

or maybe correspondence to Gaussian error be exactly same: for Gaussian error, so $\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\bar{x}^2}{2\sigma^2}} = \bar{p}$

$\frac{\bar{x}^2}{\sigma^2} = 1$ & $\frac{1}{\sqrt{2\pi}\sigma} = \bar{p}$. $\sigma^2 = \frac{1}{2\pi \bar{p}^2}$ so $\sigma^2 \propto \frac{1}{\bar{p}^2}$

wt $\propto \frac{k}{\sigma^2}$; so $\propto k \bar{p}^2$

$\bar{p}$ = geom mean of pc of correct answers.

Actually, in the Gaussn cases, the prob density ratios are $\propto \frac{1}{\sigma}$ — which is a more general result. So $\bar{p}^2$ gives "like" $\bar{p}^2$ ratios of wts.

Two impt. Q's ① do we want $\left(\overline{p}\right)^2$ or $\overline{(p^2)}$? ← They are normally quite diff! — a diff. by σ^2.

② We should take cross correln. into account. If we have > 2 pc's we will need the correln. matrix of the relevant pc's. A statistical analysis in terms of cross corrs. could give a not-bad estimate of pc's! — Tho I think the more computational intensive (hyper cc) method of 272.28–.40 is probably more accurate, if it gets the Hessian. (we may be able to estimate the Hessian by cross corr estimates)

Note: Since each IA makes several prodns for each Q.... well, we are only interested in pc's of correct Ac's. — which simplifies things. — Tho for small SSZ cases, using info on other prodns could be very useful (T. STEIN effect).

.00: 27340 : Probly best way (fastest, most accurate) way would be to use cross corrln. to get first approxn., then use 272.29—.40 for final approxn. (only 1 or 2 iterations will be needed.)

Actually, no corr methods 272.29ff could involve about same amt. of time: That per cross corrs & Rho matrix elements of 272.29ff could take ≈ same cc, or actually be identical!

.04 Akyuz: 272.29ff could be used to not only get good single predn., but to optimize

.05 t. internal continuous & discrete params of t. [IA's]! T. discrete params are optimized, by trying various combinations & comparing Hessians. (or just using Hessian-wtd mean of those preding) This could be extremely time-consuming b/c, since t. various combinations of discrete params are many in number, & one has to do a complete train continuous optzn. for each!) It might be possl to do a sample of t. [Q_2 H_2] dual set for this optzn of discrete params!

.10 .04-.05 is nice because it gives a very exact (tho not really precisely computable) Govt for t. set of IA's & t. predn. paramsof t. system AS a whole.

[[[DOD IN FACT one could also/similarly optimize t. params of FC) & (both continuous & discrete) along w. t. params of t. IA's!! — This giving a nice ≈ Non EL Govt for t. entire system!!!]]]

.14

.15 As a Special Case .10ff gives an "exact" soln. to t. ANN predn. problem (w.o. cross validation). T. discrete params" in a ANN involve how many nodes & how to connect them. T. "Continuous" params are t. wts. To get a predn., we do a Hessian wtd mean of all combinations of discrete

.18 params. A more accurate way would not use Hessians, but integrate over all possl values of t. continuous params. (t. Hessian is an approxn. to those integrals). I think this is t. soln. to it.

.20 ANN problem I got in SAARB. — It was part of a lecture I gave for Rich "Course".

.21 Hvr, t. ANN soln. of .18-.20 (w. or w.o. t. Hessian approxn) could be used for any predn. model (i.e. ≈ any IA!) & .10-.14 (including FC) is just that!

.23 This defines t. "Soln. Space" for QATM in a relatively Complete Way! It is, essentially,

.24 t. "wtd Sum over all models" version of ALP. → Note 465.11 → This is another alternative way of looking at t. 3 input unc. model of QATM.

In t. case of GA, we also have continuous & discrete params to distb. t. system — & we can use same soln. of .18 plus wtd mean over discrete param combns. We could use "Meta GA" to optimize these params: They are initialization params; Size of population, freqs of Mutation, freq of Crossover & Nature of Mutations & of crossovers, etc.

.23-.24 theoretically solves all "theoretical" problems in QATM!: In t. case of t. "Corrln problems" in Lsrch — We try various tricks to deal w. it & .23-.24 gives a criterion for selecting

.32 best way a/o using various ways. Well, not so clearly! .23-.24 gives solns. for CB = ∞ only!

.34 for finite CB, it suggests the soln: Namely; ~~sum/integrate over all params combns. But~~ To pick the best Hessian pt. of all discrete params that satisfy t. CB. T. hessians are

.36 constrained to params that satisfy t. CB, also. (I'm not so sure of this!!)

So it looks like a good Genl. Soln. to t. QATM problem! (Except, of course, for t. problems of Corrln. in Lsrch — (which is sort of solved "theoretically" (.34-.36)).

Go thru 165-265 again & look for ideas that might be critical of this present "Grand Soln".

In doing this also make list of implr. ideas worked on & their reference cards. A sort of Bibilo. Review of

at last 100 pp.

REFERENCE

pages 270 - 274

.00: : A new Method of Hill Climbing in Continuous Spaces:

Pick a small unit distance, D. (smaller possibly, yet larger accuracy so that noise is small)
In k dim. space, evaluate at initial pt. Then ± R (R is random direction in the space & |R| = D.)
From these 2k pts, pick jump in that direction of max G. (using parabolic approx).
Evaluate G at new pt. Then do trials ±R', again, but choose R' so it is only 80% of first R
by subtracting out that component. From these new 2k pts pick a jump in direction of
best of these 2k pts. Repeat ± R'' but this time we want R'' only to be ⊥ all previous R's. . . .
Continue until we do k jumps; then start over, w. random R, not necessarily orth to any prev.

This method is "similar" to Plateau 267.26 (fitting a quadratic form to local function), but
apparently cheaper, at first glance, but not nearly successful (sad)! .00 ff involves many function evals.
267.26 would involve $\frac{1}{2} \cdot 2k^2 = k^2$ evals of a funct. to get k second directional
(using $\frac{d^2}{dx^2} f(x) \approx \frac{f(x+R) + f(x-R) - 2f(x)}{R^2}$ or $\frac{1}{2} R^2$?)

On conservation of Univ Lipp Function see w 3 inputs

In writing a Review: The Section 16 ff is quite good. There are a few
Things missing (from it that I solved recently). (Tiuniversal/sup, a how to Mix IA results), but the rest looks fine!

.19 I'm uncertain about just what F( ) does: Say it (discretely) chooses a set of IA's for each Q.
Then we update the set of "chosen" IA's on these QA.
To make a pred/d.f. wrt. any fixed Q, we first ask F( ) for relevant IA's, then
Compute wts of those IA's wrt. that pred'n. using the methods of 272.36 – 274.24. (There are 2 or 3
methods described there, for assigning wts: One is fairly non-el.). We can also update "the parameter" R().
.23 While .19 – .23 may more or less work, it is not much in the spirit of what I used before!
— I.e., I was having F( ) make a pd on IA's, of the pc of a command, IA assigning the
In .19 – .29 F( ) doesn't output a d.f. on IA's: it outputs a subset of IA's.
Eventually, the system would have to take the pd on IA's & either select a subset for pred'n.
or use CC = Tpi as CB for all of the IA's: While this is attractive, I'm not at all
sure its such a good idea! For almost all IA's, Tpi would not be anywhere close to the mighty
useful. Also (perhaps) most IA's have a fixed cc for pred'n, so giving them a CB is
the same as a yes-no decision as to whether to use a subset of the IA's anyway!
We may, however, not be able to know how much time a IA needs, in advance!

Another way to look at it: F( ) wants to find the IA giving largest pc to the G → A. — an optimization problem.
So, if we did do (Lsrch, deciding time share) all of pi to IA_i — we find
certain IA's have no output until certain times — then the give the pc of the correct A. OR for IA's given more
varying pc's to correct A. Their assigned pc's may not be monotonic w. time! — But
its supposed to become "More trustworthy" as a IA spends more time on it. At the end of a "run" for a single QA update,
F( ) has gathered lots of data on how (frequent) the all of IA's that have been tested. This data is used to (277.11 Spec

7/21/01
ID

stochastic
↓
for S.OPs.

.00: : On searches for Meddins of a useful kind (of 3 types).

Then "OSL" ideas maybe critical. Hvr, my impression is that coding Phase longer operators is somewhat difrnt from coding a sequencial corpus. [tho! 277.00-06 makes them very similar

If a subtree has occurd only once, we can still (perhaps usefully, w. boosting) use it as part of d. defins of another tree or subtree. Perhaps an impt difrnce betw. coding Sequencial Corpus v.s. coding a Function: In d. Seq corpus, every thing has already occurred (all of it text) — so we have to parse it into subsegments: In Function Coding, we are inventing both the code elements (and d. parsing) as we code it.

.09 In d. SMAB TSQ's, Each Macro Op (that was actually usd for Predn) was put into Memy: As a result, we had, at all times, a PRE Corpus that consisted of d. nested defins of all older MacroOps. - These nesting orders was same as d. order in which d. assoc. problems were given. My Complaint at d. time, was that there were no other function defins! Later I decided that this was because d. probs (in d. TSQ) were very simple! Normally, w. harder problems, there would be new defins betw. d. defins of MacroOps.

.19 Also note that any sub-tree occuring in an Op can be used as if it were defined w. not very large extra cost. This is done by recoding, in which d. first time that sub-tree occurred, it was named defined and d. Prec corpus is used in all cases as a transform as a single symbol.

So in this definition process, we don't have to specify where, d. sub-tree occurred.

.27 This will be specified in Phase 2 (or more) actual uses of d. defined symbol (⇔ sub-tree)

In trying to modify a MacroOp (during Updating), one can back track

.29 by one or more MacroOps Definitions. This may make it easy to construct MacroOps that

.30 give same/acceptable results as before — Perhaps so/one downing have to verify from wrt d. entire corpus. } .29-.30 would be great if true! Well it is a little bit true at least!

Say we back tracks by d. 2 MacroOps needed for d. last 2 QA examples: For d. new QA, & d. previous 2 QA's, we devise a recognition operator & a special Op to X form those QA's into good QA's. We then need to test d. recogn. op & d. new special op on d. 3 QA's only. (d. recognition op may have to be tested on much of d. corpus, but there may be "logical reasons" why it has to work wrt d. entire corpus)

7/21/01 ≡ Compression based Lrng/Ganzn

ED Explanation-Based Lrng/Ganzn (.30) A very Good Approach!

But still, I'm not sure it's what they mean — But this discussion seems Very Generally Useful.

RNK tal.

;00: 276.40: An implication of 276.19 :27 : Say one is constructing Gaud. Ops. One has op_A & op_B as args for the choice of an new op; Opx. If op_A & op_B have been the args of op_C over or more times in the past, then the pr that Opx = Opc is hyer than if this were not true. (This is because we can then usefully define $Op_C(Op_A(), Op_B())$.)

It amounts to context sensitive choice of operator.

An analogous thing occurs in ZI: say AB has occurred in the past: Then if A occurs subsequently, B is more likely to follow (than if AB had never occurred in the past).

.06 (N.B.) Much (if not all) of the discussion of I 76.09 ff is true only for references to the parts of the Macro op as already parsed! If we look at the previous parsing of the Macro op, coded in primitive ops, there may be many unused, repeated sub-trees — but we can't use them, unless we backtrack to prim trees.

.11: 275.40: Update FI() (which then creates FC) by intersection. So, we have this "update cc" available & the Q is: How to use this cc "most efficiently". The output of the updating is the updating of some of the IA's — each "to some extent" (tho some IA's would be completely updated in the (finite) available time.

The Criterion for "Goodness of an Update of the IA's" is unclear.

.17 A possl. Goal: That the IA's that are likely to be used on problems like the present Q&A, have been evaluated using cc's that they are likely to employ in the future on such problems (Hyr Note 2582.01!)

So: 3 problems: (1) updating IA's (2) prediction using updated IA's (3) updating FC() ... (tho I may .19 do this "by hand" at first — so it looks like a pure "wrapper" problem.) Search I can do that re "wrappers" I have some hard copy. — Maybe use Google! → 278.13

SN As far as I know, only one of the IA's (Azr) is "Universal". (Also, the "R" input version of Az1 is "not bad". That it should use a "short R" to get more likely A's seems very "natural" & perhaps "workable with"!

Well, GA can be Universal; Recurrent ANN can be universal; Vepnik's system (prob). On p 228: of the 16 "IA's" listed, at least 8 can be made Univl. (228: lists of 150 "15 IA's") T. non-universal can be simulated by the Universal — the universal can simulate each other!

.30: SN On Explan-Based Lrng. How is an "Explan" → a Ganzn? : An Explanation answers the "why" question. It gives(a) short code(s) for the "what occurred". Usually short codes imply Ganzn.

An "Explan". makes the data seem "reasonable" (hype) by showing just how (for example) it is a special case of a more General rule. As such, it does not increase the rule; at most, it is another case of a "T. rule" so it may update the (usually continuous) params of "T. rule". (It shows how the data is a special case of a ganzn. that has already been made.)

Sometimes the "Explan" will be the discovery of a new rule, in which case it is true Ganzn. So, the new rule is stated; it is clear that the new data sets the T. rule; It is implied that other cases have occurred in the past.

If an explanation just shows how a set of previously accepted rules, & implies the present data, then I don't see where the "Ganzn" occurred.

On the other hand: If we show the new data can be accounted for on the basis of a (long) string of applics. of known rules — this can be a "short code" for the data. — But how is it used for Ganzn?

— How does it "Generalize"? 278.28-.34 has a useful example 278.00

7/22/01
ID

EBL .00

150.00
~15 IAs: 22800

→ Nite 297

.00: 277.40: So t. Q is: How does a short code (compression), (Always?) imply Genzn?

Well, t. "short code" means that t. batch of data is "more likely than it looks to be". So if one saw part of t. data, t. rest would be implied w unusually hy pc.

An example of an "Explana" would be t. parsing of a Nat Lang Sentence. — How do we get "Genzn" or "Lrng" from this? . Well certainly, after parsing, we are in a better position to "Understand" t. sentence more fully: — perhaps to give it an even better "explana".

"Compression" ↔ "find t. regularity in" ; "finding t. regy" implies lrng/genzn.

Very often, an "Explana" will be "compression by parsing" — not by making New Definitions.

"Parsing" shows how Data could have been generated from already defined Concepts (≡ old definitions)

Its Clear How a New Definition would imply Genzn/Lrng: But how does ~~finding~~ parsing fa— t. data do this?

In general, Parsing can be "hard work": i.e. it can involve much srch before one finds a good Parse! → 297.00

.14

.13 : 277.19: As I see it: 277.11-17 "more or less (vaguely) defines what "updating IA's" is about.

As for prediction, perhaps F( ) has to choose which IA's are to be used. [It might be that that choosing which IA's to update wrt a particular QA, would be ~~was~~ a non-critical decision — but — Not so!

If we update a IA on data for which it is ~~poor~~ inept, it will ~~confuse~~ often end up a bad mean score. Obscuring t. fact that it could, indeed, be v.g. if applied to proper problems.] (So we would lose much of t. benefit of that IA, because we would assign it a spuriously low wt. for problems it is very good at)

So in both updating & ~~prediction~~ prediction, choice of proper set of IA's is very imp't.

Initially, TM would do updating in a very broad, (Catholic) way, to ~~the~~ one be discovery of what IA's were appropriate for what Q's. Later, (Post Hoc) TM might want to declare certain Q's from t. sub-corpi of certain IA's. Here, we want to be sure (Want to be sure) that at least one (a practicable) Parse IA's are updated w. each QA, so the system is able to learn how to solve Parse probs. ↳ i.e. F( )

.28 A single example of a parse could significantly change/ continuous parameters of context-dependent application of Grammar rules, if t. SSZ's were very small.

.30 One kind of "explain." that Implies Induction: An "A" or Q: A can be "explained" by an Op or a
.31 short code that xforms Q into A. We can then use this same O_p to induce other A's from other Q's. → 297 on EBL

It would seem that selection of IA's for predicting & for updating could be almost identical.... Except that one might want to be more "diverse" in update, to explore t. capabilities of AI's in unexpected areas. So, if a IA would be selected for

.34 answering Q_0, then IA's would certainly update on Q_0. Here, perhaps it

.36 IA; it's selected for update on Q_0, IAs would not necessarily be used to help answer Q_0.

A possible annoyance: A given IA could have several disparate (non-overlapping) subcorpi. Each could be very useful in its own area — but because each [subcorpus implies a different update history, that IA's corpus pair will have different skills! → Maybe not so bad ☺ → 279.00 at all — maybe good!

↳ 297 on EBL

ID

Def. S.C. = Sub Corpora

.00: 278.40! If we use a single universal IA (for all of TM), it could have several disparate S.C.'s. We could start each S.C. by giving the IA certain implementations to start w. that are particularly useful for that S.C. In fact, constructing IA/corpus pairs in this way could be a having a different set of IA's to start w.

The idea of having a F() "front end" was suggested by how humans seem to work: i.e. one apparently solves a problem (F()) by deciding on the general set of tools one will use to solve it — also, perhaps, a "General Plan" for solving the problem.

.09: 278.40 For probm. updating all we need is the ratios (relative wts) of various IA's with particular Q: To obtain these using the optimization methods (quadratic form hill climbing of 267.29–269.40 it may be possible to only do a hill climb on relatively few wts, if all wts are not interdependent. It may be possible to do an optimization on several indep subsets of wts of coirfs of IA's. — This would probably take less time than a simultaneous optimization on all coirfs.

When one needs to optimize one set of coirfs: the rest of the coirfs are irrelevant & need not be re-optimized

I suspect that it is common for there to be non-overlapping sets of IA coirfs: It would almost amount to having separate TM's! But not so, since many pc's of common concs. are shared between these "separate TM's".

.18

.19 F()'s learning how to choose IA's for each Q, is diff, because any change in F()'s params would involve (perhaps), some calcns of IA's working on problems they never worked on before! — But perhaps small changes in F()'s params wouldn't need this much, & info into on IA's work on Q's that has already been collected will constitute most of the info needed. However, improving F() will be a diff task, & toward this goal, we should compute updates on many entire IA's (as in 278.24–36)

This looks like a serious problem. TM will have accessible to it at all times a large no. of (S.C.$_j$, IA$_i$) pairs [an IA$_i$ can be (simulated) by a fixed, & universal IA$_u$ and various equivalent of various S.C.$_j$'s. — This "Equivalent" could be in the form of certain (Heuristic) concs being given to that & IA$_u$ at very low recost — & (= its initialization).]

.30 Anyway, from these set of (S.C.$_j$, IA$_i$) pairs, it must induce how best to propose a small set (of just one) (S.C.$_j$, IA$_i$) pair

.31 for a new Q. So F()'s "self-modifn", is simply normal induction. — As all normal induction, its "update" usually consists of small modifn. of continuous params, w. less frequent invention of new concs & reparsing (of a small part, or very much in much concs available) of its F() operator.

Normally, the effect of a S.C. on a IA will be order dependent. (Time) — Since the Q's are given in some order empirically, the order will be the same for all S.C. almost — Here, F() could decide to allocate all of a Q to a S.C. that would have occurred in its "Middle" — This would involve

(allocate?)

280.10 space

7/23/01
ID

[redacted]
280

Reparsing: often/usually not Needed : .00 — .09

.00 : 279.40 : **REPARSING** : this is not so common as I had it way! Usually when a new (conc def'n) is made/discovered, it is via OSL (SSZ = "zar") so reparsing is minor — or, in general, the SSZ is quite small, or it would have been devd sooner! (The same cases are ok per new, so they need a large SSZ before they can be recognized as a big conc)

Occasionally, a conc. will be discovd "late" (≡ by SSZ), simply because the search missed it — (i.e. search technique could change w. discovery of a new search Heur) !!!!

Since the conc. of interest only occurred a few times in the past, usually not much reparsing is needed/warranted. → 307.08 .09

.40 : 279.40 : recomputing ½ of the S.C. to get rite order! Alternatively (faster) we could just compute the update of Mark of the Q by putting it at the (temporary) end of the S.C. — as a cheap approxn. For a large CB ordering is irrelevant, since the Q's are theoretically "unordered Defs"; TM could try all poss orderings to see which gave best pc; but for finite cc, the coding of the S.C. is incremental/sequential.

.17 Normally FC() takes various S.C.'s & does a "trial augmentation" w. a Q&A. Usually this gives a low pc for the Pot Q&A, but if it is hi, FC() will try the augmented S.C. as the proper S.C. for that IA. If the Q is not easily recognized, this will bear recognition will be an impt attribute to test. — So an impt part of FC()'s trng consists of finding good ways to recognize that certain Q's should be added to certain S.C.'s of certain IA's.

.22 This aspect of FC()'s trng, in which is able to take pick (known) Q&As & try (S.C., IA) augmentations, is Supervised trng: trng w. a teacher: (The Q&A's are constructed by a available set — TM can't invent a new Q to try, since Person is no way for it to get the assoc. "A".) Hwr, (etc in TM's career, it may be able to infer Q's of the internet, or of an encyclopedia — at some large cc. — or for it to do RW experiments — which would be very expensive in "cc".

.30

.31 Even for most, (if not all) Person "Q trials" (whichever "exp per (mean)"), TM would Probably be using a very Greedy CHEAP alg'm. to decide which trys to make. The cc for a trial would not normally be large enough to warrant much cc in selecting the trial!

For very expensive trials (like Q&A & some ANN or RANN) the cc of trials will warrant much more expensive algms for deciding what trial to make.

.37 Hwr, from the pt. of view of Diversity we don't want to be too conservative in trial selection (too "Elitist") since we Need "Diversity" → 281.00

ID

Context dependent pc's of Concs .22

.00: 280.40 : T. forty. (279.19 - 280.40) suggests that "empirical" optzn. of wts of (s.c., IA) pairs is unnecy. — That we'd do better getting wts via an "optimized" $f(\cdot)$ function. FC) would find/ increase various c.s./IA pairs & give them wts for any particular new Q & depending on a pt is empirical pc of s. selection Algm.

of 268.28 - 269.40 — 279.09 ....

.04 Note: In t. forty. I have discussed only very little (280.31-.40) effects of limited cc in both choices of (s.c., IA) pairs & even of QA's (updating) by those pairs.

Hwv, in any Universal IA the cc must be stated or implied, since true universality is only possl.

.07 if cc = ∞.

Hwv, I may want to draw up decided up a picture of QATM w. minimal attention to cc — just as a kind of Bookmark/ summary of present Pict.

Here FC) is able to a.) assign pc's to each (s.c., IA) pair in its available set#

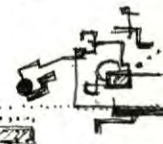
These pc's are used to determine which s.c. IA's are to be updated on what QA's — (since pc will be assigned zero for all but few (s.c. IA's,) wrt a particular QA) & also pc's will give wts to readings of various s.c. IA's for each particular Q. & potentially spurious

Also FC) is "updated" by ① creating new (trial) s.c. IA's & assoc pc's;

② updating of continuous & discrete params & a possl. reversion of part of FC).

T. restore t. QATM problem: ① Initial choice of IA's & assignment of s.c.'s to them.

② TSQ learning. ③ How Best to deal w. cc utilization



.22 Context dependent pc's of concs (in op construction) : also To Enable Discovery of Concs

As a corpus Grows, t. #'s of contexts Grows, so it's easier "in a pc sense" (but not necy easier in a cc sense) to find (legit contexts which enhance the pc's of various Concs. Whether This & t. pc's of c. concs fast enuf to cancel t. ↓ in pc's of concs due to their ↑ in number w. #'s, is unclear. Either this effect is sufficient to make contexts good enuf, or we have to find other ways to ↑ pc's of concs, or we simply can't deal w. large corpi! (Away out of using large corpi is to use t. time history of corpus.) → 300.00 is a progress on this problem

See 300.26 ff & 300.35 ff for very Good. soln to this problem! — But Words work!

7/23/01

ID

PSM (≡ "Prob Solving Machine" — QA machine, sounds too narrow)

.00; ; Back for review

.01 [SN] 277.11-17 on Meaning of updating "F( )": My present impression of meaning of F( )'s outputs: They are meant to be wts of outputs of IA's. We want those wts, p_{ij} to be $\Rightarrow \prod_{i=1}^{n} \left(\sum_{j=1}^{m_i} p_{ij} \right)$ (n = no QA's, m_i = no IA's on $(QA)_i$) → is Max.

.03 p_{ij} is / assignment that IA$_j$ gives to t. correct A for Q_i.

F( ) could update its wts for t. IA's when would get one p_i for each IA$_i$ — depending on what t. problem is.

If F( ) updated its p_{ij}'s V_{12} ☹ 268-28 ff — instead .03 allows F( ) to give different p_{ij}'s to IA$_i$,

.10 No! .03 is not right! I want to Maxz $\prod_{i=1}^{n} \left(\sum_{j=1}^{m_i} p_{ij}^{i} \right)$ p_{ij}^{i} is the pct that IA$_j$ gives to A_i — (≡ T. correct answer to Q_i)

p_{ij} is t. wt that F(QA) gives to A_{ij} for problem Q_i.

If F( ) was updated via 268.28ff + p_{ij}'s would be adapted ∴ IA's would get weights indep of t. problem under consid?

When we update F( )'s params we chose t. continuous & discrete params → .10 mult by t. Hessian of t. continuous params, is max. — Each column of discrete parameter forms has a peak for the contin. params, & has a Hessian assoc w. that peak.

These peaks in F( )'s continuous params also as well as t. assoc Hessians are available via 268.28ff — But we do not obtain the p_{ij} ($\equiv p_j$ only) this way

.21

.00-21 Tells what our goal is in "updating" F( ): It says "via 268.28ff for t. continuous params", but doesn't say how to optimize t. discrete params.

We should be able to get "PSM" (≡ QATM) to work on that problem.

Ideally, I'd like to be able to make trials in F( ) so that only a certain (perhaps small) set of its outputs are changed, so we don't have to get F( ) to re-evaluate a very large no of QA pairs. So: "Incremental Changes in F( )" is what I want. ("Incremental" in t. sense that only a little (i.e. for a small no of Q_i's) of F( ) is changed.

.34

So .00-34 tells what F( ) does. (Presumably, it assigns p_{ij} = 0 to IA$_j$'s that it doesn't want to work on problem Q_i.) ↓

Hur, Note 280.31 on desire for Diversity! This seems perhaps quite different from t. Goal of .00-.34

The in .00-34, t. search for good Sets of params for F( ) is an OZ problem — I'm not at all certain that (S rch is t. best way to solve it: Because of "coupling" bet w. t. cands. Other than that "I fall in to is in t. pd", it may be t. best way! → 253.04 spec

.00: 282.01 : ~~I next by problem area in retrieval~~

The IA's themselves, are regarded as having input Qi & processing a p.d. on Ai's in "some cc"

Note ⇒ 281.04 - .07 : if our has Universal ops, cc has to be considered.

Another point Univ. ops was 253.37 } 259.40 } 263.00 } 3 input unc; 264.12 equiv 1 input unc.
(I forgot what this point was! ☺)

.04: 282.40:- Also: updating the IA's is also an OZ problem — but in this case we usually have space "reasonable methods" to solve them: In the cases of universal unc's we will (probably) be using search over the discrete params that describe a IA.

The recent previous notes have some criticisms & comments on the proposed system.

The stuff on 164 ff to prosfans is also of much interest in those regards.

.10 Re: the cc problem : ~~If~~ we could have F( ) assign a time limit for each IA. — But whatever we do, the complete characterization of a S.E. IA will have to include used the E. amount of cc ~~used~~ for each of the Q's in the S.C. We can usually improve a (S c IA) by ↑ cc spent on any of its Q's. — but doing this retroactively can cause serious problems! Say we get a better value for an early (in some way thru) Qi than before. — This done by a new definition. To use this def'n. in subsequent versions of the S.E. IA, we'd have to do a lot of reparsing, etc. — Sounds very expensive! pricey

.22 So "We" (≡ F( ) ?) somehow have to decide how much cc to spend on updating each Qi by each ~~S~~c IA. spend

~~S~~c IA is also how much cc to be used on each Qi by each ~~S~~c IA.

.24 Perhaps use some funct of Qi, ~~S~~c IA, ~~etc~~ to both Q answering & updating.

.25 cc spent on updating/improving F( ) can be "↑ of all cc" E.- since this

.26 is a "SI problem" ☺

One possl soln to the cc allocation problem : F( ) does, indeed, do it, & as before, F( )'s params are evaluated by some thing like 282.10.

.30 → we want some thing like ~~Max~~ total Input divided by Σcc_i MAYBE ! → See 284.26 for an IMPT Consideration

How one would go about searching param space & complexity of. Gorc, is unclear — But it would be nice to have a clear "Pareto ranked" Gear for F( ) (& ↑ countless systems)

A desideratum of .30 $pc_i \to (pc_i)^{x^t}$ leaves ↑ environment. & x^t is a trans. that preserves Φ & 1 maps, so it preserves ↑ normed pc.

In .30 for each Σcc_i value we want ~~min~~ max $\Sigma \ln(pc_i)$, so perhaps it must be of the form.

$\max \Sigma \ln(pc_i') \cdot G(cc)$, w. $G(\Sigma cc_i)$ some ↓ funct of Σcc_i.

We want some functional equivalence betw. $G(cc_i)$ & $\Sigma \ln pc$. As it is, we have a partial ordering, w.r.t. Σcc & $\Sigma \ln pc$.

= cc of F( ) modifn. ∝ pc of F( ) modifn.

284.00

ID.....

Blackmail
Bribery
Burglary

.00: 28330: "Study Problem": Consider a single QA problem. Agem we have a partial order of Core below. cc & pc. ↑ pc is good, ↓cc is bad, for any pt. in pc, to space — but does a particular # of cc & # of pc make Core better or worse? It would seem to depend much on the nature of particular problem. Say we are studying Nuc Reactor designs. We have an evaln. funct that gives prob'y of failure. We have several empirical cases of failure. Would like them each to be between pc's of $> 10^{-6}$/yr. The evaln. funct now gives them 10^{-9}/yr. Should we spend 10^6 to get evaln.s up above 10^{-6}, or simply regard these failures as "unavoidably to be random events"? So it's a factor of 10^3 in pc — are we willing to pay $M for it? — we may!

On the other hand, there would be cases where a factor of k in pc would not be worth $M.

Also note that Σcc_i may not be correct! spending $1 on some problems is not equiv. to spending $1 on others. On the other hand, to get any particular result (e.g. pc) $[pc_i]_{i=1/n}$, we would be interested in total cc_i. cc is designed to be a function that is additive in this way — so perhaps Σcc_i is OK.

On the other hand, in SM analyses (in fortunes) seems to be the best thing to Maximize!

.10 It may be that the Core of 28330 is Good Enough to design a very effective TM. We can use it temporarily & keep working out the problems of Core Design. —

.19 If we change this Core, it may not be a amount to much changing TM's pen. (.26)

So the Q is: Do I have enough Detailed understanding of PSM to start on TSQ's. If so, I must write a good report, w. lots of Footnotes for myself on the details, explanations, Refrances.

.26 (.19) In 283.30 (& assoc. problem) Do we want to include pc of FC) modifies a/o cc of modifying FC) in the Core? Probably include pc of FC): as for cc of FC) modifn — [redacted]. I don't know + Nvr, see 283.25 - .26 ("Soft" solution)

Th. dizen. of 283.10 — .24 isn't always specific enuf on just how cc's are taken into account. — I should have somewhat more specific ideas before going to TSQ's. 283.22 - .24 is about as "specific" as it gets! Assuming the probs. & updating problems have same cc assignment function: cc is a funct of $\langle C, IA \rangle$ Q.

.32 FN on .32 The updating from operation & the prodn operations are slightly different:

.33 Updating involves pc of Ai (& correct soln) only, prodn involves pc's of all Ai of interest — normzd / for prodn pc's of all of the other Ai's involves

So it normally takes longer. If we wanted normzd pc's, we would have to get Actually, the cc for prodn is "up to the user". It could involve much updating of one or more $\langle C, IA \rangle$'s — it may indeed involve discovery of new cases. After a Q has been "answered in full" (i.e. lots of condy recovery, replanning, etc), the updating process is trivial & involves ≈ 0 cc! Nvr, in any case, TM has to run all of the $\langle C, IA \rangle$'s associated w. the particular Q. Note hvr: have good UMC w. larger corpus, the difference betw. 285.01

.39 .40 normzd & unnormzd pc is very small. → 286.00 spec

7/24/01
ID

.00: 284.40 — So just consider update cc in 7. eq. (Gore) of 283.30. (to start out!).

I think this problem is related to 284.00 − .16 — i.e. Each problem has its own "Importance level" — So we want to spend more cc on "Very impt. probs"! On the other hand, a problem that is "unimportant" in the sense that its soln. is not of great value, Maybe impt because it unlocks impt new concs to be discovered, that have much utility in the long run.

So how much cc to spend on an "update" is not clear!

.05 — Perhaps it not so simple, because in later updates one could add any needed cc:

Perhaps the best way to "Do" a larger corpus, is to update it incrementally, QA by QA, but w. small cc. Than using the prelim concs discovered, go over the whole corpus/again & again w. each time doing it incrementally — QA by QA — each time using all good concs from previous runs.

ABCDE

— Still, I have not addressed the problem of how IA's decide probs w. sharp cc values, & others (all Universal IA's) always giving pc's that vary w. cc — "better" pc's for larger cc.

A Good Set of "Study Problems": Consider following cases:

1) all IA's have fixed cc's for solns.) FC) knows these cc's in both cases

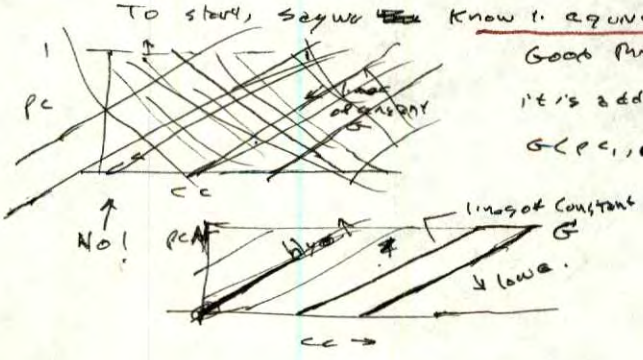
2) 1) but the IA's have different cc's for solns.

3) same as 2) but FC) doesn't know cc's, & they may vary from Run to Run

A soln. to 3) would be close to solving the problem of Universal & IA — because we can consider a Universal to be simulated or ≈ 3) "Take longer"

.28 — In case (1), there's not much of a problem — except that IA's that take longer" might be expected to use several "fixed cc" IA's. "In parallel".

.30 To start, say we know the equivs. between cc & pc: So G(pc, cc) tells how Good that pair is: We subject G to a Monotone function → it's additive wrt. pc. so G[pc₁, cc₁; pc₂, cc₂] = G(pc₁, cc₁) + G(pc₂, cc₂). We want to Maxz G.



-.00: 284.40 : On updating v.s. predicting! In updating, TM only has to find good codes for Ai's t. correct Answer. It can quickly reject Ops & R's that don't give this. Here it does try a variety of [redacted] s-ops to get this.

For predn., usually FC) specifies Rel IA's — that have already been updated — so it just runs a sequence of R trials to get a D.R. on [Ai]. As in t. note at 284.39-.40; Norman usually is unnecy in a "mature try"

T. Moral is, predn is usually trivial. Updating is t. diff'lt task: T. only time predn. becomes diff'lt, is when we decide, for a particular Q, that we'd like more updating (more cc) so we do Rest & then do standard predn.

.10 So: as for deciding how much cc to spend on what tasks: Updating is t. main task.

Note: FC) also spends time creating New sc's for [redacted] a given IA — these correspond to strong biases in t. operation of that IA. If we only have one IA (presumably Univ), — All of our scIA's are based only on datum cc's — or on pseudo sc's. A pseudo sc. is a sc that would have given rise to a particular IA; this IA having been created by t. Universal Reference Machine (hypothetical) "pseudo. sc"

What FC) tries to do is to find Q's that have an easily identifiable similarity, so that they are easily answered w. an assoc scIA. Essentially, it is a discovery of "problem classes" that are answerable [redacted] amenable to certain Problem solution methods — perhaps w. a certain pseudo sc Augmentation!

.20 So, [redacted] T. MAIN problem (≡ updating): Given a QA, which sc IA's to update (which sc's to add this QA to): Who shall be new members? How much cc to spend on each scIA?. (Default guess is $T_0 \cdot P_i$ (P_i is old wt. of that scIA)

Note that a "certain cc" is a function of (sc, IA, Q) may not be t. whole story — at first, we may want a soft cutoff of cc perhaps slowly ↑ (To) (as in LSrch) until t. pc of Ai is "Large enuf" (whatever that means!)

Or, in some cases, we would be about to complete a study that might result in a big pc, but "it will take a little more time" ☺. — So how much time to give it?

Well say .20 (w. $T_0 \cdot P_i$) maybe enough for a stuff, working update alg'm.

.30 Each scIA will have its update pgm in t. form of searches for short codes involving [redacted] fixed Q, modifns of Op's & of course finding good R's. Probly use LSrch.

.32 A part that is quite vague in my mind is FC)'s invention/discovery of new sc's for t. main univ. IA. It is certainly important. Part of it is automatically done when FC) updates a subset of scIA's w.r. particular QA: that QA is automatically adjoined to all of those sc's. A more interesting (& diff'lt) thing is to create sc's [redacted] similar to — 279.19-.40 + 280.10-.30 is relevant to (.32) 280.17-.37 is most relevant.

.37 In 280.17-.37, FC) takes various sc's & augments them w. various QA's of t. past, in trials to get new, useful sc's. In order to do this, FC) stores data on t. history of each sc → 287.00

.40

sec. 32
FC) also creates new sc's for a IA, & also creates new IA's
not so important if we have Universal IA's.

ID : [PSM]

startup 7/25/01 ~ 12:05 AM

:00 : 286.40 : its pc & its code & its identity & pc's & of the components Concs. of those codes. This means it possible for FC() to create new sc's by branching off old ones & adding new QA's to them.

:03 This could be done easier if we allowed branching of a sc into 2 new sc's; like original sc is $Q_1 Q_2 Q_3$; This can "branch" into $Q_1 Q_2 Q_3 Q_4$ and $Q_1 Q_2 Q_3 Q_5$

So a Big Q is when we do $Q_1 Q_2 Q_3 + Q_4 \rightarrow Q_1 Q_2 Q_3 Q_4$: we have a new sc, Q_{1234} : Do we save & old $Q_{1,2,3}$? In 286.37 – 287.03, we do effectively save all such sc's, & their assoc" parameters.

So this effective Branch construction of 286.37 ff could be the basis for any kind

:10 of sc (construction / creation / Birth): 279.30-.40; 280.10-.30 gives some Heuristic ideas on How this might be done — tries in 279.31 to show how this aspect of FC()'s behavior can be expressed as a normal induction problem — a kind PSM normally solves.

A minimal thing we want to store about each Q is what set of scrs/& what wts. they were given at the time that Q was first introduced into the Corpus.

The stuff ~ 270.00 to the present seems like a good understanding of how an input part of FC() operates: Mainly on updating, but also on creation of new scrs.

:20 HA! We can also make new (mal) sc's by AND, OR, NOT & other Boolean Ops on sets! — But FC() must learn how to do this so that it would be a reasonable pc of being Careful! 279.30-.40 has some ideas on How FC() learns things (like this?).

When FC() selects a set of scrs for a Q: and it turns out, one of them does badly (relative to the rest) on that Q; Then FC() may want to either find a way to give that scr a very low wt. for that kind of Q or to give it zero wt. — so that Q isn't added to that sc.

1) not automatically Q for scr if the scr does poorly on that Q
2) Developing initial scrs.

:28 On initialization of PSM: i.e. Construction of initial scr's:

We can start w. a Universal IA, & give it problems that all belong to same cc. Next, we start w. some subset one IA, & give it different subsets of problems, appropriate to a "different scr": :30

We can do this for several problem areas — so we get several scrs.

Then Purge is all done w.o. any FC(). In each case, there's only one scr being tested, so it always gets all of the wt. If it doesn't get reasonable/expected by pc for the city Ax's, then we might Acc or modify the TSQ. (This is essentially Sol89, but with stochastic ops, rather than the Deterministic Ops of Sol89) → (288.05 / 290.05 $/sec)

:35 Developing several scr's of this sort can be good initial training for Me — to develop TSQ's. The development of FC() can be done later.

It may be possible to do PSM w. only one scr! Perhaps look into this!

288.00
290.00

:90

7/25/01

ID

290
288

(i.e. solns. of problems à sub-funcs of solns)

257.40
.00: 289.40 : For SI of such a system, we want to improve the techniques that are used in searching for appropriate functions. Also, any heuristics that a human could discover, must be discoverable by the system. So a sufficiently advanced PSM should be able to successfully work on the problems of finding heuristics, using its own history as a sc, à suitable Q's. (Th. Q's would have to be "standard" a/o self-generated)

.05 : 280.35 : We can do this very nicely, by having TM learn about many of the sc(index)-types behaviors simultly: Just have clear indices ("names") on each QA example, so it can learn how to answer Q's of that index type. This way, TM discovers skills that use concs common to the various sc(index)-types.

Later, we can omit the index info occasionally, à have PSM try to reconstruct the index. — à eventually, we would omit those indices.

.10 Mor. for (even) later problems, we may want to introduce indices again, to quickly tell PSM that a set of Q's is in some common identifying field, à should be treated in at least one kind of common class (at least one kind of common class)

For "the report" I'd like to describe a MTM variant: the UMC has no "R" input. Trouble is, that wanted Context dependent pc's assignments to concs, — if we want to avoid one aspect of the Scaling problem's. Stochastic PSM can obtain pd's for Concs à parts of its normal induction activities. — This is not true of MTM. To work at all, the context dependence of the concs. has to be an externally supplied Alg m. — à MTM isn't able to

.20 improve it.

It's conceivable that at a certain (large!) point, MTM would stop growing & would grow very slowly in response to new input problems. (Kolmog/Levin suggest that for CB=∞ MTM would yet be a very large Alg m.) Hwr, w. limited CB, I suspect this Machine would be very large — larger even than the Earth to be in practical.

I had that of the "recency" context. How far one could go w. this is unclear. Depends on size of "recency" à on degree of Shiftiness in the T.S.

Hwr, in that stoch PSM would not be any good at storing contents of Concs. — I'd have to put it in "by hand" anyway. How advanced stoch PSM would have to be before it is able to work on problems of conc context discovery, is unclear. At first, we can start w. the "Recency Context."

.30

As for description (for the report) stoch PSM is not much harder to Describe than Deterministic PSM. The assoc UMC has 1 more input: that's it!

289.00

.40

7/25/01

ID

Sections: §1 Intro 289.09
§2 T. form of the training Data 294.10
§3 The Prediction Operator 295.00

.00: 288.40: So: Re: "T. Report":

Introduction: Derb. QA's (T. nature of t TSQ). — why its called "PSM."—

How practically any problem can be presented as to t. machine in this form.

No!: Simpler: Just talk about a string of QA's, & a stoch operator.

Perhaps t. intro is ≠ t. Abstract!

T. intro should derb. t. problem.

.09 §1 Introduction: We are given a large set of finite pairs of strings, $[Q_i, A_i], i=1 \cdots n$ that represent questions and correct answer pairs.

.10 We are then given a new question string Q_{n+1} ~~What is the probabilit~~ What is a good probability distribution on possible answer strings A_i? The problem is of much interest, since very many problems in machine learning can be expressed in this way.

Our solution takes the form of a probabilistic operator. Its input is any Q_i string; its output is a probability distribution on all possible A_i strings. There are many operators of this sort. ~~We chose one that has a short description and assigns high probabilities to the correct answers in the data sequence, when presented with the corresponding questions.~~ ← Replace by (.32–.35)

It is possible to search over all possible stochastic operators of this kind and chose the one that fits the data best, using this "Goodness of fit" criterion. However, for all but the smallest sets of Q, A pairs, this approach is well beyond

.20 the capacity of any computers envisioned in the next century. Approximation techniques are needed. The approach we use is a variety of "Incremental learning":

~~of which the stochastic operator appropriate to a string of Q, A pairs is~~

For that set of Question answer pairs $[Q_i, A_i]$, $i=1 \cdots n$ we have a stochastic operator O_n that fits that data well. We are given a new data pair, (Q_{n+1}, A_{n+1}).

~~We then modify O_n so that it fits the augmented data set.~~

We then "update" O_n by modifying the parameters that describe it — so that it fits the augmented data set.

[T. rest of t. introduction summarizes sections of the paper.]

§2 This describes the QA pair sequence: stochastic v.s. Deterministic data sets: The concept of

.30 use of a stoch op. to generate an ordered seq. of QA pairs. → (294.13)

.31

.32 We chose the Operator, O, such that

.33 $2^{-\ell(O)} \prod_{i=1}^{n} P_O(A_i|Q_i)$ (i.e. capital "Oh")

is as large as possible. Here $\ell(O)$ is the length, in bits, of the description of the Operator, O.

.35 $P_O(A_i|Q_i)$ is the probability that Operator O assigns to Answer A_i, when O is given Q_i as input.

$\ell(O)$ is defined Oh. here. $P(O) \approx 2^{-\ell(O)}$

292.00 spec for outline of main ideas in Report.

.40

.00 : 289.40 : Present tentative plan is ≈ 287.28 ff: to use one a universal IA & several indeed sc's "simultaneously" (≈ in //) for initial TSQ's. Perhaps try them individually (cascade mode) — to daboy TSQs.
In this system, there is no external FC(): i.e. IA learns to do this on its own. (FC)'s Functions

SN One old idea on how to train TM: start w. determ. probs. After TM learns them, use ≈ problems but w. a little noise., then add more noise. TM should learn "about" probability" this way. (1998 notes)

Reading Old Notes on TM seems like a v.g. way to put good ideas!

For "Report": Give abstract analysis of umc; univ. system.

.10 SN The Core of 289.32-.35 is "not bad" but we want to force TM to maximize as possl. (for data) on its truly as possl. A better Score would assign "degrees of Acceptability" to its various ~~to~~ possl. A's responses. Sometimes several responses would be about .13 equally valid. Sometimes certain responses would merit a larger measure of success. This kind of Response is "USR-intensive", hvr. & we'd like to avoid it, if possl. Maybe have some in Early part of Training only.
.14 The former Core is, essentially, a Remembrance machine RTM w. FEL (history) — It has no "Look-ahead" capabilities. — which is fine! Much easier to program and little (if any) possy of Machine Manipulating User. (See 291.23 ff for a bit more on Remembrance of this kind) 291.16-.20
If I want "Look-ahead" I can Get this PSTM to work on it. problem (e.g. see Cure Cancer 293.00ff)...

.20 SN Opt. possy of starting (IN: a term-matrix) NTM-type problems only: (≈ D-PSM.)
The heuristics & Context-dependance of Concs. could be done by a GA system — or ANN or RANN (Recurrent). — Slow, (but often "creative"; Mock "Diversity").
If we use GA, there is no BLOAT problem, since, for Induction problems, we ~~use~~ to implicitly make "size of Sol'n" part of "fitness function", i.e. best possl. way (for ∞ induction)

In Report: On "updating" discuss ~~the~~ updating Continuous parm. values, finding use of (daddy cases) function, reparsing. Its 3 main aspects of Updating: Also discuss "Backtracking" by examples.
.30 (no of Q's of Backtrack). Hvr, main ~~focus~~ development, addition of new heuristics, of new methods of Hvr discovery, will come by my studying 4. its Q's & how Women's seem to work them. This will suggest how to do (Improving Updating) — it's a well-defined problem & I should be able to program it to TM so it can work on it.

.00 ! Perhaps have several (several) ops developed in ||, w. occasional "crossbreeding" ("crossover")

To Get "Hybrid Vigour" — 2 "times" of Breeding would have somewhat distinct S.C.'s & no crossovers; — then Hyb vigour would suddenly be much crossover in offspring.

If we have several ops that derb. some S.C. & they have pc's within a factor of 10 or eq, say, we should store all of them, to be used in "Theory Revision" if some of them fail w.r.t. new QA. This is partly for better pc's but mainly to succeed, in "Backtracking" (presumably storage costs less than trials).

In the "2+3; 7×8 ..." sequence of problems: (in ANL (Sum 2,3) works), (mult 2,8 works)

T. change is a problem: "2→+" or "3→×" are expensive to derb (if t nos. are 32 Bit numbers!)

but "+"→"×" is easy to derb. if only 10 or so operators are available at first time.

—, &, ÷ can also be learned tiously, but even easier because of "context"

.11 (possibly recency concept among others). Try to find another or others: Try not to depend on "recency".

By stochastic TM: def.'s for Numbers could be found as special cases:

Integer Arithmetic: I forget arithA: ~ zero errors; Errors in float pt. often in last decimal, — but could be worse if many calcns are done to obtain Result. — our zero error if TM does exactly same calcns as wonderful "true" generator of data

T. idea of probby on Reals is rather different from D.F. of accuracy to be answers "to Q's in Natlang." ! dup. See 290.10-.13 on better way to deal w. QA's

.16 where many different answers are to be about equally good.

.19 Perhaps a better Core: Say we know t. stoch Op that generated t. QA, data seq. Then a v. pc Core is to minimize $\sum_j (P_{ij} - P'_{ij})^2$

P'_{ij} is t. correct pc for j-th answer to i-th problem. P_{ij} is TM's answer.

.30 We could have data of this type in infant TM, but very diffl in later TSQ's!

An approx in some cases: Either Give RTM (for updating) a list of acceptable correct answers, or, if t. no. of possl. correct answers is hard to anticipate: Have "Teacher" look at TM's many answers & give them "scores" (or yes/no). TM tries to get max of Σ pc that it gives to acceptable answers. If scores are given for t. entire set of pc values for given Q, P_{ij} is a regular "RTM" — but history = 1, so not so bad! (see 290.14 ff for dicta.)

.24

.29 SN ON t. IMPORTANCE of Σ1 or AΣ1 V.S. "Shortest Code".

.30 If shortest code is used is used in a sufficiently runaway; for a conc that occurs K times, into past (A)Σ1 can be K times as good as "MDL"! MDL could have K different codes for this/same concept, which is very wasteful in Lsrch. A more sophisticated MDL would use Huffman Coding, so usually t. difference between Σ1 & MDL would be small (about ½ bit on t. average) in this case. Anyway, ½ bit on a code of length 2 bits is 12½% — Not much import for ruff guesses. For a big pc conc, (pc .75) th. error of ½ bit is (larger, but I don't think t. average pc will will be bad by a factor of 2 at most.

One imp advantage of (A)Σ1 is that t. pc's are easy to compute

289.40 space ← introdctn. / abstract of record

.00 : 291.40 : The main critical ideas in the present psch model: The default "ideas" & alternatives:

(1) T. form of the data as unordered [Q_i, A_i] pairs. A BAG of such pairs.?

Alternatives: set v.s. Bag; ordered to some extent (weak precidence relation) | possibly Deterministic

partial ordering. : some subsets of QM's could be (weakly) ordered.

(2) Model of predictor: ~~a~~ a ~~map~~ input Universal device: Opdown, Q_i; $R \to A_i$ ← random.

stochastic models: wkd set of deterministic input models & set of input models. | Opdown, $Q_i \to A_i$

Interpret R input as Random for Monte Carlo Model.

One pair is $[P_{ij}, A_{ij}]$; fixed, j varys for i-problem Q_i: Listing of P_{ij} size - large finish

A_{ij} are outputs for input P_{ij} is output.

Maximum Significance of Universal operator — Metric ?
guaranteed to work, given has a reasonable amount of data &
one answers in limit of computation time.

P_{ij} is input; output is all A_{ij} with pc's of P_{ij} (1 ± ε).

.10 Predictor gives only one output (Gore is something like no. of corrected votes, or can be

~~a~~ to Real Research related. This is a MTM (deterministic). Could be checked if it
We never many of these MTM's in U. ~~form~~ of depend on operators of this sort → use AZT Any version post-y? of strings.
(2) (3) Gore: 289.32-.35 $2^{-l(\cdot)}$ $\prod_{i=1}^{n} P_0(A_i|Q_i) = \text{Max}$. Possible use of parallel codes of same O operator to AI post-hoc operator

.15 See (.25)

Variations: $2^{290.10}$ ~~of~~ Gore for each sector recognizer (sort of $[P_{ij}, A_{ij}]_{index}$: A form of RTM.
w.d. "Look Ahead"

If we do post-n "Look Ahead", TM tries to predict future Q's, Very Greedy
~~tries to~~ Could try to influence person (Manipulative use). We could have
limited "look ahead" w. ability to predict, but no desire to influence user (?) ← is this possible / feasible?

T. Gore of 291.18. ~~Min~~ $\sum_i \sum_j (P_{ij} - P_{ij}')^2$: P_{ij}' are "new" pc's

How pc of 0 ($\approx 2^{-l(0)}$) should enter, is unclear. P_{ij} are TM's pc's.

.20 See 291.18 for discussn of More Gore possys. ↳ if enters as $2^{-l(0)}$, but then $\sum\sum (P_{ij} - P_{ij}')^2$

Determines as measure of error in approxn. — use since Gaussian or better & Best distribution to represent.
& generalized ~~error~~ feedback-liable error

(AZ101)

(4) Updating techniques: Main type: AZI: Update pc's of each; find new facts; response.

occasionally back track to going back ~~to~~ k Q, A's & modifying t.

~~Update~~ t. ~~of~~ O_{n-k}, in view of the corpus augmented from $\{Q_i, A_i\}_{i=n-k|n+1}$. Is it st. AZI model

~~Update~~ Update the Beta matrix ~~Update~~ ~~Update~~ is recovered

.27 (.85) EN Perhaps discuss continuous param optzn. v.s. discrete param. optzn.' The (Hessian)$^{-1}$ factor: At least
If not in the report, write it up for my own Remembrance/reference. It's probably applicable in a critical way
to Barron (1993) on Gened ANN. To apply it, we need to info needed to deriv. the Data corpus,

.30 exactly, by means of the "ANN" coeffs & the ~~code~~ code for ~~the~~ Σ square error. T. ~~the~~ post-conv. set of
coeffs is the (Hessian)$^{-1}$ of the set of coeffs. mult by $\frac{1}{\sigma^2 N}$ (for N datapts, & MS error of σ^2.) — maybe multiply by
$e^{\pm N}$ or $e^{\pm \frac{N}{2}}$ also N^M power of $\frac{1}{\text{SSE}}$, maybe.

Could be subject of very Serious paper. Maybe post Marcus s/o Juergen to write it (?)

.36 (5) How this approach ~~differs~~ compares to/other some approaches to Machine Learning. → 293.20

.40

IP **LISP**

.00 ; 292.40 : One aspect of Lisp I haven't investigated is "side effects":

.01 One simple side effect that doesn't affect the language is printing. → 298.00

~~Is~~ asking for or accepting keyboard input : Does they affect language?

One thing that clearly affect lang : Use of Memory MU(A, X) ; as side effect, it puts X in Mem address A. MO(A) retrieves content of Address A.

.05 These operations make it easy to get data from one part of a function to another.

It seems unneccessary in my normal formulation of AZI : Each function can have as input, & output of any previously occurring function in the progression. The trouble is, to get records via expressions, I did, have to use a special device, & I don't know how important that was.

.10 [Well not that special: My "Functional" had/function & functionals. The functionals, could take several functions & create one recursive function from them. — I don't know if I considered a functional w. several functions as output.] → 298.00

.13

.14 A Possible distinction betw. my "functional lang" (which doesn't use memory addressable RAM) & Lisp —

.15 is that the pc's can be diff. In AZI, the pc of all inputs to a new level of "next values" functions, overall are the same: it's $\frac{1}{k}$: the bounty & no. of available outputs. In Lisp, the

.17 RAM addresses can have pc's — for the content & for the address.

.18 Probly I will decide on modifications/revisions of the language when I start doing TSQS. → 294.00 Spec

.20 :292.36 → HAVE SECTION in REPORT on How this problem/approach is (+/-) different from other probs/approaches: Compare to:

1) Sequencial sentence prob (Time Series ... ordered inputs)

2) Curve fitting : (Pd. is always continuous values & direction. PSM is nonsequencial.

3) # Sequencial Prob by linear/n.l. regression. (w to curve fitting : / each function of the previous values)

.25 4) Reenforcement lrng: Can be regarded as special case of Recognition lrng :

a) No look ahead b) simple, special kind of recognition function for set of $[P_{ij}\ A_{ij}]$ outputs.

Is easier than Recognition lrng, yet suitably educated machine can be given Recog problem : Is more capable, but more dangerous than PSM.

5) ANN (feedforward) : ANN is usually used as a continuous curve fitting Alg.

As such, ANN might be v.g. for certain kinds of PSM problems, but the error criterion used in usual

.30 feed forward ANN's is incorrect. Mt Shasta Huber's "Projection Pursuit" is much closer to the

Core used by PSM. [Now it's not obvious to readers, but I could outline a reason: (Note that if used

in this way "cross validation" is unnecessary & counter productive. — wastes data. Usually small no.

of wts is used & the Look Many algm is used to find soln.

Maybe discuss how ANN's universal for certain spaces, functions (viz Barron 93) — but

not Turing universal for discrete funct in the Turing sense.

.35 7) RANN : I think they are Turing universal : probably they are very difficult to fit to data as other ones;

they may be better for certain kinds of problems — & should be used for problems that have.

8) other IA's : see 150 & 228 for list. 150.33 228.00

.40

IP

Spec

.00: 293.18 ; Lisp (recur) One big Advantage of RAM is cc, w.o it, we have to recompute a value each time we use it. IN for stored in RAM can be osbagzi'd again we recall. The AZ (model, (which doesn't use RAM) could get similar pc's for funcs here. (Except for 293.14 - .17) → 269.00

Thata is also the possy of using a universal lang like Juergen used in His "Levy how to Lun" paper. This is simply a RAM computer lang. w. only a few (presumably universal) instructions. If more are needed for certain problems, add more instructions. Perhaps use ≈ FORTH — perhaps Hypercompiled version — (very fast)

In fact, the initial set of "prim." insts (+ others) around to the simple "or the four-tuple space described by the system. Perhaps I can simulate other systems / Univers (systems like (269.00

recurrent Support SVM's
RANN or vector machines on whatever.

.10 292.40 SN The intro (§1) should be a very clear, more or less exact prescription of the problem. So the later sections will only go into more detail & (perhaps) discuss Alternative choices.

.13 :289.31 §2 : The form of the training data.

The training data is a finite, (unordered) Bag of Qi, Ai pairs. [Qi, Ai] i=1...n.

Both Qi and Ai are finite strings. We may think of Qi's as pseudo-and Ai es as "correct" answer, but this is not the most general form or interpretation. More generally, Qi is the description of the problem, and Ai is a possible (not always correct) solution to that problem. (23) Ai need not be the

.16 There may be more than one solution to the problem. Though Ai need not be optimum, it will "often" be one of them. Since we are modeling a stochastic process, it will

n If the problems are all deterministic, each will have at least one correct

.20 answer .....

SN: Give examples of QA's at different stages of reduction.

.23 (.16) Since we will be modeling a stochastic, rather than deterministic process, the data set should contain "errors" (e.g. Q = 147 = ? ; A = 3) as well as [illegible]. [illegible] repetitions of the same Q as questions — sometimes with the same answer, sometimes with different answers.

Since we will eventually want the machine to be able to use data from the "real world", it must be able to deal with many errors in this data.

Another, perhaps much more important reason for creating a stochastic

.30 probabilistic rather than deterministic problem solver, is that a stochastic problem solver can more effectively work on the problem of improving itself. In advanced machines, a good fraction of the available time will be devoted to this task.

.41

295.00

Note In P_O, O is uc.

.00 | 29440 **§3 The Prediction Operator.**

Each prediction operator has at least one input — the finite string Q_i that describes the i^{th} problem. It output is a probability distribution on all possible strings, $[A_{ij}]$ that might be solutions to the problem Q_i.

This probability distribution might take any of several forms!

omit 1) A Monte Each output in which the probability that A_{ij} will occur as output, for input Q_i, is just $P_O(A_{ij}|Q_i)$.

(1) A sequence of pairs. $[A_{ij}, P_O(A_{ij}|Q_i)]$, ordered so that the pairs of most probability occur earlier in the sequence

.10 (2) In addition to the input Q_i, the operator O has an input for an arbitrary string A_{ij}. The output of O (u.c.) is then $P_O(A_{ij}|Q_i)$.

(3) The operator O has two inputs, Q_i and R. R is an arbitrary, potentially infinite binary string. The output is A_{ij}. If we feed random bits into the "R" input, the probability that the machine will output A_{ij} and stop is just $P_O(A_{ij}|Q_i)$. The part of the input string that the machine reads before stopping may be regarded as "a description of A_{ij} with respect to operator O and problem, Q_i."

.20 In this paper we will be primarily interested in the last form. we can realize such a device

omit It can be realized using a 3 input universal machine. Though a universal Turing machine with 3 unidirectional input tapes, one unidirectional output tape and one bidirectional work tape, there are many other universal devices that are in many respects as powerful. There are many universal devices or processes that can realize this behavior. In our explanations we will use a Universal Turing machine with 3 unidirectional input tapes, On unidirectional output tape, and one bidirectional work tape.

The first input is a finite string that describes the operator O. Since the machine must know when this description ends, the acceptable input set must form a prefix set known to the machine.

.30 The second input is Q_i, the problem description. It, too, must be member of a prefix set known to the machine.

The third input is the binary string, R. This input is an arbitrary, potentially infinite binary string and is not restricted in any way. However, only subsequences of the tape that are read before the machine stops prints the output and stops, acts automatically constitutes a prefix set.

+.00 : 295.40 : The machine first reads all of the O input, then it reads all of the Q_i input, Then it reads all of the R input. During any of this reading of the data, the machine might produce some output. The bit string, if any, that is on the output tape when the machine stops is regarded as the output. ~~If the machine never stops, the output is being monitored.~~ (However, for some kinds of problems (e.g. sequence extrapolation) the prefix of the output contains usable information and can be obtained before the machine stops — and we don't care whether it ever stops

Well, I may decrib t. 3 input functions abstractly: Then say it can be simulated by a one, 3 or 1 work tape one.

Then, say ~~it~~ for the discn. we will be using (instead of Turing's functional langs — Such as Lisp - [very reduced versions of Lisp have been decrib. by Chaitin & Kozen] for this purpose!

Or just don't bother w. the Turing! — the many Advantages, is that it's easy to see that input R gives a universal f. with inputs 1 a/o 2. Inputs (i.e. zero report.

The "Functional (any" : we want it to be a block decrib. any decribable finite of 2 finite strings

Not exactly! The three 2 inputs are equivalent. (O & Q_i) The R input is different in that it can be an infinitely long string. (well O & Q_i can also be infinitely long)

While I can't see any point to O being possibly infinite string. — Q_i is certainly R could be altered by beg R — i.e. we could have printing before the machine reads any of Q_i or all of R. (or before it reads all of O for that matter!)

Q_i could be the beginning of an infinite time sequence, & the machine could print out predictions "thereafter" of the next symbols of Q_i or of the next symbols of a parallel (time series) to be predicted.

How infinite "O" string would be interpreted is unclear. One way: The long R of O grows with Corpus (monotonic) Unclear as to how to compute but (pc of corpus. Also, any increment in info needed for a particular Q_i (at a particular Q_i (length-so-far) can be given by extra "R" input.

So far, it looks like O will be a finite input & the machine will read all of O — then it can read Q_i, R, & Print.

Perhaps increment in O input could give amount of "permanent learning" that O has acquired from Q_i thus far (wrong!

Consider the binary Bernoulli seq. w. parameter, p. O could represent the "latest" value of p & would also ~~be~~ more exactly, O would contain n_0 & n_1, from which the p.f. for the next symbol could be obtained. All O means is $P_{n+1}(0)$ & $P_{n+1}(1)$. ; $P(0) = \frac{n_0+1}{n_0+n_1+2}$; $P(1) = \frac{n_1+1}{n_0+n_1+2}$ & Laplace's

Drop §3 for a while! I'm getting into some interesting points, but I don't want to use up my limited cerebral many spaces. it just now! Go on to §4 on Prediction and Updating.

PREFIX Sets -.21 **EBL** (.00)

.00 : 278.14 (on **EBL**) If an explanation gives no predictive power (≡ compression), it is cognitively worthless! "Why did she say that?" — "because she believes in God". This "explanation" is useful iff we can use it to (help) predict her behavior — which it can be used for.

If an "explanation" simply applies a well known rule ↑ gets confirmation: then it only strengthens (by how much it is ↑ in ssz or the rule) so ↑ in its precision.

He flunked it. Why? because a) He didn't prepare for it.
b) " " got only 4 hrs sleep night before
c) " was stupid!

.08 Each explanation has a different prediction about his future behavior → EBL 309.00

=H

Write about Generality of methods for generating parsing sets, that were used in OSS (84 ff) see (.12 (AF)

ABCD efg

.10 In General about PSM: I expect PSM will have various S.C.'s & learn to switch problems betw. them. (at first by ordering SC's — later, less ordering).

Note that various areas of endeavor, effectively using different IA's can be "simulated" by suitable ranges (sometimes very large) SC. or pseudo SC's. — Even this way, the present model of PSM, using only one IA, may be O.K.!

Perhaps write section on "up dating" now. I am forgetting it, yet it is certainly the most crucial part of PSM! But only the more "work out" ideas in the report — (otherwise I will never finish!) But do make a copy or refs to my notes, & write expansions of various ideas. — Conjectures on what difficulties might arise, corrections etc. → 298.00

.20

.21 **PREFIX SETS**: in OSS·85 I had this scheme for L such in which I did an exhaustive search over all codes w. $\frac{\text{lc}}{\text{pc}} < T$. I conjectured that this scheme was a completely general algorithm: that all prefix codes were generatable by this mechanism. Then I found a counter-example! (which I don't remember).

.23 Anyway I must now have (in OSS) a proof of the original conjecture — & of .23.

It runs a little this: If the set of acceptable codes does represent a prefix set, there must be a way to recognize when the end of the word occurs. If the UMC used such a scheme to decide to stop, we would simulate that prefix set.

.30 But I had the idea that the constraints in the UMC were rather weaker than that specific knowledge.!

My impression that OSS 85 simply listed trials in order of size of the binary fractions. This highly prevented duplication: for instance, it was very easy to keep track of what (prefix codes) trials had already been made (& were ∴ illegal) to do or to continue)

Also, I may have been unduly interested in Complete prefix sets. (i.e. $\sum 2^{-l_i} = 1$)

(In "complete" prefix sets, it is often necessary to have lots of 1's or 0's as long as! e.g. all sets starting in "0101" are code words. — (we could cut them off after extensions of > 10 bits, say).

.40

ID

LISP .00 HW. note .12

.00 : 294.09 293.13 293.01 Lisp has Please "SideEffects". RAM is an impl. one! Hur.

Kleene's representation of recursive functions (say Prim. rec. Funct.) does not seem to use RAM! How come? It suffices to find, every time a value is used, we have to generate it ab-initio from primitive concs. This is certainly O.K. from th. pt. of what is representable by the system, but is not so good for cc or even pc! perhaps.

For non-recursively defined concs, defuns are unnecy. For recursively defined concs, Defuns may be necy in Lisp.

In My functional lang, defuns are not necy, but they make pc's to give, what believe to be, better pc's! perhaps cc's are better (functions are always realized by "inling" construction.

.12 SN on "HW." I had this idea of a computer in which the main thing occurring are moving strings around. Each string would contain refs to other strings or to "primitives". A "Primitive" is an actual HW function. We make pcs. letters of strings in Que's per use of various primitives". When Que gets too long, a dup of a primitive is made. If a Que gets too short, its HW is pressed to make copys of functions we. long Ques. The "HW." could be Field Programmable Gate Arrays. FPGA

I do have a More lengthy writeup on this Machine Type.

.20

.24 → Outline of Report (Intro)

Intro, a Summary! Tells what the problem is in terms but not complete detail: Also what the soln looks like. Global v.s. Incremental Updating. use P_0 not $2^{-R(0)}$ — but say, often $2^{-R(0)}$ is reasonable approx to P_0.

Dcrn of QA problem strings: Back thot of as problem's Soln.

Since Operation is stochastic, problem state should have occasional "errors".

Note that this CPM is meant to be a general purpose intelligent Machine. Perhaps Discuss its being able to solve most problems explainable to a human (worse explain th. nature of th. problem to th. Human)

The mechanics of th. stack operation: Various poss. forms:

1, 2, 3 ... T simple unc; input unc; the Lisp machine is regular machine code.

Advantages of Lisp formalism ① Easy to Express How ideas ② Easy to get reasonable Arrays via AE

.30 Advantages of Machine Code: ① fast ② can simulate Lisp via "Macros"

perhaps use FORTH (≈ Lisp ≈ Machine Code).

How the basic probabilistic functions are realized: How $M(S, Q, R)$ can be thot of as a univ. d.f: Eg. $f_{M(S,Q,R)} \to A$ If S, and Q are suitably fixed

domination on A, w. R as random input to th. function $M(S, Q, ())$.

How, $M(S, Q, ())$ need not be universal.

The prediction problem: usually very easy, Once Q was been updated:

Just put state of $M(S, Q, ())$ in RAM, Run thy random R, with returns to "restart", in.

299.00

.40

In general, I say that there is a complete formal system (say Lisp, but usually not in any good) acceptable sense — That is, one of: ① incremental ② TSQ's ③ Screen recursives.

.00 ; 298.40 Global update, used for early problems in TSQ.

Incremental updates is usual update methods; can be for one or several QA's.
If several, they will usually be similar problems. Sometimes they will not be: e.g. backtracking!
Explain! It is usually by cc operation. backtracking.

260.01ff on UPDATING TRICKS & variants

Before explaining Incr. updating: Explain how to derive trees from pc's.

First explain of Z1; Give unupdated ref to 5646: Discuss updating in Z1
Discuss OSL in Z1

A Z1! Say "is w/o Z1" - Give examples.

Finding common subtrees is harder than finding common subexpressions: But OSL in A Z1 is also more difficult: More pitfalls:

sol646 ① update pc; ② new defns ③ reparse ④ update pc ⑤ reparse.

Any function that has been used in past can be parsed a new op, w.r.t. pc (often).

.10 There are only a few of methods of import in updating: I have not discussed probabilistic proofs, explicitly.

While they can be dealt w. in same way as deterministic proofs, ~~but~~

It is likely that have different assertions.

This is only an outline of a few updating techniques.

That many more techniques need to be discovered!

How I expect to discover them using TSQs:

- TSQs are ① to teach TM
  - ② to teach designer how to make better TM (= better update techniques)

Induction

Mention special update techniques in various branches of

.20 See 304.24 — 305.18 for some similar topics to (packets) discuss:

It discusses TSQ's, transition from $\xrightarrow{det}$ dTM to $\xrightarrow{estoch}$ sTM — that idea of TSQ is common to as well as TM.

In discussion of Z1: perhaps give my (selected) letter to Wolff (in Z-tech → PS.)

One of big problems will be to explain Z1: ≠ its updating process —

Then explain A Z1 ≠ its updating process.

It might be easier to Explain A Z1 directly, w.o. first explaining Z1:

Do mention the necessity in which one must do (update) new QA in isolation by design:

a cheap method for it, then find a way to recognize how it differ from all previous

QA's (or "almost all" - since it can make errors - since it's stochastic OP)

.30 The — Errors are not allowed in MTM probs.

.40

CONTEXT of Concs

→ Hvr, also note 417.09: on hov "context" ≡ "condl. probty" ∴ it is ∴ what QATM is normally working on.

.00 ! .26 ff is Great Breakthru } ← This is REALLY GOOD!

.00 ; 299.40 ; I had the idea that w.o. context, pc's of concs would → ± at $\frac{1}{\sqrt{n}}$ (n = corpus size)

This is not true. Ass $n \to \infty$ pc's of concs settle down to certain values.

This are uncond. context free pc's ≡ Unconditional pc's. Since they are generated over the entire corpus, they tend to be small's & work "small" about particular

.04 situation (≡ context).

Whereas instant TM shows only uncondl. pc of concs is not both, because abs. no. of concs is small (< 20, say), so their pc's are not vanishingly large enough for achievable searches (i.e. tolerable "cjs")

Hvr. as n (i.e. corpus size) ↑, we get into trouble of .00 mode

.10 A (perhaps the most important) way to deal w this is to use not context free (≡ uncondl.) pc's for cons, but context dependent (≡ conditional) pc's.

However, one should not think of context dependence in a narrow sense (like in "context dependent" Grammars) but in the general sense of condl. pc — where the condition (≡ context) can be any thing — any info assoc w. the current (≡ conc) to be (predicted/however its pc evaluated)

.17 Now PSM is essentially a conditional plausibility evaluator (.29).

Perhaps Rather than PSM, call it Conditional Probability Machine

was CPM on old (pre dos) computer lang — I think so.

XM Utility — ~ 1956 McCarthy, = Common Collection.

CPM. CPE CPC Evaluator Computer Condl. Prob Calculator

Bayesian Nets — Add to IAS.

.20 (17) So we could put the machine to work on the problem of assigning condl pc's to concs. An impt problem is to decide what to use for "context". If we are very broad, the problem becomes very diffl. If we are very narrow, we don't get v. g. condl pc's.

Lets look at the context in the Broadest way: Say we are doing incremental updating of

.26 Also note 417.09 "context" ≡ "conditional pc"

On the previous operations, was constructed, using a certain set of (partially ?) their ? functions, their assoc. uncondl pc. & condl pc's for those functions.

The conditionals are on the present problem (Qn) & previous Oni & its assoc functs & their condl & uncondl pc's. (So a kind of (non-unique) record of dependencies. So in the broadest sense of conditions are all of previous dependancies) of previous concs, all of previous Oi's & their codings & type's of their codings.

.30 .31 .26 - .30 is, hvr, a bit clearer on just how the dependancies occur, & suggests which should be & transient, etc. [If context includes (past) Times as well as "space", we can have living down, & such.

.33 Hvr, in general .26 ff is usually too Broad to be directly useable. If the narrowing

.34 the context down enough (Actually, it need not be narrowed down": Anything can be "Q" practically

.35 I can use it as a Q for CPM. See infra at 301.33-.40 on how .29 ff is a bit Elitistic

.36 N.B. As realized by a 3 input Machine, Each O is a Function

it is easy to realize as a Monte Carlo device. Here as we construct new mixl Oni's, & Each modifn. of the operator (i.e. additional weighting function) can be given a (conditional) pc by On — so we get a Monte Carlo D.F. of possible 301.00

See 397.00-.10 for a distinct kind of Gonzen. of IAS

"context" is defined by any "ob". (≡ Quotation strings to Booleans (TF)

.40

.00: 300.40 : **On+1 trials.** Two impt. conds on this Generator: (1) T. time needed to Generate On+1 should be much less than time needed to Recognize (? — I'm not sure of this, here). (2) I'd like a way to do small (trial) modifns of O_n (conservative approach) as in attempts to get a fast Soln. ("Quick & Dirty").

→ See 382.11-.15 for some ideas on what "small modifns" could mean. This is best done when O_n is an ensemble (≡ D.F.) & our target O_{n+1} is also a D.F. Exact for being represented by "population" in GA.

.03 .04 On more trials for Constructing O_{n+1}: They can be done via deterministic

.05 Lsrch over the R's of the various Matching functions that are to be generated → (Note: 33)

7/13 60
7/31 100
18 days to run
so ~2 pr/d. not bad — but not Great!

.06 [Remember that Lsrch is "imperfect" because of "Correlns" betw. the Conds

→ Must see 302.00 on Correlns

Also Remember that Usually (almost Always), O_{n+1} will be constructed a couldy a few (say 3 or 4) functions — a Nervous Pr's ~~get~~ ~~CJS~~ get too small CJS too by

.09 .10 So: 300.36 − 301.05 (−.09) could be a ∴ final soln to the TM problem!

I do have to work out (300.35 − .39) — how to "narrow down" the broadness of the "context" described in 300.26 − .30.

→ T. "small modifns of On" is not easy to do! but see 382.11-.15 for some good ideas

→ See 302.00ff NOT BAD APPROACH ← Also involves Learning rather General (?)

During Lsrch

Well, not so final! There is still the problem of correlns of Conds in Lsrch., A probly many other impt. unsolved probs: BUT, it DOES look very Good!

Other Very impt. probs are General Updating Algms: see 260.01ff for 8 good ideas.

→ The 300.36ff is a v.g. updating Alg.! It says that On is the Building intelligence

.20 that Builds On+1: Which is the way people seem to work! → 302.10

.21 [To Teach TM how to do "problems" like 300.26 − 301.05! Give it examples with an "index" that indicates those examples are all of "the same type". — (i.e. a 'sc'). Still, it's not clear how I get these "examples"! — Just whatever they are — a few examples!

When I solve a prob in a TSQ, write down all the (concs) I can think that were needed/used in those solutions. TM must have acquired those (concs) before it can set up that problem that way!

.29 8/1/01 INDEXing: This seems Very impt! It is an easy way to break up corpus into SC's. Also, the INDICES can form a tree, so they can have varying amts of "overlap" — This would be the normal tree structure of Categorization devices — like an "Outline".

.33: The analysis of "context" of 300.20 − 301.05 is strictly EI., in the sense that its output is a D.F. over functions. It does not consider the inputs of the functions or whether this output is to be "final output" or "lower level" output.

.36 → A less "EI" approach would use the same "current info" as 300.20 − .31), but its goal is not a "useful function", but On+1. Actually, it may be easier to assemble a usable corpus for the Goal of .36 than for the goal of 300.20ff: i.e. finding a good function to continue building out the partially constructed O_n+1. But it's a very useful elm of .36.

See 302.10

8/1/01
ID

Correlations (between Cands). .00 | Looks Very (Empty Good)!!

.00: 301.00 : A possl. (Bicorrected) approach to "Correlns" in Lsrch:

1) One picks the most likely cand & works on it for Time T. It is not (yet) successful.
This fact [alone] changes P.D. on rest of cands: particularly w.r.t. cands that are "correlated" w.r.t. one that was worked on. Using this new P.D., one picks the next cand.

If one updates the P.D. periodically, this will deal w. "correlns" to some extent — This is Not BAD: it is a nice SOFT soln.
— Trouble is one has to update after every cand is (partially) processed

An easier way to deal w. this: When a cand is processed, immediately, the cands it correlates highly w. have their pc's modified. Pc of rest of cands stay the same.

.10: 301.40 & 301.20 Spec. T: way .00 fit works in updating $O_n \to O_{n+1}$: We create cands for O_{n+1} viz 301.00-.04, .09 (first read 300.36 - 301.03). Using determinized Lsrch we create a cand O_{n+1} for time T. We insert this info into O_n (as part of its "context" of 300, 20-.33 ← it tells O_n that this trial did not succeed after a time T trial.). From this info, O_n creates a PD for the next trial for O_{n+1}.
.04-.09? ← seems likely!

This sounds v.g. but it is very greedy: The system doesn't do "experimenting" — Ahead There is no "Look Ahead". Hence it could be a very smart system! — If we want "look ahead" we can get CPU to be invoked by the Dynamic Resg. aspect of the problem — Like the "Cure Cancer" problem. [See Index on this at top of Page about "CureCancer".] .20 — After it is smart enough to work on such problems.

243.00 = Cure Cancer

Actually using the updating schemes, modify & by attention to correlns — devote on .10 pt, is not always so "optimum": — Because w.r.t. "attn to correlns" (or attention) features, after O_n has generated one O_{n+1} cand, it usually can quickly find another using a modification of the "R" inputs. However, if the "Q" input is changed, finding a new O_{n+1} cand usually takes much longer.

N.B.

.37 Perhaps a fast way to take correlns into account is to let O_n generate several O_{n+1} cands, & use correln. info to ↓ pc of certain of the cands. When more than one O_{n+1} cands have been tried, the no. of cands w. lowered pc will grow much. .30 However, eventually we will double T (or equiv) & we may have to

.34 retry many of the cands that were rejected for trials of length T.

This sounds much like the approach of 378-379! Also note 379.37 for some similarity of 378-379 to 300-302

also note 385.00 391.00

In my Month at IDSIA, I developed a way to do efficient Parallel Lsrch (the best kind of Lsrch!) Using a MC method to select trials. This was for OZ search, in which it was looking for cands of high prob of Max G — which is ~ to search for Good O_{n+1} cands.

.37 The way I was done: Each Cand was for each cand, was would generate a MC value G according to its G d.p. .38 max of the G's given by the MC trials. This is repeated for a certain successive trials for Lsrch. This is "the Monte Carlo Contest"

Now, we could do something for Lsrch over O_{n+1} cands to reach, but → 303.00

.40

8/1/01

ED

Correlng betw Gones

Learning Dorny Lsrch .08

Effect of CC & <B on System

(22) 285.00-.40

.00: 302.40 : In each Mt. Carlo context of 302.37-.39, we first penalize each G by an amount that depends on its correln. w. t. ckt's already hired. Or — alternatively, there is picky & way to modify t. pd's of G's of funds, before they enter t. Mt. Carlo Context

T. fancy is simple. because t. Lsrch via Mt Carlo is a v.g. way of doing

.07 Lsrch.

.08 Also Note t. fancy technique is applicable in all of our cases of "Irng during Lsrch" which was a principal Criticism of t. true optimality of Lsrch as a prob solving method (Assuming "all info used in t. pd")

.10 Since I will be using Lsrch for control operations in CPM it is impt that it be "close to optimum" (.08-.10) plus the older arges on t. optimality of Lsrch (within a factor of 2) makes it likely that Lsrch may can, indeed, be v.g. possibly, indeed within a factor of 2" (But note what this means: Using Lsrch as I do, is no worse than using an optimum method w. an optimum ÷ 2 clock rate) Here, an optimum method at t. same clock rate could be many times faster than Lsrch — i.e. it could take much less time to get same results ← this last seems unreasonable, since changing clock speed will always ↑ soln. speed by exactly same factor! ← Go thru t. original argt about t. "factor of 2" Its a very simple argt as I remember — Also t. argt. that Optimum value of s.i. to "productive" could ↑ speed by very large amt. (.30)

T. Weakness in t. Argt. This is "unreasonable" in sequential pln. ops. Argt. — in sequentials it is not easy to divide up TM's Behavior into "SI" and "not SI".

.20

.25

.22 NB In Forgg Analysis, I haven't much considered cc limits, KB's. — Tho they Are certainly Critical in t. Operation of t. system. (I wrote a Note re this within last Month.... A possl conclusion may have been that it was not very hard to take cc into account for t. Model of CPM that I look at that pt.)

285.00-.40 Has v.g. ideas 285.38-.40 on Equil. betw pc & cc is v.g.! — I don't know what t. relation relation (variation) is, but .1 pc/T is a reasonable guess. I can use any assumed (functional form) of equil. to use it of set CPM to find optimum behavior in view of it.

.28 .29

.30 (21) T. Argt. of .08-.21 — On t. Optimality of Lsrch within a factor of 2 "if all info is in PD" ; since Lsrch is t. main activity Going on in CPM, This becomes a strong argt. for it's being very near optimal! As I construct CPM, t. specific work pgms in t. argt. may become clearer, as well as ways to deal w. these "Non-Optimalities"

NOTE, Hvr: This CPM is not truely optimum since it Doesn't Do "Lookahead"! Its "horizon" is 1. If can work out General RCPM problem (see 24300 on "Cure Cancer") perhaps I could Allow CPM to have "lookahead" within a problems of certain types. T. Big Problem of "Lookahead" is t. machine's Manipulating OSr.

.40

ID REV Of present CPM: The path I expect to take

175 x 4
50 x 4
28+7+12 = 47 4

.00 : 30340 : Summary of present state of system:

1. Steady state behavior for a (somewhat) Mature CPM (Condl. Prob'y Machine)

O_n is state of Machine operator after updating on $[Q_i, A_i]$ $i = 1 | n$.

A pd. on possl. cands for O_{n+1} is obtained by O_n w. Q input recognizing one of (Q_{n+1}, A_{n+1}) (This way also a relatively short set/or sequence of Q A's.)

Th. current def'n of O_n in terms of primitive & defined Fcns.

[For discussn of O_n's input for this updating problem: see 300.20 — 301.05
Also, a CB (common bdy) should be mentioned in relation to O_n —]

Lsrch is done on Phase O_{n+1} cands, at cost of 289.33 is unclear what we use Lsrch to try to Maximize.

.10 See 302.00–303.20 for discussn of how Lsrch should be modified to deal w. "correlns betw. cands". The end of this discn. is on optimality of Lsrch (...if all info is int. P.D.'d)

O_n has, as input, a Q_2: Insane prog'ing code It has an auxiliary input R (Random)

.12 Random input on "R" gives a Def'n of output (defined to be output when machine stops)

.13 (.12–.13) tells how the system answers Questions, Q.

2. On the nature of the Q,A's: Any "learnable" / recursive / processable relation is usable (I don't mean "learnable" in the sense of Computational Learning Theory of relation etc.). So Questions & Answers, problems & solns; Any I/O relation that is defined (or that is to be defined) by a finite string or a finite string plus a finite no. of continuous params. variables).

.20 So we can use CPMs for curve fitting, linear/some linear Regression. It is expected that it will also be able to answer Q's about a data base — but I haven't worked out much detail on how it would do this.

.24 3. "Initialization" aka The P&P to "steady state":

We start out w. MTM probs (Deterministic Q's) from Algebra, Math TSQ's.

Initially $O_0 \equiv \Lambda$ & we do a search over Deterministic functions, using AZ1 to assign pc's to cands. This is for very short TSQ's. We want a 0 w. max pc, max prior responses for all Q's.

The initial pd's on sub-Functions to create O_i's will be uncondl pd's (essentially, a same pd for all situations). When the no. of tasks is small, this is a tolerable system, tho it is slow.

.30 If is necy to get condl pd's for each primitive or defined function — the (context / conditions) being some of those discussed in 300.26–.33.

.31 Another trick is to discuss how O_{n+1} in the basis of Min modifn of O_n

.34 (B) Min no. of QA's have to be verified on the proposed O_{n+1}.

I expect to learn how to do .31–.34 by studying how humans learn the simple TSQ's CPM will be initialized on. That we will use for CPM's initial tape. The method of 307.17–.19 may be very common.

Essence of "Falsifiability": To propose theories (models), that are easy to test. To propose theories (models) that are easy to test.

= inexpensive/cheap

REV

3 to

.00: 304.01: The transition from DTM to STM: (performance, stochastic)

When we want probabilistic answers: We make the index "S" (stochastic) appear as a Q that need stochastic answer (i.e. P.D over A's)

We start out w. ~~problems~~ deterministic problems w. occasional random errors.

2/0 det. probs — but w. several equally good answers.

Next, we may try Bern seqs. (Just Binary, Prin of symbol) Than ZL

Try learning curve fitting: (First Linear then non-linear: It could learn w zero error curve fit

first, then learn MS error curve fitting then other error criteria.

Then HMM's: See ~~3111.40~~ 310.03-311.40 for a very nice way to make TSQ's for NMTM, & how to pack

.08 TM to Learn Them (Chapter being built up on MTM problems!)

4. Sub. Corpl Indexing of Q's: (301.21, 29 eg. Very Often Q's will be given one or more

"indices" that tell "what kind of problem" is being given. Some indices correspond to

.10 "Context" for a Human! (It could relate to where the problem came from, what areas of science, when

the problem was given, what other problems is it "similar" to.

As w.r.t. other aspects of Q's, CPM is not told what they mean: It has to be "induced"

from the data.

~~The elements~~ of a sequential TS could have a common index plus a "Time" index

Alternatively a Q could consist several "synchronized (time locked) time series, i.e.

problem could be a regression problem: to find an eq. that probabilistically relates ~~one~~

.18 ~~the~~ elements of one of the seqns to previous data in all of the seqns.

.20 A Great Breakthrough of 310.33ff is a way to do NMTM very nicely!

.30

.40

8/2/01
ID

Gen notes:

.00

(1) On QM: T. accuracy of QM has been mainly tested w.r.t. the positions of peaks & zeroes of diffraction patterns. I think it more likely that the exact shape of the patterns between the peaks is likely to be wrong!

(2) I have been using a "UMRE" in CPM to replace implement the Comp Prog DoF. While this may give better pc's it can be very poor for EC. Lisp users & RAM's via its "side effects" ID 293.00, so it may be potentially exceeded by our "seq varieé" machine. T. idea of most Intro proc. Copy part of spent is max Q. So alternately, I want to get some result from TM w. min cc — & this probly involves modifn of H.W. A machine that redesigns itself using FPGA's (Field Programbl Gate Arrays) — info in GA's, would probly be v. g. (see ID 298.12)

.10

(3) I may want to write up a detailed Augt. about 6:/ Factor of 2 of optimum "If all info is in PD". Mainly to make it clear just what assumptions I'm making. see ID 303.08 — .20 — Improvements (303.20-22)R
There is an apparent flaw in the Augt. that I need to address; i.e. 303.18-.20

(4) In the Hessian of $\vec{F}\,\vec{G} \in \vec{H}$! : Rank is the no. of non-zero eigenvalues
so $|\vec{A} - \vec{I}\lambda|$ must be expressable as "$\lambda - A$" ← Tri-ached Hessian is closer to $\sum_{i=1}^{N} \frac{1}{\gamma_i} x_i^2$
a b c

.20

(5) {Z-1 & Lap's Rule as continuous coding} Say we have a Binary Bern seq. w. $p(0) = p \cdot \{p(1) = q;\ p+q = 1\}$
Using Z-1, the pc of the seq. of n_0 0's & n_1 1's is

$$\frac{(z-1)!\ n_0!\ n_1!}{(n_0+n_1+z-1)!}$$

[for an alphabet of β symbols: $\frac{\prod n_i!\ (\beta-1)!}{(\sum n_i + \beta - 1)!}$]

To code Bin. Bern — v. whole seq. — takes a pc = $\left(p^p \cdot q^q\right)^{n_0+n_1} \cdot \sqrt{\frac{p \cdot q}{n_0+n_1}}$
we code p & its "width" is $\sqrt{\frac{pq}{n_0+n_1}}$ so that's a fine pc of describing p.

$$\frac{n_0!\ n_1!}{(n_0+n_1)!\cdot(n_0+n_1+1)} = \left(\frac{n_0}{n_0+n_0}\right)^{n_0}\left(\frac{n_1}{n_0+n_1}\right)^{n_1} \cdot \sqrt{\frac{n_0 \cdot n_1\,\pi}{n_0+n_1}} \cdot \frac{1}{(n_0+n_1+1)}$$

$$= \left(p^p + q^q\right)^{n_0+n_1} \cdot \sqrt{\frac{n_0 \cdot n_1}{n_0+n_1}} \cdot \frac{\sqrt{\pi}}{(n_0+n_1+\pi)}$$

.30

So a 2 ways of doing it agree except for the flaw (n_0+n_1+1)

So, Doing the whole sequence by p & q & width of p knowledge takes as $\frac{n_0!\ n_1!}{(n_1+n_2)!}$ as opposed to $\frac{n_0!\ n_1!}{(n_0+n_0)!}\ \frac{z-1!}{(n_0+n_1+1)!}$ for $z=1$!

.38

AH $p = \frac{n_0!}{n_1+n_2}$? : perhaps just take $\int_0^1 dp\ p^{n_0}(1-p)^{n_1} = \frac{n_0!\ n_1!}{(n_0+n_1+1)!}$

Which Gets more answer

There are 11 codes for discrete seq. & one exactly wtr for uniform applied. SREC 307.25
That it should be identical to the sequential coding result, is 307.00
Surprising!

.40

9/2/01

ID

Gauntlets

.00:306.40 : (6) In Z1 : T. method I described to Wolff! I did have formula for pc of corpus when one gram was defined. I then had to decide on a specific parsing of t-corpus & use that for next decision that. This specific parsing gives rise to specific Case counts for all symbols & resultant pc's for those symbols.

In AZ1 I'd again, have to decide on a particular parsing: perhaps not

The O's are created (along w. their parsings). This gives pc's to all component substructures.

So, any sub-tree (trivial) could be treated as a segmented component & Essentially OSL.

.08: 280.09 : Noted in Z1 & AZ1 t. problem natures of t. problems are quite different. In Z1, we have t. corpus subset of primitives, but now is defined must be made & t. corpus must be parsed wrt. them.

In AZ1, we have created t. O_i & we know its unique parsing. To create O_{n+1} we look for common subtrees of O_n: Since only a little has been modified in O_n (from O_{n-1}) we don't have many new subtrees to check for repetitions (most subtrees have already been checked for repetition on previous O_i constructions.)

.17 [(7) It may be that it is very common to solve a new problem w. a specall purely Bum Find way to recognize t. latest Pam. Do this many times than look at step of

.19 Solns: simplify Pam : Anyway, This avoids having to test t. whole corpus. verify

.20 (8) Certain cases. involving MRI, Logic, communication w. USR C's. I say a term which problems is expressed) can have special treatment, specialindexing — To denote They over of much wt. — perhaps equivt. to very hy ≤ 32. Therefore Certain Phrases would want TM to balance w. P close to I.]

One of t. indices (sc's) could be an "Advice Channel".

.24: 306.40 (9) Actually, t. 306. 3g-40 result is trivial! Each conditional pc $= \frac{n_0+1}{n_0+n_1+2}$ is obtained by taking ratios of pc's of $\frac{\text{augmented corpus}}{\text{unaugmented corpus}}$. In soft, product of these, all but 4 cancels

with pc of t. final corpus!

I think this result is close to Lemma 2 of S78T3; which is t. Guts of t. Thm) Doug Campbell may have been sort of "trivial" — it may have been "trivial" — combinatorics. complicated

$\prod_{i=1}^{n} \frac{p_i}{p_i'} = \frac{p_n}{p_m'}$ updates easily from n to n+1: That t. updates is also "easy" for Expected values of Shannon quantities is t. Big Q.

Say $b_i \equiv \ln p_i \quad b_i' \equiv \ln p_i'$

so $\sum (b_i - b_i') = B_i - B_i'$. Does this imply $E \leq \sum_i E(b_i - b_i') = E(B_i - B_i')$? "is simplicial by" $= \sum_i^E (b_i - b_i')$

Normalized unnormalized.

Also, is it true or duply of whether variable b_i is a measure or some measure? In particular, I suspect that E doesn't have to be t. same as $e^{\pm z b_i}$ — Thm is then barely optimal. The only part that requires E to be wrt. t. true p" $\doteq (e^{\pm z b''})$ is t. Thm that t. K-L distance is always ≥ 0.

8/3/01

I>

308

38 pp w. new pagn

At start of ≈ 309 I added me to Σ: ALVIN (1) Pan.

Grav notes

What people mean by "Real world" is "Yesterday's Theory".

(10) ! The idea in "Sci"/"Philosophy" that "There is a Real World out-there"!

I think Big means that we will continue to get data in accord w. our old-models present "intuitive" models of the world. These "intuitive models" are, necessarily nothing more than "Yesterday's theories": ∴ That there is a RW out there" ≡ T. would continues to act in accord w. Yesterday's theories". — This is the idea of

Hogan? (who wrote a book on "The end of Physics") that we will soon hear the

end of serious modifn. of physical laws (≈ "probly zero" from my pt. of view)

[for Serious Physicists Rules of capitol. to Proprietor of cafe comm'ing a

With his little ball announcing "Closing time Gentlemen — The party is over" —

A nice way to say this! / Hans Fantel (?) US & Rosiva fine painters; At a

certain pt/ her would get to his piano and begin playing loud, E.G. dark music.

in the celebration!

We knew then that/it was time to go home — the party was over.

It was time to go home — the party was over.

When — talks of the end of physics — I hear my loud piano music in my head.

but I cannot because this is Hans is long dead and it cannot be heard

that the party is over.

When Hogan talks about "the end of Physics" — I heard melodies in my head — but

I find Hogan's argument, not at all compelling, Their music is

not very strident I assume not to be over

the party will

for a long, long, time.

(11) (SN) On STOC Turbo radix 1 t. analog of Lemma 1!

This appears to be the hard part of the Theorem! The Lemma by itself, is stronger than

the Thm.!

Anyway's possible way to prove Lemma 1: Say we have k components of PC!

$\sum_{j=1}^{k} x_i^j = 1$ The diff. betw. the 2 expressions is zero if $x_i - x_i' = 0$ ($i = 1(1)k$)

(Probly) its partial drivs. $= 0$ in all directions, along that line.

T. Lemma is $f() = \sum_{i=1}^{k} p_i \ln \frac{p_i}{p_i'} \geq \sum (p_i - p_i')^2 \geq 0$ $\sum p_i = \sum p_i' = 1$. Anyway a proof might be possible along lines

of Gacs & Lovász-Vi 1990 p 329; first ff after eq. (5.6)

$p_i = p_i'$ $i = 1/k$ is the common constraints. $\sum p_i = \sum p_i' = 1$ gives the constraints on 2 levels.

So $k-2$ dim space. : T. value of $f()$ in that space is 0, next show by partial drivs of $f()$

over all zero on this sp. Next show second drivs all > 0 in that space — so that t.

second drivs are perhaps > 0 in a larger space; but that first drivs are > 0 on our

region, but < 0 in another. These are regions divided by the manifold $x_i - x_i' = 0$.

ID **EBL** (cont)

.00: 297.08 : I had postulated that any Lay'd "Explanation" is a "compression" & all compression can be used in induction. Compressions (perhaps) "Recoding" or just "Coding". Does this nearly imply induction? If a coding implies ways to extend the code, then it is inductive. E.G. In many Models of the data, we have certain fixed, certain random params. The implication is that this coding will continue on the future in the same way. If a coding is by showing that the data is an example of a general rule (may be all this "Explanation (or coding) by Parsing"?), If it's an old rule, we update params (SSZ) of the rule. If it's a new rule, we put it in our "dictionary" or "rule set" w. its initial params & Tomorrow it's rather we may have to reparse the entire corpus. (see any point of (Maximization)).

.10 Reparsing can be a big job. In theory nothing & Scientist will have to "Think" all of his & Scientific experience relative to the "New Paradigm".

One trouble w. Kolmogorov Complexity's "Execution Complexity": It does give codes for corpus & its extensions, but no clear way to extend the codes — since machine stops after each code.

Well, perhaps not so: Any method used to code a corpus implies that any method could be used for other corpi. E.G. the BDS method of coding "Non-linear, Arb", can be used for any finite corpus. Here, it's non-incremental (never mind) & there to use for prediction — But, in theory it can be used for predictions.

I have used "continuous" Universal D.P. Then once has appended codes for the corpus — any continuations of the code give predictions w. proper prob's.

.20 Here, in "discrete" Univ D.P. we have Solc. Decl. coders. There is no way implied to extend the code. — Unless we regard the code as an example of a "BAG" (true — an implied stock (e.g. à la Instance & ass) of the lang. output. In this case, the rules used in the derivs, give part of the Grammar — & the Grammar can be used for induction.

IN GENERAL: It may be that Each "method of Coding" has its own associated method of "Generation" of "Induction". If it does not, it's hard to see any "Value" in it! → But see (38) ↓!

.30 Explanation ≡ UPDATING : To the extent that this is true, Explan. can involve discovery of new rules. "Updating" means Modifn (usually incremental, but it could be Global) of the predictive structures.

Any compression/coding of the data (even a raw, uncoded data) "implies" a "soft" future compression.

Even A random seq. imposes a non-uniform d.f. on the future.

.36 SN A long (Kolmogorov) random seq. for it to be followed by (01)^7 is now much less likely than it the initial seq. had not been "random"? increases however.

.38 U.S. Placebo Explan: The "Unexplained" gives peoples feeling of uncertainty, ominous doom, nameless Dread

Functional A functional explain (e.g. understood) can heal w. this feeling — But a non-functional "placebo" explanation, seems understanding & will often fix the "Unexplained" feeling.

—1 4/1

The New Predictive utility of an explanation is its "Placebo Power" (309.00 spec)

8/6/01
ID

Modified Universal Stock Operator .03 → Looks very Important

.00 : 309.40 (Placebo Explanations) : Large Industry built on supplying Placebo Explanations ("Industry") (Philosophy) (Religions)
Also part of industries: Things that don't need explanation, but seem to: A Placebo explanation will be supplied by the "Industries". (Also News Paper Headlines: "Market goes (down / up) on news of (merger / no-merger / Hostile Merger / economy inspite of ↓ in interest rates)

.02 → 305.08 On the "universal P.D." used for CPM! I had in mind Mix 3 input machines since I knew it is universal in some way I needed to be. It is not clear, however, how one can easily express other IA's in it. So what I'd like idea of finding a benchmark IA's known to be useful & unbiased. Grammar that expresses them as ass's seems likely Very Good! V.G. T. Grammar is likely to be universal if I do enough IA's.
This is a rocky start, in the direction of solving a serious problem! i.e.

.10 I didn't have a good way to express IA's that I'd normally use, in the "Input Mix" format.

.12 Say I had a list of IA's (as I do) & (one / some) of them (at least) (were / was) universal. To start off, each IA has an "unconditional" pc of being chosen for an application. Later, I'd want these pc's to become "conditional" (on the Q) — which is what f(C) was for — f(C) looked at a problem (& now, it would also look at its "context" — the one could simply record "problem + context" as the entire problem-item.)
So it could well be that my "factoring" of the system into f() & the set of IA's may be a not-bad Elegant of a problem that was very diff. in "Non EL" form.

.20 Now, now, I think I'd go beyond having the IA's be ass's out of a IA Grammar. I think I'd want a [IA] in which each of the IA's used a subset of general rules that could describe any reqts in a (probable / estimated) delay. I'd have my largest set of cases used in starch, prodn. — each of the IA's would pick a subset of those cases or combos, so seem to do prodn. — This seems like a better use of Genes than having a "IA Grammar".

Grammar?

I.E. "f(C)" or its equivalent would construct "IA" bits from the Grammar.
A big Q is how "Updating" is done. Do we simply update the Grammar & its continuous & discrete params or

I think the for-88 approach preserves "Universality" in the sense that any describable pd. can be expressed by a "finite string + continuous params". → 324.30

.30 In particular, when I use various IA's in solving the TSP — I will be able to use the things ideas to enable CPM to discover & solve those probs

.33 in some way I think I do. → (space) 311.00

.34 Any possy of starting out on SM data? (SM DATA) Well now I know roughly what kinds of data to use, but the (needed) forms of the reqts are not certain to me. — For this reason, I don't see that I'd be able to test CPM anything viz a TSP using modern data. I.E. I could get very old data (it is likely to have simpler reqts to learn from). Conceivably I might use "pre-figure" analysis. — Try to find most popular books on this. → (space) 311.00

.40

8/7/01
ID

-.00: 31040: On Learnable P.P.s (cont'd): In TM's general knowledge about P.D.'s: When given a P.D. in ztsq, (w. proper index, so TM knows a probabilistic soln is needed), it would I will include a set of functional forms, & computational methods, that it could use to solve those problems.

I will be periodically, (tempus es tempus — from time to time) be putting in new probabilistic methods, — I should try to put them in in "packed form" so TM could see similarities (analogies maybe) between solns, — & use structural elements to derive new forms of stochastic algms. (These new forms would be constructed mainly when lots of CC was available for prob. solving).

-.10 Just as an Algebraic TSQ gives TM "basic" operations like + − * x ÷ ; 1, −1, x^2, x^3, x^{-1}, x^{-2}, sin, cos, sin⁻¹, cos⁻¹, e^x, ln x, etc., along w their derivs in terms of more basic operations (we may want TM to know a basic complex variable early on in its TSQ). — TM will have $\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}$, erf( ), xe^{-x}, $x^n e^{-x}$ & other useful P.D's available for it. For form eq. fitting, it will have linear & n.l. eq. solving methods — including methods to check for errors in computation — as well as methods to correct for them.

It could have access to ANN, RANN, GA or any other such methods — but these involve, usually enormous CC allocations — so I'd want to have

-.20 Special ways to train TM to use them.

So MTM probs can what TM starts out with. Gore is then to find model of w. max pc (pc of data in terms of the model always = 1)

When NMTM probs are introduced, TM has a larger & different set of concs. for use in attempts to solve problems & its general methods) like the same as before but the Gore is a bit different — the unit previous Gore by pc of corpus w.r.t. model.

-.30 So starting out w. MTM is best bet, since going to NMTM involves very little change.

8/12/01 ↑ This could be because 310.12-33 involves very context dependent conc. selection! In fact, 310.12-33 is mainly talking about NMTM, rather than MTM! → like 300.26 ff

-.32:

8/7/01

312

ID

P313 may not be worth it

.00:311.40 : On TM's ever becoming Human(ish): TM will clearly not learn Things in t. order that Humans do — but maybe they will not be an unsurpassable bottleneck!

As an example, I know several people who ~~seem to~~ act (cracked) like certain parts of their social education were missing! Mo, Alice, Joy Tyger, Murry D, Charlotte Mowery, Mul. — but most of them learned in later life to pick up the tips — sometimes in a catastrophic conscious, cognitive way, via Rec Mail subscriptions — which is the way most people implement the Human Image.

So, similarly, TM may (thru fuzzy "clearances") be able to get as

.10 **their it had** a gerrock-ish "social understanding" in the way Humans do via [redacted] Nets.

So: [Present Model of [redacted] CPM:]

.11

.12 1) Use of Lisp-like, Forth, AZ1 cart. as ≈ deterministic loops for early MTM type Problems

2) Use of ≈ AZ1 type of PC assignment to trie/s using t. langs (2) & (1).

.17 3) Use of uncondl. PC [redacted] of cones per early TSQ, smallssz.

4) Use of condl " " " " later " largessz.

.30 (large ssz needed for corpus of "contexts".

5) Parallel [redacted] development of NMTM's in ways ~ to .12 but use of various stochastic assign 310.03ff. We need NMTM's type problems scaling capability in order to find cond. PC of cones in .17

6) T. general problem of [redacted] updates! Out, as a function of Out ≈ (Out, Ant+1) ;....

≈ other kinds of "contexts" in the is discussed in 300.00 ff 300.36 — 302.21 of PC's of Cones

impl. Breakthru: Post this is most General t. loop of Context for Context dependence

:28 so "On builds One" ← This is many, many-col. TM ideas! T. main "weakness" is no "Lookahead" (i.e. t. "curse concave problem") But it can be a main concern of t. workaround most problem!

7) Correctly better cones on Lisp-ish! A reasonable approach is 302.00 — 303.21

.30 → 8) On an overall Gore [involving CC]: 303.22 — [redacted].

9) 304.01 — 305.18 describe a final CPM, but w.o. t. ideas for stochastic models, of 310.33ff

Very impt. development!

6) is, essentially, t. Goal, & t. rest of t. pts. both & other some ideas on how to get to that pt.

A possl serious Q is: How far can we go w/o. "Lookahead"? (.28). If it turns out that some "lookahead" is needed for many problem areas. — Well, we'll have to put it in!

28/8/01 TM;

Grammar Discovery : This May be an adequate & GOOD (practical) method for it. Critical Points: 3/4.20 3/4.30 3/5.13 -24 (Gone) 3/5.30 ← Hillclimbing.

VERY GENERAL! 3/4 * P3/3 may not exist

3/4.30 & 3.13 -24 are rather General for any kind of Grammar

.00 ! Another Slow way of Getting Grammars,

Try to find a set of words $[A_i]$ & $[B_j]$ > to pc of declining Power times

to pc saved in using Rule, is max,

The pc gain of using Rule is via to Cartesian product $\Sigma[A_i B_j]$

The idea is that if A_i occurs, it is more likely to be followed by ΣB_j member. Each time a pruned such ngrams is deleted, use % pc of corpus,

Other Multiplications, & Boolean Additions of Ngrams Create a CFG.

.10 Each such Grammar rule (Multiply &/or boolean Addition) must be % pc of corpus.

Also, after various Grammarless (Mult & BooleanAdds) are found, we may be able to % pc of code by folding or deleting elements from various ngrams.

The Parsing can be adapted to other by use of Grammar Rules

Context Dependent recursive.

like Context dep. Substitution, maybe "Chomsky" etc.

.20 ((Superficially, this looks like an outline for a complete set of word Grammar discovery programs : IS IT Really?

Well, our Query is : Is it good enough to discover enough ngrams to "Get off to Grade?", If they work - but perhaps only w. large corpus ?

printed file from Ghostscript?

The "size of corpus" needed, is related to how many cases of $A_i B_j$ (concats) are in the corpus. If we have 3 ngrams (A)(B)(C) Ran we infer more passes of instances of concats — if we have enough concat rules in the Grammar,

.30 .31 ((One could study various stock Grammars — see how large corpus must be before contain "instances" of the specific Grammar rule appear in Corpus,))

Using .30 - .31 — Make up various simple stock Grammars, w. various pc params, Get Recursive Frequencies of various concats of pruned elements (eg. "leaves")

Say one has discovered many word pair concats w. unusually hi frequency. If ssz. is large enough, one can save out most of corpus — look! A LOT

8/8/01 7 PM

3/5

Grammar Play

:00: By picking out a pair of words, associating pc's to members of a sample for concat: So if M & N are sets of 2 sets (e.g. digrams) one would need M+N concat pc's: using 2 sets only M+N+1 (grand) are needed.

One can contrast total pc's needed to distrib. corpus — using phrase keys. That "info" needed to distrib. corpus using 2 sets of continuous params, is distrib in ID250.09–.19 $\int\int\int p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \, dp_1 dp_2 dp_3 \quad (\sum p_i = 1)$ this an integral over all of pspace of params, i.e.

:10 Background is a pc of a corpus unit Markov model (= set of params),

consider 2 sets of M=2, N=3, so 2×3=6 vs. 2+3+1=6

So no gain! b 4×5=20 vs 4+5×1=10 is big difference.

:15 in case of M=2, N=3, 2×3 params, Integrand $\prod_{i=1}^{2} \prod_{j=1}^{3} p_{ij}^{n_{ij}}$ w. constraint $\sum_i \sum_j p_{ij} = 1$ (?), w. pc of corpus.

:16 for 2+3+1 params: M M & concat poss = pc. Integrand is $\prod_{i=1}^{2} \prod_{j=1}^{3} (p_i p_j p_c)^{n_{ij}}$ $\sum_{i=1}^{2} p_i = \sum_{j=1}^{3} p_j = 1$

What is constraint on pc? Maybe just $0 \le p_c \le 1$.

:20 or, w. pure Bernoulli parameter constraint $\sum p_i = 1$. 2+3 params = freqs of 5 symbols $\prod_{i=1}^{5} p_i^{n_i}$, $\sum p_i = 1$

:21 :14 $\sum_{i=1}^{2} p_i = \sum_{j=1}^{3} p_j = 1$ may mean that the intervals can be expressed as 2 factors.

It may be possible to devise a Z.141-like formalism to "compute" the ζ's of .13 to .21 [I think Proof would be of form "THE U. Benificence of GOD". $\frac{(B-1)! \prod n_i!}{(B+\sum n_i)!}$ or products of these]

:30 Even if the Gaps of .13 ff are correct, a big problem is to decide which words to perm which groups. What is needed is [a formula telling how much pc of one gets by adding or removing a word from a set.]

This is like (a MY Letter to Wolff about Z.141.

8/9/01

270.27 first pen 46 IO pp.
Second Advice } started.
Tech Note 0.1

316

Grammar Discovery.

.00; 315.40: The methods of 314.30, 315.13-.24 are explicitly seeking for discovery of Nguists. Once one has a adequate no. of them, it becomes possible to discover other Grammar rules of almost any type! (i.e. Nguist- (≡ Sub language) accepts in principle perhaps almost all (if not all) Grammar types that have been ever proposed!

Technique of the Technique.

Basic idea of Grammar rules other than $N_3 \rightarrow N_1 ? N_2$, Nguists will except that do not correspond to NT's.

.10 (Non Terminals). — So we will "discover" many "Nguists" that do not correspond to NT's in the "Target Grammar". A BIG problem is to find rules that pick a small subset of these Nguists, & produce a good Grammar from them.

One Mile process the Corpus incrementally, starting w. a small corpus Gathering all useful Nguists, — then ↑ corpus size, Grammar Nguists, etc. This way, the no. of Nguists ↑ w. corpus size. The Nguists that are created earliest, are perhaps most likely to be the ones usable for

.20 more complex Grammar rules.

The first, the Nguists are created in pairs: (Any last, (≡ terminal) has is a legit Nguist).

Anyway, once we have a few Nguists, we can like A, B, C (A, B, C also all Nguists) has to try various/all rules like $A \leftarrow A \cdot B$ or $A \leftarrow A \cdot C$ (C is a last & is also a Nguist); or $C \leftarrow AB$; where C has been found to be equal wi $C \cdot D$, etc.

In general, to start off, the only way to get Nguists is by using $C \leftarrow A \cdot B$.

.30 (A and or B can be (arrays)). — The concat. of 2 (array pairs) a new ≡ Nguist.

Also, since early copys will be ufms: The first candidate outputs will be $\exists b \ni a \cdot b$ is an ufm. We can then try $[b_i] \rightarrow [b_j]^c$ or $[b_i] \rightarrow c \hat{} [b_j]$.

Here, the size of the Nguist generated, will (usually) be an ↑ function of size (≡ corpus size)

8/9/01
ID

Grammar Discovery.

:00 : One way to usefully study Grammar discovery techniques like the ones I'm trying:
Pick a small family of Grammars, (say Rules are < 4 grammar rules
Pick one of the family & use it to generate a corpus of size "SZ".
See how small SZ can be & yet one can identify the Grammar.
Increase size of family of Grammars. See how SZ must grow to enable
identification, using a certain set of heuristics and Rules (like the ones I've
outlined in earlier discussions.

What kinds of Grammar pairs are hard to differentiate, using these techniques?
:10 If 2 Grammars are hard to distinguish between. — Is prediction, using either
Grammar, about the same? If so, this is a big criticism.

However, if the predictions are much different, one should be able to use this fact
to get better versions between the Grammars. [Good prediction is usually equivalent
to good "updating" — Updating means both Q & A are given in all cases.
Ideally, one should be able to tell how good a grammar is by just a Q A
set, w/o having to "predict"]

:20 One may one heuristic or set of Heuristics (like the ones) for Grammar discovery.
One has a stack Grammar that generates (usually small) Grammars — & each
Grammar generates its corpus. The Heuristic rules are applied, & we see which
Grammars & rules have trouble w. This is used as feedback to USER (ME) to
:26 generate newer, better Heuristics.

General Remark: Items 319.00 — 319.26 is a very primitive, very General approach to
Grammar discovery — usable w. practically any type of Grammar.

:30 For each Grammar type, there may be certain idioms (words)
that occur w. unusually high frequency, even w. small (SSZ). We should look for these
to discriminate between various Grammar Types.

8/9/01 ±0

318

REV : .00 also note .22 : list of earlier Reviews of Report-outlines &c

Perhaps rename Stump of GS view / Ghostscript

.00 : Another run at an outline of the "Report".

1) Describe problem data, $[Q_i, A_i]$ w. some detail — Give Examples of different kinds of $[Q_i A_i]$ subcorpi — & Maybe discuss "Meaning" & "TM" "known" what kind of "Q"ity.

2) Desc. $M(O_w, Q_i, R) \to A_i$: Discuss prelim sets &c for O_w, Q_i.

Stopping Condition for R.

3) Discuss Gore. If > 1 O_i 's v.g., we will research Rumsfeld — "for Theory Revision" & "Updating".

4) Operation &c for prediction

5) "Steady State" operation for Advanced System,

.10 Updating as "normal problem" : either for O_n or "Anytime Problem".

.11 6) Initialization : a) Early TSQs.

b) Early Updating Methods : Difficult for Each A_i, but General Simulated Annealing

Use of Leech : & "correcting" problems How solved by "drug during Leech"

Wts. Vocabulary & Distance ; Vocabulary in RAM computer ANN wts. of solved problems Populations of GA solns.

7) Later TSQs. — perhaps non-Greedy Leech

— "Cancer" "Cure Cancer" problems

.17 Very General Theorem for USR

8) Any problem in & Operation can be considered to be a QA problem : either Given to O to solve, or — prelim Solns can be derived by user & user can try to find TSQ that would lead to that Soln.

.19

.20

.21

.22 Look at ● early writeups on this Report : See if I can use any of the material.

312.11 304.00 – 305.40 is latest on most advanced form ; (Query Series : included many examples. "Breakthrus")

v160 was first start of QA idea

Before that was § 289.09 mixed in ; § 294.10 formal QA date § 295.00 Tiparda operator

② 298.21 – 299.30 Again an Outline of & report.

③ 304.00 – 305.40 (Very imp.)

3.5 312.11 3.75 318.00 – .19

④ 259.08 – .14, .22 – .23 & early outline of & Report

⑤ 208.00 ↑ 164.00 ← v.p. previous revision

⑥ 164.00 first — Seen 270 – 28 → 271.19 for criticism, discn. I now have ways to derive principal doctrines.

.30 if ⓐ now universality ⓑ not doing Long during Leech ∴ not being able to realize all theories,

true $O_n(O_n, Q_n A_n \cdots) \to O_{n+1}$ model process also in & 164.00 model but still, has no "Leach ahead" ($\in R TM_h$)

T. 164.00 – 169.90 system ; w. $f(\vec{w}) \in [IA_i]$ was a Factorization of & many Non-EI " system of 312.11

($O_{n+1} = O_n(O_n, Q_n A_n, \overset{\text{context}}{\cdots})$) — So a good idea for a path to take to a more Non-ET system.

Also & Non-EI System can give a Unified Gore to & EI approach of, &.

8/10/01

ID

243.00 : Core Concepts.

~.00: : ~ 150.00 I wrote about "Breakthroughs" since 5-89. There have been many "breakthroughs" in last month: A summary: 312.11–.40 may do this! ← I may want to do this in more detail.

1. a) Use of 3 input unc. (or 1 input w. 2 prefix set-formalisms) to realize Universal Operator Def. (253.37) → 259.40 $\Sigma_{\text{on universality}}$

b) My old model of 245.09–17; 253.37 (~~253~~ 253.21) 267.00 ← counter-examples (>surviving)

2. $O_n \cdot (O_n, Q_n, A_n + \cdots \cdot) \to O_{n+1}$: Very General, Form for TM 254.16–.19 ←
(5 discover later)

That they Considered + must General Forms of Concept possl.

8/10/01
EP

List of Reviews .28

.00 : A new start-of outline of report:

1) Verb Qth. problems, showing how it all probs are expressible in Pq form

2) Goal: Given $\Sigma \cdot Bay\,[Q_i, A_i]$ i=1/n ; new Q_{n+1} ; what is good D.f. for A_{n+1}?

.03 3) We already know formal soln, theoretical soln. Give 2 input universal structure.

Machine soln. : If we had short codes (Given Gone) prompt for A_{n+1} we'll be up.

4) As is, prob. is too difficult, expected cc is too large to find short codes!

We simplify problem: Given O_n (and a "good" Pure On is a reasonable soln. for $\Sigma\,[Q_i, A_i]$ i=1/n, and given (Q_{n+1}, Ans_{n+1}), How to compute O_{n+1}? — (How to modify On). This is the incremental updating problem

→ Better! Given A & Pd. for O_n how to get a Pd for O_{n+1}, given Q_{n+1}, A_{n+1}?

.10 5) Again a v.g. formal soln is known. We can obtain a good D.f. for Guesses as to O_{n+1} by recursively O_n, $[Q_i, A_i]_{i=1,n}$, $[Q_{n+1}, Ans_{n+1}]$ as the "Q" input for O_n ....

The output will be a required D.f. of Q_n. We can test the members of highest pc, since Bayes updater is int. formal etc version( ) to obtain best one(s) as O_{n+1}. We are using O_n to update itself.

(Construction of / The Construction of)

.16 6) The machine described in 5) is one of our long term goals. Given a suitable, not very interactive O_1, put a suitable learning sequence, $[QA]_{1,n}$, we could, in theory, eventually evolve a certain O_n that would be quite effective. Almost certain minimum amount of intelligence in O_1 is required

.20 (is needed in O_1 as well as) a suitable $\Sigma\, Q_i [QA]_{i,n}$

.22 7) We will describe some preliminary ideas on how to realize O_n — not necessarily by doing (6) directly. Roughly following (sometimes only approximately) the plan of (6)

So now, how to fit the various "partial Tao's" and "hows" together to approximate (.16)

I could review parts of 318.21 -.40 to see how!

.24 (168-169) (270, 271)
208
259 ← 3 input unc.

.30 264
298
304-305
312
318 318.11-.19 of some interp.
320
327.19 - 328.23 330.40
331.00 - .11

ID

.00 : Qu't. problem of 320.22 : It seems that there is a tradeoff betw. t. Sophistication used in O_1 & t. kind of "teaching" in t. early TS Q's : Pgm

.02 (subject to some lower bnd. of intelligence) we can always put enough in to it in t. early

.03 TS Q's to bring O_1 up to an adequate "initial level". (.31)

.04 What is not clear in my mind, ~~either~~ was how I expected t. initial TM to operate. If had all of these "indexed" TS Q types, a TM would learn each kind of sub-corpus separately — as if it were a bunch of indep TM's. Each index type would have a known IA assigned to it. Then as a new problem, I could have TM learn to assign indices to problems — using ex. index, IA pairs of each, & perhaps some new ones.

.10 Some: Make a listing of "Initialization Techniques" — refs to 320.10 : refs to (5)

.12 1) a) (larger ?) set of IA's (each kind) w. assoc. SC's & update techniques. : Each instance in a SC is indexed by USR. These IA include both MTM's & NMTM SC's.

b) TM looks at index's & assoc. IA's (assigned by USR) as a QA problem SC. — Induces stoch variation : so indices are not very necessary : TM can choose IA for an unindexed QA. : I think this is = t. FC) idea in my earlier QATM models.

To some extent (unclear as to how much) cases (equivalently, pc's of cases, w/ less context dependence) are shared by all IA's. (I have not worked out details of this).

.20 2) Did I have any ideas different from 1)?

3) Learning during Lsrch, context dependent pc's of cases: How do these are process relevant to 12 ? Well, in 12 we do Lsrch to solve (perhaps all) problems. "Learning" in Lsrch is impt. in ($\tilde{\alpha}$) 2 ways a) learning from permits one to (+ or −) b) Solving a "correct" problem (no 2) & b) may be ideal kick (concepts!).

T. Context dependence of cases: This is (among other things) very relevant to the sharing of cases betw. different IA's & their assoc SC's.

.30 .31 (03) SN I know of a O_1 ⇒ It's top of .02 : It is conceivable to machine 320.03 using Lsrch to find soms. Or, (closer to practicality) a AZI-type machine w. Lisp or some other good universal ~~Turing~~ Mechanical Law., — using AZI for mental assignment of pc's. — This should be for stochastic models (NMTM) so e.g. t. 3 input machine (O, Q, R) would be one poss. — probably a better way of expressing PD's should be used, here. (Some 310.00 - 311.40 source v.d. ideas on this).

.31 Actually, ~~it~~ write be a v.g. way to get a T.M. "Oft. God". Decider or good set of IA's & then pose a "reasonable" TSQ for each IA. "Adequate" means TM would know what it had found a soln. but TSS would be too big ($\sim 10^{20}$, say). We either can choose by any

:00: : TM soln. completely "by hand", or give it an ut hints so that CJS is down to "accessible size". This CJS limit will depend on our cc resources we have available: 10^9 may be ok, but 10^{12} or 10^{15} if we had bigger machines.
Getting TM to do this search means it could find more complex codes than those we know — also, it could build on codes used in soln. of other IA's to which it had found "solns" — which we might not put in our "A. H." soln.

So, if we have a big cpy computer available, we can train TM "better". — In general, we want solns w. as large CJS as we can afford. — Since this makes it more likely that TM will find something better than its soln. we had in mind.

.10 In general, when we put in an "AH" soln, there will be usually much fewer useful (remote) sol-codes than it takes, more obtained more legitimately.

Having smaller computer available, means we have to spend a lot more time carefully constructing our TSQ's — so they have lots of generally useful codes in them.

SN In ANL, I had the following "flow" in mind: 3+5=8 has soln sum 3,5, → 8.

8×9=72 has soln mul 8,9. → 72!

+→× & sum→mul, so make a function +→sum, ×→mul.

.20 I TM would be unlikely to be able to do that kind of reasoning at this point in education; It would need some future ability to deal w. "logic of solutions" — & it has had at this point, no access to TSQ or a suitable IA that would enable it to deal with such problems.

.24 SN Confusion in my mind about what a "problem" is, what a "soln" is; what an "SC" is, & what an "IA" is:
B26.00 –.17 (on Stable & Dynamic IA's) seems to solve this problem.
Problems: an Q; Soln: Any Ai from Post use decides to a "soln": This can be of value by way of VSR.
e.g. say there are several equally good solns: As a TSQ, it would be well to include all Q; Ai; (j=1/k if there are k good solns). Even if the solns are not of exactly the same good but within some "ε" — it would be well to present the solns as a part of the corpus "BAG".

.30 An "SC" is a subset of the corpus, that all have an index on their Q.
Since a Q can have several indices, it can be in several SC's.
Perhaps have indices ordered (linear or partial) so there is a Main index (TOP) index.
The Main index determines the SC; How the other indices are used, how SC's can share corner into on pc's of cores, is unclear.
Perhaps certain problems can have > 1 Main index: → so they are in > 1 SC.
So: the indexing is more the problem of how & when to pool data from SC's — Unclear.
(Their usage is related with reference to

ID distinction? 322.40?

.00! 322.01! Looks at 331.04-.08! Here, the sc's were like indiv TM's (at first). This was just to teach individual IA's. Later, TM learns categories of Nouns Q's and sc's & somehow did it appropriate updating.

Say we've done this prelim exp. of .01! Also, we have TM able to give probabilistic assignments of Q's to sc's (if Q's are unheard).

.04 At a less abs. level, F() assigns naturally pc to the IAs (wrt a given Q_i), but (1) interprets/
.05 weighs the readings of the various sc's used (2) So somehow, figures out how to Update the various sc's wrt. this new QA. This last is diff. The one fixed by which it is decided may later influence the wts in

At the neurophysiological level: F() has complete control over production & updating. It also controls the context dependence of pc's of concepts used in constructing new mental forms to problems. F() is mainly the "part" of TM that works on updating — & because of that it must be involved in

.10 production as well (since updating ultimately depends on production).

.11 Enough for a while! Main Conclusions in 322.24: a problem, sc & IA:

In the simplest case, an IA works a problem, Q_n. Having done so, the pattern Q_n & A_n pairing is added to its associated sc — The main problem of updating of the IA also concerns the sc. It is also updated. Certain patterns are changed & bad, but also, new concepts can be found, because of the analysis

Assignment Pattern. (In AZ1 there is no "repeating" — but I feel uneasy about this)

It was mainly thinking about Algebraic things as in SAARS TSQ's.

Suppose TM "learns" to solve by linear eqns. If we give it lots of new linear eqns & it solves them all, in some sense, its skill in Algebra is increasing — but what is mainly getting more realistic is our confidence in its ability to do a certain sub corpus of problems — namely linear eqn. solving.

.20 Its pc's for solving other kinds of problems doesn't change much. — Some continuous parameters (pc's of concepts) will change slightly.

Actually, since the range for a MTM corpus, the values of pc's of most concepts will not change at all — as long as the linear eqns are all solved! The operator does not change if the corpus is in size, as long as the rule answers keep coming in.

In General for MTM problems, every one sc is noisy. If one had different sc's and different assoc operators One would find a Q discriminant function & improper the 2 operators!

Is an analogous thing possible for NMTM probs? Say we have 2 different sc's! S_1, S_2.

S_1 has a pc of P_1 for its corpus & if the pc of one is P_{01} so total pc is $P_{01} \cdot P_1$

.30 similarly for S_2 . . . $P_{02} \cdot P_2$.

If we had a discriminant function between 2 sc's: w. some pc = P_{disc}, Then we could decide to have S_3 that switch on S_1 for its corpus or S_2 for its corpus; the total pc would be

$P_{disc} \cdot P_1 P_2 \cdot P_{01} P_{02}$; It might be possible to get $P_{disc} \cdot P_{01} P_{02}$ by using common cases or [illegible] otherwise recoding, so this is a lower bound for the pc. This would be [illegible] better than using either O_1 or O_2 to code & enforce output, if O_1 & O_2 were much better on S_1 & S_2 respectively than S_3 & S_1 respectively. & the joint decision (O_1, O_2 & Discriminant Funct.) didn't have a too low pc. This assumes infallibility of the Discr. Funct.

.38 If we give the discr function wts w_1 & w_2 for the O_1 & O_2 or w_2 & w_1 (depending on which way it decides), then we end up w. a mixed prodn. for each Q in the concepts — constant v.s. dynamic IA's

& we may (or may not) find this better than $P_1 P_{01}$ or $P_2 P_{02}$.

326.00 space

This seems to [illegible] the [illegible] of uncertainty: I had in this area!

REPARSING → .20 ← This is a BIG change of way of looking at things!

.00: 323.40: So, say we did find a suitable discrim. funct. so that it made a pseudo-loss better: How do we then update O_1, O_2, S_1, S_2 & perhaps the discrim. funct.??

Also, is it o.k. if S_1 & S_2 have common elements??

.03 Using what was in 323.38 — it's expensive to complete pc of entire corpus because

.05 O_1 has to compute all of S_2's pc's & O_2 has to compute all of S_1's pc's:

An easier way would be to use a O_1 | discriminating new discrim — It would say "use O_1 on S_2" : If it were wrong occasionally, O_1 would have to do approximately as well on S_1.

.08 Now if the Discrim funct. was kept constant throughout the history of TM, & was used to define S_1 & S_2, it would make no errors in assigning past problems to O_1 & O_2 ☺

.00 Another approach that I may have used in past? → I may have done some MATH on this. For each problem, Get the 1 accepting heavily trained IA of TM), most problems are worked by more than one IA, so we don't worry about .03 - .05. An added advantage, is that it gives us a better idea as to how good the various IA's are on various problem areas. Then FC) uses the discrim. function. — But still, — how to do updating? How bad is it to update a pc/IA w.r.t. a problem that the pc/IA isn't particularly good at? Well, it would seem that we'd like a SC to be as "pure" as poss., so it didn't pound its own "operator dern bits" trying to "fit" a problem in which the operator was not used. Actually, we could "change" a Op/IA only for problems in which FC) chose Premises best. So this would clearly define S_1 & S_2 (like in .08).

.20 SN I had the idea that Reparsing would not be applied to this / newest model of TM! Maybe wrong! Say we get a new conc. Somehow, it enables us to reparse the same old O_n w/ much lower epc. Also several concs may have many pc's & a lot. (But updating an old conc. can be very tricky — may be not do it, unless it is easy, & "makes new waves" ☺. By "reparsing" I mean, in most extreme case, breaking down entire O_n into primitive functions/operators, Then reparsing in terms of Largest defns & their (new "first approximation") pc's. Explain The "unparsing of the "O_n" operator can be done very superficially, or to any depth. — Usually it's done not very deeply. After Reparsing of much depth the pc's change, so it may have to be repeatedly reparsed to get true weak pc.

.30: 310.29 I think the idea of 310.12—.30 was: To start out w/ a "user given" set of IA's: to use them on the corpus to calculate their pc's, so I'd have a pc of each IA as an "as" in a stock Gramm. Then create a stock Gramm. in which each IA is an "as" in that Gramm. This Hylevel Gramm. & its IA's can now be regarded as an ordinary stock Gramm w/ the POS or OP's as its terminals (leaves). As such, it things we can do to modify/optz it are much broader, than in its initial relatively formal "Grammars & w. Matrix/reanner" ON NMTM operations

310.12-.30 was first Gramm. of FC) from a very simple MetaGrammer — that just gave pc's to IA's) to a stronger Met Grammer that also generated new IAs (= stock Gramms), (For first Gramm this is not much of a gramm, if at all a gramm! It's a simple realization of what's going on! additive) so it needed not have "subgrammars" (Do most Grammars do have subgrammars) — or that it had new subgrammars & perhaps much of pc's of the form of the "like" "IA" Grammars.

In addition, I wanted this Meta Gramm to be context dependent: not only on the input problem, Q, 325.00

.00: 324.90! but on t. leading dern of @ t. present O_n, i.e. its structured internal pc's of its concs, etc.

.01 Using such a model, t. Question of what is to be updated? what SC's to use, etc. — become unclear

(perhaps irrelevant!)

Superficially, it would seem that t. very non-el version of TM would be easiest to update — conceptually consistent, that is, because t. Gorc is (somewhat) clearer. This least-El. TM is t. $\exists$ a $O_n(O_n, (QA)_{n+1}, \text{context}) \to O_{n+1}$: T. Gorc for any O_i is clearly

.08 defined (if not practically — its "max pc of corpus x pc of O_i dern": ← eq. of: (289.33)

[A possl additive complexity is that of "pc of dern" involving many ll derms.]

.08 should make it possl to evaluate any TM model. "Updating" a IA, e.g. would mean

.10 modifying t. params of that IA so that t. ⟨Gorc of t. exhaustive system⟩ of .08 is max.

.12 We deal w. optimal continuous params, viz t. $exp - \frac{1}{2}\left(\det(\text{Hessian})\right)^{-1}$ approach of

[≈ (250.09 -.19) ← this is on t. $\int_0^1 \int p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ w $\sum p_i = 1$, but is a special case of t. more

.15 general function that .12 approximates. We use AZ1 for t. Discrete params & t. Hessian for t. continuous params,

So, t. Q is: Is .08 - .15 an adequate framework for the quantification of optimization

of t. evolutionary sequence of TM models from Z10.12 to Y0311. ? 324.30 - 325.01 is a short

Summary of t. seq. of Model's from Z10.12 to Z311.90 (perhaps!)

Well, .08 - .15 seems reasonable. A possl Q is: The Gorc implied Gorcs may be too

.20 Hard to Evaluate! — So Either approximate or Modify t. Models to facilitate easier Gorc Evaln.

Applying the fuzzy ideas to t. fuzzy dualn. of CPM:

Say we have this F() & set of IA's: F() looks at Q_{n+1} & gives a P.D. over t. IAs: which in turn gives me a P.D. over the $\{A_{n+1}^i\}$. F() & [IAs] constitute a mapping from O_{n+1} in a d.f. on $\{A_{n+1}^i\}$. So this → is a "O_n". As such, we can use t. eq. of 289.33 to optimize params (both continuous & discrete) — which is what "updating" is.

As noted before, Each IA will have its own update techniques. They will usually have elements in them corresponding to elements in AZI: ① treat as continuous etc.

.30 ① update Contin params. ② reparse ③ find new functions, concepts: How to do ② is what's how much of each depends on CR available.

In updating, ideally, {F() & all of t. IA's params} are simultaneously updated. In practice, Each Q_{n+1} will usually involve only a few params in a few IAs & perhaps in F().

[Also Note 326.00 on "STATIC V.S. Dynamic IA's"

8/12/01

ID

STATIC & DYNAMIC IA's

323.40 spec

.00 : 325.40 : on 322.34; 323.11 : In general, a IA is a pgm in which user ports an input Q in & one ports a P.O. or Answers [$A_{\#}^{*}$]. There seem to be 2 kinds of IA's: Dynamic

One kind is static, & never updated. Another changes as its sc & resrs. — i.e. its I/O behavior changes.

I think ~~Rivers~~ basically 1 sort of thing I was worried a fact re 322.24 ff & 323.11 ff.

So; say both kind of IAs used in 1 IA pool, [IA$_i$]. They are both used by F( ) to solve problems.

A static IA could be a component of a dynamic IA: E.G. O, i.e. overall Operator, is a Dynamic IA, but it can, via F() use static IA's when they seem appropriate.

A IA Predicting a time series would vary its params (both continuous & discrete) as its corpus grows. A correlating Alg. could be static — unchanging as its corpus grows.

.10 Somehow, 1 updating Alg. must recognize what needs to be updated & what doesn't.

Well, 1 Master Updating Model" (present) is 325.08 - .15 1 current O is modified to make 325.08 ≈ (289.33) (eg). In this Modifn procedure, certain IA's will be modified & others will ~~be~~ perhaps never be modified.

.17 More examples of static/dynamic IAs: Static: how to solve quadeqs. dynamic: how to solve n.l. eqns. ; Somewhat static is solving linear eqns. — but IA's could start out dynamic — then "settle down" to static.

The Master Updat model of 325.08 -.15 is supposed to take care of 1 "sc" problem → what sc's over

.19 assoc w. what Dynamic IA's, & act] 1 detailed Mechanics of this are as yet unclear to me.

.20 Hm., I think I have now have more than enuf to write an intelligible report. In my own personal hard copy & several copys: have refs to my notes, expanding various points in 1 report.

Some of these expansions should be in 1 or more hard copy also.

The preliminary outline of 320.00 looks fine: 320.22 (?) needs to be expanded — which is what 821.00 - 826.19 is about.

Descriptions of "Initialization of CPM": ~~Sets~~

1) Set of IA's; F( ): Indexed Q's: so F() is not much needed to assign Q's to IA's.

~~Acts like~~ Each IA acts separately on its indexed sc.

How do IA's work? I have idea of how IA for Mitch works. — its AZ1. - at first, w/ unconditional pc's of (Turing) types.

Try to find examples of static IA's! Static & Dynamic.

.30 In more advanced machine: Given set of old Q's that were indexed, assoc A$_i$ are 1. indexes. Given: To induce index for a new (or old) Q$_i$. — eX derives PD (usually!).

Some static IA's: (nonlinear linear regress), curve fitting, ANN.

These can also be used as Dynamic IA's: e.g. in linear regression, CPM learns 1 sequence & improves its recfs & no. of coeffs as it gets more data.

Updating is trivial. adjustment of continuous a/o consideration of "new terms"; new functional forms.

Analy predn. of Beam Eqq. is static or dynamic.

ID REV 3.27 .19 → 328.23

.00 : There is some distance betw. stoch op. (O_n is a stoch op), a P.D. ($\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$ is a "PD") & IA's: I guess a IA has a corpus of examples of input & associated as output. They differ as to what the corpus elements are. O_n is not an IA, but CPM is an IA. In CPM the corpus is $\sum Q_i A_i]_{i=1/n}$: [in this representation, (linear or Newton-Raphson) the input is a time series. / A stoch op. is a degenerate IA: it ignores all previous data corpus data. It is a static IA.

An IA can be for sequential data (time series) or for unordered data

(* Stoch (any for simple preds; Stoch operator for any corpus of QA pairs)

3 common types of IA input: (1) time series (2) Bag of objects (3) Pkg of I/O pairs.

.10 These objects can be strings s/c ~~Reals s/c integers~~. (Nums. are Real or integer) or complex Integers, reals, complexes...

T. system presents a new (Q / QA) to a IA for (predn / updatn): It already has the previous part of the assoc corpus.

T. IA's of interest must be able to deal w. a $\sum [Q_i A_i]_{1/n}$ / type corpus., time series,

Hyp. a IA that is a time series predictor, can regard the $Q_i A_i$'s as elements of a Time Series

Then given a Q not in it can give D.P. for Ans! — Hyp. it may be losing info that

T. data is "unordered".

.19 Anyway: As present, FC) (when educated properly), looks at Q & sorts Q (or QA)

to certain IA's; or better, FC) outputs a PD on IA's — pc that each will be kept"

.26 (cf. "best" means highest pc for value A_i). So, ideally, FC) obtains a PD over

.2C T. IA's for a given Q. → (.29)

Woops! In .19-.2C; T. pc assoc w. each IA will depend on how T. in Fo is used.

If a prediction via T. IA was assigned high pc for that Q, then T. pc assoc w. each

IA in T. pc that it will ~~have made~~ assign high pc to A_i.

Hyp. If predns are T. pc wtd Σ of T. outputs of T. IA's, then [those pc's will be adjusted

to maximize optimize predns on past exs & present cats.

.28 → Which way to do it? (of T. 2 ways) is something I don't have to decide now: (See 337.23-.32 Another Dem of T. "2 ways of Forces for FC)" problem.

.29 (.31) → So to continue. Even if FC) were v.g. this system would be little more, than using

.30 a bunch of IA's in II & deciding which (or using a wtd mean) to give preds: Updatn

.31 of IA's would be same as if they were independent IA's.

.32 → A first improvement over .19-.31; Call T. resultant IA; O_n (after updatn term (QA))

Use O_n to do context dependent "selection" of cases in updatn T. IA's. This

means that in a Universal (or very extensible) IA's finding good updates,

finding v.g. new cases for updatn would occur much faster than w.o. this feature.

Also Lsrch is either eliminated & uses Irug during Lsrch to deal w. curvel/n.

→ To do sequential runs during Lsrch.

.39 Another improvement is in "Irug during Lsrch": So after 1 or more trials, T. PD vs for

srch changes. It also helps deal w. Corrects betw. Lsrch trials.

Rev.: 326.24 – 330.40

.00 : : 327.39 – 40 are important in helping realize 589 Fn 2 — We should check to see that all kinds of leaves are, indeed, accessible to this system.

.02 A perhaps Nonrec. method of Doing Lsrch: T. input to CPM is T. problem being "Lsrch'd". Also inputs are triples of these results up to now. — Output is essentially what next Mal is. When (in .02) CPM is "called", ~~it is also~~ CPM is also given a time limit telling how much CL to spend on this problem.

.07 A serious criticism of all of the above is that there is no "lookahead": Y. searches are all Grossly "Greedy" (= "myopic"). How we could teach CPM how to solve "lookahead" problems & then it could apply them to "Aided Lsrch". — At the lowest level, all Lsrches are controlled directly by CPM: It looks at the problem, the previous trial results & devises the next trial.

.10 One defect of present Systems: I don't seem to have any good practical ways to do Stochastic-Functions for stochastic IA's:

There is the 3 input one., but I find it difficult to put intuitive probabilistic ideas into it.

Our 22800 ff on 16 → IA's: It's not clear how many of them are stochastic! — It seems that at least 10 of them are, indeed stochastic.

.20 How, I shouldn't worry about this now. ~~In~~ In designing a TSQ, I if I know of a probabilistic soln. to a problem, I will design a IA that has Access to that soln.

In particular, various Heurs will be of probabilistic form & I'll have to design a IA (or IA's) to "exploit" them. "explores"

How, 310.00 – 311.40 have some Good ideas on developing a good, useful set of stoch IA's. —

.27 TS Q's for them, etc.

So: Present Summary of initial System!

.24 1) We start w. a set of IA's! Both M & NM. (M). An IA has a ~~Segment-Procedure~~ [Q, A] as corpus [2 obligations it is able to accomplish]. Gn. a new Q; a Pd. on Ai'. ~~An~~ Defn. of an IA includes its (incremental) updating algm.

2) USR gives system a set of Q's, Ai' pairs; Each Q is indexed, telling which 1 (or maybe >1) IA is to work on Y. problem. These IA's incrementally update themselves on Y. Q A's they are given. TS. IA's at first pt. are indep of successive Q's

.30 3) F( ) is created: [Its inputs are Q's] uses 2 a set of IA's (& nearby trainer) to look at Y. ~~problems~~ Q's & ~~IA assignments by H's.~~ ~~F( ) then is the probabilistic relations found between~~ & Main associated indices, & finds a probabilistic relationship betw. them. From this relation, F( ) is inductively able to assign indices (→ IA's) to new, unindexed Q's.

.38 4) The combination of {F( ) & Y. set of IA's} is a new IA = O (theirs) when a new Q & comes in, this O is updated by updating Y. relevant IA's & also updating Y. F( ) function. — But before that, note that F( ) has probabilistic outputs, so when ~~the~~

SN Here
I've been ignoring Y. utility of common Concs. into Various IA's. I.e. "Sauvers" IA using Y. "same" Conc.
↑ This Needs work.

8/13/01
ID

REV

2.4pp/d. 1.6pp/d exactly since 5/1/01
1 Month
+ 1 DSIA!
50% Fighter Pilot Hours

329

.00: 228.90 : for a new, unindexed Q, it doesn't pick any IA exactly — But we can apply the IA's of least 2 or 3 pc to the problem & update them on the relevant Q, A. Predictions can be made using wtd means of the IAs! But see 327.19–28 : There are at least 2 ways that FC) can assign pc (wts) to IA's (for a given CQ input).
I don't think it's a critical decision, but it has to be made.

With FC) in place & suitable training, we can get to work on 2 critical SI Problems : 5) & 6) [But note 330.16 : It may be possible to do 5) &/or 6) earlier in the TSQ than this]

89 4.1.01
72pp / 30 = 2.4pp/d

161 sym.01
329 8.19.01
Σ = 168 103+2 = 105
1.6pp/d exactly/since

5) (Context dependence of pc's of concs.)

Updating of IA's typically can while IA's will differ somewhat in their updating algorithms, these updating algorithms will, typically, have 2 aspects. As new data is added to the scope of an IA,

.10 omit

IA, a) Certain (various) parameters are changed. These are usually frequencies of occurrence of various elements in the sc.

b) New functions, or concepts are found in the sc. In the early stages of training, [learning] the updating of various concs will be assigned pc's based on frequencies of occurrence in the sc. The changing of these freqs will be recorded & noted in the updating process.

b) The concs of a) are combined to form higher order concs, which are to be evaluated by the System Designer. The frequencies of a) are used to assign pc's to the final concs before evaluation.

c) When new useful concs are found, the prediction program of the IA will be reparsed in terms of them. Using them & in attempts

.20 to find the optimum of the current IA's program. (i.e. find an effectively "shorter description" of this pgm)

As the amount of data in the corpus of each IA grows, it becomes possible to calculate the pc of a new concept, not only by the frequencies of past occurrences of its component sub-concepts — but by the contexts in which these sub-concepts occurred. [This context dependent probability can be done by the human CPM operator if the system has had adequate training experience in this area.

This context Discovery of context dependency of pc's of concs is critical in discovering new concs in larger systems. Without it, the probability assigned to new concs decreases rapidly with system growth and rapidly increases the cpu's of new problem beyond our computation capacity.

.30 — But whether it ever gets completely out of hand or not, a system without context dependency of pc's is grossly inefficient in computation cost.

6) In the process of search for new concs, there is part of the updating algs of most IA's, the search is normally guided by a pd that remains constant during the search. The constancy of these pd's makes it impossible for the

Rev

.00 : 329.90 Search to take advantage of information gained in earlier trials. By recalculating the guiding Pd after each trial, or after every few trials, the efficiency of the search can probably be markedly increased. This modification of the guiding Pd corresponds to "Learning during search". It also deals with an equivalent inefficiency of L search — that if trials are not statistically indep, but highly correlated, the search becomes very inefficient. "Learning during Rx Search" ~~remains, with the same~~ appears to be a good way to deal with this correlation problem. This "learning during L search" may require learning beyond that needed to construct the original static, guiding Pd.

.08

.10 ~~Notes~~ While re reading 328.24 – 330.08: (328.30R): 1) ~~The~~ At the early stages of EA's is indiv. It would seem that Common Cores in the AI's would tend to be more alike PC's. — But this needs work! As is, the system will probably work "adequately" w.o. this feature

2) In (4.) 328.38ff: T. FC) mixing algm: ~~Edit~~ I have worked out (in previous notes) some analysis of optimum n.l. mixing. However linear analysis of linear mixing using cross corrs, might be easier/faster as a good "first approx" for the N-l ~~early~~ optim.

.16 3) In 328.24: Initial Operation: The sophisticated forms of SX (context dependent pc's for components since ~~is~~ in updating ~~by~~ is G) (during search) could be done to some extent: That assignment of suitable IA's for each of these problems may have to be done by USR, even in ~~early~~ very early TSQ's.

.20 4) The cc's are considered in Search, there is no mention of cc in the forego. discussion. I have however written a framework for a general option of the system wrt. cc, see 303.22 – .29 for preliminary discn.: 285.00–.40 for immediate(, but this needs much more

.23 5) Note 303.08–.21, .30–.40 — on optimality (within factor of 2) of the system. This amounts to an over-all methodology for optimizing of the system — i.e. so it is capable of ~~see~~ realizing/simulating any describable (human) ~~search~~ search heuristic (This is ~~surely~~ nearly PC oriented heuristic — the problem of "Quick abort" seems, dimly..... I may be able to get CPM to work on this, if I can find a general way to describe the over all cc problem to CPM!.

.29 6) Be sure to mention the over-all theor: That any problem describable by a human can possibly be usefully worked on by CPM. — So if there are any deficiencies in my Model of CPM, I can probably get CPM to try to deal w. them.

.30

7) 327.19 – 328.23 is the previous review (w/ the interlocutor)! Has other Comments, Criticisms?

8) The system as described has no (codified) ~~of~~ Maximum Myopia ("Greedy"): This could be done via (.29) also see 328.07. Backgammon

3/14/01 ED

M Op → M̄ Op

Det Operator → Stoch. Operator .26

MOp → M̄Op

.00: 330.10: Summary of impt ideas in System: (= "Claims" in a patent Applicn.)

1. Use of set of IA's (rather than a single IA) to solve great variety of problem types
2. Suggested list of IAs.
3. Use of Learn to help update IAs.
4. Training of FC) run Gen to decide which IAs should work on problems how much wt. to give scores of each IA.
4. Impt. methods or SI of system:
   a) context dependent pc's of concs
   b) (run during Learn.
5) overall Gen of System.
6) Final Goal of $O_{n+1} = O_n(O_n, Q_{n+1}, A_{n+1}, \cdots)$ using Gen of 5)

228.00ff

Ofl. 16 IA's on 228! ⑥ ANN, RANN } Please see not actually IA's: they are Optzn. Methods.
⑦ GA, SGA, Simanneal.

Their Optzn. aspects can be used as part of a Prodn. Scheme; e.g. they can be used to implement (or help implement) the "updating" of an IA. Or they can be used directly to find "shortcuts" for a corpus — which amounts to a very general method of Induction.

In particular, ANN & RANN can be used to derb. M operators (deterministic operators). They can also be used to derb. stochastic operators, by giving a central prodn/M w. & an observed σ^2.

In a similar way GA (etal) can be used to hunt for deterministic Operators ($M \pm \delta$) as above.

Both (RANN / ANN) & (GA etal.) can also be used to optimize Gen on a corpus in terms of a M for every prodn. & a possibly distinct σ^2 for each prodn. [usually, 1 prodn system gives only M & we must use a mean σ^2 for the whole corpus; from which we induce the best σ^2 fit for the model.]

Looks Good!

SN A Way to take a M Operator (Strings → Strings) & approximate to a M̄ operator:

1. M Op is observed to be correct say 80% of the time. When it is correct its pc = .8.
When it is incorrect its pc = .2 must be unfound pd overall other resp. probs. — This can be given via the R input. It's necy to do this because otherwise if the Op operator is wrong, it would give zero pc for the right answer, — which means that the Gen would give it Zero score! We can contrast the Gen obtained, if the system gave the Standard (Bernoulli) d.f. for each Q — This is our D.f. being the Uncondl P.D. → (340.31)

.26 Here is a perfectly general way to convert any M Op to a M̄ Op
Det Op (deterministic) | Stoch Op (chaotic).

So: nilcost/resp = $-.8 \times (\ln .8 + \ln 1) = .2(\ln .2 + \ln k)$ k = geometric mean pc of items in default (uncond/usual) d.f.
W. respect to det. predictor, nilcost/case $= -\ln k$ $= -.8 \ln .8 - .2 \ln .2 - .2 \ln k$
$\equiv \ln(.8^{.8} \cdot .2^{.2}) \cdot .500 - .2 \ln k$
v.s. $-\ln k$.

$.5 \ln .5 = -.3465$
$\epsilon \ln \epsilon \to 0$

ID

.00 : 331.40 ; Title (anybo) : CPM – A Conditional Probability Machine. [1] – FN1

FN1 ; CPM was the nomenclature early (Basic DOS) operating system for Micro Computers. However It is unlikely that this Acronym will cause confusion in the mind of the sophisticated reader.

.03 The problem we are addressing is that of using a general probabilistic prediction!

We are given a sequence of (Q_i, A_i) pairs $\ldots i = 1 \ldots n$, and a new general Q_{n+1}, to give a probability distribution over all possible A_{n+1}.

Here the Q's and A's can be strings, numbers or mixtures of them. The Q's can be questions, or we the A's as answers. Think of Q_i as input to an unknown stochastic operator, whose output is A_i.

.30 Q_i can be any description of a problem and the an acceptable solution to it.

Since we are considering stochastic models, the data can have "errors" in it — interpreted perhaps, by the machine as "noise".

We will first describe a general solution that we expect to reach after a very large, suitably constructed set of $Q_i A_i$ pairs.

.17 O_n is an operator that is able to look at any Q_n (string and/or number) and output a probability distribution over all possible A_n (string and/or number). Whatever we want, ultimately, is the updating scheme. FN 2

.19 $O_n(O_n', Q_n, A_n) \rightarrow$ a probability distribution on O_{n+1}'. (1) ← Equ

.20 Here O_n' is a "short description" of the operator O_n.

(O_n', Q_n, A_n) is regarded as a single input to O_n — it is a kind of "Q".

O_{n+1}' is a "short description" of the new "updated" O_{n+1}.

O_{n+1} is a somewhat more "educated" version of O_n.

FN2 : This expression relating O_{n+1}' to O_n is only an approximation. The requirement of O_n also includes cc_n (l.c. subscript), which is the amount of computation cost we are willing (usually) to spend on a "subsequent" problem. Other requirements that O_n might use are that O_i', Q_i, A_i values for $i = n-k$ to n (l.c.) — If cc_n (l.c. subscript) is large enough, O_n will be able to "back track" (l.c.) and derive a satisfactory version of O_{n+1}' based on the most recent k examples of data. (cc_i)

A "Global" back track would compute O_{n+1}' on the basis of all of the $Q_i A_i$ data from $i = 1$ to n — This can be feasible only if cc_i is very large and/or if n is very small.

.00: 332.40 : The goal for ~~Conviction That we are for~~ constructing O_n is that

.01 $P(O_n^{\bullet}) \cdot \prod_{i=1}^{n} P_{O_n}(A_i|Q_i)$ (1) ← (eq. no.)

be maximized ← a maximum likelihood criterion.

Here $P(O_n^{\bullet})$ is the apriori probability of O_n. It is approximately $2^{-\text{length}(O_n)}$ (states).

$P_{O_n}(A_i|Q_i)$ is the probability O_n assigns to A_i for input Q_n.

u.c. / u.f.computable

In general, it is not computationally possible to maximize expression (1) ← in any finite time. We will try to do as well as possible with the available computation resources (usually time).

.10

We do not know of a way to obtain ~~directly~~ in a direct manner, an O_n operator that satisfies both (1) and (2).

Instead we will describe a ~~sequence~~ sequence of operators of increasing power, that will ultimately culminate in O_n.

347.00

Some ideas on the next section:

1) FC() could optimize on $\Pi_{O_n}(A_i|Q_i)$ as soon as the indexing steps — ~~[illegible]~~

I don't see any advantage to trying to duplicate the indices — (two perhaps, computationally, it would take much less CC — the no. of cases is large so many IA evals would have to be done. However, look into this! — from CC pt. of view. Note that after FC() has obtained a function) form for $FC(Q_i) \rightarrow IA_j$, the updating of FC() is then incremental & doesn't involve any

.20 which IA evals.

2) GA's as IA's. When FC() chooses a ~~GA~~ to work on Q_i, FC() should specify the GA ~~as~~ as narrowly as possible, by giving population size, initial population, ← This is where simple characteristics of the GA. nature of mutations & crossovers. What this means is that to use GA's effectively,

FC() would have to study their behavior a lot. Essential!

Also, FC() should be able to watch the GA at work & suggest changes in crossovers & perhaps the GA fitness function.

modifications of mutations ← simple!

344.16 specific on GA's as IA's.

→ 334.00

.29

.30

SN On punctuation (1) It wouldn't be needed if I used prefix codes. (Ri Ray seems Expensive I should look into that on (2) in any $\hat{\propto} \frac{1}{n^2}$ approx; & look out normalization — look it up in Rissi's Papers (3) ~~Perhaps~~ To delimit a few long integer chunks or codes, I could use prefix coded integers to tell how many bits (or symbols) integers in each section. (4) Could Modifying punctuation get rid of the "$\frac{1}{e}$" factor in Z1?

— I doubt it, but look into this! (3) In Human ~~[illegible]~~ language compl: frequency of symbols dependent complexity sizes so $pc \approx \frac{\text{case count of symbol}}{\text{total no. of symbols}} \cdot \ln(\text{total no. of symbols})$: But look at that Book & for Details!

Entire entire.

(4) In some situations (like part of Learn) the system is self-normalizing; so using pc = at would be ok.

ID MINT = "Mint is not TRAC"

.00 (Spec 333.29): Note that the only Main pt. of using IA's other than a single Universal IA, is that these IA's act like "pre-educated" Univl. IA's — so it saves many cc's of lines of Univ TSQ's to calculate via a single Univ. IA. Normally, there would be difficulty in assigning proper wts to these IA's — but in the present case this is done by "Irng".

.05 → A bit of confusion on "Universality": AZ1 employs a universal d.f. of stochastic functs, by using a Σ_1-type d.f. on cones used in construction of new d.f.'s, — using context-free pc's of cones. However, using context dependences of pc's of cones, would seem to introduce a lot more "extensibility" to the system. Not only that, but w.o. context, the system becomes less "Scalable"! Also, "Context enables pc's of correct cones to be much higher, so Irng value is much ↑! (This is = need for "Scalability!!" This (at least) seems to be extremely impt.!

.10 → My impression is that w.o. context dependence, the system lacks an impt. aspect of Universality! Thus suggests that my ~~concept of~~ idea of the "Universality" of the 1 level system — the 3 input func. — is faulty: That this "Hyr. level" of context dependence is necy; but not nearly sufficient — that there may be many needed, & there may be alternative Way(s) to achieve it! — perhaps a simpler overall system.

So, the 3 input [O, Q, R] case employs a Universal d.f. over all O's. — i.e. every possl. O has a pc > 0. ("O" = "stochastic relation between Q & A") — which seems (superficially) to be O.K. — we are hunting for a particular O & would be happy once we'd found it.

.20 My feeling is that certain methods of coding can be much more efficient than others. So, context dependent coding should be better (or at least as Good) as AZ1, — It takes advantage of Aspects that AZ1 doesn't see!

A possibly relevant "Study": N.L. represents w., say 10 continuous params: pc of corpus will be ~ K + (plus Discrete) (N is length of corpus). If the Discrete part is coded poorly, this will not make much difference for large N, since the (plus N term will dominate. However, if K is large, then reducing it by better coding could be important. Perhaps "Context sensitive coding" corresponds to better codes for K.

Any shortening of the code for the discrete part reduces its CTS: which is usually what we're interested in. In straight AZ1: we get this code. This is a min code is unlikely! It looks for a very limited no. of

.30 ragy types in the corpus → Probly more, but see 335.16's conjecture → 335.16 →

.31 A less El. version of "context dependent pc's" is: given past history of $O_n(QA)_n \cdots O_1'(QA)_1$, $O_{n+1}'(QA)_{n+1}, \cdots O_N'(QA)_N \cdots$; Given $(QA)_{n+1}$ to good p.d. for O_{n+1}. — Here we have ≠ sequences/ prodn. — which is the final TM (self updating) that I'm aiming for.

Context dependent pc of component cones is once small step I'm in the right direction.

.37 Distinguish 2 kinds of pc: 1) the pc's the AZ1 Irng. assigns to functions by composition, definition & frequency. 2) The pc's assigned to operations in connection with a function, based on the target problem sets (Q_{n+1}, A_{n+1}) the AZ1 assigns pc's, & experience in constructing

8/15/01
IDEA

.00 : 334.40 : functions that solve new probs, reclassifications of old problems is heavy by pc (according to AZ1), i do this w/ acceptable cc. The final Score is the function of 333.01 being max.

AZ1 gives the P(On) factor. — we hope to keep the "TT" part constant except for the letters Qnt+1, Ant+1 →

$P_{On+1}(Ant+1 \mid Qnt+1)$.

We have the role of AZ1 as part of the Score: Also its role as a first approx of pc's to be used in trial solns of problems).

An important point is that the AZ1 pc's are completely & almost completely indep of the problem Qnt+1, Ant+1 that are partly because they've been used to solve previous problems i (Qnt, Ant) is (usually) related to (but not identical to) earlier problems.

"To solve the problem" — one must search backwards from $(QA)_{n+1}$ — a technique not suggested by AZ1's pc values.

.10 .11 The problems discussed on 334.05 (to present) is of much import. While the discussion of 334.37 to present must recount clarifies a lot, there is still a lot that is unclear — in particular 334.05-.10 — on "universality".

334.05-.39 talks about universality, i.e. the fact that the code that AZ1 has for the corpus may have redundancies init — that as is, AZ1 is unable to "see" & thus reduce those redundancies.

32431 — 3335.11 Discusses the practical problems finding a Min code for the corpus.

.16 **Conjecture:** That removing the redundancies (if any) in AZ1's code for the corpus, will not improve TM's effectiveness much. Hwr, finding ways to rapidly construct operators that give high score via (eg GORC of 333.01) is much more impt. Using "Auxiliary EA's" to help solve this problem is one such way.

.20 The idea of "context" is "any info that will help solve the open problem" 2337". If is "context" includes [illegible] as well as specific Plans this includes successful Nods info on what happened in earlier trials — we have "temporally local search" included in "Contextual considerations".

300.00ff (& 300.26ff in particular) expands the idea of "context" in very useful ways.

8/17/01 This idea of "context" seems very impt, even in Easy TM (simple, schemas eg).

I think I had no idea that usually the info pointed to — that was used for Score Evaln — was "too small" to account for the amount of "intelligence" used in problem solving! I.E. The Function O, isn't ordinarily large enough so that the pc's of the sub-problems such, give (a sharp enuf d.p.) to make it such practical for large O. The SSZ within O is too small to give us usable CJS. The guidance of the search has to use other info: Some of it heuristic (from a much larger auxiliary corpus) : some of it from the present corpus ... [redacted] ... i.e. "context". Hwr, I do have to work out just what the corpus is SSZ for this "context" is.

& the SSZ for the/a prob is indep of the no. of examples of use of O (which could be very quite large).

I think it may be necy for me to actually work on a TSP to get a good understanding of "Context" & its SSZ.

8/17/01

ID

.00: 335.40: One" intuitive" idea of "context", was that it was a narrowing of the sub-corpus — so that in this small s.c., the pc's of concs. were much different! That there was a small set of concs w. hy pc & the rest had very low pc. That in this "context" s.c., one or a few (the narrowed concs. implied) knew which small set of concs to use — That in a sense, the context (the hows) "solved the problem".

It may be more than always referred to the context, the context decided which how(s) to use.

Version 1.414 (19-8) build 1598.1002 Win 95/98/NT

This whole Question seems to be one of the most critical in my Theoretical Understanding of TM's Operation!

Try to frame the Question More Exactly: the appl of the concs within O: this is completely in dep of the corpus. It depend only on the "Language" AZ1. (There may be more than 1 way to express the primitive & defined functions. But this is not the "context" that

.10 I'm concerned w.: 300.26 is more what I mean by "context": Note 301, 36

Actually, all of 300.00 – 303.21 is very much a "context". First part of this stuff (302.00 – 303.21) is on "correlations" & "trying during search", which is perhaps the same kind of reference to such.

Order no 426772 $40 to "Res. Mach"

HA! A nice way to think about this! Say O_n was actually the result of cc=∞. Actually ALP = which is even better!) Q_n, is then a "Summary Machine" for the (perfect) $(Q_n A_n)$ ∑ 1/n corpus. — w. a "built in sse" We apply Q_n to the "corpus" that we're using to control trials for O_{n+1}. If O_n were finite (cc=∞) Summary Machine O_{n+1} needs as info, only O_n & O_{n+1}, A_{n+1}.

.24 So, superficially, it would seem that using anyRun info (like that of 300.26 – .35) would amount to "double counting".

Hm, given O_n, Q_{n+1}, A_{n+1} I'm not sure one can find O_{n+1} exactly! See .22

A possl. way out is that RLP ≠ ALP. — But finite cc makes a total distance!

.20 O_n, Second Rob, it is/ not really "double counting", because the final Gen, 289.33 doesn't "double Count"!

.22 While (.24R) O_n & Q_n, A_n appear to be totally unrelated to (new) O_{n+1}, one can, in principle, if O_n were given as its (finite) all possl. O^i_{n-1}'s w. suitable wts, w_i ($\Rightarrow O_n() = \sum w_i O_{n-i}()$) then $O_{n+1}() = \sum w_i O^i_{n-1}(A_n/Q_n) \cdot O_{n-i}()$. Therefore

.25 In fact, the one would really have to sum over only the O^i_{n-1}'s that are important. → 352.10

Say, as "context", TM has all traces of its behavior since Start; the O_n updates in trying to Prediction for O_{n+1} in terms of O_{n+1}, A_{n+1} would depend much on how "good" TM was in predn during time 0 to n. — if it was poor, the O_{n+1} distribn would be poor. At best, this induction .30 will speed up finding reasonable trials for O_{n+1} — so w. given CB, TM would do better.

.31 Remember In this particular "SI" model, TM doesn't really know what the problem is — it's merely trying to predict what O_{n+1} would be on basis of past results.

If would be much better if TM really understood what the problem was. So it would work it as an OZ problem. I don't particularly like the (Levin) Lsrch solm of the OZ problem: It merely consists of looking for a good Optim problem Algm. Perhaps use GA for that aspect of the problem! It would be much better if the GA were more tightly linked to TM's "normal" operations of Operator Induction.

8/17/01

ID

.00 : Another possy is that I'd clean. t. problem : That I'd try to find ways to construct functions that were useful for pred.

Perhaps a good Protocol "Solu."

1) Try to Define t. problem as clearly as possl.; w. some impt-Q's & Refs to discuss about them. (.09)

2) Do a TSQ w. t/ several IAS's & a PC) pondrum. Try to see just how "context" needs to be used. Try to define some useful "contexts" that actually work, then see if I can Get TM to learn to do this.

.09 3) T. problem : ~~to Get OA for~~ Essentially, t "OZ problem": (See 228.21 for T. problem of getting CPM to understand what an OZ problem is.)

.10 To propose a O_{n+1} of the unk Gore (Usual hyp of O_{n+1} dcr. ambit by PC of computations of O_{n+1})

Using as info, everything that t. (cc limited) TM has done, working on this problem.

In general, a human① working another person②, trying to solve a problem, that human① knows t Gedanken, should be able to continue, & to get ideas about solns, from ②'s "traces". — Particularly if ②'s trace is in "shorthand" i.e. t. basic concs are indicated. These "concs" can be concs in AZ's lang, but need not be — since they are/ could be ~~code~~ some thing else. AZ's codes are for O_n, — an attempt for by Gore. The concs & traces are various attempts by a cc limited machine, to solve a certain optimization problem.

.20 4) Even if I don't have a general domain of context usable by CPM, I could write an Intelligible kernel CPM : i.e. FC)'s operations "learned" by TM by using t. post hoc utility of t. solns: (There are 2 possl. Gores for this FC) .. (framework w. min less

.23 Wk. on so, but ~~(not finished)~~ ① (version FC) gives a pd. on a particular ~~problem~~.

.23 IA is giving Best PC for t. A of interest. — This is filtering normal Search for ~~OA~~ OZ problem

2 is appropriate if we will = later use these PC's to Guide Search.

.24 ② FC) assigns a PC to each IA for a particular A. The meaning of that PC is that t. estimate for t. PC of a A is obtained by this PC w.d measure & DPS on A obtained by each of t. IA's. T. Gore is then t usual $(pc \text{ of } O_n) \cdot \prod_{i=1}^{n} P_{O_n}(A_i \mid O_i)$

.30 $$P_{O_n}(A_i \mid O_i) \equiv \sum_{IA_j} P(A_i, Q_2) \cdot PC(IA_j) \quad : \quad \forall c(IA_j) \text{ a thing's PC.}$$

Since they are used distrutly t PC's of ① & ② how different Gores are produced distrutly. → See 3 27.19 -.28 for an initial discn of t 2 poss'l Gores" situation.

.32

8/19/01

Artificial Intelligence Copy of this at:
kdd.research.org/KDD/Workshops/IJCAI-2001/
orig. Copy in ss\ wrappers.txt.

For More on this:
Google:
Wrappers KDD

A.

IJCAI-01 on : on Sat 4 Aug 2001

IJCAI-01on by W. H. Hsu

Wrappers for Performance Enhancement inDiscovery in Databases (KDD)
[workshop code ML-5]
Saturday, 04 August 2001
Seattle, Washington, USA
Workshop Description
(Last updated 05 Dec 2000)

The rapidly increasing volume of data collected for decisionapplications in commercial, industrial, medical, and defense domains has made it a challenge to scale up knowledge discovery in(KDD), the machine learning and knowledge acquisitionof these applications. Many techniques currently appliedKDD admit enhancement through the wrapper approach, which usesperformance of inductive learning algorithms as feedbackoptimize parameters of the learning system.

Wrappers include algorithms for performance tuning, especially:of learning system parameters (hyperparameters) such asrates and model priors; control of solution size; and changeproblem representation (or inductive bias optimization).for changing the representation of a machine learninginclude decomposition of learning tasks into more tractable; feature construction, or synthesis of more salient orinput variables; and feature subset selection, also known aselimination (a form of relevance determination).

This workshop will explore current issues concerning wrapperfor KDD applications, including:
problems to which wrappers can be successfully applied
dimensionality reduction
feature subset selection with irrelevant variables
feature extraction
other change of representation for KDD
transforms
ensemble learning (mixture estimation)
empirical performance tuning and experimentation
meta-learning
learning methodologies that admit or support wrappers
search-based wrappers (selection, partitioning)
probabilistic wrappers
metric-based model selection
statistical validation methods
evolutionary computation approaches
genetic algorithms for feature selection and extraction
feature construction by genetic programming
"parameterless" GAs for performance tuning
detection and reduction of GP code bloat
linkage learning
multi-objective optimization
foundational theories
statistical learning theory
genetic and evolutionary computation
intron detection and elimination
theory of linkage learning
finding the GA "sweet spot"
the role of prior knowledge and interactive knowledge elicitation in developing wrappers for KDD

Workshop Audience

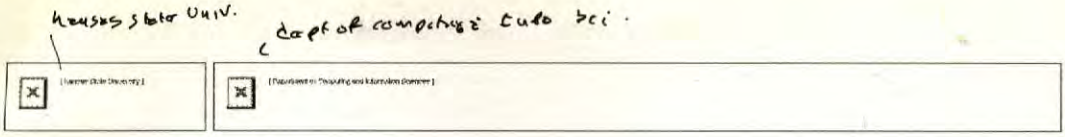
This workshop is intended for researchers in the area of machine, including practitioners of knowledge discovery in databases(KDD) and statistical and computational-learning theorists. Intelligentresearchers

- meta-learning
- learning methodologies that admit or support wrappers
  - search-based wrappers (selection, partitioning)
  - probabilistic wrappers
  - metric-based model selection
  - statistical validation methods
  - evolutionary computation approaches
    - genetic algorithms for feature selection and extraction
    - feature construction by genetic programming
    - "parameterless" GAs for performance tuning
    - detection and reduction of GP code bloat
  - linkage learning
  - multi-objective optimization
- foundational theories
  - statistical learning theory
  - genetic and evolutionary computation
    - intron detection and elimination
    - theory of linkage learning
    - finding the GA "sweet spot"
- the role of prior knowledge and interactive knowledge elicitation in developing wrappers for KDD

Workshop Audience

This workshop is intended for researchers in the area of machine learning, including practitioners of knowledge discovery in databases (KDD) and statistical and computational learning theorists. Intelligent systems researchers with an interest in high-performance computation and large-scale, real-world applications of data mining (e.g., inference and decision support) will also find this workshop of interest.

width=22 height=22>



IJCAI-01 Workshop on Wrappers for Performance Enhancement in Knowledge Discovery in Databases (KDD)

[workshop code ML-5]

Saturday, 04 August 2001
Seattle, Washington, USA

Workshop Description (Last updated 05 Dec 2000)

The rapidly increasing volume of data collected for decision support applications in commercial, industrial, medical, and defense domains has made it a challenge to scale up knowledge discovery in databases (KDD), the machine learning and knowledge acquisition component of these applications. Many techniques currently applied to KDD admit enhancement through the *wrapper* approach, which uses empirical performance of inductive learning algorithms as feedback to optimize parameters of the learning system.

Wrappers include algorithms for performance tuning, especially: optimization of learning system parameters (*hyperparameters*) such as learning rates and model priors; control of solution size; and change of problem representation (or inductive bias optimization). Strategies for changing the representation of a machine learning problem include decomposition of learning tasks into more tractable subproblems; feature construction, or synthesis of more salient or useful input variables; and feature subset selection, also known as variable elimination (a form of relevance determination).

This workshop will explore current issues concerning wrapper technologies for KDD applications, including:

- problems to which wrappers can be successfully applied
  - dimensionality reduction
    - feature subset selection with irrelevant variables
    - feature extraction
  - other change of representation for KDD
    - transforms
    - ensemble learning (mixture estimation)
  - empirical performance tuning and experimentation

8/19/01
tD

"WRAPPERS"

.00 : The term "wrapper" originally used as a means of facilitating communications betw. heterogeneous "Modules" of a larger System. Each "wrapper" converted t. language of t. Module it "wrapped" into a common format/language, so it could communicate w. other modules in t. System: An "Interlingua".

In the IJCAI 2001 conf, "Wrapper" is used in a different way. In t. words of HSU: "It uses performance of indicative running systems as feedback to optimize performance of t. system."

Any kinds of "tuning" modifns are permitted.

This looks pretty much like what I have in mind for CPM! — T. Q's: Do these guys know how to do it right?

.40 Some Refs: To get info on next Doc Google : Wrappers | This brot up many Candy wrapper collections, but also a paper dealing w. definition of .00

W. Google wrapper, KDD was best mainly for IJCAI 2001 conf, but also also an earlier one in 1998.

W. H. HSU was organizer of t. 2001 workshop: He has a website:

kdd research.org /KDD/Workshop/IJCAI-2001/

HSU is at U of Kansas; Dept of computing & info sci; & Has papers there on wrappers.

.30 .21 SN As General Problem: Try to Define just how one would go about teaching my CPM ~~what~~ t. Definition of an OZ problem. One could give many examples, but would TM "realize" that's good ones often, if not t. specific Teaching techniques in t. examples? One way to be t. idea of "coding by consistency" — of Giving some constraints, ~~Reinforcements~~ — which narrows t. object down to "1 of N", Then given N.

But also t. idea of "Best Value w. B $c_B = c_0$" is diff't. Maybe first teach what a "Best value" is, then "Best w. $c_B \leq c$."

T. advantage of TM really knowing what "OZ" means is that if we find it to do OZ probs, say by [illegible] several other Methods, then TM would at least make it possible for TM to generalize & find other, possibly better methods of ~~OZ~~ soln. of OZ probs.

.30 Perhaps Next (easier) How to teach TM what a "INV problem" meant! — That it has to find soln. w. min CF.

A perhaps useful Approach: Given a solution possible "solns" ~~that should be~~ A* & A'. both find soln. to INV problem but A' takes longer — so A has relatively more probability of being a soln. for INV problems. Probly ~ approach could be

.00 : 338.40 used for OZ prob. deDn.

Normally, all poss. A^{**} values have pc > 0.

So: At present, T. system proving system I have in mind is not Bad: It must conceivably be good enough to be able to be very smart using a conventional TSQ.

This CPM works fully. way: (1) It has a set of IA's: — Each of which is able to work $Q_i A_i$ sets, but often only certain kinds of such sets.

(2) ~~Then~~ T. initial Q A set is inputted, telling which

328.24 — 330.08 Covers most of the operation of this preliminary system, w. some ideas on how to continue its & some general criticisms. + problems $(Q_i A_i)$

.10 for "+ Report" It would be fine to just describ. the set of IA's & T. initial indexing of problems by USER; T. type of indexing by FC) is finally, basic general ~~app~~ Core to FC)'s operation.

This creates a CPM of some power, but can be further improved by use of context, i.e. (eng. doing Lisuch.

Further development: whatever goes CPM to "understand" better is better, & T. true depths of ~~IEW~~ IW & OZ problems — what "Optzn readymeans, etc (338.21ff)

+ from 328.24 - 330.08; We start w. these IA's will & we get FC) to learn to select proper ~~ones~~ IAs for specific problems. At a ~~the~~ more advanced level, FC) will start of .20 "create" new IA's because existing IA's would have many input- "Initial params" that characterize them. Later yet, the actual IA's will be described as data of a stack grammar (~~String~~ for (T. construction of this grammar will probably be done by a USER/trainer) so FC) can construct new ~~IA~~ IA's more appropriate to new Q's.

At another level of development, O_n can be used to ~~update~~ update or modify T. updated/syntax of one or more of T. IA's. ~~For case of~~ for T. IA $A \geq 1$, This could involve calcn. of pc's of things based on (local or (greatly) extended contexts. Also learning (even could involve O_n: $O_n(O'_n, Q_n A_n, \cdots) \rightarrow O_{n+1}$.

.30 Thus the "General possl. context" is usable in A relatively non-el. form of TM.

SN In T. past, my Model of human prob solving, on which T. FC), [IA] interaction was based: That a Human would look at a problem & from his experience, decide what kinds of prob. solving Methods should be tried on it. — This seems Quite different from e.g. FC), [IA] systems. In T human system, T. Prob.-Solving methods do not learn much from being used. In T FC), IA systems, T IA's are designed to learn. Yet, I still have that concept of "Stable IA's" that never did (nor anything of that kind stopped Learning.

340.00

ID

Abcdefghijklmnopqrstuvwxyz

$A B c d e f g h i j k \int_{-\infty}^{+\infty} e^{\frac{x^2}{2\sigma^2}} dx = \sqrt{2\pi}\cdot\sigma$

Abcdefghijklmnop

.00:339.40 : Hvr, t. problem solving "spectrum" of an IA changes in time. — So F() must adapt to this fact.

.01 The BIG Q is: As I "work" t. TSQ: will I be able to express all t. hours I need in T. existing CPM format, or will I need really gross modifns of it? My impression is that Lsrch w Irng" can accommodate almost all hours I can think of — Hvr. "Lsrch w Irng" simply means Any kind of Modifn of t. P.D. (guiding Whether one uses Lsrch or not!).

T. Answer to .01 seems to be ≈ "Yes": So it probly worth trying!

So t. present state is fairly Optimistic! — That practically any hour

.10 Should be discoverable directly by AZI or By On. Hvr, I'm not at all sure about how this is supposed to find them!

I guess (. idea is that any hour will amount to a cond'l. p.d. — t. conditions being t. problem, t. previous Irng in Thanover & (traces of previous attempts to solve that problem. Any cond'l. p.d. of this sort is probable by t. system. [Pro AZI as is, probably M d.f.'s, I feel I should be able to devise a suitable practical (as opposed to fr ≥ impetuous)] model of a stock P.D.

.20 A possibly Serious deficiency — No "look-ahead" is implemented — i.e. some hours probly use "Look Ahead".

It may be possl to insert Limited Look Ahead — say start Horizon. When I'm doing TSQ's see when I need hours of that sort & then try to (expand modify) t. language to include it.

Perhaps nec. & suff conds for Complete TM (as a CPM machine)

1) TM_1 ("HC") must be able to express all poss. H(Q) → [A] d.f.'s (H can be H(Q,R) rewrite or anyother form).

2) TM_2 must " " " express all possl. hours in t. search for H'. (H' is dem of H — as calls search)

.30

.31 : 331.32 SN A comm kind of P.D. (for strings or Numbers related to 331.26) — Clustering: TM outputs one or more (wtd) cluster centers. Also a matrix giving how Pc & w. "distance" from center. Usually only one matrix, but each center could have its own matrix.

Actually, Pro & any mean & variance (μ & σ^2) is an example — but t. objects are numbers, rather than strings.

An interesting case would be ANN, in which t. output is 2 variables μ & σ^2, that describe t. d.f.

8/21/01
ID

"Trivial" Extension of convergence thm for Inductive BMGs (≃ unordered (finite) strings)
↓ naturally, hrr.

Convergence Thm for Stock Operators:

:00 Say P_{univ} is a stock operator $H(Q_i)$ that enumerates a PD on P_i's for every Q_i.
Say Q_M is the universal stock op. Then if H is of dominance K_H; (in bits)
for any sequence of Q's $[Q_i]_{i=1}^{n}$, the total expected error^2 of pc of individual bits, is $< K_H$.
If H was continuous params, K_H may be a funct of just what seq. of Q's was used — & how many Q's.

:08 It would nice to have a Prim-recst Heuristic in this vein. Vam
I'd like a general formalism for heurs.
:10 Perhaps these such problems are of 2 types (2 ways to decrib. what is being sought)
1) INV ② OZ. (NEW idea! for OZ problem Conds could be partially ordered (?)).
A theor is a funct. that looks at a such problem decrip, & outputs one or more ways to do it such more rapidly.
So this does look like an induction problem. We could give TM many examples of problem/theor pairs. — But basically, we'd like TM to really understand what the problem was — that it was to do sth of such.
So if TM "really" understood what the problem was, then giving examples would not result in a simpler induction soln; it would analyse the examples for "hints" on solving the "TRUE" problem.
:30 Perhaps the Main problem is to get TM to "Understand" what an optim problem is.
In General "Understanding" involves a potentially ∞ of ideas! e.g. all the ways of comp Something.
For optim. to start off, TM would know which one kind was better than another.
It would also "understand" the "Better than" relation: transitive, ~~reflexive~~ non-commutative, (assymetric)
not irreflexive $a > a$ is never true. If we want to know what "$\geq$" means also $<$, $>$, $\leq$, $=$ etc.
→ If a is 10 times as fast as b, & c is 100 times as fast as a then c is 1000 times as fast as b.
In some cases, there is a central/critical idea to be "Understood"; & then understanding of others derivable from it by "pure logic".

My General defn. of "Understanding" is that it is a way of "explaining" something that is predictive. Its a predictive Code of the Data that is "understood"
:39 Hrr, just as Humans can be made to "understand" things in whatever seem to be the described way, it should be possible to get TM to understand things in similar ways. I assume Humans (in to understand/interpret language, by using normal (ALP) induction & so we should be able to get TM to do similar inductions.
Perhaps a good "STUDY PROBLEM": To take various problem defns designed for Humans, & find an (inductive) way (a TSA) to get TM to understand these problems.
T. Simplest (given ALP) is: to/find $x \Rightarrow M(x) > c$! T. problem is to find x. no maximum of c.

8/22/01

ID

.00: 341.40: One way for TM to understand & cc-free INV prob. of 341.39!

It just tries "x's at random" to see if M(x):c. It has no hours. —

A much better way, is Lsrch. It is a more arbitrary way! Also "trying x's at random" does not define a method, since some tests may take Time = ∞(!).

Anyway, The next step is "heuristics"; to method do a "better" guessing p.d. than $2^{-|x|}$. in the sense of less cc. (or just less "Time").

The next Thing is for TM to find "better hours". — We seem to have gotten into this collapse of all hours being modifications of p.d. — & they are certainly more than that.

Def A no-look-ahead (L.A. = Look ahead) hour (No LA hour or LA hour) would involve good PC offline solns for Lsrch; Use of PC to get next trial, run time of previous trials. And somehow, realizing that was a "good pc" also involves small cc. So we want a p.d.

.10 That gives max $\frac{pc}{cc}$ to solns! cc is time needed to generate trial; "Generate time" is main part of the PC def'n. & characterization.

.13 [So: Good hours find pd's that give max $\frac{pc}{cc \text{ to generate \& test}}$ part-solns.

How .13 works out when one is "Thinking during Lsrch" is Unclear.

Again: One first level (TM_1) TM_1 must recognize soln. of problem & also how to recognize when one soln. is "better than" another.

On second level (TM_2) TM_2 tries to find methods of working problem that take as .20 — good as poss'l. This involves trade off between total cc & total pc. — for the USR has to specify just what the methods is: say f(cc, pc) is a linear ordering of

.22 — (cc, pc) pairs — to tell which is best.

My present Orientation is to have TM_2 working very early in the exp. That any discovery of Hours, or just simple context dependence of pc's of cores would be a "TM_2" problem.

Probably, it should be diff't. to teach TM the meaning of INV & OZ problems, by giving examples. First cc-free INV: Give examples of f(), c, and x, so it could tell if any particular new G(), c, x was a case or not (MTM problem).

.30 Then, looking at the traces of several solns to INV problems, teaching which traces had less cc than which other. — the concept of "less cc" as a relation between traces.

The concept of shortest Soln. of Min cc.

So, I think it may be poss'l. to "tell" TM what we want it to do. Hvr, since there will be only 1 or 2 situations in which we need this critically, we might as well program it in! The needed concepts are cc-free INV soln., cc-min soln of INV prob.; perhaps find Optimum(cc, G) in OZ problem. Given a user supplied (cc, G) linear ordering function (see (.20-.22)).

ID

"D Mode" Deterministic.
i.e. "S mode" stochastic

.00: 342.40: Hvr., the main Idea of "In Theory" teaching TM the concs. needed to understand what it's all over Conc is, by a "theoretical, large TSQ" — This is to put the Conc. in proper form to fit in w. other things it runs.

So: It would seem that for any of TM_2's well defined problems TM can be taught to understand & work on those problems. W. suitable TSQ's TM_1 can usefully/successfully work on TM_2 problems.

For a immature TM_1, TM_2's problems should be elementalized so that

.10 TM_1 can easily work on them. A more mature TM_1 can work on the full, non-el problems of TM_2.

So, it looks like I have a complete form of TM. — Good enuf to start on TSQ: Write report,* w. special # for items that I have written about that need expansion! Give refs to my notes.

Put indices in "Original report" Then remove the indices for the copy sent 99 1 ; 99 2, etc.

"99" indicates index.

Say "remove all 99****." or whatever code is. There are other stylechgs.

.17 do IDSIA. (There's probly work'll be more than 10 or so of them.

341.08 – 343.17

A recent Breakthru: was the realization that it may not be difficult to "teach" TM the meaning of various "commands" — things that I want TM to do: (like solve a INV, or OE problem ... Get a soln. w upri cc or max some funct of cc's pc, etc.

.20

A reasonable format: Find x such that F(x) = max. Actually, this is an example of a function : the arg. is x, f() : the output is a value for x.

In learning how to evaluate this function, TM could examine F() : what kind of function it was: Did X have to be numeric or string? was F() continuous? Did it have a max slope/curvature? If X must be a string? — Ask what string "better" also.

Find x such that F(x) = c || Find x such that f(x) = Max in ≤ time T

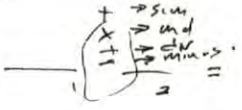
.30

TN A reasonable way to teach ANL: from several examples of 2+1=3, 3+5=8 } each. TM finds Math in — + — = —; the format — + — = gives the soln (—1, —2 sum).

In a similar way, —, ÷, x are solved.

From these 4 solns, TM learns indices of the whole soln.

There is much redundance to eliminate compression.



Space 349.16

8/23/01 ID

.27 / .40 — Simpl MTM (stochop) Model: How to realize it / How to Actually use this Model in a TM → .27 — .40

SN

.00: 342.40: Why not "perm" solns to problems "by hand"

Say one has already factored a soln. as well as seems possible. One "gives" TM a soln. in terms of "factors" & = cases" it has already acquired. What's wrong w. this: Saves a lot of time! Well, by getting TM to solve the problem itself, we find out if there are any factorizations that are better than the ones we've given TM. If there are, it is possible that we, manually have acquired the soln. using those factorizations, & by not getting TM to work the problem in the right way, we are not giving our pc's the factors for future problem working.

If it involves 3 or 4 cases, each case may have its pc doubled. The set of cases used, may be a useful set & have a rather low pc otherwise. So the combinations of 3 or 4 cases as an input case could have its pc ↑ tremendously! — which could make a lot of difference in future pro & -solving. Main Argument

.10 One oft. Big dangers in writing tsQ's is that one will omit input cases, that one's not aware of. I somehow feel that this is more likely to occur if TM does not give me Big decisive input/cases. "Logic anomaly" (i.e. if they are given to TM). The detailed reasons for this idea are unclear (uncertain).

.16: 342.40: 333.29 in GA. HA!! Away to use GA for a IA: The O_n is slightly to viz GA. At all times, we have this population of O_n^i cases. When for corpus $\{Q_i A_i\}_{i=1|n}$, $n \to n+1$, we continue w. this population to look for a good O_{n+1}.

.20 Of the cases of O_n we have for $i = 1/n$, we try (Q_{n+1}, A_{n+1}) on them to see how we get as "O_{n+1}^{i}".

When we (mutate/crossover) we get new cases: we can do quick statistical tests on various $Q_i A_i$'s — perhaps find a sequence of $Q_i A_i$'s that are particularly good in discarding bad cases most rapidly!

Meanwhile TM's can be watching this GA & will be productively discovering ways to improve it! Better mutes, crossovers — perhaps combinations of K cases to produce a new case or set of cases (prob w. $\binom{n}{K}$)

We could actually stay within GA (no other IA's) for a universal CPM! → Also see 345.00 for a different way to get stochops out of GA

.27 However, it's not clear how this GA form of CPM would do for stochastic QA's. — In .16 ff, this may work fine for MTM problems using 3 AZ1. I could, formally use the universal stochastic operator of an Umc w. 3 inputs (O_i', Q, R), but I don't really know yet how to use this to express most common forms of stochastic instructions. I can do it, but practical induction.

.29 always in a very formal way. I may be able to fix it up so it's closer to practical induction.

I could use a MDL approach, in which we use the shortest R that will give A_i' — Still, it would seem to be very time consuming to find a suitable R for each Q_i! — Maybe not so bad, since R is usually only a few bits (since the pc of A_i' is used by fairly by — if not, it's a bad Kit!) Also, we could do a MC version (on R's) for the entire corpus wrt O_n, & get a reasonable approx. value! (Not so easy! — one has to put in random R until we get A_i' as output) If we don't do it in 20 trials, we give up & use a "default soln"?

.36 .37 (See 331.26 & 340.31) Both relevant to stochastic IA's but 331.26 is most relevant to .36–.37 (the default soln).

.00 : 344.00 : In t. GA EA model: we could use deterministic models only; the Gene for each card would be t. no. of correct responses to set of QA's. — or perhaps a wtd sum. To weed products, we used wtd mean of t. cards, wt. being a fcn of "no. of correct responses". Each card has a pred for each Q_i: for each Poss A_i^j we add wts of all cards that gave that answer

We may also score as $\frac{1}{cc}$ per response or

.05 Give a score [illegible] $\frac{1}{cc}$ for t. entire corpus.

Some time back, I had had a proof that [illegible] was a universal d.f. — but my counter example was wrong! It may be able to give a universal d.f. — but whether it's a "good" practical model, is unclear!

.07 In fact, it is easy to prove that for any seq. of Dfi cvs A_i's, ∃ an ensemble of fixed sequences of deterministic choices for each A_i → t. mean of ens. ensemble = t. desired stochastic d.f.

To get say a particular member of t. ensembles A_i; just pick a random A_i^j w. pr = P_i^j.

.10 The mean of 6 ensemble will give t. rtn pr's for all A_i.

Does t. current ensemble model of .07 – .10 suggest anything about how to get such an ensemble empirically? — (From data) — it would seem not: which is perhaps why it might be a bad model!

In GA, we certainly wouldn't worry about BLOAT — since t. Gene takes care of that.

The param to be adjusted is population size: would this have to grow [illegible] much as corpus size ↑?

Since I am now, to some extent, what kinds of modifns of t. GA are needed, I should be able to design better crossover/mutation Algms. I will be trying to implement t. A_i's distribn. over functions. — However, I would want more context dependence & modifn of genes' R.D. & by "Irng". If I can't implement these in GA, it will be much weaker than t. original model — that starts w. A_i's & then uses context & "Irng" during Lrng to improve them. In GA, I'd somehow have to find a way to implement context dependencies & Irng.

.20 There is an automatic variety of "Irng" in GA, in that cards of low Gene are less likely to be used to generate new cards.

How I'd get context dependence (e.g. O_{n+1} [illegible] and D.F. would depend on Q_{n+1}, A_{n+1} — also various past O_i, Q_i in A_i's → Q_{i+1}, [illegible] data.

.26 ANN [illegible] (RANN): In my own experience this would be for numerical I/O :- (No ANN has been used w. strong data & graphics.) = On

Anyway, t. state of a system can be represented by a pt. in wt. space. If t. no. of wts is minimal, we want maximally fit soln.

.30 New data, then would try to move minimally from old position in wt. space.

[illegible] When w. weak priors, we should retain many "near flat" pts in many. As we make trials of moving from old pts, we could bifurcate & generate new good pts — also discard some not-so-good branches. → 346.00

.34 If we use many wts, we again should retain many pts in wt. space, but as we drift, we will, because of too many wts, end up "overfitting". Normally, this is dealt w. via "cross-validation" (also fearful) but I don't see how to do it in present case.

Back to small no. of wts: we could try using different sets of pts in wt. space — → 346.00

8/24/01

Fri **ID**

$x = 0 \text{ to } \infty$

$y = 0 \text{ to } 1 \quad = 1 - \frac{1}{x+1} = \frac{x}{x+1}$

$\frac{dy}{dx} = \frac{1}{x+1} - \frac{x}{(x+1)^2} = \frac{x+1-x}{(x+1)^2} = \frac{1}{(x+1)^2}$

.00 (space 345.34) : [redacted] **so we** would simultaneously track several (many) distinct wt. sets in wt. space.

Note: For each "set of wts", there can be > 1 pts of Good fitness & yield.

Each "set of wts" is a subset of possible connections betw. neurons; & for each set

there is a subspace — w. one or more pairs of Gore — each w/ its own Hessian —

so $\frac{\text{Gore}}{|\text{Hessian}|}$ = true Gore of that pt. — The Gore has to be a product of pc's of eigs

times pc of Model. (pc of model is at [redacted] | Hessian |$^{-1}$ — if all eivals of

matrix are >> 1 — if any eivals are small, this amounts to reduction in dimensions —

Too [redacted] many wts!)

TM₂'s job in ANN is to add/remove Neurons/wts also to give initial values for

.60 new wts.

Forth

Fo: outline of report.

1) Intro: [redacted] part I wrote; Also add stuff on general pattern recognition system.

2) Details of what IA's are: Give list; perhaps more detail on some showing

how they work.

3) FC) how it [redacted] initially chooses IA's by indices; later learns which IA's to use.

4) [redacted] Updating: a) Pruning: remove FC) & update(?) the GA

b) Use of On to do(p) update: [redacted] context (6) long dynamic task: ← for AZ1

Models of [redacted], population size.

notes & can be usually cross Models — so it no longer looks like GA/

[redacted] has a IA?

SN **For AZ1**: GA can be used to update IA's because IA's update; CosANN [redacted] 345.26 – 346.10

has some [redacted] (does how) ANN can be used to learn numerical (vector I/O) — other is an

.20 input subclass of IA. 345.26 – 346.10 discusses updating a Gore for it. update my

So that for IA's I have AZ1, GA, ANN, (see recent on-line Complexity paper (c:\ps)

On ANN/GA for some how-does on Pt's.

The beginning I wrote, was 332.01 – 333.13: "If dark & unknown system"

$O_n(O_n', Q_{n+1}, A_{n+1}) \to O_{n+1}'$ also Gore for O_n.

Next, dark. IA's, & general (dark. — say "O_4 is a IA!"; We will dark One or Our IAing

same detail (AZ1): GA T. ways in which GA & ANN/(RNN) can be used as IA's

are dark in Appendix. How new IA's don't have to be separate + on All inputs. Also, how will

add/new IA's to system based on our future experiences w. TSCP's.

O_n is not an IA, but it. A(n) Pat generates O_n & not a. A(n).

generating O_{n+1} from O_n does constitute an IA.

.30 A linear or non-linear "curve fitter" is a handy sample IA w/ a simple updating system.

When given a set of $[X_i\ Y_i]$ pairs ($i = 1(1)n$) then given new X_{n+1} it will give a Pt on Y_{n+1}.

Linear & non-linear regression can be very adaptive types of IA's. One way to do this:

$Q_n = X_n; X_{n-1}, \ldots X_{n-k}$; $A_n = X_{n+1}$ The $[Q_i, A_i]$ data set can then be obtained from

any time series.

AZS stochastic
AZD deterministic

.00: 333.13 Section: Induction Algorithms:

An Induction Algorithm (IA) is an algorithm that is able to look at a sequence of N (Qi, Ai) pairs, like the system of 332.03 and, in reply to a new input Qnew, give a probability distribution on possible values of Anew.

IA's vary in their efficiency and accuracy. They will also differ in the kinds of Qi objects that they will accept. Some accept only strings, others only numbers, others only vectors. While IA's must give probabilities for all conceivable Ai values in their range, some are deterministic, and give only one Ai value for each Qi, with a probability of 1 for that Ai, and zero for all others. Accordingly

[Note: 331.26 tells how to convert any deterministic IA to a probabilistic IA]

We may view each IA as a kind of "specialist" in a particular area of prediction.

.10 For our preliminary approximation of the system of 332.03, we will randomly choose a user and weighted mean of the predictions of the current IA's. At first, the trainer of the system will assign one or more of the current IA's to work particular problems in the training sequences. Later, the system will learn to assign IA's to problems without the aid of the trainer. → 361.24

.16 The I.A.'s that will be described in most detail are based on universal functional languages such as Lisp or Forth. We will also consider IA's based on Neural Nets (ANN or RANN) as well as IA's based on Genetic Programming. The last two will be described in appendices I and II respectively.

Commit

.20 We will describe in most detail, two IA's based on functional languages such as Lisp. One of them (called DAZ) does deterministic prediction, the other, SAZ, does stochastic (probabilistic) prediction.

IA's based on Genetic Programming and on (Neural Nets, both ANN and RANN) will be described in Appendices I and II respectively.

Many different kinds of IA are possible. During the course of our training of the system, we expect to be adding new kinds of IA's to the set used by the system, and discarding IA's if their functions have been completely emulated by other IA's. N.B.

.24

.26

.30

Insertion 361.26

361.24 →

.00 : 347.40 : A way to get 2 ideas Across : ⓐ Idea of ↑ generality of what CPM does ⓑ Idea of what various

IA's can be (= specialists / consultants / experts in special field)

This would be a listing of some IA's

.03 1) Curve fitter : linear or N-linear [redacted] Vector → vector $z^0 + \sum z^1_i x_i + \sum\sum z^2_{ij} x_i x_j + \cdots$

or AANN.

.05 2) [redacted] regression. Inverse or non-linear : if input vector $\vec{x} \in Q_n$ [redacted] $z^n_i = x_{n-i}$ $(i = 1 | k)$.

.06 3) Classifier : (e.g. ID3) The i'th data set is $Q_i = x^i_0, x^i_1 \cdots x^i_k$; $A_i = Y^i$; Y^i [redacted]

.08 [redacted] is the class discriminator for Q; vector → .34 ← objection to classifiers as examples of IA

.04 4) Theorem Prover : Corpus is set of theorems and assoc. proofs. Q = Theorem, A = Proof of Theorem.

.10 Some theorem provers are "Global" ; they do proofs [redacted] from primitive Axiom set.

As incremental TA we will be [redacted] primarily in creating & incrementing Theorem provers :

that use proofs in previous parts of ↑ seq.

Theorems can be in Algebra, Logic, Geometry or any other part of Math. → 350.10

.13

.14 SN In regular GA : on "crossover" If we have 2 "parent" function trees ; To create third tree for next ↑ pc of ↑ 3 is minimal ; or create a "third" as a seq. of polys in order of pc.

In ordinary Koza crossover : swapping sub-trees is coded as ;

tree A sub-tree A' B tree B's sub-tree → $\frac{A}{B'} \cdot \frac{B}{A'}$; We similarly as A, B', B, B' are as defined in pre corpus.

so ↑ 3 examples are : A, A' ; BB' ; AB' [redacted] (say) : In each case we have to [redacted] just how ↑ connections are to be made : i.e. What "terminal" of A, A' is

.20 connected to ! A simple way to calculate $\frac{1}{pc}$ would be ino. of terminals on A.

In a similar way, on can code 3 and parents → several [redacted] offsprings.

At this "Level" this way of looking at things doesn't generate alts much different from normal Koza crossover of pairs of parents : I think a v.g. part is where we define common sub-trees of the Gore cands. It may be possible to get this & check w.o. hunting for common "subtrees" which sounds expensive.

A possible way to get grammar that generates cands w. assoc expected ∈ to pc ; We generate cands with some simple grammar ; We divide ↑ cands into maybe 4 percentiles w.r.t Gore : for each of ↑ 4 we try to find a separate grammar. The 4 grammars will have certain common elements. — This will be ↑ "Mother Grammar". The 4 offspring

.30 grammars will be snobiously addressed to ↑ "Mother". It may be easy to parameterize ↑ grammars w.r.t their percentile no.s {0, 1, 2, 3}. → 350.24

.33

.34 (.08) Re : Classifiers : T. [redacted] theory says $[\vec{x}^i, y^i]$ is always very limited ; it is assoc w. one particular class.n. T. info from one classifn. is not used with next classifn. problem. —

So in this sense, ∈ This kind of classifier is not a IA.

.37 In fact, examples 1) (.05) 2) (as) 3) (06) all have this objection ; ↑ from power of 4) (.09) is o.k., tho. We may be able to use them for updating improvements, tho, since we always have ↑ same problem in this case. → 349.06 [redacted] space

.00 : 3 43.40 : [redacted] In any of these systems, we can grossly modify them in a discrete way by changing the "basis" of the system [redacted]
In curve fitters, used frequent "Basis" functions, or else a difficult kind of squashy function in ANN.
(Note that "Hinges" are very fast to compute — they have discontinuous first derivatives ("piecewise continuous" if scalloped second derivatives.)

.06 : 3 48.40 : While this is true, we may get much useful recursion" (i.e. having the system improve itself.)
if we always use the same "IA" for updating, & it is not linked to other problem areas.
Consider a curve fitting IA! It could work on many different curve fitting problems — each would have a different "knock no.". While each curve would be fitted individually w. its own "update parameters", [redacted] params for all of the curves could have certain features in common — & these could be "updated" in a more "global" way. So this particular IA would have a "individualized" update algm.

.12 [SN] An alternative to having all of these IA's in it, selected by FC() for a particular problem — would be to have individual IA's as better, but have them be "CALLable" from the "Mem pgm". In the present scheme, it might be possible to do "calls", so that AZ1, say, could call [redacted] to solve one of its sub-problems: This overall would then be the FC() which [redacted] would decide which IA's to assign it to.
In Sol 86 I had this "Planner" heuristic that called various prob. solving routines.
— Perhaps "in the spirit of" .12

.20 [SN] A kind of sop I'll need for Hearts (I think): A value of the type should be discoverable! In problems of type a (Cl. in list x); If class B conditions occurs, then response of class Y is helpful w. problem p.
Example: In symbolic/integration: When integrand is of form $U V'$, then int. by parts is usually useful. I think this sort of stuff rules can be fixed up by correlations / statistical studies. We use ALP to see if it correct is "useful" — i.e. a compressive rule.
Long ago I had this idea of OP formalism, & studies of correlations between obs & ops.

.26 Another Go at correcting: One makes a certain Ob out. Q_i & gets a certain result. (to be used for decision) This result correlates highly with (one basis to predict result) w. A_i or certain features/parts of it. — So this is a compressive Code. How could such correlations be found using a 3-part model of operators? $[M_{\mathcal{G}}(O_n, Q_{n+1}, R) \to A_{n+1}]$. ?
.29 .30 .31 Is it form: If $\alpha(Q) = a_i$, then $[A_i]$ is probabilistically limited in 1 of 2 ways: i.e. D.P. on A_i. — This may amount to a p.d. on an aspect of A_i', as which, coupled w. an aspect on A_i gives us an aspect on A_i'
How [redacted] (29K) — .31 is related to formalism of (.29L) is unclear!
Actually, it's not clear how this can be put into a usable def. on A_i — let alone get it w. d.P. in the (.29L) form. ! So, TM not only has to find "predictive regs", but it has to find "predictive regs" that are expressable in a useful form — that can be used to help get a p.d. ← (in some useful form).
→ More exactly, TM has to find obs on Q_i that can control operators that construct A_i'
As this IA matures, it gets better obs for Q_i & better control operators to construct A_i'.

8/25/01
FD

3.50

Hard v.s. Room
Criticism
R. H. problem

"Eliminativism" .35

Several experts in same Room v.s. " " " " Head.

↳ The Rest's not exactly it. The Room v.s. Head dichotomy .35

.00 : 349.90 : Other than TSQ design, the main conceptual problem is improving & updating AIXI.

.01 This means finding contextual requirements/dependencies for concs used in modifying O_n to create O_{n+1}. As TM matures, these contexts will become larger. Noticing a very small set of successful/failure of previous trials coins to problems, is also an important thing that has to be learned.

.02 One trouble is that I have no way for AZ's (both to store or AZ2 to notice or even express regularities or structures that are needed for ε .01-.04. — The 349.26-.40 is a step in this direction.

Essentially, the way I'm dealing w. this unknown is to allow room for lots of "IA's" ("specialists") that are supposed to be able to do this job. .32

.10 : 348.13 On examples of (IA's) that "use continued irony":

1) One that runs Algebra: New Defns, new Rules, new problems.

2) Same as 1) but for other areas of Math-say Geometry.

3) Learning to work symbolic integration/both definite & indefinite

→ 4) Learning Q answering in ordinary English. Assume that TM has already found much basic (e.g. Algebra).

5) Regression of Time series: Each TS is a QA. but give TS's set more & more bitty — w. more & more "side info".

→ 6) Improving & updating AIXI (see .01-.09)

7) Trying to solve alg. problems symbolically — non-linear types of greater complexity

8) symbolic solns. of diff. eqns. Ramanujan

9) Go thru Rubin & text Ramanujan use. (Summary of Results in Elementary Math: Carr.)

.20 → 10) Learning Logic, applied recursion — applied to Math problem — able to apply

Recover Basic logic to practical problems in Math & other areas.

.24 : 348.33 Use of GA v.s. AZS heuristic for Updating.

1) GA is standard, technique; doesn't have to be developed much. — AZS is heavily library-esque.

2) GA appears to be capable of very non-elitist — AZS seems to be essentially elitist — (but I'm not sure it really bad!)

3) AZS may deal more directly with need for context dependence of concs. in updating — Alt. it may be able to introduce irony from previous trials — & I don't know how to do this (yet) w. AZS. Stochastic AZ!

.30 4) GA automatically returns Many "good" codes & remarked "not good" by trials usually moderately AZS (but could be). → 351.22

.32 (.09) : A critical idea here, is that IA that works on update improvement after bugs as part of its S.C. (subconscious — data set — TSQ), "main line" problems to solve that will help it to run (TSQ) to solve IA update improvement problem.

.35 IMPT Concerning What present Model does not seem to do: is mix ideas, concs from distant IA's. A possible way to deal w. this: There is a "main IA" that watches other IA's & tries to incorporate their methods into itself. — however, it has to "understand" the methods before it can usefully do this — i.e. the "external experts" operations have to be factored into concs. used by "main IA" but for the "main IA" can adopt them as its own. Main IA could regard other IA's as 351.00

The Rth problem (experts in same Room v.s. Expert in same Head).

.00: 350.40 part of t. "advice channel" (.19)

.0' HA!/ (normal) ANN (and GA) as IA's (in t. long term lrng sense). → see (.22) Clam for ANN, each problem, w. its data set, is regarded as a "Q". What t. ANN learns, is how to take a Q (≡ problem descr.) & decide on how many & which initial wts to use &/o how to introduce & "prune" wts. during t. lrng. Also how best to use momentum, (not the exact) steepest descent calc. for solns. (actually t. Newton method using second derivs. would be better than steepest descent — this is t. standard ☆ Non-linear Optzn method ☆ — but for >100 wts, cc ea N^2, so it may not be practical. — is' there a way to use a simpler matrix, e.g. Toeplitz, that would be ☆ solvable w. cc $= N^2$? (By t way, Hessian usually only uses first derivatives! so it would use/remind as "steepest descent." — I think would have to use info averaged from last several pts (may be x winds) in order to get ~~usable~~ matrix not of $f_i \cdot f_j$.

.10 — But rather ☆ $H_{ij} = \sum_k f_k(i) \cdot f_k(j)$. ← This is to be regarded as an axiomatic "Newton-like" computation! An adjustable param is t. "width" of t x wind ☆. If may well be that t. system uses "second derivs" directly at any pt, but when ex winds (.22) average over several pts, it does a kind of "steepest descent" calc. (Some evidence for t. Hessian to guess where t. peak is. →.22)

.14 (.00) Actually this I am trying to understand is use "external experts" can be an important aspect of TM's mainly. We introduce "expert IAs" into t. system, a' makes them as "open"

.16 as possible to "dumb IA" (≡ "MIA"). MIA ☆ Plan: Mike to ☆ do as well (or better) than M. experts, internally, by expressing new methods in its own internal lang, & also using parts of new methods ~~of~~ d/o modifications of them, to solve t. same problems at least as well. It may be that there is a natural GORC for this involving

.20 usual t. ~~wanted~~ Pc of (corpus + operators) dcm. ■

.21 On external experts → 352.00

.22: (JGR), (.01) 350.31 Re: GA as a IA: Like ANN (.01 of it but different)! GA would work (industrial) problems in essentially: each would be a "Q": what it does for continued lrng is: for each Q, TM learns to design a set of mutations, crossovers & initial populations appropriate to that problem. (I think t. usual "wrapper" approach does some or all of this, but TM should be able to do it better) — In particular, t. "Initial population" can be a v. good guess at what t. soln. is like. I noted some time ago, that this would perhaps cover critical weaknesses of GA systems.

Another critical weakness is excessive time spent on automatically t. "fitness function" — T. System could also learn how to do this better & better: also modify fitness function — Or ☆ given a problem, it has to devise a fitness ~~function~~ function — .30 It could do this w. cc & memory in mind — also modify t. fitness functions & populations .31 moved toward a solution.

So I have essentially ☆ 3½ IAs: AZD, AZS, GA, ☆ (R)ANN.

What about classifiers. 348.06, .34-.40

Each classifn. problem, with its data set, is a simple Q. Hvr. note 348.37-.40 A classifier can be used for updating — it works on t. same problem (updating) all fixtures & parts in new data. 349.06-.11 is another "weakness" in t. use of classifiers in this way — that its not in t. recursion loop of S.E. This is less true of GA & (R)ANN because they can be significantly improved! see (.22-.31) on GA, (.01 - .16 R)) on (R) ANN.

8/26/01

ID

Aug 19 — Aug 24 — Sun.

{ Money, Things, Power, Command; Esteem, popularity, Love; All in "short supply". } — { Understanding, knowledge, "Truth"; Understanding, Insight, knowledge for Pt. Wisdom }

352

.00: 351.21 : T. relation betw. GA & AZ$_D^S$ can be/close! t. method GA uses for mutation & Crossover quite related can be related to AZ$_D^S$'s Update Algm. Align — How O_{n+1}' is to be $\approx$ (Mutation/crossover) of O_n'.

[T. reason $O_n' \to O_{n+1}'$ can involve "crossovers"; But "crossovers" are related to Grammar elements, S-E-trees can be defined on Functions, etc.] GA & AZ$_D^S$ are very close: Eventually will want to Merge Them.

{{ What AZ$_D^S$ can learn from R(ANN) is unclear. — But prob. AZ$_D^S$ can improve (R)ANN considerably. — (If PD is true, then AZS is crazy from (R)ANN ! ← !! }}

.10: 336.25: So t. moral seems to be, from (P.D. on O_n : Q_{n+1}, A_{n+1}) one gets P.D. on O_{n+1}. which suggests that one should retain a lot of O_n codes a/o P.D.'s on O_n "around" various pts.

? A (perhaps expensive) Alternative for Large (retaining Many Codes) is t. Ability to "Backtrack". retaining — also several methods to deal w. OSL

.13: On GA's for updating $O_n \to O_{n+1}$!

One big problem is eval'n. of cands! 2 diffs: (a) calculating Pr(O_{n+1}^{cand}) code!

This can be made easier if we only try mods that don't break any functions defined > the problems also. (∴ backtrack & the problems) (b) eval'n of pr of sols of all of $Q_i A_i$'s! This can be done by sampling. Th. $Q_i A_i$'s for $i \in$

.18 Also by using mut/crossovers restricted as (a) we will perhaps mainly have cands w. steps evaluated for $i \le n-k$. — Then we may find other mut/cross restrictions so (a) is true!

.20

.21 To extent that we conform w. restrictions of (.13 — .20) we will be $\approx$ (AZ$_D^S$)!

.22 Very close to t. such cands selected by normal (AZ$_D^S$)!

((If GA becomes very close to AZ then this will give us an understanding of how best to do mut/cross in GA in General! — since GA has no e.g. General Theory.

Contextual considerations should improve "hit rate" for both AZ & GA.

GA should "learn" from (low score bad) cands as well as (high score Good) cands. — but how to implement this is unclear! Also how to implement Trust in AZ in General, needs to be developed.

.30 (SN) Way to Get Time on Peoples Computers: Use 10% of "Patrons" time to answer Patron's Q's! So they can keep track of how smart TM is being. Perhaps limit briefness to answers "your SOL! ☺". Answering Q's may not take much time: Its "updating" that takes Most of time. "Patrons" can watch t. development of TM as it gives smarter & smarter Answers to Q's.

8/28/01

2 2 3 3 0
8 9 0 1 1
Su M Tu W Th F Sat

353

DAZ.
SAZ.

.00: 352.40 · So: What has to be done on report?

1) Decn of # w/ 4 IA's: AZD, AZS, GA, (R)ANN (RANN may be w/o 1st first 3) (347.24–36 tells how would actively use many more/different IA's & change from doing TDI's Tuning)

For each one, describe how updating is done ~~exact~~ first & how it will be done in More Mature TM.

— That to some degree, TM can work on improvement of update algm when very Huge. ... (not)

Or " avant. input TM can work on improvement & certain aspects of updating.

Begin discn. of 4 IA's as separate/(uncoupled) systems — each w/ its own TSQ — each w/ its own updating Algs.

Next discuss how USR selects TSQ & for each data — but later a different Core issue.

" " " F( ) Is initially trained on this

" " " each update Algm is improved. — perhaps discuss "Context" "Irng"

.10 Somewhere discuss Lsrch for Updating: How it is actually improved by (a) Modifying & Guiding PD

(b) ~~trading~~ the Laziness (Modifying Guiding PD) between trials. Use extra speedup might be on Pts (was spent) as/on the previous trial.

At end, discuss [redacted] "Lookahead" v.s. Guess (≡ Myopic) & Reason. Machines.

Advantages/disadvantages.

Derb. DAZ in some detail, & how it's related to Tmc, a Umc (Alg for PL), & Umc (Alg for PL). (All Lisp functions can be derb'd in PL.)

Also derb. update algm. (Machine except Lsrch or GA can be old for update — Gives a PD or Only)

Then Tell how SAZ differs from DAZ. (How Machine decays differ).

Then Discuss GA! Tell how it differs from SAZ. ~~only in its update Algm~~

Actually, a test: / GA does each problem from scratch: Next, We look for 4 Params

.20 of the GA such (size of popn, initial popn. mut/cross algms...) as a function of 4 problem (≡ Qi).

Updating consists of improving these Algms. (Use of Lsrch for this?).

.22 [SN] It may be poss(& perhaps Easier) to Maybe DZA & SZA & not use & Impl Umc. for SZA: just a set of On's. Hvr, while I had a proof that a/mixture exists to simulate any (finitely derbable) PD, I'm not altogether happy w. this model, (couldn't find proof: Hmm very complex!

The ~~finite~~ deterministic model had for a solu of 4 problem, A_i^* (w probs $P(A_i^* | Q_i)$)

So one has considered this ensemble of deterministic predictors, & all of & wt's; Their mean is 4 desired P.D.

After writing this Report: Write down 4 possl. Choices for each component in 4 System:

.30 As many Reasonable options as I can think of.

1) Listof IA's: I have some lists! Add correlational IA's: 349.26-.40) (These will be very important for Hours & for new IA's suggested by 4 TSQ.

2) Different ways to do Updating: Lsrch v.s. Straight GA.

3) 2 v.s. 3 input Umc for stochastic AZ (see .22)

.00 : 353.40 : **UNIVERSALITY** of certain IA's: If IA_0 is universal, then for any TSP, T

and any other IA, IA_j, the pc given to A_i sequences of T, by IA_0 will > k · the pc given by IA_j. k will depend on IA_0 and IA_j, but not on TSP T.

Wrong!

Q: Do universal IA's exist? Do they necessarily involve infinite cc?

Well, the standard ALP as IA is a candidate for universality! If it doesn't do it, then it's unlikely that any other IA will do it!

I think the above is wrong! Instead of say TSP: pick any (stochastic) $Q_i \to A_i$ function (finitely describable). Then pick any Q_i sequence. Every IA will have a certain expected error in assignment of pc's to the seq. of correct A_i's.

.10 Or maybe a certain expected product of pc's of " " .

.11 This (.10)/expected product, roughly, dominates" ("categorizes") the set of any other IA. Stochastic Generalized $Q_i \to A_i$

T. most Serious Competitor for the univ. IA would be the IA that knows and expects the (Think it can be zero in this case)

.13 formation. It will have a certain expected error

.14 So, the proof. Seems to work! Say K is the (bit) description cost of the possible A_i sequences.

.15 T. IA that can get this seq. exceeds his description cost of at least K.

T. Error of the universal IA will be K (of .15) (Via the "Reason" of .11-.13).

.20 (EN) The problem of Derbyshire (b)AZ: **Recursive Functions**: described in AZ!

Say $F(x) = f(x) \cdot x$; $f(1) = 1$ is a function of x.

$F(x) = f(x-1) \cdot (x-1)$: $F(1)$ is defined as $F(x) = x!$

In our definitions of functions F, the function name, is followed by a dummy variable, x.

(F, x) which is followed by an expression that defines the function. If this expression does not contain F, then that's the end of the definition. If the expression contains F, then we need at least one more eqn to define F, so we write $F(1) = 1$ (usually pc crisp to 1; end of function symbol).

.28 Or assignment of pc's to strings that describe functions or strings:

We start with a finite set of symbols alphabet and the possibility of adding symbols at certain points in sequences.

.30 We also have a set of rules that tell which symbols are legal at each point in seq. — at certain

times, new symbols of certain types can be introduced. This is expanding the alphabet.

Say we want the pc of a particular symbol in a code string. Then

Say d different symbol types in string has for

of the symbols that are legal at the present pt., they have occurred a total of r times in the string thus far.

The symbol we want the pc of, has occurred X times in the string thus far.

So $P_X = \frac{X}{d + r}$ This is a kind of version of "Laplace's Rule"

.39 {A possible (trimming/modification) (not exactly good!) }: r includes only those cases in which a symbol was in a situation corresponding to the ones now being considered. → See comments here → 355.00 spec

5/29/01

ID

Reports: .04 — .40

Is IA "Induction Alg"?

.00 ; 354.40 Re: Constraint of 354.39-.40 ! This constraint may be O.K. If we have a pc assoc. w. defining each new constraint type! This seems like a good idea! Needs Prof. Burr. ——

It may be a way to introduce arby regularity typ into the lang!

.04 AH! A nice way to start out! around 332.03 start out by defining a IA as a device that will deal w. QA sys like 332.03ff. Then

1. "report" as a tool! What I will talk about is general program for learning.

first some detail about the Problem I'm trying to solve! Then some solutions to that problem! Very accurate, very Procedural and impractical to implement.

.10 others Some are known algorithms commonly solutions, easy to implement but not very accurate.

Others will be between these two extremes.

I will then take kind of weighted sum of several of these techniques and use it as a general solution to the problem.

It is characteristic of some of these algorithms that may improve accuracy as you give them more and more data, so after we have trained the system with lots of data, we will solve the problem of improving itself.

start Section I Perhaps write a main talk then write an introductory portion.

.20 The problem we will be addressing is very general problem in inductive inference: we have a string or a number or a mixture of both, which we feed into a black box; we observe the output — another string or number or both.

Symbolically, Q_i is input A_i is output observed. We are given a large set of pairs Q_i, A_i pairs — Given a new Q_n, what is a good estimate of the probability distribution of A_n? — something in the direction that gives a probabilistic relation between Q_i and A_i. — Here we assume that the rule is some

This problem is very general! The Q's and A's can be pictures and answers in a formal or natural language. The Q's may be descriptions of problems or

.30 & A's may be solutions.

We might consider the Q's to be the stimulus of a biological organism — a cell an animal or a human — the A's are the response of that organism.

The Q's could be scientific experiments — the A's the results of these experiments. — ~~[illegible]~~

Viewed in this way, the general problem we are trying to solve, is the problem of all the sciences — prediction.

→ 356.00

ID

P → We will define any algorithm that attempts to solve the QA sequence problem a Induction Algorithm (IA)

.00! 355: 40 ~~There are many such methods~~ Many IAs are known ~~methods are known for dealing with this problem~~.

~~The most general of them all~~ It is known to provide both the best results possible, but takes entirely too much computation time.

We will consider time limited versions of this algorithm.

usually less general

We will also consider other, ~~simpler~~ algorithms that are known to work well in certain areas of inquiry.

P One ~~important~~ critical aspect of an IA is its "updating algorithm".

Suppose we have an "IA ~~operating~~ system" so that after the system has seen

$[Q_i, A_i]_{i=1...n}$, and is given Q_{n+1} a new input, its output is $P_n(A_{n+1} \mid Q_{n+1})$

a conditional probability distribution on A_{n+1} | given Q_{n+1}.

~~P_n characterizes~~ The ~~output~~ probabilistic function P_n, characterizes the state of the system after it has seen n QA pairs. Loosely speaking the Updating Algorithm for the system looks at P_n and at the true Q_{n+1}, A_{n+1}, and creates the new P_{n+1} distribution function. (3) footnote

~~the update~~

~~Each IA has its own update algorithm. That Algorithm embodies most of the information that is in a IA, and characterizes that IA.~~

More precisely, it may also look at many earlier P_i functions and earlier $Q_i A_i$'s, ~~that are stored~~

A Apart from initialization, the update algorithm is a complete description of an IA. Though update algorithms of different IA's can differ considerably, we will see that many of them have many similar features.

~~Our approach to the solution of the induction inference problem is to take a small~~

~~start~~ IAs as it is updated on

A very important characteristic of a useful IA, is that ~~it has become~~

~~a suitable long sequence of QA pairs (its "training set") "learns"~~ successive QA pairs, its accuracy

~~many QA pairs, its accuracy~~ becomes better and better — just as a student becomes more skilled with more training. We will use this characteristic of an IA to get some of

~~interest~~ them to usefully work on the problem of self improvement.

P Our approach to the problem of induction inference is to take a small set of carefully selected IA's and train them individually on sequences of QA pairs. These training sequences will differ among the IA's since ~~they~~ each IA ~~usually~~ is usually ~~adapted~~ applicable to only certain domains of inquiry.

8/24/01

ID

Page goal: .00 – .30

↕ introduction.

.00: 356.40! We will then combine these individually trained IA's to create a more general, "Macro IA" that can work on a greater range of problems than any one of its component IA's. This can be done by using a kind of linear weighting of the output probability distributions of all of the component IA's.

Among the problems that we will give to the Macro IA, is that of improving the update algorithms of each of its component IA's.

Next on agenda is its improvement over-all system of induction.

In the foregoing discussion it may be unclear as to how we define "improvement of an update algorithm" — or how we define "improvement of the over all system". Fortunately, though the theory of optimal IA's does not tell one how to obtain a optimal IA in any finite time, it does tell us which one IA is better than another — so "improvement" is a well defined problem.

Section 2 Summary's contents:

Section 3 will describe a Macro IA. The really optimum IA. / Such an algorithm is necessarily incomputable in finite time; we will also describe some computable approximations to it.

.20 We will also discuss the use of Genetic Algorithms and Neural nets as IA's.

Section 5 ~~Discuss the combination of several IA's to~~ We tackle how to combine several IA's to yield a "Macro IA" that is more versitile than any of its components. ~~In the combination the parameters of the combiner~~ The combining Algorithm itself The improvement of the combining algorithm is one of the first problem that we

(SW) Maybe have a FAQ at end of report.?

Section 4 Discusses training IA's; how a sequence of well designed QA pairs can increase the problem solving skill of an IA.

.30 (SW) (8/24/01) On update Algms for GA, ANN: They necessarily involve Irng — by some external Irng device. Tho, since it's an AIE, it can be regarded as a "problem" to be solved by the GA or ANN itself. If so, the initialization of the GA/ANN must be set fairly by hand! Not exactly! The early TSP's for Theory, the initial state is set up by USR for each problem. The system then trains on this set of (problem desc/initialization params.) pairs. This will "mimic" can be done by a different system — it can be done by Trainer or AZ; or ANN can work on GA or vice versa.

ID

→ A paper on how to find good [redacted] settings for Evolv. Algms. [E3 Evol Comp. Vol 9 #2 April 2001 p 129]

1 PM
30 Min

.00: 357.40 In GAs the main initial params are pop. size, Mut/cross algms., initial (population). T. simulat[ion] for Mut/cross can be obtained from Thomas's given values; those had some "forms of functions". — Unclear as to just what was done: T. "initial population" can be improved by studying the [fitness] functions for various problems (= fitness functions). Since it's an open problem & ordinary GA can work on it,

.04 I guess (ANN) Nat'l world be similar.

Rose Publ 1230 Friday 1 PM R. of TM 7 87 (Hearing test). 893 9170 3566 Brit Computer Museum Problem Marketing org. Dave Copley Sq.

SN A way to do ANN: First just get pgms to go from Q_i to A_i individually so for each Q_i A_i one has a best soln. (This works only if we use a large random nos. for variables or each $Q_i A_i$ retrieved of 10 or 20 problems w. t. same "soln".)

.10 So we have these triplets (Q_i, I_i, Pgm_i). We then try to find simpler pgm from Q_i to Pgm_i (for all i)

Presumably, this is easier than finding a simpler pgm for $Q_i \to A_i$ for all i.

.13 † problem of .00–.04 is interesting, perhaps very tmpt! Solns. mean that GA w/o ANN could be realized (or be close to) a Final TM. ANN is never Church-able, since it doesn't seem to deal w. digital/data properly

"I" = "initialization"

So: Consider a sequence of Q_i, I_i pairs. (I_i = mut/cross algms, initial pop, [redacted] of includes pop sizes, say — is consider pop size

This can be worked on at 2 levels: (a) Given Q_i to get a d.f. for I_i.

(b) Given Q_i to get as "Goodness possl." an A_i in r. available CC.

.20 In (a) we have a problem of finding a stochastic function of hyper dist. given; hyper rect. corners of $\{I_i\}$ — This is a natural problem [redacted] Real IA's normally work. → Are "Separable" (w.r.t. indiv)

.23 In (b) If we assume that r. 2 components of I_i are "separable" [redacted] (mut/cross; init. pop.) are indep. (probably not true but usually not very false). — We can optimize them separately.

A reasonable Gene for [redacted] init pop is hyper r. mean "fitness" (as deduced by Q_i)

Gene for mut/cross is unclear.

.28 Actually, during Normal GA such, we could be [redacted] trying new mut/cross methods & keeping Scores on them, keeping goodones. E.g. One way is to "Mut/cross" r. "Mut/cross"'s.

If .23 is false then r. Mut/cross will (perhaps) slowly vary during r. such as r. mean fitness of r. population ∫.

.30 Anyway, if .28 is used, then, for data, we can have not r. initial mut/cross as std, but r. final mut/cross's used can be paired w.t. Q_i's. — So we want TM to be able to go quickly from Q_i to [redacted] (close to) r. final set of Mut/cross 'cross/muts' used.

So in this model, we try to find Q_i → init pop, that gives max area (fitness/curve?) Also TM has an Algm that goes quickly from Q_i to r. final set of cross/muts used for Q_i's soln.

N.B. In for r. problem Q_i should include "context" info that Human would naturally have available to help him do initializations.

8/30/01
ID

3.59

Einstein
K
C Main Karl
Jeff Eugenia.

.00: 358.40! 358.13 – .40 is probably enough for this report! Going deeper would be of more academic interest.

Re 358.04 — To what extent can we do this w. ANN? — well, look at the seq. of initializations.

Assign 358.13 If we have this seq. of Q_i's, Is pers:

I_i = Number of connections to I/O of neurons/wts., initial wts, squashing function.

All of these can be varied during the search a/o changed on successive epochs.

We end up w/ a seq. of (Q_i + "state" + trial on "Test set" Corr.)'s

This corresponds to variation of chess/wts during GA (358.28 – .30)

We end up w/ a sequence of Q_i's corresponding to each Q_i will be a final set of connections + wts. Pair were "Best." — "Best" being wrt Corr. on {Test Data Set}.

We may want to retain use ≥ 1 such "Best" soln.

.10 If we do ANN's w. as few wts as poss., we will use a Max Prob. Estimate criterion (usually) = 1/(Hessian)

mult by the empirical pc of the soln. (Comp. ≈ Max likelihood Estimate).

.13 The goal of the Q_i → initialization function is to ascertain an initialization of hy Corr, or one that can be converted to a set of wts by Corr "very quickly".

Perhaps the "very Quickly" means that v.s. Non-linear Optim (Marquardt – (?)) method.

perhaps a few jumps from it each.

A possible goal like .13 could be a purely discrete goal: Pick of choosing a certain no. of Neurons,

i connections. If we are doing only a few wts than v.s.n.l. optim. may quickly lead to a Peak. (The Pick point could be much "influenced" by initial amounts "Approx" wts.)

8/31/01 In Both GA & ANN: The "problem form" Q_i, can have various degrees of Understanding

.20 "Preprocessing" by USR: One most extreme is Main preprocessing is Vocab. input, or Text in English. Input English meant for Humans.

-(Contextual info has to be added in by USR if not already present.) "pre processing" would

"abstract" the problem for TM, pick out relevant variables; perhaps help w. functional form of

soln, ect. —— Normally, we are somewhere better. Have a extremes. (Also a "Hint" channel could be added!).

In describing the GA IA; first tell how GA's normally work; Then tell how they need to be modified to become a IA. — Plunge into a little detail on how this modification can be done.

Do Similarly for ANN.

.30 9/1/01 For AZ+ just do AZ (3 input). (Maybe Multiply the 2 input DAZ "impedance")

Perhaps, in describing how we compute epoch of the function: 354.28–.40 (& 354.20 for recursive funct)

The definition of a complicated function proceeds by its effected by defining a sequence of definition

of functions, each of which is allowed to use only those functions

defined before it as well as a certain set of "primitive functions"

+, −, ×, ÷ (F1), ½, X, Y, X, Y+, X, Y, −, X F2, F1, X, X, X, mult, F1,

dummy vars, values, functions

F1(X,Y) = (X+Y)·(X−Y) = $X^2 - Y^2$ F2(X) = F(X, X) = $X^2 - X^4$.

For small integers < 100 say, pc of $\frac{1}{n^2}$ is not bad. $\sum_{n=1}^{\infty} \frac{1}{n^2} > 1.5188$ $\frac{\pi}{3}$?

9/1/01

ID

360

P344

:00; : Comments on "Grammatical Evolution": (O'Neill & Ryan / Ryan & O'Neill ei.) — IE's xtn on Ev. Comp August 2001

They start w. a BNF style Grammar. They use uniform def. over choices to generate objects.

Dependent Grammar, ∴ objects can be functions or Multiline progs or — whatever.

Since the chromosome they use is short (say 2x bytes) they only get 2x decisions per chromosome — so if the object needs more decisions they use the 2x bytes in recycle!

They sometimes certainly decisions in the Grammar, but in a not-apparently-unreasonable way (& — Cd. correctness depends on residues to various/moduli diff. &, but 2→2+1 mutation always moves all decisions 1 choice unit in some direction)

So they can end up w. objects very large objects (many decisions) but are always limited in content — limited by no. of bytes in chromosome.

To do Lsrch, start w. chromosomes of length 1, then 2 then 3, etc.

From a theoretic things they do, I get the impression that the Authors are not very skilled on Math!

Anyway, mutation on individual bytes can make small or very large changes in phenotype —

A small change would be + → − or 1 → X ; a larger change! 1 → +

which changes no. of parts; a change all decisions in rest of phenotype.

The effect of most crossovers would be to completely lose any memory of the searching exchanged!

e.g. A B , C D → A D or → A C ! So long the not protected (or needed)

My impression of O'Neill & Ryan: That they are much excited by the idea of Generating legal expressions by Grammars — by the broad scope of Domains in which it will work. They pretty much would like the actual search operation to be separated from the Constraint creation part — but they are excited by some "automatic" side effects of their coding method (e.g. reuse of chromosome Bytes) for coding needing many decisions.

O & R suspect that Better results from those in "2 outputs" problems. Why this is true is not clear. The symbolic regression & trigonometric are essentially the same problem, yet they get wildly different results for those on these 2 problems!

ID

Report: .24 - 1.36

CPA could possibly Align. device.
CPD system calculator evaluator Learner Inducer

.00 : : General Remarks:

1) One Big thing I expect to get from TSQ's : Ideas on How O_{n+1} is a function of $(O_n \& Q_{n+1})$
i.e. When a new problem is [redacted], How do we try to fit it w.r.t. past [redacted] prob. solving Methods?
What do we normally try first on a new problem? — Well, we try "the standard alg." — by observing just How & why it fails, we get ideas on how [redacted] O_n has to be modified to deal w.r.t. new problem
This should give info like "Which kinds of cross/mut's are needed for a GA soln.?"

.06 2) Hyr, perhaps the only BASIC IDEA Needed is, if O_n codes $QA]_1^n$ then, to code $QA]_1^{n+1}$ we want to try to code just $(QA)_{n+1}$ using as cheap as possible parts from O_n along w.r.t. recognition algs for Q_{n+1} — also referred to as "parts" of O_n.

→ Hyr. .06 seems to miss an Important common heuristic: we ask "How does the new problem differ from old, & solve the prob? We then try Methods that have worked in analogous situations. .10

.12 3) An important feature of the present system: Because of the "Bootstrap" features, we can start out w.r.t. not w.r.t. IA. It is important, hyr, that we have the proper core for O_{n+1} —

.14 Matrix dim = pc • pc of corpus — will it be max. — Also there are likely a potentially Universal (system) set of algs to try for O_{n+1}.

If our initial IA is slow, we can push it along w. hints, by raising massive computing power, or by various ad hoc additions of skills. Also, a good TSQ can really speed things up!

It may be that we have to have some minimally competent method to search for pockets of IA. w. occasional limited backtrack — which is a common theme for humans — so is, I'm hoping it mainly just modify O_n to get O_{n+1} (apparently), but it may be very limiting! (conceivably, we may "backtrack" more or "updated" a correct .20 branches of Q's (which amounts to same thing).

4) Today, I want to make deadlines for writing (≈ final) drafts of the 3 IA's: Should only take > 3 hrs.

~~begun 10:50~~ 9/2/01 start 10:51 A

.24 347#6 We will describe three of the IA's in some detail: They are ① A method based on SAZ, which is similar to LISP, ② A method based on Genetic Algorithms, ③ A method based on Neural Nets, Insert 347.24-.26
N.B.!

.26 All three of them have a way of generating O_n and essentially a probability cost

.29 Section: The SAZ system. ~~This version of SAZ will be based on a LISP like~~ Though SAZ is similar to LISP, ~~modified versions of~~ language. Modified versions of the system can be made using other languages such as Forth, APL, assembler languages, etc.

.30 We will first describe the language SAZ, & show how it generates & evaluates Functions such as O_n (w.r.t. 332.19), and how we assign a probability $P(O_n)$, to its description.
We will show how O_n is used to evaluate $P_{O_n}(A_i|Q_i)$ in 333.01
We will discuss the initialization of O_n for a small corpus, and then explain
How O_n is incrementally updated as new (Q, A)'s are presented to it. — 362.0#

.36 Maybe omit this word.

9/1/01
ID

Report! .07–.40

$\frac{6}{(\pi n)^2}$ = epsi of n

stochastic Asymmetrical ZIF1.

.00: 361.40 : [redacted] Perhaps write a BNF Grammar for Function Generation: It may be impossible! —
ie. it may not be at CRUX — but it is uniquely parsable. A complicated function can be parsed into
sub functions in > 1 way, hmr.

I may (eventually) be able to find a way to clarify SAZ as a recursion BNF, but this will take a bit of time. Drop for now.

→ see 364.37–.39 for a "Grammar" to Generate Functions

Do give several revealing examples.

.07 : 361.36 The language SAZ begins with a set of primitive unary, binary and possibly ternary functions.
Functions are defined as Sequences of primitives and previously defined functions. An economical
way to describe a large complex function, is to first describe various
.10 simpler functions, sometimes first or important sub-function components in Reverse Polish
or Prefix form. In general, it is not economic to define a sub-function unless it is used
at least 2 times. In some cases, a function must be written & stored even more
.13 times, before it becomes "economic" to do it.

.15 Suppose our primitives are , sum, add, sub, mult, div, sin, sin^{-1},
.16 and we want to define ~~F1 = (x² − y²) sin x~~ F1(x,y) = (x²+y²) sin(x), Then, F2(x,y,z) = F1(x,y) · F1(x,z) · [illegible]
.17 Using Reverse Polish notation, we would write F1, 2, X, X, mult, Y, Y mult, add, X sin mul
.18 followed by F2, 3, X, Y, F1, X, Z, F1 mult
.19 When we write f1, 2, the "2" indicates the number of arguments F1 has.
.20 Similarly in "F2, 3", "3" is the number of arguments. F2 has.

To evaluate the probability of the descriptions of .17 and .18 ;

First we write a sequence of all legal symbols: [redacted]

.23 F, add, sub, mul, div, sin, [redacted] sin^{-1}

[redacted] we will call .23 the "pre corpus". Symbols following it will be the "corpus".
The first symbols in the corpus are (from .17) F1, 2. Since we are defining a
function, the first symbol must be F. Its probability is 1. Since this must be the name of
the first function, the probability of 1 is also 1. The next symbol, 2, denotes the number
of arguments. The following would be a non-negative integer. [Footnote]: A zero argument function can
such as √2 or π, which could be defined by suitable arguments
.30 e.g. [illegible] we will associate with
the "2" the probability $6/((2+1)\pi)^2$. The integer n gets probability
.35 $6/((n+1)\pi)^2$: we note that $\sum_{n=0}^{\infty} (1+n)^{-2} = \frac{6}{\pi^2}$, which gives our normalized Run constant.

N.B. recently this is using arbitrary [illegible] To be done by USR! Incidentally for possys.

363.00

NB, In .16 – .23 change X → X1, Y → X2, Z → X3, etc.

In SAZ (in doing generation of the code) it is a "FSM" ! ("State" = Name) = No. of args
the Function has. — Also, if the word should be finished (so a last Pushdown store is needed) the
.40 State changes.

363.00

ID

(and Laplace's rule)

parallel Lsearch

.00: §62.36 We can generate functs in a Montecarlo way, then use the Monte Carlo, [illegible] mode.
At each pt. in the generation of the funct. defn, the machine keeps track of (updates) how many
symbols of each type have occurred thus far. The (various grammar) tells it which symbols are legal
.03 at each pt. — from this [illegible] the probability of each symbol can be computed.
During the generation of functions, the system has to know "where it's" in order to know what
symbols legal. It has to know it all times: It is inside a unary function, when it completes that function
it goes to the stack to tell what N-ary function it is in; i.e. which of N variables one has just finished defining.

← At this pt. D switched off into a more exact gramm. §64.37-.39 recursions & specializations

At the end of a funct. definition, the system can either halt or start defining a new function.

Say one is generating a 3 any funct: On the stack are F7, 3, 2 if one is working on the 2nd argument.

When one gets to the 3rd arg is finished, one goes to stack, looks at top 2 entries, changes them to
.10 3, 3 & starts work on 3rd argument. — when this is finished, F7 has been defined.

.12 We can End or [illegible] change F7 to F8 & declare define F8. [or, when one finish a function that is not → (.16)
The probability of End is unclear. We could make "end" a primitive symbol, so each time there is a possibility
of ending a funct, it either does or doesn't — so at diverse pts, the pc of end is 1/(no of times plus one
that end opportunity has occurred)
so first time, pc = $\frac{1}{2}$; next time $\frac{1}{3}$ next time $\frac{1}{4}$ next time $\frac{1}{5}$ . . . .

.16 (12) being newly defined funct, we also have possibly some stacks of having finished using a previously defined
function.

The trace structure of a function: Each node is characterized by an integer "branch no" and, if
.20 there "branch nos." are kept on stack. The trace is traversed in the "usual" exhaustive way, by starting at the left most uncompleted branch.
branch we are now in
So each stack says like 4, 3 (we are on branch 3 of the 4 branch node)

So: stack has any n, m; n is arity of present function, m (≤ n) is no. of argument being worked on:
m = n+1 — put n, m+1 on stack
.22 if m < n, then/ sprout a new [illegible] expression. (if m = n we pop stack & look at n, n
.23 expression → funct [illegible] (we can consider consts: funct of arity 0.)
.24 to evaluate an expression: we choose a function & its pc is determined in usual way. IP-F has arity zero, (recursive)
we pop stack because m = n of . Otherwise; if arity is n ≥ 0, we put 1, n; 2, n; ... n, n on stack
n words on stack.
We then look at stack; if m < n, we do m → m+1 and make expression — which means 122

So each stack word is of form n, m (n ≥ m; n is arity of expression, m is arg. no. of arg. being evaluated)

Brouchard set to define O_2: we define F1: we choose its arity, n. We
.30 sub n, 1 at TOS, next n, 2, next n, 3 ... to n, n; so we put n, n on stack from n, n-1 .... to n, 1.
We will occasionally put 0, 0 on stack for a zero arity function & constant — so this is inconsistent.
may be not completely — look at it more carefully.

Say n = m = 1: arity = 1.

→ (SN) I may want to do Regular Polynomials, functions none free.
(SN') Indeed arity n, x1, ... xn are possible expressions, if arity ≥ 0, no x's are possible (i.e. no variables).
The problem seems to be any ϕ (constant) w.r.t. arity 1 n=1, m=1. sum 1, 1. | sum, 1, sum, 1, 1
stack
So 0, 0; 2, 1, 2, 2

→ Perhaps it's different defining a unary funct, as opposed to evaluating a unary funct!

.37 F 5, 3 means [illegible] F5, a random funct, but we can have x1, x2, x3 as expressions, & this is only difference.
.38 F 5, 0 " " " " " " " " " can't " any k as expressions.
in
so defining a function; if arity only extends as .37-.38 — which is not much difference.

ED

$+ - \times \div$, 1, ϕ, sin, $\sin^{-1}$. 8 functions, 0 1 2 arity; 2 2 4

.00: 363.40 : In elementary $F5,\phi$; we just tabulate f_5; we know from the properties of f_5 that it is unary, ϕ.

The definition of $F5,\phi$ tells us that it has no x_i's in it.

Here when we write F5,2 e.g. and develope a defn, we use at most 2 variables in the defn. but we may use only 1 or none! If only 1 variable occurs, say x_2, then $F6(x_1; x_2)$ will be a power of x_2 only in this way: x_2 appears in the defn.

In the defn of F6,n; every time x occurs, it is followed by an integer from 1 to n. = probability

The probability of "x" occurring (n times) is $\frac{1}{n}$ or $\frac{n-1}{n}$ if occurred in past.

So to define F7,3; The 3 is not very important (see 363.37,38)

We start w. some function F say $F5$; ① which is arity 2. We put just 3,1 on stack. 3,1 / 2,1

We write F5,3, we chose a function; say F2,1; we write F2,1, we write 2,1 on stack.

.10 We chose F8,0 (a constant) write f8,0, then pop stack, put F2,2 on stack.

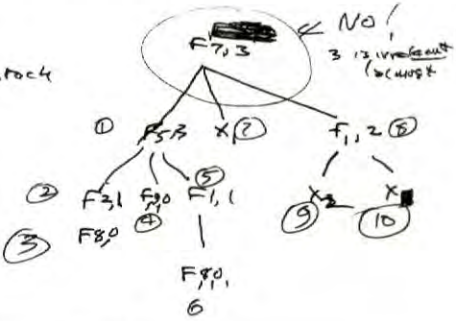
" " F9,0 " " " F9,0 " " " " of 2,2 then look at stack.

it says 2,1, we do 1→2 so 2,2 on stack

we chose f 1, put 1,1 on stack.

chose f 8,0 (constant), write f(8,0) pop stack → 2,2 ∴ pop stack

F7, 3 ; F5,3:	3,1
F7,3 : F5,3 : F2,1	3,1 : 2,1
F7,3 : F5,3 : F2,1 : F8,0	3,1 : 2,2
F9,0	3,1



F7,3	:	
F5,3	:	3,1
F2,1	:	1,1
X2	:	(pop) 3,1 ; → 3,1+1 = 3,2
X1	:	3,2 → 3,3+1 = 3,3
F3,2	:	3,3 | 2,1
X2	:	3,3 | 2,2
F8,0	:	pop 3,3 | pop end.

.20

only X_i and $f_{n,\phi}$ can cause pops.

If a pop occurs

n, m on stack and

if $n > m$ then $m \to m+1$ and new function

if $n = m$, pop stack again

(if X_i or $f_{i,0}$

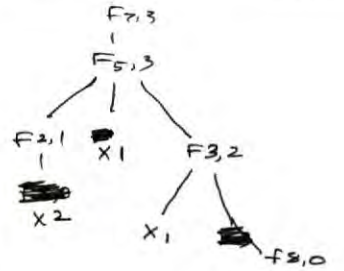
keep pop until $n > m$

then $m \to m+1$ and

start new function — of arity n'

If $n' = 0$ or $f_{n,\phi}$ or X_i, pop stack,

if $n' > 1$ write $n', 1$ on stack.



F7,3	—
F5,3	3,1
F2,1	1,1
X2	pop to 3,1 then 3,2
X1	3,2 → 3,3
F3,2	3,3 | 2,1
X2	3,3 | 2,2
F8,0	pop all, end.

.30

If X_i or $f_{i,\phi}$, pop until $n > m$

If $n > m$, stack no need to pop.

β then $m \to m+1$, write new function $f_{\phi,n'}$

push if $n' = 0$

If $n' > 0$, push $n', 1$ on stack go to β

α write new function, $F_{\phi,n}$ or X_i

If written function is X_i or $f_{i,\phi}$ then pop until $n > m$ | else function is $f_{i,n}$ push $n', 1$.

$n > 0$

push $n^*, 1$

Go to α

start write new function; $F_{i,n}$ or X_i.

If new function is $F_{i,0}$ or X_i then pop until $n > m$; $m \to m+1$ else push $n, 1$

go to "start"

If stack is exhausted, END

Repart: .02 - .40 arity

.00: 364.40: T. pgm of 364.37-39 is probly fine! I should clarify (Declarations/notation) for myself. It may be that I'll give less detail in report.

.02 A prior probability is assigned to functions such as On, in the following manner: (Good!!!)

We start out with a list of primitive functions and primitive constants. The functions might be sum, minus, mult, div, sin, sin^-1, The constants could be 0, 1, π, e ...

We will use Polish notation to define functions: sum, 1, 3 = 4.

So the evaluation of sum, 1, 2 is 3.

.08 To define a simple function we will write its name followed by its "arity", the number of arguments the function has.

[FN is using "arity" to be the number of arguments it has.]

.10 followed by a sequence of functions and constants defining the function:

F1, sum, mul, X1, X2, Mul, X1, X2 Defines F1(X1, X2) = X1·X1 + X2·X2. (subscripts)

To define more complex functions compactly, we will define a sequence of functions and constants, culminating in the complex function we want to define.

Each function or constant can refer only to primitive functions and constants that are either primitive, or have occured previously in the definition's sequence of definitions.

At each point in a definition string, only certain symbols are legal. Except for special cases that we will discuss later, The probability associated with a particular symbol, appearing at a particular point is (n+1)/(m + Σ_{i=1}^{m} n_i) ... → (n+1)/(m + Σ_{i=1}^{m} n_i)

Here m is the number of different symbol types that have occurred at this point in the definition sequence.

n is the number times x has occurred previously in the definition sequence.

n_i is the " " the i-th legal symbol type for this point has occurred previously in the definition sequence.

This probability assignment method is very close to Laplace's rule.

.20 .21 There are two kinds of symbols whose probabilities are assigned in a different way.

.23 .24 .26 OMMIT If r is the arity of a function being defined, it is assigned a probability of c(n). c(n) is an assignment of probabilities to the non-negative integers. The form of this function should depend on how it is being used. In the present case, c(n) = 6/π² · 1/(n+1)^2 : 6/π² is a normalization constant.

.27 .29 .30 .32 replaced by 367.30-39 In functional definitions, a positive integer must follow each X. This integer will be between 1 and the arity of the function being defined. We will give each of these integers a probability of n^-1.

In definitions of functions, X i will represent the i-th argument of the function.

If the function is of arity, n, we will assign each possible integer, i, [2] probability of n^-1.

.34 By an example: Suppose our primitive functions are sum, mult, and the primitive constants are 0, 1. Consider the probability associated with the symbols in the function definition F1, F2, Mult X1, X2, F2, F1, F1, sum X1, sum X2, X3, X1, Δ

Here Δ is a symbol denoting "end of definition sequence" it only occurs once.

Since F1 is always the name of the above function definition, its probability is 1.

.35 2 has probability c(2) = 6 · (2π)^-2

consider mult: After 2 points, the only legal symbols are sum, mult, 1, 0, X, X not possible. Since none have occurred before, the probability of "mult" is (0+1)/(0+5) = 1/5

The next symbol is X. The legal symbols at this point are sum, mult, 1, 0, X, only mult has occurred before

so X gets probability 1/(1+5) = 1/6

ID **REPORT** .00–.12

.00: 365.40: The 1 following x has probability 1
The next x has probability $(1+1)/(1+5) = 1/3$, since the legal symbols at that point are sum, mult, Φ, 1, x
The following z has probability $\frac{1}{2}$
Since the definition of F1 is completed, the next symbol can be only be the "end" symbol, Δ, or F_2, the beginning of the definition of the new function.
So F_2 gets probability $(2-1)/2 = 1/2$.
omm it! [3] has probability $c(3) = 6\cdot(3\pi)^{-2}$
At the F1 position, the legal possibilities are F1, sum, mul [redacted], so the probability
.08 of F1 is $(1+1)/(1+1+3) = 2/5$
At the next position the legal symbols are sum, mult, Φ, 1, x, F1
.10 The probability of "sum" is $1/(4+6) = 1/10$
At the next "x" position sum, mult, Φ, 1, x, F1 are all legal
.12 The probability of x is $(1+2)/(2+1+1+2+6) = 1/4$.
Fact: the probability of the Δ at the end is ~~1/3~~ → Looks like pc of Δ is always $\frac{1}{2}$! Fn & Δ never occurred before! This makes long strings of functions very unlikely!

.13 Discussion; Note (.08–.12)R: Perhaps have adjustable param, "a" so probability of Δ is a. Expected no. of functions in a string is $\frac{1}{a}$.
If we had a pc of "F" at that position: pc of Δ would be
$\frac{1}{2}$; $\frac{1}{3}$, $\frac{1}{4}$ ..... $\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4}\cdot\frac{4}{5}\cdots\frac{n-1}{n}\left(\frac{1}{n}\right) = \frac{1}{n^2}$ (= pc of Δ) = pc of n function sequence. → So it does converge: If we [illegible] much, it will not converge?
If we allow "the no. of previous F's" to include all uses of F, the probably will be $\gg \frac{1}{n}$.
Not clear how we can prevent this! We can do it Ad hoc: Have special way of computing pc of a new defn. (Modifyable by the system of course) — use a low a param. d.f. → .36
.18

.19 Another point: when we ask for arity n functions, we may get arity < n — perhaps the "arity" could be specified after the funct. is defined. The first "x" is x1, the next has perhaps uniform d.f. over x1 & x2; the. After xi has been defined, the next x has
.20 uniform d.f. over x1 to x(i+1).
Hvr, the "asking" of .19 only occurs when we are generating functions Monte Carlo-wise —
Otherwise, its post-hoc assignment of pc's to functions of known arity.
Actually, .19 would probly be fine! We may not need to specify arity when we first define a function. It is something that is discovered after the funct is defined. The method I'd been using up to .19 was not so good in this area! If arity n was specified, a monte carlo generation of a function could yield any arity from Φ to n (!).
In the method of .20 ff, the arity is found after the function is generated. After generation, the [illegible] name, arity, & case count are kept in the function's "property list"
So, in .20ff we want to keep track of the "highest x following integer" in the definition of a function! This will,
.30 when the defn is completed, give its "Arity".

.32 This gives us at least [3] params to be adjusted in the system:
(1) if n x types have been named this far what do we want the pc of a x(n+1) to be? → (See 366.19–.30)
(2) If m new functions have been defined this far, what pc do we want for a new funct defn. → (366.08–.10)R Then .13–.18, .36–.40 →
.35 (as opposed to the "end symbol", Δ.) (3) the function c(r), of 365.24

.36 (.18) → In discn. of .13–.18; Instead of the sequence $\left(\frac{1}{2}\cdot\frac{2}{3},\frac{3}{4}\cdots\frac{n-1}{n}\cdot\frac{1}{n}\right) = \frac{1}{n^2}$, we [illegible]
a slower converging seq $\prod_{i=1}^{n}(1-f(i))\cdot f(n)$: $\prod(1-f(i)) \approx e^{-\Sigma f(i)}$
if $f(i) = i^{-1+\epsilon}$, $\Sigma f(i) \approx e^{i\epsilon}$ so $e^{i\epsilon}\cdot\frac{1}{i^{1-\epsilon}}$ will diverge. So this may be one of the more slowly converging sequences of f(i) — (Tho it should be possible to make f(i) ↓ slightly faster than $\frac{1}{i}$. So we could have $\prod_{i=1}^{n}(1-f(i))$ be any strictly ↓ funct of n. (say slower ↓ than $\frac{1}{n}$).) → in fact, one can define f(i) → 367.00 spec

The z is the probability of a sequence of n definitions, then stop.

.00 So $\left(\prod_{i=1}^{n}(1-f(i))\right)\cdot f(n) \equiv g(n)$; where $g(z)$ is any (?may be unary conditional) function of z.

The Main point is: We can choose $g(n)$ & hence $f(n)$ so that the sequence converges as rapidly or as slowly as we like.

& $\frac{g(n-1)}{f(n)}$ $\prod_{i=1}^{n}(1+f(i)) = \frac{g(n)}{f(n)}$; $\prod_{i=1}^{n-1}(1-f(i)) = \frac{g(n-1)}{f(n-1)} = \frac{g(n)}{f(n)(1-f(n))}$

So $\frac{f(n)(1-f(n))}{f(n-1)} = \frac{g(n)}{g(n-1)}$ ∴ $f(n)(1-f(n)) = f(n-1)\frac{g(n)}{g(n-1)}$

So, given $g(n)$ as a function of n we can solve for $f(n)$ as a function of n.

$x - x^2 = \alpha$; $x^2 + \alpha - x = 0$ $x = \frac{-\alpha \pm \sqrt{\alpha^2+4}}{2}$; $x = r$; if we want r to be > 0

$r = \frac{-\alpha + \sqrt{\alpha^2+4}}{2}$

$= -\frac{\alpha}{2} + \sqrt{(\frac{\alpha}{2})^2 + 1}$

SN One of the IAs can be the TRAINER! But we won't make it to be used much — Maybe it will have expensive ? as part of its instructions. (Actually, "Trainer" will have to be slow)

.10 Corrections needed in 365.03 ff! (in view of 366.32.-.35)

365.08 [return UT { followed by any # no. of ...]

365.10 (remove 2 (cadr))

.21 two kinds of (cadr zeds!

.23-.26 on CEP) = $\frac{6}{\pi^2}(n+1)^{-2}$ ← ommit

.27-.32 Also ommit, but replace with M! 367.30 ÷ .35

.34 ommit Z and Z.

.36 omit this line.

I want to rewrite all of 365.03 ff using subscripts on X's and F's. (Even Paris is easy to put in, in LaTeX: \underline underline.

Also in rewriting perhaps write data string as vertical column: ll columns; set off key symbols, probably. like $(2+1)/(2+3+1+1+6)$

Maybe try for better function to decide.

.20 Probability of a F_i occurring a particular # times is decreased, is or no. of uses including definition.

There should be no "1" added in numerator or denominator! ← perhaps write correctly later

We could fix this by using a "pure corpus" i.e. "straight rule".

In (368.15 R) weight 8.8×10^{-8} a pc. of function.

Now, in any use of this function, we can use given x & some definition but arguments permuted!

So in 368.15 R we have to PC of F_2 — a function of 3 args. So 6 permutations.

F_1 has only 2 permutations, mul, sum have a weight $8.8\times12 = 105.6$ so → $105.6\times10^{-8} = 1.056\times10^{-6}$.

So whether F_2 use okay mul once, sum use twice.

Actually $F_1 \equiv$ mul(1) not so interesting!

Try $F_1(x,y) = x+x$; $F_2 = F_1(x,y) = y^2+2y$

; F_2 mul $F_1, x_1, x_2, x_3, \Delta$: perhaps use sum, mul, ϕ, 1 => "pure corpus"

F_1 sum mul x_1, x_2, x_3 sum + mul commutes. x9 for Param. x 2 for F_1; x 6 for F_3 8 x 2 x 6 = 8x12 = 96 → 368.18

.29 ~~Mul x_1, x_2 = Mul x_2, x_1~~ sum Mul x_1, x_2, x_3 = sum x_1 Mul x_2, x_3

.30 (to replace 365.27-.32) Whenever the symbol "X" occurs in a function definition, it will have a subscript that is a positive integer. The first time an "X" occurs in a function definition, this subscript must be 1. Thereafter, the subscript on an X in the same function definition can be any integer from 1 to n+1; where n is the largest X subscript that has occurred thus far in that function definition.

.34 Each of these integers has probability $(1+n)^{-1}$.

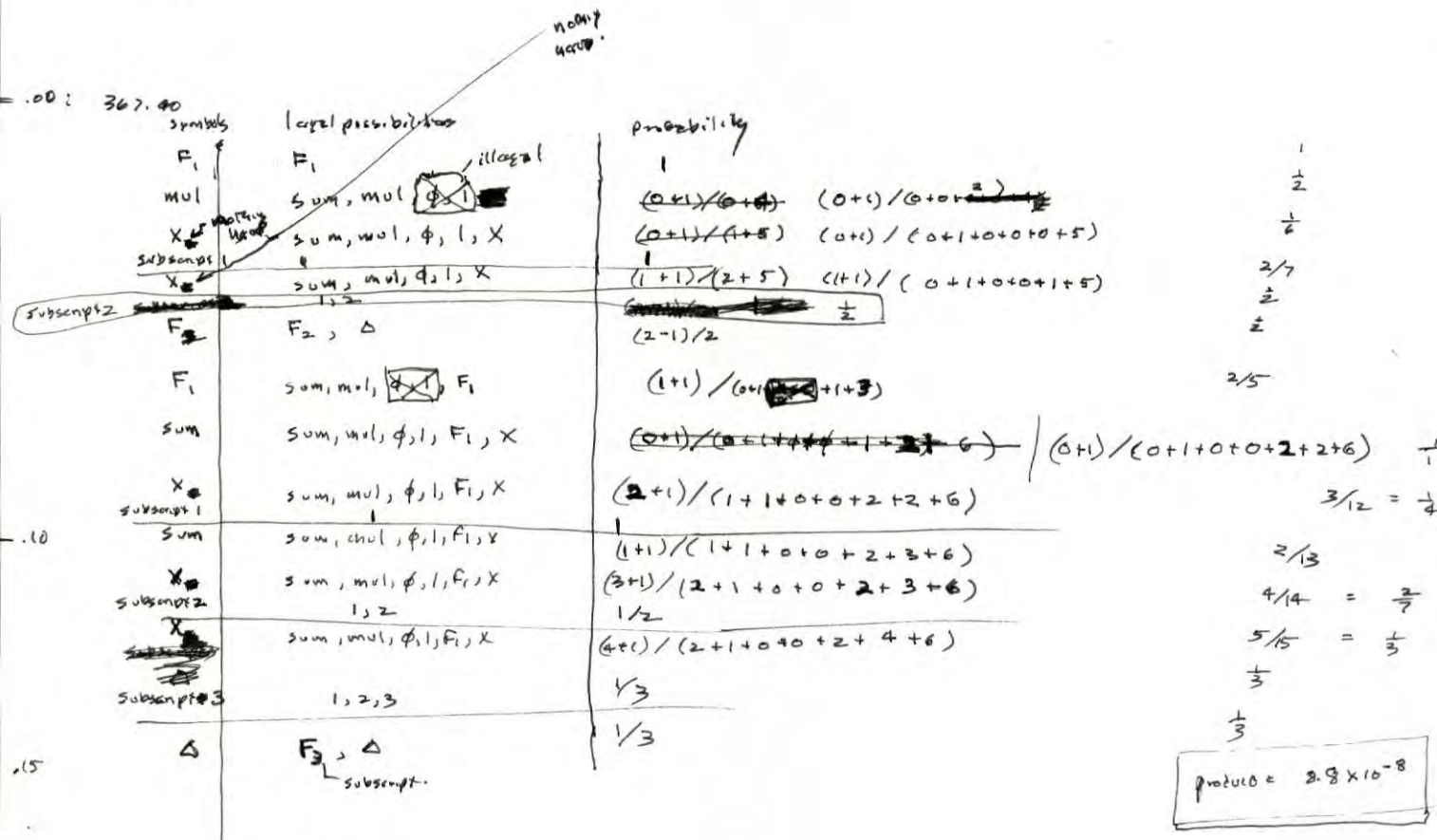
When the function name F_1 occurs, it is the only possible symbol at that point. It is given probability 1.

For function names F_i with $i \geq 1$, the probability of F_i is $(i-1)^{-1}$. The only other legal symbol at these points is the stop symbol, Δ, which has probability $1/i$. The stop symbol

.38 indicates that the sequence of function definitions is completed. It always occurs at the end of the

.39 sequence of definitions

$(i-1)/i$



.18 : 367.24 **Multiplicity**

redundancy
Degeneration :

sum Mul k1 k2 X3 = sum Mul X2 X1, X3 = sum X3, Mul X1 X2 = sum Y3 Mul X2 X1

X2 X1 X2 X1 X2 X2 X2 Y1 X2 Y1 X2

Also, if we have an arity N function, we have N! functions like it. That could be used where ever it is used, to give a difficult coding. However, various symmetries (like commutativity) can reduce (or multiply) this redundancy. Associativity might also give equivalent expressions. — Also function could ↑ pc of a function by a multiplicative factor.

→ Also, any subfunction used in a def^n has its pc ↑ by its (arity!). Whether this occurs each time this funct. is used in a def^n is not clear. I Really Have to Look into all of this!

Have a stack for STACK for ARITY of functions we are working on — Tho it may be unnecessary. Their int^ed info may be already on the stack in some form or other. "n, m" pairs (n ≡ arity).

Summary of work on expositing how to compute P.C. of a function:

361.24 Derbs IA's in general. (Also note 347.00)

362.07) Derbs & Cayuga SAZ (exposition): how PC's are assigned to functions.

383.12)

363.16 – 364.39 : Def^n valuespaces pgm-to Generating functions (& GFs & BNF)

365.02 – 366.12) Expositions: How PC are assigned to functions. This has to be rewritten — w. on the function of 368-01 – .15 used as example. — Also note errors in this whole thing! — 367.20 (Fix w. procedures)

5/6/01

ID

Listing of some impt pp.

INDEX

~3685

320.28 is a pure listing of remarks of QATM.

List of Remarks pp. & subjects of Remarks:

.00

1) List of Induction Algms 228.00 ; 150-151 ;

2) EBL 297. , 308, 278

3) "Cure Cancer" : Pwr's(soln. 243.00

4) Loop 293, 294

5) H.W.: Programmable Gate Arrays 298.12 : Olde HW idea + see MIKEBOX "A" ~ 1990-1992

6) On how to deal w. cc in TM : 285.30 [How to put cc in Global Gate]

[7) Combining products of IA's ! 272.28 ; 273.11

8) Early Stochastic OPS (MOPS) : Calculus , ~267.35 ff.]

.10 → 9) How to use t. Permutation of Az 1 to derb. Recursive Pwr's

~~[illegible]~~

44! : 354.20

10) Stochastic Operations & some possible forms: (331.26) (340.31) ∃ clustering. ∃ e.g. very general discussion: (375.20-.30) & counterexs very strong

11) On Context as "conditional" for PC of Concs: for Updating Algm: 300.26 & stuff surrounding it. ← Very impt!

.16 12) UPDATE Algm (perfect if cc=∞!) Given P.D. on O_n; (esp. wts. for all possl O_n's) & = W_i^n = wt. for O^i

[O^i = subset pairs] $W_i^{n+1} = W_i^{n+1} + P_{O^i}(A_{n+1} | Q_{n+1})$. Algm. for updating O_n may not exist,

but algm. for updating D.F. on O_n is simple for cc=∞; but may be more diff for cc<∞!

13) "Final Soln." to UPDATE PROBLEM: Claimed by 300.35-301.20; 302.10 ff

Hvr, what about t. very time consuming diffy of finding $P_{O_n}(A_i | Q_i)$ if O_n is is a Monte Carlo device) 375.20→40 discusses this.

.20 14) Logical, Mathematical reasoning used by TM to solve Induction(&Other) problems! 224.04

15) (One Main) Track for QATM project: 378.00-375.40 : "Needs work" But Good general pap.

.30

Inadequacy of present Model for doing a Common Heur: .00
possl. Adequacy of Plans for UPDATE ALGM

.00 : SN I had this idea that trng a context-dependent pc's of concs, might be enuf to get a fairly smart TM. Perhaps NOT! This "Trng" that I'd considered started out by doing a trial of the Best looking Conc. (Greedy!). Then, on basis of what trng (and, from this trial), attain TM tries for the best looking Conc. (Greedy!)

Trouble is, it Doesn't do experimenting ever! Its first trial was not an experiment, but a trial to get the final soln.

Many hours are in the form "Do α, then (on basis of what happens) do β (α)". I don't see this model discovering hours of this sort!!

One hope is that since the QA problem model can express any problem — then we should be .10 able to get TM to learn do experiments. This seems to be an important aspect of it.

["Look ahead" problem — a very sophisticated case is "Core concept" → (243.00)
292.15 discusses "Limited Look ahead".]

[When TM is looking for a good Update algm, it could use "Look ahead" or any technique; a problem is: in order to be able to do this usefully, we'd have to give it problems like that in its TSQ. This might not be so difficult! TM could "watch" "Experimentation" being done in the soln. to a problem — (perhaps this is one kind of "Hint")]

9/8/01 SN .20 The Main problem would appear to be [Finding a reasonably O.K. "Updating Alg" — Good enuf so that "Bootstrapping" is possl. in a useful way. —

T. important features of the Update Algm that I now (have/understand.)

1) T. "cc/free" Core of the MDL (the cc would be inserted automatically if Lsrch was used)
2) I don't see how to insert cc in to this Core (285.30) → Normal Lsrch is one way, but GA(3) may need an algorithm expression for cc and performance in its Core, using a mechanism of
3) T. possy of using GAW. large CB (say CJS ≈ 10^15) (Pb. [This/copy should] be necy, it would make ■ doesn't the TSQ a lot easier!) At any rate, # 1) & 2) make this a real possy! .25
4) ■ T. foregg. ignorance exists assume "Global" search for updater using best to problems of "Backtrack" (not very (expert)). [Spec] 443 Kpsy Bilbao J. (412)
5) A real Automated Updater algm would use $O'_{n+1} = O_n(O_n, Q_{n+1}, A_{n+1})$. [a possibly other "Context". expanded idea of what can be "Context" maybe] [find Relevance on this! 300.26]
6) I Expect to get 5) by study of TSQ's. Any details of 5) such as Context dependent pc's of concs, learning during Lsrch, Look ahead; I expect to devise methods to do these by observing Beam being done in TSQ's. This is the Main Idea (≡Hope ☺). That I can have a variety of IA's in the system, should make it possl. to include any type of Heur that I might need. .30 [5 T. Heurs needed for]
7) In (6): T. mode of "insertion" of a Heur; Try to see how the Heur might have been devised & insert the simulated TSQ in the Lsrch to find it. This will (perhaps) give rise to "parameters" (pc's) for the heur, so it will be properly used.
8) Definition of a "Heuristic": anything that enables cheaper solns of (1), 2) — the Global Conc. [Eqn of O']
9) Possy of using "Patron pc's" to get ■ cjs ≈ 10^15 ("patrons" are allowed to give TM occasional problems to "observe its progress"). (see 370.00ff) [Spec. 370.00]
(10) (384.06-.11 also 384.12-.16): Solve Q_{n+1}, A_{n+1}, then find way to recognize Q_{n+1}
(11) [(378.00-379.40) Update d.f. of O^i by $a^i_{n+1} = a^i_n + \rho_{ij}(A_{n+1}/Q_{n+1})$ (5) Then "Expand" the O^i d.f.
385.00 discusses (378-379) V.S. (300-302)]

ID

Practical HW cost estimates: ill ff

3 69.25 is most relevant !

.00: 369.40: Discuss of 369.30! 369.25(3) in particular! I was impressed by koza's being able to get GA to do what appeared to be "Hard problems". This could be an illusion! e.g. koza may be faking it.

(b) T. problems be solved in ckt design (e.g.) may not be so hard: Real's may/probably may not Genz. so other Domains (...?) ← (Re analysis of Prob/systems as "Circuits" — be Scaling to be effective over several domains! i.e. Gabriel Re van (Tenure Answer at Networks): I think he did try to use his T.A. for Gen Relativity: Also his Guy from France who gave talk at IDSIA when I was there. But first, I'd have to Go over koza's Methods w/ some Care! How New Functions are created, & how his "Circuit Grammar" works

Circuit Grammar.

Hm. Koza doesn't seem to understand induction (& GA) as well as I do — so I should be able to get GA to work more like AI in recognizing/discovery using subtrees as other "sub-functions".

I'll be able to get 20x5x10^9 for ~ $7.5K in a yr. from now (see .12 - .21)

.10 Also koza ⊗ Didn't use Machine lrgs: He used Lisp which can be rather slow. Hm, I think to mean

.11 bottleneck was spice — & overall Gone. I may be able to speed this up much.

.12 W. 5x10^9 ops/sec, it took him several wks to solve a serious problem. I can certainly get 5x10^9 ops/sec inexpensively, & probably 50x10^9 for ~ $50K or less. (or 10" sec about now, for ~$50K)

100 Ice Mother boards: perhaps wood ⊥ bookshelves to "hold them up".

Fuse: try 5 or 10 more bds! Ask Seth to help. Note: I don't need sound or video on Bds. — Some Asst MB's come w/o Sound & video — just plug-ins for RAM, HDD & floppy.

"Sharp sound" for appropriate MD. Say $100 for MB, $40 for .25 GB ram; $50 for 20 GB HDD $100 for 1 GH cpu Sat 300 for 1 GH !/$ 30K per 100 GHz. = 10". Ram: 25 G, HHD = 2000 GB = 2 TBy

Look at speed of HDD for parallel Lisp even! But HW choice will depend (& Rina) on just what Stock Algms I use.

.20 Also, in 2 yrs, prices (will probably) ÷ 2 & speed avail to be for $100 cpu will be 2 GHz!

.21 So probly 10" GHz for 30/4 = $7.5K. or 10^12 for $75K. 10^12 GHz for $75K.?

(layer kernel) + cost for electricity + Hardcoding + Hardset speed, cost

I MAY want to do not too narrowly (If Pubs) / PC ordering not AZ(but ordering by heuristics) since this is too Greedy & not "creative"! So may be do (PC)^α ordering w. α ≥ 1. (danger of non-convergence?).

Also, Examine koza's "Circuit Grammar" w. some Care. Even if I can't figure out how his Grammar works, I should be able to devise a "circuit Grammar" that would work. It might be useful to look at how "SPICE" solves ckts!

.29

.30 Ok. So where am I now? Last was at 368.30: Examine how pc's are assigned to functions.

This seems "under control", How to be rewritten.

The flow: perhaps Go on to next step. This expo follows: start 347.00 - .80 starts section on "T. indicies Algms"
361.24 — 363.03 : starts by saying "I will discuss 3 of T.LA's in some detail"

362.07 - 363.03 and 365.02 - 366.12 Cover ≈ Same Ground: Cover ≈ same Ground: They Overlap much. ⊗

So perhaps read thru this carefully & rewrite.

Perhaps first read earlier expository stuff: start: 332.00 — 333.13 ⇒ 347.00-.14 ; 361.24 - 362.06 (362.07 - 363.03) || 365.02 - 366.12

374.24

9/9/01

ID

3×(1+7)

listserv machine to previous pp.

362.13 : Explain what we mean by "economic". corresponds to a "shortcode" — tends to express regularities in the domain. It assigns shortest possible pc. to most.

.00 332.00 – 333.13 [section] Intrdctn! Data objs Q.A. Tsq. problem. Intrdctn of

O_n, O_{n+1}; The 2 main, critical equations.

347.00 – .16 [section] Induction Algorithms. General discr! How we use several IA's in it as "specialists".

361.29 – .36; continue or / section of 357.00 – .16

368.29 – .36 [section] Intrdctn. SAZ system. Introdctn of SAZ

362.07 – .13 More discussion of SAZ, how complex functions are described.

↑ Generators by Program [LISP]

.20 362.14 – 363.02 A first attempt to explain assignment of probs to SAZ functions.

365.02 → 366.12 stores 362.14 over again — it only partly analyzes the functions.

T. Teller 368.00 – .15 is a much clearer way to present much of this — but I should use my corpus.

.15 Rewriting 365.02 ff!

An apriori probability is assigned to functions such as O_n, in the following manner:

We start with a list of primitive functions and primitive constants that are usable in the function definitions.

"Primitive functions" could be X, sum, minus, mul, div, sin, sin^{-1}, ...

"Primitive constants could be ϕ, 1, π, e, ...

.30 We will use Polish, [parenthesis-free] notation to define functions, so 3×(1+7) is written mul, 3, sum, 1, 7

sum, mul, ... commit commas! use spaces

To define a simple function, we will write a list of primitive functions and constants it might use, followed by the function definition in terms of those primitives.

X, sum, mul, ϕ, 1, F_1, sum, mul, X_1, X_1, mul, X_2, X_2, Δ defines the function

$F_1(X_1, X_2) = X_1^2 + X_2^2$

The "Δ" symbol indicates that the definition sequence is completed.

For example, to define $F_2(X_1, X_2, X_3) = X_1 \cdot F_1(X_2, X_3)$ we write the definition of F_1 followed by the definition of F_2 in terms of F_1:

(commit commas, use spaces)

X, sum, mul, ϕ, 1, F_1, sum, mul, X_1, X_1, mul, X_2, X_3, F_2 mul X_1 F_1 X_2 X_3 Δ

N.B. F and X all have subscripts

.30 To define more complex functions compactly, we will define a sequence of functions and constants culminating in the complex function we want to define.

The definition of each function or constant in a complex definition can only refer to symbols that have occurred earlier in the definition sequence.

At each point in a definition string, only certain symbols are legal. Using illegal symbols results in a meaningless (uninterpretable) string.

Except for a special class which we will discuss in the next few examples, the probability of each symbol in the definition string is given by the formula,

Some symbol, α, occurring at a particular point in the definition string is

.00 : 371.40 : ~~[illegible]~~ m/n

~~[illegible]~~

Here n is the number of times that α has occurred previously in the definition string.
m is the total number of times that all symbol types that are legal at this point, have occurred previously in the definition string. This probability assignment is similar to "Laplaces Rule".

.06 IR [Insert 367.30-.39]

There are three kinds of symbols whose probabilities are assigned in 3 different ways: X, Δ and $F_{i=}$ subscript.

X and the.

We are also making ~~[illegal]~~ constructs ~~[illegal]~~ illegal if they follow f_i. This is because

.10 the ~~[illegible]~~ function defined would be redundant; e.g. F_i followed by 1 ~~[illegible]~~ simply ~~[illegible]~~ defines F_i to be the constant 1.

The symbols in 371.30, the definition of F_2 are assigned probabilities as follows:

F_1 is given probability 1, since it is the only possible symbol at that point

sum is given probability 1/(1+1) since $n_{sum} = 1$ and $n_{mul} = 1$

mul " " " 1/(1+1+2+1+1) since $n_{mul} = 1$ and the legal symbols X, sum, ~~[illegible]~~ φ, 1 have all occurred once before that point, and mul has occurred once.

X is given probability 1/(1+2+2+1+1)

the subscript 1 # is mandatory, so it has probability 1

X has probability 1/(2+2+2+1+1)

.20 the subscript # has probability (2-1)/2

mul has probability 2/(3+2+2+1+1)

...

Δ has probability 1/3 since F_3 would be ~~[illegible]~~ also legal at that point.

Table 1 summarizes the probabilities of symbols defining F_2

The product of these probabilities gives a first estimate of the a priori probability of F_2.

Because F_2 can be described in many other equivalent ways, this first estimate has to be multiplied by a sizable factor to get closer to ~~[illegible]~~ a priori probability.

If we want to include recursive function definitions, we must modify the rules determining which symbols are legal at each point, ~~[illegible]~~

.30 The forging rules are only for functions that are formed by compositions of other functions. If we want to include more complex recursive functions

(Table 373.01)

9/9/01
ED

Table 1

Symbol	Legal possible symbols at this point.	Probabilities
X	X	1
sum	sum	1
mul	mul	1
ϕ	ϕ	1
1	1	1
F_1	F_1	1
sum	sum, mul	1/(1+1)
mul	X, sum, mul, ϕ, 1	1/(1+2+(1+1))
X	X, sum, mul, ϕ, 1	1/(1+2+2+(1+1))
subscript 1	1	1
X	X, sum, mul, ϕ, 1	2/(2+2+2+(1+1))
subscript 1	1, 2	1/2
mul	X, sum, mul, ϕ, 1	2/(2+2+2+(1+1))
X	X, sum, mul, ϕ, 1	3/(3+2+3+(1+1))
subscript 2	1, 2	1/2
X	X, sum, mul, ϕ, 1	4/(4+2+3+(1+1))
subscript 2	1, 2, 3	1/3
F_2	F_2, Δ	(2-1)/2
mul	sum, mul, F_1	3/(3+3+1)
X	X, sum, mul, ϕ, 1, F_1	5/(5+2+4+(1+1+1))
subscript 1	1	1
F_1	X, sum, mul, ϕ, 1, F_1	1/(6+2+4+(1+1+1))
X	X, sum, mul, ϕ, 1, F_1	6/(6+2+4+(1+2))
subscript 2	1, 2	1/2
X	X, sum, mul, ϕ, 1, F_1	7/(7+2+4+(1+1+2))
subscript 3	1, 2, 3	1/3
Δ	F_3, Δ	1/3

9/9/01 ID

Florence Nightingale

.00: 3 29.40 It will be noted that we have described techniques for assigning probabilities to functions of any finite number of variables. These are deterministic functions — but Oh, The function of interest is a probabilistic/stochastic function of one variable. There are several ways to convert deterministic function generators to probabilistic/stochastic function generators. We will describe one very general Monte Carlo technique.

To obtain a very general stochastic function of 1 variable, Consider all deterministic u.c. Math functions of 2 variables — The first variable is the Qi in our "Q# machine", The second variable, R, is a member of an infinite, discrete string that is a unambiguous infinite prefix set.

.10 If we have a string and is in which each of the variables is a member of a prefix set (not necessarily the same prefix set for both variables). The first variable is Qi, the input question, and the second, is R, a random number. If F(Qi, R) is allowed to be any/arbitrary function of two such variables, we can find such an F so that a random distribution on R will induce a distribution on F generates identical to any finitely describable stochastic function of Qi. By repeatedly giving different random R values for a given Qi, we get a Monte Carlo distribution of Ai for that Qi.

.17 We may regard the foregoing description method as equivalent to a 3 input, Universal Turing machine with unidirectional output. The machine has 3 inputs and The first input describes the function F, as does our AZI language SAZ The two other inputs are Qi and R. standardize! (what parameters are used for AZI or SAZ or whatever -

.20 Perhaps do .17-.20 first, then say that AZI language is a way of implementing it. → 376.32

.24: 370.40 Anyway: Where Am I now?: → 370.30: I guess I've just finished part of AZI on step by step of Q# functs. & prip evaln. of functs.

SN My def of SAZ Didn't go into its details of Recursive functions. → (354.20) each

I doubt if this will be a bottleneck of actual lang defn. or for Exposition.

Topics: How umc's do Q, R → A:

problem: we need P(A|Q): The MCarlo form does not readily Give it. [375.20ff has a lot on this Q]

.30 HVr, in practice, it's almost always possl. to map this "3 input form" into any of these (cogit V.S. — but vice versa. Hmm! — changing 3 input into non-Mt Carlo forms is not so easy!

I think I did write a lot about this — How to Get useful pds in useful forms:

300.35 ff · · · · Bacon PP purports to solve the update problem Using MCarlo & Gren. — My impression is that cc would be too large! — Unless Only a few new QA's had to be evaluated. This may be v. much!

Certainly all the Qi,Ai pc's can't be re-evaluated for each new On trial! → (258.24 Discusses this)

Perhaps that was what 300.35 was all about! Must I assume? Align:

Also Note "Perfect Update" of $(368 \pm .16)$! 271.31 considered that various IA's would have different "formats" for P.D. outputs Mt Carlo, listed Ai's & PD orders. μ, σ^2,

9/10/01

ID

Graeolas 371, 2, 3

On ANN

.00: 37 ff to : **On ANN:** I only have a vague idea of what how I expected to get a IA out of ANN. Tho I usually don't use cross validation, I think I had devised a way that was able to use it in an IA — in General, in ANN one can either use small (no. of Nodes & edges) & use Max Flatness, i.e. MDD; or use lots of nodes, edges & division of data into typ. 3 test sets.

Small updates & modifications: Lengen, add, delete wts, Nodes, & s/o RANN connects. Perhaps add, delete memory / change Memory types. Change Ranges, Squeeshy function.

The goal is to get the ANN to work in the setup QA context: We want to update algm. to look at present state, recent QA's, latest QA, & propose modifications of that.

.10 Testing would be especially statistical on subset of QA's.

Some possl. diffy in Max Approach: in RNN we usually have not able to localize just what is being important in a particular QA. IN AZI, this is done by noting frequencies of use of sub-functions.

However, to an initial Extent & Page, discuss a bit "premature" The main methods of "update" will be obtained by studying human search Heros & finding ways to express/guess them using AZI, GA, ANN or any other IA's that may be suggested. Though, to implement a Heros, one usually doesn't need an entire IA (?).

On $P_{On}(A_i|Q_i)$ as a (Universal) stoch op. (other than the standard 3 input Umc). ABcdefghij

.20 244.07 had some good ideas. Also Note (244.02–.25)R

One way is via Stoch Grammars that are derived from Determ. Grammars by using probabilistic rewriter rules: to find (some form of) pe of a as, parse it all possl ways & insert proper p.c's. Also note 376.07: on 3 input Umc w. R as "connected" to central Ai. That it is a kind of "clustering" approach to inductive coding.

Another way 250.09 ff, 253.37 — Discussed in 266.40; on what Paradigms are universal; 267.35 ff (they may be, but still not be good Heuristically!)

.25 → I think there are several notes on counting P.D.'s on strings, etc.

331.26 How to convert a Map into a MDP. (on clustering 376.07)

.28 340.31 On "clustering" as a kind of stoch d.f. & its use

.29 Perhaps one (very) impt earlier idea! That as I observe heuristics in human prob. solving, & devise tsQ's that could have given rise to them, I will discover many impt forms of stoch operators. That I will depend much on this idea for evolving a V.G. TM.

.30

Another Note! The 3 input Umc is not very slow if the thought-for Ai is not by PC! Usually this will be the case. Also, there may be several equivalent Ai's that are Sat's Factory

9/11/01: Some diffys in the report Register: (A in my plan for CPM, in General)

1) As derived, the SAZ (eg) is not Univ. — it didn't do recursive funcs

.33 2) The 3 input Umc for the general stoch operator seems to take too long to evaluate → 376.10

$\prod_{i=1}^{n} P_{On}(A_i|Q_i)$: In general, the 3 input modul may be very slow in evaln/use. main

Replys 1) This is not difficult to fix: The point is I do have a way to assign a priority to terms of (all) finitely derivable functions.

2) (a) for many input update schemas (usually most) Not many $On(A_i|Q_i)$ have to be recalculated for a new trial On (Also Note .25–.28)R

(b) see .20 – .30 for some ideas on other forms of P.D.'s (.28 – .30) is my main interest (expected solution/hope)

Seems very Reasonable, very Practical!

ID

CPC Calculator computer

cpc

.00 375.40 : A Better way to organize & report!

1) Descrb. problem (as in introduction) of $[Q_i A_i]_n$ input & Q_{n+1} query/ans.

$U(O_n, Q_{n+1}, A_{n+1}) \to O_{n+1}$

2) A ~~system that~~ way will learn IA & system that's able to do this to any extent.

Discuss IAs as in §2.

Also Discuss & Gore$ (cap. 1)

That for a large class of IA's, this Gore is Approximable.

Update algms like $U \cong (O_n, Q_{n+1}) \to O_{n+1}$

U need not be a IA. i.e. it would not learn

In general, & U algm is not really legitimately possl.

or

Ex/ Lisator, perhaps (but note .10 !) $O_n(O_n, QA_{n+1}) \to O_{n+1}$.

3) There are 3 aspects of a IA that characterize its behavior: ① initial O_1, ② TSQ ③ update Alg.

~~[illegible]~~ & start of ⑤ consider ①. & big problem is ③ (see .10)

.07 4) Explain that ont. 3 input Umc, The R input can be regarded as "correction" to & "control" As output — which occurs when R represents ~~[illegible]~~ zero. (R is is not R ≡ null, hrr. — it's & representation of Zero in & prefix code being used. "Clustering" is one example of this kind of coding. The R value represents & "Mean" of & Cluster

.10 375.33: 3) $O_n(O_n', QA_{n+1}) =_{df}$ or O_{n+1}' is incorrect:

.11 $O_n(d.f.\text{ on } O_n', (QA)_{n+1}) \to$ d.f. on O_{n+1}' is correct. It is & d.f.'s of O_n' that are recursively defined. equ. .11 is easy to write out exactly for cc = ∞, but not so easy to implement for cc < ∞.

.13 Hrr, it would be worth looking into as an approxn for updating "d.f.(O_n)" is ~~defined~~ developed recursively.

I ~~may~~ have done .13 when I was considering "Min Backtrack Algms". My impression at that time was that I ~~could~~ couldn't get much out of it. With present view, however, It might be possl. to Revive it. ("Revisite" is.) I don't remember exact reference. I think it was at IDSIA or past IDSIA.

Also Related is & idea of a univl. stackOp being Expressable as a simple P.D. on all D.OPS. (rather than a P.D. (wrtd. mean) on All stackops — with list of normal input umc)

.20 374.34 370.40 Where I am now in report: Just finished how talking how pc of SAZ functions are Evaluated.

Brief mention of how 3 input machines works

.22 Actually I have to rewrite (374.00 – .20) on how we use SAZ to get a stackop.

New approach: Discuss 3 input umcs, ~~[illegible]~~. Then discuss Gore: Global soln, then Incremental soln.

Then discuss SAZ as a practical way to get a fragment of a universal umc.

Note that the formalism used for assignment is similar to Lap's rule and treats & symbols in & term somewhat like & symbols in a Bayes seq.

While .22 ff would indeed be clearer & halfway more useful) in & mtb it would be longer — & take more time. what I want to do now is finish a "first draft" as soon as possl.

.30

.32 374.00–.20 updated stab as pap: Section: Probabilistic Operators:

We have shown a method that generates arbitrary functions of any finite number of variables and a way to assign apriori probabilities to those functions.

We will now show how to obtain very general stochastic operators and associated apriori probabilities from those functions.

First Consider the very general

ID size

.00 : How to Explain this ?? : $U(X,R)$, by varying X, we can get arby stoch relation betw. R & output. I could "state w. proof --- Pco we ~~have~~ do have a proof"

since Σ error is bndd a good TS Q Arranges to / ~~[illegible]~~ errors in induction on itself to be max, so it "uses up" error — so it's more accurate later. Examples in which TM makes most errors, are most instructive. Hvr, I need to look into this More Carefully!

On second thot, not unreasonable! If TM has a wrong idea in a certain area, giving it a Q & in that area allows it to make a big mistake and learn from that Mistake. It amounts to getting low wt, quickly, for those components of the universal D.f. that are "Bad". We can learn from mistakes if we find out

Though ~~a proof has been found~~ we do have a proof, we will state without proof!

~~If $P(Q)$ is any [illegible] prob~~ $P(A_i|Q)$ ~~[illegible]~~ a probabilistic relation between Q and A_i.

.10 Suppose that ~~[illegible]~~ is ~~some~~ probabilistic function of Q. This relation can be expressed in several different forms, eg for each Q:

1) for each Q at pair, ε gives a probability ~~value~~ p_i for A_i such that $\sum p_i = 1$.

for $P(A_i|Q) \Rightarrow \sum_i P(A_i|Q) = 1$

2) For each Q, we get a listing of A_i, $P(A_i|Q)$ pairs in approximate $P(A_i|Q)$ order.

3) ~~For each Q and every random number R, we obtain a P_i value and the probability of obtaining that P_i is $P(A_i|Q)$.~~

3) There is a ~~[illegible]~~ function $F(Q,R)$ ~~[illegible]~~ w.r.t. the following properties

The Q argument is ~~[illegible]~~ a binary string, a finite 0,1 string, a member of a certain prefix set.

.20 The R input is any semi-infinite ~~[illegible]~~ binary string.

If the R input is a random string, the output for Q will be A_i with the probability that A_i will appear as the output is $P(A_i|Q)$.

Theorem: for Any probabilistic relation in form (1) or (2) there exists an $F(Q,R)$ corresponding to them.

Conversely, for any probabilistic relation in form (3), there ~~is~~ can be constructed ~~a~~ relations of types (1) and (2).

.26 Theorem: $\exists$ a universal conditional probability distribution P_u, ~~[illegible]~~ $P_u(A_i|Q_i)$ such that for any finitely describable $P(A_i|Q_i)$, and ~~any sequence of [illegible]~~ any finite set ~~[illegible]~~ $[Q_i, A_i]$, $i = 1 \ldots n$

.29 $$\prod_{i=1}^{n} P_u(A_i|Q_i) > c \prod_{i=1}^{n} P(A_i|Q_i)$$

.30 P_u ~~[illegible]~~ $(A_i|Q_i) = \sum_j a_j P_j(A_i|Q_i)$ Σ is over all finitely describable P_j; a_j = total p of all descriptions of P_j ($\in$ 1)

Hvr, .30 doesn't imply .29.

.29 is probly wrong. Lets look at perfect "updater schema" (for $c = \infty$).

Eq a_j are initial ($s=z=0$) wts of ~~[illegible]~~ P_j.

.34 (.37!) Wt. after $s=z=k$ is $a_j \prod P_j(A_i|Q_i)$: So I guess the product is the wtd mean ~~[illegible]~~ over all j → 378.00

Probly go back to this "mood" show form for Bye induction (Problems in "2 kinds of Prob. Ind")

I think I considered all ~~[illegible]~~ stoch ways, L_j — L_j gave a pr of P_j^k to finite strings "k" (There are n distinct finite strings of length k. Simple $n = 1/n$.)

.37 Wt. of L_j was $a_j \prod_{k=1}^{n} P_j^k$ (.34) → has nice idea. for QATM → (.34) →

A MAIN TRACK for QAT^n : 378.00 – 378.26

↳ Theoretically optimum system, for cc = ∞. SEE 379.30 for more recent developments. (probably meant 379.26)

.00 (377. spec 34): So, say $O^j_n(A_i|Q_i)$ is the j^{th} possible operator ($j = 1|\infty$). It starts w. a^0_j = init. Universal D.F. for $n = 0$.

Say a^n_j is wt. of O^j_n for corpus (Q_i, A_i) $i = 1|n$.

$a^0_j \equiv a_j$

.02 $a^{n+1}_j = a^n_j \cdot O^j(A_{n+1}|Q_{n+1})$ ← This is the "update" rule $\equiv \| a^n_j = a^0_j \cdot \prod_{i=1}^{n} O^j(A_i|Q_i)$ ← same as $a^0_j \equiv a_j$

.03 T. prob rule is $A_{n+1} \equiv \sum_{j=1}^{\infty} a^n_j O^j(A_{n+1}|Q_{n+1}) \Big/ \sum_{j=1}^{\infty} a^n_j$ } This is the normal Bayesian Rule

(← probdist)

I think .00 ff is the correct system for cc = ∞. It would have the corrected bounded $\sum_{i=1}^{n} (\text{Error})^2$, for the preducing of .03.

.06 If we approximate this w. finite j range, would it work at all? If we did only 100

.07 best O^j's we could find, say. Well, what I expect would happen: As $n \uparrow$,

.09 the d.f. gets narrower & narrower, & we end up w. a single O^j. Past O^j becomes constant,

.10 & doesn't "learn". (AT THIS POINT we "BACKTRACK" to the no. of Available O^j's. — see NIPS 195.00

It may be not so bad if O^j if "100" is large, say 1K or 10K),

.11 & it is TSQ is fairly long when "convergence" occurs. → Hmr. This "convergence" may not occur! see 379.33

One possible remedy that I hadn't considered: When O^j has converged, backtracking can take 2 forms: (1) conventional: Go back to $n \to n-k$ and try O^j's that hadn't been considered (2) new: " " " " " " " " " " " " O^j's but w. larger CB's — which could discover new ways by Which. At first glance, it seems like a better way to expand the search than consider "new O^j's" — This last is too "multidimensional" an expansion of the search. More "clues" would be needed to guide the search.

A more "continuous" way to prevent (.08–.10). Do not "continuing" expansion of the search, so that we have at least 100 or 1000 reasonably good O^j's at all times.

.20 We could do this expansion by CB on recent (n to n-k) "rounds".

.22 Could it be possible to get clues on how to do the "Continuous Expansion" &/or "Backtrack" by

.23 observing my own Problem Solving Behavior?

.24 My impression of my own prob-Solving behavior: That I have a large bag of heuristic tricks that I am

.25 continually improving & "Adding to" (Augmenting) — and that's it! Some of the tricks are very "Global", Hmr.

— That I am designing TSQ's w. the idea of bringing about .24–.25. To start w. a few (perhaps)

such heuristics, & build them up via a suitable TSQ eventually by "Bootstrapping".

.28 Could any of this be brot to bear on .22–.23? (I think so: this (should) be a main direction of Rsch)

.29 One kind of common common d.f. over complex objects is "Clustering" by choice of central pt $\vec{m}$,

.30 & a "metric" σ^2. In the pleasant case, $\vec{m}$ could be, for O_n: $\vec{m} = O_{n-k}$. The "metric" $\vec{\sigma}^2$,

would be the set of functions to "expand" O_{n-k}. However, it is clear that just doubly

Composing" O_{n-k} w. various functions is inadequate. What we need is more like O_{n-k} + fine tuning to it

→ an objective stochastic set. This is more like Mutation/crossover or a variety of "cond'l.

prob" (which is what $\in P^5_{n+1}$) is supposed to do. I.e. $C'_{n+1} \stackrel{df}{=} C_n \oplus (C'_n, (QA)_{n+1})$

Note (!) a deterministic $\oplus$ $F(x) \to y$ applied to the D.F. C'_n will give a new d.f. on C_{n+1}

.39 Hmr, would like an $\oplus$ that takes each C_{n-k} & inputs it into a d.f. of width that is as

.40 much as Q_{n+1}, A_{n+1} info reduces the width! — So on the average, the width of the d.f. stays ≈ constant.