Q: If 209.06 ff solves t. problem! Than shouldit not solve 208.22 ff..?
i.e. T. problem of assigning p.c.'s to various spaces of coifs, ect. — The divergence of various S's, ect... ?

The demo of 209.06 — .40 can be divided into 2 parts! ① The p.d. that we got for t. next day as determined by a given ktup. ② T. wt. (apriy) given to t. pred d.d. of each ktup. 1:45P

The Maxm factor $\frac{N+K}{N-K}$ is part of ①: It is not necly related to ②.

$\sum_{i=1}^{10} \frac{1}{i} = 2.9289$
$- \ln 10 = .624$
$\approx \gamma$
$\approx \gamma \approx .5+$
$\gamma \approx .5772$

In coding & giving wts to various Model types: Note that Σ of all wts = 1
t. implication is that one should use all of t. models if possl. — or as many as possl.

The coifs of P(1), P(2), P(3) (for t. no. of coifs) also, all add up to 1.
Than for each no. of coifs; t. wts of all t. models considered, add up to 1.

Another Q: These apriori wts are impt., but ssz supposed to "take over" at large ssz.
Just how does this work? — Well, it may not, if t. ssz's of competing models are all t. same, (or in constant ratio).

If a very large no of models w. poor prody. capabilitys all point in t. same direction, They can add up to a strong., narrow, prediction.

.20 So: Devn of a methodology for prediction, using a large no. of models:

The models are divided up into sets that have t. same no. of continuous params,
Say S^i is t. set w. i params. M_j^i is the j^{th} model w. i params.

We find a way to assign pc's to the S^i: P(i) t. apriori on positive integers is a default way.
$\sum P(i) = 1$ of course. Once an i is chosen, t. total pc of all models w. that i, is 1.
So we have to assign pc's to this set of i param models.

We have to select i param types. Each type may be finite choice or ∞ choices.

.28 In t. case of using many ll translation series to predict 1 time series, using Multilinear regressn: Each model is charakterized by k difnt Time series that are used for pred. Since order is irrelevant, use 209.31 to get p.c's

Say we order t. time series: Say P(n) is t apriori of n^{th} t.s.
Say consider a set of k T.S.'s m_i (i=1/k) are the order nos. of t. k T.S.'s.

$\ni m_{i-1} < m_i < m_{i+1} \cdots$ Than t. pc of t. k tupl. is $P(m_1) \cdot \left(P(m_2) \cdot \frac{1}{\sum_{j=1}^{m_1} P(j)} \right)$

$\cdot\ P(m_3) \cdot \frac{1}{\sum_{j=1}^{m_2} P(j)} \cdots\cdots P(m_k) \frac{1}{\sum_{j=1}^{m_{k-1}} P(j)}$ | The $\frac{1}{\sum P(j)}$ factor arrises

because when m_j is selected, it is known that all values from 1 to m_{j-1} are now impossl.

So, t. Loredo. Tells one way to assign p.c.'s to t. (models / k tops).

Each K top is a K param model.

For linear & continuous N.L. models, we can use t. original methods (~1962) of Maxm. For k contin params, each w range $\pm\infty$; we use a k dim. box $(\pm R)^k$, w. a uniform apriori for t. k params. Using a Bayesian model for predn, this gives a apriori d.f. in t. k dim. space. For finite (& large) R, we are able to use each model to obtain a density distribution for each possl. pred.

T. way Maxm works: Say $[X_i]$ $i=1|n$ is t. T.S. to be predicted: we want t. p.d. for X_{n+1}. Using our model & t. uniform apriori for t. k params (out to $\pm R$, each), we obtain to total p.c. for $[X_i]_{i=1|n+1}$ for each possl. value of X_{n+1}. In general, these values of probty will be $\propto (2R)^{-k}$ — so they get very small for large R — but we are interested in t. relative probys of t. values of X_{n+1}, & these relative values become constant as $R \to \infty$.

So, for each model, we get a predicted d.f. for X_{n+1}. We then simply use a wtd mean of all models, using wts as described in 210.20ff.

n = 0 : x = 0
n = n+1
x = x + 1/n
print x - ln x
pause

To use a default d.f. for P(u):

Use: 1) Best.... an apriori based on previous experience in t. relevant area.

2) Riss' $\left[\frac{1}{2}\right]^{-\lg^* n}$ or my simplified version for integers (See my file on Rissanen)

3) $\approx \frac{1}{n} \cdot \frac{1}{\ln N + \gamma}$ More exactly: $\frac{1}{n}\left(\frac{1}{\sum_{i=1}^{N} \frac{1}{i}}\right)$

where N is t. largest value of n we need consider.

I need to compar log Riss or my version v.s.

$$-\log n \quad \text{vs} \quad \log\log \ (\sim \ln N + \gamma)$$

find cases in which one is "better" than t. other.

One criticism of my version: it does have large "discontinuities" — i.e. (large) ↓ in P.C. occasionally for ↑ of n by 1. I don't know to what extent Riss' method has this trouble (I imagine it does).

Also t. log basis of t. log used in Riss method is arbty. Lavin says is diverges if t. basis is e or larger.

210.28 ff will also work w. continuous non-linear models, "in theory"! But in practice, one will probly assume "local linearity"

To what extent can 210.20 ff be applied to prodn of AA's? (i.e. AAE problem).

SN The relative wts of 2 models: If one has smaller MSerr, then for large enough SSZ, it gets most of the wt: indep of apri wts of models.

For 2 models w. rms errors σ_1 & σ_2 resp, the max change in relative wts per data pt is $\frac{\sigma_1}{\sigma_2}$ —

For K data points $\frac{\sigma_1}{\sigma_2} = 1.01$ gives $\sim e^{10}$ change in rel wt.

9:07 click
cousin
9:07 click ... ~ "+10".
9:08:01P

.09: 10/1/97 Impt Q. I usually think of prodn in terms of stoch. loops: I fit some (or many) such programs to + data, then use those programs for extrapoln.

But would this work for "OSL 1 shot lrng"? "1 shot lrng" is characterized by: SSZ is so small that it must include the "data number one" in the extrapoln, before the relevant concept is of usable SSZ.

→ This seems to be Very Impt. in Search Simulation of GA.

Fn Extrapolating time series of discrete Symbols: We can take the last n symbols in the seq. & see if they have occurred before! If we find one only, then the previous continn. of that seq. gives us a SSZ=2, & is possibly useful concept.

If, here, we want to start a new program, Then we have to consider all previous programs in the corpus: Superficially a formidable task, — but short programs that have occurred often in the past will be much more likely, → which would not be "1 shot lrng".

Have I written about this before?

.21 One way to deal w. this in G. GA cond. Generation problem: For any continn. of a structure partially completed: One possl. continn. is to refer to some sub structure of the already partially completed object. If the sub str. occurs several times, its pc is appropriately Multiplied. We can do fractal derivs this way: I.E. a section of the object can be coded as a specification of part of the larger object (which may or may not include itself), "multiplied" by a (contractive) xfmn. Mce (Fractal) "Collage" Thm, tells when such a deriv. converges.

In " "Iterated functional Systems", a single xfmn matrix can derive an entire picture (i.e. "attractor"?). I would use a "top down" functional

.35 loop, to derive. Such recursive derivs. unorder! → 213.01

This part has spec that depends on n & on whether "self" or "n" is chosen.

In coding the n+1 B extrapoln. of a set of objects: A coder fragment could consist of 1) A spec that tells whether it itself or one of previous n objects is referenced 2) If one of the previous n, the pc. is then exactly n; 3) then instructions on how to find the relevant part of the previous object. (sometimes just where it starts & stops: for a finite object, this is easy to code.

10.12.97 TM LHL.

.01, 212,35 An impl. part of t. problem of 212.09 ff, is putting rules in p.c. order, if t. choice branches don't be all have same choice branches. This occures in stoch cfg's, and it makes t. problem of finding t. most likely as, not so easy!

.11 solves this → HVR: NOTE .30

The problem of ordering trials! I have solved many times (once this yr. I think), if each rule is t. cart. product of several Pr by vectors. — so each rule's pc is

$p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot$, say; Where p_1 can have n_1 known values;
p_i " " n_i known values.

picking t. hyest pc. as of a (non-consistant) stoch cfg.; If one can solve this, then one can find t. second most likely as, by finding t. most likely as in t. [illegible] from which t. hyest pc as has been removed. This s.cfg will of necessity be "inconsistant" (poss terminology)

.11 Perhaps when one must make a sequence of probilistic decisions (like for gg), t. best way to do Lsrch is by "timesharing" a/o "parallel search".

Can .11 be used to chose very hy as's of a stoch grammar? What would be analog of cc? One possi. way to do it "11 srch": start out & retain all branches: But do this so that the total pc of each branch retained up to present, is about t. same. When, eventually, one branch terminates, it will be $\approx$ maxprobty. We can continue other branches this way to obtain other "hy pc" ass (= acceptable sentences).

Pet

T forg. is practical in some grammars (that don't branch too much) but not practical in others. If there is too much branching, we may have to randomly chose subsets to branch into.

One way of thinking About it: At any time in t. production, we have branched a certain distance. Each branch will have a total p.c. thus far. We can pick t. hyest pc. branch & either

.215 (a) develop each of its sub branches 1 step only: This gives a new tree: loop to .215 and (c).

or (b) develop any sub branch that gives a total p.c. still $\geq$ that of any other sub branch. When this is no longer possible ---(its not clear what to do) — so alternative (b) is out of door.

But (a) is probly good enuf.

.176

Another approach would be closer to Kevin's T-share version of Lsrch:. — But

.30 Actually t. problem of "ordering probabilities" of, say a stoch cfg, does not occur in normal Lsrch — if we simply do t. time-share Version of Lsrch.

Anyway: A BIG Q! Does .30 (or perhaps .01-.28) Solve t. "1 shot Lrng" problem of 112.09? 212.21 seems most relevant.

0.18.97 TM SOY : Soy ABcd

In the SOY problem: When we observe [redacted] n's & σ's of yields of various States: Any mean ≈ its S.D. may be best derbd as a gaussian deviation from the mean of the Ensemble.
The idea is that we want a unified dcrn. of all of the data.

Why Koza's work is of interest to me:

1) The problems he solves are difficult.

2) The methods he uses to solve them can probly be modified & improved using ALP & Lsrch.

R. J. SOLOMONOFF
PHYSICIST
26 BOYLSTON STREET
CAMBRIDGE, MASS. 02138

3) His methods of using subroutines & other techniques suggest directions for useful development of ALP & TM.

4) The GA systems he uses are Universal & should map very easily into ALP ideas.

5) A big, common criticism of Koza: That he starts each GA problem solver w. an enormous input of info by chosing the representation of the problem - & the fund. operations used by the GA.

I feel that this is a difficulty I can & should address using Suitable TSQ's.

6) The GA method of solving OZ problems is not the best way! It uses something like maximum slope hill climbing in complexity space. i.e. given a point, X in config. space, with a certain G value, he chooses points at random w. frequency of their "conditional probty" wrt X.

However, it is ~~easy to do~~ relatively easy to find OZ probs for TM & allow it to solve them in this way — each problem's ~~the~~ a kind of self constructed Training seq. This enables me to get into TM inexpensively

7) Re: (5). Look at the problem of representation, ect. that Koza et al. border (may have) solved (Here) his GA on it. How can a TM solve this problem? What TSQ is needed? Are there other ways the machine can learn to solve this problem?

8) One known soln. to this problem is organic Evoln. How this very expensive & not directed in a way that need be useful to us. I'm more interested in a narrower goal: "T. Scientist's Assistant" — (The sometimes a more General Goal is easier to solve than a narrow one!).

ABCDEFG

Discuss w. Sasha!

1) On neg. cases: Easy to deal w.: all positive cases start w "1"
" neg. " " " "0".

Make single grammar to fit both kinds of cases.

However, suppose the + & - cases are "rigid": we want a grammar that fits data exactly (well enough just add in extra bits during "error". This is the "MTM" problem: — it has same soln. as NMTM problem"

2) If info is distributed. Fine: If each case has cost, then we want to make model that has best $\frac{\text{post}}{\text{comput. cost}}$ ratio: ~~Yeah~~ No! An inexpensive soln. is to use a small, cheap, corpus.

This may be the same as the (perhaps unsolvable) problem of Science" problem. How best do use our resources to get most knowledge: I have not been able to define the problem: It may be not really definable!

We may need a better defined Goal.

A better problem: I have cancer in my toe: What's best thing to do? How much time to spend (and how to spend it) surfing the net, browsing w library, ect; How much time on experiments, (how much to pay for MD advice) Medicines, ect.

Maybe regard this as an OZ problem ("Anytime alg"): use the shortest soln:

→ However, we do want "Learning" to occur during the soln., so perhaps use Genetic for a "mate soln": i.e. we look for a methodology: an "Action Algm": Is this getting close to Jaeger's LHL goal type?

Well, a v.g. Hill Climber does look at previous trials, tries to find out why they weren't so good, & uses this info for the next set of trials. This is not necessarily the same as improving the H.C. method.

Also, some v.G. H.C. methods can use info in the derv. of the problem itself. (cf. functional form of $G(\vec{x})$, the optim. function the "Gene").

N3097 TM

2162

From: "Alexander 'Sasha' Chislenko" <sasha1@netcom.com>
Subject: Re: Paper
In-Reply-To: <199711281936.AA02878@world.std.com>
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"

Ray,

I am still reading the printed paper you gave me.
It is such a pleasure!

A couple of things I don't understand so far:

- what is the a priori probability of a grammar?
- I wonder how Levin's abstraction search algorithm works...
- M(xa), etc. - how do the results change if we change the Turing machine
  and the way the grammar is described?
- "if h is the description of a probabilistic language and r is an infinite rand
om
  string then the probability that M(hr) will print out a string s and stop is
  the probability with which s occurs in the language"
  Doesn't this probability depend on the choice of s?

I would try to relate the probability that the data would be generated
by a given grammar, to the complexity of that grammar - but then, what do I
understand...

I wish you gave a class on your theory...

I keep reading...

Alexander Chislenko <http://www.lucifer.com/~sasha/home.html>

From owner-exi-east@extropy.com Sun Nov 30 05:06:58 1997
Received: from maxwell.kumo.com (gen101ip126.cadvision.com) by world.std.com (5.
65c/Spike-2.0)
id AA03723; Sun, 30 Nov 1997 00:16:39 -0500
Received: (from majordom@localhost)
by maxwell.kumo.com (8.8.7/8.8.7) id WAA27918
for exi-east-outgoing; Sat, 29 Nov 1997 22:07:54 -0700
X-Authentication-Warning: maxwell.kumo.com: majordom set sender to owner-exi-eas
t@extropy.com using -f
From: Simon Levy <levy@cs.brandeis.edu>

10AM 10:30

D11·97 TM : (Maybe) Great BREAKTHRUs ! (·01) & (·20)

·01 1) The main idea of showing that Lsrch for inv problems was ≈ optimum if all one's info was in t. p.d. —— was that if one had any way better than Lsrch, one had, by def'n, not included all one's info in t. p.d.

T. assoc. proof for OZ problems can (perhaps) be shown in t. same way! If one has a better way to solve OZ problems (say a better hill climber) than that hill climber, should have been in one's p.d. Any info about how one was able to devise this "hc", should have been in t. p.d.

→ As in INV problems, t. key ① is to ② show all heuristic or (domain-specific) factual info can be put

→ into t. p.d. —— also Give examples, problems, etc.

So t implication, then, is that t. techniques of §01 86, 89 may, indeed be ≈ optimum.

In General, it may be wise to think more about optim of OZ prob solns, since they

·19. are most common & include INV probs as a special case.

(D1897)

·20 2) This is an even more revolutionary idea! That t. T.Time shared Lsrch for both Inv. & OZ probs is ≈ optimum as is. No factor of 2 because of "50% soln". (I had within factor of 4 but this was because my "T ← 2T" method was with a factor of 2 of optimum! Later, hvr, I got t. idea that T.S. Lsrch was better than T ← 2T by a factor of 2 × (t. no of levels of (no "2" factor) ← 4/11/05 functional complexities used) — i.e. so T ← 2T was worse than a factor of 2 from optimum! — I haven't yet resolved this paradox!)

The reasoning is this! Consider INV problems! We give one to a TM that has solved many INV & OZ problems, a TM uses whatever info it has in its p.d. to help solve t. problem. We watch TM solve the problem, & you say "I know a faster way". If this is so, then I knew you must have learned this method somehow. If TM doesn't know that trick, it because he hasn't had t. (data/training) that led you to discover that "faster way". If we gave TM that (data/exp) he would "know that faster way".

Now the new idea here is that t. for. argt. can be applied to any kind of prob solving method or even to methods of organizing t. S.W. & H.W. of TM.

— So TM doesn't have to do t. "50% soln": He is given a problem (INV or OZ) & he tries to solve it as well as he can in view of his past experience.

All very well, but just how does TM go about solving a problem? ——

·38 In t. older "50% soln" method, TM would do Lsrch on t. present problem using t. present P.D. (Taking time T) Then he would speed time T, on [redacted] updating t. p.d. or more generally on "self improvement" (of which

219.01spec→

219.01spec

4TM

Spec.
20:268.40 : A troublem. 268.30: It's quite expensive to say which symbols to change and to what:
A better kind of S-function will have the symbols to be changed (indexed / Marked) — which is
like 268.28; but we could use a default def. for t. symbols or use t. d.f. implied by
a recent context.

Just how to assign pc to forgg. is unclear! Say I use a special Symbol in t. Code to denote the d.f. of symbol values. Say I have 4 of such symbols in my code for a cond. Then, to evaluate t. pc of t. "A" value that I need, I have to scan over all combinations of all 4 "special symbols" to find a comb. that gives t. vrite A. T pc's of "special symbols" is 1, (boost=0). T. d.f. implied by t. "special symbols" gives t. final pc of t.
code that produces t. vrite A.

One might have a standard "Correction Code" that is used to "xform an unsatisfactory A, into t. Correct A value.

→ How does forgg. relate to Maxent of pc of O^j functions in QATM ??
Do t. pc's of t. special symbols correspond to t. "R" values in t. simple model of S-functs?
Ideally, I'd like to be able to take any O^j function & find a pc (>0) for t. corpus w.r.t. that O^j function.

T. problem Seems to be: I have a few ways to represent S-functs: Now O^j's are S-functs. What are good ways to search over a space of S-functs, w. goal of obtaining Σ O^js that assign (w. its apr) high pc's to t. corpus. — w. populations or individuals
I have a few such methods: ① Lsrch ② Monte Carlo GP search, using Mut/cross ③ M.C. and search of outputs of S-Grammar.

Woops! In general, one must simply replace any symbol of a pf. in a funct. w. any other Symbol: we have to replace them w. symbols of same "arity".

To study S-funct forms: Give many Examples of S-functions that could be solns of QA probs.

1) In very early ANL, TM has learned exist. of +, −, ÷, ×, but doesn't know which to use! — so for all binary operators there are 4 poss's in some known express like α, 3, 7 (in PN) in which α can be t. add, sub, mul or div w = proby. T. probys themselves could be obtained by a Bern seq. analysis of t. corpus. "pattern" in t. sense of "Mathematics"!

2) Consider predicting a simple Bern Seq. w. known finite alphabet. We have a Null (= constant) Q. So We want an uncond. pd. This seems like a very fundamental problem.
How best to express a k-way probabilistic jump? Perhaps have it be a "primitive concept":
We have a random no. bet. 0 & 1. Then are k intervals on [0,1] & so t. random no. chooses one of t. k: w. pc (defaulting at to ...) Interval Sizes.
For t. Search process, we might start out w. k equal intervals,

3) One method of representation of S-functs and search for them is used of S-CFG's:
Start out w. a simple Bern model (a promiscuous model) — Then hunt for production rules.
T. "best" probys are computed by parsing & counting no. of times each choice was made. This is usually a reasonable Approx.
I'd like to generalize this-PCG to universality. Context Sensitive Grammars can be universal, — But I'm not sure (this is) a good, useful representation. So try other Modifns of CFG's that have more heuristic meaning.
e.g. "contexts" can be subtrees that were found in t. past, to be useful. → Spec. 270.00

9/16/04

4TM

10:269.40: 4) A variation on 269.34: We have Universal math set, $ s = strings of its symbols are ≈ "Lisp" pems
Grammar$_2$ G is a (perhaps) CFG whose leaves are strings of $ symbols i are: PEMS.
G_2 generates CFG's. By looking at a corpus of $ pems, we can devise G_2 grammars for them. We get a density d.f. on $ pems. I had t. idea some time ago that this could not
give a s. funct that was universal in t. interference. I want (& wanted) a universal df. on s. functs. T. idea being that I had a way of generating
S. functs, that was universal — so it would assign pc > 0 to any finitely describable s. function. ((Want a universal S-funct D.f. on S-functs =)
Well, 4 (.00ff) seems to do this. — NO, it doesn't! It is a (univsl d.f. S-funct) on D-functions.

07 → Any (d.f. S-funct (universal or not)) on S-functs is itself an S-funct.
Any d funct on S funct is an S-funct
19 Any S funct on d functs is an S-funct.
0

Are .07-.19 deceptive? I.e., a S funct D (i.e. P.d.) on a (D or S function f_1) is a function of t. form of f_1 & is indep of t. argument of f_1 — I'm not talking about a composition of functions.

3 In learning d-functions: I use data a/o (≈ PPM contexts) for induction: to create new conds. If any of t/ universal primaries a. are s. functs, then my old induction methods of data & contexts will give results that are sub-c functs & contexts that are s funct. — we will end up w/ sfuncts of (probly) arby complexity — proby all/possl S-functs could be generated, so method is univl.
·17 How to do t. such?
While .13—.17 seems reasonable, I can't think of any Examples of How to use it! Well: Could I use it for t. Born problem of 269.28? (If I knew any (such, perhaps) it would be clear! → 271.00 spec)
19

In APL & other similar langs there is an instruction that combines 2 1 dim arrays by + - x ÷ $\vec{a} = (a_1 \cdots a_n)$ $\vec{b} = (b_1 \cdots b_n)$: $\vec{a}\, Op\, \vec{b} = (a_1 Op b_1, \cdots a_n Op b_n)$
I may want to use this sort of thing rather to assign pc's to a list of "Tokens" or whatever.
(NB) There is a common (or just as in PPM or Augmented "PPM") Bart type ogey in which one defines a context & this automatically defines a p.d. on t. following Token. In this sense, t. dern of t. context is a complete dern of t. s. funct. A context also can be used to define a pd on things longer than a single symbol —
·6 A token can be an npm of any size — (or perhaps be a s-string (≡ P.d. on strings).)

27 — In ~ .13-.26 Consider a common ("Bern") method of induction: One does this solely by defining npms, npsts, functions, contexts. Each of these operations can t pc of corpus &
·29 — — Can use previously defined objects or "parts" of them in its new definitions.
? → ? An arbly complex OJ function can be thought of as a special case of .27-.29 ← [Later: How?] ? 35
0 Heuristics (for "suspecting" that a new conc. might be constructed in a class of ways)
are also constructed of old heurs & "parts" of them.
Initially, all of this can be done in PPM style, using BW (unwrights, harder) xpmn.
Later we look for sub-(hunts / graphs).
Re: .30: Perhaps this: That OJ operates on t. string of $Q_i(A_i)$ pairs in t. (sq) (≡ corpus).
35

8/18/04 221

4TM

-0.01 Spec 00:270.19 ! — → This is for Continuous D.F. : T. Discrete D.F. seems much different: I have to find good way to do it! — see (33)

In t. case of Lsrch: If I have a uniform d.f. over 0,1 as some of t. elements of t. dcon — I will start t. Lsrch w. 0 & 1 as possible trial conds. — brute? for each possib.

Next we try the new 2 bit code of ¼, ½, ¾. (I'm not sure how we should do it). If we ~~test ⅛,~~ use T ← 2T we do 0, ¼, ½, ¾ on next round — Then

.04 0 ⅛, ¼ . . . ⅞ on next round.
I'm not entirely at ease with .00 – .04 — There are other symbols in t. code! — but

.07 This is Lsrch, & we do try things in PC order.

Note that Koza did have a way of using GP to compute numerical params to hy precision.
— just how did he do it? . . . . I think there is a Mt Carlo version of (.00 – .04)

.09 If its not Lsrch, then we can just use a normal s-grammar (or a GP population based s-grammar)
10 to generate Mt Carlo conds — using something like Moj.

11 A possl serious difficulty w. GP: That one could often come to branches as one "climbs up t. landscape". Normally, GP makes decisions at each branch & continues. To some extent this is "fixed" (or made less bad) by doing many GP runs out.

Same problem — so alternative branchings must be chosen.
Lsrch does have advantage of looking at all branches (up to current CB level). — but its looking for a somewhat different thing.

There are "Multiple Goal" GA schemes; It may be that one can use them to investigate "Branching paths" better.

20 Ideally this (11) branching should be addressed by a "subgrammar" for each of t. branches: T. subgrammars would, of course, have many common concs, but would correspond to different Branches on a generating tree.
In a cross/mut system, we'd want parents to be (mainly) from t. same Branch, if we wanted to do a Branching exploration of t. "Fitness landscape". ← How one identifies "which Branch" a parent is on, is unclear! Would use of a PPM type "Grammar" be able to deal w. "Branching" of this type?

27 (.00 – .10) (Note .09 – .10 in particular) does seem like a general way to [illegible] search s-functs.
28 I'd like to link it to t. Bernoulli ideas of 269.34 & 270.27–29)
In (.00 – .10) we choose pc's values & try them out on t. corpus. In 270.27–29 we chose elements to do Laps rule on, & we compute t. pc's. We should get about t. same pc values.
30 .. So first, I want to be sure I know how to use .00 – .07 for Lsrch; then generalize so I can do (.09 – .10); Then see if I can fit in (.22 – .30).

33 : (-.01) In line (-.01) I noticed that t. Discrete D.F. seemed different from Continuous: i.e. Discrete: p_i (i = 1/h) p_i's pc of token$_i$ at a certain context. (I perhaps uniform d.f. modified by function)
Continuous is uniform D.F. over 0,1 mapped by some function onto a different range so it gets a different (usually non-uniform) d.f. In t. first case t. D version is a single token at that context. In t. other it is a single real (or complex) no. at a context.

In both cases there are ways to find "best fit" for params that define t. D.F.
In t. Discrete Case, Bernoulli (Laps rule) & in t. Continuous case, maybe similar trick: (say divide 0,1 into a few divisions — then use finer divisions (more bits) if needed / warranted. (Spec 272.00)

D1397 TM: CIAP:

This has to do w. TM being goal oriented: i.e. its always trying to solve an OZ problem! That its "Top Goal" is an OZ problem.

This seems to give trouble because the machine could re-wire itself to get ~~closer~~ closer to the goal (yet not "really" closer): Its like stimulating one's reinforcement center (if any!) electrically &/or chemically.

To prevent this behavior in TM. T. machine's Behavior would not be entirely defined by a goal. — part of it would be a non-goal pgm (that could still have feedback in it).

I suspect humans & animals are not entirely definable as goal oriented — but how to get some effect in a very smart machine, is unclear.

Note: Goal oriented behavior can also be equivalently denoted as a specific pgm in which Goal orientation is not specifically introduced. So "pgms" — is a more general kind of behavior than "Goal Orientation".

Perhaps the more General problem is simply getting a very smart Machine to do what we ~~want it~~ "want" it to do: — Even w.o. the added difficulty of "The Monkey's Paw Problem =(MPP)" (or the "3 wishes" problem (or the Nwishes problem)

R1897 TM

01: 217.40 spec (updating t. p.d. is an impt. part.)

217.33ff does tell how a TM could work.

For t. Machine of 217.20 to work, it has to have a way to emulate any [legitimate] trng. technique that is describable. Lsrch seems (maybe almost) able to do that.

Perhaps all learning has to have 2 parts: ① Lsrch ② "updating" (or otherwise improving) t. P.d.

.06 It may be possl. to have a single, Unified Gore, like U's "Learn How to Learn."

.07 → Q: Is a divisn into ① Lsrch wrt a P.D. ② Improving t. P.D. ("comparing t. P.D.") an adequate way to deal w. all the environments?

So: Think about 217.01–.19 (t. first Breakthru)

" " 217.20ff? (t. second breakthru).

219.06 (unified Gore)

219.07 Is a "2 part" TM adequate? Can I show how it can solve all problems that I know how to solve? — It seems to be adequate: I spend my time either [solving problems using solutions & parts of solutions of previous problem.] or [improving my problem solving methods] "Improving" t. P.D. is part of this last, but is it t. only way that one can improve prob. solving methods?

Keeping Mind, what t. Main Problems of t. Moment are!