1.25.79 Rev (Levin):

This will review recent results in ~ Levin 1979; ~20.01 to 37.01; Especially $\boxed{31.25 - 37.01}$
Try to refer to exact region so that an expanded explan will be available if necy.

1.34

This is a plan for a prelimy. T.M. – A dcrn. of t. trng. seq. & just how t. Trng. seq. is solvd. It includes some discn of TM2. In particular, t. search routine is treated exactly.

T. corpus itself is an ordrd sequence of induction problems. Each problem is a seq. of symbols (≡ "sub corpus") T. problem in each case is to find a short code (a/o many short codes or many longer codes) for t. seq. or an "honest" way to assign a hy aprip to that sub-corpus. (an "honest" proby assignment is one that can be backd up by codes).

.15 This gencl. problem is nowr "adequate": in t. sense that a soln. to it would be readily xfmd into a soln. to all other induction probs — & ∴ probly just about all scientific problems, including non-induction probly like NP probs. (See 90.33)

How t. problem is solved: For each problem (≡ sub corpus), we want to find an algorithm (algm) that takes t. entire sub corpus (≡ s.c.) as input (this is not a sequencial input, t. machine knows when t. sub corpus ends), then presents as output, a set of induction codes a/o an honest aprip to t. sub-corpus. If M is an Umc, then $M(q, \text{s.c.}) = s_1, s_2, s_3 \ldots$; q is t. dcrn. of t. algm, s.c. is t. finite sub corpus of interest, $s_1, s_2, s_3 \ldots$ are various short induction codes.

Def

T. algm $M(q, \cdot)$ looks at t. entire s.c. & tries to find short codes for it.

Examples of algms. that do this are: linear (or any other) regn., clustering, etc. (see 33.20 for brief list of examples).

In t. case of linear regn., t. algm. makes a correln. matrix (which looks at t. entire s.c.) which it uses to obtain t. linear coifs & σ2 for Gaussian predictive coding.

Sometimes there will be ambiguity as to how to divide a code for t. s. corpus into a "q" (≡ algm. dcrn) & t. rest of t. s. corpus dcrn. e.g. in linear regn: a code (set) could be (L.R.), 3, .11, 1.2, -1.5, 2.5, seq. of Gaussianly coded errors.

No. of coifs. 62

Name of algm (Lin. Reg.)

t. 3 coifs

pc's of concepts depend on pc. of corpus as coded by those concepts.

"L.R., 3" (no. of coifs ≡ m) would give a single aprip to t. s.c. and a set of codes clustering about t. shortest code.

"L.R." alone, would give a seq. of aprips: one for each value of m — these would be summd to get t. total apripd.

Usually, we will chose t. shorter algm. dcrn, because this usually ↓ total search time (see 35.30 – 37.01; 35.01 – 29 is also helpful) (hvr. note 98.01).

1.25.29 **Rev** (Levin) A uio umc is used (≡ Process): so that turn about $\Sigma cc < \frac{pc_i}{pc_i}$ is true

Search Technique: The Algms are tried in order of $\approx \frac{2^{-q_i}}{T_i} \approx \frac{pc_i}{cc_i}$

2^{-q_i} is t. aprip of t. i^{th} Algm.

T_i is t. cost of creating t. i^{th} algm. & using to obtain t. pc of t. sub-corpus wrt it.

T. search is done by a "partial" (as in "partial recursive functions") listing of all algms in order of their pc_i's. ($\approx 2^{-q_i}$ if t. shortest derv. is much better than other derns). This listing must be "partial" since certain derns will not derb. actual predn algms — i.e. they will not take finite time to compute t. pc of t. sub-corpus. [hot that t. set of induction algms (≡ cpms's) is not recursively enumerable]

If one has total cc available of T_a one discards a algm

.10 as soon as its $\frac{2^{-q_i}}{T_i}$ becomes $< \frac{1}{T_a}$, so only $\boxed{\frac{2^{-q_i}}{T_i} > \frac{1}{T_a}}$ are accepted

If there is a v.g. algm. w. derv. q_0 & its Σ cc is T_0 for t. s.c.,

then this algm. will be derv'd in a search for which t. search parameter

$\frac{1}{T_a} < \frac{2^{-q_0}}{T_0}$ is used. .....

If search parameter T_a is used, $T_i < T_a 2^{-q_i}$ (.10) (for all trials used.)

The total search time is ΣT_i which is $< T_a \Sigma 2^{-q_i} < \boxed{T_a} \approx \boxed{T_0 \cdot 2^{q_0}} = \frac{T_0}{pc_0}$.

Since $\Sigma 2^{-q_i} = \Sigma$ pc's of dern. < 1. Also by Kraft inequality since t. q_i are derns of finite objects (including stop instructions) & ∴ form a pre fix set.

In cases of interest, I think pc0 will be large — say $> 2^{-10}$

so 1000 T_0 will not be a very long search time.

Since $T_0 \cdot 2^{q_0}$ is not ordinarily known, we will chose a certain T_a as a search parameter, & complete that search; then use $2T_a$ as a search param ,, ... ,, ,, ,, ,, $2^2 T_a$,, 4:55p

,, ,, ,, ,, etc. This doubling technique will at most make total

.30 search time $2 \times T_0 \cdot 2^{q_0}$; (See 29.01-.10, 29.20-.29 & 33.35-34.20 for discn. of shortcuts).

.32 **Construction of t. initial language used to derb. t. Algms:** This is done by listing a large no. of algms that have been very useful in predicting t. kinds of sub-corpi of interest. (See 33.20ff for a few examples of such algms). We then try to "factor" these algms into a set of concepts that can (most compactly, minimally) express all of the algms. Each of t. concepts derived is assigned a pc, in a way ∋ t. corpus of algms being used is expressed most compactly.

.36 **Updating old concept pc's, & assigning usages and pc's to newly created concepts:** This is done by considering t. entire set of sub-corpi & various codings of this entire corpus. In a code of an entire corpus, it is easy to assign optimum pc's to concepts. These pc's are t. observed probys of those concepts in t. various environments in which they occur.

Ideally, these p.c.'s should be updated after each new sub-corpus is worked. Practically, it may be better to update less freqly.

1.25.79 **Rev (Levin)**

95.06

Heuristics : how t. system finds them automately (see 32.15–.35; also 36.01–37.01) | cross-coupling 36.01 ≡ 37.01

In our corpus, we try to arrange so that the search thresholds for t. various sub-corpi are ≈ equivalent. This could be done by having all sc's same
.04 length & all search thresholds, T_a, t. same. Possibly one could have T_a & u.
.05 sc. length in a suitable manner if sc. lenth did vary. Another possy is to ~~vary~~ allow t. "equivt." T_a to vary over t. sc's, but slowly change.

The idea here is that t. pc's of concepts & of Algms., depends on how effective those abss. are in predn., subject to t. constraint that t. final use of t. Algm constructed must have cc < Threshold. This automatically gives hy pc to abss. that ~~help cr~~ create or help create algms. ~~other~~ hy that (a) give hy pc to sc's (b) do this w. cc that is < Threshold.

These (a),(b) conds. are for a first device that is good for predn. T. speed of implementation condition makes it a kind of "heuristic". That t. lengths of sc's & t. T_a's be t. same, is nacy, so that info on concepts obtained from different sc's will be comparable.

Part of t. heuristic concept. is t. TM_2 concept. Here t. machine (TM_1) is allowed to look at good codings of t. entire corpus (≡ all sub corpi) & tries to find regys in t. construction of algms that have been most useful. — ~~then~~ any such regys can be used to ↑ t. pc. of existing algms & their sub concepts. This operation is a TM_2 function, but it can be performed by TM_1 if those code is regarded as "just another sc. of TM_1". (A possl. diffty is that this code will not, ordinarly, be of a length comparable to t. other sub-corpi — in which case see .04–.05 for an idea) . . . .

.25 Another impt. idea is that of "learning" as t. search for a gn. sc progresses.
We try Algm$_1$ & it gets final sc. pc. of P_1 } & so on for Algm$_3$ ... From this info,
" " Algm$_2$ " " " " " " " P_2 } we should may be able to get ideas for new Algms that are more like to have very hy final sc pc. — I.e. we use t. degree of success of Algms$_{1,2,3}$ to modify t. aprip (≡ pc) of t. subsequent Algm. trials. (36.01–37.01 discusses this idea some; 36.25–37.01 in particular). This process is equivt. to a new Algm, with (presumably) hy ~~comparn~~ Δ total pc per Δ cc than others — so we give this new Algm. hy ~~upd~~ pc. (small |q| for its derv, q). However, this "learning" idea isn't yet too clear in my mind & I haven't properly formalized it for analysis, → see 78.01ff for better disc. of this. | also 71.01–.35 to some extent

.30 I conceive of this system as being part of PMTM. I suspect that many of t. concepts needed to develop statistical predictors like linear regn., will come from fields of "experience" that are not included in t. usual statistical sub-corpi — so ≈ PMTM is needed. T. ways in which these other modes of PMTM can be ~~worked~~ implemented using t. ~~present~~ formalism of this note, have not been worked out — also how info is transferd from one Mode to another.

See 95.06 for disc. of "Production" v.s. "search" heuristics.

1.25.79 REV (LEVIN)

Gen'l discn: T. ~~problem~~ form of t. prodn. problem used seems adequate (1.15)

The idea of t. sub-corpus not being nearly "sequential" is imp.

Th. search method using $\frac{\text{pc of Algm}}{\text{cc of Algms eval'n. of sc}}$ is v.g. & impt. (2.01–.30) —

— it makes it possl. to get good results w. short ~~[illegible]~~ total search times

That heuristics are automatically included in t. formalism is impt. (3.01–.40). — The

This part is a bit vague in my mind.

This system seems like it might take care of heuristics as well as prediction w.o. any special devices — looks like what I've been looking for for many yrs.!

Some major problems!

(1) Devising an ~~[illegible]~~ initial set of concepts that are complete (≡ univ'l.) so that all possl. Algms. can be partially listed (2.32) — also t. concepts should give good aprips to good Algms.

(2) The stuff on how heuristics are automatically implemented (3.01–.40) is not so clear & hasn't been adequately analysed or formalized. We definitely want to make sure that all possl. heuristics can be worked into t. system.

(3) How to implement PMTM using t. present formalism (3.30–.40 is a prelimy discn). I'm not sure an actual PMTM is needed — but many of t. sc's must be from fields superficially distant from t. ~~an~~ statistical prodn. sub-corpi.

(4) T. way(s) in which, say, linear regn. (in various of its forms) could be derrd., & t. tag seq. needed has not been gone into at all — This is related to (3)

These also tie in w. (1) somewhat. also (5)

(5) T. gen'l. problem of devising of Trg. sequs. for this TM. ((1), (3), (4) are relevant)

(6) T. problem of how T_2 (or some other search param) must vary as t. ~~[illegible]~~ s.c. length varies. (3.01–.05). Possibly it is not t. length of t. corpus but it final pc, (funct of (corpus length × redundance of corpus)) that should determine T_2? Perhaps t. way info on a gn. concept ~~[illegible]~~ derived from diffrnt. sc's, is combined ~~[illegible]~~ can use "wts." dependent on t. respective ~~[illegible]~~ s.c. lengths.

(7) The IPC problem of how to construct real machines that will compute soln's to these probs. w. minimum ~~[illegible]~~ cc. It is likely that a factor of ~ 1000 in speed up is possl. W. technologically doubling of cc/$ every 3 yrs, say, this factor of 1000 amounts to ~ 30 yrs. of hard wave improvement!

(8) In RW t. corpus is not broken down into s.c's of about equal length. Just how can this system deal w. RW data? → (79.01–.15 for ideas on TM constructing its own Trg. sequs. ~~[illegible]~~ from RW data).

76.01-.28 review also | 53.01 - 54.40 ← (is on a pure induction machine A sort of review of stuff up to that pt.) | 75.01-.76 reviews much of 56.02 - 74.40

2.20.79 Rev (Lev) | 37.02 - 54.40 is main thing reviewed, then 71.01 - 79.40 ← 53.01 - 54.40 reviews 37.02 - 52.40, to some extent.

This will cover Lev. | 37.02 — 79.40; I'll start w. what I think is t. major development since

37.01: **N.B.** 46.01 - 47.40 discusses "Heuristics": Also impt. of using "historic" info on cc & pc of Algms that were actually used.

My present impressn. is that t. device of Rev. 1.01 - 9.40 may, indeed, be an adequate "study problem", in t. sense that, Very many problem types can be expressed as either "NP problems" or "Gray NP problems". [NP prob. is: $U(\alpha, x, \cdot)$ is given: U is a Unc. we have to find a y of length not much > |x| ∋ $U(\alpha, x, y) = 0$. α is t. dcrn. of a fixed Algm. x is some string of arby but finite length. t. time to compute $U(\alpha, x, y)$ is comparable to |x|. For a Gray NP problem, t. conds. are similar, but $U(\alpha, x, \cdot)$ is a real no. & we want to find a y that Max's it, or gives a largest value in t. cc available for t. search.] finding a good induction code for a corpus, is a "Gray NP prob." → Hvr., see 6.02

.16

.17 For $TM_1 = TM_2$, some of TM_1's problems will have to be to both (a) Get a good code for a corpus (b) Get a seq. of hy^ct proby extrapolns of t. corpus. Devising a algm. to do this wrt. an arby corpus is a Gray NP prob. (≡ GNP prob.). See 50.18 for a discn. of t. problem, & 50.37-.40, 52.01-.05 for what looks like an adequate soln. Hvr., probs like .17 - .20 are of t. NP & GNP type, so fine!

Def. .20

1) $U(q, \alpha, x) = y$.

~~Two kinds of uncertainty~~

.25 → **Note** that for t. NP problems defined by $U(\alpha_i, x_i, \cdot)$; we want to find a algm dcrn, q_i ∋ $\boxed{y = U(q_i, \alpha_i, x_i)}$ is a soln. I think this is a better way than to try to find $y = U(q_i, x_i)$, because in t. $U(q, \alpha, x)$ problem, we want to find a q that is relatively constant for all α's; we want a gen'l. problem solving technique that can look at both α & x & then try to find a soln. The q_i's will then not change much with i (by impt. — reduces search time since q_{i+1} will be a "short distance" from q_i ∴ t. "Δq" needed will be small. say < 10 bits. If it is this 10 bits that occurs as 2^{10} (Tcomputing + Testing))

.30 **DEF** Some impt. kinds of probs. that are (G)NP: (where (G)NP = NP or GNP)

(1) H.ll climbing problems — either direct search for optimum (history of 1) or History of a few or history of h >> 1 → [see 78.02 - .12 for discns] ← so this includes sequencial optzn. probs.... t. hardest type of Scientific problem that I've thot about much. — in seq. optzn., t. soln. is an algm. for making trials.. t. algm basing t. trials on past trials' success & is able to do expts., because of t. "averaging" structure of t. Gore. Finding a soln. to a (set of) eq(s) could be regarded as a GNP prob. Each closer approxn. is hyer G. Usual H.C. methods need not be used — ordinary equ. soln. methods can be used.

(2) Finding a good Grammar for a corpus. [Note: search for such a grammar will, if done, well, involve noting what trials in t. past for this seq' have worked badly & well, & why they worked badly or well.... so it could be a more advanced type of search such as discussed in 78.02 - .12.

(3) Finding solns. to ordinary NP probs, like (a) find y ∋ $Axy = x$; solve linear & non-linear eqns in 1 or more unknowns; perhaps probs. in "Number Theory", etc. An entire very interesting tng seq. could be constructed from NP probs along.

2.21.79 Review (Lev)

See Lev 53.01 – 54.40 (this is a pure induction TM): It is a kind of Review of t. TM contemplated.

.02:5.16 **SN**: I'm not sure that GNP probs are so "naturally" world: Tho it would seem that if one considers t. problem to be that of selecting an optimization alg., & "q" is t. alg. dcrn., & T(q) is t. time needed to "test" alg q, then $2^{q_0}T(q)$ sec's are needed to find t. alg, q_0. In t. case of $U(q;)$ being a optzn. alg., testing times for each trial, can be rather large. Also, t. test result is not W.B., but Grey. Finding induction codes is a Gry prob. of this type.

19 ~~4~~
157
37

See top of P5 of this review for list of partial reviews of t. material of interest: Read these reviews.

Then, try to characterize my "New Approach" as exactly as possl.

(Brief) Impt. / New Characterizns. of Proposd Approach:

1) Break up of corpus into sub-corpi (ES.C's): this is so that:
a) Coding of t. corpus as a whole is much more difficult than t. sum of coding t. parts (ε sc's). b) One obtains partial / F.B. faster

2) The ordering of t. sc's into a "Tug." say. This is so reqys observd in earlier parts can be usd to help code later parts.

[w.o. both 1) & 2) coding of large corpi is usually "Transcomputable". 1) is possl. w.o. 2), but is not v.s. w.o. 2): 2) is not possl. w.o. 1), hvr.]

3) In coding a sc, we do not try for t. most genl. type of code, but rather one of t. form: [code of a cpm] ^ [probty (codes) of sc wrt. that cpm]. Here, t. codes of t. cpm's can be relatively short (say ~ 10 bits), even tho t. probty of sc wrt that cpm is very small. We can code t. cpm by using a single string. (Chaitin showed that Σ pc of such finite objects is within constant factor of $2^{-K\,cost}$) ... T. proby of t. corpus w.r.t. t. cpm can be computed directly by t. cpm (or occasionally by summation, $\Sigma\, 2^{-K\,cost}$).

(w.r.t. t. codes used for t. previous sub corpi.)

See Lev 98.01 for serious diffys w. this idea!

S.c.'s can be either induction problems, NP problems or a more genl. type of optzn. problems — providing they are solvable by a short code (10 bits ... or perhaps up to 20 bits — depending on amt. of time needed to generate t. cpm (or whatever) + t. time to check or evaluate t. cpm (or whatever)). [Total expected cc of search is $\lesssim 2^q$ (cc of generating & checking an adequate soln.) Hvr., note diffys w., say, linear Regressn.]

4) $\frac{cc}{pc}$ search: This, in general, may not be an optimum search strategy but it may be (adequate / acceptable). Later (see (5) ≡ 7.0) we can devise better search strategies. (72.01–.40 on non-optimality of $\frac{cc}{pc}$ search, summarized by 75.01.24)

2.25.78 Rev (Lev)

5) An understanding of how $TM_1 = TM_2$ is to be implemented;
Just how it is an elzn. of t. "total problem"; & what t. "total problem" is.
How $TM_1 = TM_2$ may be improved — & just what it is "an approxn." to.
Criticism of simple $\frac{cc}{pc}$ search. Suggs. for better forms of search
that can be practically implemented — form of what
may be t. best possl. search strategy. [71.01–.40, 79.16–.20; 77.01 – 79.15
76.01–.28]

.12 (SN) One thing that I am not at this moment clear about: T. details of just what TM
does its $\frac{cc}{pc}$ search on. Does it try to find (a) an Algm$_i$ (x_i) whose output is a code of logit probty of x_i
or (b) An Algm$_i$ ⇒ Algm$_i$ (x_j) has proper output for all [redacted] $j \leq i$. (seq to be coded; seq to be coded)

.15 → See 8.01–.40 for what looks like an adequate model — "it's" (b)

I think I [redacted] analyzed this to some extent within t. last wk or 2.
5.25 (of Rev.) discusses this a [redacted] little. 3.2579: t. soln. of (b) is that t. latest Algm$_j$ (is not necessarily changed for each Seq_i) is our best bet for a TM_1 at Time j: Algm$_j$ need not work well for t. beginning of t. corpus, since
→ See 90.33 for what's probly an adequate approach to NP (& w) probs

(c) For NP problems: say $\alpha(x, y) = 0$ is t. problem: Gn. α, x to find y:
we want Algm$_i$ (α_i, x_i) that generates a good trial y_i in short time or
if generates a soln y_i in short time. say Algm$_i$ = U(p_i, ·, ·). p_i will have a short
dcrn. w.r.t. t. set of previously acceptable p_j's. Again we want Algm$_i$ ⇒ Algm$_i$ (α_j, x_j)
solves all probs for $j \leq i$. Testing these trial p_i's can take much time, &
various sampling methods could be used.

Algm$_i$ is a search Algm & t. solns to early probs provably speed bound. Algm$_i$ is more best bet for a good search Algm for t. remaining (or near) future of our corpus.

in .15 (b) we also have a similar cond., but by suitable sampling, we can
[redacted] discard a trial Algm$_i$ if t. pc's of t. Seq_j's tested thus far give a
total bcost > [redacted] acceptability threshold (sort of mindfull of α, β heuristic, in
that one saves more time w. this trick if one happends to get good guesses early)

What I can do, is list t. various possl. kinds of probs., & typl seqs,
& methods of treatment, — so I can [redacted] criticize them &
decide on t. best ones.

Hvr., try reading what I've written on these probs. ... I think I've [redacted] forgotn. a lot.

.30 Bibl. on t. prob. of .12: 43.28 – 45.40 (43.28–44.27 is on N.C., but relevant.); 48.01 – 50.40 (somewhat "talky"); 52.01–.40; 53.01 – 54.40 (is on pure induction machine)

71.01 – .40 → 76.01 – 79.40 (on sub-optimality of t. Trg. Seq. method.); (72.01 – .40 → 75.01 – .24 on sub-optimality of $\frac{cc}{pc}$ search)
76.01 – .40 summarizes both of these.

77.14 – .40 Discusses & criticizes various possl. forms of TM ← (pure induction TM only).

Note: t. machine of 77.14 (also 53.01–54.40) is a purely induction machine: It doesn't do N.P. probs.: Superficially, t. improvement & genzns of 78.01 – 79.40 (onto t. machine of 77.14) are all toward pure induction machines. Is this nearly true? — Can I make good genzns to work N.P. probs?

Note: 90.33 ff really deals w. t. N.P. probs & adequately solving induction probs. is all I need to work on.

Aux. Q: would t. pure induction machine of 1.01 – 4.40 (Rev.); 53.01 – 54.40 work?

Narrowly Relevant to t. Q of .12] 5.25 – 30 (Rev.); 44.28 –

(I think there is a continn. of this discussion a "P8" — [redacted] — Levin 88.01 ff may be that continn, hvr. (looks very likely).

3.25.79 Rev (Lev):

Discn. of how t. picture has changed since Rev 1.01–7.40: (t. forgg. is almost ≡ 94.01–95.40 with exceptions ~~denoted~~ noted on (94.20–.40)R). ≐ (94.10–.12)R

(0) See: 97.01 –.40 for another brief dcrn. of t. desired system.

(1) T. ~~PAAM~~ corpus is as before, divided up into Sc_i's, sequentially.

.03 (2). ~~[illegible]~~ Around 1.15 ~~[illegible]~~ I had a different idea as to what

.04 $M(q, Sc_i)$ was: Now, I use Alg_{mj} ([redacted] Sc_i, (b)) ≡ U (q_i, Sc_i, (b)) → $P_1, P_2 \ldots$ for Sc_i w/o probty distribn.

(b) can contain — see 96.04, .19–.20, .25–.28 for what } also see .25, below (≡ (6))

umc — pgm. to simulate Alg_{mj} — set of ~~output~~ codes

Alg_{mj} is t. particular form of TM_1 at "time" j.

.10 (3) Sc_i can be either a shorter long seq. We do not confine ourselves to codes of t. form (dcrn of cpm_k)(dcrn of Sc_i wrt cpM_k). See 98.01

(4) Alg_{mj} can use any method of search. At first (small j) pure ~~sc~~ $\frac{pc}{cc}$ search will be used. Later, as Alg_{mj} is improved (by TM_2), we will use more fanc. types of search using (using (b) data only (≡ apri. to Sc_i)) ~~[illegible]~~ obs (≡ observations) on Sc_i to direct t. search (we will use)

(5) Initially, I will be TM_2 & design various Alg_{mj}'s. & improve them. Since Alg_{mj} can ~~[illegible]~~ guide t. search by looking at t. whole Sc_i, it can bias t. search — so I want to be careful of this when I am TM_2 — One way to avoid bias is to avoid using Sc_i observations. [98.20 –.30], here, later, when $TM_2 = TM_1$, there need not be any bias, since TM_3's goal is to get good predns., & any bias would be ~~[redacted]~~ against this goal.

.25 (6) In 2.36 we considered updating pc's of old concepts, naming new concepts & assigning pc's to them. This may all be automatically taken care of by t. avgt. (b). (b) includes any & all ~~[illegible]~~ info that is apri to Sc_i, so it can be used to help code Sc_i.

(b) includes strtns, defns & concepts & their pc's. It includes those for parallel codes of t. previous Sc_i's — ~~[illegible]~~ These ∥ codes are useful & contain various ~~[illegible]~~ hy $\frac{pc}{cc}$ abss. Anything assoc. w. t. previous Sc_i's is Logit. to put into (b) (which is ≈ a library). But how to optimumly use t. info in (b) is a Q. ! We have to include various IR tricks to use ~~[illegible]~~ (b).

(b) also includes t. raw uncoded previous corpus.

(7) We will concern ourselves w. induction probs. only. NP probs. are a subclass of induction probs (as are all problems) & can be included in t. trg seq. in this form. [90.33 – 91.13 (91.13)].

(8) Bibiog. on improvements needed: 95.01 [Actually, by defining (b) properly: (see 8.04) ~~[illegible]~~ this Alg_j becomes a potentially very non-al. solr. & overturns as many objections!]

(9) Discussn. of heuristics & how they are based on finite CB: 95.06 –.25 (also see 9.10 (Rev.) on heurs)

(10) T. forgg is ≈ ≡ ⟨94.01–95.40⟩; ~~[redacted]~~ exceptions are noted on (94.10–.12 (R), 94.20–.40 (R))

(11) All of t. sub-problems of optimizing Alg_j [such as, optimizing TM_2 or finding good set of obs on Sc_i or finding best way to make obs in RW, etc.] are regular "induction probs" & can be treated by t. usual methods.

(12) T: Main Methodological Idea: If there is any problem$_1$ (→ Induction or Non-induction), write up t. human soln a human would find: derb. t. concepts, strtns, defns, needed for that soln.

• Next problem: Devise a new problem$_2$ ∋ these abss get large enuf $\frac{pc}{cc}$ so that they are within our available cc.

Next prob. devise new prob., which ends up w. abss. that give t. abss. needed in Problem$_2$ hy enuf $\frac{pc}{cc}$, etc.

Essentially a "working backwards approach. This is alternative to trying to devise a good set of primitive abss. & working up thru a suitable tng. seq. to get to diff. probs. It is possl. to use both approaches, hvr.

.10 (13) Off theoretical interest & very impt: On Heuristics: 95.06-.25

32.15-.35; 36.01-37.01; 78.01 ff; 71.01-.35 Many more refs.;

.17 but an impt. prob is that of 40.25-.30; 46.01-47.40. T. prob. is that various abss. have obtnd various pc's wrt. various sc_i's using various CB's (CB ≡ $T_0 < 29\ T_q$).

— So how should one assign pc's to abss. used w. a particular CB — which can be diferent from t. CB's that these pc's were obtnd with?

40.25-.30 was a "ruff & dirty soln." for a certain kind of TM: I'm not sure how relevant it is to t. present T.M.

One not bad approach was to express t. pc of each abs as a funct. of CB, so we can directly assign a pc for each possl. CB lvel. While pc of a gn. abs. may not be monotonic in CB, (so this funct. can have a complicated form), — due to pc's of competing abs — it may be that t. unnormal pc's of various competing abss. can be approxd by monotonic functs of CB.