Boring to GB

1) Trap
2) Garbage
3) Lens

1) Fix rigda hole
– first examine from inside
2) Put ladder under house.

Bottom

Genl.

IPC

Hair Man

Punctuation [redacted]

FOR's should → willis.

[redacted]

Pol [redacted]

PSOD [redacted]

Bibliog

[redacted]

[redacted] Top

topics in this sub-folder.

Bottom & top inverted in this Xerox copy

The this folder contains impt. papers! I haven't been able to find the original! (8 12 77) well finally found these in 2 other sub folders in TM 75A

One sub folder on TM Gen has

This

Aug 23, 73

Genl.

304

1) On Why Questions: Their use in TM's:

CM I makes "Why" a fairly specific Q. w. a perhaps useful answer: E.g. One has a method for working a rather large class of problems. One tries the method on a new prob. It doesn't work, (or works poorly w.r.t some Genl). One asks "Why?". The answer via CM I is: How is this result made most likely by info in the previous corpus? Several parallel paems can be gen. w. max Genl. From these paems & the structures of these paems, one may then get clues on how to modify the method to obtain one that works in the new case.

.03 processed & automatic combin. of Paems

Essentially, this is meant to model after human processes — in that humans often use the Q "why" as a heuristic device.

Asking "Why the method didn't work in this case" is a more genl. Q than asking "How can I best solve this new problem" — in that the "Why" Q is usually of more general applicability in future problems.

"Why" in the .03 situation could result in a more stringent selection criterion for the "method" used. A selection crit. is a part of a method that tells when it should be used &/or looks at the problem & sets up certain params in the method so it can be used.

306.01
306.18
more specific

.01 ; 304.40 General approach to A I devices:

At all times, retain large store of Poms around that have been useful. Srtns & "components" & "mutations" of these Poms should also be retained & indexed so one will be able to get at parts that are likely to be useful in forming new trial "devices", or "demons".

The storage will be hierarchical so that access time (~ cost) will be related to & expected utility of each item stored. It may be more practical to store parts of an item, w. construction instructions, in rapid access, than to store the whole item in slow access.

.18 : (304.40 spacly) : After the answer to why the Method didn't work, one has many operators that one has accumulated, that will have suggestions on what to do. The "why" question "factors" the non-working of the method — it breaks it down into useful "parts" so that various "difference operators" can look at it — see how the method differs from what is desired, & suggest corrections to modify it to make it work more like what is desired.

Note that here we are working on the "Engineering problem" of how to bring about certain effects — in this case, how to get a "Method" (that is usually mutations of old methods), that will "work" the desired problem.

.32 Also on Method of Use of CMI in TM's; From 339-340 (on Punctn.)

& from other disins! It would seem that one could devise a set of methods of generating Poms, & one ended up with A=0 & $ is small. (e.g. — also, the Max Method.). However, it might be that one could save on C cost by looking at the corpus a bit before generating & testing Poms — so A≠0 & $ is larger than if A=0; but one may gain in C cost. So one has a trade off betw. C cost & A=0 or A≠0, & one must learn how to choose betw. Pom generative methods to get min $ for a gen. C cost.

Gen'l Approaches to TM devices.

.01 : 306.40 : (Perhaps!) New idea: Say TM has been working for a long time, it has preserved a history of its own operation — in some abstracted form.

Now, we present TM w. a new problem. Just how does it use all of its previous experience to try to obtain a soln. to the new problem?

We can, via CMI or approxns to CMI (like SVH) write out a "soln." to the new problem. But such solns will usually be too lg in cost. The true problem for TM is what to do about the new problem — → a greatest possl. utility per cost expenditure will be obtaind.

Σ Note that the Q. of how to take into account the cost necy for working on this "admin." problem, is unclear — This Q came up in SP in a ~ way. I don't remember just what I did about it. — One way mite be to have fixd srtns to solve those problems — with TM_2 continuously working on the problem of devising such srtns. Untill TM_2 becomes adequately good, I could be . TM_2 when TM_1 is good enuf, we let $TM_1 = TM_2$ — i.e. TM_2 is one of TM_1's many jobs types (say TM_1 is a PMTM).

It looks at "~" problems in the past & from the approaches that were used, & their successes & failures, it decides on an approach for the present problem. Note that TM need not use the "same" approach as was used in a ~ prob. in the past. The "similarity" may be fairly distant, so a rather complex xfmn. may be necy. to change the old approach into something appropriate to the new problem. This "similarity" problem is similar if not identical to that of extrapolation by analogy (L. Roberts' Pat Winston).

→ T. forms (343.01 ff) is about the same as Sussman's "Hacker" (Pol 328.36) Sussman Also includes my earlier idea about going to the library first! I think I wrote a fairly large amount on this: Perhaps mainly in SP. Try to find previous work on this. Also PMTM, also BSTM.

Every time TM solves a new problem, it ① betters its statistics in the usualt way by ↑ SSZ ② If it solves the problem in a "creative" way at all, it will have a new powerful tool to

Gen'l. Approaches to TM.

01:39340 use on new problems.

→ Essential Q's in this or any purported "Gen'l. Approach":

① Is it Univl.: in the sense that by giving a suitable trg. seq., we can get it to learn any derbabble trick?

② Is the SSZ required for ① reasonable? (Note that SSZ is / available IPC (≈ cost) are reciprocally related).

Some new (& remembrances of old) ideas in this area:

Say a PMTM is used w. a SP administrator — using a library of obs & ops. At a particular state of experience, the obs, ops, the various known correlns, & the particular statistical params. at that time can be used to solve a particular set of probs. w. some facility.

However, New Ops & Obs are created by CM1 methods. They are evaluated tentatively evaluated using a small sample of recent problems. If they seem to be useful, they are evaluated using a larger SSZ from the past.

The end result of these new creations, is to throw out some old, little used, low prost ~~concepts~~ obs & ops & store the new ones.

Genl. (remarks)

01: 344.40: If we have a deterministic corpus: Say 11111.... = 1^∞.

Then using Laplace's rule, we get $p_{n+1} = \frac{1}{1+n}$ for the probty that the $n+1$'th symbol will be 1.

However, from CMI - for deterministic seqs. $\prod_{i=1}^{\infty} p_i$ = prodt of dcrn of the seq. which is >0.

$$\prod \left(\frac{1}{1+n}\right) = \prod_{i=1}^{\infty}\left(1-\frac{1}{1+n}\right)$$ which $\to 0$,

since $\sum_{i=1}^{\infty} \frac{1}{1+n}$ diverges! T. moral is — for this seq.

p_{n+1} must converge to 1 more rapidly than $\left(1-\frac{1}{1+n}\right)$

i.e. $\sum_{i=1}^{\infty}(1-p_{n+1}) < \infty$. So Lap's rule would be wrong in this case!

It may well be that for all deterministic seqs Lap's rule is wrong. I.e. CMI might always give an essentially better approx to deterministic seqs.

I think Carnap got variations on Laps rule, but his expressions were always like $\frac{\alpha}{\alpha+n}$ whereas $\sum_{n=1}^{\infty}\left(1-\frac{\alpha}{\alpha+n}\right)$ also diverges — So Carnap is also wrong for the deterministic case.

⇒ Note Also that if we have a Bern seq. w. $p=\frac{1}{2}$ or $\frac{3}{4}$ or any "non-random" no., then $\sum_{i=1}^{\infty}(p_i - p_i')^2$ converges, since $\prod_{i=1}^{\infty}\left(\frac{p_i'}{p_i}\right)^2 > 0$. which it does not do for the ordinary Laps rule. i.e. $\frac{m+1}{m+n+1} - \frac{m}{m+n}$

$$= \frac{(m+1)(m+n) - m(m+n+1)}{(m+n)(m+n+1)} = \frac{m(m+n) + (m+n) - m(m+n) - m}{(\quad)(\quad)} = \frac{n}{(m+n)(m+n+1)}$$

$$= \frac{p}{m+n+1} \quad (p = \text{const}) \quad \text{whereas} \prod_{(m+n)_1}^{\infty}\left(1-\frac{p}{(m+n)+1}\right) = 0$$

since $\sum \frac{p}{(m+n)+1}$ diverges

Laps rule is obtained if one assumes uniform apriori for p; then uses expected value of apsip. CMI assumes non-uniform apriori for p. However, I think the difference between the 2 methods may be stronger than that — since I suspect small differences in apriori would not give convergence in 1 case & divergence in the other! } But this must be checked.

356.01

Gen.

01! 351.40 : A New type of PEM : (= Computable proby measure).

W. defines Comp Proby Measures. — These are defind for all corpi.

Would it not be possl. to define a Partially computable proby measure, that is defined for some, but not all, corpi.

In this case, one inserts t. corpus in Q., ~~[illegible]~~ into a machine, & th. output is either ① t. binary expansion of t. oprip of that say. — ② ~~[illegible]~~ some computation then a "stop" signal w.o. other output ③ t. machine computes forever w.o. output.

If only ① & ② are true, t. measure is probly much more tractable than if ③ is possl.

Gen - [redacted]:

(Blum. 372.40): (356.40 Gen) A new tack! Actually, I'm not so terribly interested in t. CMI version of Blum's prob. I'm at present willing to consider, say, all pfns w. [redacted] CB linear in s t. corpus length. (or s^2 or whatever).

T. impl. Q is — how to get something like t. most prosp per expended cost.

One guess: spend a cost of $\sim 2^{-N}$ on codes of length N.

→ This is really poor: To see what happens: (& this is t. impt. pt.) Make a tree of codes that have tracked t. corpus as far as t. branch went. At each branch end we write how much cost has been expended thus far w.o. convergence (≡ output symbol or stop (≡ request for more input>).

What happens is that we get branch ends w. very long input segs. that do converge — but there are an enormous no. of such segs that must be tried — & many do converge. What many of such input segs. corresp to, are pfns of various kinds. They are "setting up" pfns like "linear regressn" or "PSG induction" or "clustering" etc. Almost all of these pfns will not work at all well when they start giving output (i.e. they will have relatively little output increase per ↑ of input — (e.g. linear regressn systems w. t. wrong coefs). — So one wastes an enormous amt. of cost on these output less / input segs.

Also, there are probably even more input segs that have no output ever, but do converge in this sense.

As a result of .10 ff, I abandoned "phase sequenced / property machines" approach to CMI realization (i.e. 124.02 −.09 | 118.15 − 119.25)

→ .10 ff is, I think, a more exact statement of why that approach is bad, that my previous discn of it.

375.01

Gen
ok: 379.40 : Anyway 379.60 ff may suggest more reasonable ways to search for good codes — or simply better ways to do CMI.

One much better way, is to make various measure observations on Y. corpus. These observations will then tell one which pgms, are likely to work well.

Some simple observations:

1) Freqs of symbols: Ngms. From this, Y. effectiveness of Baru coding can be determined — also use of defns of various Ngms.

2) Auto correln (for a corpus that is an analog time series). From this one can determine which linear regression coefs to use & how much "coding efficy" one gets.

3) Auto & cross correln. of various functions of Y. time series of 2) — to do same thing as 2) but use of non-linear terms, power series (multidim Taylor) or exponential terms, Fourier terms, etc.

4) Concordance construction for est. estimate how good CFG coding would be. See PSGD 346.01 ff

5) For certain kinds of Data, make cluster diagram maps & find cluster boundaries, etc. One can then tell how much auto compression one can get this way.

ect.

So: Assoc. many sets of pgms, are sets of measurements to make on a corpus to determine which of that set of pgms will code that corpus well.

It is clear that one can define the set of all possible pgms as Willis does — Also, one can define Y. set

Gen
.1: 37540 of all possl. measurements on a corpus (This is the set of all functions from the corpus in to the set of all integers — one can have pairs or n-tuples of functs for vector functs of the corpus)

But anyway, it is clear that at the present time, I don't have a good formalization of the search problem for induction rules. — Certainly not in a form to present it to an audience of mathns. interested in min computational complexity algorithms.

As for this formalization, I'd perhaps best look at what humans seem to do in solving a thg seq.

However, the General Problem of the lowest cost method to
.17 do ICMI does remain.

.18 What seems to be done by Humans: we start out with an ongoing induction process, in which we have many poms that we have used on the "past corpus". By factoring the poms into named srtns a/o combination rule. We thus extrapolate the set of old poms to devise new poms. This can perhaps best be done by regarding the old good poms as ass in some "thg" — perhaps a ~~thg~~ Cf Gramm as good env'l — The other types of rules might have gramms that are easier to devr. Anyway, it might be nego to devise special grammars that induce new hdrs more like the way humans do.

So, a way to construct a TM: Start w. a good bunch of poms, that have been usd w. a hypothetical "previous corpus". Use Grammatical induction to extrapolate this set of poms. Use the new extrapolated poms on the new sub corpus. This new sub corp will reenforce some poms & not others. Keep the good poms in one's

"Sample set of f. lang" à extrapolate some new poems à try on a new sub-corpus — etc.

The forgg will have to be formalized some, — but f. Q of how to optimize it, isn't so clear. One looks at some hyper level (TM_2) devise better grammar types ~~[illegible]~~ to extrapolate f. poems.

So: Present concept: TM_2 generates new poems in perhaps a MICAC way (≈zero boost). These are tested (or f. entire set of poems is tested) w.r.t f. continuing corpus. T. selection of poems w.r.t f. corpus is used to ① select new à discard old poems from f. "retained set" of poems ② Vary f. params of f. poem generating "Grammar" (or other device) of TM_2.

We write usual to consider f. "set of poems" as a single POEM. [Most of those poems can be "given" - ie. ≈ zero boost.]

T. ~~[illegible]~~ approach to actually designing such a device: Use a fairly-well-understood Type Seq., w. a known-to-be-adequate set of poems. — So one knows which poems should be added in what order under pressure from f. "driving" Type Seq.

— So a mechanism that is adequate to do this could be looked for.

So, TM_2's problem is to find a good seq. of POEM's — which ~~[illegible]~~ does by "mutation" trials — (perhaps more intelligent since subsequent trials can be guided by knowledge of ° of goodness of various previous trials).

So, in a sense, TM_2's problem is to guess a good seq. of poem's — while TM_1's prob. is to guess a seq. of simpler symbols (i.e. f. symbols of f. corpus)

So, given a type seq. that would be reasonable for humans, we can assoc. with it a seq. of poems that a human could learn. From such a seq. of poems, we must then design a large, [illegible]

Gen
1: 377.10 such that the differences between successive pairs of the seq. are all "small", in the sense of likely enough so that trial guesses would take too long. (i.e. expected ~~Guess~~ time needed to guess the next pair is within a reasonable cost.)

06 Actually, the forgg. TM_2 could just as well try to find (by ~ mutations) an/ I.O. device (= operator). This could be; ~ an **RTM**. The operator could be a gen'l. F.B. device in the sense of ~~the~~ obs have ops in them & ops have obs in them.

In fact this .06 device would be a fine simulation of an animal, because at any particular time, it has a relatively fast response to its input (if that is what it was reenfd for).

The technique of Design of this TM: Make a Turing seq. than devise a seq. of operators or pairs that would be an appropriate seq. Then look at the differences between successive ops. If they are too large either ① put additional things in the Turing seq. to make some jumps smaller ② Devise a suitable set of abss & a hypothetical Pia Turing seq. that could have given rise to these abss. The abss are → the jumps were (then the old set of abss).

Moreover generally, the seq. of machines that we're moving along this Turing seq., can be any type of TM — say even most generally, a **PMTM** — & then we can begin to give this PMTM optimizn probs like TM_2's — etc.

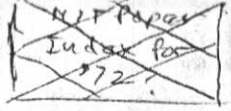
→ However, first try this with some sort of simple, non-PMTM Turing seq. — See to what extent it's possl.

→ It may well be that it is almost easy to have several modes in the TM, if reasonable jump sizes are to be achieved.

CMI Gen'l.
↓
407.01
Most specific
↓
Phm 400.01
spec
(His)
↓
3 > 9.01

N·12·73 PCA, TM 383

Gen'l. Notes Typ. Saqs. (.20).



01:32.06 1) On Provably Convergent Automata. (≡ PCA's). One oldish idea was to consider all pams that corresponded to PCA's — i.e. ~~PCA's that~~ pams s.t. they could be shown to converge for all input saqs. of all lengths. Given a fixed set of postulates for proving convergence, the set of all PCA's is then r.e.

2 Q's (a) How does one go about adding new postulates? By God's Thms (~~Also~~ or by a simple diagonal argt.) to find a Pam that frustrates the resolvent system, there is always an additional postulate one can add.

(b) Is the resultant set of pams the same one specified by the "All ipams Method"? Even if not, if one deletes pams from the "all ipams method" in an "unbiased" way, one will probably end up w. a system that's about the same as CMI →

on Typ Saqs. 381.20

Make ref. in "Typ. Saqs".

.20 2) Notes on Typ Saqs, PMTM, QATM:

PMTM	795.01 – 796.18	just how it works
QATM	776.15 – 788.40	(mods of PMTM)

} These are all somewhat burnt.

Typ Saqs, probs for PMTM 789-790

3) The set of pams used is perhaps significantly difft. from those used in "normal" CMI — In CMI one uses a pam for the prodn of the N^{th} symbol, if that pam converges for saqus up to N symbols long. That pam can diverge for the $N+1^{st}$ corpus symbol.

In the PCA method, one uses a pam only if it converges for all corpus lengths — i.e. all corpi longer than N: eg to 2

See R 1, 2, 8, 35, 9, 10 for a discn. of provable convergence

D273 Genl.

TM407

01;378.40 CMI General : IE³ Info Th. Xtns. Nov 73 p783 : A longish art on "Universal Noiseless Coding." — Also some review in it. This problem appears to be close to the CMI coding problem. There apparently has been very much work done on it. Kolmog & T. Cover are some people who've worked on it.

Aug 24, 73

IPC

TM 308

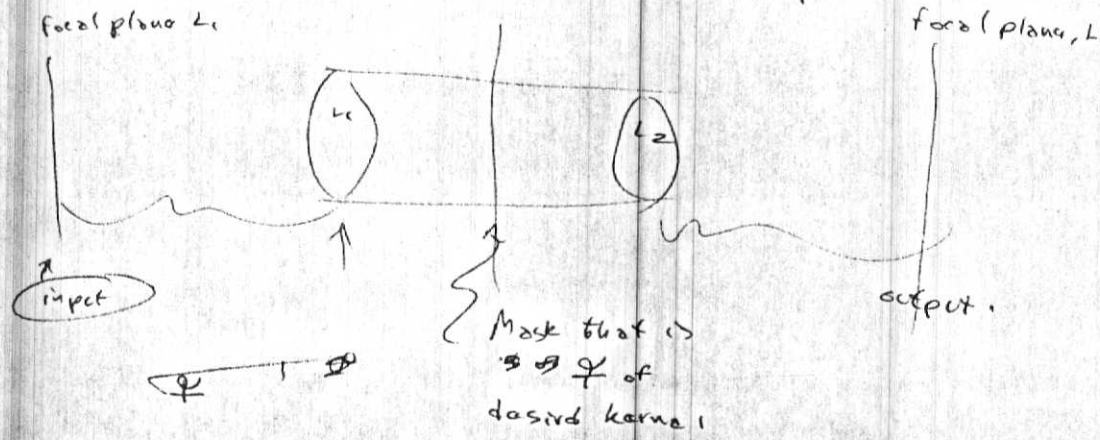
Ipc: [Phase may be later reflection etc.]

63.01 - 64.20
35.40

Optical computer; "Game of Life" can simulate a Turmac. — so it is a universal machine. It consists of a Boolean kernal on pts in a 2 dim. cartesian array.

Scientific American; Martin Gardner; Also William Gosper, MIT AI Group.

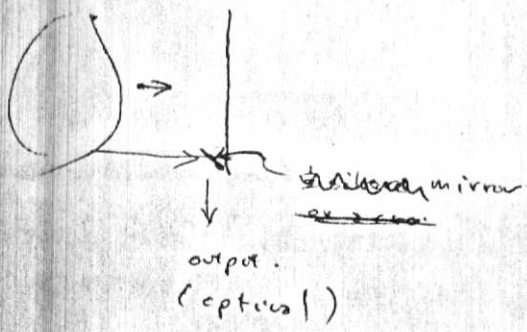
One might be able to do 2 dim Boolean kernels optically, by using a linear kernal & a threshold:

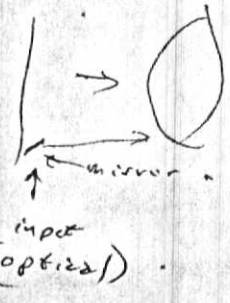


If this can be done rapidly w. shortwave lite, it would be very good. The full kernal might have to be done in several stages.

For a 10cm x 10cm array at 10^{-4} cm (= 1 μ) spacing, we have 10^{10} points. At even 1 ms per turn time we have ~ human IPC; at 10 ps (10^{-11} sec) we are ~ $\times 10^9$ human IPC!

Input & output can be at the edge of any 2 dim array:





One problem is how to get one's gain for the next kernal (turn).

21.01
309.01

Hair Memy

& perhaps simpler way to do t. forgg: simply have only 1 very long "stack" w. all t. info in it. The last word used goes to t. top of t. stack, etc before. Here, t. first N_1 members of t. stack have fast access time τ_1, The next " " " " " slower " " τ_2, etc.

CAM's are used for all parts of t. stack.

The result, is that the time it takes to retrieve a word from t. stack, depends on how far back in t. stack it is.

SN Bubble memorys may have much utility in making these stacks — the amount to a drum in which one can reverse them instantaneously. No! (This reversal ability only halves t. access time over an irreversable "drum" type memy).

So, one gets a fetch order. One moves from t. top of t. stack to t. position of t. wanted word. The retrieval time is k, t. distance down t. stack. It then takes k more units times to move that word back to t. top of t. stack, where one waits for t. next "fetch".

Hvr, by using many stacks in ||, one is concerned only w. t. access time of k, not t. "cycle time" of say $\geq 2k$.

28 One can use bidirectional shift registers in t. same way.

A (perhaps) Big trouble w. t. forgg (282.01 — 309.28) memy organizn: It would assign t. some very long address length to all words. Ideally, t. quicker access words should have shorter addresses.

One way to perhaps get around this: Use only a few randomly chosen bits of t. address to retrieve info. When a find is made in hyaccess part of memy more bits are compared. If a discrepy is found, one goes to a slower div. of memy (using more randomly chosen address bits) & repeats t. process.

21.01

11574 IPC

309.10
1: 308.10 Usually components by themselves will have zero IPC. Combinations of components as well as certain larger components, can have IPC > 0.

[Hvr. assoc. w. each component, is a no., α, ⇒ if α_i are r. α's assoc. w. a set of components, r. IPC of r. set is always $\leq \sum_i \alpha_i$ for r. set

I think I defined ~~something~~ r. IPC of (a set of) components(s), to be $\lim_{T \cdot N \to \infty}$ of $\frac{1}{T \cdot N}$ (No. of (selset) component(s) needed to realize r. N-th I,O relation) × (Time needed to do N-th I,O. relation).

These I,O relations could be listed by starting w. a fixed Turing machine, & listing all of its possl. input strings. We may want to compare r. sets of components IPC w. that of r. Turing machine.

In r. case of a Turing machine, r. cost will be a fixed cost of r. FSM + head, plus r. length of tape used, times r. time needed to complete r. calculation. For large calculations, this is → Tape length × calcn. time.

Blum's old paper on "A Machn. indep. Theory of computational complexity" (ACM ~ 1966) might be very relevant here.

This I heard Turing machine, seems to be clearly very inefficient. We would want a very large no. of heads (say → ∞) to get more ~~reasonable~~ efficient operation. However, we may not want no. of heads to ↑ as fast as tape length. If $\frac{\text{No. heads (+ FSM's)}}{\text{Tape length}} \to 0$, then for ∞ tape, r. (heads + FSM's) cost nothing. Any way ~~since~~ since r. no. of heads can ↑ < linearly w. tape length, we can ↑ IPC of Turing machine by a factor < tape length. — So IPC of TRMC is < $(\text{Tape length})^2$ — but it can be $(\text{Tape length})^{2-\epsilon}$, where ϵ is any no. > 0.

TM21

113.01

3.25.7+ 21.40

IPC

TM113

I previously computed an upper bound on the IPC of human brain by assuming 10^{10} neurons, each having direct access to all other neurons for each computation in .01 sec. Certainly a poor estimate. While each neuron might indeed have this access, in even, perhaps in < say 10 x .01 sec., this is a deceptive norm of what goes on.

Very probably there are Multiplexed "trunk lines" so that while each neuron can contact every other in a short time (as in a telephone system), they cannot do this simultly (as in a phone system), because of the limited capacity of the common trunks.

It may well be that most of the IPC is not contained in the trunks, so one can compute the IPC of the Brain w.o. knowing their capacity (or just having some upper bound on it).

Anyway, say a small arith unit does "its thing" in 1 sec. Then, the Q. is, how many bits outside itself does it have access to in T secs.) If we keep T small, various arith units will not interfer w. one another, & we can add their IPC's to obtain the ≤ ipc of the net. If T is large, we obtain a spuriously large net IPC, because the arith units would interfer & would not be able to use the same trunk lines simultly.

We might be able to get a simple expression for an upper bound on the IPC of a net in which each unit has n input bits & m output bits (w. m, n, small) & does this in 1 second. However, can we assign an IPC to a large "core" memory? — or, what about a "trunk line" (or a multiplexer), that has many inputs & 1 output? The ipc of these last 2 devices is very impt. because we will probably use a core memory as a (hopefully) main source of IPC in our TM.

N x 1 bit RAM can be regarded as a N bit word, w. some kind of multiplexed access to the bits — so one can only get at one at a time.

A ROM (N x 1 bit) is also a word & multiplex, but unchangable. Also a ROM can be regarded as a fixed function, w.o. memory.

Punctuation (among effects (Rings)

↳ A way in which $\alpha \cdot P$ may be better for MM than DM:

Using the previous Analysis, I get $C_{DM} \cdot A_{DM} = C_{MM} \cdot A_{MM}$ + a constant indep of S.
$\neq$ $\neq$ "
0

$A_{DM} + C_{DM} = C_{MM}$ + const indep of S.

Hwr, I didn't consider that there is a term in A_{DM} $\propto$ $\ln \ln S$ — This is a puncturing term that falls when the Pan beam ends & when the corpus beam starts. — I may have noticed this, but it was not regarded as important at the time, since I that that I had to correct a much larger difference between $A_{DM} + C_{DM}$ & C_{MM} — i.e. one of $\sim \frac{1}{2}(N+1) \ln S +$ const.

Hwr., I'm not sure this effect exists — I may have to go into the exact coding in much more detail.

268.88 — 269.40 goes into this. I think, hwr, that the algebra is very wrong!

Starting at 268.25 the precision of a single param of accuracy δ, will be $\left(\frac{1}{2}(1-\alpha)\right)^{\log_2 \delta} \cdot \alpha = \frac{1}{2^{\log_2 \delta}} \alpha (1-\alpha)^{-\log_2 \delta}$

$\text{n cost} = -\ln \alpha - \ln \delta + \log_2 \delta \cdot \ln(1-\alpha)$ $\quad \left| \approx -\ln \alpha - \left(1 + \frac{1}{\alpha \times 1.4}\right) \ln \delta \right.$
$\approx 1.4 \ln \delta \quad \approx -\alpha$
$-\ln \alpha - \ln \delta - 1.4 \alpha \ln \delta$

The α in n cost arose not considering the α's, is

$\boxed{-\ln \alpha + 1.4 \ln \delta \cdot (\ln(1-\alpha))}$ for each of $N+1$ components.

(The $N+1^{st}$ is δ & is a bit different from the other N).

Say $\alpha = -\frac{1}{\log_2 \bar{\delta}} = -\frac{.7}{\ln \bar{\delta}}$ $\quad \left|$ so 1 in n cost is $\right.$ $\bar{\delta}$ is average accuracy — perhaps $\alpha = \frac{1}{\log_2 \delta}$ with a fudge factor

$-\ln \alpha = \ln(-\log_2 \delta) = \ln((1.4)(-\ln \delta))$

$+ \ln(-\ln \bar{\delta}) + \ln 1.4 + 1.4 \ln \delta \times \left(\approx \frac{.7}{\ln \delta}\right)$

$= +\ln(-\ln \delta) + \ln 1.4 + \left(\approx 1.4 \times .7 = 1\right) \approx +\ln(-\ln \delta) + \ln 4 + 1$

Aug 27, 73

Puncture ~ (eff). $\approx +.35$ =

.01 : 314.40 ! So $\ln(-\ln\delta) + \ln .7 + 1 \approx \ln(-\ln\delta) + 1.35$

$\delta \propto s^{\frac{1}{2}}$ so this → $\ln(\frac{1}{2}\ln s) + .65 = \ln\ln s + \ln\frac{1}{2}$

$- \ln 2 + 1.35$

$- .7 + 1.35$

$= +.65$

$\approx \ln\ln s - .05$

Actually, since we must do N+1 params,

the total correction is ~

$$(N+1)\ln\ln s + (N+1)\times .65$$

This term is very approx — it may actually be zero.

More exactly, $-\ln\alpha + \frac{\ln\delta}{\ln 2}\left(\ln(1-\alpha)\right)$ | for $\alpha = -\frac{\ln 2}{\ln\delta}$

$= -\ln\ln 2 - \ln 2 + \ln\ln s +$

$\frac{\ln s}{2\ln 2}\left(\ln(1-\frac{2\ln 2}{\ln s})\right)$

$\approx -\ln(.7) - .7 + \ln\ln s + 1$

$.35 - .7 + 1 + \ln\ln s$

$\approx +\ln\ln s + .65$

$\delta = s^{-\frac{1}{2}}$ so

$-\ln\delta = \frac{1}{2}\ln s$

$$\alpha = \frac{2\ln 2}{\ln s}$$

$\ln\delta = -\frac{\ln s}{2}$

$\ln .7 = \ln\frac{1}{\sqrt{2}} = -\frac{1}{2}\ln\frac{1}{2}$

$= -\frac{1}{2}\ln 2 = -.35$

$-\ln .7 = \ln 1.4 =$

$\frac{1}{2}\ln 2 = .35$

so $\approx (N+1)\ln\ln s + .65(N+1)$

for small α ; i.e. small δ.

The Poisson analysis is not exactly right — i.e.

a more exact treatment would take $\alpha = -\left(\frac{1}{\log_2\delta}\right)$ i.e. α is

an average for all of the δ's. However, since each δ is $\propto s^{\frac{1}{2}}$

The results are about the same.

The Poisson result is imperfect & less good (but simpler)

SVH Gore of $\frac{F+A}{}$ is used. In this case MM is better

by $\sim (N+1)$ lnlns than DM.

If the Gore $\frac{S\cdot F}{S+A}$ is used, then DM is still better than

MM by $\sim \frac{(1-k)\frac{1}{2}(N+1)\ln s}{}$.

329.2013
on Puncta.

Punctuation

329.75
.01 : 315.90

A rather General Approach to punctuation: Code t. corpus using as many symbols (including punctu. symbols) as necy. This produces an intermediate code for t. corpus = C_0. Say there are r symbols. Use C_0 = C_1. Then code C_1 as if it were a modified Baru Seq. "Modified" means that at each point in t. corpus C_1, there may be constraints limiting t. next symbol, so t. probby distribn. over "t. next symbol" may vary (but in a known way) ~~with each successive symbol.~~ at each pt. in t. corpus. The set of symbols in C_1 & t. constraints of t. symbols in C_1 are regarded as "free" (i.e. given).

18 One way to get t. costs of coding C_1: Use laplaces rule in each "situation" that arises. e.g. ~~in~~ At a certain pt. in t. corpus we are in situation (≡ state) S_3. In S_3 only symbols ~~abcdefgh~~ α, γ & ϵ are possl. Then t. cost of γ, say, will be $\frac{n_\gamma}{(n_\alpha+1)+(n_\gamma+1)+(n_\epsilon+1)}$ where $n_{\alpha,\gamma,\epsilon}$ are t. no. of times that α, γ, ϵ has appeared in t. past in situation S_3.

In t. case of certain punctuation symbols, they can only occure once (or twice or some finite no. of times). e.g. "t. symbol that says that the previous set of 0's & 1's ~~is~~ represents an integer that is now terminated, & that this integer tells how many cols we want to use in our linear regressn".

It might be possl. to have more complex rules for "pooling of data" for t. various symbols. In t. forgg. (.18 ff) we regard each "situation" as separate & unrelated to ~~any~~ data accumulated in other "situations". In general, I think this need not be so.

.35 In general hwr., one can make up any rules that one likes for probys of t. various symbols. If these rules are devised before one has seen t. corpus, then one can legitimately do something like an "All pairs Method" to get

Punctn.

.01 339.40 a ≠ Σ prcby distribn. This Σ prcby distribn. will have A = 0, so one can use its observd recent Σ's to extrapolate directly into t. expected future Σ's.

The idea here is that the stochastic rules for generating C_t, give one an apripd over a set of Poms, so one can use Bayes directly to make predictions. I do not know if t. "All Poms matched" is in general correct, (see Index 323.05-.07 for Refs to discn.

.12 but in t. present case, it is legitimate.

FOR's!

v Willis 139.40

This may be because FOR's can only represent n-computable prob. measures. (i.e. rational), but this is because, for any finite output, one can only have finite inputs.

Q: If I give an FOR finite input — can I tell if it will ever stop?

.02 I suspect so — i.e. I think W. says it could only have rational fractions as output in this case — i.e. periodic. (see 352.30) (see 357.01) (Tho I haven't been able to find such a remark in ACM after a perusal of it)

.06 If so, a FOR is probly a FSM;

.07 very probly conversely, every FSM can be slightly modified to become a FOR?

If true, this fact would make it a lot easier to remember & derive propertys of FOR's.

It probly is easy to show every FOR ↔ a FSM → (Not easy) (Also, every FSM is not an FOR)

However, (FOR's don't nec'ly have S & U. yes they do!) ~~Every CB is a FOR,~~

Note ACM 248.03: W. says a CB can be made w. finite tapes & no repeating colns! This may destroy ideas .06!

.20 SN W's remarks (ACM 257.23-.28) may be almost identical to my proof that no other probly equiv. system can be better than CM!

Perhaps I can disprove .06 or modify it. I.e. try to find a FSM that violates having an upper bnd on the length of a pgm that will give an output of length m (or a specific output of length m). Or has finite output for infinite input:

.25 Consider a FSM: binary input, output: It counts (total) no. of 1's in the input: As soon as there are just 2, it prints 1 & stops. For each n (pgm length), there are a finite no. of pgms that will give 1 as output. — But for many infinite inputs, there is no output, much less infinite output.

.30 Also, there is no bound on the length of a pgm necy to produce a pgm of length 1.

So .02 is probly wrong. Tho — what about "slight modifn"? (↳ Hmm., see 337.10)

So, all FSM's are not FORS! But are all FOR's FSM's (or simulatable by FSM's)? (see 337.24ff for a good discn. of FSM's & FORS)

Q: Given any $L(m)$ ($L(m) > 0$, $L(m)$ integer)

can we construct a FOR ∋ (ACM 296.30) is satisfied?

I think so. We could first construct $M_R(m,n)$ to satisfy, then we could, by Thm 2 (ACM 243.32), construct the assoc. Pgmable Machn.

Note that $L(m)$ is an upper bnd on n for a gn. m. n for that m can be $< L(m)$.

.39 For FSM, must $L(m)$ be bounded in some way (e.g. $L(m) < m$ or $< m^2$?) (No! see 352.33)

337.01

FOR's

.01 330.40 ! W shows (ACM 2 #6.39 THM6) that for any FOR, R

$\eta_R(m,n)$ th. no. of m pgms of length n

$L_R(m)$ A limit on th. length n, of a shortest inputless pgm. that can produce $A^{(m)}$ as output

Def $\mu_R(A^{(m)})$ — The Kolmog. complexity (Ekcost) of th. seq. $A^{(m)}$

Are all effectively computable.

.10 Can these results be used to show that a FOR is a FSM?

Note: Since every FSM can be simulated by some Trmc, or "Pgmable Machn", when it is probably a CB or a umc it can be simulated by a FOR that does not have the symbol U in its output? Hmm, consider 330.25-.30 — th. counter example! conjecture.

FOR's may be more powerful than FSM's! — Hmm Note

Reason: I suspect that for FSM's $L_R(m)$ cannot grow too rapidly w. m, because of th. finite memory. (I'm not so sure of this) — also, because of counter example 330.25-.30, → Also FSM's can get into no read, compute, write loops but FOR's can't

.20 SN Note that a simple Time limited Umc is not a FOR, But can define a FOR. If th. time limit is T, then for all inputs of length $\geq T$, it must be in a S or U loop before it reads the active input. → 352.2

.24 A machine w. finite memory (a FSM) usually cannot be directly xfmd into a FOR. A FOR can get into pure Comput. loops, a FSM can easily get into such loops. One can fix the FSM by having a watching device to see that it never repeats the same state — if it does, the System automatically → U.

Hmm, a FOR also cannot read an EOF of input w.o. printing some output. We can assure this by having an aux device see that the FSM never repeats a state at 2 diffent times, unless it prints at least 1 symbol betw. those times. If it does do such a repetition itself w.o. printing, we go into a U loop. The aux device resets every time a char. is printed. Then it looks for repetitions of states since that last printing.

.37 If a repeat state, so were allowd w.o. printing, then the input seq., betw. those 2 identical states could be repeated

.36 352.01

FOR's

.01 ! 337.40 indefinitely, as we would have finite output for infinitely
.02 long input. Also, if there exists any input sequence that causes
that results in printing a finite amount of (say t. seq α is printed),
then we could get a certain input seq; terminating in α,
that has an arby no. of codes, of lengths that are
.06 arby long (in multiples of t. input repeating seq.).

But anyway, my impression is that a FSM that ~~had an~~ read an infinitely long input w.o. additional output could not be a FOR — unless we somehow knew it was in this situation & flipped it into U/ — in which case it would stop reading. / or S
So I think such a FSM behavior could not be worked into a FOR.

Willis mentions a FOR (ACM 248.03) that is probly I doubt to the device I described on 337.33-.37

.20 ! 337.23 Re: limited machines! on ACM 247.43 w. described a T(m) limited machine as a FOR. Here the time limit is a computable function of m, the no. of output symbols printed thus far. We can, from this use T(m) as an upper bound to L(m) (ACM 246.30) is the longest input needed to generate an output.
T(m) need not be an ↑ funct. of m. It can be 6 or even constant. — But it must be computable —

.30 ! 330.02 : Any FOR w. finite input must stop eventually.
If it did not, we would either have an ∞ comput. loop., or inf. output w. finite input — both of which are illegal for FOR's.
→ No! Inf. output w. finite input is o.k. — In fact, if an inf string is deterministic, it must have a finite determ. a.: be of probty (≡ prost) > 0. → 357.01
.33 ! 330.34 : For a FSM, L(m) need not be bounded (i.e. L(m) need not exist)
— An example is 337.38 — 352.06 (352.02-.06 in particular).

So, A conclusion is, that by ~~discarding~~ properly treating recurrent states (337.24—.37) we can xfm a ~~FORTRAN~~ FSM into a FOR.

→ Whether all FORs are FSMs or ≈ FSM's is not, hwr., clear

→ Consider the fole. FOR: For any string of input symbols

10.8.13

FOR's

.01; 353.90 of length n, it prints a total of just n^2 1's and stops. This cannot be a FSM because it requires potentially infinite memory.

It appears to be a FOR w. $L(m) =$ [$\sqrt{m}$] ← smallest integer >

It never uses ε, S or U symbols.

↑ reword

FOR's

352.30
353.40
330.01

The problem of 330.01 is open. I can't think of any way to tell if a gen. finite input has ∞ output or not. I think that ∞ output w. finite input is legal for FOR's — but whether it is possl. to tell in finite time — I don't know.

If it ever passed thru the same state twice, it would be in an infl. sequence of printing (loop — & we could tell this. But then, it would be using only a finite amt. of memory — a FSM.

In general a FOR can have ∞ memory.

Say the FOR was computing $\sqrt{2}$ — could a FOR do such a thing? I think so. Well: say if its input is 0, its output is 1, $5^{(\infty)}$,

" " " " 1 " " " the successive digits of $(\sqrt{2}-1)$.

So for any finite output seq., there are only 2 possl. pgms.

1	has 1 code (i.e. 0)	00 has the code
0	" " " : 1	0 " " "
01	" " " : 1	1 " " "
011	" " " : 1	
05	" " "	
055	" " "	

So $L(M)$ is fine. $\eta_R(1,\blacksquare) = 2$

$\eta_R(m,\blacksquare) = 1$ for $(m>1)$ } so $\eta(m,1) = 2$

$$\sum_{r=1}^{\infty} \frac{\eta_R(m,1)}{2^r} = k = \frac{\eta_R(m,1)}{2} = \frac{2}{2} = 1.$$

So it looks like a sw O.K. for FOR's.

* I doubt if any FSM could compute $\sqrt{2}$ — I couldn't because FSM's can only compute rational (= periodic) fractions.

26) Al Hyatt, Mike Fisher
Chomsky?

27) Rapaport? (Canada)

28) Giorgio Levi
I.E.I.
VIA S. Maria 46
56100 PISA, Italy

29) A. Rosenfeld, U of Md.

30) Ed. Fredkin, MAC

31) Peter Elias [interacted in "info retrieval" trump pf.]
Elias will be more Lof view of info theory oriented toward CMI as a AI approach, than are most AI workers. - i.e. use of prob theory, info theory, analysis, ect.

32) Bill Ma

SN ① Suggestion that Pat Winston's Thesis would be a good thing to apply CMI to — to Quantify some of his "Sort of" Grammatical Induction

② Marv. Said that Berlicamp had written a more complete analysis than Slagle, on Task Nets. (≡ SP)

③ Pol Policies: Papers on applicn. of CMI to various problems would be useful. e.g. ① also perhaps its application to linear regression & all of Y. associated results.

④ There are a few papers at Y. / Aug 73 A.I. confs. on Induction. They don't look V. G., but check them.

⑤ Pol / Another Approach to getting more people working on CMI: Write Paper or Give Seminar on Teach Course on: CMI: An integrated approach to Problems in A.I. Treat: ① Prediction (discrete & continuous) ② Operator extrapolation ③ Pattern discovery. ④ Mechanics of how CMI can guide the construction of inductive hypoths. — e.g. The "induction method" how is treated by CMI.

⑥ Some impt. ideas around the A.I. lab. to get familiar w. I can perhaps show how CMI could be applied to Thesis to optimize them! Perhaps put this in "CRIT" file.

a) Langs: Planner, μ planner, Conniver, QA4 (see μ Jan 71 to Jan 72 MAC Progress report for descrn. of "Planner" pp95-107 — Ibid p 88), Lisp, Trac, Snobal

b) Winston's Thesis on "induction" — How it is an improvement of Larry Roberts Thesis

c) Winograd's Thesis

d) Sussman's " "Hacker" — Aug 73 Sigart newsletter p26 Probly written in Conniver

e) Iva Goldstein's "

f) Marv's ideas on "frames" — how they differ from previous stuff (see my notes on Marv.)

e) Actor(s) — Sig Art newsletter Aug 73 p24; Also IJCAI'73

f) Chavniac's stuff on learning to Read

2-13-73 Pol

342

.01: 328.40 (6) (cont.) g): Read MIT AI Lab Quarterly or yrly reports to get General idea of what's going on

~~(h) Read the last Report [illegible]~~

(7) Publish Lengthy Abstract in AI newsletter.

373.01
373.01

Politics:

01: 342.40 General Meanderings:

Most immediate things to get done:

1) Review Results on Uniqueness of CMI : Mail out to various people.

2) "Study Paper" on CIAP : Various Scenarios ; On need for discussion.

3) Papers on "How CMI is useful."
   a) In Sci. Method.
   b) In A.I.
   c) It has interesting Math probs assoc. w. it.

4) Give Papers at various Conferences.

Present immediate Goals:

.10 1) Get lots of good people working on a) theoretic CMI
   b) CMI-based AI.

2) Get people thinking & writing about CIAP : Perhaps form informal "Colloquia"

Wrt .10, I think I want more general work on Theory of CMI. For this reason, the "Uniqueness" paper is important. Also, if I could get results or motivational "results" on optimum trial sequences of Pams, etc. — this for "computational complexity" people.

Specifically, what I want/is — most immediately — various derivative inductive methods — so that the Q. of $\max \frac{prost}{cost}$ becomes reasonably close to what is desired in practice.

.33 5) Paper: ~~Ex~~ a/o Talk: The Quantification of Historical Explanation — The Application of Historical Analysis to Problems in Artificial Intelligence. To be given at a symp on the history & Philosophy of Science a/o a similar Journal. Perhaps ask Hil. Putnam where to send this. [TN#5 Seq 401.30-.40 is also on this Q]

74TM 19.01

11474 Pol

:373.40 : TALK — say at MIT — slightly popular. TM19

Start out by listing ~~some important problems~~ in induction.

1) Prob. of Geom. Probly. Also Now it is just another form of assigning a prior

2) Expected error in curve fitting, if one is allowed to use any functional form one desires. How does one ↑ expected err. due to Atness? Show how one can fit ~~==~~ data arbly well w.o. any continuous priors (∴ observed error = expected error)

3) Grue, Bleen problem

4) Look in Sci Amer in Math section, I think, for /art on difficulties in induction. 1973, I think. Possibly a regular article by (maybe more likely).

5)

PSGD.

Previous refs (voluminous) exist:

Given a corpus: First make a concordance. This is a list of all substrings occurring in the corpus, w.r. freq. of occurance of each.

Unfortly, a complete concordance of the entire corpus is probly too expensive. It can contain no greater than $N \cdot \ell$ strings! where N is the no of characters in the entire corpus, & ℓ is the longest string. A better upper limit is $\sum_i \frac{\ell_i(\ell_i-1)}{2}$ substrings; ℓ_i is the length of the ith string. $\approx \frac{1}{2}\sum \ell_i^2$

So it may be possl. to have a complete concordance. We may want to put parts of it into slower memory. Say 100 strings, each 20 symbols long: $\frac{1}{2} \times 100 \times 400 = 20K$. 20K is not very large, but we need to store the names of the 20K substrings as well as the no. of occurances of each.

Using such a concordance, it will be easy to try out various grammar rules of the form $\alpha \to \beta\,\gamma$ (β, γ are interm. or term. symbols) or $\alpha \to \beta\,\gamma\,\delta$ or $\alpha \to \beta\,\alpha\,\gamma$ etc. I don't know how good it would be at testing rules like $\alpha \to \beta, \gamma$ i.e. the class α is the Boolean sum of the classes β & γ. or $\alpha \to \beta\,\gamma, \delta\,\epsilon$.

→ 347.15

Anyway, it should be possl. to compute under what circumstances one would save on cost by doing the concordance first, before making trial rules; or just keeping the corpus & doing a freq. run for each new rule

3470

01:346.40 to be tested. I suspect t. concordance would involve less cost — but one has to include in cost, the cost of many of various access times — i.e. a properly designed heir many.

I am thinking of this concordance as being vaguely analogous to a spectrum.

Some version of t. Bible has an IBM sponsored concordance. Perhaps I could use this to devise a reasonable set of grammar rules for part of "English" (or Latin)

15 :346.35 — I could try all $\alpha \to \beta\ \gamma$-type rules by (α = no. of symbols in α) taking all phrases ($\exists x$) in looking at t. N_α ways that $\alpha \to \beta\ \gamma$ could occur. This may be a big task, but I think there will be many zero entries involves little computation.

To search for $\alpha \to \beta, \gamma$ type rules is not apparently obviously related to t. Concordance. One way to use t. concordance for this: Say t. conc. is in alphabetical order. We can then ask for t. (wtd) set of strings that follows bcc (say). Call this wtd set S_1. We then ask for t. wtd set following aac (say) — call this set S_2. If S_1 & S_2 have a similar profile then we will gain something by combining bcc & aac into a single set.

Actually, we also will want a "backward concordance" in which t. sample of all ass is given backwards (i.e. each ass is on. backwards) & a concordance is made.

In general. Much of t. foregg. ideas & other good ones, are discussed in my "Mechn. of (ing. Learn" (Namur).

358.01

N Oct 18, 73

TM 358

PSGD

.01; 347.40 : One problem in evaln. of t. "fit" of a grammar - Say, for each sample, s., one has 3 difent possl. parses wrt a gn. Gramm. If there are N samples s's, we have 3^N difrnt. ~~apsipe~~ apsip's.

Each parse can be represented by a vector, & choice of N vectors "sums" ~~together~~ these vectors in some way & t. result gives apsip.

~~How~~ To get t. total apsip of t. deterministic grammar, we wite want t. Σ apsip of these 3^N parse sets. I.e. we would be most interested in t. gramm for which that Σ apsip was max.

For prediction of t. apsip of a new possl. s., we would have to Σ t. apsip of all 3^{N+1} parse sets.

Very probly, we would need to consider only t. few "top" parse sets.

Actually, most s's will have >> 3 parses, & t. total actual parse vectors can be expressed in some simplified "factored" form.

Biblo

T Pgmrs Introdn to Lisp

W. D. Maurer (1972)

MacDonald/Amer Elsevier Computer Monographs

$4.95 102 pp.

Looks rather simple & has lots of exercises, Answers to many. Looks easy to read.