~~We we~~

so taken as a pair, 48).24 could be used to find

A_1, B_1, U_{s_i}, U_{Nt_j}.

Then, using the above info, 48).25 could be used to find

A_2, B_2, U_{Ng_j}

However, we also have 48).26, which has U_{Ng_j} in it.

We want [redacted] $U_{Nt_{ij}}$ $\approx A_3 U_{s_i} + B_3 U_{Ng_j}$ as well as possl.

so it is clear that 48).26 and 48).25 are coupled.

Essentially the involved calculations used on 48).24 (which are described in detail on 18)), should also be used on 48).25 and 48).26; but .24 is more important as far as accuracy is concerned, so the ~~complex cal~~ involved calculations may be worth while.

From 18)

we want to chose U_{s_i}'s and U_{Nt_j}'s $\ni$

$$\sigma^2 = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{\left(\frac{c_{ij}}{T} - (A_1 U_{s_i} + B_1 U_{Nt_j})\right)^2}{\left(1 + \frac{c_{ij}}{T\sigma^2}\right)}$$ is minimal.

this gets us σ^2, U_{s_i}'s, and U_{Nt_j}'s.

Then we want

$$\sum_i\sum_j\left(U_{Nt_{ij}} - (A_2 U_{Ng_i} + B_2 U_{Ng_j})\right)^2$$

$$+\sum_i\sum_j\left(U'_{Nt_{ij}} - (A_3 U_{s_i} + B_3 U_{Ng_j})\right)^2$$

to be min. by chosing A_2, B_2, A_3, B_3 U_{Ng_i}'s properly.

with very little trouble, we can add str → str.

Be sure to mention use of U_{apri} in ~~making~~ deciding which objects are to remain in memory.

A big trouble: If U's simply add linearly to obtain U_{apri}'s, then if an object has hy U, it will combine with hy U_{apri} to give reasonably anything — even objects of $U = 0$, to give reasonably U_{apri}.

We can avoid this by multiplying U's to get U_{apri}'s — There is ~~only one~~ only one ~~extra~~ coiff. we can adjust, however —

No we can write .. $U_{ij} = (N_{S_i}^{\alpha}\ U_{N_j}^{\beta})^{\frac{1}{\alpha+\beta}}$

These "optimum curve fittings" can all be done by a Monte-Carlo method. — This may be nec. if multiplication is used for U_{apri}'s.

Th. work of T.M. consists of:

1). Making predictions by finding th. pngms appropriate to a gn. $\square$. This may involve a "search" process, but it can be a simple exhaustive search in a simple t.m.

2) Keeping C_{ij} up to date. — This is ~~a~~ more difficult than 1), since one must try to get all pngms that are relevant to every $\square$

3) reevaluating all of th. U's in view of th. various equs.

— Actually, th. only U's that need simultanious optimization are ~~the~~ th. ngram U's. — except that if one gets pngms by str x ntps; and $\frac{ntps}{ntp}$, and some other ways, then ntp U's may need simultanious optimizations.

4) Throwing ~~out those~~ objects out of th. memory if they have too low U's., and bringing in new objects of hy enuf U.

Th. initial ngms and ntps. are th. d initial digits — they each have U's of $\frac{1}{d}$ to start off.

---

equ. 3.2.1.5.3 ≡ page 25 of Dart Report

.37 In th. equs. of α116.21 perhaps th. various terms should be weighted in accord with th. freq. with which they occure.

Perhaps, for the exposition, just use 3 or 4 combination methods and write all of their eqns. Then show just how far one can go in predicting with them. Include also some extra comb. methods — with th. statement, that in using them, one can go much further.

It is possl. to do away with ngms. I.e. make them th. result. of multiplying various utps by th. str [1]. For purposes of defining things, however, a ngm may be a useful concept. Perhaps we would like to lump pngms and ngms together — tho even so, we would probably want to separate them in various search processes.

Th. Search Problem (for th. relevant pngm)

This may be a very formidable problem. Th. reasons why it may be tractable in T.M.

1) Many parts of the search are indipendent, so that they can be done in || to save time. Probably th. Human brain does it this way

2) There are many clues as to how th. search should proceed.

a) try pngms of any U first, since they are used most frequently.

b) The methods of formation of th. pngms will suggest search techniques.

c) We can use some simple T.M. techniques, to categorize situation into (pngms$_2$). These pngms$_2$ suggest places to look for th. appropriate ordinary pngm to fit a situation.

Note that in the neural net approach, there appears to be no search problem.

Another approach that may avoid th. search problem is one that subjects th. input to a series of tests. Th. result of each test helps determine what th. next test should be.

Thurs ~~Wed~~ Aug 2, 1956

The results of all th. tests — or perhaps th. last test only determines th. output.

There is some Q. as to ~~whether~~ Math T.M. is worth persuing to th. end. — It may be worth while to get everything arranged up to th. search problem — i.e. make sure that it can devise any desired useful abs. — then abandon it for one of my more recent random net approaches.

Another possy. mite be to use cont. T.M. (described around Minsk 50) for th. search problem.

However, it would seem that I should be able to solve this search problem by refrence to th. ~~lan~~ intuitive lang. — which is th. normal way to solve all difficult problems.

A snag: ~~E~~ in = 1 0 1 1 ect. —
1 0 1 1

suppose we have found ($\boxed{1}$ 0) to be ~~[illegible]~~ inconsistant. We really should store this fact, so that we do not have to try it out every time ~~[illegible]~~ ~~one~~ we start hunting for ~~at~~ a new ~~[illegible]~~ pugm.

Perhaps it is reasonable, that we should store ~~E~~ inconsistant pugms of hy apri U. We may store them in th. foll. form: suppose Pugm, $P_k \equiv S_i \times N_{tj}$. and P_k is found to be inconsistant. Then in S_i's info register, record N_{tj} — which means that S_i will never again try to mult. itself by N_{tj}

But suppose P_k is ~~—~~ consistant, but merely has low U's? → 54).30

.34
.35

The set of all Computing Machines can be ordered in th. foll. way: Take a set of Boolian ops. that are complete. (using unity time delays in each) — say joint denyer and unity delay time. Then say one has i ~~•~~ joint denyers and $(n-i)$ delays. One can then connect ~~any~~ each output to one or more inputs. Consider th. inputs to th. machine as ~~as~~ "outputs" and th. outputs of th. machine as "inputs". One can then construct all machines for a gn. n, i, n_i, n_o, where n_i and n_o are th. no. of input and output lines, resp.

n, i, n_i, n_o are pts. in a 4 dim. space, and ∴ can be linearly ordered.

·15 ~~If~~ say one took a ~~ll~~ computer~~s~~ ~~in order~~, and gave ~~them~~ it the first n elements of a time series, and ~~changed~~ asked for th. $n+1$ th element. And one did this ~~for each~~ ~~for each computer~~ ~~for each~~ ~~element~~ at 10 positions in th. series. If one searched thru all computers in order, to get one that gave th. rite ans., ~~was~~ all 10 times, and one found one, would it be more likely to give th. rite ans. th. 11 th time?

A counter example! Consider all positive integers, in decimal notation, with ∞ of 0's extending to th. left. Th. ith digit to th. left of th. decimal pt. is th. ith member of th. series. Let the rth computer construct th. time series represented by th. no. r. — Then th. extrapolation is always zero. Now this counter example is not in th. spirit of a machine that actually constructs th. T.S., but I think, in general, that one does have to have strong apri. info about a universe before one can ~~pr~~ predict in it.

---

30. 53).34 A possl. soln. of this: Th. primary T.M. carries all info in its mem. that it has computed. As such, it also carries info about which prgms are inconsistant, and which are of low U.

Now, as an economy measure we can remember, in groups, which abss ~~ab~~ have been combined with which abss. already. Th. useful results of such combs. will be in th. rapid access mem. Th. low U or inconsistant results will be in a large, ~~sl~~ slow access storage (say tape or drum).

When a new abs. has been created, and gn. hy Us all of th. old ~~ab~~ combs. will be tried on it to ~~determine~~ ~~which~~ an try to get some new hy U ~~ab~~ abss. The content of th. ~~ta~~ slow access register can be periodically

Fri Aug 3, 1956

(35)

~~ranwanted~~ examined to see if some of th. low U ~~objects~~ abss have ↑ in U, or some of th. pugms have become inconsistant.

Note that th. above is rather ad-hock — but it is an economy measure, so this is permissable.

Note also, that certain info is storable on tape for non-random access — i.e. sequencial searching.

Also, certain info can be stored photographically, for rapid random access, but ~~[illegible]~~ no erase, and accessability only a few seconds/ or longer, after writing.

The / Boolean operator T.M., may be an easy way to avoid some search problems — tho clearly ~~[illegible]~~ avoidance of all searching seems inexpedient, since humans do do much searching.

In particular, one can record by their group names, pairs of sets whose β products has been tried out to make new abss. We can then record that all th. resultant pugms are inconsistant ~~[illegible]~~ except th. foll: (.......). Th. hy U pugms are kept in rapid access; th. low U ones on tape/ on drum, for occasional systematic modification.

Th. procedure in th. present T.M. is to assume we have about all th. memory and speed we want. ~~[illegible]~~ This model should be very thoroughly described — and then later, ad hock measures for economy of time and space will suggest themselves.

Sun Aug 5, 1956

Future T.M. work!

1) New comb. methods } work these together and reduction in no. of comb. methods
2) More difficult probs. ← simplification to a small basic set.
3)
~~4~~) Economy methods to be studied (include search methods)

Explain gen. orientation — why this wouldn't be such a difficult problem. Explain why search of 10^5 ~~ngs~~ pngms over 10^9 cases takes ~ 1 sec.

Use 1) neighboring strs.
2) creation of utps from ngm, pngms.
3), 4) adding, subtracting $\square$ to get pngms ngms.
5)

In the "rules" ~~#~~ 1 to 6 of $\propto$117.34 ff, it would seem that "count" is irrelevant, since the equations $\propto$115.15, $\propto$116.02 do not depend on $\aleph$. Th. only importance of count is a) in resolving conflicts (which are rare) b) making pngp inconsistant. However in Non-Math T.M., "count" is important —

In th. "rules" of $\propto$118: perhaps th. prediction should be made after th. new abss. have been made and tested. There is no attempt in this scheme to invent new abss. that are relevant to th. problem in hand.

Perhaps divide rules into A) Making predictions B) ~~\$~~ Updating th. prarameters.

For A): 1) try to find pngm with at least 1 previous case. (This condition is automatic if only pngms with ~~1 previous~~ at least 1 case are stored)

$\boxed{\text{SN}}$ If a pngm has no previous cases up to T, total $\square$'s, then its ~~####~~ U is $\sim \frac{1}{T}$. However, since th. tng. seq. continually changes in character, this may not be such a good approximation.

Mon Aug 6, 1956

~~is~~ invented due to convenience, ~~but~~ in book keeping, but have no cases yet.

Search procedure for prediction: 1) search pugms with 1 or more cases. 2) pugms of hy U with no cases or pugms that have been 3) creation of new pugms that fit.

4) After fit has been found, search for previous cases (positive or negative).

much seriousity

At this pt., perhaps I should go into some of th. "economy measures" in searching ......

5) Previous case may or may not be found.

Perhaps no "new" pugms need be created untill required by specific q. elements

.18 So: T.M. starts out with a q. element. It has no pugms — only th. digit. ngms. It ~~then~~ makes pugms of them, by adding □. Since none of these are consistant, make addition and subtraction of ▭ free at first. This would save a lot of trouble. ~~This~~ would save a lot of trouble. — with no luck.

SN

This would almost elim. th. difference betw. ngm and pugm.

→ it tries mult. by strs. to obtain new pugms — with no luck. Next, it tries creating new strs — e.g. etc. and mults. them by th. mongms 1, 0, =, ect. — these prove useful.

It ~~will~~ may try for various °'s of goodness in th. pugms it gets. It may settle for 1) a pugm that fits 2) a pugm with at least 1 ~~case~~ count 3) a pugm with at least 1 case. 4) ≥ 1 case.

All of th. above is somewhat out of th. spirit of th. T.M. with very large memory and hy speed. It is based on sequencial, rather than ∥ operation

(to 61)

SN ~~Method of~~ physical realization of th. scanning ~~for~~ various ngms or ~~pugms~~. It can be done, say, optically, but

Great idea!

Need not be. To scan for th. ngm 1 0 = , we
1
simply raster scan, using this optical kernel. — or we simply use th. ~~[illegible]~~ kernel

1	2	3
4	5	6
7	8	9

, and recognize when squares 1, 2, 3 and 5 simultaneously have th. rite digits in them. When this happends, a pulse is emmited from a certain ckt..

To recognize a ngm. that ~~[illegible]~~ is formed thru multiplying ~~a~~ an ntp. by a str., we watch for any of ~~[illegible]~~ a certain set of temporal spacings in th. pulse outputs of certain ngm. recognizers. ~~[illegible]~~ It is possl., probably, to work this, so as to get ~~configurations~~ ~~[illegible]~~ configurations of these configurations, as well.

In fact, <u>all</u> of th. scanning can be done in <u>time</u>, rather than space, even at th. lowest order. This makes possl. th. same kinds of ngm. recognizers at <u>all</u> levels.

We can have a kind of ~~[illegible]~~ "filter" for each ngm. It will have a "weighting function" ~~That~~ recognizes certain temporal digit configurations. A str., will be a certain kind of set of such ~~temporal~~ digit configs.

This technique gives <u>true || operation</u>!! → to 59).01

<u>All</u> one needs is discrete delay, ~~and coincidence ckts~~!
~~All~~ of th. past history can be stored on a <u>linear tape</u>!

30

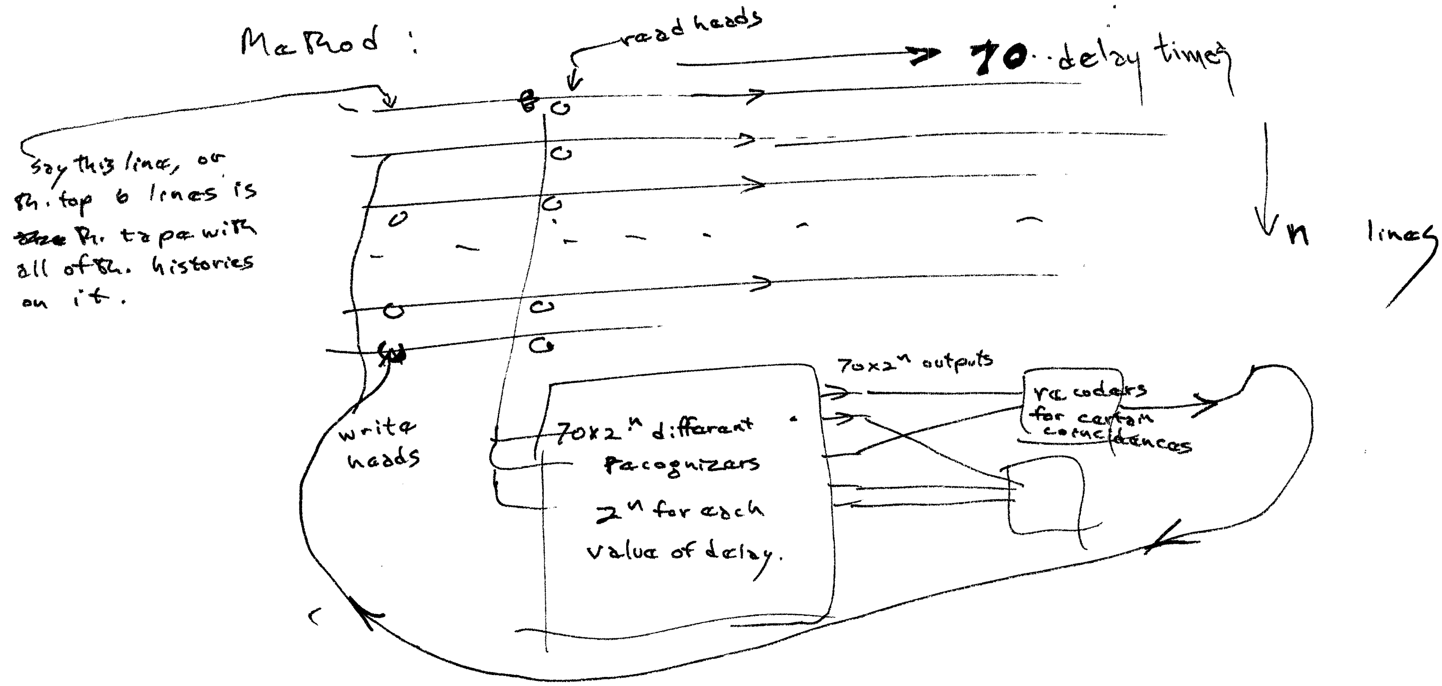
UN fortunately, it isn't at all clear as to how this technique would work for "sets of sets of . . . " , — but I'm not yet sure how I'm going to use these, anyway! It may, indeed, work out o.k.!

Work on this opus is proceeding at an exponentially slower rate. I am afraid th. series will converge before I finish.

.01 from 58).30 : ~~Delay~~ Only a few delay lines are nec — i.e. If there are 70 squares, ~~delays of~~ then 70 delay lines are adequate. — A few quartz ~~...~~ polygonal "lines" may be enuf. Then for each level of abstraction one may want these same 70 delay lines. Perhaps for all levels of abstraction, above th. first, one will need several separate lines for each str.

Well : for each set of n delay lines of 70 taps, one can have 2^n different input ngms being delayed, if th. tape is n bits wide. ~~...~~ One can get as many ~~...~~ ngms as one wants on a set of lines in this way, since for a delay of $70n$ Δt's one gets 2^n different adss, it is clear that th. delay lines are not a bottleneck. Th. only problem may be th. vast no. of diodes required for th. coincidence ckts.

Method :



So one need not have too many delay lines — only lots of diodes and perhaps amplifiers.

Using this method, ngms that have a hy of ambiguity, due to their being derived from a str. mult.

by an ntp., can be resolved into several cascaded coincidence ckts, rather than ~~3~~ 2^r coincidence ckts in an "or" arrangement. for a str that mults. by a 3 gram and a 5 gram, only $3+5=8$ coincidence ckts are needed — rather than $2 \times 5 = 15$. This becomes very imp. for 3rd or 4th order strs. — say $3+2+5=10$, but $3 \times 2 \times 5 = 30$. This also seems ~~to~~ about rite, intuitively, in th. sense that this is about th. amt. of ambiguity one expects.

Perhaps th. "closeness" conditions can be built rite into the recognizer matrix — ~~in th. sense of coincide~~ One can construct abss. at random, by making coincidences betw. any 2 outputs. Th. aprip of ~~2~~ coincidencing any 2 outputs can be made ∝ th. U's of these 2 outputs and a ↓ funct of th. "distance" betw. them.

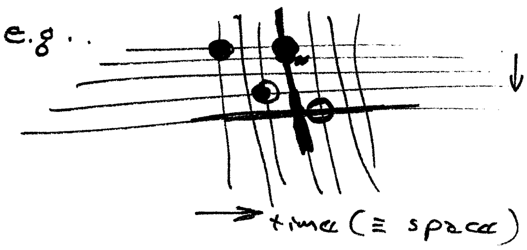
Th. idea of an ntp. or of similarity of **str.** is not so easily put ~~into~~ this form (as perhaps also, th. sets of sets ... idea)

| Th. sets of sets ... T.M. will probably work with them in ~~a~~ lot dift. way from that contemplated for ~~a~~ th. simpler kind of Mstr T.M.

Use **ZATO** coding! — This makes addition of new abss. into th. matrix very easy! By using only a very small % of extra digits, th. probty. of overlap can be made very small. — but this should be examined

A str., in this notation, is an ordered set of spacings:

e.g..



in a ~ way, an ntp. is a set of horizontal lines.

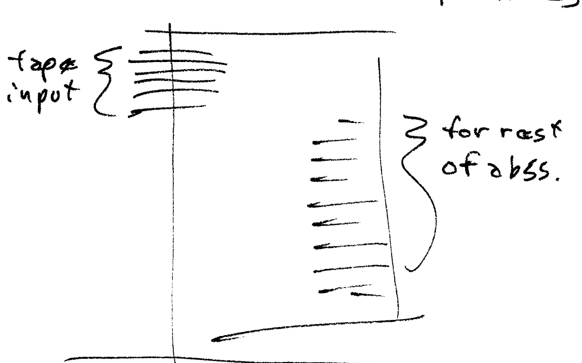
We can, in this way, make th. product of a str. and an ntp., unique

An ntpst, is simply th. "or" of a bunch of ntps — but this doesn't work too well for ntpsts* where one has an intensional definition — i.e. a set of instructions ~~that tell~~ one how to recognize it.

It is perfectly clear that as defined here, an ~~ntpst, can~~ ngmst can be an ngm, by simply giving its "or" a code ~~name~~, and giving it one of th. horizontal lines.

It isn't clear to me just how I want to use th. sets of sets ... idea, so I don't know just how I want to fit it in.

Th. nice thing about this, is that one can ~~simply~~ add on more abss., without too much ~~trouble~~, Th. ∥ delay lines can be an H-drum, fed by a H-tape, for th. 6 top lines.



We can have 10 heads/inch in th. direction of drum travel. For 200 pulses/inch, this is 20 pulses betw. heads. Each 20 pulses can be a different abs. — we will need many short, 20-pulse time, delay lines

SN

It may be possl. to use T.W. Tubes with long distances betw. ants. for a very hy speed sequencial memory such as is needed here. This can give very large

Many of Th. problems of overlap, when multiplying a str. by an ntp., are eliminated, since impossl. overlaps never occur. Also, Th. members of an ntp. can each be multipositioned &/o multivalued, ~~very easy~~ in a manner that is very easy to instrument with This device.

Th. method of storing ntps and strs is not yet clear.

I will not differentiate betw pugms and ugms in this discussn. at present.

Since Man's total life input has been estimated at $< 10^{10}$ bits (check on this (I think V. Neuman talks about it in th. Hixon symp.)). At 2×10^6 bits/sec, (= 1Mc B.W.) this is $\frac{1}{2} \times 10^4$ sec $= \frac{1}{2} \times 3 \times 3{,}600$ sec $= 1\frac{1}{2}$ hrs. to scan thru Th. whole thing. With a t.W.T. at 1000 mc. B.W, we can go thru in 5 sec. With a $\frac{5}{N}$ of 128, we can go thru in $\frac{5}{7} \approx \boxed{1 \text{ sec.}}$

McCullough mentioned ~ ~~___ bits/sec~~ < 25 bits/sec for humans. In 20 yrs:

$3{,}600 \times \frac{1}{3}$ (8 hr day) $\times 1000$ (hr/mo) $\times 12 \times 20$

$= 1200 \times 1000 \times 240$

So we ought to be able to scan th. whole memory in not too much time. $= 1.2\,M \times 240 \approx 250M$ bits

200M bits is 1 second. — At 100Mc B.W.

Angular invariance is hard to get with this schema — but we can use a special mechanism — I think in Humans, Th. angular invariance isn't too good — it is certainly not good over $> 90°$.

How to get ntps(?) and perhaps strs (?)

An ntp. can be the "or's" of each of its rows, all "anded" together. Let P_{ij} be th. point of th. ith abs., of time delay j ($j = 1/70$; $i = 1/2^n$) for th. ntp. (N_{g3}, N_{g1}, N_{g5}) we want $\left(\bigcup_{j=1}^{70} P_{3j}\right) \cap \left(\bigcup_{j=1}^{70} P_{1j}\right) \cap \left(\bigcup_{j=1}^{70} P_{5j}\right)$

But what about th. order 3, 1, 5?

In general, one must simply write out a Boolean expression ∋ it is exited every time a certain ntp. is used. N/ly, one can do this for ~~all~~ each str. To multiply a/ partic. str. by a partic. ntp., one simply "ands" them. This isn't entirely clear.
—Also, it isn't clear that this is an economical way, as far as diodes is concerned.
Also, what about ntps of th. form (N_{01}, N_{03}, N_{01}) and strs of th. form $\begin{array}{|c|c|c|}\hline 1 & 3 & 2 \\ \hline & 3 & \\ \cline{2-2}\end{array}$?

This super hy speed // method seems to be very close to human psych. and physiol. — but in several ways it seems ~~far off~~.

— this doesn't seem very economical ~~to~~ do for strs.

Troubles: Method not to hot for strs, ntps, and possibly sets of sets of . . . .

Th. trouble may be that ~~ntps~~ ngms do exist in th. brain, while strs, ~~and~~ ntps do not exist as such in th. brain.

See 66).01 for trial way to get out.

Tues Aug 7, 1956.

From 57). I have become stuck on 57).18, for about a day. What the problem seems to be: It surrounds the "rules" of α118. I would like to put them in a form so that I can use them as a routine in the illustrative examples. Essentially, the suggested routines of 57).18 are a short cut. In α118, the rules are more in the spirit of being done continually — at all times while T.M. is operating — true ∥ operation.

A general description of initial T.M. operation:

The tng. seq. starts ~~off~~ with a q. element, scans thru the ~~memory for~~ available pngms for an appropriate one. There are none appropriate ones, (none at all in fact), so it starts trying to make them.

First step: Adds $\square$ to available ngms. — scans thru memory — none are consistant.

Second: Mults strs. of hy U by ntps of hy U. — no/strs. useful available yet. — just $\boxed{1}$. Results are useless.

3rd: Makes new strs, new ntps., multiplies to find new ntps. $\boxed{1} \rightarrow \boxed{1\,1}, \boxed{\begin{matrix}1\\1\end{matrix}},$

$$d_i \rightarrow (d_i, d_j)$$

$$\boxed{1\,1} \times d_i \rightarrow d_i\, d_i \; ;$$
$$\boxed{\begin{matrix}1\\1\end{matrix}} \times d_i \rightarrow \begin{matrix}d_i\\d_i\end{matrix} \rightarrow \; \vdots \; , \; \begin{matrix}\circ\\\circ\end{matrix} \; , \; =$$

4th add $\square$ to get $\boxed{1}$ etc. These prove consistent and useful. Retain them in the pngm. memory. (probably retain all consistant pngms in the pngm memory)

~~Also retain all consistant pngms in the m~~

~~(Later as an economy measure, all pngms~~

5th ~~modify~~ (update) ~~all~~ U's of all pngms, ntps, strs, ntps.

A ~~more~~ more economical ~~better~~, way: When ~~and only when~~ T.M. gets a q. element, 1) → it records it in th. array memory 2) it scans its memory for th. rite pgm.

3) if it finds th. rite pgm, ~~it stops~~ makes a prediction and stops. if not it goes to 4) ~~If not, it~~ and waits for th. next q. element or element.

1) When T.M. gets an element, it records it in th. array memory and waits for next q-element or element.

4) Not having found a suitable pgm in its pgm memory, it trys to make up one, using standard means. — It makes up ~~several the~~ many that ~~will~~ fit th. □, then it scans thru th. ~~new~~ array memory. During th. scan it

a) finds which of th. ~~new~~ (and old) pgms are now inconsistent, and also updates ~~[illegible]~~ all pgm U's.

b) then it updates all ~~other~~ abs. U's.

This step needs going into.

If it didn't find a consistant pgm., it ~~[illegible]~~ ~~tries~~ again, making newer abss. with th. new, updated abs. U's. More detail for each step, with a justification of each thing that is done.

It then scans ~~th.~~ newest pgms thru th. array memory for consistancy ~~th. array memory~~

if unsuccessful, tries ones with even smaller Uapri ~~[illegible]~~ ~~Uapri~~.

A reason ~~this pro~~ why one would want this process done in many small steps, is that a ~~[illegible]~~ new hy empirical U abs. (≠ pgm), may be discovered, that will give imp. results in ~~getting~~ th. desired pgm.

There are 2 Q's:

1) How much less ~~2~~ accurate is this system than th. one that updates at every (q) element? no good! ~~[illegible]~~ → see 69).30

2) What about th. manipulation of utps, strs, ugms that is constantly going on? — This is imp., and may suddenly result in a abs. (~~of~~ pugm) of hy empirical U, tho low Uapri. → to 69) →

SN → th. scanning of th. pgm. memory is done by th. same ~~mechanism as th. [illegible] done~~ equipment that scans th. entire array memory for updating, ~~[illegible] put in rapid access m~~ with perhaps small modifications. Th. ~~[illegible]~~ q. element is simply run thru th. pgm raster once. → to 69) →

from 63)

.01 $\boxed{SN}$ If I can't find a neat way to do strs. and ntpsts (and perhaps sets of sets), try to get some new, =ly powerful concepts that are more easy to instrument. Th. easy instrumentation of ugms and ~~other~~ complex ugms, suggest that th. delay line scanning idea may be ~ to human brain. Perhaps hum. brain has only th. part that I have outlined — with provisions to (perhaps) — Tho an ugm can be useful without being a cons. 3 pngm,

A big trouble! strs, ~~and~~ ntps ~~and ugms~~ aren't ever discarded —

→ ~~they~~ develope certain apri pngms from others, but only via neighboring strs, if th. ntp remained fixed, or relatively fixed. — Perhaps th. brain can work in th. foll. way: useful pngm$_1$ $= S_* \times N_{t1}$

to get to pngm$_2$ $= S_* \times N_{t2}$ is of apri hy U_s N_{t2} must be "close" to N_{t1}. Similarly, to get from

$P_3 = S_2 \times N_{t1}$ to

$P_4 = S_3 \times N_{t1}$ — S_3 must be close to S_2.

To get $P_2 = S_1 \times N_{t2}$ to be of hy apri U, there must be 2 pngms P_1 and P_2 ∋

$P_1 = S_1 \times N_{t1}$

$P_2 = S_2 \times N_{t1}$ with S_2 close to S_1

$P_3 = S_2 \times N_{t2}$ and N_{t2} " " N_{t1}

or

$P_1 = S_1 \times N_{t1}$ with S_2 close to S_1

$P_2 = S_1 \times N_{t2}$ N_{t2} " " N_{t1}

$P_3 = S_2 \times N_{t2}$

actually that $P_2 = S_2 \times N_{t1}$ with N_{t1} close to N_{t2}

or $P_3 = S_1 \times N_{t2}$ " S_1 " " S_2

with P_2 and P_3 of hy U, is enuf to get

What we need.

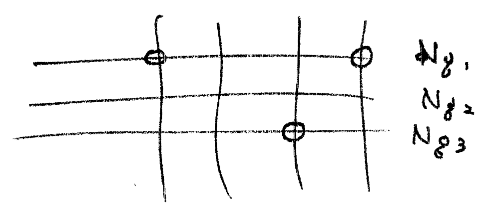
Trouble is — if $s_1 \times N_{t1}$ is of hy U and
$s_1 \times N_{t2}$ " " " " (say N_{t1} is
(L, C, R), N_{t2} is (Mass, springyness, friction)) then why should T.M. decide that N_{t1} is "close" to N_{t2}, in ~~this way~~ th. previous way of looking at things?

There are various plugs in th. 70 × n raster. Can I scan this raster for "similar" strs, as th. raster scans for ngms? Perhaps I can copy th. raster into th. array memory, and use ~~[illegible]~~ th. raster to scan its own plugging!

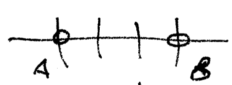
This idea would work ~~better~~ if I were thinking about NMTM (Non-Math T.M.), since th. goodness of th. raster pluggings is probablistic, and th. scanning method should be th. same scanning method that was used for probablistic problems.

How to scan th. plugging for a gn. str. First, on th. rasters, each plug of th. same str, will be ~ly labled.

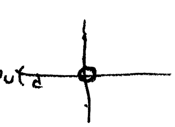
Consider th. str. pluggedas:



to detect this, thru scanning,

put a 

in every row, with a ckt to detect coincidence in th. lables of th. ~~pts~~ plugs scanned by A and B.

Similarly, put a  in every row. ~~[illegible]~~

"=" ← Boolean coincidence op.

now ~~[illegible]~~ th. outputs of all rows. ~~[illegible] but But coincidence~~

~~[illegible] expressed as "="~~

This isn't exactly rite, but by properly

Labling th. plugs, we may be able to scan for a gn. str.
— Th. ~ity of this method to possl. brain models isn't obvious.

Anyway, th. basic idea, is to use a rather simple model of neighboring strs and ntps, to get neighboring ngms. This would give a very elementary and un rich concept of "neighboring pngms" — but this "un rich" method, is used to scan th. ~~str~~ str and ntp. configs. to suggest new strs and ntps. I think that this <u>may</u> ↑ th. richness sufficiently (until sets of sets are added in).

An H-drum would be fairly good. A typical one contains 100 k bits and some run at ~ 500,000 bits/sec.
— So one can scan th. drum in $\frac{1}{5}$ sec. 10 drums contain 1M bits and can be scanned in 2 sec., if done sequencially. We want th. drum to also control th. coincidence ckt. configs.

One may want a <u>sorter</u> to be working continually, to keep th. abss. in order of U.

→ from 65)

A big trouble: strs, ntps and ngms aren't ever discarded, so th. memory becomes overloaded with them. However, after

imp

one has obtained a bunch of pugms, that ~~have~~ gotten their ~~Ups~~ $U_{\text{empirical}}$, one can discard all of the old scaffolding — i.e. discard all abs. (≠ pugms) of low U.

Def: ab's means an abs. that is not a pugm.

ab's = ngm, ntp or str. (ab's ≡ "secondary abs." pugm = primary abs.)

However — forget about these economy measures and return to th. report.

From 65) → a) ~~How does~~ Just how does T.M. go about fitting old abs. together to get new trial pugms? — i.e. Just what does it do, and in what order?

b) How does updating take place? — I think this is simple. Th. pugms are updated first. — Then th. ⟨strs and ntps.⟩ then th. ntps., Then th. new, unused strs. Th. description of this ~~updating~~ updating on ∝ 118 is O.K.

Note: Only pugms are scanned in th. array memory — not ngms. ngms may be derived from pugms, but their counts are never ⟨taken ~~seriously~~ because they are ngms⟩

.30 Woops: This bussiness of not scanning th. array memory until a new pugm has been created, is n.g. Suppose T.M. learned $= \begin{matrix}1&0&0&1&1\\0&0&0&1&1\end{matrix}$ very well, so it had th. pugms $\begin{matrix}1\\1\end{matrix}$ and $\begin{matrix}0\\0\end{matrix}$. When it was presented with $\sim \begin{matrix}1&1&0&0\\0&0&\blacksquare&\square\end{matrix}$, it would still use $\begin{matrix}0\\0\end{matrix}$ since this was appropriate, and had no counter cases, since it had never been tested after the "=" ~~ts~~ trg. seq. I think that th. moral is; One should scan thru th. array mem. after every ~~pugm~~ (q). element is gn. — for economy, one need not do this every time — in fact one may do it only when one has spare time, or when ~~one~~ has