Dart:

8402 – 467

Starts out with a reference to Plan 401.40 – T. idea of an arithmetic Trg. seq. for TM is gn. in rough outline in .

433 – 467 is reviewed in Dart

534 – 542, w. tales. and genl. discn. on 546–558 | 537.30 is a summary w. refs.

is on arith & learning T.M.

8402 – 422 is concerned w. t. trg. seq. — what sorts of things to include — like arith evaln., solving linear eqs., two 2 eqs in 2 unks, etc.

8423 – 432 is genl. epist. discn.

8433 begins to get into notation for grammars of arith evaln.

Sun Sep 10, 61

Part. (402)

Plans 401.40 spec
.01 393.40 → What I want to do here, is write

form, a (Backwards but ≠) trip. seq. corresp. to the

Plans 399.30 — 400.10 ← i.e. it starts with arith. — works up to

→ linear equs. in 5 unknowns.

There are 2 big. concepts that I can work toward:

1) to jump from 1, 2, and 3 unknowns, to 5.

2) just to jump from 3 to 4 " .

First do 2) — which is simpler. Th. idea is to reduce

4 eq. in 4 unknowns to 3 in 3 unknowns — i.e. to a prob. of

known soln. In 1) we can't use such a simple concept,

a somewhat more complex concept must be used.

Def ~~When we speak of~~ When we speak of M_i, in Th. forgoing

folg., we will mean Th. pgm. that ~~which~~ is able to solve Th. trip.

seq. being discussed.

→ O.K. So work on consider Th. trip. seq. that ends with Th. ability

to solve 1,2,3 linear equs. in ~~1,2,3~~ 1,2,3 unknowns, resp.

1) There are 2 simple possl. ways to reduce 3 eqs in 3 u

to 2 in 2: i.e. ① solve for x and / substitute ② subtraction (whi

is not too ~~big~~ difrnt from subtth.).

Say we use ① (subsn), since it is more general.

— later we will consider just how M_i can "know" both ways

SN Th. Q of just how M_i would handle a set of indetermino

equs., is interesting!

So — M_i solves one of Th. 3 eqs. for x (one of Th. 3 unknowns),

This yields a set of 4 equs. — 2 of which are equivt. Then 2 of

Th. 4 equs. are modified by Th. subtrn. At this point M_i looks

403.01

Part 2

.01 402.01 : at th. set of 2 eqs in 2 unks, and rea
can solve th.s, for y and z. This works part
Then substituting y and z in one of th. eqns containing
results in a solvable eq. for x. Th.s ∴ solves th.
Tho there are a lot of very rupt. "loose ends" here — go o
"next" (≡ previous) step.

2) 2 linear eqs in 2 unk: [illegible] eq.
solve 1 eq. for x and subs. in other eq. Then solve i
for y. Then subs. y in one of original eqns and so
for x.

.18 3) 1 linear eq., 1 unknown: These look like $\begin{cases} 3+4+2x+1+3 \\ x+\blacksquare+3x+5+5 \end{cases}$

4) " " e.g. $4 + 10x = \blacksquare 8x + 3$

5) " " $5x = 23$

SN How to "show" TM how to do things. If TM is "open" completely "open" to th. o
closely, one can look at TM's abss., and cons
th. solu. to th. problem of interest, using (of these abs
a "minimal" or
Another, more natural way: If one wants to show TM
how to work a problem in several steps, one simply gi
TM th. ordered seq. of steps, then TM ∈ will himself
Mt. C. will find seqns. that yield th.s seq. of steps. Since
th. steps are much closer together than (th. entire problem is
from th. previous state of M₂ >, TM will find a solu. to a
problem far more quickly, if he is "shown" how to work it.
This corresponds to some extent to what a human's

404.01

Dart

.01:40340 : reaction to "being shown" — i.e. if Th. st
Th. human will "get it" ~~xxxx~~ quickly — if not, it
more time (if it is ever solved at all!). NOTE: Th

Lets Go back to 3) [≡ 403.18] : Suppose we have Th
of "truth" for eqns. Every eq. gn. to TM (in Th
is considered "true" (for that prob.). Then TM devel

↳ Thp. seq. can, in this way, be viewed as a way of "sho
TM. how to work th. diffl prob. at Th. end of Th. thp
— there are, however, differences, in that in Th. thp seq. th
search statistics change during Th. thp. seq. — This is
true of Th. normal "showing" situation.

↳ ways to combine "true" eqns to yield more true eqns
Th. goal is reached when a true eq. of Th form $x = 18$ is obtain
Perhaps one would like to teach TM things by "showing"
many examples of legal xfms of eqns.

In fact, perhaps we can have an alternate form of
input to TM, in which TM is given some info in Th.
form of true eqns., and ~~xxxxxxxxxxxxxxxxxxxx~~ an eq
or set of eqs that are derivable from Th. gn. set. —
and TM must somehow furnish a set of intermediate
"true" eqs. leading to Th. "derivable" set, ~~xxxx~~ - perhaps
TM telling how he derived each step.

No — furnishing Th. "intermediate steps" is a rather ambiguous
problem. What Th. steps are, depends on what abss thms TM has made.
He may ~~decide that~~ be able to get th. desired eqns. from

405.01

Sun Sep 10, 61

Dart

01:404.40: Th. gn. ones in a single step!

Perhaps we could ask that TM construct some sort of

op. leading from th. gn. to th. desired eqns. — But just

would TM then trust such an op? How much ~~ ~~ [illegible]

wt. would its nums get in determining total pcost of thos[e]

No — I think th. easiest way is to ~~be~~ just give TM. [illegible]

and be able to "show" TM, possl. intermediate ste[ps]

to th. soln. of that prob. Just as with a human, [illegible]

intermediate steps should be as far appart as possl., and s[till]

enable TM to get them in reasonable time. We want TM [to]

do as much of th. work himself as possl., so he will d[evise]

his own abss., to fit in best with th. other abss. he has

devised.

Of course th. idea of "intermediate steps" in th. sol[n.]

of a prob. is a very el. idea (— not so el. — I think that there are alway[s]) and It would be well

if I could avoid ~~ ~~ forcing TM to work probs. that

way (— altho humans seem to work probs that way). Actually, th. co[ncept]

of "showing" seems to imply "intermediate steps".

~~Perhaps~~ Some of th. info learned from ~~about~~ th. tng. seq. (by T[M])

will be contained in M_i; other info in G_3. It isn't

clear to me as to whether this "dividing up" of th.

info is ~~ ~~ unambig.

There is another kind of ~~Q to~~ Q we mite ask TM

Gn. $x = y + 3$; $w + z = v + 3$; Does $x + w + z = y + v + 5$?

What ~~ ~~ must be done now? — Well, I am a bit encourage[d]

that the present system allows "showing" so easily. Hvr., I

would like T.M. to be able to have th. folg. 406.01

Mon Sep 11, 61

Draft

.01 : 405.40 : "concepts":

1) Th. idea of truth of an eq. in a gn. prob

2) Th. idea that new true eqns. can be produced from old, in certain ways

3) that one type of problem is getting a "true" eq. is in a certain form. e.g. like $x = 17$.

4) I also want TM to be able to learn (I,O) pairs of any kind — not only math.

SW Th. work on Th. typ. solns. thus far, has assumed TM can already recognize what Th. prob. is, and is only trying to find an op. that xfms Th. input data into Th. desired soln.

I still want Th. principal work of TM to consist of xfmg. input strings into output strings. Th. "concepts" discussed above, need not be taught directly, but may be taught by giving many difrnt types of probs in which such concepts would be useful. In general, I will expect TM, at first, to need much more typ. for a gn concept. than would a human, since Th. human has many relevant built-in concepts, as well as concepts from non-math probs. that carry over to some extent, to math probs. e.g. Th. idea of "true s's", Th. idea of "equality", etc.

.35 For Th. problems I plan (at first), to give TM, Th. final soln. (for M_3) will always be Th. same: i.e. to xfm. Th. "gn." eqns. by permissable xfmns, until an eq. of Th. form $x = 17$ is obtained. This will be true until we give TM 2 eqns in 2 unknowns. For eqns in 1 unknown, L_3

407.01

Mon Sep 11, 61

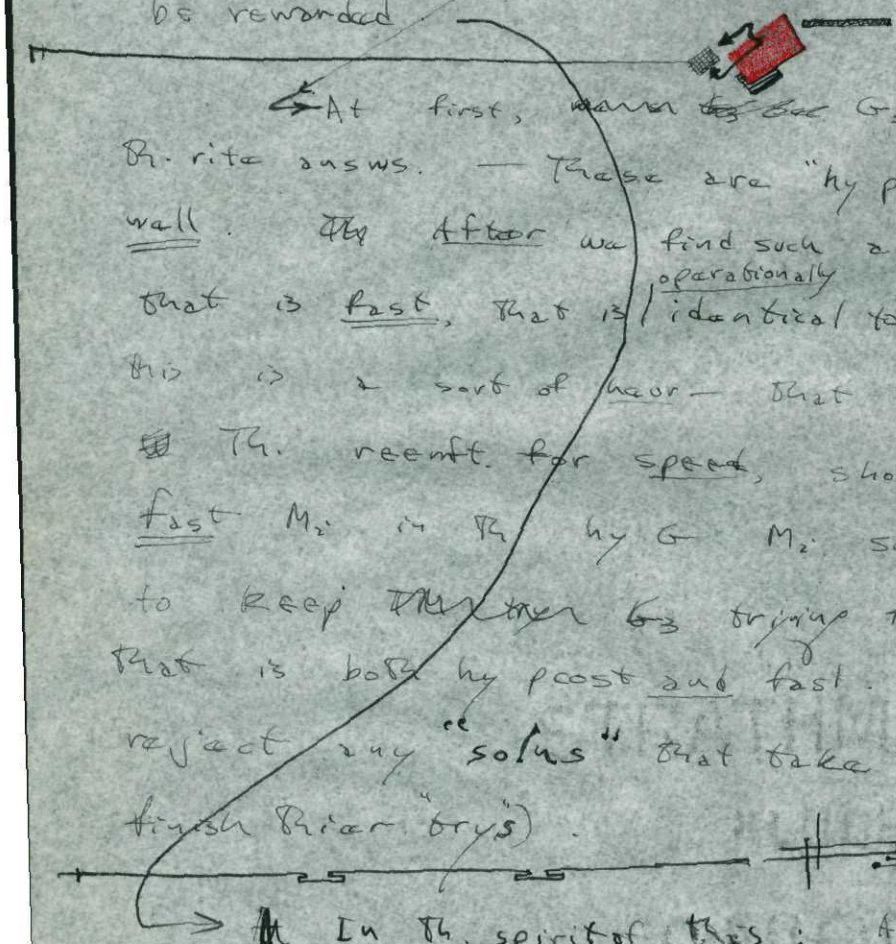
Part

.01:406.40 need not search at all! It is M_i that does

and we would like an M_i that does this most rapidly

A way to teach TM: At first, reward TM for answers that are not nearly correct, but that show that TM might have a desired concept. This corresponds to rewarding for a good motor vocabulary phrase — even tho this is not nearly relevant to the situation.

E.g. if we give TM the problem $X+2 = 7+1$ and ask for X, TM may say $X+2 = 8$ — which should be rewarded. — or $X+2+3 = 7+1+3$ — which should be rewarded.



At first, G_3 searches for M_i's that give right answers. — These are "hypcost" M_i's that extrapolate well. After we find such a soln., we try to find a soln that is fast, that is operationally identical to the hypcost soln. Actually this is a sort of hack — that might be learned by TM. The reenft. for speed, should result in a large no. of fast M_i in the hy G M_i set. What we might do, is to keep the trace G_3 trying for M_i's — until one is found that is both hypcost and fast. (Actually, we will automatically reject any "solns" that take too long — even before they finish their "trys").

In the spirit of this: At first, give TM probs like

.37 $1+3$; 9×3 ; $(4+8)\times1$; $(3(7+13))\div2$, etc. Note that this is the entire input fn. to TM for these probs. The correct "solns" are 4, 27, 32, etc.

408.01

Mon Sep 11, 61

Part

407.40 : Later probs are of Rl. form $X+3=2X$ and $X+3=3y$, $2y+1=2x+2$. In these cases prob must figure out what is to be done, and do it. After work a lot of probs like this, TM has a bunch of operators that are sort of near minimal. Each operator is assoc. with particular problem, and TM doesn't know how to recognize which op. to use for which problem.

Trouble with this, is that solns. of more complex probs., involve both recogn. of sub. probs. as well as use of old ops. This means that TM should be able to recognize as well as "work" Rl. arith. probs. of 407.37, before going on to probs like: $2X+3=3+17+X$, etc. It means that Th. more complex ops are comp. of both ops and obs of hypoth.

So, my first idea on Th. Tyg. Seq:

.22 1) First probs like 407.37. Give enuf untill TM can evaluate any arith. expression in ordinary alg. notation, that uses $+, -, \times, \div, (,)$

2) We write [if it proves instructive] teach TM Rl. same thing as 1, but use Polish notation — first with parenths. — than w.o. parenths.

3) $X=2$; $2X=5$; $X+2X=7$; $2X+1=3$; $X+3=2X$; $X+3=2X$; $X=2(1+3)+5$; $2X+2.3X+8X=2$; $2(2X+3)=7$; $(X+3)\div(2X+5)=8$

One of Rl. concepts I would like TM to learn as soon as possl.: a) recognition of a problem type that has been solved. b) Using "legal xfmns" on problems c) seeing if Rl. "legal" xfmns result in solved previously solved prob. types. ||| Unfortly, this

409.01

Mon Sept 11, 61

Part

.01: 408.40: results in TM's being able to take only one
at a time — i.e. a problem type must be run in Th. typ.
each step of complexity. Each "step of complexity"
to "one permissable xform. set."

To get beyond this "one step" barrier; One
is to get a closeness "goodness criterion" to tell whether or
,10 his brought an expression a prob. closer to solu. or not.

Th. difty: I could just use th. typ. seq of 408.22
on into 3 eqns in 3 unknowns. Probably I could find a
seq. of solu. for th.3 typ. seq.

Hrr., on th. other hand, I am much interested in
sure that T.M. works these problems pretty much th. way I
and: I want TM to "have" certain concepts — wh
learned or "wired in".

I think th. approach should be this:

1) In th. past, I had been excessively concerned wi
th. exact format of problem presentation. Don't!
Instead, Use th. more abbreviated presentations of 407.37 or 408.22 ff. Let
TM figure out what its all about. If its dift. for TM, wel
fine, this may be th. sort of difty I'm interested in having
TM deal with. If discovered to be not worth-while difty,
then and only then, change th. presentation format.

Also, note that M_i can be a time-varying operator.
Th. techniques used earlier in th. typ. seq. need not be
th. same as those used later. i.e. Th. identification
of a prob., may .um be a difrnt op., later in th. typ. seq.

4.10.01

Now Sept 17, 80

Dart

.01 : 409.40 : This means that th. apparently is no prob. to TM later in th. trg. seq., may have a soln. e.g. if "1+2" is presented early in th. seq., th. answ. is "3". If presented later in th. seq., this input may be meaningless.

2) Use th. "working backwards" technique — sticking to Explicit as long as poss!. This really is th. best method. Just work th. most diff't probs. th. way I seem to work them, then list th. concepts that seem to be needed, and write a trg. seq. for each concept.

.18

→ 412.37

O.K. suppose th. trg. seq. actually is 407.37 and 408.22ff

Let us see what mean th. solns. look like, and what must be added to th. trg. seq.

Consider th. problem $x+3 = 2x+5$

First, get all literal things on left side, all nos. on

$x + 3 + 2x = 5$

then

$x + 2x = 5+3$

Group terms

$x+2x = 8$

$3x = 8$

divide

$x = 8 \div 3$.

$x = \frac{8}{3}$

Now: all of th. above could be taught by using a trg. seq. consisting of 1 x fmn. step per problem, and th. ability to recognize a "solved problem type". TM would then only have a 1 step x fmn to learn for each prob. in th. seq.

Also, here, th. recognition problem would have to be solved.

411.01

Mon Sep 11, 61

Dorf

.01: 40.40 Would it be worth while to do this typ. say jump detail — with Th. 1 step xfmn limitation? An id to be solved would be Th. "recognition" problem.

If I did, perhaps I could formulate my criter so good improvement better.

SN Th. idea of having a non-stoch creative grammar given, and required to find an as of Th. gramm. that satisfies co conds., appears to be a common prob. E.G. ① M.T. of (if a MTPSG exists) can be put in this form. ② Also, Th. sol of alg. eqs. ③ Also Th. proof of a math theorem (to som

These can be viewed as "pure" search problems"

Would it be worth while to try to formulate all TM probs. this form?

.25 For th. "1 step xfmn"-type probs, a soln. of Th. folp. simple type, will work: say α is an op that will work as an M_2 for all probs. up to (but not including) Th. present say β is is an "ob". Its output is 1 for "problems of Th. now (present) type", 0 otherwise. γ is an op. that a xfmn. op. that changes a prob. of "Th. now type" into one of Th. old type ($\equiv$ covered by α). Then th. soln. to th. entire prob. (= probs of both nowm and old type) is:

$\beta(\gamma, I) \cdot \alpha$ | This means: "Apply β to th. input: if th. result is 1, do γ to th. input; if th. result is 0, do Identity to th. input). (Apply α to th. result).

412.01

Tu Sep 12, 61

Dart

.01 : 411.40 : In a ~~hyper~~ ~~grammar~~ dern method ... ~~...~~ it should be noted that th. cont... of a seq., in a manner employing a new defn., ... be of hy pcost. E.g. say th. ~~...~~ sub-seq. ... has appeared only once in th. corpus. ~~Th.~~ Say th. ... is repres. by α. Then ~~...~~ if we define $\beta \equiv abacdf$, we can perhaps write $\alpha\beta$ at low pcost — β will shorten th. dern. of α — (tho we must ↑ th. total dern by defining β) ~~...~~

In th. dern. of th. M_i, this sort of continuation of a dern. may become very impt. — e.g. we may want to simply repeat th. entire previous dern.

Th. 1 step xfmn "soln" of 411.25 will work o.k. — but of much interest, is th. hyer order heur. involved: i.e. of searching thru seqs. of hy pcost, 'permissable' xfms, till one is found that reduces th. gn. unfamiliar prob., to a familiar one. This can be regarded as a regularity in th. [M_i] set that is noticed by G_3 — or it can somehow be made into a M_i that will extrapolate better than any of th. previous M_i...

This particular point looks very interesting, and would perhaps in itself justify doing this tng. seq. in more detail.

.37 : 410.18 I think that I am making too much a fuss over this business of constructing a tng. seq. ~~Th~~ A basic ~~...~~ approach that should work: (1) Construct any old tng. seq. that seems to lead to an interesting induction (2) construct

413.01

Tu Sep 12, 61

Dart

.01: 411.40: In a ~~hyper using~~ ~~Gramm. usage~~ dern. method ... it should be noted that th. con... of a seq., in a manner employing a new defn., ... be of hy pcost. E.g. say th. ~~...~~ sub-seq. ... has appeared only once in th. corpus. ~~The~~ Say th. ... is repres. by α. Then ~~...~~ if we define β = abacdf, we can / perhaps write $\alpha\beta$ at low pcost — β will shorten th. dern. of α — (tho we must ↑ th. total dern by defining β ...

In th. dern. of th. M_i, this sort of continuation of a dern. may become very impt. — e.g. we may want to simply repeat th. entire previous dern.

Th. 1 step xfmn "soln" of 411.25 will work O.K. — but of much interest, is th. hyer order heur. involved: i.e. of searching thru seqs. of hy pcost / "permissable" xfms, till one is found that reduces th. gm. unfamiliar prob., to a familiar one. This can be regarded as a regularity in th. [M_i] set that is noticed by G_3 — or it can somehow be made into a M_i that will extrapolate better than any of th. previous M_i.

This particular point looks very interesting, and would perhaps in itself justify doing this tng. seq. ~~a~~ in more detail.

.37 : 410.18 I think that I am making too much a fuss over this business of constructing a tng. seq. ~~Th~~ A basic ~~...~~ approach that should work: (1) Construct any old tng. seq. that seems to lead to an interesting induction. (2) construct

413.01

W Sep 13, 61 TM 8.
Part)

.01: 412.40: a Mi set that solves these probs — as possl., th. heurs that I seem to use. (3) think of other heurs and alternate solns. (i.e. that are of close to ~~max~~ max prost. (4) If necy, examples are needed to th. typ. seq. to make it at reasonable speed by Mt.C. methods. (5) Th. basic

.10: 409.10: Th. concept of "simpler" can be used in a method for eqns. that tries to "simplify" th. existing set eqs. To get th. "simplicity" concept needed, th. ideas of counting th. no. of terms and th. idea of > and < applied to th. no. of terms) + (or th. ideas of ↑, ↓ remain th. same) can be used.

↳ is to start with any old typ. seq., ~~then add on~~ and see how th. solns. look — then add in examples to th. typ. seq. to make more likely any particular soln. method that I can think of.

Th. final Q, of course, is how long it will take TM (after such a suitable typ. seq.) to solve a signif. induction prob.

A second hassle is with th. notation form: try to keep this close to English as long as possl. When more specificity is **necessary**, do th. necy formalism (perhaps to **Lisp**). Try out a reasonable formalism for a while. **If it turns out to be n.g.** — try another one.

Th. trouble is, I hate to get deeply involved in one of these probs, because after I do, I often have trouble "seeing th. forest because of th. trees" — i.e. I loose th. over-all-picture.

4.14.01

W Sep 13, 65 TM

Part

.01: 413.40 : A trouble ~~thus~~ has been, that the writ[...]
have not been terribly effective in ~~[illegible]~~ elimi[...]
At the present time, my spirit is about at the point of [...]
— Tho now, of course, I know much more about codi[...]

Some continuing ideas on the all-over picture [...]
to how this MTM will work: (for previous review in this [...]

Inputs will be fed in: TM will try to find an M_i [...]
gives the rite outputs for all inputs up to that time. Th[...]
will be done by Mt. C. trials, using as good codi[...]
as I can get. A hyer order TM will try
(a) to simplify the existing "best" M_i — to reduce [...]
(b) to try to ~~[illegible]~~ find ~~[illegible]~~ (say) relationships betw. successi[...]
M_i's, so as to ~~[illegible]~~ assign them probs on this basis
and : ~~[illegible]~~ order all strings problistically as
M_i trials. The better this is done, the ~~[illegible]~~ fewer no.
of trials that are nec'y to get a M_i that "fits".

There will always be ~~a~~ some selection for M_i's that
work probs. fast. This is true because if a M_i that
is being tested, takes > a certain # ↑ to work any prob
then that M_i will be rejected.

O.K. : start: A soln. to ~~the~~ prob 1) of 408.22:
I will ~~[illegible]~~ use the op, ob. formalism: The input will be subje[...]
to various xfmns, that eventually yield the output.
So: ~~[illegible]~~ the input is an expressn like:
$(1+3.2) \div ((1+3.2)\times 8.1 + 14)$.

first, scan thru the input for any nums like
$\begin{pmatrix} - \\ + \\ (\end{pmatrix} 1 \; \begin{matrix} - \\ + \\ * \end{matrix} \; 3.2$

W Sept 13, 61

Part

.01: 414.40: or perhaps better — first remove all paren. This is done by removing any innermost paren. pair, by doing again and again until its impossl.

So, the prob. is to eval. unparenthd expressions.

They are of the forms (3×5 ; 3÷5 ; 1+3−8+2×3−1.5+.8×1×6−2×5)

Let us adopt the notation convention that that all mult. division must use parenths. So a permissable expression is

1 + (8×(3×5))

No! Only ÷ must be parenthd — but × must be indicated by "×".

Clear up this notation business — perhaps use parenthd polish — or perhaps some simple convention.

Say 3÷5 is ok. but 3÷5×7 is illegal.

Say We use the Fortran notation, with / instead of ÷ and the same convention of I'll let

4×3/5 and 3/5×4 both be illegal. — So only parenths. or integer (w. or w.o. signs) may appear either side of "/".

So: first ① isolate the first "(" & the first pair following it is ")". ② Then "simplify" this expression between "(" and ")" pair. ③ To "simplify" an expression: ⓐ remove all of the "×" signs ⓑ remove all of the "/" signs ⓒ perform indicated additions sequentially. [It may be of hyper prosthesis for ⓒ to remove all − signs, then remove all + signs, since

4/6.01

W Sept 13, 61

Part 5

.01: 415.40 This "remove all" idea is a common one (

Then do ① and ② again and again, until ... are gone, Then do ② on Rh. resultant expressn

It would seem that Lisp notation would be good for ex... these ops.

Th. idea of removing an "inside paren" — is diff. to express compactly, in terms of any symbolism I can think of. Some possl. ways. 1) a): Consider all compact substrings of input — there are about $\frac{n^2}{2}$ of them for an input string of n sym... Consider Rh. subset that start with "(", end with ")" ... Consider Rh. subset of Rh. latter that contain only 1 "(" ...

Th. idea of looking for substrings (in the 2 sub-corp... containing all having certain properties, and xfmg. this substr... and then substituting back Rh. xfmd. form, seems like a not too unusual operation.

Consider another method of solving Rh. "evaluation of an arith...

1) Replace all Replace ngms of Rh. form 3×4 by their "value" ... continue doing this until it is no longer possl.

2) Replace ngm of Rh. form $3/4$ by its value ... continue until impossl.

[Note: At this point, all "inside" parens will be of Rh. form $1+3+5-2+6-1$]

3) Replace Rh. first ngm of Rh. form $2(\overline{+})3$ by its value. ... continue till impossl.

4) replace all ngms like $(+3)$ by $+3$.

5) Go to 1) unless th. expressn remaining is of Rh. form -18.1 in which case, stop.

417.01

W Sep 13, 61

Part)

.01 : 416.40 : This last method is close to the way I

A slite modification:

1) replace the upm of the form $3 \overset{/}{\underset{\times}{\pm}} 8$ / by its

~~evaluation~~ repeat till impossl.

2) replace all upms like (+3) by +3.

repeat untill impossl.

3) If expressn is of form +18, stop

If not, go to 1).

Or:

1) replace expressns like $3 \overset{/}{\underset{\times}{\pm}} 8$ or (+3) that is

last by its value.

2) repeat till impossl, then stop.

or: 1) replace any upm of the form $3 \overset{/}{\underset{\times}{\pm}} 8$ or (+3)

its value. Repeat untill impossl. / — then stop.
i.e. no more such upms.

.23 This / latter wouldn't work: e.g. $3 + 8 \times 5 + 2$ in we would not

replace $3+8$ by 24. "x" has precidence.

So 1) replace (+3) by value. : repeat till impossl

2) " 3×4 " " " " " " "

3) " 3 ∓ 4 " " " " " "

.33 4) Go to 1) unless expressn is like +8 ; in which case

If, in this part of the pg, IM views the probs

being "simplifications" — then the part about solving eqns

for unknowns will be more natural. In / with expressns,

the legal xforms are almost all the same as those that
evaluate

are legal / xforms used in eq. solving.
simplificata.

HB.01

Th Sept 14, 61

Dart

.01 ; 417.40 : th. legal xforms in arith evaln :

.02 $A\left(\begin{smallmatrix}+\\-\end{smallmatrix}\right)B\,b \rightarrow a\left[\begin{smallmatrix}\text{evaln}\\ A(\pm)B\end{smallmatrix}\right]b$ if / a ... or / stated positive / a and b ... "t or

.03 $A(x)b \rightarrow *\left[\begin{smallmatrix}\text{evaln}\\ A\not{x}B\end{smallmatrix}\right]$ in all cases.

.04 $(A) \rightarrow \left[\begin{smallmatrix}\text{evaln}\\ A\end{smallmatrix}\right]$ " " "

These kinds of x fmns are, I believe, useable to th. most general lang. possl. — i.e. th. full Turmac lang.

[th. A form of th. full Turmac lang : $M(N\ D) = S_N$ ...

N is any integer, D is a descn. of Th. lang., and S_N is N th S in Th. lang. Th. Th. s's are denumerable, one not be able to tell if a pu. string is in ... in a ... no. of steps]

For simplification of linear alg. eqs in 1 unknown :

.25 $A\,z\left(\begin{smallmatrix}+\\-\end{smallmatrix}\right)B\,z \rightarrow \left[\begin{smallmatrix}\text{evaln}\\ A(\pm)B\end{smallmatrix}\right] x\, z$;

.26 $\left(\begin{smallmatrix}+\\-\end{smallmatrix}\right)A\left(\begin{smallmatrix}+\\-\end{smallmatrix}\right)z \rightarrow \left(\begin{smallmatrix}+\\-\end{smallmatrix}\right)z\left(\begin{smallmatrix}+\\-\end{smallmatrix}\right)A$;

.27 $A \times z = B \rightarrow z = \left[\begin{smallmatrix}\text{evaln}\\ B/A\end{smallmatrix}\right]$; $B = A \times z$

.28 $z \times A = B \rightarrow$ " " " ; $B = z \times A$

.29

So — to evaluate an algth. expn : do .02, .03, .04 none are possl. any more.

To solve an eq. : first $\gamma(\alpha)$ do .02, .03, .04 until no longer possl.

(β) Do .25 until no longer possl.

(γ) Do .26 " " " " then go to α

(Δ) Do α, β, γ loop until nothing is possl. — then go to .27 and .28 then stop

Th. Sep 14, 61.

→TMS←

Draft

.01: 418.40: Actually, for solving eqs. this wouldn't ... we would get stuck in Rh. form $A + B \times Z = \ldots$

We can get rid of this form by moving all ... to Rh. fits and all "unknown" types to Rh. left:

e.g. $A + B \times Z \rightarrow B \times Z + A$;

$(\mp)C = \rightarrow = (\mp)C$

$= C \times Z \rightarrow -C \times Z =$

Also, we can elim Rh. nacty of 3 of Rh. xfmns of 418.

by using $Z \times A \rightarrow A \times Z$

and just $A \times Z = B \Rightarrow Z = \left[\begin{smallmatrix} \text{evlu} \\ B/A \end{smallmatrix}\right]$

also we then wouldn't need $= Z \times C \rightarrow$

xfm 418.26 need not then, be augmented by xfmn: $Z(\mp)A \rightarrow (\mp)A + Z$

These sets of xfmns can be probably easily written as a grammar. Note that Rh. only context dep. subs. are 418.02. Also, it seems that Rh xfmns are not sequ... ordered, other than by Rh. context dependence and Rh. impossibility of using certain xfmns before others have suitably altered Rh. corpus

As it is now, it appears that we can do arith. eval. and lin. Eqs in 1 unkn. by another subs. xfmn — with one of them being context dependent. We can do it w.o. cont. dep., if we use some ordering and loops for Rh. xfmns, as 417.23-.33.

420.01

Th. Sep 14, 61

Dartmouth

.01: 419.40: Well — the problem of writing out the rules is somewhat like writing out a PSG. There probs. there, if one really wants anything near optimum. I hope that I will be able to tolerate non-optimum at this point.

The xfmns used all "improve" the corpus, in the sense doing something to it that certainly gets it closer to the desired. If there are no "dead ends" — then they must eventually the probs.

Well — now consider 2 eqns in 2 unknowns.

Suppose we only try to solve ones of the form

$$A + B\times Z + C\times Y = D + E\times Z + F\times Y$$
$$G + H\times Z + J\times Y = K + L\times Z + M\times Y$$

with perhaps some repeat of occurance of the 3 forms like $+3$, $-3\times Z$, etc.

By means say we get these 2 eqns with the punctuation Ω between them.

By means of xfmns of the type previously discussed, we can get the input into the form:

$$A\times Z + B\times Y = C \;\Omega\; D\times Z + E\times Y = F.$$

We could also get the form

$$A + B\,Z = Y \;\Omega\; Y = C + D\,Z$$

By the xfmn $Y = Y \;\Omega\; Y = \rightarrow =$ we can get

$$A + B\,Z = C + D\,Z$$

and then solve for Z in the usual

Then, the method of solving for Y isn't clear.

While T is does solve for Z and I might be able to work out a correspondence method to solve for Y, it all seems a bit too far from my intuitive method of dealing with

421.01

Th Sep 14, 61

Dart

.01: 4 20.40 these probs. Parts

.02 → A step in a slightly more intuitive direction would be certain temporary (i.e. for doing this prob.) changes to th. grammar — like th. addition (temporarily) of th. rule: $Y \rightarrow C + DZ$.

Th. xfmn. method outlined in th. foregoing sentence is distant from th. way I work with eval. and linear eqs. — However, ideas of (a) substitution (b) subtraction; seem more difficult to express in this f

I might modify th. formalism, so that th. concept of "equality" in sense of legal substitutability, is more natural — as .02.

Th. idea of a set of unordered eqs. as being an unordered set of facts, doesn't fit in so easily with th. formalism.

Also, th. idea of "solving for x" — as meaning "find a value z → when it is subs. for x in this eq., th. result is a "tr arith. eq.

One approach would be to preserve th. solns. to th. arith eval. and 1 unkn. eq. soln. in th. present gen'l form, but to view th. problem as one of (a) finding permissable xfmns (b) decidi on which xfmns to use in what order.

I think that I want something closer to what I would write (in Eng.) as a descn of th. intuitive method. Try a more direct descn. of th. intuitive method, for arith, and eqns in 1 and more unknowns.

Consider th. folg. type of soln to these probs:

1) look at th. input. Decide if th. prob. is (a) arith. eval. (b) 1 lin. eq. in 1 unkn. (c) 2 lin. eq. in 2 unkn. (d) n lin. eq. in n unkn. with $n > 2$.

Th is th. "ob." part. We may be able to again categorize (a) and (b) under 1 type, since we can

Fri Sep 15, 61

Part (422)

.01: 421.40. Then use something like #418.29 - .40 (a single op.) to solve both probs ⓐ and ⓑ.

As far as Th. typ. seq. soln. is concerned, when ⓐ and ⓑ have been gn. to TM, a single ob. will used for them. When ⓒ is gn., there is some that ⓐ will get one ob. with ⓑ and ⓒ another. When ⓓ is gn., then ⓑ is clearly a special case ⓓ, so ⓐ will have 1 ob., and ⓑ, ⓒ, ⓓ another.

Th. trouble with the actual formalism used in 418.29-.40, is it is peculiarly good for certain types of probs - not really all. May, hwr., be useful enuf to warrant a special ("free") dcrn. of it.

It is fairly easy to tell if a prob. is of type ⓐ, ⓑ by simply counting the no. of difrnt. "unknown" types (like z, y — So this particular ob. is fairly simple, for that particular univ. of

To solve ⓒ (≡ 2 unk.) after ⓐ and ⓑ have been solv

1) Get eqs. in the form $AZ + BY = C$ & $DZ + EY = F$

.30 There is a Q. at this pt.: If TM really understands what is required, when presented with various eqs. to solve, then it should be able to try various soln. methods, since the criterion for the correct soln. is clear to it — and it knows when it has solved it. Into-wise, all sets of eqs. in n unk. have the same soln: see thru all possl. sets of n nos. The set (or sets) that fit

.35 is the "correct soln."

Hwr. the criterion for "correct soln." is not only that it really fit, but that the soln. take a "reasonable" time. With this constraint, the previously mentioned "soln" is u.f. — and would not even be tried, since the operations that make it up would not have been useful for other probs. → [hwr., save 423.30]

Fri Sep 15, 61

TM 8

Port

.01: 422.40: But anyway — TM. idea is something like this:
what is required in TM. prob. of solving eqs. is
exact "G" function for that univ. of probs. Ord.
it would seem that MTM would be better at finding this G funct. — perhaps with
trials are costly. In TM. present MTM in mind, trials
little or nothing.

Knowledge of "what is required" would seem to be useful
G_3. — i.e. / G_3 could find an acceptable M_i more q.
Using this info,
One way to do this would be to make a metric for TM
of xfmn" of TM. input — to tell how close it is to
form of soln.

A possl. form of soln. to "n eqs. in n unk." ... A set of
"legal" xfmns — none of which lose info. These are then applied
at random to TM. input, until a set of eqs. is obtained
constitute an "acceptable soln." to TM. prob. "Heuristics" are problem
constraints on TM. use of TM. xfmns, that reduce search time
.30 Heurs of this sort can also be used to make TM. exhaustive
search soln. of 422.30-.35.

.32 Note that there are at least 2 ways that TM can decide
when a proposed soln. to linear eqs. is acceptable: ① TM. soln. satisfies
TM. original eqs by substn. or ② TM. soln. was obtained by a sequence
of "legal" xfmns from TM. input ⓑ TM. soln. is of TM. "proper
.35 form" e.g. like [z = 3.1, w = 4.2, v = 2.]

These heurs. are added to TM. "raw" soln. to make what is
perhaps an inacceptable or "marginal" soln., into a better, perhaps
acceptable soln. So M_i exist with "solve" TM. probs., but
these M_i are poor because they are too slow. G_3 then finds
ways to improve these M_i, by noting statistical chars. of

424.01

Fri Sep 15, 61. 20. IMy

Darb.

01: 423.40 successful searches.

Note that if there are any "problistic" choices, in ... always take Th. most probable for Th. first try, next most ... Th. next try, etc.

Note that of Th. 2 types of solns (423.32-.35), wi... hours, become acceptable solns — only (2) would be acces... thru Th. presently contemplated typ. seq. This is because (... hours is o.k. for certain simple probs. — and hours ... be developed as Th. probs. in the typ. seq. become more diffl. ... not (w.o. hours) acceptable for even Th. simplest eq. soln. ... so it would never be reached — unless other/entirely different types of ... in Th. typ. seq. made this a reasonable thing to try.

Note that we are sort of abandoning MTM, since ... working on RTM, since we are considering "soln. time" ... an impt. part of "acceptability of soln.". If we get ha... MTM and nothing but a "soln. time" threshold, there would still, be an unacceptable motive for G_3 to try to find hours to ... soln. time ... Since this is always (even in MTM), an acceptable goal for G_3 ...

Th. action of G_3 required in "improving Th. M_i" here, is ... difft than I originally conceived. Formerly, G_3 looked ... successful M_i's and tried to find regularities in them ... their ordering, so that Th. M_i that would be ... successful for Th. next input, could be found most ... quickly. Here, Th. goal of G_3 is somewhat difft. It wants ... to modify Th. M_i so that they work past (and ∴ we expect, ... for) probs. more rapidly. It is expected by G_3 that Th. presently acceptable M_i will work for Th. next input, but G_3 just wants to ... make that M_i work better.

Another very impt. difference in Th. new ...

Fri Sep 15, 61

Darpa

.01 ! 424.40 requirement of G_3, is that its source of ... Before, it just looked at th. sequence of ... and, from their strs, induced good trials for ... Now, on th. contrary, G_3 must observe th. M_i's ... on th. past inputs, and from these observations, ... to speed up th. solns.

.10 A difnt (perhaps not as good) way to get G_3 to ... isn't as far from noting regularities in th. seq. of ... Let th. set of M_i's that all fit a pu. pt. in th. ... in order of speed of operation. Then, th. decay of th ... rule would involve studying th. sequential regularities in this last set of M_i's.

So, we could give each M_i 2 nos. — one tells how far in th. tng. seq. it got: th. other, how long it took to solve th. last prob (or set of probs) that it could work in th. tng. seq.

.20

REMEMBER: One goal, here, is to derive a lang. for ... M's solns. to probs. ⇒ any regularity that I can think of and any heur that I can think of, is fairly readily expr in th. system. There might be some point to my familiarizg my with Lisp. One disadvantage of Lisp is its slowness. ... advantage, is that ∃ many known useful ops. that have be derived for it. Another disadvantage of Lisp (that may or may not exist) is its extreme slowness in arithmetic.

So: I should write down various solns. to th. tng. seq. considering — write down all th. reasonable trial solns. that I can think of then try to devise a formalism that minimized their code ...

Hvr, this problem is still impt. It deals with 1 particular soln. to this tng. seq. — and its a fairly impt. type of soln. This soln. does need (as was noted) a RTM, since speed of soln. is considered.

A new point: that while (.10–.20) it is possl. to implement

Mon Sep 18, 61

Dart.

.01: 425.40 Such a soln. with an L_3 (i.e. a hyper order TM) in "normal way", it seems that th. type of ~~[illegible]~~ ~~recognizing th.~~ decrp. th. regularities in < a set of th. same prob. - in order of ↑ speed > ~~[illegible]~~ that I mind was somewhat difrt. than usual. Th. "usual"

Def looks at th. symbolic strings in α''' and tries to find regularities in this M_i set. Th. method I had in mind watched these M_i's work on problems, and noticed regular in th. successful searches. This / latter method is certainly a subclass th. "usual" method, but it is a fairly abstract / and ~~[illegible]~~ rather quite from ~~[illegible]~~ th. / simple direct ~~[illegible]~~ scanning for regularities in th. symbols of

Also, if L_3 was to detect this / kind of regularity by looking at α, L_3 have to retest α on something or other, to observe α in action th. examples upon which α is to be applied are not included th. decns of α, so this new method is essentially difrnt from th. "usual" method of L_3 operation — i.e. in th. "usual" ~~[illegible]~~ meth L_3 does not have TM's inputs / directly available to him.

Well - no - in RTM, th. L_3 does have RTM's inputs availa to him for testing trial M_3's.

.30 Consider th. 2 dim. analog of RTM: x is input, y out Say we know G as a funct of x, y. We want to find f(x) ① th. decn. of ~~[illegible]~~ F is hypcost ② F(x) will be large for x in th. near future. One way to do this is to take trial f_i's [corresp. to M_i's] that have had hy G's for re x's, and make th. new trial f_i "close" to them, decn-wise — Note, hvr., this idea just tells how we make preliminary trial f_i's before we use them, we test them on recent x's

Th. point is, we want a f. that will tend to be good for x's of th. near future, and ~~[illegible]~~ an easy (tho not th. best) way

427.01

Mon Sep 18, 61
Dart

.01 : 426.40 to do this, is to get a F that is maximally good for recent

So: In order to devise the best possl. F, we must, in addition to $G(x,y)$ [which we don't have to know, since we happened to in the present case], all of the past x's from this data, we can predict the next x's and evaluate any F, thereby. .10 . SEE 557.20 for am.impt. pt.

Actually, strictly speaking, L_3 should take the known x's and try them on the various F_i's on his own time. That is to watch while the trial F_i's are actually being used, is a sort of — It just about halves the necy. time. Probly L_3 wouldn't want watch all F_i's in action — just some of them that take for some probs, very little time.

Anyway, this is just one (and perhps and impt one) example of another type of regularity that L_3 will have to be able to recog. It is more like a regularity in the relation of α to the inputs than a reg. in α alone.

What we want, is a F_i that has an optimum relationship of to the set of x's and the function, $G(x,y)$. Watching F_i's, that is fairly good, in action on past x's, is a heur device by which F_i's of hy expected goodness can be found.

~~The trouble is — here's old L_3, trying to find regularities in α — now how could he ever think of trying to find regs. in α. Furthermore, how would he~~ Now — how could he think of

In a more el. way, say L_3 is trying to find a F_i that will max max $G(x,y)$ over the present x's. He might do this by looking at a certain bunch of F_i's and noting how the very good ones differ from the poor ones. This would be the "normal" method. The more recent idea is that L_3 doesn't just satisfy himself by knowing which of the past F_i's were "good" and which were "bad" — he wants to know "why" — and this is found out by watching the F_i's being evaluated.

428.01

Mon Sep 18, 61

TM 8

Dartmouth

.01: 427.40: So — a rather small extension of my previous [illegible]. Not only does L_3 know which F_i's are good and bad and how [illegible] L_3 can watch (as much as possible) Th. evaluation of Th. G [illegible] proposed F_i's. This eval. process usually (but not always) [illegible] of some heur. value in devising new F_i's of by expected G.

I think that I wanted to restrict L_3's [illegible] given F_i's and their $\bar{G}$'s, and being asked to make good [illegible] trial P_i's. A L_3 of that sort would be a pure? G and so we could almost do Th. $TM_1 = TM_2$ trick directly — except that here, Th. TM_1 was an operator TM.

Th. Q seems to be one of localizing Th. Learning Th. to either M_2 or L_3 or some hyper-hyper order TM. As yet, there seems to be no Q. of just how to evaluate how good a gn. M_i is. — This is simply a matter of its empirical effectiveness and its cost. Th. Q. seems to be one of how i.e. whether certain M_2's are "heuristically accessable" to L_3. What I would like would be some generalization of Th. operation L_3, so that it would be perfectly "natural" for L_3 to [illegible] Th. M_i's in action. It is certainly "legal" for L_3 to [illegible] at < Th. M_i's [illegible] that are all consist a gn. [illegible] Th. typ. seq., subs are arranged in order of speed).

.35 Th. idea is something like this: Say we saw a bunch of sea animals, ranked in order of "viability in Th. sea"; and we were to propose a new sea animal that would be more viable than any of them. Then it would seem possible to compute good trials simply by looking at Th. known animals only; But it would seem also to be of very much additional value in designing such an animal, if one

429.01

To Sep 19, 61

Dart.

.01: 428.40 also knew th. characteristics of th. envt. i[n] in. In fact, one good way one would extrapl[t] th. animals alone, would be to postulate an envt. (a set of envts) consi with th. known animals this envt. for helping / extrapln. → 431.10

Perhaps th. only reason I assumed that it was o.k. (in L_3 case) to leave out info about th. envt., is th[at] for large enuf. samples, th. envt. can be induced (presence / would) and is ∴ not absolutely necy. — tho it's accuracy available in extrapln. for a gen. samp. size — which certainly a worthy goal!

Even if we simplify th. goal of L_3, and define this goal to th. finding of a M_i that is fastest (and consi) for th. last 5 inputs, then it would seem best that L_3 know what the inputs are (tho they are inducable with large samp.), and "w[ork] them "interact" with past M_i trials.

This sort of narrows down th. relevant parts of th. problem th[at] I must work on. Th. 2 dim. analog 426.30 is useful at first but must be modified a bit, for this goal type

So: th. M_i are to "solve" th. input probs. in min. time

G_3 is a creative grammar. It is to order all possl. M_i's probabilistically, so that th. probty assigned to a M_i gives th. probty that that M_i will be both. (a) consi (b) of hy speed. — This is, needless to say, a rather vague probty! If we like, we can run th. output of G_3 thru a "filter" that removes all M_i that are inconsi — So (b) becomes th. main characteristic. We would also (like G_3 (with its "filter") to be fairly

.40

430.01

Th, Sep 21, 61

.01: 429.40 Fast ... in producing Cousi M_i's — G_3 should produce a by % of cousi M_i's.

Perhaps we could pin down Th. "probty" meaning of th. probtys in G_3 be Th. probty that that M_i will have a ... sp... Th. hyest speed thus far ... and be cousi>. Actually probly want a somewhat difrnt. meaning for this probty — but one will probly be good enof for a while.

Unfortly, → a known G_3 that can produce only M_i's that ... and faster than any previous M_i's. This G_3 ... simply mak... exhaustive search of M_i's in ... aprip order (so low prob... are first). Hvr., this G_3 is itself, too slow. For this ... arose in another context — ie. 429.40 ff. So we do wa... G_3 to be fairly fast (e.g. take ... on Th. average secs. to produce a cousi... as an auxy constraint on G_3.

Superficially, it would seem that G_3's job is to devise "good" as "rapidly as possl.", and G_3 could use any info he likes to accomp... this task. If G_3 finds that features of "Good" M_i's correlate in so... with some other objects in Th. env't (e.g. certain features on Th. input problems and Th. soln. operations.) than G_3 could u... this fact to help him create Good M_i's more rapidly.

There is perhaps another way to look at this — ie. say writes intermediate results on a sheet of paper. This paper and whats writ on it become new inputs to M_i. This is sort of the way humans operate.

→ Actually, G_3's using "tested" M_i's and their G's is part of this. Strictly speaking G_3 doesn't need ... into on... M_i's — he should be able to work with only Th. known inputs, and Th. known $G(I, O)$.

431.40

Th Sep 21, 61

TMJ!

Dart

.01: 430.40: We may look upon the set of Mi's (in solving probs) and their ... as all being usable info ... to ... how hi Mi trials at a rapid rate. In this sense we ... this as being "heur. info" — i.e. it ↑ "search speed"

.05 There will be later, another Q involved — of just what info is "worth storing" in rapid access storage — since ... has a limited amt. of such storage available, or more such storage ...

.10 Just as in th. case of designing sea animals (428.3...) th. postulation of an envt. (and a fortiori, th. "seeking out" of th. charstics of such an envt.) has in th. past, led to good predictions — so we use this technique now. In th. sea animal ... this means that we want th. "animal designer" to have available to him all possl. "relevant data" — i.e. info about anything that mite interact with what he may design. Now, this "info" could be gn. by a "complete scan" of th. expected envt. — but this would not be really all we want. We would also be able to ... info about related animals and objects, and how they in... with th. envt. Such info is not necy, (i.e. equally good results ... be achievable w.o. it) but such info can speeds up th. ... of a result of gn. goodness.

Again note (similar to .05) that by adding in too much "aux info", we may hide th. "very good" aux info" in "noise". We can avoid this by ordering th. aux info in order of "aprip. of relevance" (this would have to be defined more exactly).

.37 So: th. function of G_3: To take "all available info" that is likely to be relevant" and construct th. next M_i trial as a funct. of this data. — presumably in a Mt.C. way. Each G_3 trial

432.01

Th. Sept 21, 61

Dartm. (432)

.01: 431.40 Will then be "graded" on Th. basis of its speed by G M_i's. — So Th.s G of G_3 depends on with which it constructs trials, as well as upon Th. of Th. trials.

Similarly G_4 (which tries to make good G_3 trials) has to it, all possibly relevant info — including Th. ability of G_3 and Th. M_i's in action.

Th. discovery of part of th. M_i's G "function" (i.e. knowledge (in either sense of 423.32) what constitutes a "soln" to Th. prob. set of eqs.) is part of G_3's technique — i.e. G_3 uses Th.s G_4 discovers that it is useful to try to find a G funct. for Th.

.20 It would be possl., at first, to stick to a rather pure Th. by using a suitable typ seq. Such a study would be somewhat instructive, and would avoid Th. diffy of G_3 having to observe th. M_i's in action. However, my present view is I'd rather not use Th.s simplification, but direct myself to some Th.s (e.g. el. prob. of a RTM with typ seq.).

As of now, Th. typ. seq. consists of 1) arith. expressns. (evaln. of +, −, /, ×,) and (. 2) soln. of linear eqs. in 1, then 2, then 3, Th. 5 unknowns. [note ommission of 4. This makes a particularly interesting conceptual jump reqd.].

If I do want TM. to learn to do solving eqs. probs. by learning Th. certain xfmns are "legal" and then one must then devise methods to min. Th. no. of xfmns to get to Th. acceptable form — then this typ. seq. will require some rather impt. things to be incorporated into Th. entire TM — So a TM that can use this Typ seq. appears to be a "worthy goal".

F Sep 22, 61

433

Part

.01 | 432.40 For arith. and 1 eq. in 1 unk. Th. xfmns of ... mite be O.K.

A sort of objection to this: that it doesn't do a xfmn, then the result is of th. form of a prob. that has already been solved. What may do, is work out an alternate solution seq. to this typ. seq. this idea. Actually, I will want several difrnt typs of

For Arith evaln: Consider th. follo. possl. solns:

.14 (1) → Context dep. Subs. Using th. 3 unordered xfmns of 418.02 — .04.

.15 (2) → only 2 xfmn types: ⓐ $A\left(\frac{\times}{\div}\right)B$; ⓑ $A\left(\frac{-}{+}\right)B$; (A) ← this xfmn can be of type ⓐ or ⓑ; ← actual parenths.

we do ② , until impossl. than " " ⓑ " " than →

This is a kind of ordering.

(3) 435.01
(4) 435.18

.24 Th. idea here, is that we have to find some way to / and × precidence over + and —. Either way would do it.

Let's go on to 1 linear eq in 1 unk. ↑:

Some CDS (or sequencial constraints) here: i.e. in a $Axz\left(\frac{\times}{\div}\right)Bxb$: a and b cannot be

.20 $Axz\left(\frac{\times}{\div}\right)Bxz \rightarrow \left[\begin{matrix}\text{ev.}\\ A\ \frac{\times}{\div}\ B\end{matrix}\right] x\ z$

.31 $A \times z = B \rightarrow z = \left[\begin{matrix}\text{ev.}\\ B/A\end{matrix}\right]$ } This is cont. dep. — since we can do this xfmn if there is anything in th. eq. i.e. in $a\,Ax B = b$ both a and b must be nonl.

.32 $z \times A \rightarrow A \times z$

.33 $A + B \times z \rightarrow B \times z + A$

.34 $= + A \times z \rightarrow - A \times z =$

.35 $\mp A = \rightarrow = \mp A$

.36 $z \rightarrow 1 \times z$ ← I think 1 is one of th. initial vocab. symbols; where does this symbol come from.

Again we can make unnec this CDS by making this xfmn possl. only when all others have been tried and are impossl.

For th. 1 unk. case, we have 7 new xfms! — otherwise, things are th. same — we have th. same kind of mathy choice betw. CDS and a kind of ordering of xfms. The have, th. ordering seems simpler (i.e. no loop).

Sat Sep 23, 61 TM 8

Dart

.01 : 435.40 : It would seem that 3) and 4) would [illegible] solving N eqns in N unks. — Particularly 3) mite be [illegible] go from 3 directly to 5 or 6 unks., if it has developed proper "closeness" concept in G_3 (i.e. see 435.

Working out more details of these solns. (even more det. 1) and 2)) mite make it possl. to unify these 4 into a

.10 single formalism. 437.20

About R. "showing" bit: If TM works by observ. "type" of prob. each presentation is, and xfmg each p. into new presentations, until R. prob. is "solved" — th. it is possl. to "show" TM intermediate steps, so as speed of his search. Unless TM works this way, hwr I'm not sure that it is really meaningful to "show" anything.

Note, hwr, that this "successive xfmn." idea may [synchronous digital] as el. as it sounds — since all computers do have "successive states". These "successive states" can be R. entire computer or of a specific part of it — e.g. a certain part of R. "memory". Also, note that all 4 "solns." discussed thus far, are solns. in which it o. to be meaningful to "show" TM [illegible] intermediate st. Soln # 4) in which TM recognized "soluble" appears to be most amenable to "showing".

In particular, here → I would like a formalism → no matter which method (1, 2, 3 or 4) TM was using, th. "showing" of intermediate steps would have a proper interpretation.

In a real typ sep. I'd like to use R. "show" technq. as little as possl., because it narrows down

437.9

Sat Sep 23, 61 TM

Dan

.01: 436.40 TM's methods to what I think are th. rite [illegible]
(th. same thing can be said of a typ. seq. with [illegible]
steps in it). On th. other hand both "showing"
a typ. seq. with small steps, reduce th. no. of
that TM must make, tremendously.

Perhaps what should be done is to space th. steps
seq. or insertion of "showings" just close enuf so th.
th. search time for TM is "reasonable" but is long
is "practical".

So far, in th. typ. seq., I think that I can devise typ.
steps of arb. closeness, so as to simulate th. effects of a
"showing" of interm. steps.

.20: 436.10: Actually, it isn't hard to unify 3) and 4) somewhat
i.e. TM can recognize various "states of soln." (as in 4)) and learn
which xfmns are most likely to bring them closer to th. final
in 3)).

.25 In 1) and 2) th. concepts of xfmns are used. Th. re
op" (≡ ob.) used in 1) is th. context of an ngm.
This determines whether or not p.q. xfmn should be done or not
In 2), a certain type of ngm xfmn is possl. a) at least if
another certain type is not possl. or b) if th. last xfmn. th
has been of a certain type. So in 2) (≡433.15) ② is possl
if either ① th. last xfmn was of type ② or ② that two xfmns of
type ⓑ are not possl. ‖ ① is possl. ① if th. last xfmn was of type ⓑ
or if xfmns of type ⓐ are impossl. ‡

.35 Or, more simply: ⓐ xfmns are always possl. ‖ ② ⓑ xfmns are
possl. when ⓐ xfms are unavailable. available when

So, using .35, th. availability of a type ⓐ xfmn. is an ob.
If this ob. says "yes" then do ②. If it says "no" then do ⓑ.

.39 If, then, ⓑ is unavail, stop. This formalism would work for
arith, eval. I'm not sure about soln. of 1 unkn.

438.01

Sat Sep 23, 61 TMS

Dart

.01: 437.40: My impression is that, intuitively, soln 1) is not much used — but that 2) (noting that ... are in) is much used.

But in R. case of null AXZ = ... this seems ...

So, to solve eqs. in 1 unk: first do xfmns ⓐ and ⓑ on 437.35 - .39 — but when both ⓐ and ⓑ are unavail, do ⓒ consists of 433.30, .32, .33, .34, .35 and .36. (note .31 ommited) When ⓒ is imp... (we don't have to look to see if ⓐ and ⓑ are possl. now) we ... ⓓ, which is 433.31. and we are done.

So, using R. ideas of 437.25 ff we can make soln. type 2) a special case of soln. type 4) for both arith. eval. and 1 unkn.

Also, using R. idea of 437.20, we can unify soln types 3) ... 2.0 — so — fine!

.21 SN I think it might be wise to tell TM at R. outset that "Z", 34.18012 and "=" are 3 difnt. kinds of objects. (Any number not a ref.)

We may or may not want to tell him that (,) and = are difnt from +, −, /, ×. [Actually, TM should have a hour that tells him about "facts" upon sts — so that for a reasonably good TM, it should be unnecy to give him this info.]

One way to tell TM that Z is a particular kind of ob... is to make R. code name for Z, V, Y be 1,1; 1,2 and R. codes for =,), (could be 2,1; 2,2, 2,3 ||| R. co... for +, −, /, × could be 3,1; 3,2; 3,3; 3,4 etc.

So we can pre-categorize with our notation. TM must, of course, have access to both parts of our dcrn., or else we have to tell him anything!

.36

I think that R. pt. of R. present work is to see if I can adequately formalize all of R. intuitive ideas that I have on problem solving for a reasonable universe of problems. In particular, I want to see how much of these heurs must be allocated to

439.01

Sun Sep 2?, 61

Da...

01: 438.40: Varioos levels of TM (i.e. M_i or L_3 or hyer).
of much practical impt.) I want to see how much ...
is required between various "reasonably sized jumps" in Th. ...
If Th. jumps take too much power, then this could be be...

ⓐ I haven't really ~~appx~~ formalized all of Th. heurs that I use fo...

ⓑ " have ~~[struck]~~ " " " " " " "
but I haven't done it properly — i.e. certain parts of ...
of Th. heurs are of too low pcost, because I haven't derived ...
heurs that use similar sub. abss.

ⓒ Th. work has been done correctly, but it just requires an enor...
amt. of computing power to do any interesting learning.

ⓓ A form of ⓐ — i.e. I may have most of Th. nec heurs for some
problems, but not enuf for other — i.e. Th. probs. involving interesting extr... ~~...~~

This ↑ gives a fair (but perhaps still not exact enuf!) idea as to what "computing power" mite
mean. It is essentially, how many trials can be made/sec.
Since we presume that all heurs have been already used to reor...
Th. trials suitably, this raw "no. of trials/sec" is about Th. vit...
measure.

Unfortly, this seems to omit Th. Q. of memory size (and mem. s...
is impt., since one can save time by replacing commonly used computa...
by tables — *if* one ~~has~~ has enuf memory) and ∴ can't be entirely vita...

Also, in a ∥ network computer, where Th. memory is non-loca...
~~[struck]~~ — and, Th. "no. of trials^d/sec" may be difft. to compute.

→ It would seem that certain "partial computations" could be tabula...
in Th. mem., and other tricks to reduce mean time/trial.

Another trick modification of present day machines that may
speed things up: Have 10 or 100 arith. units working in ∥ on Th. same "large" slowly changing
memory, and ~~...~~ give each arith. unit its own small rapid access memory.
Th. *large* memory mite be a "fast read, slow write type" if such
are feasible.

Sun Sept 24, 61

Part I

TM

·01 439.40: It seems to me that usually (if not always) whenever a discovery has been made — either a small or a large jump — there has always been found a way of "reasoning" in which this induction is obvious, so that one "should" have tried it. ---- Sooo - If I could estimate the cost of these jumps for various inductions, I could see whether they were within the power of present day computers or near future ones.

The point is to show how each induction follows reasonably from the heuristic sequence. Also to show how each heuristic follows from the heuristic sequence) with reasonable cost.

What we have to do, is put each heuristic or regularity or coding in ab-initio — or devise a typical sequence. That would lead to its discovery.

So now: Suppose I do the typical sequence and the corresponding M_i sequences for arithmetic and I unknown. Then I could make an outline of a TM, that would be rather weak — ie. it could work these 2 prob. types and perhaps a little more. I will have to work a rather large no. of prob. types by hand, gaining trying to give all possible solns. using all heuristics I can think of, and the TM will begin to be able to "take off" a bit for itself, on probs. more distant from the hand given ones.

But even so — in the hand prompts I will begin to get some idea of the "computing capacity" that I might need for a really useful TM.

The Big Point, however, is that if I make this TM work on the typical sequence using only my own intuitive heuristics, then I should not have to discover any new regularities or methods of finding such regularities. The whole TM mechanism that will use will be a direct outgrowth of the heuristics that I, myself, use, and will need no other heuristic discovery methods. This means that almost all of the really creative work that I must henceforth do, will be discovering my own heuristics, for various typical sequences. - I.E. I shouldn't have to devise much in the way of new coding methods.

442.0

Mon Sep 25, 61

Dartt

.01 : 440.40 : Th. problems at present seem to be: (1)

(1) Devising a formalism (say like M_i, L_3, ... L_i) in which to express all of my intuitive, heuristic ideas in.

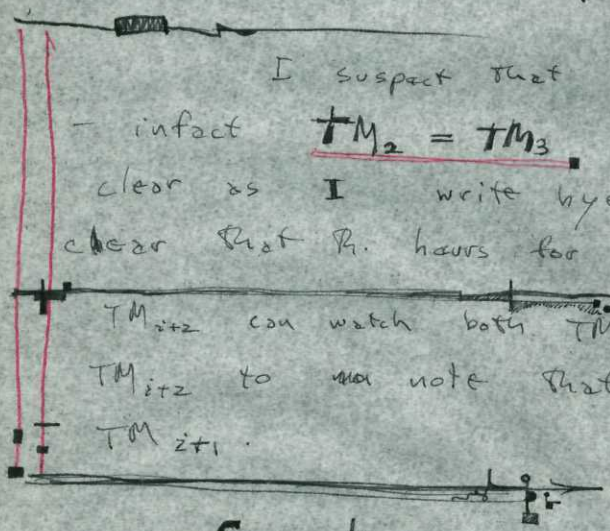
(2) Expressing as much of my heuristic ideas as possl. for seq. of interest.

(3) Deciding how to deal with problem types in which I really know in any detail, how I work these problems - e.g. of English, and probably most fairly difft induction problems.

I'm afraid that I mite be drifting too much in the simon-direction — with their fear of non-human methods of solving Th. big difference, perhaps, hrr, is that they don't really understand induction inferences quantitatively, and I do.

Th. usual way that I use to formalize these induction problems is to work one of them, then give heur. reasons why each step is a plausable trial, then try to find a Th. seq. so that reasons for plausibility (i.e. Th. heurs) could be learned from that Th. This work should be done largely in English at first.

I suspect that I will not need a very large i for TM_i — infact $TM_2 = TM_3$ (with $TM_2 \equiv L_3$, $TM_3 \equiv L_4$, say.) may be O.K. This will be clear as I write hyer and hyer order heurs — i.e. it will become clear that Th. heurs for TM_i should work equally well for TM_{i+1}. TM_{i+2} can watch both TM_i and TM_{i+1} in action, it is clearly large TM_{i+2} to note that TM_i's heurs can all (or almost all) be TM_{i+1}.



So: the problem now, is to write th. typ. seq. ruffly, then write out a soln. to it — mainly in English. Also write out Th. heurs (i.e. L_3), and Th. reasoning and typ seq. that led to these heurs, etc.

Mon Sep 25, 61 TMg.

Dart,

.01: 442.40! For each "state of education" of TM any prob. will have a certain pcost. ~~[illegible]~~ tell ≈ how long it will take to find th. soln. For pres computers, we will need a trg. seq. in which known to th. problems have a pcost of $> \sim 10^{-6}$. In a few y may be able to use pcost $\approx 10^{-9}$. This figure gives a betw. successive members of th. trg. seq. that we must no exceed.

On th. other hand, we want th. "spacing" betw. th. probs as large as possl., so that th. trg. seq. will have less "forcing" on TM's solns. ← i.e. th. solns. will be "better" and less conditioned by what th. trainer thinks are th. proper "elementary a Hvr., I'm not so sure this is good — we can't afford for TM get "off th. track." — Tho if he does, and gets too badly st we can always determine what abss. he didn't get, by ~~a~~ se how long he takes to solve certain test problems. — Then take h back several problems and give him a trg. seq with step that are closer together.

→ In a sense, hvr., anything that TM picks up "off track" is o.k. — he should be made to keep these abss. in addition to th. ones that are known by th. ~~[illegible]~~ trainer to be adequate for th. trg. seq. of in

There is th. Q of "wiring in" certain abss., as oppos TM's learning these things. ① Certain things would certainly TM too long to learn &/o would require a very large trg. seq. ②Also, TM tends to have things better organized if he learns them himself. I.e. he will use better sub-abss. ③ We are often unsure that TM has learned th. "proper" "on track" abss. if he learns things himself. It will take lots of troublesome back-tracking if he learns th. "wrong things". Hvr. → . ④ After TM has been

Mon Sep 25, 61

TMf

.01: 493.40: running satisfactorily for a while, it will be [illegible] to devise a "small-step trg seq." for TM, [illegible] "program in" certain desired abss. (order hours / at first it may take TM too long to get an adequate [illegible] make it possible for him to dcvr. these hours himself.

I'm not at all certain about how to formalize hyer order hours — so that all trial hours are simply [illegible] combinations of old parts of hours. My impression is that one regard all "successful" hours of a gn. order, as being in the corpus and then each trial addition to this corpus can then have a pcost evaltd. on this basis.

About the 4 solns: of #433, and 435:

1) C.D.S.: Is probably the easiest for me to write; requires few hours little work in L_3.

2) no C.D.S., but a kind of ordering of operations. Tho this soln be written as is, it is probly best do as a sub-type of soln (in 3), M_2 knows about permissable xfmns and goals but L_3 must speed up dcvry in 3) TM must also dcvr. what the soln. criteria are.

3) Involves a L_3 that must speed-up an otherwise O.K. soln. see 435.01 This soln. type could also do 2), and is perhaps a simple example of an L_3 that must watch all of the M_i's' operations.

4) the ob-op. formalism. (see 435.18) would involve [illegible] an L_3. Is perhaps very good for n ops in n unkns. To [illegible] from n to n+2 unks, hvr, probly must have an L_3.

Probly the best course of action: Write up all 4 solns, and attempt to unify them (437.20 - 438.20). Then see if they conform to my ideas about being able to put all of my heuristic ideas of 440 into this particular TM form (i.e. the M_i, L_3, etc. formalism). It will give me practice in application of hours in this form. In particular, my being able to

Mon Sept 25, 61

Dartf

.01 444.40 deal with th. hyer order heurs, is something about — i.e. whether I can or not w.o. much revision

Soln 1), using cDS, and allowing ↓ in s length, is powerful ~~Lang~~ Gramm. type. — It is possl. that it is, in some .05 a "universal" grammar. 446.01

.06 SN] Perhaps th. basic idea that I'm trying to exp th. concept of a heuristic heirarchy." This is th. idea of a set of probs (≡ level 1.); Then a set of heurs for plausib of level 1 — which is ℓ_2; Then a set of heurs for ℓ_2 - w is ℓ_3. I would like to make a general study of thes "heur heirs" — so that I understand them indipy of a particular notation or "technology". Studying them "in English" seems to be a good way, since it seems readily xltd into any other technology or notation or Lang. type.

Th. idea here, is that my intuitive solns. of various sags, is expressed in a certain notation in an approxt way. What I'd like to be able to do, is express this intuitive soln. in th. other notations, and then patch up th. holes in this soln. Th. various other notations would focus at ou th. "holes" and often suggp. methods of fixing them.

Also, by projecting" th. intuit. soln. into other notations, so easily ~~comput~~ prosts can be computed — and they can't be, in intuit. form.

What I want, is to be able to really look at and thes "solns" and understand them as well as possl., in various ways. Th: idea of being able to express them in difrnt notations and technologies, is usually a good way to do th.is — aside from being closely related to phys. realizn.!

Tu Sep 26, 61

Dart.

445.05 spat

.01 445.40 ! → Getting back to "Soln 1)" ☞ C.D.

Th. xfmns are:

$$A\left(\overset{\prime}{x}\right)B \rightarrow \left[\begin{matrix}ev\\A\left(\overset{\prime}{x}\right)B\end{matrix}\right]$$

$$(A) \rightarrow \text{[redacted]}\ A$$

$$a\ A\left(\overset{-}{+}\right)B\ b \rightarrow a\left[\begin{matrix}ev\\A\ \overset{-}{+}\ B\end{matrix}\right]b$$

or: a or b can

a may be +, −

b " " +, −

As a pgm, we can write this in Th.

.06 formalism: $A\ /\ B \rightarrow M(/, A, B)\ \phi$ ← linearstop

.07 $A \times B \rightarrow M(\times, A, B)\ \phi$

.08 $(A) \rightarrow$ [redacted] $A\ \phi$ ← grammar stop.

← we don't want to stop here yet. Th: "+"

Here $M(/, A, B)$ means "Th. evaln. by "Th. machine" (= M) of A/B

Here A and B are any "numbers". "Numbers" are a sub

or symbols — i.e. "numbers and non-nos. we may want more class

see 438.21 [redacted] .36 but leave this for awhile.

.20 In .06, .07, .08 we expect Th. rt. side of "→" to contain most, if not

Th. symbols on Th. left side — so these xfmns are of [redacted] not

low **p cost.**

Some other hours: ① M tends not to occure on Th. left side;

② M tends to occure on Th. rite side. ③ A, B, C etc. tend to occure on (

3.5 If a symbol occures on left side, it tends to occure on rite side.

④ Th. no. of symbols on Th. left and rite sides is small — we can repre

this by 2 by probty of "→" occuring after Th. left side has started,

2 by probty of "φ" occuring after Th. rt. side has started.

Th. above pgramm. can be written using little modification o

Th. code used in ZTB 141 ("the" code with dots"). Also us

.30 ideas of [redacted] for notation of classes of symbols. →(447.25)

SN) I am clearly inserting many hours when I define Th.

roles of this pgramm. It would be interesting to see just wh

sort of extrapolns can be made if I stuck to just Th. formal

machinery of .06 – .30. ~~Then see what~~ E.g. could all of

arith. evaln. be learned? Clearly, "yes" — since C.D.S. is included

in Th. formalism. However, as will be shown, using only Th. above

formalisms give too low an aprip. to C.D.S. — A special mechanism

for C.D.S. is expedient, to ↑ its p.cost.

447.0

Tu Sep 20, 61

(part)

.01 : 446.40 : SN In learning Fr. C.D.S.'s for + and
try a non-C.D.S. other with Fr.ese symbols,
for certain probs. They always work, and for others,
sometimes work - depending on Fr. order in which r
are used. On this basis, an L3, watching, could d
ordering rules for Fr. xfmns. This would result in
grammar — i.e. soln. #2). So this makes soln. #2) va

However, since Fr. xfmns are unordered in Fr. gramm.
I've been writing it, an xfmn. should not be accepted
Fr. grammar unless it works for all possl. orderings of
applications of Fr. xfmns. This is not quite as time cons
as it sounds — since there are usually only a few
that will work in some ordering and not in others — so
xfmns would be discarded early in Fr. gramm. In testing xf
Fr. order of use of Fr. xfmns would, at first, not have to be ran
Then when an apparently good xfmn was found, it would be more
tested with random choice of xfmns, in situations where > 1 xfmn
IMPT Even so, I cannot ordinarily afford an exhaustive test of all possl. orderings, and
Mt.C. test does have Fr. potency. of errors
.25 : 446.30 : To continue on com. #1) ; errors will be very diffl. to elim. —

We could write Fr. 18 xfmns :

$$a\,A\,(\mp)B\,b \Rightarrow a\,M(\mp, A, B)\,b \qquad \text{with } a \text{ may be } +, -, ($$
$$\text{and } b \text{ " " } +, -,).$$

This will, of course, give (an exponential cost, unless we design
seq. that practically tells T.M. what to do!

A better way, is to introduce classes of gramm. rules — so
we can define Fr. ngmsts $a \equiv [+, -, (\,]$: $b \equiv [+, -,)]$.
This way, we have a "hyer order grammar", that has rules for
constructing Fr. rules of Fr. final grammar.

We mite even find it useful to define Fr. classes ngmst / $\alpha \equiv [+, -]$
so $a \equiv [\alpha, (\,]$, $b \equiv [\alpha,)]$ [or, just $a \equiv b \equiv [\alpha, (,)]$
mite be better]. Then our "rule is : 1-pot.
see 448

Tu Sep 26, 61

Dirt

.01: 917.40.

aA α B b → a M (α, A, B) b

α = [+, −]

a = [α,), (]

b = a

maybe ok.

We: [illegible] + A + [illegible]

But – [illegible]

We write this as a finite PSG: i.e.

S → a A α B a → a M(α A, B) a

α, α → +, +; −, −

a, a → [illegible]),); (, (; α, α

Actually, the division of the code into various levels of B [illegible] too hard. This notation is poor: return to [illegible]

This PSG [illegible] this)

.20

Another heur. idea that mite be built into [illegible]

the idea of "context" ≡ the above formalism does [illegible]

treat "context" in a special way. To do this

Write A, a → B, a This means A → B

.30 "context a" most than be described:

e.g. a = , + or a = +, + or a = +,

these mean A+ → B+ ; +A+ → +B+ [illegible] ; A+ →

We would also like to be able to write; This is meant to be equivt. to the set of [illegible] 9 rules defns: $a = \binom{+}{(}, \binom{+}{)}$

a = α, β

α → +, −, (

β → +, −,)

If it may be that the idea of "context" is automatically obtain[ed] if we use a non-dim PSG for the "metagramm".

The idea of "Grammar of a grammar" incorporates some of the ideas of hyer order heurs – but I don't think all of them.

cont. on 45

449.01

To Sept 26, 61

Dirt

.01 : 447. 40 :

$aA\alpha Bb \to a\ M(\alpha, A, B)\ b$

$\alpha = [+, -]$

$a = [\alpha,), (]$ } maybe ok.

$b = a$

But — I may be [illegible]

Th. [illegible]
will have [illegible]
If a suitab[illegible]
it will [illegible]
new desig[illegible]

We might write this as a finite PSG : i.e.

$S \to a\ A\ \alpha\ B\ a \to a\ M(\alpha, A, B)\ a$

$\alpha, \alpha \to +, +\ ;\ -, -$

$a, a \to$ [illegible] $),)\ ;\ (, (\ ;\ \alpha, \alpha$

This is a non-
PSG. I'm not
This is a good

Actually, the division of the code into various levels of grammar and the [illegible] too hard. This notation is poor : return to it later.

.20 Another heur. idea that might be built into a gramm
the idea of "context" The above formalism does not
treat "context" in a special way. To do this we can

Write $A, a \to B, a$ This means "$A \to B$ in context a" most then be described :

e.g. $a = , +$ or $a = +, +$ or $a = +,$

.30 these mean $A+ \to B+$; $+A+ \to +B+$; $A+ \to B$

We would also like to be able to write ;

$a = \alpha, \beta$

$\alpha \to +, -, ($

$\beta \to +, -,)$

This is meant to be equivt. to the set of 9 rules defns : $a = \left(\frac{+}{-}\right), \left(\frac{+}{-}\right)$

It may be that the idea of "context" is axiomatically abstract
if we use a non-dim PSG for the "metagrammar" cont. on 455

The idea of "Grammar of a grammar" incorporates some of the ideas of hyer order heurs — but I don't think all of them.

449.01

Fri Sep 23, 61 TMJ

Dartmouth

.01: 433.40: Hvr., simply th. need of 7 additional xfmns

we either need a rather large trg. seq. or else it be

exessively long search.

→ Another possy., is that if TM "realizes" in some

what these xfmns could mean "intitively", he would be more

to dcvr. them more quickly

This brings up Q. of just how much I want to be

get TM to do things in what seems to be my own intuitive

What I would like to do, would be to be able to tell TM w

"equality" "means" — i.e. to tell TM all of th. relevant prop

of th "=" symbol. This would be useful. It would in

TM's knowing that $a \cdot b = c \cdot d$ [a, b, c, d are expressions with unknowns in them]

implies

$(a) \times (b) = (c) \times (d)$ and $(a) + (c) = (b) + (d)$, etc.

This concept would be useful in solving eqs. in 2 unks..

Also, if TM knows that $x = x$ is true for all x,

then th. some of xfmns 433.30 - .36 will have a much hyer probty of

being dcvrd.

Tho I would like to be able to "tell TM" things r directly, even at th. outset, I don't see any obvious way that this can be done. (Tho it may be possible)

Also, one way to "give" TM a certain concept, is to give him many examples in which that concept is used, so that I'm fairly certain that TM has used this concept in his coding.

What a "good teacher" often does, is give th. student certain "test problems" — by noting whether or not th. student works these "test probs" very rapidly and correctly, th. teacher feels he can tell whether th. student has "understood" certain concepts.

435.01

Sat Sept 23, 61 TMJ

Dart (435)

433.24 ?

.01 434.40: 3) A kind of soln. to these probs. (not in any great detail): TM has, either thro. innate "wiring" or previous exp. gotten th. concepts of "permissable xfmns", "solu" th. eq." This means that TM could not make "formal solu" to any set of eqs. by simply decrp. what the criterion was for that set of eqs. (The latter could be decrd. if some sort of convention was used for th. input th. problem, then, of making a "real" solu. would be to find hevrs. — or methods to shorten th. seq. of trials. Such hevrs could be of th. form of "closeness criteria" (ie. whether a xfmn. brot TM closer to solu. form or not). The hassle on 432.20 is about how G_3 could "legally" deal with this sort of soln.

.18 4) Another kind of soln. not yet discussed in more detail: (TM) has a means for recognizing whether a gn. input is amenable to any one of a set of soln. techniques. So TM tries various xfmns and seqs of xfmns to see if it can get th. problem into a form that has been already solved — and use that soln. on it. If this works, th. new original problem (in suitable generalized form) is assigned an ob (recognition operator) so it/too can be one of th. "recognizable solvable" ones. This soln. uses something like th. ob, op formalism that was pre-
.30 viously much discussed.

What I might do, is write both 3) and 4) in a bit more detail — in th. case of 3), give some detail on how G_3 or

I would like TM to be able to use any one of these 4 methods on future probs. For a Mt.C. trial, TM would start with th. R_1, R_2, R_3 or R_4 — which would indicate which formalism was being used — then would follow a string of symbols whose interpretation would depend on the nature of R_i. Th. probty of th. different R_i symbols would depend on how freqly each of th. 4 methods had been "th. best method" in th. past.

Part

.01 : 448.40 : In the forgoing work, to [illegible] had to modify the gramm[ar] / formalism appreciably — i.e. expand it. It is hoped, that after I have softly expanded it, there will be no longer necessary. ~~that~~ that any new heur. that I happen to think of will be expressable in the [that-time] formalism, and will be discoverable after a suitable sample.

Hwr., the above TP is sort of irrelevant. The basic ideas are 440.10 and 440.30 and perhaps 445.06 — in which I insert all the heurs that I think are necessary, and / than the system should be automatically capable of expressing anything. This "universality" is an essential thing that must follow, if I expect TM to be able to do much "on its own" after I stop putting in my own [intuitive] heurs. — i.e. Not only must this TM be able to express all conceivable heurs — but the [cost] with which it expresses any heur should be large, if it is "easy to think of" this heur, or if it is easy to express this heur in English & using commonly [used] mathical concepts.

.025 The degree to which I should formalize the cons. of the typ. seq. at this pt. is Qable. A lot of work and cleverness and time can go into this formaliz., and yet have it all undone by some requirements that will come later in the typ. seq. What I should do, is give a listing of the heurs I think necy (&/o adequate) to do a gen. typ. seq. — and also tell a lot about the str. of the formalism that could express these heurs. i.e. which order langs. each heur. is [illegible] in.

Later I can devise special formalisms that can express many heur devices very economically. Hwr, now, I might as well postulate a separate device for each heur.

If I have several ways to do a gen. prob., then write each one out separately.

Dart

.01: 449.40: It would seem that there are 2 [illegible]
seq. and its "solns." They are:

① Th. formalizn. of th. soln. has, (due to [illegible]
hours built into it), resulted in a "univ." formalism —
can be expressed within th. formalism. Hvr., many impt. hours have not
incorporated into th. soln., because th. tng. seq. is not yet big

② Th. tng. seq. is long enuf so; ② & ① has occured and inough
enuf hours have gotten learned so that TM can learn things fairly
well in certain fields.

③ Th. $TM_i = TM_{i+1}$ level has been reached. This may or
may not occur before ②.

Using certain simple universal formalisms, it is probly possl.
to achieve ① very soon (ie. short tng. seq) if not immediately.
One can go from ① to ② by a "small step tng. seq," w.o. th.
teacher continually revising th. internal formal long. of T.M.

I don't know how ③ fits in — tho it seems impt.

There was much discn. (I think in Dart) about a month ago, on how
to get a "univ." TM formalism at th. outset — so that all one needed was
to keep feeding in more tng. seq. until TM got
very clever.

My present approach is to not try to hurry to get to point ①.
I will have to get an awful lot of hours into th. tng. seq. before
we get to ②, anyway, and I'm not sure that it would be of much
advantage to actually go from ① to ② via a real short "small step"
tng seq., rather than via my actually modifying TM's internall formalism along with th. tng.
seq
.30 Th. advantage of this, is that I will be watching TM develope,
and will be able to understand him better, and sugg. improvements
in him that will more readily lead to his phy. realizn. If I do
this "hand development" of TM, I will reach ① automatically in
a while. I can go from there to ② either as before, or via
a real computer simulation, and a "small step" tng. seq.

So — my present mode of work: To continue th.
tng. seq. and all possl. solns. to it that I can think of. To
incorporate all these hours into th. TM formalism. To look

452.01

Dart

.01 ; 450.40 : at th. present "soln formalism" I try to see if it has yet become "universal" to decide if I should put it on a computer and pr... with a "small step" typ seq. or proceed as befor, unt... is much more powerful, before simulating him and continu... with a "small" (or even ormaddon large) step typ. seq. Th. advantage of ... keeping TM "on paper" as long as possl. is pn. in 450.30.

.10 From 446.01 – 30 we have th. dcrn. of a very simple TM, with some of its hours. Th dcrn. of th. formalism is almost complete. 447.25 – 448.40 (attempts to put formalism for ngusts, and CDS.) is a dcrn of a few more hours, and some attempts to formalize them.

What is available at 448.40, is th. the partial dcrn of a wash TM. This dcrn. is a bit more detailed on th. notation side, than necessy, and perhaps should give more on hours.

At any rate, th. dcrn. up to 448.40 is somewhat in th. direction that I want (except for these faults), and I should continue making TM dcrns of this sort, covering all solns. that I can think of, for th. continuing typ. seq.

However, I should devise a more "standardized form" to give these solns. in. E.g. ⓐ give th. direct soln itself. ⓑ give th. reasons why this was a "reasonable trial" – i.e. what regularity this soln. is an example of ⓒ Tell how this regularity was dcrd – i.e. what rep. is it an example of ⓓ etc... continue to hyer orders of reps, untill "basic reps" are obtained that have simple dcrns. in English or in th. formalism to be used – or "reps" that I have no qualms about "wiring in"

Th. only danger I can think of, of "wiring in" fairly complex reps., is that this way we do not obtain (as useable abss) th. sub-abss. that were used to make th. complex reps. – We would very much like to use the sub-abss. to

Dart

.01: 452.40 make new trial vaps.

.02 I could do this by listing the roles, plus the line that tells the hour that made the guess." After each hour, I tell where to find "meta hour", etc.

This will tell all the info allrite, but I don't know just how good it will be for visualizn.

A Q. of some impt.: The foregoing work has been almost entirely directed toward MTM, not nary toward RTM. I think, hwr, that after doing MTM, there will be parts of it that are RTM-like in function — and since I have intuitive hours for these parts, I should be able to work them allrite.

Very probably I'll have no additional trouble with a pure RTM. The basic idea here is to find an op. of hy \ G approxted over the previous input set that is of hy cost. Here, the "closeness criterion" will be of much use in devising an approxt. G function. As I try to explicate my intuitive hours in these probs., I will doubtlessly think of other general techniques for "hill-climbing".

The thing is: Suppose I get a TM that can work arith eval. and ilinear unk.: This is then a definitely programable TM. When it has completed this a typ. seq of arith. eval. and ilinear unk., then (1) for the next prob., we can figure out just what the cost of soln. will be (if we are pvn. the prob.). This gives us, then, a set of probs. that this TM is, w.o. further typ seq., capable of working in a reasonable time. (2) With a suitable (continuation of the) Typ. Seq., it is possl. to [illegible] the no. a/o quality a/o order of the hours, and ∴ work many more probs. This postulates no changes in the TM formalism. This defines a much (usually) larger set of probs that could be worked if a suitable typ. seq. were provided. ∎

If the "proper formalism" is used for T.M.! (i.e. a "univ" formalism) any prob. can be solved if a suitable typ. seq. is used.

454.01

Dart

.01:453.40: As such, this "little TM" could be a clearly defined sub-goal, & make the work easier. I don't want to go into some detail on this "little TM". I want to extend the formalism so that it can do several linear unk. operations in much detail, hwr.

It would seem that there should be, at the outset, a form that I could put T.M. into, so that it is universal. Even so, I'm not sure that this would gain much, because I have to do mainly hand-insertion of heurs anyway. If I try to do this by the seq. it will just be that much extra work — i.e. to translate heurs into smallstep tup-saqus.

So: the sub goal, is a "little TM" that can do arith. eval and 1 linear unk. (and perhaps > 1 unk.). I want to design this in soft. detail so that (1) I can see if there are any impt. ideas that I've left out (2) I can estimate the coost step betw. probes in the tup. seq. and estimate how much real time a 7090, say, would take for each step. (3) I can get a good idea as to what "expansion of the formal lang." means.

.30 SN An example of a possl M_i, G_3 comb. — both of which are universal. Say each M_i, is dcrbd. by a string, D_i, so $M_i(I_j) \equiv M(D_i \frown I_j)$ — where M is a "universal machine". [I_j is the input to Machine M_i.]

For G_3: G_3's input, is {all that has occured in the interaction of all M_i's and their inputs — including G's (if $\overline{TM}$ is a RTM) and computation times}, and a "random no." G_3's output, is the next Mt. Co. trial M_i (More exactly, it is D_i, the dcrn. of M_i; D_i is a sentence in L_3) The "random no. input" is what G_3 (as a creative Mt. C. Gramm.) turns into a "sentence in L_3". G_3 operates via a turmac — like the M_i do — so G_3 must also be "universal".

455.01

Art:

.01 454.40 : Th. trouble with these universal G_3, is that for most forms of Th. turmac, of Th. M_2 and Th. G_3 are exessively long & wite help a lot, hvr. in going in Th. rito direction.

Anyway: "Th. goal for Th. day": Work out in some detail, all of the heurs. necy for arith eval and 1 lin. unk. Don't pay much attention to formalized lngs. Use Th. format of 453.02.

433 ; 435 ;; 446.01 – 448.40. Have much work on this.

In particular, 446.06 – .30 is pretty much what I want in Th. direction of "today's goal". Th. prob. now, is to do close to Th. same thing for Th. C.D.S. xfunns of 447.01 – 448.40.

This work should be done in out-detail so that I can make approxt. pcost calcs of various of Th. wateyn. parts of the M_2's. This amounts to a criterion for telling when "Th. day's goal" has been achieved.

.25 It would seem to me, that if I really concentrated on this work, that by about 3 days work, I should have a fairly detailed outline of a "little TM" that can do: arith eval, lin. eqs. in 1, 2, 3 unks; extrap. from 3 to 5 unk.

Also, after doing this, I should be able to expand th. typ. seq. every day for, say, 10 days, (with explanation of th. corresp. solns., M_2's) until ① I have a formalism that is universal ② that $TM_1 \equiv TM_2$ ③ that Th. TM is softly powerful to go on to solve probs. that don't have very detailed typ. seq.

This assumes, of course, that it don't run into any big snags. — My previous experience with this sort of thing, is that I probably will run into some! However: try to avoid them if at all poss!

.36
.37

Well, to return to Th. problem at hand: T. CDS of 447.25 problem – 448.40: Just keep th. work in Eng. : ① As a first step, define th. concept of "context". Then, assoc. with each substitn. rule, a ⊙ or ⊥ symbol that

456.01

Dart 3

.01 : 455.40 tells whether its a CDS or not.

if it is a CDS, then low cost. (2) If it is [illegible]

th. context : like , a or ab ... etc (see

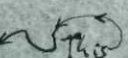
.04 Hvr., more in th. present spirit is th. formalism:

A → B | , a — which means,

A → B is a CDS, with context " , a " , so

Aa → Ba . (3) We would like to be able to define ngmsts

So if we have this we can define α = +, −, =)

and write A → B | , α  will then amount to 3 CDS rules.

Also, we can write β = +, −, (and [illegible]

th. rule A → B | β ; α which gives us th. 9 CDS

rules, since th. "choices" of α and β are indep.

Th. improvement of this notation over th. attempts of 447.25 to

448.40 is that there, we tried to indicate a linkage in " α " in

αA → αB and let α be an ngmst. We run into th.

problem of what αAα → αBα means i.e. which α's are

tied together, if any. By treating CDS as a "special

.30 thing", we make it clear as to which symbols are "tied together"

One of th. pts. of this 447.25 – 448.40 discn, hvr., was to

make a good formalism for a non-dimensional PSG. This might be

fairly difficult, because th. constraints on an ordinary 1 dimens. PSG are rather

complex — a forteriori for non-dim PSG ! I think I had an

idea for a good 1 dim PSG formalism, hvr. — don't know where I

.37 wrote it, hvr. [Perhaps mentioned in "Reviews", or "PSG discy." or "Code"

Dart 468 – 483 develops this — Also see later section of Info and control paper [written ~ March 1962]

Anyway, as of .30, I think there is enuf to get pcosts of all xforms

for arith. eval and linr unk. of 433.14 – .36. ← I'm not so sure of this! — esp. I also want linkages betw. things that are not in "context"

Note that when we start th. typ. seq. on linear unk., we introduce

th. new symbol, " = " — so we have to "add in" more

Draft

.01: 456.40: CDS's of th. + A + B = → + M(+, A, B)

old rule

A + B → M(+, A, B) | β, α

with α ≡ +, −,)

β ≡ +, −, (

We would have to change it to th. same thing but with

α ≡ +, −,), =

β ≡ +, −, (, =

This change could be done by first discovering th. rule

A + B → M(+, A, B) | =, α

then, after writing this down, simplifying th. grammar so that we eventually (after discarding several rules like), we end up with

Another point: we would like to be able to write

A (x) B → M (x), A, B)

and have th. upmost (/, x).

We have, at present, no formalism to do this. It seems that we ought to be able to find a formalism that will do it, as well as th. CDS in a "Uniform Manner".

It might be done with subscripts. e.g. If we define α as th. upmost α = +, −,), then wherever α_1 occures, it must be th. same α. However, α_2, α_3, etc., may be any other α.

We might be able to do ntpsts this way too: e.g.

α, β ≡ (a, b ; c, d ; e, f) ; Then $\alpha_1 A \beta_1$ can be either

a A b ; c A d or ; e A f.

This notation also automatically defines th. upmosts α and β.

So α ≡ (a, c, e) ; β ≡ (b, d, f).

So, for th. CDS & th. rules could be writ;

$\alpha_1 A \beta_1 B \alpha_2 \rightarrow \alpha_1 M(\beta_1, A, B) \alpha_2$

$\alpha_2 \equiv +, -,), ($: $\beta \equiv +, -$

458.01

Dart

.01: 457.40: The this subscript notation is ve
th. older notation of 456.04 that makes a
out of cDs, give hyer pcost to xfmns cont
does th. more general notation of 457.37. This sort
is, of necessity true — i.e. by using a "more genl." notation
reduce pcosts of th. more likely cases. Ideally, hrr, a notation
make defns of all kinds possl, so that with enuf typ.
notation can be simulated in terms of its pcost for so simple
.10 in terms of its mean pcost/instance). → {See 467.10 and many follg. pp for more dvlpmnt of this notation

In any of these notations, we have th. "hyer order gram
— i.e. gram. of a gramm". i.e. th. notations of 457.37 - .40
tell how to construct a gramm (≡ a set of subs. rules). But I don't exactly
see how we can go one step hyer — much less arbly hy!

It would seem as tho Lisp would be a good lang to write all
of these things in, since it is "universal", and many th useful
functions have been pgmd. (and recorded) for it. Th. troubles are
(1) Lisp is slow (tho it may well be that it is possl. to construct a
special purpose computer that would do Lisp rapidly).
(2) Lisp does arith. slowly (this may no longer be true)
(3) Lisp is built around th. frequent use of recursive defns.
I think that in English and / most human that, recursive defns. are
not that common. Note that Lisp can be used for non-recursive
things, too, but most of th. interesting functions that have been
devised for it are recursively defined, and are probly
fairly slow.

My impression is that I should do this work in my own notation -
sticking as close to "Eng" as possl. This notation should be
good enuf to work out pcosts of various "conceptual jumps". Then,
when I've gotten pretty far, in the typ. seq., I
can either simulate it in Lisp, or "hand simulate" it
in direct computer lang. (this should not be so diff

459.01

Dart

.01 458.40 since th. chosen concepts used are ... that it ought to be easy to pgm. directly — particularly if I use "defns"), or hire s.o. to make a compiler or interpretive routine.

.05 A perhaps serious ditty with much of th. present concept: If an "error" is made (e.g. deciding that a certain f_{mn} is "permissable" because it has been permissable for "th. typ. seq. up to now") there seems to be no way to correct such an error. — E.g. there is no mechanism for removing incorrect or even redundant subs. rules. → trial & elim. of some rules is discussed in 536.30; Also elim. of redundant rules is discussed a bit in 540.10 and .20.

To solve this prob: ① try to devise methods that will be helpful for th. specific type of M_i's being used up to now. ② Try to see what, in general, people do about this — keeping remarks close to intuitive English.

One simple form of this problem: If $\alpha\beta\gamma$ is th. soln. (i.e. M_i) for th. typ. seq. up to t, then how can TM access th. soln. $\alpha\beta\Delta\epsilon$ for th. typ. seq. up to $t+1$? In general, th. possy. of modifying part of th. original M_i as well as concating onto it, reduces th. Δcost of th. new soln. tremendously. 536.30 Has a fine idea about this! (= th. use of symbols that mean that a previous symbol should be erased)

I think th. way to Solve it: just go thro th. typ. seq., and, for each case of this that comes along, try to see what just now I, in fact, solve this problem intuitively.

Actually, this is by no means a "New Problem": ① I had worried much about it in early Dart (I think): ② One has all of th. old M_i's — and one has to make a new trial M_i, since none of th old ones fit. Tho th. simplest way (when it works) is to add something to th. old "best" M_i, this, in general, will not work and one must often do much more by way of modification.

460.0

Barb

.01: 459.40: One method of "modification" for Th. ob-op. method — in which TM, noticing that "op" doesn't work in Th. near input + finds a Th. new input type and make a new sub-op for Th.

Th. thing that I seem to be worried about now, is that I may not be able to devr. "suitable reasons" (i.e. heurs) for Th. solns. to lots of Th. probs - that I solve "by hand." An example is this most recent problem of 459.05 - 460.04 — i.e. teh How to get appropriate changes in Th. old thru, one-time-adequate M_i's, other than simple "additions" to them. Well, in general, I will not be able to list all of Th. heurs that I use in solving a prob — hvr., Th. hope is, that by listing all of Th. heurs that I can think of, for a large enuf set of probs, that I will "cover" Th. space of heurs well, so that TM can "get started" on its own. At th. present time, this seems like a fairly reasonable hope.

With Th. notation of 457.37 is probly adequate. Also, (for up to arith eval. and linear rule. See 462 for refinement of 457.37)

(Th. previous heur that makes it likely that symbols appearing to Th. left of "→" will appear on Th. rite too > will help much here, and will, to some extent, compensate for not introducing Th. more narrow heur & concept of "context". Hvr., even with Th. notation of 457.37, I think we might be justified in introducing Th. concept of "context" anyway, in addition.

My impression is that Th. problems of my finding heurs is very similar to Th. probs of finding solns. — So that when one makes this heirarchy of heurs for a gn. prob., one is simply doing more probs. of Th. "same kind." This makes a $TM_i = TM_{i+1}$ more likely sooner. (I'd like to see more detail on this!).

461.

Dam

.01: 460.40: xfmn 433.36 — i.e. "z → 1z"

Th. symbol "1" come from? What is its ?

Th. symbol "1" into TM's initial vocab, and allowing

we make it possl. for TM to count things. E.g. we

$C(\alpha, z)$ as th. no. of times / α occurs th. symbol

string z. To evaluate $C(\alpha, z)$ we subject z to th.

$b\alpha c \rightarrow \alpha bc$ where b is any non-null non-α symbol and

c is a null symbol. Then do $c\alpha \rightarrow 1\alpha$ and

$A\alpha\alpha \rightarrow M(+,1,A)\alpha$. Then do $A\alpha d \rightarrow A$, where d is

any non-α symbol.

I'm not so sure this → is exactly correct — but it doesn't matter for just now, anyway. I think that th. inclusion of th. symbol "1" in the initial vocab. is justified. It is, of course, of very low priority since it's probly only used once / in this sug. seq. thus far. Hvr., it may be an essential symbol (or operation). It may be possl. to find a way to get around this symbol — but I should spend more time on it now.

One way to avoid this "1" bussiness: Every time z occures, we make th. convention that it must be preceded by a numerical coeff. This reduces th. no. of probs TM has to face, and I don't see any point in that! — I want TM to solve as many probs as possl., and in a realistic manner.

.30 O.K.: Th. soln. thus far: arith. eval. and / in. unk:

Th. formalism is this: that all xfmns of th. form $z \rightarrow b$ (a, b are sub-strings) are to be tried in any order on th. input string, which yields a new string. These xfmns are to be continued until there are no more legal xfmns. Th. resultant unxfmable string will be th. "output."

Th. over-all M; operator will, then, be completely defined by this set of xfmns. It is assumed that th. terminal result is indep. of th. order in which th. xfmns are used, — so if a set of xfmns is found which, for a given external input, can give ambig. results thru difrnt choice of order of xfmns — Then that set of

462.01

Dart

(8462)

.01: 461.40: xfmns is illegal.

A set of xfmns that appears adequate for ...

Here, M always has 3 symbols following ... See 446.06-... See 446.20-.30 for ...

.03 A / B → M / A B ϕ

.04 A X B → M X A B ϕ

.05 (A) → A ϕ

~~[illegible]~~

.06 α ≡ + − ( ) ϕ ← wrong! see top of 448 (Maybe OK. — also see ...)

.07 β ≡ + − ■ ϕ

.08 $\alpha_1 A \beta_1 B \alpha_2 \rightarrow \alpha_1 M \beta_1 A B \alpha_2 \phi$

Note that α and β must be ... (i.e. .06 and .07) before they ... be used in th. set of xfmns.

Th. subscripts can be written without ambiguity as integers following th. symbol that they are meant to subscript — as α 1 A β 1 B α 2 etc.

Consistent with th. subscript notation, we may want to write A 1 and A 2 instead of A and B, resp. Here "A" means "a number" and "1" means th. first such no. introduced.

Similarly, we may want to categorized non-numerical symbols, so N 1, N 2, N 3, N_4 etc, mite represent →, /, X, M, etc.

Hvr., since A 2 will be used very often, it will usually be convienient to define th. 2gm A 2 ≡ B, so we would be just about back where we started — except that we could introduce as many letters as we like — which mite be diflt. otherwise.

This may not be such a good idea — but using subscripts on α and on A, sounds quite reasonable — since we want to be able to have an arby no. of these symbol types.

Perhaps always have all defs. first: so th. op. would be;

.31 α 1 ≡ + − ( ) ϕ

.32 α 2 ≡ + − ϕ

.33 α11 A1 α21 A2 α12 → α11 M α21 A1 A2 α12 ϕ (see 557.35 for hvr.)

(α11 ≡ α_1, A1 ≡ A, α21 ≡ β_1, A2 ≡ B, α12 ≡ α_2)

Note that 2 subscripts always follow α. Th. first tells whether th. symbol represents α, β, δ, ... and th. 2nd subscript tells what th. regular subscript ...

Th. basic symbols, other than integers, ...

.335 α A ≡ + − X / ( ) M ϕ ϕ — 13 types.

.34 A1 / A2 → M / A1 A2 ϕ

.35 A1 X A2 → M X A1 A2 ϕ

.36 (A1) → A1 ϕ

For 1 lin. unk we will have to introduce several new symbols. i.e. Z, 1, V

V is th. null symbol. It is used to characterize th. content of xfmn 433.31

Dart

.01: 462.40: In Th. [redacted] op. dcvn. of 462.31-.36, there are [redacted]
e.g. ① ≡ cannot follow ≡ ; ② After → has occured, ≡ can never
(This puts all defs. at Th. beginning); ③ M cannot be followed too
; ④ ϕ must occure betw. successive →'s or ≡'s [redacted]

.05 betw. ≡ and →

.06 There are a few other "shortenings" we might try: e.g.

$\gamma \equiv / X \phi$

$A\,\gamma, B \rightarrow M\gamma, AB\phi$ instead of 462.03 and .04. i.e. There are fewer symbols here — Tho not nearly hyer post.

$\begin{cases} A/B \rightarrow M/AB\phi \\ A \times B \rightarrow M \times AB\phi \end{cases}$

.10 Even before we go on to Lin. unk., we can look at Th. requirements for G_3 — see just what sort of machine it would have to be in order to recognize Th. hours of 462.20-.30 — and also (perhaps) recognize Th. "shortening" possy of .06. Whether G_3 can or should recognize actual constraints (like .01-.05) is apparently related to Th is.

This looks like a tuff prob.

This "shortening" bit is one of Th. "normal" trials for PSG discy — apparently. As for the other hours of .01-.05, they are of a difrnt sort than those I've tried to use in PSG discy — but they do look interesting and of a generalized form

Well, I can certainly write out Th. hour rules that constitute G_3. Obtaining their post in a reasonable way, is a more difft. This problem is certainly an impt. one — i.e. Taking fairly intuitive forms of {hours or prediction methods} and expressing them in a form so that their posts can be readily computed. This problem is also of much impt. in any very good method of hr. or S.R. prediction — i.e. one in which one synthesizes a lot of intuitive methods into one quantitative method.

Perhaps first define clearly what this G_3 is supposed to do — making Th. boundaries betw. it and Th. M_2. Then take a set of hours or constraints that would be part of G_3, and express them in a way that their posts are computable.

Looked at this way, it would seem that Th. basic M_2 of 462.03—.08 could have all 18 rules represented by 462.08 written out in expanded form. Th. function of expressing these 18 rules in a single compact form, is G_3's function. G_3's function is

Dart

(464)

.01: 463.40: to find coding means to reduce the costs of R as much as possible. Also, finding and defining useful "good M_i's" is G_3's function.

My present conception is that G_3 must express all of R. regularities that've been found in good M_i's, and as such, G_3 is a program to produce the best possible set of stochastic trial M_i's.

A form G_3 can take: It could be just a set of symbols with their intersymbol constraints, their relative probys, and definitions of these symbols in terms of R. basic alphabet. With this (or any other) form of G_3, G_3 is represented by a string of symbols. This string requires a certain machine to interpret it. — just as R. operator described by 462.03-.08 (or 462.31-.36) requires a special machine to interprate it properly as an operator on an input.

For both G_3 and R. M_i's, the nature of this "machine" will determine to what extent G_3 or M_i can be "universal".

It is G_3's function to "find a better M_2". This includes (as a special case) finding a new M_i that is considered better than the best known one is not (in view of a new (known Input, Output)).

There is some Q of whether the "factorizer" of R. 18 rules of 462.08 should be part of G_3's work or not — i.e. should M_i's interpreter be pgm. these 18 rules in expanded or factored form? Either would seem "legal", but I'd like M_i's interpreter to be pgm. R. expanded 18 eqns., so that I will have at least 1 of G_3's simpler functions to examine — i.e. G_3's ability to "factor" sets of rules.

All G_3 has to do is find regularities in M_i's and incorporate these regs. into its M.T.C. method for creating new trial M_i's.

Actually, not entirely. G_3 also must look at R. entire operation of Good M_i's and on this basis, decide on how to improve them.

(W oct 7, 61)

TM 8

Draft

(466)

.01 464.40! Dropping G_3's more extensive funct. for a while, I feel that G_3's main present functs. in solve ① Making gram. ② factoring. Th. translation of the results of these 2 operations into Th. Mt. C. creation of new Ms trials, is Th. function of "G_3's interpretive Machine".

Very probly, Th. heur concepts of symbol freq., defn. of nouns and perhaps factoring should be regarded as basic inborn heur principals.

.08 If we do regard these 3 heurs as "basic" and build them into a TM, and allow no further heurs (i.e. fixed G_3), we can make a TM that will learn and extrapolate — but it will not extrap. very well — i.e. it will be very limited

.11 as to the kinds of regs. betw. known good I,O pairs that it can recognize.

.12 What I want to do now is find a way to allow G_3 to expand. One somewhat general concept of "regularity in a corpus" is gen. in Epist 441.20. — i.e. derb. a set of subsets of elements of the corpus, and make a statistical distrib. of some function on this set. I think regs. of this kind are often fairly readily translated into specif. of a Mt. C. string generator — i.e. a stoch grammar.

A set of heurs that I mite try to express in some kind of compact notation, is 446.20 — .30. ||| Heur #4 is easily expressed by $\rightarrow$, $\equiv$, ϕ and Φ having certain symbol probys.

Also, Th. constraints of 463.01 — .05 should be expressable. Heres ① — that $\equiv$ cannot follow $\equiv$ can be obtained by defining Th. gram $\equiv\equiv$, which turns out to have zero freq. ||| In ③ we can do similar things by defining $M\phi$ and $M\Phi$. We would like to define $M_{\alpha_1}\phi$ and $M_{\alpha_1\alpha_2}\phi$ in which α was the set of all basic symbols. This suggests an expansion of G_3's formalism. ||| constraint ② is unnecy I think. It would help in computing P costs, but is irrelevant if Mt. C. generation of strings is used. |||| Constraint ④ (alternation of $\rightarrow$ and ϕ) is diffl. to formalize. Perhaps discussit in conjunction with ①, ②, ③, 3.5 of 446.20.

For these Th. concepts of ⓐ "left" Th. set of ("rite" "left) sides"

ⓑ Th. relations betw left, rt., left side pair, are impt. concepts.

It is clear that when I say that ④ is hard to formalize", I mean that I can't think of any formal lang. in which this idea is easily (i.e. by P cost) expressed. (A finite state machine can alternate "→" and "ϕ" if it has only 2 states.)

467.01

Dart

.01 : 466.40 : It is also clear that any Univ. lngo could express th[illegible] not "readily" with hy pcost.

Some possl. ways to express this alternation: ⓪ Use of a [illegible] states. ① has [illegible]

① Between any pair of ϕ's there is at least one "$\equiv$" or "$\rightarrow$"

② In every upm of th. form $\phi a \phi$, th. string "a" contains [illegible] $\equiv$ or $\rightarrow$ (inclusive or).

③ If ϕ_2 "follows" ϕ_1 ("follows" means "comes after immediately or later") then either $\equiv$ or (incl.) $\rightarrow$ must follow ϕ_1 before ϕ_2 does.

④ We can ↑ pcost by a factor of about 4 by also making statements @, ② [illegible]

.09 in th. form that "betw. a $\equiv$ and a $\rightarrow$, there must be a ϕ."

.10 → SN 458.10 has a good precursor of this tack

We can express a lot by th. set of defs:

$M \alpha_1 \beta_1 \beta_2$; $\alpha \equiv +, -, \times, /$; $\beta \equiv A, B$ (i.e. any numbers).

.13 → Th. Defs $M \gamma_1 \Delta_1 \Delta_2$; $\gamma_1 \equiv$ (not α) ; $\Delta \equiv$ (not β) are perhaps of equal or greater value.

Th. defining sets of upms mite have to be done in a more complex way: so if we define $Q \equiv M \alpha_1 \beta_1 \beta_2$, and

$\alpha \equiv +, -, \times, /$ (4 values) and $\beta \equiv A, B$ (2 values) then

Q whenever used, will have 3 subscripts following it to denote which of the $4 \times 2 \times 2 = 16$ upms is meant. Th. symbol Q, itself, will have a certain frequency, which will be $\propto$ th. total pcost of all 16 upms. Also, the symbols following (within 3) of Q may or may not be gv. a special freq. count — depending on just how one wants to do it. I think that there is nothing to lose by a special freq. count here.

Note that this notation would very probly be very useful for PSG disc in th. non-looped phase. Non-looped, because each defn. will have a known, fixed no. of "subscripts" following it.

Woops! Perhaps we don't even need that constraint!

Say we have $Q \equiv M \alpha_1 \beta_1$, $N \alpha_1$ [if we write $N \alpha_1 \alpha_1$ have we have an interesting lngo.!]

Then, in our corpus dern, when we write Q, it will be followed by 1 or 2. "1" denotes $M \alpha_1 \beta_1$; "2" denotes $N \alpha_1$.

If "1" is written, th. next integer(s) tell(s) which of th. α set is meant. α may, itself have to be designated by several subscripts. When

468.01