.01 : Instead of trying for a RTM, perhaps I should devise an essentially a program optimizer. – This would make TM₁ = TM₂ ... implement. ~~Sm~~

One Q: Is a ~~H~~ H.C. TM. as good as an RTM? – i.e. can it be taught anything.

Well, an HCTM can do a great deal: first let us desc. its ba...

.05 we give it either ① a black box, whose input is a string of symbols (≡ pgm) whose output is Th. G. of that pgm. Th. box is not "open" to Th. HCTM.

DEF

.07 ② ~~[illegible]~~ a pgm $P_1$ for evaluating any proposed program $P_x$, so that $P_1(P_x)$... $P_1$ is "open" to HCTM. ||| HCTM's function is to find a pgm $P_x$ ∋ $G(P_x)$ is max. or as large as HCTM can get it.

Applications: ⓐ If $P_x$ is a pgm to find proofs of theorems, ~~Th~~ G may be Th. real time it takes $P_x$ to solve all problems in a gn. set. ⓑ Say $P_x$ ~~is~~ describes an operator i.e. a relationship betw. $\vec{I}$ and $\vec{O}$. We have a black box that scores each $\vec{I}$, $\vec{O}$ pair – so we can ask HCTM to make an operator, $P_x$, ∋ Th. mean score over a certain set of inputs, is max. This looks much like RTM.

ⓐ is with an "open" criterion for G – i.e. ②, ⓑ is a black box, "closed" criterion for G – i.e. ①

---

I'm not sure that Th. machine that can do ② ("open" criterion) can also do ① ("closed" criterion) — Tho a machine that can do ① can do ② to some degree but may not be able to take advantage of all Th. info available.

---

Tho I'm not sure of all of Th. above: to continue:

One can make a H.C. TM. that will work, but very slowly. One starts with a set of universal ~~[illegible]~~ computer orders, ~~∋~~ – ∋ any pgm can be expressed as a string of them. (e.g. Lisp, or any ~~[illegible]~~ universal set of computer orders). The exhaustively search thru all pgms, starting with Th. ~~shortest~~ pgms of length 1, Then all those of length 2, etc. Eventually, it would begin to get some $P_x$'s of hy G.

Now – any HCTM (and this is one), can be gn. Th. problem of improving itself, if it has some external problems also, so, then, we do have ~~a~~ ~~the~~ BSTM – Tho a very slow one.

Now – next, be begin to improve this HCTM by putting various heurs, and we see how much faster it gets. Eventually, when we put in enuf heurs, it will, within a reasonable amt. of time, suggest a few more. – And then, we may be off Th. gnd!

560.01

.01: 559.40: An apparently good method of perhaps making a T.M., is "correlational coding", and correlating "everything with everything else". As one example: Say we have a hill climbing prob., like 559.05 with a "black box" evaln. criterion. We have a few trial $P_x$'s, and their G's. We "correlate" (in the general correl. coding sense) various "features" of the $P_x$'s with the values of various G's - (e.g. use 2 or 3 G levels to start off. - then use more levels when we have more data).

"Features" ("Properties") of a $P_x$ are the results of various operations on that $P_x$. We can associate "values" with various operators (telling how useful they are in prediction of G) and we can combine very useful operators to obtain new ops. of hi expected usefulness. Also, the operations for combining ops. have Utilities assigned to them.

With regard to G itself, we can devise various ways to approximate the G value that a gn. $P_x$ would get, w.o. putting that $P_x$ into the "black box." I.E. we devise various functions involving the presence or absence of certain "features" — or (since a "feature" is the result of an operation), as a function of various parameters (like the no. of 1's in the $P_x$ - which is the result of an operation). One reason we want to get G as a "smooth" "function" of features of the $P_x$'s, is that we can thereby get hyer values of G than have occured befor.

Also, we can devise various xfmns on the $P_x$'s that have the effect of ↑ their G, or have the effect of changing their G in a normal distribn. of large variance (tho the mean may be a ↓, large variance makes the possy. of an ↑ in G suffty reasonable). These xfmns of the $P_x$'s that are supposed to ↑ their G's, can also be assigned Utilitys, and can be combined, using hi U xfmns combination methods.

I think it mite be possl. to do fairly well with a "single stage" T.M. (i.e. no $TM_3$ or $TM_4$) and no $TM_1 = TM_2$. In fact, one should have a very good $TM_1$ befor trying $TM_1 = TM_2$.

Actually, I have considered only 1 way to get new trial $P_x$'s — i.e. xfmg old $P_x$'s of known hy G. There are other ways to get good trial $P_x$'s — i.e. applying xfmns and combination rules to various sub $P_x$'s (i.e. sub programs or

561.01

.01: 560.40: subroutines). These subpgms will also be assigned U's.

DEF Th. idea of U (≡ Utility) as I am using it here, is not yet cle[ar]
What I mean, intuitively, is that a hy U object has h[y]
of ~~being~~ yielding a hy U object when combined with o[ther]
hy U objects. — A $P_x$ of hy G, or a very (statistically)
successful ~~≡~~ xfmn is defined to have hy U. — and Th. Obs of
things that combine to form hy U objects are recursively
"defined" to have hy U. This is perhaps Th. same as
Th. vague concept of "Utility" that I had when I wrote Th.
Dart. Rep. There is a difference, hvr., in that now
I have a much better idea as to how to use this concept.
Actually, I can have many difrnt types of U's for a
gn. abstraction. ~~≡~~ Apri, these U's may be Th. same, but
they will go to difrnt values as more data comes in —
— also, we may devise some new kinds of U's for a gn. abs., as other new abss and types of abss. appear.
Note that this U is not identical with ~~≡~~ pcost, tho
there may in many cases, be a strong corresp. betw. Th. 2.
I may still want to use that old Zator "internal report" on
Utility — since it did have a lot of desirable properties.

Note that Th. presently contemplated approach is a very
direct one. → Instead of looking for an object that has certain
propertys, (tho one does do this for Th. final $P_x$'s, they are Th. only
objects that get this treatment), one generates objects
(like xfmns and trial $P_x$'s) that have hy apri U's, ~~≡~~ by
combining objects of ~~known~~ betterly known hy U's.
(Tho, actually, all objects are created in a trial way (by combining hy U sub objects)
and then tested — so this idea isn't exactly rite), Th. idea is that
one searches for hy U objects by combining 2 or 3 or only a few
hy U objects — so I think Th. probty of success is greater.

It would be well for me to develope this correlational HCTM
of 560.01 ff, as much as possl., on a hy theoretical level — sticking to
abstract, intuitive ideas, as long as possl., and not getting involved
with mundane, picky problems

spec. 563
562.01

.01: 561.40: I think th. ~~big~~ most recent "Big Plan" was
To get a fairly good MTM working (Theoretically, the
(An MTM is one whose problems have answs. that are
rite or wrong — no gray scale). From this, it seemed that
it wouldn't be hard to go to a RTM (each answ. is given
a G, and th. machine trys to give responses that maximize G)
because th. heurs and methodology in general seemed about
th. same. From RTM to an HCTM is no distance at
all, and since a HCTM can improve itself, we can
have $TM_1 = TM_2$, if our RTM (= HCTM) has been
gn. enuf probs and has solved them (so asking $TM_1$ to work
$TM_2$'s probs. would not be giving it a too difft task).

More recently, on 559, 560 and 561, it was suggested (more specifically on 560 and 561) that it mite be easier to start work on an HCTM directly — using various corral.coding methods and th. old idea of Utility of th. Dont Rep.

---

The approach here was to make a good MTM, starting with arith. evaln. with lots of heurs, so that ① th. heurs were good enuf to solve probs. with reasonable speed ② I would then try to express th. heurs in a "compact notation", so it would become clear as to how I could devise new trial Heurs (using th. basic symbols and defs and subroutines of this "compact notation"). In working on th. problem of improving this MTM, I would be $TM_2$, and I would have a good idea as to what a HCTM would need, in th. way of heurs, etc.

561.40 spec
.01'. 562.40 ⟶ Another impt. trick is th. concept of "partial ...
e.g. say we have a $P_x$ that is "supposed" to solve all p...
in a gn. set, and its G = − (time to solve them all).
if it doesn't solve all of them, but solves most of them ...
rapidly, we may want to give this $P_x$ a U value, that ...
at first, a measure of its ° of success, and is meant to give
an idea as to how useful that $P_x$ would be as a thing
to be xfmd or a thing to give component subroutines to result
in $P_x$'s of hy G.

At th. present time, I am much more able to deal with th. problem of devising a method to evaluate U. Now, I can view U as simply a useful device for ① making estimates of th. G of new $P_x$'s and ② suggesting how to create $P_x$'s of hy expected G, by telling how to make useful abss. that combine to create $P_x$'s of hy expected G.

---

In working out a HcTM of this sort, I must be sure that th. basic abss that I use, as well as their comb. rules are adequate to ① express any ppm that I may think of ② can express definitions and any other code compression devices that a "Univ. machine" can construct.

---

Th. thing is, then, to get a good ppm. optimizn. problem to work on.
→ There is a very large list of TM-type probs very early in "Plans".
— but befor that, I should think of this more, on a purely abstract level.
I think that there was a lot of abstract work on HcTM's in "BSTM".

One trick was to try to approx. th. functional form of $G(P_x)$ — so we can predict, to some extent, what th. G of a particular $P_x$ trial will be. Th. point of this would be that this puts th. problem in th. form of 559.07 — i.e. TM can "see inside" th. black box. Presumably this extra info could be well used to devise $P_x$'s of hy G. It enables TM to understand th. mechanics of evaln. of $P_x$ — so that it can, presumably, sometimes work backwards to devise $P_x$'s that will have hy G. I don't know of anything I've writ. on this partic. "working backwards" trick — but it seems impt.

Another trick is to "allocate responsibility" for th. goodness or badness of an abs., by cross-correlation of th. G of abss. with th. presence or absence of various sub-abstractions in them a/o "properties". Getting th. responsibility down to

564.01

Tu, May 8, 62

TMJ

564

.01: 563.40 : component abss. (i.e. sub-abss. from which th. main abs. was [illegible]
is usually more useful than just evaluating th. U's of "proper[illegible]
since th. U's of sub-abss. can help us directly to make new abss. of hy[illegible]