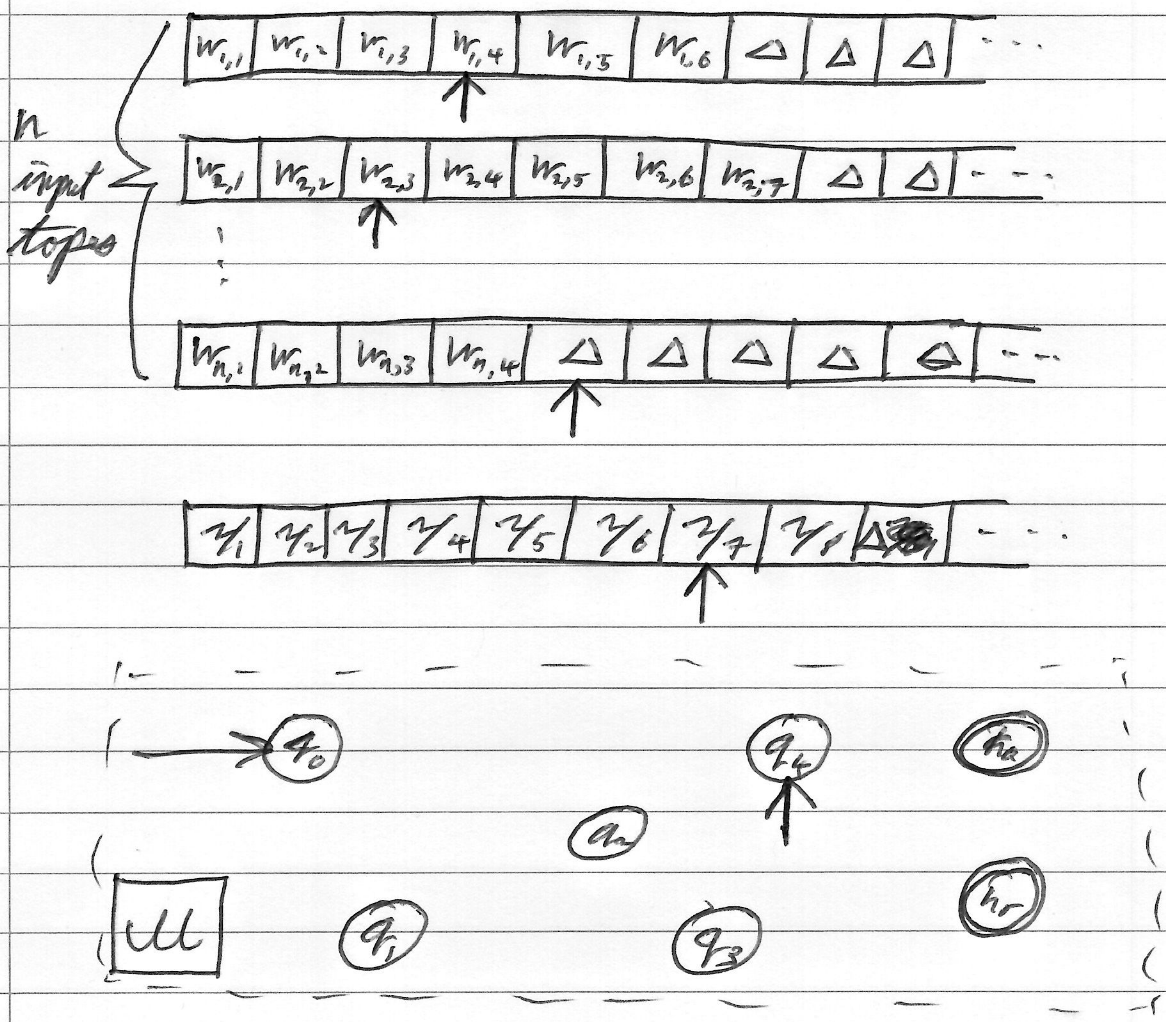
Definition 13: Diagram



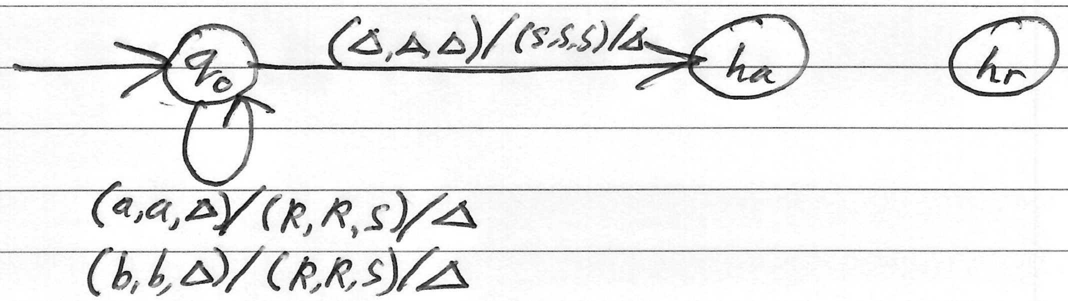
The above machine M is in configuration

$$(w_{1,1} w_{1,2} w_{1,3} q_4 w_{1,4} w_{1,5} w_{1,6},$$
$$w_{2,1} w_{2,2} q_4 w_{2,3} w_{2,4} w_{2,5} w_{2,6} w_{2,7},$$
$$\ldots,$$
$$w_{n,1} w_{n,2} w_{n,3} w_{n,4} q_4,$$
$$y_1 y_2 y_3 y_4 y_5 y_6 q_4 y_7 y_8)$$

Example TM 1

The following 2-TM accepts iff both input strings are equal.

$M = (\{q_0, h_a, h_r\}, \{a, b\}, \{a, b, \Delta\}, q, \delta)$ where δ is described by:



Some example computations:

$\cancel{(q_0 aba, q_0 aaaa) \vdash (}$

$$(q_0 aba, q_0 aaa, q_0) \vdash (a q_0 ba, a q_0 aa, q_0)$$
$$\vdash (a h_r ba, a h_r aa, h_r).$$

$$(q_0 ab, q_0 ab, q_0) \vdash (a q_0 b, a q_0 b, q_0)$$
$$\vdash (ab q_0, ab q_0, q_0)$$
$$\vdash (ab h_a, ab h_a, h_a).$$

$$(q_0 a, q_0, q_0) \vdash (h_r a, h_r, h_r).$$

Proposition 19

Thm SA is r.e. but not recursive.

Proof: To see that SA is r.e. is easy, by constructing a 1-TM $\mathcal{M}$, which on recieving $e(\mathcal{M}')$ as input, simulates $\mathcal{M}'$ on $e(\mathcal{M}')$, and if it accepts, we accept, else, compute forever.

To see that SA is not recursive, recall Proposition 18: a language is recursive if both it and its complement are r.e. So, it suffices to that NSA is not r.e.

Suppose by contradiction that $\mathcal{M}$ is a 1-TM s.t. $L_1(\mathcal{M}) = NSA$. Then, either:

(1) $e(\mathcal{M}) \in L_1(\mathcal{M})$,

or

(2) $e(\mathcal{M}) \notin L_1(\mathcal{M})$.

If case (1), then $\mathcal{M}$ does not accept its own code ... a contradiction!

If case (2) then $\mathcal{M}$ does accept its own code ... a contradiction!

So it must be that no such $\mathcal{M}$ exists, so NSA is not r.e. $\square$

Example 26

Thm The uniform halting problem is undecidable in general.

Proof: Suppose by contradiction that the uniform halting problem was decidable. Then there exists a 1-TM M that halts on all inputs such that $L_1(M) = UHP$, where UHP is the language of all the codes of TMs that halt on all inputs.

We will show that this machine cannot exist by reduction of the self accepting problem...

Let M' be an arbitrary 1-TM, and let M'' be the TM that ignores its own input and simulates M' on $e(M')$. If the simulated machine enters the ha state, then so does M', and otherwise, it computes forever.

~~But~~ So M'' is a machine that halts iff M' accepts its own code, and M is a machine that always halts and tells you if its input halts by accepting or rejecting. So by running M on $e(M'')$ we can decide if M' accepts its own code, which is undecidable by Proposition 19.

Theorem 42

A more complete version of this is theorem is available in pages 2·11–2·14 of the notes for my lecture series:

Intro to Computable Analysis,
Summer 2019.

Propositions 46, 48, 49

See pages 2·6, 1·11, 1·13 from the notes from the above lecture series, respectively.