A Universal Turing Machine

A Universal Turing machine ($\mathcal{U}$) has been produced for Anthony Morphett's Turing machine simulator at http://morphett.info/turing/turing.html. This link automatically loads the code for the UTM into the simulator. $\mathcal{U}$ emulates 3-symbol Turing machines whose symbol set consists of ␣ (blank), `0` and `1`.

The current configuration of machine $\mathcal{M}$ is encoded on $\mathcal{U}$'s tape as follows:

$$\ldots \;\text{␣}\;\text{␣}\; \textit{left-tape}\; \mathtt{[}\; \widehat{\mathcal{M}}\; \mathtt{]}\; \boxdot\; \textit{right-tape}\; \text{␣}\;\text{␣}\; \ldots$$

- *left-tape* is the contents of $\mathcal{M}$'s tape to the left of the current cell;
- $\widehat{\mathcal{M}}$ encodes the transition rules for $\mathcal{M}$, with the current state marked;
- $\boxdot$ is the contents of $\mathcal{M}$'s current cell;
- *right-tape* is the contents of $\mathcal{M}$'s tape to the right of the current cell.

The states of $\mathcal{M}$ are considered to be numbered $q_1, \ldots, q_m$. If q_C is the current state, the encoding $\widehat{\mathcal{M}}$ has the form

$$\widehat{q}_1 \;\mathtt{:}\; \widehat{q}_2 \;\mathtt{:}\; \ldots \widehat{q}_{C-1} \;\mathtt{:}\; \widehat{q}_C \;\mathtt{!}\; \widehat{q}_{C+1} \;\mathtt{:}\; \ldots \widehat{q}_m \;\mathtt{:}$$

where $\widehat{q}_i$ encodes the transition rules for state q_i, states being terminated with colon (`:`), except for the current state which is marked by having a terminating exclamation mark (`!`).

If the transition rules for state q_i are

$$\begin{aligned} q_i, \text{␣} &\to s_j, D_j, q_j, \\ q_i, \mathtt{0} &\to s_k, D_k, q_k, \\ q_i, \mathtt{1} &\to s_\ell, D_\ell, q_\ell, \end{aligned}$$

where $s_j, s_k, s_\ell \in \{\text{␣}, \mathtt{0}, \mathtt{1}\}$, $D_j, D_k, D_\ell \in \{\mathtt{L}, \mathtt{R}\}$ (the head always shifts) and $1 \leqslant j, k, \ell \leqslant m$, then the encoding $\widehat{q}_i$ has the form

$$s_j\; D_j\; \sigma_{j-i} \;\mathtt{,}\; s_k\; D_k\; \sigma_{k-i} \;\mathtt{,}\; s_\ell\; D_\ell\; \sigma_{\ell-i}$$

where the three 3-part transition rule encodings are separated by commas (`,`), and

$$\sigma_\delta = \begin{cases} \mathtt{.} & \text{if } \delta = 0, \\ \mathtt{+}^\delta & \text{if } \delta > 0, \\ \mathtt{-}^{|\delta|} & \text{if } \delta < 0, \end{cases}$$

superscripts denoting repetition. Thus a sequence of $|\delta|$ pluses or minuses encodes the relative change in state number, a period (`.`) being used to denote no change in the state. If there is no transition rule for (q_i, s) (interpreted as a halting configuration), then an empty string is used in $\widehat{q}_i$ instead of the 3-part transition rule encoding.

The size of $\widehat{\mathcal{M}}$ is $O(m^2)$, where m is the number of states in $\mathcal{M}$.

Example. The simple 2-state Turing machine for incrementing a binary number

$$
\begin{aligned}
q_1, \text{␣} &\rightarrow \mathtt{1}, \mathtt{L}, q_2, \\
q_1, \mathtt{0} &\rightarrow \mathtt{1}, \mathtt{L}, q_2, \\
q_1, \mathtt{1} &\rightarrow \mathtt{0}, \mathtt{L}, q_1,
\end{aligned}
$$

with initial state q_1, initial input **`1011`**, and initial cell the rightmost bit of the input, would be encoded

`101[1L+,1L+,0L.!,,:]1`

Further examples can be found in comments at the top of the source code.

Implementation

$\mathcal{U}$ makes use of the sixteen symbols ␣ **`0 1 [ ] , : ! L R . - + # < >`**.

It starts by moving right (to just after the **`]`**) to set its current cell to the current cell of the machine ($\mathcal{M}$) being emulated. It then repeats the following steps in order until halted in step 4:

1. Read the symbol, s, from the current cell.
2. Find the current state, by moving left until the **`!`** is found.
3. Find the current transition rule by moving left beyond 2, 1 or 0 commas, depending on whether s is ␣, **`0`** or **`1`**, respectively.
4. If the current rule is empty, halt.
5. Otherwise, mark the current rule by modifying the encoding of its target state, changing a period to a **`#`**, each minus to a **`<`**, or each plus to a **`>`**.
6. Read the target symbol from the current rule, move right and write it to $\mathcal{M}$'s current cell.
7. Move left to the current rule (marked by the presence of **`#`**, **`<`** or **`>`**) and read the shift direction (**`L`** or **`R`**).
8. Shift the whole encoding of $\mathcal{M}$ left or right one cell, thus updating $\mathcal{M}$'s current cell.
9. Change the state if necessary by replacing each **`<`** with a **`-`** or each **`>`** with a **`+`**, and each time moving the **`!`** to the position of the next colon to the left or right, respectively. If no state change is required, the **`#`** is just changed to a period.
10. Move right to $\mathcal{M}$'s current cell, in order to start step 1 again.

$\mathcal{U}$'s performance is linear in the number of steps executed by $\mathcal{M}$, and cubic in m.

David Bevan
The Open University, England
April 2016
http://mathematics.open.ac.uk/people/david.bevan