und.1 Representing Turing Machines

tms:und:rep:sec

In order to represent Turing machines and their behavior by a sentence of first-order logic, we have to define a suitable language. The language consists of two parts: predicate symbols for describing configurations of the machine, and expressions for numbering execution steps ("moments") and positions on the tape. explanation

We introduce two kinds of predicate symbols, both of them 2-place: For each state q, a predicate symbol Q_q, and for each tape symbol σ, a predicate symbol S_σ. The former allow us to describe the state of M and the position of its tape head, the latter allow us to describe the contents of the tape.

In order to express the positions of the tape head and the number of steps executed, we need a way to express numbers. This is done using a constant symbol o, and a 1-place function $\prime$, the successor function. By convention it is written *after* its argument (and we leave out the parentheses). So o names the leftmost position on the tape as well as the time before the first execution step (the initial configuration), o' names the square to the right of the leftmost square, and the time after the first execution step, and so on. We also introduce a predicate symbol $<$ to express both the ordering of tape positions (when it means "to the left of") and execution steps (then it means "before").

Once we have the language in place, we list the "axioms" of $\tau(M, w)$, i.e., the sentences which, taken together, describe the behavior of M when run on input w. There will be sentences which lay down conditions on o, $\prime$, and $<$, sentences that describes the input configuration, and sentences that describe what the configuration of M is after it executes a particular instruction.

tms:und:rep:defn:tm-descr

Definition und.1. Given a Turing machine $M = \langle Q, \Sigma, q_0, \delta\rangle$, the language $\mathcal{L}_M$ consists of:

1. A two-place predicate symbol $Q_q(x, y)$ for every state $q \in Q$. Intuitively, $Q_q(\overline{m}, \overline{n})$ expresses "after n steps, M is in state q scanning the mth square."
2. A two-place predicate symbol $S_\sigma(x, y)$ for every symbol $\sigma \in \Sigma$. Intuitively, $S_\sigma(\overline{m}, \overline{n})$ expresses "after n steps, the mth square contains symbol σ."
3. A constant symbol o
4. A one-place function symbol $\prime$
5. A two-place predicate symbol $<$

For each number n there is a canonical term $\overline{n}$, the *numeral* for n, which represents it in $\mathcal{L}_M$. $\overline{0}$ is o, $\overline{1}$ is o', $\overline{2}$ is o'', and so on. More formally:

$$\overline{0} = \mathsf{o}$$
$$\overline{n+1} = \overline{n}'$$

The sentences describing the operation of the Turing machine M on input $w = \sigma_{i_1} \dots \sigma_{i_k}$ are the following:

1. Axioms describing numbers:

   a) A sentence that says that the successor function is injective: tms:und:rep: tm-rep-a

   $$\forall x \, \forall y \, (x' = y' \to x = y)$$

   b) A sentence that says that every number is less than its successor:

   $$\forall x \, x < x'$$

   c) A sentence that ensures that $<$ is transitive:

   $$\forall x \, \forall y \, \forall z \, ((x < y \wedge y < z) \to x < z)$$

   d) A sentence that connects $<$ and $=$:

   $$\forall x \, \forall y \, (x < y \to x \neq y)$$

2. Axioms describing the input configuration:

   a) After after 0 steps—before the machine starts—M is in the inital state q_0, scanning square 1:

   $$Q_{q_0}(\overline{1}, \overline{0})$$

   b) The first $k+1$ squares contain the symbols $\triangleright$, σ_{i_1}, …, σ_{i_k}:

   $$S_{\triangleright}(\overline{0}, \overline{0}) \wedge S_{\sigma_{i_1}}(\overline{1}, \overline{0}) \wedge \dots \wedge S_{\sigma_{i_k}}(\overline{n}, \overline{0})$$

   c) Otherwise, the tape is empty:

   $$\forall x \, (\overline{k} < x \to S_0(x, \overline{0}))$$

3. Axioms describing the transition from one configuration to the next:

   For the following, let $\varphi(x, y)$ be the conjunction of all sentences of the form

   $$\forall z \, (((z < x \vee x < z) \wedge S_\sigma(z, y)) \to S_\sigma(z, y'))$$

   where $\sigma \in \Sigma$. We use $\varphi(\overline{m}, \overline{n})$ to express "other than at square m, the tape after $n+1$ steps is the same as after n steps."

   a) For every instruction $\delta(q_i, \sigma) = \langle q_j, \sigma', R \rangle$, the sentence: tms:und:rep: rep-right

   $$\forall x \, \forall y \, ((Q_{q_i}(x, y) \wedge S_\sigma(x, y)) \to (Q_{q_j}(x', y') \wedge S_{\sigma'}(x, y') \wedge \varphi(x, y)))$$

   This says that if, after y steps, the machine is in state q_i scanning square x which contains symbol σ, then after $y+1$ steps it is scanning square $x+1$, is in state q_j, square x now contains σ', and every square other than x contains the same symbol as it did after y steps.

tms:und:rep:rep-left

b) For every instruction $\delta(q_i, \sigma) = \langle q_j, \sigma', L\rangle$, the sentence:

$$\begin{aligned}&\forall x\, \forall y\, ((Q_{q_i}(x', y) \wedge S_\sigma(x', y)) \rightarrow \\ &\qquad (Q_{q_j}(x, y') \wedge S_{\sigma'}(x', y') \wedge \varphi(x, y))) \wedge \\ &\forall y\, ((Q_{q_i}(\mathsf{o}, y) \wedge S_\sigma(\mathsf{o}, y)) \rightarrow \\ &\qquad (Q_{q_j}(\mathsf{o}, y') \wedge S_{\sigma'}(\mathsf{o}, y') \wedge \varphi(\mathsf{o}, y)))\end{aligned}$$

Take a moment to think about how this works: now we don't start with "if scanning square x ..." but: "if scanning square $x+1$ ..." A move to the left means that in the next step the machine is scanning square x. But the square that is written on is $x + 1$. We do it this way since we don't have subtraction or a predecessor function.

Note that numbers of the form $x + 1$ are 1, 2, ..., i.e., this doesn't cover the case where the machine is scanning square 0 and is supposed to move left (which of course it can't—it just stays put). That special case is covered by the second conjunction: it says that if, after y steps, the machine is scanning square 0 in state q_i and square 0 contains symbol σ, then after $y+1$ steps it's still scanning square 0, is now in state q_j, the symbol on square 0 is σ', and the squares other than square 0 contain the same symbols they contained ofter y steps.

tms:und:rep:rep-stay

c) For every instruction $\delta(q_i, \sigma) = \langle q_j, \sigma', N\rangle$, the sentence:

$$\begin{aligned}&\forall x\, \forall y\, ((Q_{q_i}(x, y) \wedge S_\sigma(x, y)) \rightarrow \\ &\qquad (Q_{q_j}(x, y') \wedge S_{\sigma'}(x, y') \wedge \varphi(x, y)))\end{aligned}$$

Let $\tau(M, w)$ be the conjunction of all the above sentences for Turing machine M and input w

In order to express that M eventually halts, we have to find a sentence that says "after some number of steps, the transition function will be undefined." Let X be the set of all pairs $\langle q, \sigma\rangle$ such that $\delta(q, \sigma)$ is undefined. Let $\alpha(M, w)$ then be the sentence

$$\exists x\, \exists y\, (\bigvee_{\langle q,\sigma\rangle \in X} (Q_q(x, y) \wedge S_\sigma(x, y)))$$

If we use a Turing machine with a designated halting state h, it is even easier: then the sentence $\alpha(M, w)$

$$\exists x\, \exists y\, Q_h(x, y)$$

expresses that the machine eventually halts.

tms:und:rep:prop:mlessk

Proposition und.2. *If $m < k$, then $\tau(M, w) \vDash \overline{m} < \overline{k}$*

Proof. Exercise. □

Problem und.1. Prove Proposition und.2. (Hint: use induction on $k - m$).

Photo Credits

Bibliography