

## Turing machine variants

- Goals:
  - Show that several “augmentations” of basic TMs are equivalent to ordinary TMs (i.e. decide/recognize the same languages)
  - Demonstrate technique (“simulation”) for proving this kind of claim
  - Allow ourselves to use these augmentations to define TMs for more complex tasks
  - Gain some evidence for Church-Turing thesis

### Canonical Example: TMs with “stay-put”

- A TM with “stay-put” (TMWSP) is like an ordinary TM, except the tape head can remain in the same position from one step to the next
- Formally, the transition function is

$$\delta : Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \mapsto Q \times \Gamma \times \{L, R, S\}$$

- **Claim:** TMWSPs recognize/decide the same class of languages as ordinary TMs
- Proof: there are two things to prove
  - Given a TM  $M$ , there is a TMWSP  $M'$  that recognizes/decides the same language as  $M$
  - Given a TMWSP  $M$ , there is a TM  $M'$  that recognizes/decides the same language as  $M$
- The first statement is obvious, since every TM is also a TMWSP (which never stays put)
- For the second statement: let  $M = (Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject}, \delta)$  be a TMWSP

- Define an ordinary TM  $M' = (Q', \Sigma', \Gamma', q'_0, q'_{accept}, q'_{reject}, \delta')$

$$\begin{aligned}
 Q' &= Q \cup \dot{Q}, & \text{where } \dot{Q} &= \{\dot{q} \mid q \in Q\} \\
 \Sigma' &= \Sigma \\
 \Gamma' &= \Gamma \\
 q'_0 &= q_0 \\
 q'_{accept} &= q_{accept} \\
 q'_{reject} &= q_{reject} \\
 \delta'(q, \sigma) &= \begin{cases} (p, \sigma', L), & \text{if } q \in Q \text{ and } \delta(q, \sigma) = (p, \sigma', L) \\ (p, \sigma', R), & \text{if } q \in Q \text{ and } \delta(q, \sigma) = (p, \sigma', R) \\ (\dot{p}, \sigma', R), & \text{if } q \in Q \text{ and } \delta(q, \sigma) = (p, \sigma', S) \\ (p, \sigma, L), & \text{if } q = \dot{p} \in \dot{Q} \end{cases}
 \end{aligned}$$

- Intuition: whenever  $M$  stays put,  $M'$  moves right, then left
- Then  $L(M') = L(M)$ ; also,  $M'$  halts on input  $w$  iff  $M$  halts on input  $w$

### Simulation

- Simulation is a technique for proving that one model of computation can compute anything that another model of computation can
- In the above example, the models were TMWSPs and TMs
- In general, the models do not need to be related to TMs
- To simulate model  $M$  with model  $M'$ :
  - Define a way of representing a configuration  $C$  of  $M$  in the memory of  $M'$  (call this  $\langle C \rangle$ )
  - Describe how  $M'$  can compute  $\langle I_M(w) \rangle$  from  $I_{M'}(w)$ , i.e. how to get a representation of the initial configuration of  $M$  on input  $w$
  - Show how to simulate a single step of  $M$  on  $M'$ , i.e. if  $M'$  has  $\langle C \rangle$  in memory, how to compute  $\langle C' \rangle$  such that  $C \vdash_M C'$

### Multi-tape Turing machines

- A  $k$ -tape TM is like an ordinary TM with
  - $k$  tapes
  - $k$  independent tape-heads
  - The input is written on the first tape as usual; all other tapes start out blank

- A  $k$ -tape TM still has only one finite state control
- Formally, the transition function is

$$\delta : Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma^k \mapsto Q \times \Gamma^k \times \{L, R\}^k$$

- **Theorem:** For all  $k \geq 1$ ,  $k$ -tape TMs decide/recognize the same class of languages as ordinary TMs
- Proof: there are two things to prove, namely
  - Given an ordinary TM  $M$ , there is a  $k$ -tape TM  $M'$  that decides/recognizes the same language as  $M$
  - Given a  $k$ -tape TM  $M$ , there is an ordinary TM  $M'$  that decides/recognizes the same language as  $M$
- The first statement is obvious: just let  $M'$  be a  $k$ -tape TM that ignores the last  $k - 1$  tapes, i.e. whose transition function restricted to the first components is identical to that of  $M$
- For the second statement, we will show how to simulate a  $k$ -tape TM  $M$  with an ordinary TM  $M'$
- The first question is, how will we represent a configuration of  $M$  on a single tape?
  - Many ways to do this; here is one way
  - Store the contents of each tape sequentially in memory, separated by a new symbol  $*$
  - Mark the start of the first tape by  $\rightarrow$  and the end of the last tape by  $\leftarrow$
  - Use a dotted symbol to indicate the position of a tape-head
  - For example:  $\rightarrow aab\dot{a} * b\dot{a}b\dot{a} \leftarrow$  is a representation of a configuration of a 2-tape TM, which has  $\dots \sqcup \sqcup aaba \sqcup \sqcup \dots$  on the first tape (and the first tape head is scanning the last  $a$ ), and  $\dots \sqcup \sqcup baba \sqcup \sqcup \dots$  on the second tape (and the second tape-head is scanning the second  $b$ )
  - The current state of  $M$  will be remembered as part of the state of  $M'$
- Next, on input  $w$  we need to write the representation  $I_M(w)$  on the tape
  - Let  $w = w_1 \dots w_n$
  - Then we need to write  $\rightarrow \dot{w}_1 w_2 \dots w_n * \dot{\sqcup} * \dots * \dot{\sqcup} \leftarrow$
  - This is easily done: on input  $w$ , replace the first symbol of  $w$  with its dotted version; then move left and write  $\rightarrow$ ; then move right until a  $\sqcup$  is encountered, and write  $*\dot{\sqcup}$   $k - 1$  times, followed by a  $\leftarrow$

- Finally, we need to describe how to simulate one step of  $M$ 
  - Move left until  $\rightarrow$  is encountered
  - Move right until  $\leftarrow$  is encountered, remembering the dotted symbols  $\overset{\bullet}{a}_1, \dots, \overset{\bullet}{a}_k$  that are encountered along the way
  - Note that since there are only finitely many possibilities, these can be remembered as part of the state
  - Let  $q$  be the current state of  $M$  (we have remembered this in our state as well)
  - Suppose that  $\delta_M(q, a_1, \dots, a_k) = (q', b_1, \dots, b_k, D_1, \dots, D_k)$ , where  $D_i \in \{L, R\}$
  - Move left until  $\rightarrow$  is found
  - Move right, and count the number of  $*$  symbols that are found
  - When the dotted symbol  $\overset{\bullet}{a}_i$  is found after  $i - 1$   $*$  symbols, replace it with  $b_i$  and move the dot one square in the direction  $D_i$ , **unless** the next square in the direction  $D_i$  contains  $\rightarrow$ ,  $\leftarrow$ , or  $*$ .
  - If this occurs, then shift every symbol from the current symbol to  $\leftarrow$  (inclusive) to the right by one cell, and write  $\square$  in the empty cell that is created by this process
  - Change the remembered state of  $M$  to  $q'$
  - If  $q' = q_{accept}$ , then accept; if  $q' = q_{reject}$  then reject
- Thus  $M'$  will accept  $w$  iff  $M$  accepts  $w$ , and  $M'$  will reject  $w$  iff  $M$  rejects  $w$ ; the theorem follows.