

# Notes on Inference Devices

Santa Fe Institute

Edward G. Huang

Summer 2018

## 1 Notation and Definitions

This manuscript utilizes standard notation taken from set theory and vector algebra. We clarify notation specific to Turing Machine theory and inference devices.

### 1.1 Turing Machines

- $\mathbb{B}^*$  The space of all finite bit strings.
- $\Lambda$  Symbol alphabet of a Turing Machine.
- $\sigma$  A symbol on a Turing Machine tape.
- $Q$  Set of finite states of a Turing Machine.
- $\Delta$  Transition function of a Turing Machine.
- $k$  Number of tapes of a Turing Machine. The first tape is assumed to be read-only.
- $\eta$  Non-halting state of a Turing Machine.

### 1.2 Inference Devices

- $U$  Set of possible histories of the universe.
- $u$  A history of the universe in  $U$ .
- $X$  Setup function of an ID that maps  $U \rightarrow X(U)$ . A binary question concerning  $\Gamma(u)$ .
- $x$  A binary question and a member of image  $X(U)$ .
- $Y$  Single-valued conclusion function of an ID that maps  $U \rightarrow \{-1, 1\}$ . A binary answer of an ID for  $X(u) = x$ .
- $y$  A single-valued answer, and member of image  $Y(U) = \{0, 1\}$ .
- $\Gamma$  A function of the actual values of a physical variable over  $U$ , equivalent to  $\Gamma(u) = S(t_i)(u)$ .
- $\gamma$  Possible value of a physical variable, a member of the image  $\Gamma(U)$ .

$\delta$  Probe of any variable  $V$  parameterized by  $v \in V$  such that :

$$\delta_v(v') = \begin{cases} 1 & \text{if } v = v' \\ -1 & \text{otherwise} \end{cases}$$

$\wp$  Set of probes over  $\Gamma(U)$ .

$\mathcal{D} = (X, Y)$  An inference device, consisting of functions  $X$  and  $Y$ .

$\bar{F}$  Inverse. Given a function  $F$  over  $U$ ,  $F^{-1} = \bar{F} \equiv \{\{u : F(u) = f\} : f \in F(U)\}$ .

$>$  Weak inference: a device  $\mathcal{D}$  weakly infers  $\Gamma$  iff  $\forall \gamma \in \Gamma(U), \exists x \in X(U)$  s.t.  $\forall u \in U$ ,  $X(u) = x \implies Y(u) = \delta_\gamma(\Gamma(u))$ .

$\gg$  Strong inference: a device  $(X, Y)$  strongly infers a functions  $(S, T)$  over  $U$  iff  $\forall \delta \in \wp(T)$  and all  $s \in S(U)$ ,  $\exists x$  such that  $X(u) = x \implies S(u) = s, Y(u) = \delta(T(u))$ .

## 2 Turing Machines

### 2.1 Deterministic Turing Machines

Arora and Barak denote a Turing Machine (TM) as  $T = (\Lambda, Q, \Delta)$  containing:

1. An *alphabet*  $\Lambda$  of a finite set of symbols that  $T$ 's tapes can contain. We assume that  $\Lambda$  contains a special blank symbol  $B$ , start symbol  $S$ , and the symbols 0 and 1.
2. A finite set  $Q$  of possible states that  $T$ 's register can be in. We assume that  $Q$  contains a special start state  $q_s$  and a special halt state  $q_h$ .
3. A transition function function  $\Delta : Q \times \Lambda^k \rightarrow Q \times \Lambda^{k-1} \times \{L, S, R\}^k$ , where  $k \geq 2$ , describing the rules  $T$  use in performing each step. The set  $\{L, S, R\}$  denote the actions *Left*, *Stay*, and *Right*, respectively.

Suppose  $T$  is in state  $q \in Q$  and  $(\sigma_1, \sigma_2, \dots, \sigma_k)$  are the symbols on the  $k$  tapes. Then  $\Delta(q, (\sigma_1, \dots, \sigma_k)) = (q', (\sigma'_2, \dots, \sigma'_k), z)$  where  $z \in \{L, S, R\}^k$  and at the next step the  $\sigma$  symbols in the last  $k - 1$  tapes will be replaced by the  $\sigma'$  symbols, the machine will be in state  $q$ , and the  $k$  heads will move *Left*, *Right* or *Stay*. This is illustrated in Figure 2.1.

**Figure 2.1. The transition function  $\Delta$  for a  $k$ -tape Turing Machine**

$(q, (\sigma_1, \dots, \sigma_k))$				$(q', (\sigma'_2, \dots, \sigma'_k), z)$				
Input symbol	Work/output symbol read	...	Current state	New work/output tape symbol	...	Move work/output tape	...	New state
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$\sigma_1$	$\sigma_i$	$\ddots$	$q$	$\sigma'_i$	$\ddots$	$z_i$	$\ddots$	$q'$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$

*Remark:*  $\Lambda$  can be reduced to  $\mathbb{B} = \{0, 1\}$  and  $k$  can be reduced to 1 without loss of computational power. Then, any Turing Machine can be expressed as a partial recursive function mapping  $\mathbb{B}^* \rightarrow \mathbb{B}^* \cup \eta$ , where  $\eta$  is the undefined non-halting output. Since  $|\mathbb{B}^* \times \mathbb{B}^* \cup \eta| = |\mathbb{N} \times \mathbb{N}| = |\mathbb{N}|$ , the set of all Turing Machines is countably infinite.

## 2.2 Non-deterministic Turing Machines

Non-deterministic Turing Machines (NDTM) differ from deterministic Turing Machines by having two transition functions  $\Delta_0, \Delta_1$  and a special state  $q_{accept}$ . From Arora and Barak:

When a NDTM  $M$  computes a function, we envision that at each computational step  $M$  makes an arbitrary choice as to which of its two transition functions to apply. For every input  $x$ , we say that  $M(x) = 1$  if there exists some sequence of these choices (which we call nondeterministic choices of  $M$ ) that would make  $M$  reach  $q_{accept}$  on input  $x$ . Otherwise - if every sequence of choices makes  $M$  halt without reaching  $q_{accept}$  - then we say that  $M(x) = 0$ .

## 3 Inference of Turing Machines

In the next three examples we examine strong inference of integer-valued functions.

**Example 3.1** Let  $T(U) = \{0, 1\}$  and  $S(U) = \{0, 1, 2\}$ . We construct  $(X, Y)$  in the table at the left such that it strongly infers  $(S, T)$ . The right table indicates  $x$  for each  $s, \delta$  such that the definition of strong inference is satisfied:

$u$	$X(u)$	$Y(u)$	$S(u)$	$T(u)$
1	1	1	0	0
2	2	-1	0	0
3	3	1	1	0
4	4	-1	1	0
5	5	1	2	1
6	6	-1	2	1

$s \setminus \delta$	$\delta_0$	$\delta_1$
0	1	2
1	3	4
2	6	5

**Example 3.2** Let  $T(U) = \{1, 2, 3\}$  and  $S(U) = \{1, 2, 3, 4, 5\}$ . Again, we construct  $(X, Y)$  in the table at the left such that it strongly infers  $(S, T)$ . The right table indicates  $x$  for each  $s, \delta$  such that the definition of strong inference is satisfied:

$u$	$X(u)$	$Y(u)$	$S(u)$	$T(u)$
1	1	1	1	1
2	2	-1	1	1
3	3	1	2	1
4	4	-1	2	1
5	5	1	3	2
6	6	-1	3	2
7	7	1	4	2
8	8	-1	4	2
9	9	1	5	3
10	10	-1	5	3

$s \setminus \delta$	$\delta_1$	$\delta_2$	$\delta_3$
1	1	2	2
2	3	4	4
3	6	5	6
4	8	7	8
5	10	10	9

**Example 3.3** Let  $T(U) = \{1, 2, 3\}$  and  $S(U) = \{1, 2\}$ . In this example, the inferred function  $f : S \rightarrow T, f(s) = T(S^{-1}(s))$  is not single-valued.

$u$	$X(u)$	$Y(u)$	$S(u)$	$T(u)$
1	1	-1	1	1
2	2	-1	1	2
3	3	-1	1	3
4	4	-1	2	1
5	5	-1	2	2

$s \setminus \delta$	$\delta_1$	$\delta_2$	$\delta_3$
1	2	1	1
2	5	4	4

**Theorem** A deterministic Turing Machine  $(\Lambda, Q, \Delta)$  can be strongly inferred by a device if

$$\forall s \in S(U), |S^{-1}(s)| \geq 2.$$

*This holds for both the representation of a Turing Machine as a partial recursive function and the representation as an update function.*

**Proof** We first examine the partial function case. Let  $f$  be the partial recursive function that describes the given Turing Machine tuple. Let  $U := \mathbb{N}$ . Choose any convenient single valued function  $S : U \rightarrow \mathbb{B}^*$  and define  $T : U \rightarrow \mathbb{B}^* \cup \eta$  given by  $T(u) = f(S(u))$  as the single-valued function

mapping  $U$  to the halting and non-halting outputs of  $f$ . Then  $f$  can be written as the single-valued mapping  $S \rightarrow T$  given by  $f(s) = T(S^{-1}(s))$ .

Let  $V_s = \{u : S^{-1}(s) = u\}$  for a value of  $s \in S(U)$ . Define  $n(s) = |V_s|$ . Enumerate the elements of  $V_s$  as  $v_s^1, v_s^2, \dots, v_s^{n(s)}$ . We also enumerate the elements of  $S(U)$  as  $s_1, s_2, \dots$ . Then define  $X$  and  $Y$  as follows:

$$X(v_{s_j}^i) = i + \sum_1^{j-1} n(s_j) : i \in 1, \dots, n(s) \quad Y(v_{s_j}^i) = \begin{cases} 1 & \text{if } v_s^i = v_s^1 \\ -1 & \text{otherwise} \end{cases}$$

The condition  $|S^{-1}(s)| \geq 2$  is required to guarantee  $Y(v) = -1$  for some  $v_s^i$ . For each pair  $(s, \delta_{t \in T(U)})$  choose  $x_1$  if  $t = T(v_1)$  or otherwise choose  $x_2$  to force  $S(u) = s$  and  $Y(u) = \delta_t(T(u))$ . Since the choice of  $s$  was arbitrary, this holds for all  $(s, \delta_t)$  pairs.

Now consider the update function that describes the given Turing Machine. The update function is written as  $\Delta$  mapping  $Q \times \Lambda^k \rightarrow Q \times \Lambda^{k-1} \times \{L, S, R\}^k$ ,  $k \geq 2$ . Consider a convenient single-valued function  $S : U \rightarrow Q \times \Lambda^k$  representing the possible inputs for a Turing Machine and a corresponding single-valued  $T : U \rightarrow Q \times \Lambda^{k-1} \times \{L, S, R\}^k$  given by  $T(u) = \Delta(S(u))$ . Then  $\Delta$  can be written as the single-valued function  $\Delta(s) = T(S^{-1}(s))$ .

We proceed analogously to the partial function portion of the proof. Define  $V$ ,  $X$ , and  $Y$  as described in the preceding paragraphs. For each pair  $(s, \delta_{t \in T(U)})$  choose  $x_1$  if  $t = T(v_1)$  or otherwise choose  $x_2$  to force  $S(u) = s$  and  $Y(u) = \delta_t(T(u))$ . Since the choice of  $s$  was arbitrary, this holds for all  $(s, \delta_t)$  pairs.  $\square$

## 4 Inference Complexity

**Definition** Let  $\mathcal{D}$  be an inference device and  $\Gamma$  be a function over  $U$  where  $X(U)$  and  $\Gamma(U)$  are countable and  $\mathcal{D} > \Gamma$ . Let the **size** of  $\gamma \in \Gamma(U)$  be written as  $\mathcal{M}_{\mu, \Gamma(\gamma)} = -\ln[\int_{\Gamma^{-1}(\gamma)} d\mu(u)1]$ . Then the **inference complexity** of  $\Gamma$  with respect to  $\mathcal{D}$  and measure  $\mu$  is defined as:

$$\mathcal{C}_\mu(\Gamma; \mathcal{D}) \triangleq \sum_{\delta \in \wp(\Gamma)} \min_{x: X=x \Rightarrow Y=\delta(\Gamma)} [\mathcal{M}_{\mu, X}(x)]$$

*Remark:* This is only a working definition and may be revised depending on its behavior.