It would be unfortunate for any rigid criterion to inhibit the practical development of algorithms which are either not known or known not to conform nicely to the criterion. Many of the best algorithmic idea known today would suffer by such theoretical pedantry. . . . However, if only to motivate the search for good, practical algorithms, it is important to realize that it is mathematically sensible even to question their existence. For one thing the task can then be described in terms of concrete conjectures.

1.7 PROOF OF THEOREM 1.9: UNIVERSAL SIMULATION IN $O(T \log T)$-TIME

We now show how to prove Theorem 1.9 as stated. That is, we show a universal TM $\mathcal{U}$ such that given an input x and a description of a TM M that halts on x within T steps, $\mathcal{U}$ outputs $M(x)$ within $O(T \log T)$ time (where the constants hidden in the O notation may depend on the parameters of the TM M being simulated).

The general structure of $\mathcal{U}$ will be as in Section 1.4.1. $\mathcal{U}$ will use its input and output tape in the same way M does and will also have extra work tapes to store M's transition table and current state and to encode the contents of M's work tapes. The main obstacle we need to overcome is that we cannot use Claim 1.6 to reduce the number of M's work tapes to one, since Claim 1.6 introduces too much overhead in the simulation. Therefore, we will show a different way to encode all of M's work tapes in a single tape of $\mathcal{U}$, which we call the *main* work tape of $\mathcal{U}$.

Let k be the number of tapes that M uses (apart from its input and output tapes) and Γ its alphabet. Following the proof of Claim 1.5, we may assume that $\mathcal{U}$ uses the alphabet Γ^k (as this can be simulated with a overhead depending only on $k, |\Gamma|$). Thus we can encode in each cell of $\mathcal{U}$'s main work tape k symbols of Γ, each corresponding to a symbol from one of M's tapes. This means that we can think of $\mathcal{U}$'s main work tape not as a single tape but rather as k *parallel tapes*; that is, we can think of $\mathcal{U}$ as having k tapes with the property that in each step all their read-write heads go in unison either one location to the left, one location to the right, or stay in place. While we can easily encode the contents of M's k work tapes in $\mathcal{U}$'s k parallel tapes, we still have to deal with the fact that M's k read-write heads can each move independently to the left or right, whereas $\mathcal{U}$'s parallel tapes are forced to move together. Paraphrasing the famous saying, our strategy to handle this is: *"If the head cannot go to the tape locations then the locations will go to the head."*

That is, since we can not move $\mathcal{U}$'s read-write head in different directions at once, we simply move the parallel tapes "under" the head. To simulate a single step of M, we shift all the nonblank symbols in each of these parallel tapes until the head's position in these parallel tapes corresponds to the heads' positions of M's k tapes. For example, if $k = 3$ and in some particular step M's transition function specifies the movements L, R, R, then $\mathcal{U}$ will shift all the nonblank entries of its first parallel tape one cell to the right, and shift the nonblank entries of its second and third tapes one cell to the left (see Figure 1.8). $\mathcal{U}$ can easily perform such shifts using an additional "scratch" work tape.

The approach above is still not good enough to get $O(T \log T)$-time simulation. The reason is that there may be as many as T nonblank symbols in each parallel tape, and so each shift operation may cost $\mathcal{U}$ as much as T operations per each step of M, resulting in $\Theta(T^2)$-time simulation. We will deal with this problem by encoding the information

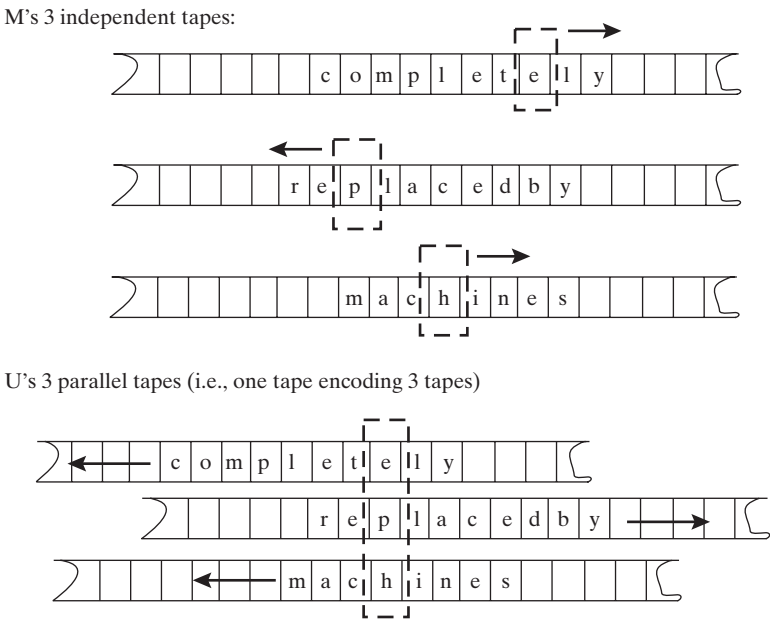

Figure 1.8. Packing k tapes of M into one tape of $\mathcal{U}$. We consider $\mathcal{U}$'s single work tape to be composed of k parallel tapes, whose heads move in unison, and hence we shift the contents of these tapes to simulate independent head movement.

on the tapes in a way that allows us to amortize the work of performing a shift. We will ensure that we do not need to move all the nonblank symbols of the tape in each shift operation. Specifically, we will encode the information in a way that allows half of the shift operations to be performed using $2c$ steps, for some constant c, a quarter of them using $4c$ steps, and more generally 2^{-i} fraction of the operations will take $2^i c$ steps, leading to simulation in roughly $cT \log T$ time (see below). (This kind of analysis is called *amortized analysis* and is widely used in algorithm design.)

Encoding M's tapes on $\mathcal{U}$'s tape

To allow more efficient shifts we encode the information using "buffer zones": Rather than having each of $\mathcal{U}$'s parallel tapes correspond exactly to a tape of M, we add a special kind of blank symbol $\boxtimes$ to the alphabet of $\mathcal{U}$'s parallel tapes with the semantics that this symbol is ignored in the simulation. For example, if the nonblank contents of M's tape are 010, then this can be encoded in the corresponding parallel tape of $\mathcal{U}$ not just by 010 but also by $0\boxtimes 01$ or $0\boxtimes\boxtimes 1\boxtimes 0$ and so on.

For convenience, we think of $\mathcal{U}$'s parallel tapes as infinite in both the left and right directions (this can be easily simulated with minimal overhead: see Claim 1.8). Thus, we index their locations by $0, \pm 1, \pm 2, \ldots$. Normally we keep $\mathcal{U}$'s head on location 0 of these parallel tapes. We will only move it temporarily to perform a shift when, following our general approach, we simulate a left head movement by shifting the tape to the right and vice versa. At the end of the shift, we return the head to location 0.

We split each of $\mathcal{U}$'s parallel tapes into *zones* that we denote by $R_0, L_0, R_1, L_1, \ldots$ (we'll only need to go up to $R_{\log T}, L_{\log T}$). The cell at location 0 is not

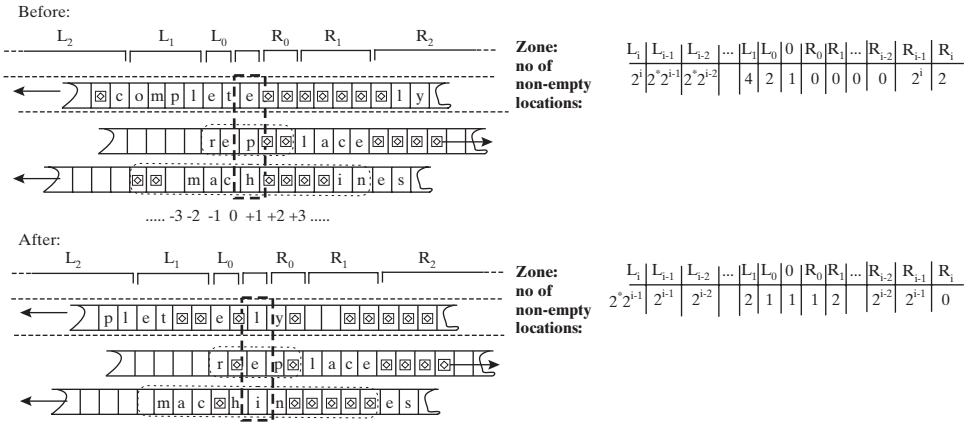

Figure 1.9. Performing a shift of the parallel tapes. The left shift of the first tape involves zones $R_0, L_0, R_1, L_1, R_2, L_2$, the right shift of the second tape involves only R_0, L_0, while the left shift of the third tape involves zones R_0, L_0, R_1, L_1. We maintain the invariant that each zone is either empty, half-full, or full and that the total number of nonempty cells in $R_i \cup L_i$ is $2 \cdot 2^i$. If before the left shift zones $L_0, \ldots, L_{i-1}$ were full and L_i was half-full (and so $R_0, \ldots, R_{i-1}$ were full and R_i half-full), then after the shift zones $R_0, L_0, \ldots, R_{i-1}, L_{i-1}$ will be half-full, L_i will be full and R_i will be empty.

at any zone. Zone R_0 contains the two cells immediately to the right of location C (i.e., locations $+1$ and $+2$), while Zone R_1 contains the four cells $+3, +4, +5, +6$. Generally, for every $i \geq 1$, Zone R_i contains the $2 \cdot 2^i$ cells that are to the right of Zone R_{i-1} (i.e., locations $[2^{i+1} - 1, \ldots, 2^{i+2} - 2]$). Similarly, Zone L_0 contains the two cells indexed by -1 and -2, and generally Zone L_i contains the cells $[-2^{i+2} + 2, \cdots, -2^{i+1} + 1]$. We shall always maintain the following invariants:

- Each of the zones is either *empty*, *full*, or *half-full* with non-⊠ symbols. That is, the number of symbols in zone R_i that are not ⊠ is either 0, 2^i, or $2 \cdot 2^i$ and the same holds for L_i. (We treat the ordinary □ symbol the same as any other symbol in Γ, and in particular a zone full of □'s is considered full.)
  We assume that initially all the zones are half-full. We can ensure this by filling half of each zone with ⊠ symbols in the first time we encounter it.
- The total number of non-⊠ symbols in $R_i \cup L_i$ is $2 \cdot 2^i$. That is, either R_i is empty and L_i is full, or R_i is full and L_i is empty, or they are both half-full.
- Location 0 always contains a non-⊠ symbol.

Performing a shift

The advantage in setting up these zones is that now when performing the shifts, we do not always have to move the entire tape, but we can restrict ourselves to only using some of the zones. We illustrate this by showing how $\mathcal{U}$ performs a left shift on the first of its parallel tapes (see also Figure 1.9):

1. $\mathcal{U}$ finds the smallest i_0 such that R_{i_0} is not empty. Note that this is also the smallest i_0 such that L_{i_0} is not full. We call this number i_0 the *index* of this particular shift.
2. $\mathcal{U}$ puts the leftmost non-⊠ symbol of R_{i_0} in position 0 and shifts the remaining leftmost $2^{i_0} - 1$ non-⊠ symbols from R_{i_0} into the zones $R_0, \ldots, R_{i_0-1}$ filling up exactly half the

symbols of each zone. Note that there is exactly room to perform this since all the zones $R_0, \ldots, R_{i_0-1}$ were empty and indeed $2^{i_0} - 1 = \sum_{j=0}^{i_0-1} 2^j$.

3. $\mathcal{U}$ performs the symmetric operation to the left of position 0. That is, for j starting from $i_0 - 1$ down to 0, $\mathcal{U}$ iteratively moves the $2 \cdot 2^j$ symbols from L_j to fill half the cells of L_{j+1}. Finally, $\mathcal{U}$ moves the symbol originally in position 0 (modified appropriately according to M's transition function) to L_0.
4. At the end of the shift, all of the zones $R_0, L_0, \ldots, R_{i_0-1}, L_{i_0-1}$ are half-full, R_{i_0} has 2^{i_0} fewer non-⊠ symbols, and L_i has 2^i additional non-⊠ symbols. Thus, our invariants are maintained.
5. The total cost of performing the shift is proportional to the total size of all the zones involved $R_0, L_0, \ldots, R_{i_0}, L_{i_0}$. That is, $O(\sum_{j=0}^{i_0} 2 \cdot 2^j) = O(2^{i_0})$ operations.

After performing a shift with index i the zones $L_0, R_0, \ldots, L_{i-1}, R_{i-1}$ are half-full, which means that it will take at least $2^i - 1$ left shifts before the zones $L_0, \ldots, L_{i-1}$ become empty or at least $2^i - 1$ right shifts before the zones $R_0, \ldots, R_{i-1}$ become empty. In any case, once we perform a shift with index i, the next $2^i - 1$ shifts of that particular parallel tape will all have index less than i. This means that for every one of the parallel tapes, at most a $1/2^i$ fraction of the total number of shifts have index i. Since we perform at most T shifts, and the highest possible index is $\log T$, the total work spent in shifting $\mathcal{U}$'s k parallel tapes in the course of simulating T steps of M is

$$O(k \cdot \sum_{i=1}^{\log T} \frac{T}{2^{i-1}} 2^i) = O(T \log T) \,. \blacksquare$$

WHAT HAVE WE LEARNED?

- There are many equivalent ways to mathematically model computational processes; we use the standard Turing machine formalization.
- Turing machines can be represented as strings. There is a *universal* TM that can simulate (with small overhead) any TM given its representation.
- There exist functions, such as the Halting problem, that cannot be computed by any TM regardless of its running time.
- The class **P** consists of all decision problems that are solvable by Turing machines in polynomial time. We say that problems in **P** are efficiently solvable.
- Low-level choices (number of tapes, alphabet size, etc..) in the definition of Turing machines are immaterial, as they will not change the definition of **P**.

CHAPTER NOTES AND HISTORY

Although certain algorithms have been studied for thousands of years, and some forms of computing devices were designed before the twentieth century (most notably Charles Babbage's difference and analytical engines in the mid 1800s), it seems fair to say that the foundations of modern computer science were only laid in the 1930s.