Part 3: Decidable and undecidable languages

Theme: Turing machines as the most general model of computation.

1 Turing machines

We reserve a special symbol \( \sqcup \), called the blank symbol.

A Turing machine (TM) is a system \( \mathcal{M} = (\Sigma, \Gamma, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta) \), where each component is as follows.

- \( \Sigma \) is a finite alphabet, called the input alphabet, where \( \sqcup \notin \Sigma \).
- \( \Gamma \) is a finite alphabet, called the tape alphabet, where \( \Sigma \subseteq \Gamma \) and \( \sqcup \in \Gamma \).
- \( Q \) is a finite set of states.
- \( q_0 \in Q \) is the initial state.
- \( q_{\text{acc}}, q_{\text{rej}} \in Q \) are two special states called the accept and reject states, respectively.
- \( \delta : Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\} \times \Gamma \to Q \times \Gamma \times \{\text{Left, Right}\} \) is the transition function.

Intuitively, the intuitive meaning of \( \delta(p, a) = (q, b, \alpha) \) is as follows. When the head reads a symbol \( a \), if \( \mathcal{M} \) is in state \( p \), it “writes” symbol \( b \) on top of \( a \), enters state \( q \), and the head moves left, if \( \alpha = \text{Left} \), or moves right, if \( \alpha = \text{Right} \).

To describe how a TM computes, we need a few terminologies. A configuration of \( \mathcal{M} \) is a string \( C \) from \( (Q \cup \Gamma)^* \) which contains exactly one symbol from \( Q \). We call such symbol the state of \( C \). Intuitively, a configuration \( C = a_1 \cdots a_{i-1} pa_i \cdots a_m \) means the content of the tape \( a_1 \cdots a_m \) and that \( \mathcal{M} \) is in state \( p \) with the head in position \( i \).

On input word \( w \in \Sigma^* \), the initial configuration of \( \mathcal{M} \) on \( w \) is the string \( qow \). A configuration is called accepting, if it contains \( q_{\text{acc}} \), and it is called rejecting, if it contains \( q_{\text{rej}} \). A halting configuration is either an accepting or a rejecting configuration.

Let \( C = a_1 \cdots a_{i-1} pa_i \cdots a_m \) be a configuration, where \( a_1, \ldots, a_m \in \Gamma \) and \( p \in Q \) such that \( p \neq q_{\text{acc}}, q_{\text{rej}} \). The transition \( \delta \) yields the subsequent configuration \( C' \), denoted by \( C \vdash C' \), as follows.

- If \( \delta(p, a_i) = (q, b, \text{Left}) \) and \( i \geq 2 \), then \( C' = a_1 \cdots a_{i-2} qa_{i-1}ba_{i+1} \cdots a_m \).
- If \( \delta(p, a_i) = (q, b, \text{Right}) \) and \( i \leq m - 1 \), then \( C' = a_1 \cdots a_{i-1} b qa_{i+1} \cdots a_m \).
- If \( \delta(p, a_i) = (q, b, \text{Right}) \) and \( i = m \), then \( C' = a_1 \cdots a_{m-1} b q\sqcup \).

The run of \( \mathcal{M} \) on \( w \) is the (possibly infinite) sequence:

\[
C_0 \vdash C_1 \vdash C_2 \vdash \cdots \tag{1}
\]

where \( C_0 \) is the initial configuration of \( \mathcal{M} \) on \( w \).

\( \mathcal{M} \) stops when it reaches a configuration \( C = a_1 \cdots a_{i-1} pa_i \cdots a_m \) where there is no \( C' \) where \( C \vdash C' \). For such case, we say that \( \mathcal{M} \) halts on \( w \) in configuration \( C \) and \( C \) must satisfy one of the two conditions below holds.

- \( C \) is a halting configuration.
- \( i = 1 \) and \( \delta(p, a_i) = (q, b, \text{Left}) \), i.e., the head still moves left when it is already on the leftmost position of the tape and “falls” off the tape.

[1]
If $M$ halts in an accepting configuration, then we say that $M$ accepts $w$. If it halts in a rejecting configuration, then we say that $M$ rejects $w$. We denote by $L(M)$ the language that consists of all the words accepted by $M$. Formally,

$$L(M) := \{w \mid M \text{ accepts } w\}$$

Proposition 3.1 below states that we can always assume that when a Turing machine halts, it halts in either an accepting or rejecting configuration.

**Proposition 3.1** For every Turing machine $M$, there is another Turing machine $M'$ such that for every $w \in \Sigma^*$ the following holds.

- $M$ accepts $w$ if and only if $M'$ accepts $w$.
- $M$ rejects $w$ if and only if $M'$ rejects $w$.

For all $w$ neither accepted nor rejected by $M$, $M'$ does not halt on $w$.

In other words, Proposition 3.1 implies that we can assume that on any input, the head of $M'$ never falls off the tape.

**Some important terminologies.**

- We say that $M$ recognizes a language $L$, if:
  
  (i) for every word $w \in L$, $M$ accepts $w$;
  
  (ii) for every word $w \notin L$, $M$ does not accept $w$.

  Note that $M$ does not accept $w$ can have two meanings: either $M$ rejects $w$, or $M$ does not halt on $w$.

- We say that $M$ decides a language $L$, if for every word $w$,
  
  (i) if $w \in L$, $M$ accepts $w$,
  
  (ii) if $w \notin L$, $M$ rejects $w$.

  Note that this implies $M$ halts on every word $w \in \Sigma^*$.

- A language $L$ is recognizable/recursively enumerable (r.e.), if there is a TM $M$ that recognizes $L$.

- A language $L$ is decidable/recursive, if there is a TM $M$ that decides $L$.

  Otherwise, it is called undecidable.

2 Multi-tape Turing machines

A multi-tape Turing machine is a Turing machine that has a few tapes. On each tape, the Turing machine has one head. Formally, it is defined as follows. Let $k \geq 1$. A $k$-tape Turing machine is $M = (\Sigma, \Gamma, Q, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta)$, where $\delta$ is a function

$$\delta : (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma^k \to Q \times \Gamma^k \times \{\text{Left, Right, Stay}\}^k$$

As before, an element of $\delta$ is written in the form:

$$(q, a_1, \ldots, a_k) \to (p, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k).$$
Intuitively, it means that if the TM is in state $q$, and on each $i = 1, \ldots, k$, the head on tape $i$ is reading $a_i$, then it enters state $p$, and for $i = 1, \ldots, k$, the head on tape $i$ writes the symbol $b_i$ and moves according to $\alpha_i$.

A configuration of $M$ is of the form $(q, u_1, \ldots, u_k)$, where $q \in Q$ and each $u_i$ is a string over $\Gamma \cup \{\bullet\}$ and the symbol $\bullet$ appears exactly once in each $u_i$. The symbol $\bullet$ is to denote the position of the head.

The input is always written in the first tape. All the other tapes are initially blank. Formally, the initial configuration on input $w$ is $(q_0, w, \bullet, \ldots, \bullet)$.

The notion of "one step computation" $C \vdash C'$ is defined similarly as in the standard Turing machine. Likewise, the conditions of acceptance and rejection are defined as when the Turing machines enters the accepting and rejecting states, respectively.

**Theorem 3.2** For every language $L$, the following holds.

- If $L$ is recognized by a $k$-tape TM $M$, then there is a single tape TM $M'$ that recognizes $L$.
- If $L$ is decided by a $k$-tape TM $M$, then there is a single tape TM $M'$ that decides $L$.

### 3 Non-deterministic Turing machines

A non-deterministic Turing machine (NTM) $M = \langle \Sigma, \Gamma, Q, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta \rangle$ is defined as the standard Turing machine, with the exception that $\delta$ is now a relation:

$$\delta \subseteq (Q - \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \times Q \times \Gamma \times \{\text{Left}, \text{Right}, \text{Stay}\}$$

As before, we write an element of $\delta$ is in the form:

$$(q, a) \rightarrow (p, b, \alpha).$$

The initial configuration of $M$ on input word $w$ is $q_0w$. For two configurations $C, C'$, the notion of "one step computation" $C \vdash C'$ is defined similarly as in the standard Turing machine. A run of $M$ on input $w$ is a sequence:

$$C_0 \vdash C_1 \vdash \cdots,$$

where $C_0$ is the initial configuration on $w$. A run is accepting/rejecting, if it ends up in an accepting/rejecting configurations, respectively. However, due to non-determinism, for each $C$ there can be a few configuration $C'$ such that $C \vdash C'$, thus, there can be many runs. Some are accepting, some are rejecting, and some other do not halt.

**Important definitions.**

- An NTM $M$ accepts $w$, if there is an accepting run of $M$ on $w$.
- An NTM $M$ rejects $w$, if all runs of $M$ on $w$ are rejecting.
- A language $L$ is decided by an NTM $M$, if
  - for every $w \in L$, $M$ accepts $w$;
  - for every $w \notin L$, $M$ rejects $w$.
- A language $L$ is recognized by an NTM $M$, if
  - for every $w \in L$, $M$ accepts $w$;
  - for every $w \notin L$, $M$ does not accepts $w$. 

3/9
Recall that the standard TM is always deterministic. To avoid potential confusion, we will use the abbreviation DTM to mean deterministic Turing machine.

**Theorem 3.3** For every language \( L \), the following holds.

- If \( L \) is recognized by an NTM \( M \), then there is a DTM \( M' \) that recognizes \( L \).
- If \( L \) is decided by an NTM \( M \), then there is a DTM \( M' \) that decides \( L \).

The computation of an NTM \( M \) on input \( w \) can be pictured as a tree whose nodes are configurations of \( M \) defined as follows.

- The root node is the initial configuration \( q_0w \).
- The children of a node \( C \) are all possible \( C' \) where \( C \vdash C' \).

### 4 Some theorems on recognizable and decidable languages

**Theorem 3.4**

- If a language \( L \) is decidable, so is its complement \( \Sigma^* - L \).
- If both a language \( L \) and its complement \( \Sigma^* - L \) are recognizable, then \( L \) is decidable.

**Theorem 3.5**

- Recognizable languages are closed under union, intersection, concatenation and Kleene star.
- Decidable languages are closed under union, intersection, complement, concatenation and Kleene star.

### 5 Universal Turing machine and halting problem

**The string representation of a Turing machine.** Recall that a Turing machine is defined as a system \( M = (\Sigma, \Gamma, Q, q_0, q_{acc}, q_{rej}, \delta) \), where we can assume that \( \Sigma = \{0, 1\} \) and \( \Gamma = \{\langle, 0, 1, \sqcup\} \).

Without loss of generality, we can also assume that \( Q = \{0, 1, \ldots, n\} \) for some positive integer \( n \) with 0 being the initial state.

We note the following.

- Each state \( i \in Q \) is written as a string in its binary form.
- Each transition \((i, a) \rightarrow (j, b, \alpha) \in \delta \) can be written as string over the symbols 0, 1, (, ), , , \langle, \sqcup, L, R, S, \) where the symbol \( \sqcup \) represents \( \sqcup \), and L, R, S represent Left, Right, Stay, respectively.

So, the whole system \( M = (\Sigma, \Gamma, Q, 0, q_{acc}, q_{rej}, \delta) \) can be written as a string:

\[
[\Sigma] \# [\Gamma] \# [Q] \# [0] \# [q_{acc}] \# [q_{rej}] \# [\delta]
\]

where \([\cdot]\) denotes the string representing the component \( \cdot \) and \( \# \) the symbol separating two consecutive components.[⁴]

This shows that every Turing machine (whose tape alphabet is \( \Gamma = \{\langle, 0, 1, \sqcup\} \)) can be described as a string over a fixed set of the symbols, i.e., 0, 1, (, ), , , \langle, \sqcup, L, R, S, \# . All these symbols can be further encoded into strings over 0 and 1 to obtain a binary string, which we denote by \([M]\). That is, \([M]\) is the binary string representing the Turing machine \( M \). Sometimes, we will also say \([M]\) is the string description of \( M \), or the description of \( M \), for short.

*Obviously, since we consider only Turing machines with \( \Sigma = \{0, 1\} \) and \( \Gamma = \{\langle, 0, 1, \sqcup\} \), it is not necessary to include them in \([M]\). But for the sake of consistency in our notation, we simply include them.*
Universal Turing machine (UTM). A universal Turing machine (UTM) is a Turing machine \( U \) that gets as input a description of a Turing machine \([M]\) and a word \( w \). On such input, it simulates \( M \) on \( w \). (Some textbooks use the phrase “it runs \( M \) on \( w \)” for “it simulates \( M \) on \( w \).”)

Halting problem. We define the following languages:

\[
\text{HALT} := \{[M]\$w \mid M \text{ accepts } w \text{ where } w \in \{0,1\}^*\}. \\
\text{HALT}_0 := \{[M] \mid M \text{ accepts } [M]\}. \\
\text{HALT}_0' := \{[M] \mid M \text{ does not accept } [M]\}.
\]

**Theorem 3.6** \( \text{HALT}_0' \) is undecidable.

**Corollary 3.7** \( \text{HALT}_0 \) and \( \text{HALT} \) are undecidable.

**Proposition 3.8** The language \( \text{HALT}_0 \) and \( \text{HALT} \) are recognizable (recursively enumerable).

Recall that if both \( L \) and its complement \( \overline{L} = \Sigma^* - L \) are recognizable, then both are decidable. Then, the following corollary follows immediately from above.

**Corollary 3.9** The language \( \text{HALT} \) is not recognizable (recursively enumerable).

### 6 Reducibility

Consider a function \( F : \Sigma^* \to \Sigma^* \). A TM \( M \) that computes \( F \) is a TM that accepts every word \( w \in \Sigma^* \) and when it halts, the content of its tape is \( F(w) \). That is, on every word \( w \), \( M \) accepts \( w \) with the accepting run:

\[
q_0 \quad w \quad \vdash \cdots \vdash \quad q_{\text{acc}} \quad F(w)
\]

A function is **computable**, if there is a TM that computes it.

**Definition 3.10** A language \( L_1 \) is mapping reducible to another language \( L_2 \), denoted by \( L_1 \leq_m L_2 \), if there is a computable function \( F \) such that for every \( w \in \Sigma^* \):

\[
w \in L_1 \text{ if and only if } F(w) \in L_2
\]

The function \( f \) is called **mapping reduction**.

Sometimes we omit the word “mapping” and call it simply “reducible” or “reduction,” instead of “mapping reducible” or “mapping reduction.” Intuitively \( L_1 \leq_m L_2 \) means that \( L_2 \) is “computationally more general,” or “more general” than \( L_1 \) and that a TM for deciding \( L_2 \) can be used to decide \( L_1 \).

**Definition 3.11** A language \( L_1 \) is Turing reducible to another language \( L_2 \), denoted by \( L_1 \leq_T L_2 \), if by assuming that \( L_2 \) is decidable by a TM \( M_2 \), there is a TM \( M_1 \) that decides \( L_1 \) using \( M_2 \) as a “subroutine.”

Moreover, we also assume that \( M_2 \) decides \( L_2 \) in one step. We call \( M_1 \) a TM with oracle access to \( L_2 \).

Obviously, if \( L_1 \leq_m L_2 \), then \( L_1 \leq_T L_2 \). Also, if \( L_1 \leq_T L_2 \) and \( L_1 \) is undecidable, so is \( L_2 \).
6.1 Some variants of Halting problem

The following languages are all undecidable.

- \( L_0 := \{ [M] \mid L(M) = \emptyset \} \).
  That is, \([M] \in L_0\) if and only if \(M\) does not accept any word.

- \( L_1 := \{ [M] \mid L(M) = \{0, 1\}^* \} \).
  That is, \([M] \in L_1\) if and only if \(M\) accepts every word.

- \( L_2 := \{ [M] \mid M \text{ accepts the empty word } \epsilon \} \)
  That is, \([M] \in L_2\) if and only if \(M\) accepts the empty word \(\epsilon\).

- \( L_3 := \{ [M] \mid M \text{ accepts the word } 1101 \} \).

- \( L_4 := \{ [M] \mid L(M) = \{a^n b^n \mid n \geq 0\} \} \).

- \( L_5 := \{ [M] \mid L(M) \text{ is a regular language} \} \).

6.2 Some undecidable problems concerning CFL

The non-emptiness problem for the intersection of two CGL’s. We define the CFL-Intersection problem is defined as follows.

<table>
<thead>
<tr>
<th>CFL-Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Two CFG’s ( G_1 = (\Sigma, V_1, R, S) ) and ( G_2 = (\Sigma, V_2, R_2, S_2) ).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if ( L(G_1) \cap L(G_2) \neq \emptyset ). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 3.12** The problem CFL-Intersection is undecidable.

The CFL universality problem. The problem CFL-Universality is defined as follows.

<table>
<thead>
<tr>
<th>CFL-Universality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A CFG ( G = (\Sigma, V, R, S) ).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if ( L(G) = \Sigma^* ). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 3.13** The problem CFL-Universality is undecidable.

The CFL subset problem. This problem, denoted by CFL-Subset, is defined as follows.

<table>
<thead>
<tr>
<th>CFL-Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Two CFG’s ( G_1 ) and ( G_2 ).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if ( L(G_1) \subseteq L(G_2) ). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 3.14** The problem CFL-Subset is undecidable.
7 Post correspondence problem (PCP)

Post Correspondence Problem, denoted by PCP, is defined as follows.

<table>
<thead>
<tr>
<th>PCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td><strong>Task:</strong></td>
</tr>
</tbody>
</table>

Theorem 3.15 PCP is undecidable.

Appendix

A Turing machines with Stay option

In some textbooks, Turing machines are defined such that the head can stay put, instead of moving Left or Right. Formally, a transition can be of the form:

\[(q, a) \rightarrow (p, b, \alpha)\]

where \(\alpha \in \{\text{Left}, \text{Right}, \text{Stay}\}\).

If \(\alpha = \text{Stay}\), then the head stays where it is. Such Stay option is obviously equivalent to making two moves: Right, and followed by Left, thus, does not add any power of computation.

B Putting a marker on the leftmost cell of the tape

To prevent the head falls off the tape, we reserve a special symbol \(<\) that can be used to mark the leftmost cell of the tape of Turing machines. We describe a TM, denoted by \(M_{sr}\), that on input \(w \in \Sigma^*\), it will always halt in the accepting configuration \(<q_{acc}\#w>\).

- \(\Sigma = \{0, 1\}\).
- \(\Gamma = \{\leftarrow, 0, 1, \sqcup\}\).
- \(Q = \{q_0, q, s, q_{acc}, q_{rej}\}\).
- \(\delta\) consists of the following:

  - \((q_0, 1) \rightarrow (p, <, \text{Right})\)
  - \((q_0, 0) \rightarrow (r, <, \text{Right})\)
  - \((q_0, \sqcup) \rightarrow (q_{acc}, <, \text{Right})\)
  - \((q_0, <) \rightarrow (q_{rej}, <, \text{Right})\)
  - \((p, 1) \rightarrow (p, 1, \text{Right})\)
  - \((p, 0) \rightarrow (r, 0, \text{Right})\)
  - \((p, \sqcup) \rightarrow (s, \sqcup, \text{Left})\)
  - \((p, <) \rightarrow (q_{rej}, <, \text{Right})\)
  - \((r, 1) \rightarrow (p, 1, \text{Right})\)
  - \((r, 0) \rightarrow (r, 0, \text{Right})\)
  - \((r, \sqcup) \rightarrow (s, \sqcup, \text{Left})\)
  - \((r, <) \rightarrow (q_{rej}, <, \text{Right})\)
  - \((s, 0) \rightarrow (s, 0, \text{Left})\)
  - \((s, 1) \rightarrow (s, 1, \text{Left})\)
  - \((s, <) \rightarrow (q_{acc}, <, \text{Right})\)
  - \((s, \sqcup) \rightarrow (q_{rej}, <, \text{Right})\)

The construction above can be easily generalized for arbitrary \(\Sigma\).

This \(M_{sr}\) can now be run as a precursor of an arbitrary Turing machine whose head never moves left whenever it reads the marker \(<\). Thus, we can always assume that the head never falls off the tape.
C Encoding an arbitrary alphabet into the binary alphabet \{0, 1\}

Turing machines are usually defined with arbitrary input and tape alphabets. It is not difficult to show that any alphabet can be “encoded” with binary alphabet.

Suppose \(\Gamma = \{a_1, \ldots, a_n, \bot\}\). Each symbol \(a_i\) can then be encoded with a 0-1 string of length \([\log_2 n]\). For example, if \(\Gamma = \{a_1, \ldots, a_5, \bot\}\), we can encode \(a_1\) with 000, \(a_2\) with 001, \(a_3\) with 010, \(a_4\) with 011, and \(a_5\) with 100. We denote by \(\langle a_i \rangle\) the encoding of the symbol \(a_i\). For a word \(w \in \Gamma^*\), \(\langle w \rangle\) denotes the encoding of \(w\) by replacing each symbol \(a_i\) in \(w\) with \(\langle a_i \rangle\). For example, if \(w = a_1 a_5 a_2 a_1\), \(\langle w \rangle = \langle a_1 \rangle \langle a_3 \rangle \langle a_2 \rangle \langle a_1 \rangle = 000 100 001 000\).

We have the following proposition that shows that we can always assume that the Turing machines under consideration always work on tape alphabet \(\Gamma = \{\bot, 0, 1, \bot\}\), where \(\bot\) is the marker that marks the leftmost cell of the tape.

**Proposition 3.16** Let \(M = \langle \Sigma, \Gamma, Q, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta \rangle\) be a TM, where \(\Gamma = \{a_1, \ldots, a_n, \bot\}\). Let \(K = [\log_2 n]\). Let \(\langle a_i \rangle\) be an encoding of symbol \(a_i\) with 0-1 string of length \(K\). There is a TM \(M' = \langle \{0, 1\}, \{\bot, 0, 1, \bot\}, Q', q_0', q_{\text{acc}}, q_{\text{rej}}, \delta' \rangle\) such that for every word \(w \in \Sigma^*\), the following holds.

\[M\ \text{accepts} \ w \ \text{if and only if} \ M'\ \text{accepts} \ \langle w \rangle\]

Intuitively, \(M'\) simulates \(M\) by reading the tapes by blocks of \([\log_2 n]\) cells. It then remembers the block that it reads in its states, and “simulates” the transitions of \(M\) accordingly.

Formally, \(M = \langle \{0, 1\}, \{0, 1, \bot\}, Q, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta \rangle\) is defined as follows. Let \(\{0, 1\}^{\leq K}\), i.e., the set of all 0-1 strings of length less than or equal to \(K = [\log_2 n]\).

\[Q' = (Q \times \{0, 1\}^{\leq K}) \cup (Q \times \{L_1, \ldots, L_K, R_1, \ldots, R_K\})\]

\[\cup (Q \times \{L, R\} \times W \times \{0, 1\}^{\leq K}).\]

- \(q'_0 = (q_0, \epsilon)\).

- \(\delta'\) is defined as follows.

  - For every \(u \in \{0, 1\}^{\leq K-1}\), for every \(p \in Q - \{q_{\text{acc}}, q_{\text{rej}}\}\), \(\delta'\) consists of the following transitions.

    \[((p, u), 0) \rightarrow ((p, u0), 0, \text{Right})\]

    \[((p, u), 1) \rightarrow ((p, u1), 1, \text{Right})\]

  - For every \((q, a) \rightarrow (p, b, \text{Left}) \in \delta\), for every \(d \in \{0, 1, \bot\}\), \(\delta'\) consists of the following transitions.

    \[((q, \langle a \rangle), d) \rightarrow ((p, \underline{L}, W, \langle b \rangle), d, \text{Left})\]

  - For every \((q, a) \rightarrow (p, b, \text{Right}) \in \delta\), for every \(d \in \{0, 1, \bot\}\), \(\delta'\) consists of the following transitions.

    \[((q, \langle a \rangle), d) \rightarrow ((p, \underline{R}, W, \langle b \rangle), d, \text{Left})\]

  - For every \(p \in Q\), for every \(c \in \{0, 1\}\), for every \(v \in \{0, 1\}^{\leq K-1}\) and \(v \neq \epsilon\), for every \(d \in \{0, 1, \bot\}\), for every \(\beta \in \{L, R\}\), \(\delta'\) consists of the following transitions.

    \[((p, \beta, W, vc), d) \rightarrow ((p, \beta, W, v), c, \text{Left})\]
For every $p \in Q$, for every $d \in \{\prec, 0, 1, \sqcup\}$, $\delta'$ consists of the following transitions.

\[
(p, L, W, \epsilon, d) \rightarrow (p, L_k, d, \text{Right})
\]

For every $p \in Q$, for every $d \in \{\prec, 0, 1, \sqcup\}$, $\delta'$ consists of the following transitions.

\[
(p, R, W, \epsilon, d) \rightarrow (p, R_k, d, \text{Right})
\]

For every $p \in Q$, for every $d \in \{0, 1, \sqcup\}$, $\delta'$ consists of the following transitions.

\[
(p, L, W, \epsilon, d) \rightarrow (p, L_k, d, \text{Right})
\]

For every $p \in Q$, for every $d \in \{0, 1, \sqcup\}$, $\delta'$ consists of the following transitions.

\[
(p, R_1, d) \rightarrow (p, \epsilon, d, \text{Right})
\]

\[
(p, L_1, d) \rightarrow (p, \epsilon, d, \text{Left})
\]

All the other transitions not specified above are assumed to enter $q_{rej}$. 
