The Complete Classes
Log-Space Reducibility

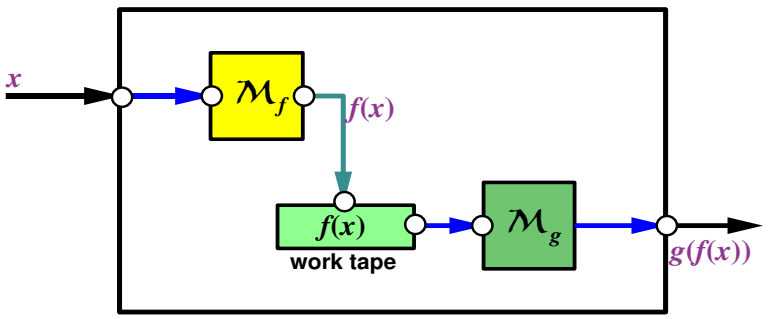
- **Given two languages L_1 and L_2, L_1 is log-space reducible to L_2, denoted as $L_1 \leq_{\text{log-space}} L_2$, if there is a function $f : L_1 \mapsto L_2$ that is computable by a deterministic Turing machine in space $O(\log_2 |x|)$ for all input string x such that $x \in L_1 \Leftrightarrow f(x) \in L_2$.**
- **Let $\mathcal{C}$ be a complexity class.**
  - **A language L is <u>hard</u> for $\mathcal{C}$ under log-space reduction if for all languages $Q \in \mathcal{C}$ we have $Q \leq_{\text{log-space}} L$.**
  - **A language L is <u>complete</u> for $\mathcal{C}$ under log-space reduction if L is hard for $\mathcal{C}$ under log-space reduction and $L \in \mathcal{C}$.**
- Log-space is stronger than polynomial-time because it uses less space and has a polynomial-time bound.

- **Let $f : L_1 \mapsto L_2$ be a log-space reduction function computed by Turing machine $\mathcal{M}$. $\mathcal{M}$ halts after a polynomial number of steps for all input x.**
  - Recall that the number of configurations in the configuration graph that $\mathcal{M}$ uses to accept input x is bounded by

    $$Q \times \left[c^{(s+2)p(n)}\right] \times \left[np(n)^{s+1}\right] \leq k^{p(n)+\log_2 n}$$

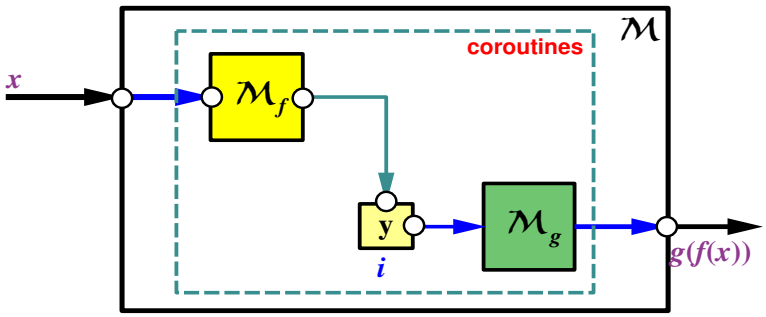
    where Q is the number of states, c is the size of the alphabets, s is the number of work tapes, $n = |x|$ is the length of the input string, and $p(n)$ is the space bound.
  - Because $\mathcal{M}$ has a space bound of $\log_2 |x|$, $p(x) = \log_2 |x|$ and the above bound becomes $O(k^{\log_2 |x|}) = O(|x|^{\log_2 k})$, which is a polynomial in $|x|$.
  - Since this is a deterministic Turing machine, no configuration can be repeated in the computation (because such repetition means the machine does nor halt), the path in the configuration graph from the initial configuration to a final configuration is bounded by $O(|x|^c)$ for some constant c.

- **If $f : L_1 \mapsto L_2$ and $g : L_2 \mapsto L_3$ be log-space reduction functions from L_1 to L_2 and L_2 to L_3, respectively, then the composition $g(f)$ is also a log-space reduction function from L_1 to L_3.**
  - It is trivial that $g(f)$ is a reduction function from L_1 to L_3. What is non-trivial is that it can be computed by a Turing machine with a log-space bound.
  - Let $\mathcal{M}_f$ and $\mathcal{M}_g$ be the Turing machines that compute f and g. We cannot let $\mathcal{M}_f$ store its output $f(x)$ on a work tape as show below, because the space consumption of $f(x)$ may be larger than $O(\log_2 |x|)$. To achieve a log-space bound, we cannot save $f(x)$!



  - What do we do? We combine $\mathcal{M}_f$ and $\mathcal{M}_g$ together in a "tightly-coupled" way.
  - The trick is that when $\mathcal{M}_g$ needs a symbol under its tape head, it asks $\mathcal{M}_f$ to deliver one.

– We need two variables: i for the tape head position of $\mathcal{M}_g$, $i = 1$ initially, and y the output symbol from $\mathcal{M}_f$.



– We then run $\mathcal{M}_f$ until it generates an output in y. The composition machine repeats the following:

1. Run $\mathcal{M}_g$ for one instruction. If it reaches a final state, stops.
2. If tape head does not move, do nothing.
3. If tape head moves to the right, increase i and run $\mathcal{M}_f$ until it generates an output to y for $\mathcal{M}_g$'s next instruction.
4. If tape head moves to the left, decrease i and restart $\mathcal{M}_f$ from the **very beginning** until it generates the i-th symbol into y.
5. Thus, $\mathcal{M}_f$ and $\mathcal{M}_g$ act like a pair of coroutines.

– Since $\mathcal{M}_f$ and $\mathcal{M}_g$ have space bound $O(\log_2 |x|)$ and since this program uses a fixed number of variables, its space bound still is $O(\log_2 |x|)$.

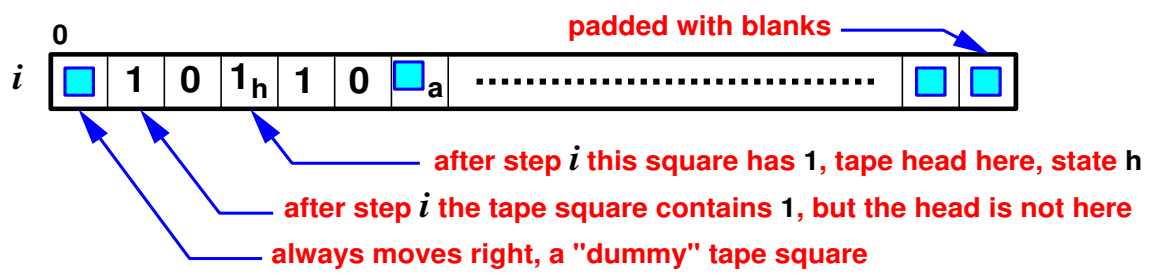
A Turing Machine for log–space Composition

```
i := 1;
```
Run Turing machine $\mathcal{M}_f$ until an output y is generated;
```
WHILE (1) DO
   BEGIN
```
Execute one instruction of $\mathcal{M}_g$.
```
      IF (it is a final state of M_g) THEN
```
Report "accept" or "reject" and exit.
```
      IF (M_g does not move its tape head) THEN
```
Do nothing
```
      ELSE IF (M_g moves its tape head to the right) THEN
         BEGIN
            i := i + 1;
```
Run $\mathcal{M}_f$ until an output y is generated
```
         END
      ELSE /* M_g moves its tape head to the right */
         BEGIN
            i := i - 1;
```
Run $\mathcal{M}_f$ from the *very beginning* until it generates the *i*–th symbol to y
```
         END
   END
```

Turing Machines: Revisited
Computation Table

- In addition to configuration graphs, the activities of a Turing machine can also be put into a table, its **computation table**.
- Consider a deterministic Turing machine with a polynomial-time bound $O(|x|^k)$. If k is sufficiently large, the polynomial-time bound is simply $|x|^k$.
- Since a multiple-tape Turing machine can always be simulated with a single-tape Turing machine, we use one tape only.
- We shall also make the following modifications:
  1. When the Turing machine halts, it writes its answer (*i.e.*, "yes" or "no") in the first tape square. We can add a "subroutine" to the given Turing machine to rewind the tape and write the answer.
  2. The tape is modified by adding a "dummy" square at the very beginning (*i.e.*, position 0). It always contains a blank and the tape should not be moved there.

- Now we shall construct a computation table:
  1. Since the machine takes $|x|^k$ steps to accept/reject its input x, we build a $|x|^k \times (|x|^k + 2)$ table T.
  2. Row i records the tape content, state, and tape head position *after* the i-th step completes.
  3. The 0-th element of row i, $T_{i,0}$, is the "dummy" tape square which is always a blank.
  4. Because the number of steps the Turing machine can take is bounded by $|x|^k$, it can scan no more than $|x|^k$ squares. Hence, all unscanned tape squares are padded with blanks.
  5. Entry $T_{i,j}$ has the following information:
     - $T_{i,j}$ contains the symbol of tape square j *after* step i completes.
     - If the head is on square j, $T_{i,j}$ also contains the state of the machine.
     - $T_{0,j}$ contains the j-th input symbol and $T_{0,1}$ also includes the initial state (and tape head).

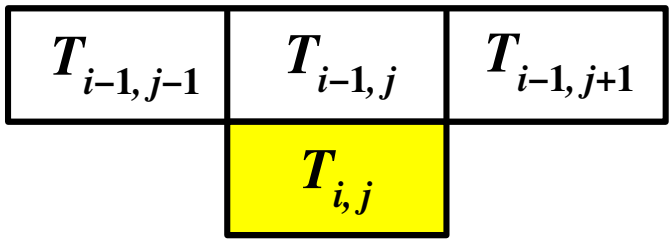


6. The Turing machine may stop before step $|x|^k$. In this case, all rows up to row $|x|^k$ are all the same. They have a "yes" or a "no" in square 1 and the the tape head is also on square one.
7. The following is an example showing the activities of a Turing machine accepting a string 0000:

Step	1	2	3	4	5	6	*Instruction*
0	0_1	0	0	0	$\square$	$\square$	$\uparrow 0, \downarrow \square, \rightarrow$
1	$\square$	0_2	0	0	$\square$	$\square$	$\uparrow 0, \downarrow x, \rightarrow$
2	$\square$	x	0_3	0	$\square$	$\square$	$\uparrow 0, \rightarrow$
3	$\square$	x	0	0_4	$\square$	$\square$	$\uparrow 0, \downarrow x, \rightarrow$
4	$\square$	x	0	x	$\square_3$	$\square$	$\uparrow \square, \leftarrow$
5	$\square$	x	0	x_5	$\square$	$\square$	$\uparrow x, \leftarrow$
6	$\square$	x	0_5	x	$\square$	$\square$	$\uparrow 0, \leftarrow$
$\vdots$							$\vdots$
t	x	x	x	$\square$	$\square_f$	$\square$	accept
$\vdots$							$\leftarrow' s$
z	yes						done!

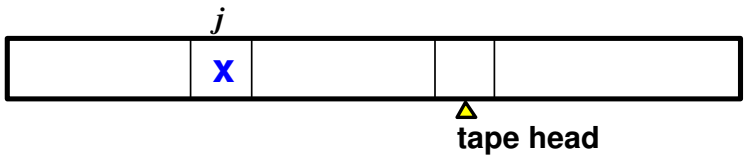
   In the above, $\uparrow$ and $\downarrow$ are read and write, and $\rightarrow$ and $\leftarrow$ are moving right and moving left.
8. Thus, each row of the computation table provides a snapshot of the execution of a Turing machine

- **How do we "compute" $T_{i,j}$?**
  To determine $T_{i,j}$, we only need to look at $T_{i-1,j-1}$, $T_{i-1,j}$ and $T_{i-1,j+1}$ as shown below. (**Why?**)

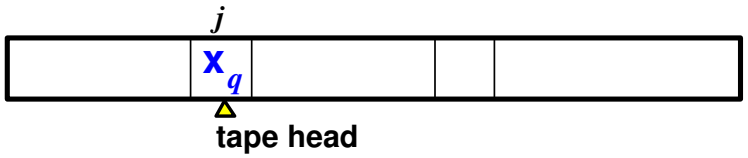


  Note that $T_{i,j}$ may contain a symbol in square j (tape content), and a state if the head is on square j.

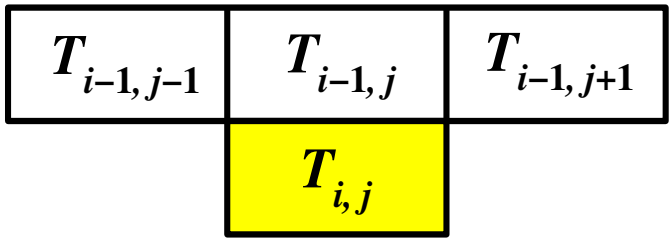
  1. **Tape Content:**
     - If $T_{i-1,j}$ indicates the tape head is elsewhere, the tape content of $T_{i,j}$ is equal to that of $T_{i-1,j}$.



     - If $T_{i-1,j}$ indicates the tape head is on square j, the content of $T_{i,j}$ is computed from the state and content in $T_{i-1,j}$. That is, what would be written into square j when the machine is in state q and the input is x?



2. **Machine State and Tape Head:**



(a) If $T_{i-1,j}$ has the tape head and the transition does not move the head, then $T_{i,j}$ receives the next state (and head)

(b) If $T_{i-1,j-1}$ has the tape head, and the transition moves the head to the right, $T_{i,j}$ receives the next state (and head).

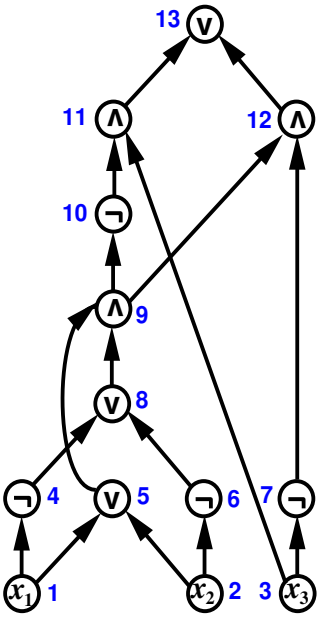
(c) If $T_{i-1,j+1}$ has the tape head, and the transition moves the head to the left, $T_{i,j}$ receives the next state (and head).

(d) Otherwise, $T_{i,j}$ has no state and head information.

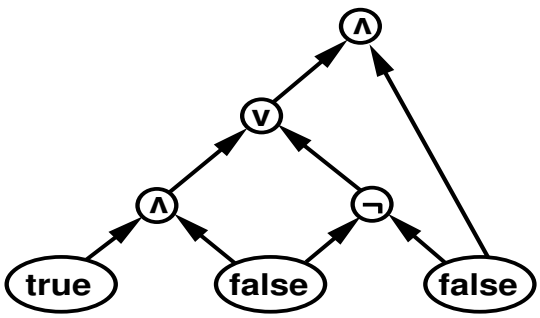
The Boolean Circuit Model

- A *Boolean Circuit* is a directed acyclic graph $C = (V, E)$, where
  - $V = \{1, 2, \ldots, n\}$ are called the **gates** of C;
  - A topological sorting can renumber the vertices such that $i < j$ holds if (i, j) is an edge.
  - All vertices have *indegree* (*i.e.*, no. of incoming edges) 0, 1, or 2;
  - Each vertex is labeled as **true**, **false**, $\vee$, $\wedge$, $\neg$, or x_i $(i = 1, 2, \ldots)$.
    1. If the label of a vertex is **true**, **false**, or x_i $(i = 1, 2, \ldots)$, its indegree must be **0**. They are the **input** gates.
    2. If the label is $\vee$ or $\wedge$, its indegree must be two.
    3. If the label is $\neg$, its indegree must be 1.
    4. Vertices have no outgoing edge are **output** gates.
- The **size** of a circuit is the number of gates it has.
- The **depth** of a circuit is the length (*i.e.*, number of edges) of the longest path from an input gate to an output gate.

- The following is a Boolean circuit with **input** gates x_1, x_2 and x_3, and **output** gate at the top (*i.e.*, the $\vee$ gate numbered 13). Note the numbering scheme. Also note that it has no **true** nor **false** gates. This circuit has size 13 and depth 6.



- The following is a Boolean circuit without variables. Its input are all truth values.

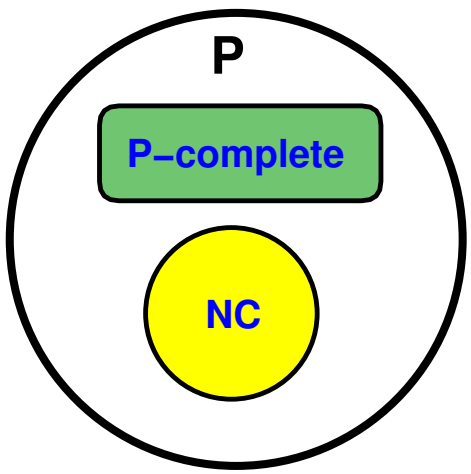


The Circuit-Value Problem

- **Given a Boolean circuit with one output gate and truth values as input gates, is the output true?**
- Note that if the input are variables, it is not difficult to see the Circuit-Value Problem is equivalent to SAT.
- However, since the input to the **Circuit-Value Problem** consist of truth values, it is in **P**. Do you know how to prove this?

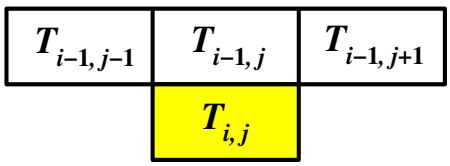
P-Completeness

- **A language L is P-hard, if $Q \leq_{\text{log-space}} L$ for every language $Q \in \mathbf{P}$.**
- **A language L is P-complete if L is P-hard and $L \in \mathbf{P}$.**
- The set of all **P**-complete languages is denoted as **P-complete**.
- Thus, **P-complete** contains the most difficult problems in **P** with respect to log-space reduction.
- Not only this, **P-complete** contains problems that are unlikely to have fast solutions on parallel computers. As a result, **P**-complete problems are often called *inherently sequential problems*.



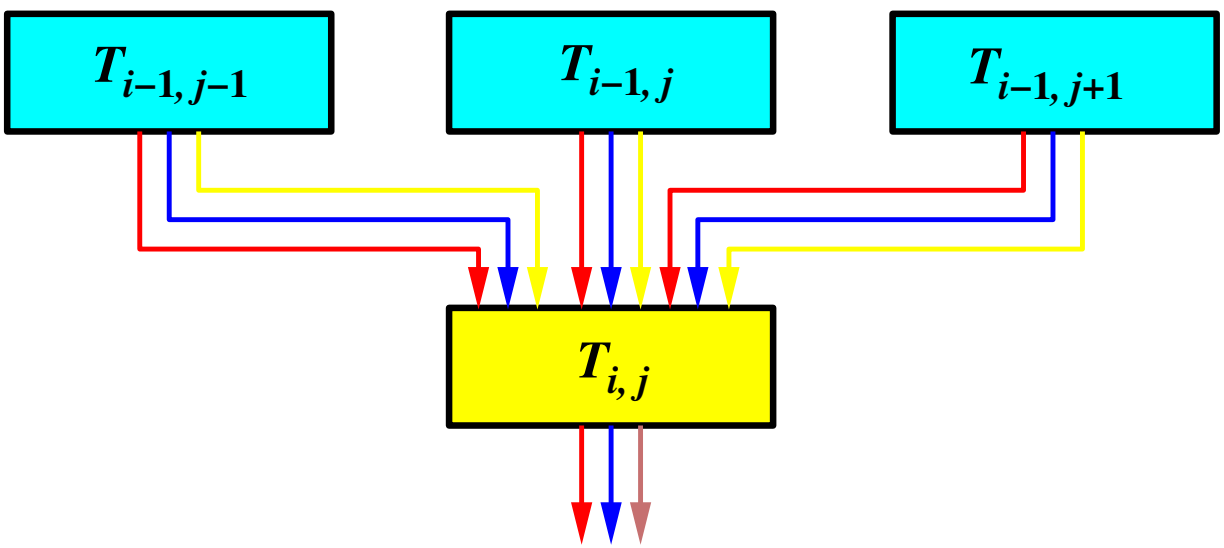
Tape Squares as Boolean Circuits

- **A computation table contains $|x|^k \times (|x|^k + 2)$ tape squares, each of which can be represented by a simple Boolean circuit.**
- The content of $T_{i,j}$ only depends on the contents of $T_{i-1,j-1}$, $T_{i-1,j}$ and $T_{i-1,j+1}$. Note that $T_{i,j}$ may contain a symbol, a state if the head is on square j, and an indicator showing if the head is on square j.



- The "output" (*i.e.*, content) of $T_{i,j}$ must be encoded. Suppose the given Turing machine has alphabet Σ and states Q, and $\Sigma \cap Q = \emptyset$.
  1. The first part consists of $|\Sigma|$ bits $\sigma_1\sigma_2 \ldots \sigma_{|\Sigma|}$, one for each tape symbol. If σ_i is 1, $T_{i,j}$ contains the i-th symbol of Σ.
  2. The second part consists of $|Q|$ bits $q_1 q_2 \ldots q_{|Q|}$, one for each state. If q_i is 1, $T_{i,j}$ contains state i.
  3. The third part has only one bit t. If the head is on square j, t is 1; otherwise, t is 0.
  4. There are $|\Sigma| + |Q| + 1$ output bits!

- Because $T_{i,j}$ depends on $T_{i-1,j-1}$, $T_{i-1,j}$ and $T_{i-1,j+1}$ and because it has $|\Sigma| + |Q| + 1$ output bits, $T_{i,j}$ has $3(|\Sigma| + |Q| + 1)$ input bits and $|\Sigma| + |Q| + 1$ output bits.
- Putting $T_{i-1,j-1}$, $T_{i-1,j}$, $T_{i-1,j+1}$ and $T_{i,j}$ together, we have the following diagram, in which the red arrows indicate the $|\Sigma|$ signals, blue arrows indicate the $|Q|$ state signals, and the yellow arrow gives the tape head signal.



- Our next step is to take the $3(|\Sigma| + |Q| + 1)$ input signals from $T_{i-1,j-1}$, $T_{i-1,j}$ and $T_{i-1,j+1}$ and produce $T_{i,j}$'s $|\Sigma| + |Q| + 1$ output signals.
- We shall use $T^{\tau}_{i,j}$, $T^{q}_{i,j}$ and $T^{\triangle}_{i,j}$ to indicate a symbol bit, a state bit and the tape head bit.

- **Computing the output symbol:**
  The content of $T_{i,j}$ depends on if the tape head is on square j at the beginning of step i.

  1. **Head is not on square j:** The content of $T_{i,j}$ is the same as that of $T_{i-1,j}$ for all $\sigma \in \Sigma$.

     $$\left(\neg T^{\triangle}_{i-1,j} \wedge T^{\sigma}_{i-1,j}\right)$$

     The above is true if in step $i-1$, the head is not on square j and square j contains σ.
  2. **Head is on square j:** The machine may write σ into square j. Define A_σ as follows:

     $$A_\sigma = \{(q,\tau) \mid \text{state } q \text{ reads } \tau \text{ and writes } \sigma\ \}$$

     The following is true if, in step $i-1$, $T_{i-1,j}$ contains state q and input τ, and writes σ:

     $$\bigvee_{(q,\tau)\in A_\sigma} \left(T^{q}_{i-1,j} \wedge T^{\tau}_{i-1,j}\right)$$

     Note that if $T_{i-1,j}$ contains no state (and hence no head info), the above is false.
  3. Hence, at the end of step i, for every $\sigma \in \Sigma$, $T^{\sigma}_{i,j}$ is

     $$T^{\sigma}_{i,j} = \left[\bigvee_{(q,\tau)\in A_\sigma} \left(T^{q}_{i-1,j} \wedge T^{\tau}_{i-1,j}\right)\right] \vee \left(\neg T^{\triangle}_{i-1,j} \wedge T^{\sigma}_{i-1,j}\right)$$

- **Determining the State:**
  The tape head can only move left, move right and stay put. For every state q, define three sets as follows:

$$\begin{aligned} L_q &= \{(p,\tau) \mid p \xrightarrow{\tau} q \text{ and head moves left } \} \\ R_q &= \{(p,\tau) \mid p \xrightarrow{\tau} q \text{ and head moves right } \} \\ S_q &= \{(p,\tau) \mid p \xrightarrow{\tau} q \text{ and head stays } \} \end{aligned}$$

  1. **Head moves left:** Only $T_{i-1,j+1}$ affects $T_{i,j}$. Thus, $T_{i,j}$ is in state q if the following holds:

$$\textbf{Left} = \bigvee_{(p,\tau)\in L_q} \left(T^p_{i-1,j+1} \wedge T^\tau_{i-1,j+1}\right)$$

  2. **Head moves right:** Only $T_{i-1,j-1}$ affects $T_{i,j}$. Thus, $T_{i,j}$ is in state q if the following holds:

$$\textbf{Right} = \bigvee_{(p,\tau)\in R_q} \left(T^p_{i-1,j-1} \wedge T^\tau_{i-1,j-1}\right)$$

  3. **Head does not move:** Only $T_{i-1,j}$ affects $T_{i,j}$. Thus, $T_{i,j}$ is in state q if the following holds:

$$\textbf{Stay} = \bigvee_{(p,\tau)\in S_q} \left(T^p_{i-1,j} \wedge T^\tau_{i-1,j}\right)$$

  4. Since the state in $T_{i,j}$ is determined by one of the three moves above, the state in $T_{i,j}$ is

$$T^q_{i,j} = \textbf{Left} \vee \textbf{Right} \vee \textbf{Stay}$$

- **Determining the Tape Head Position:**
  The head is on j if **(1)** was on $j-1$ and moved right, **(2)** was on $j+1$ and moved left, or **(3)** was on j and did not move. Define three sets as follows:

$$\begin{aligned} M_{\leftarrow} &= \{(p,\tau) \mid p \xrightarrow{\tau} \text{ and moves left } \} \\ M_{\rightarrow} &= \{(p,\tau) \mid p \xrightarrow{\tau} \text{ and moves right } \} \\ M_{\circ} &= \{(p,\tau) \mid p \xrightarrow{\tau} \text{ and does not move } \} \end{aligned}$$

  1. **Head on $j-1$ and moves right:** Since "head on $j-1$ " is $T^{\triangle}_{i-1,j-1}$ and "moves right" is

$$\bigvee_{(p,\tau)\in M_{\rightarrow}} \left(T^{p}_{i-1,j-1} \wedge T^{\tau}_{i-1,j-1}\right)$$

     "head on $j-1$ and moves right" is

$$\triangle_{j-1,\rightarrow} = T^{\triangle}_{i-1,j-1} \wedge \left[\bigvee_{(p,\tau)\in M_{\rightarrow}} \left(T^{p}_{i-1,j-1} \wedge T^{\tau}_{i-1,j-1}\right)\right]$$

  2. **Head on $j+1$ and moves left:**

$$\triangle_{j+1,\leftarrow} = T^{\triangle}_{i-1,j+1} \wedge \left[\bigvee_{(p,\tau)\in M_{\leftarrow}} \left(T^{p}_{i-1,j+1} \wedge T^{\tau}_{i-1,j+1}\right)\right]$$

  3. **Head on j and no move:**

$$\triangle_{j,\circ} = T^{\triangle}_{i-1,j} \wedge \left[\bigvee_{(p,\tau)\in M_{\circ}} \left(T^{p}_{i-1,j} \wedge T^{\tau}_{i-1,j}\right)\right]$$

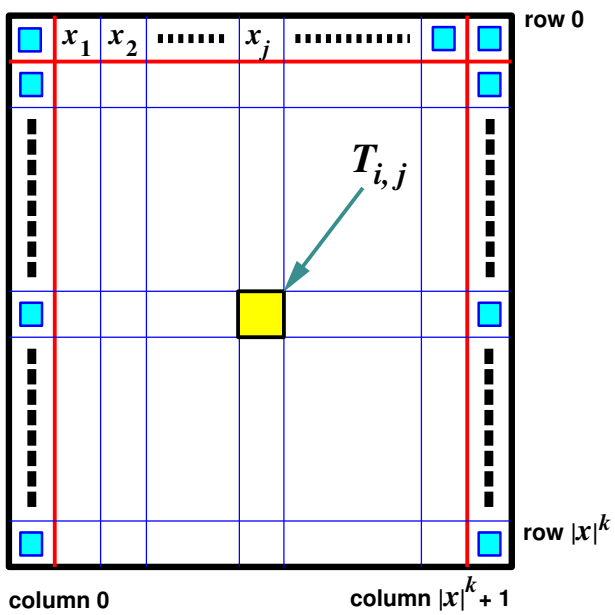
  4. Therefore, the head position for $T_{i,j}$ is

$$T^{\triangle}_{i,j} = \triangle_{j-1,\rightarrow} \vee \triangle_{j,\circ} \vee \triangle_{j+1,\leftarrow}$$

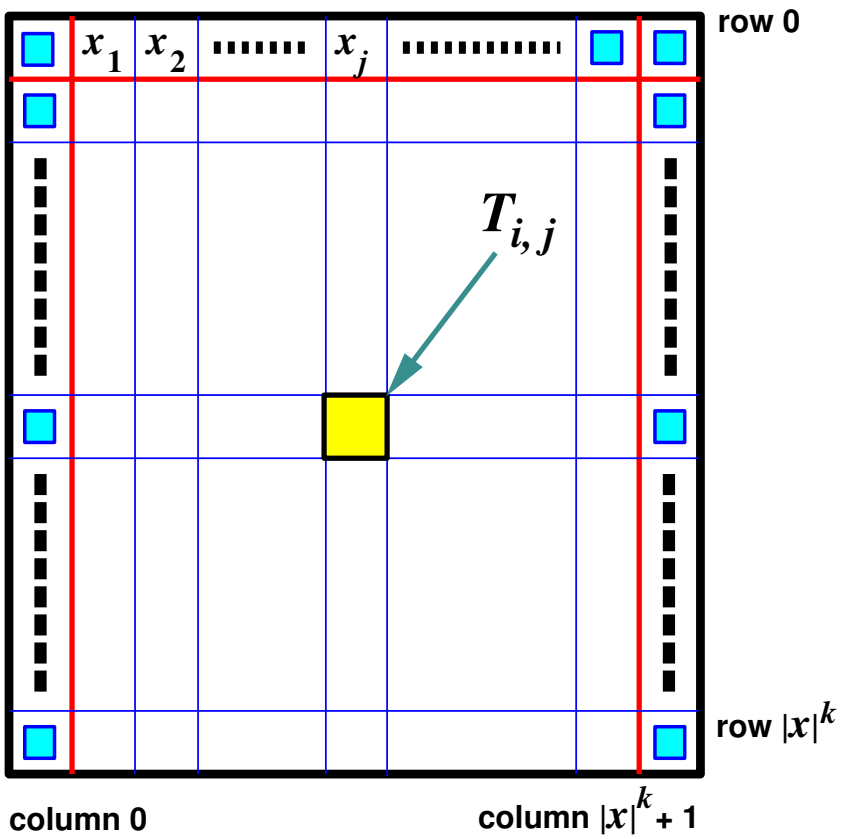
- **What is the size of $T_{i,j}$?**
  1. The size is bounded by $O(|Q| \times |\Sigma|)$ since we need to scan many pairs (q, τ), where $q \in Q$ and $\tau \in \Sigma$.
  2. It is not difficult to establish a very precise bound. Please do it yourself.
  3. Since this value is independent of the length of an input string, the size is actually $O(1)$.
- **What is the space bound for constructing $T_{i,j}$?**
  1. Only a number of variables are used for loop indices and temporary storage. Space consumption is $O(1)$
  2. Note also that the space bound is independent of the length of an input string.
- **What is the time bound for constructing $T_{i,j}$?**
  1. Because all three steps involve scanning pairs like (q, τ), the time bound is at most $O(|Q| \times |\Sigma|)$.
  2. Since it is independent of the length of an input string, the time bound is $O(1)$.

Simulating a Turing Machine with a Boolean Circuit

- **Given a Turing machine $\mathcal{M}$ and an input string x, simulate $\mathcal{M}$ with a Boolean circuit.**
- As discussed in the discussion of Computation Table, we add a dummy column 0 and a dummy column $|x|^k + 1$. Thus, we have $|x|^k + 2$ columns.
- The input string is stored in $T_{0,1}$, $T_{0,2}$ and $T_{0,|x|}$. The initial state and tape head are in $T_{0,1}$.
- For $T_{i,0}$ and $T_{i,|x|^k+1}$, **(1)** all symbol bits are 0's except for the blank's which is 1, **(2)** all state bits are 0's, and **(3)** all tape head bits are 0's.



- For all $T_{0,j}$, they only have output bits:
  1. The x_j bit is 1, all other symbol bits are 0's,
  2. All state bits are 0's except for $T_{0,1}$ which has the initial state bit 1, and
  3. The tape head bit is 0 except for $T_{0,1}$ whose tape head bit is 1.
- For any $T_{i,j}$, where $1 \leq i \leq |x|^k$ and $1 \leq j \leq |x|^k$, it is constructed from $T_{i-1,j-1}$, $T_{i-1,j}$ and $T_{i-1,j+1}$.



- It is not difficult to prove (by induction on steps) that $\mathcal{M}$ accepts x if and only if $T_{|x|^k,1}$ contains 1.

- **Space Complexity:**

  1. Given $\mathcal{M}$ and x, we need to compute $|x|^k$. We can use a simple loop to do that. Since the loop counter and $|x|$ requires $O(\log_2 |x|)$ space, this step requires $O(\log_2 |x|)$ space.
  2. The construction of each $T_{i,j}$ uses $O(1)$ space because it does not depend on x.
  3. We need a double loop to write the table to the output tape. Since the number of squares for storing $|x|^k + 1$ is $O(\log_2 |x|)$, the space consumption for this double loop is $O(\log_2 |x|)$.

```
FOR i := 0 TO |x|^k + 1 DO
   FOR j := 0 TO |x|^k  DO
      BEGIN
         Generate T_{i,j} from T_{i-1,j-1} , T_{i-1,j} and T_{i-1,j+1}
         Write T_{i,j}  to the output tape
      END
```

  4. **Therefore, it takes log-space to convert a Turing machine $\mathcal{M}$ with input x to an equivalent Boolean circuit $\mathcal{C}$ such that $\mathcal{M}$ accepts x if and only if $\mathcal{C}$ outputs 1.**

Circuit-Value Problem is P-complete

1. The Circuit-Value Problem (CVP) is in $\mathbf{P}$.
2. We shall prove $Q \leq_{\text{log-space}} \text{CVP}$ for every $Q \in \mathbf{P}$.
3. If $Q \in \mathbf{P}$, there is a Turing machine $\mathcal{M}$ to accept $x \in L$ in $O(|x|^k)$ steps.
4. From x and the Turing machine $\mathcal{M}$, we can construct a circuit $\mathcal{C}_x$ in log-space such that $\mathcal{M}$ accepts x if and only if $\mathcal{C}_x$ says 1.
5. The construction is carried out by a Turing machine $\mathcal{N}$ with space bound $O(\log_2 |x|)$. This Turing machine maps x to $f(x) = \mathcal{C}_x$.
6. Hence, we have a function f that is computable by a deterministic Turing machine in space $O(\log_2 |x|)$ for all input x such that $x \in Q \Leftrightarrow f(x) \in \text{CVP}$.
7. Therefore, $Q \leq_{\text{log-space}} \text{CVP}$ and CVP is $\mathbf{P}$-complete.

More P-complete Problems

1. **Monotone CVP:** The CVP problem remains **P**-complete even if all gates are restricted to $\vee$'s and $\wedge$'s (*i.e.*, no $\neg$'s).
2. **Lexicographically First Depth-First Search Ordering:** Given an undirected graph $G = (V, E)$ with fixed ordered adjacency lists and two vertices u and v, is vertex u visited before v in the depth-first search of G induced by the order of the adjacency lists?
3. **Breadth-First Search:** Given an undirected graph $G = (V, E)$ with a numbering on the vertices and two vertices u and v, is vertex u visited before v in the breadth-first search of G induced by the vertex numbering?
4. **Maximum Flow:** Given a directed graph $G = (V, E)$ with each edge e labeled by an integer capacity $c(e) > 0$, two distinguished vertices, source s and sink t, and a value f, is there a maximum flow of value f? That is, is the value of the maximum flow into the sink $\geq f$?

5. **Context-Free Grammar Membership:** Given a context-free grammar G and a string x, is $x \in L(G)$?
6. **Context-Free Grammar Empty:** Given a context-free grammar G, is $L(G)$ empty?
7. **Context-Free Grammar Infinite:** Given a context-free grammar G, is $L(G)$ infinite?
8. **Context-Free Grammar ϵ-Membership:** Given a context-free grammar G, is $\epsilon \in L(G)$?
9. **Point Location on a Convex Hull:** Given an integer d, a set S of n points in $\mathcal{Q}^d$ and a point $p \in \mathcal{Q}^d$, where $\mathcal{Q}$ is the set of rationals, is p on the convex hull of S?
10. **Gaussian Elimination with Partial Pivoting:** Given an $n \times n$ matrix A over rationals and an integer c $(1 \leq c \leq n)$, is the pivot value for the c^{th} column positive when Gaussian elimination with partial pivoting is performed on A?
11. **Game of Life:** Given an initial configuration of the Game of Life, a time bound t, and a designed cell c of the grid, is cell c live at time t?