Some machines defined by directed graphs

de Bruijn, N.G.

Published in:
Theoretical Computer Science

DOI:
10.1016/0304-3975(84)90048-3

Published: 01/01/1984

Citation for published version (APA):
SOME MACHINES DEFINED BY DIRECTED GRAPHS

N.G. DE BRUIJN

Department of Mathematics and Computing, Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands

Communicated by M. Nivat
Received September 1983

Abstract. This paper considers sorting machines defined by directed graphs. There is a level of
description where the operations depend on comparison of real numbers, and a more primitive
one on which the operations are in terms of bits. It is shown how the bit processing machines
can be transformed into the machines that process real numbers.

1. Introduction

In a previous paper [3] we considered a simple sorting machine, presented as a
simplified description of a machine invented by Armstrong and Rem [1]. The
description of Armstrong and Rem is on the level of processing single bits, whereas
in the simplified machine the possibility to compare bit strings, or, what amounts
to the same thing, comparing numbers, is assumed.

A machine more or less identical to that number processing machine was treated
by Chen et al. [4].

In the present paper we establish the relation between the bit processing descrip-
tion and the number processing description. We study this in a more general setting.
We introduce a bit processing machine $M_1$, defined by means of a directed graph,
and we show how it can be replaced by a machine $M_2$ processing bit sequences of
fixed word length. Finally we replaced $M_2$ by a machine $M_3$ which has a simple
structure again. The study of $M_3$ can replace the confusing picture of zeros and
ones wriggling through the graph of $M_1$. In Section 7 we shall specialize to the
machine of Armstrong and Rem. In Section 8 we apply the technique of this paper
to a new machine that sorts numbers presented to the machine in parallel.

2. The machine $M_1$

We consider a quadruple $(G, E, \text{tail}, \text{head})$, where $G$ and $E$ are sets, 'tail' and
'head' are mappings of $E$ into $G$. The elements of $G$ are called vertices, those of
$E$ are called (oriented) edges. We say that the edge $e$ runs from $\text{tail}(e)$ to $\text{head}(e)$, and in the figures this will be indicated by an arrow.

It is not excluded that $e_1 \neq e_2$ and nevertheless both $\text{head}(e_1) = \text{head}(e_2)$ and $\text{tail}(e_1) = \text{tail}(e_2)$.

If $P \in G$, the number of $e \in E$ with $\text{tail}(e) = P$ is called $\text{out}(P)$ (the 'outdegree' of $P$); the number of $e \in E$ with $\text{head}(e) = P$ is called $\text{in}(P)$ (the 'indegree' of $P$). We shall require that

$$\forall P \in G \quad \text{out}(P) = \text{in}(P)$$

(a directed graph with this property is usually called an Euler graph). We shall also require that all degrees are either 1 or 2; generalization to higher degrees is possible but not very attractive. If

$$\text{out}(P) = \text{in}(P) = 1,$$

then $P$ will be called an ordinary point; if

$$\text{out}(P) = \text{in}(P) = 2,$$

then $P$ will be called a switch.

At a switch $P$ we have two different edges whose head is $P$; these are called the inputs of $P$. We label these two edges: one of them is called the high input, the other one the low input. Similarly we label the edges whose tail is $P$: one is called the high output, the other one the low output. The distinction between high output and low output is essential for the machine's functions. The distinction between high input and low input is unessential: they only serve as labels used for our description of the switch settings.

The situation at a switch will be described by means of a set with three elements: 'through', 'back' and 'neutral'.

A state of the machine is obtained by attaching a bit (taken from the set \{0, 1\}) to each edge, and a switch setting (taken from the set \{'back', 'through', 'neutral'\}) to each switch.

In order to describe the actions of the machine, we refer to the set $T = \{0, 1, 2, \ldots \}$ as the set of time moments. To each $P \in G$ we attach a subset $T_P$ (the elements of $T_P$ are called the neutralization moments for $P$). In Section 3 we shall formulate conditions (3.1) and (3.2) which these $T_P$'s have to satisfy.

We shall describe how from a state of the machine (at time $t$) we get the next state (at time $t+1$). This will imply that if the state is given at $t=0$, it is completely determined for all $t \in T$. Let us denote the bit on edge $e$ at time $b$ by $b(e, t)$.

At an ordinary point $P$ the rule is simple: the bit on the incoming edge is transferred to the outgoing edge. That is, if $P = \text{head}(e_1) = \text{tail}(e_2)$, then $b(e_2, t+1) = b(e_1, t)$.

If $P$ is a switch, the rule is more complex. First assume $t \not\in T_P$.

(i) If at time $t$ the switch setting is 'through', then the bit on the high input is transferred to the high output, the bit on the low input is transferred to the low output. The switch setting remains 'through'.

(ii) If at time $t$ the switch setting is 'back', the bit on the high input is transferred to the high output, the bit on the low input is transferred to the low output. The switch setting remains 'back'.

(iii) If at time $t$ the switch setting is 'neutral', then the bit on the high input is transferred to the high output, the bit on the low input is transferred to the low output, the switch setting remains 'neutral'.
(ii) If at time $t$ the switch setting is 'back', then the bit on the high input is transferred to the low output, the bit on the low input to the high output. The switch setting remains 'back'.

(iii) If at time $t$ the switch setting is 'neutral', we apply the list which is given in Table 1.

<table>
<thead>
<tr>
<th>Time $t$</th>
<th>Time $t + 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>High input</td>
<td>Low input</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

If, finally, $t \in T_P$, the action is explained in two steps: first set the switch to neutral, then apply rule (iii).

We shall refer to the machine described here as $M_1$, at least if the $T_P$'s satisfy the extra conditions (3.1) and (3.2).

3. Processing binary sequences

We shall formulate a synchronization condition that guarantees that certain trains of bits running through the machine can be considered to stay together as trains. In Section 2 the sets $T_P$ played a role only if $P$ is a switch, but here the $T_P$'s will be considered for ordinary points too. The synchronization condition is

$$\forall e \in E \forall t \in T \quad (t \in T_{\text{tail}(e)} \iff t + 1 \in T_{\text{head}(e)}).$$

(3.1)

From now on we assume the synchronization condition to be satisfied.

We introduce a further simplifying condition about the $T_P$'s. We assume that there is a fixed positive integer $w$ such that for every $P \in G$ the set $T_P$ has the form

$$T_P = \{r_P, r_P + w, r_P + 2w, \ldots\}$$

(3.2)

with some integer $r_P$ with $0 \leq r_P < w$. By (3.1) it follows that for every $e \in E$ we have

$$r_{\text{head}(e)} = r_{\text{tail}(e)} + 1 \pmod{w}.$$

Let us call a pair $(e, t)$ (with $e \in E$, $t \in T$) a leader if $t \in T_{\text{head}(e)}$. The leaders move stepwise if $t$ proceeds: if $e$ and $e'$ are consecutive edges [in the sense that $\text{head}(e) = \text{tail}(e')$], then $(e, t)$ is a leader if and only if $(e', t + 1)$ is a leader. And we note that if $e_1$ and $e_2$ are the inputs of a switch, and if $(e_1, t)$ is a leader, then $(e_2, t)$ is a leader too.
If \((e, t)\) is a leader, we consider the sequence
\[
\beta(e, t) = (b(e, t), b(e, t + 1), b(e, t + 2), \ldots, b(e, t + w - 1)).
\]

We can associate to \(\beta(e, t)\) the number
\[
2^{w-1}b(e, t) + 2^{w-2}b(e, t + 1) + \cdots + b(e, t + w - 1).
\]
We write \(\beta(e_1, t_1) < \beta(e_2, t_2)\) if \(\beta(e_1, t_1)\) is lower in the lexicographic order than \(\beta(e_2, t_2)\). This happens if and only if the inequality holds for the associated reals.

If \(e\) and \(e'\) are consecutive edges, and if head(e) is an ordinary point, it is easy to see that
\[
\beta(e, t) = \beta(e', t + 1). \quad (3.3)
\]

Let us next consider a switch with inputs \(e_1, e_2\) and outputs \(e_3\)(high), \(e_4\)(low). From the rules of Section 2 it follows that (if \((e_1, t)\) is a leader)
\[
\beta(e_1, t + 1) = \max\{\beta(e_1, t), \beta(e_2, t)\},
\]
\[
\beta(e_3, t + 1) = \min\{\beta(e_1, t), \beta(e_2, t)\}. \quad (3.4)
\]

4. A simpler machine \(M_2\)

We can describe the behaviour of the machine \(M_1\) if we just look at the way the \(\beta\)'s move. We attach the number \(\beta(e, t)\) to \((e, t)\) if \((e, t)\) is a leader, but to non-leaders we do not attach anything. At ordinary points the \(\beta\)'s are just passed along (see (3.3)), at switches there are always two \(\beta\)'s arriving at the same time. If they are different, the larger takes the high output. If they are equal, they are just passed on to the outputs. Let us call this machine \(M_2\).

We remark that the behaviour of \(M_2\) does not determine what happens in \(M_1\) at an edge \(e\) before there has been any leader at that edge, i.e., at times \(t\) with \(t < \tau_{\text{head}(e)}\).

5. A different representation of the machine \(M_2\)

Let us define the notion "\(r\)-path". If \(r\) is an integer with \(0 \leq r < w\), then an \(r\)-path is a sequence \(e_1, \ldots, e_n\) of consecutive edges (\(\text{head}(e_i) = \text{tail}(e_{i+1})\), etc.) starting at a point \(P\) (i.e., \(\text{tail}(e_1) = P\)) with \(r_{i+1} - r_i\). It follows that the \(r\)-path also ends at such a point.

It is easily seen that the set \(E\) of all edges can be considered as the disjoint union of a set of \(r\)-paths (all with the same \(r\)). An example is given in Figs. 1 and 2, where we have \(w = 5, r = 2\). In Fig. 1 we have indicated the \(r_{i+1}\) at each point. The graph is drawn a second time in Fig. 2 in order to display a partition into \(r\)-paths, with
Some machines defined by directed graphs

Fig. 1. An $M_1$ machine, with indication of $r_0$'s.

Fig. 2. The graph of Fig. 1 dissected into 2-paths.

names $A_1, A_2, \ldots, A_6$ assigned to them. The paths are oriented by arrows corresponding to the arrows on their edges. The high outputs are indicated by heavy pieces of line.

We notice that for every time moment $t$ every $r$-path contains exactly one edge $e$ such that $(e, t)$ is a leader. So if we consider the actions of the machine $M_2$, with $\beta$'s moving through the graph, we observe that at each moment each $r$-path contains exactly one $\beta$. Transitions of $\beta$'s from one path to another take place at two occasions:

(i) at times $\equiv r \pmod{w}$ where the $\beta$ is just about to step into the next $r$-path,
(ii) at moments where two $\beta$'s get to a switch. Here the larger of the two is about to go to the $r$ path that contains the high output, the smaller is about to go to the other one.

Sometimes (i) and (ii) happen simultaneously. In Fig. 1 there is a switch with inputs on $A_4$ and $A_3$, and outputs on $A_3$ (the high output) and $A_5$. In such cases we connect the $r$-paths of the input to the $r$-paths of the output by dotted lines. (In Fig. 2 we have linked $A_4$ to $A_5$ and $A_3$ to $A_1$.) One might agree to always connect high input to high output, but this is by no means necessary.
The machine $M_2$ can now be represented graphically in an entirely different way. In Fig. 3 we have taken six points $A_1, \ldots, A_6$ corresponding to the $r$-paths of Fig. 2. They are connected by two kinds of arrows: single arrows and double arrows.

The system by which we get Fig. 3 from Fig. 2 is explained in Figs. 4, 5 and 6. If in Fig. 2 the head of an $r$-path $A_i$ leads into the tail of an $r$-path $A_d$ at an ordinary point, we draw a single arrow from $c$ to $d$ in Fig. 3. This rule is depicted in Fig. 4. If $P$ is a switch in Fig. 2, and $r_P \neq r$ (remember that in our example $r = 2$), then two $r$-paths pass through $P$. If $A_u$ and $A_v$ are these paths and if $A_u$ is the one that uses the high output, we draw a double-pointed arrow from $f$ to $g$ in Fig. 3. Moreover, we provide that double arrow with a number, viz. the $r_P$ of that switch. This rule is illustrated in Fig. 5.

Finally, if $P$ is a switch with $r_P = r$, then we combine the procedures of Figs. 4 and 5 into the one of Fig. 6. The dashed connections from $A_h$ to $A$, and from $A$, to $A_i$ are reflected in the single arrows from $h$ to $i$ and from $j$ to $k$. The fact that $A_i$
starts at the low output and \( A_i \) starts at the high output of the switch, is reflected in a double arrow from \( k \) to \( i \). And again the value of \( r_p \) is attached to the double arrow.

It follows from the construction of Fig. 3 that if several double arrows enter into or leave from a point (e.g., two double arrows entering into 6 and one leaving 6), then all have different switching times attached to them (here 3, 4, 0). That these switching times are different follows from the fact that in Fig. 2 these switchings correspond to different points of one and the same \( r \)-path.

The actions of the machine \( M_2 \) can now be described as follows. At any time we have a number attached to each point in Fig. 3. We write \( \beta(i, t) \) for the number attached to \( A_i \) at time \( t \). The transition from time \( t \) to time \( t + 1 \) is as follows. If \( t \neq r \) (mod \( w \)), then for every pair \( i, j \) such that there is a double arrow from \( A_i \) to \( A_j \), with switching time \( s \) satisfying \( t = s \) (mod \( w \)), we carry out a swap

\[
\beta(i, t + 1) := \min\{\beta(i, t), \beta(j, t)\},
\]
\[
\beta(j, t + 1) := \max\{\beta(i, t), \beta(j, t)\}.
\]  

If \( t = r \) (mod \( w \)), the action is two-fold: first shift along the single arrows, then apply the swaps. That is, first replace every \( \beta(i, t) \) by the corresponding \( \beta(k, t) \), if \( k \) is the index such that there is a single arrow from \( A_k \) to \( A_i \), then apply (5.1) for all the double arrows with switching time \( r \).

We shall call this machine \( M_1 \). As it will be clear from the above, the actions of \( M_1 \) can be mapped one to one onto the actions of \( M_2 \).

An example of the actions of the machine of Fig. 3 is presented in Table 2. The values of \( \beta(i, 0) \) are taken arbitrarily, the values for \( t > 0 \) follow from them. The table has been arbitrarily cut off at \( t = 10 \). For the sake of curiosity we remark that we get periodicity from \( t = 15 \) onward, since \( t = 45 \) shows the same state as \( t = 15 \).

On the right we have indicated what actions have to be taken at time \( t \) in order to get to the next line; 'swap\((i, j)\)' denotes the operation described in (5.1), and 'shift' stands for one step along each single arrow. In every line the order of the swaps is irrelevant, since no point can be involved in two swaps at one and the same moment.

Note that one and the same machine \( M_2 \) may correspond to several machines \( M_1 \), according to the freedom we have in choosing the set of \( r \)-paths, and in tying \( r \)-paths together at switches in the exceptional case that the switching time equals \( r \).
It will be clear that the machine $M_3$ has nothing to do with word length any more. We can consider any linearly ordered set $W$ (e.g., the set of all real numbers), and define the notion ‘state’ of the machine of Fig. 3 as a mapping of the set of points into $W$.

6. The machine of Armstrong and Rem

A typical case of the Armstrong–Rem machine can be considered as an $M_1$ machine of the kind depicted in Fig. 7.

There is a number of ‘rings’ (here 4), all of the same length (which is the word length; here 5). The number of rings may be replaced by an arbitrary positive integer, and the word length by any integer $>1$. Between any pair of connected switches there are two paths with total length $w$, but the separate paths may vary in length as long as they are at least 1. We take $r = 0$.

In Fig. 8 we show a possible dissection into 0-paths, and we deliberately take them such that every ring is a 0-path. In Fig. 9 we show the $M_3$ which is derived from Fig. 8 in the same way Fig. 3 was derived from Fig. 2.

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 2

Table: | $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>
We now describe how Armstrong and Rem use their machine for sorting. Let $n$ be the number of rings and $w$ the word length. At time 0 all rings are filled with 0's, and on the input queue (see Fig. 7) we have $n$ sequences of $w$ bits each, representing the $n$ numbers to be sorted (the most significant bits are on the right), and to the left of these we have $n$ sequences consisting of $w$ ones each. The contents of the output queue at time 0 are irrelevant.

The claim is that at time $2nw$ the output queue contains the $n$ numbers sorted as to non-decreasing order (the most significant bits on the left), and, to the left of these, $n$ sequences of zeroes. This means that the sequence of sorted numbers turns up in the output queue immediately after the last one of the numbers to be sorted has been pushed from the input queue into the interior of the machine.

In order to check this claim we just inspect the behaviour of $M_3$. This $M_3$ is precisely the machine we described in [3] as a simplified form of the machine of Armstrong and Rem, and for which we established that it sorts correctly. According to the correspondence between $M_1$, $M_2$, $M_3$ it follows that $M_1$ sorts correctly.

7. A parallel sorting machine

We shall now describe an $M_1$ machine that sorts a set of $n$ numbers with word length $w$ if these numbers are offered to the machine in parallel. The machine does not depend essentially on $w$. In Figs. 10 and 11 we depict the machine for the cases $n = 4$ and $n = 5$, the general case will be an obvious generalization. In both cases the input and output lines on which the numbers travel are taken to be infinitely long and free of switches. We may take $w$ arbitrarily, but it is slightly easier to think of the case $w > n$. The synchronization times are constant on horizontal lines, and are indicated on the right of the figures.
We use a standard way to split the graph into 0-paths; this is illustrated in Fig. 12 for the case $n = 4$. In Fig. 13 we have drawn the machine $M_1$ for the case $n = 6$, and that is a machine that sorts indeed. For this well-known fact we refer to [2, Section 5]. Therefore $M_1$ sorts the $n$ words of length $w$.

We can also vary the word length during the action. If we offer a batch of $n$ words of length $w_1$, followed by a batch of $n$ words of length $w_2$, etc., then in the output every batch is sorted. We only have to take care that every switch is neutralized at the moment the first bit of a new word arrives.
Some machines defined by directed graphs

Fig. 12. Dissection into 0-paths.

Fig. 13. $M_3$ version of the parallel sorter.

References