Computer Science Journal of Moldova, vol.**9**, no.1(25), 2001

Two railway circuits: a universal circuit and an NP-difficult one

Maurice Margenstern

Abstract

In this paper, first we construct a railway circuit based on three types of switches and on crossings. Such a circuit is able to simulate the computation of any Turing machine, in particular of a universal one. That result was proved by Ian Stewart in [3]. In this paper, we give another construction, indeed a simpler one: here we show that it is possible to simulate the computation of any register machine, from which the same conclusion can be derived.

The switches that are used in the universality result are re-used in order to prove another result. Define the accessibility problem for railway circuits to consist in the following. The circuit is given with, at the same time, two points of the circuit, one is called the source point, the other one is called the target point and a fixed number of other points, called flip-flop switches, that can be initialized arbitrarily. The problem is to decide, whether or not there is an initialization of the flip-flop switches such that it is then possible to go from the source to the target. This problem is proved to be NP-complete.

Key-words: circuits, switches, registers, universality, NP-completeness.

Introduction

Simulating the computation of a Turing machine by railway circuits is not a novelty, see [3]. In that paper, the author facilitated the work by assuming that the alphabet of the Turing machine has only two symbols, $\{0, 1\}$ for instance, 0 being the blank. On the basis of the

principles that are stated in [3], it is possible to simulate a Turing machine on any alphabet: encode the alphabet in binary, which necessitates a certain number of bits, say k; in the simulation that is given in [3], there is a set of n circuits that corresponds to each case of the tape of a machine on $\{0, 1\}$. Later, we remind the definition of those circuits that all encode 0 or all encode 1, depending on the configuration that is given on the tape. That can be extended to any alphabet encoded on k bits, by associating a group of n circuits for the first bit, 2 groups for the second, according to what was read as a first one, 4 groups for the third bit and, step by step, 2^k for the k^{th}.

In the first section of this paper, we introduce a circuit that is simpler and that makes use of the same railway switches as in [3]. Moreover, as a locomotive does not build the tracks, at least *a priori*, the circuit is infinite, which is also the case for the circuit that is built in [3].

As we have at our disposal a universal circuit in the *infinite* field, it seemed to me that probably, in the *finite* field, the same assortment of switches should allow us to construct an NP-complete problem. This is what we do in the second section of the paper. That correspondence between the infinite case and the finite one is not the single common feature between the two sections. Technically, for an analogous purpose, but in a different way, they make use of the same basic circuit that we shall call *elementary selector*. We shall describe it in section I.4. and it will be used again without change in section II.2.

1 A universal circuit

1.1 The principle of the simulation

Our simulation consists in simulating the execution of a register machine. It is known that two registers are enough to simulate a Turing machine, see [2], and so, by such a simulation, we shall obtain a universal computation.

Our first goal will be to simulate registers. We constitute them as units that contain a one bit information: 0 or 1. The rôle of the

locomotive is to execute the computation by changing the configuration of the switches of the circuit, each time when it passes through each of them. The configuration produces both the contents of the registers of the simulated machine and the address of the instruction under action in the simulation.

With this point in view, we associate a track to each instruction. The first section of the track is used in both directions: on the way on, to go to the register on which an operation must be performed; on the way back, to turn to the execution of the next instruction. A particular situation occurs for the decrementation instructions for which the way back does not lead to the next instruction in the case when decrementing the register was not possible because its content was already zero.

First, we remind the definition of the types of switches that we shall use. Next, we define the circuit that simulates one unit of a register. Later we simulate the circuit that dispatches the return between various instructions. Indeed, as it will be later clear from the principle of simulating registers, the main problem for the locomotive is to go back to the instruction that has led it to that register.

At last, the *current configuration* of the circuit is defined as the set, at a given time, of the states of all the switches that are present. The computed value has to be read in the part of the circuit that simulate the registers. We shall see how to do this in the description of that part of the circuit.

1.2 The switches

As we indicated in the introduction, there are three types of switches. We have also crossings, that do not need switches. It is also possible to rule out crossings if bridges or tunnels are allowed, *i.e.* if access to the third dimension is permitted.

In order to define the three types of switches that we use here, remind that a switch is a point from which a track splits. From one side of the switch there is a single track and from the other side there are two tracks. The switch allows a train to go from the unique track

to one of the two others. We shall say that this is the *active* passage of the switch. When the locomotive goes from one of the two tracks to the unique track, we shall speak of a *passive* passage of the switch.

The way in which the locomotive turns to one of the two tracks during an active passage of the switch defines the types of switches that we shall consider here. The three types of switches are:

- the fix switch: the choice of the track is always the same in an active passage;
- the flip-flop switch: it changes the track to be taken at the next passage when the locomotive passes through it; the locomotive must go through such a switch in the active sense only, as in [3];
- the memory switch: the next active passage is defined by the track that was used in the last passive passage through the switch.

Notice that fix and memory switches can be used in the passive sense. Only the flip-flop switch is a *one way* switch: in the active sense.

In the graphic representation of the circuits, we represent the different types of switches as follows:

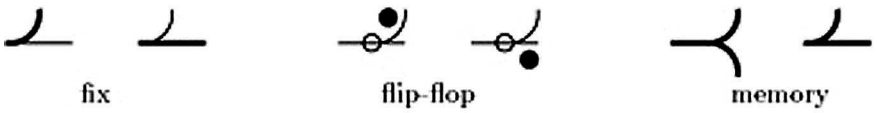


Figure 1. The switches of the circuit
For the flip-flop switch, we indicate the position of the switch in the initial configuration, which will be called the default position.

Naturally, the switches of the circuit are obtained by displacements in the Euclidean plane, the distance being fixed once for all.

1.3 Registers

3.a *The problem*

Our point is to define the circuit that is associated to a unit of a register. Such a unit must be used both for incrementing and decrementing. We take again the elementary circuit of [3] that associates a flip-flop switch with a memory switch, see Figure 2, below.

The circuit works as follows. The positions of the flip-flop switch and of the memory switch are always associated as it is indicated in the figure. The memory switch directs the locomotive to track **i** if and only if the flip-flop switch directs it to the fix switch of the other track, where $\mathbf{i} = \mathbf{1}$ or $\mathbf{2}$. When the circuit is in the position **i** (both flip-flop and memory switches pointing to **i**), the locomotive that enters through E goes out through O_i. But later, if it is needed, it is possible to change the position of the circuit, according to the track on which the locomotive was. Up to a point, passing through E is a passage to *read*, in order to know the *content* of the circuit that is encoded by the position of the above considered switches. Passing through U, the flip-flop switch, is a passage to *write*. Here, *writing* means that we change what is actually written. Passing through the flip-flop switch performs a *not* operation on the content of the elementary circuit.

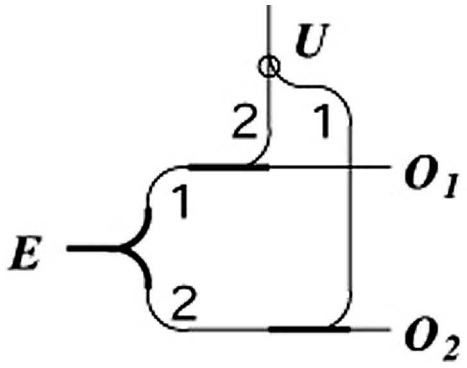


Figure 2. The elementary circuit

It is also possible to associate several elementary circuits with synchronized positions. This allows us to construct multiple and programmable switches. That possibility is used in [3], and we use it also

here.

The basic idea is that the unit circuit of a register associates two elementary circuits. One of them is read while incrementing. The other one is read while decrementing. In both cases, the second elementary circuit is not visited, unless it is necessary to change the content of the circuits. The register is made of infinitely many replicates of the unit circuit.

3.b *The unit circuit*

The circuit is given in Figure 3, below.

Figure 3 produces two units of the register. This allows us to illustrate how the circuit works, in all cases. Inside a unit, say **i** for instance, the left hand elementary circuit is called $\mathbf{i}_i$, because it is used for incrementing. The right hand elementary circuit is called $\mathbf{i}_d$, because it is mainly used while decrementing.

Incrementing the register

In the case of incrementing, the locomotive enters the register through track I. Assume that all the units, up to **i**, also included, are in position **1** and that, starting from **i+1**, all units are in position **0**. Arriving through A_i that is a fix switch, the locomotive takes the left hand track of the circuit $\mathbf{i}_i$, *i.e.* the route BCP_i which leads it a bit further on track I, in view of the visit of the circuit $\mathbf{i+1}_i$. The fix switch that the locomotive meets in A_{i+1} makes it enter the circuit $\mathbf{i+1}_i$ where, this time, it takes the right hand track, *i.e.* the route BCF_{i+1} that leads the locomotive to the flip-flop switch E_{i+1} of the circuit $\mathbf{i+1}_i$. The flip-flop turns now to position **1** and the locomotive arrives in C_{i+1}. At that point, which is the memory switch of the circuit, the locomotive passes through the memory switch in the passive sense on the left hand track, *i.e.* the route EGC. From now on, this position, *i.e.* position **1**, is memorized by C_{i+1}. The locomotive goes on through E'_{i+1} where the flip-flop switch, in position **0**, sends it to G'. The switch in E' turns now to the position **1**; the locomotive goes through G', then through C' where the memory switch will now be set to **1**, then through B' where the fix switch sends it to S_{i+1}. At that

point, the locomotive is on the track R which leads the locomotive back to the entrance of the register. We shall examine that part later.

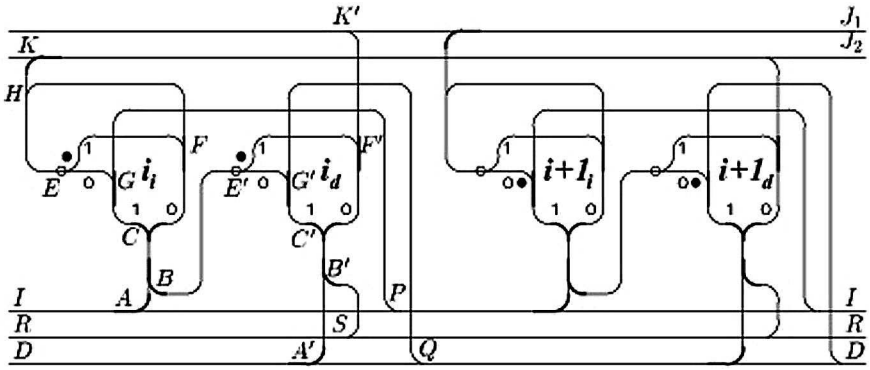


Figure 3. The circuit of a unit that contains 1
To be more realistic, we represented the circuit by using only right rails, 'vertical' or 'horizontal' ones, or rails in quarters of circles. Of course, a more elegant graphism can be involved.
In the sequel, switches will be indicated by letters, as in the above figure. They will be indexed by the number of the circuit if the context does not allow us to avoid ambiguities. Notice that the switches of the right hand part of the unit are characterized by the prime which does not occur in the denotation of switches in the left hand part. In the case of a route, the possible index is set only on the last letter.

From this analysis, it ensues that if the visited unit **i** is in position **1**, its switches E and E' are not visited. Accordingly, the position of the unit is not changed and the locomotive turns next to the unit **i+1**. On the other hand, if the visited unit is in position **0**, which the locomotive sees by visiting the elementary circuit i and by passing through F_i, the unit changes of position, turning from **0** to **1**. In that case, the locomotive visits **both** elementary circuits of the unit to change its position, in order to go back, next, to the control circuit of the register.

It is not difficult to see that, under these conditions, the number of units in position **1** goes from i to $i+1$, which was to be performed.

Decrementing the register

Now, consider the case of a decrementation.

First, assume that the register is not zero, and assume that the positions of the units are the same as previously. We see that first, the locomotive visits the circuit $\mathbf{i}_d$ of the unit **i**, as in the case of the incrementing. But this time, the locomotive enters the unit through the fix switch A', which is on the track D. As in the case of incrementing, if the locomotive *reads* a **1**, it is lead to C' on the route $C'G'Q'_i$ to go to Q on the track D. Next, it will visit the unit **i+1** as soon as it reaches the fix switch A'_{i+1}. As in the case of incrementing, assume that the unit **i+1** is in position **0**. In the circuit $\mathbf{i+1}_d$, this time, the switch C' send the locomotive along the route $C'F'K'_{i+1}$. Notice that the locomotive goes out from the unit **i+1** which will remain in position **0**. In K'_{i+1}, the locomotive is sent on one of the two tracks J whose rôle will be clear a bit later. That J track leads the locomotive back to the unit **i** through the fix switch in K_i. The locomotive arrives next in E_i and so, it will change the content of the unit **i** that will turn from **1** to **0**. Indeed, going through E_i, next the locomotive follows the route $FCBE'$, as the switch E_i was in position **1**, giving the position **0** to the switch C_i. That latter position is also the new position of the switch E_i. Of course, C_i will keep that information until the locomotive passes through that point in the passive sense. When it arrives in E'_i, the locomotive performs on the circuit $\mathbf{i}_d$ the same operation as it performed on the circuit $\mathbf{i}_i$, *i.e.* to make the circuit to turn from position **1** to position **0**. Accordingly, when the locomotive passes through B'_i, the fix switch that is there sends the locomotive to the track R. As in the case of incrementing, the locomotive goes back to the circuit that controls the entrance of the register.

It is not difficult to see that in this case, the number i of the units that are in position **1**, that is assumed not to be zero, turns to $i-1$, what was supposed to be done.

As in the key stage of the decrementation the locomotive goes out

of $\mathbf{i+1}$ in K'_{i+1} near the circuit $\mathbf{i+1}_d$ and as it goes back to $\mathbf{i}$ through K_i near the circuit $\mathbf{i}_i$, switches K_i and K'_i cannot stand on the same track. Hence two J tracks are needed, J_1 and J_2 as it is indicated on Figure 3, in order to correctly insure the return to the unit $\mathbf{i}$, starting from the unit $\mathbf{i+1}$. As a convention, we may decide that J_1 leads to the units with an even number and that J_2 leads to the ones with an odd number. Consequently, when the content of the register is zero, the locomotive goes back to the circuit that controls the entrance of the register by the track J_2, as it is indicated on Figure 5.

As a consequence, in our simulation, the fact of coming back to the control circuit of the entrance of the register through the track J_2 is the test of nullity of the register that is needed to simulate the execution of the decrementation, also in that case.

And so, we shall turn now to a closer look on the access to a register and to the return from it.

3.c *Input/output of the register*

A priori, the locomotive enters the register by the tracks I or D and it returns from it by the track R or by the track J of the first unit. From the previous analysis, it ensues that when the locomotive is on the track R, it no more remembers whether it entered the register by the track I or by the track D.

So, by entering the register, the locomotive must somehow *mark* the fact that it arrived by the track I or by the track D. A solution is given by the circuit that is below illustrated by Figure 4.

Let us check that the circuit, called C on Figure 5, allows us to do what we expect: if the locomotive enters it by a track I, it returns from it to the register by another track I and when it enters the circuit by the track R, coming back from the register, it goes out from C by the same track I by which the locomotive entered C. If the locomotive enters C by a track D to decrement the register, it goes out from C by another track D that leads it into the register. When the decrementation is performed, the locomotive goes back to C by the track R and then, it goes out from C by the same track D by which it entered C. On the

other hand, if the decrementation could not be performed, because the register was already null, the machine goes back to C by the track J, and it goes out from C by another track J that leads it to the further execution of the program.

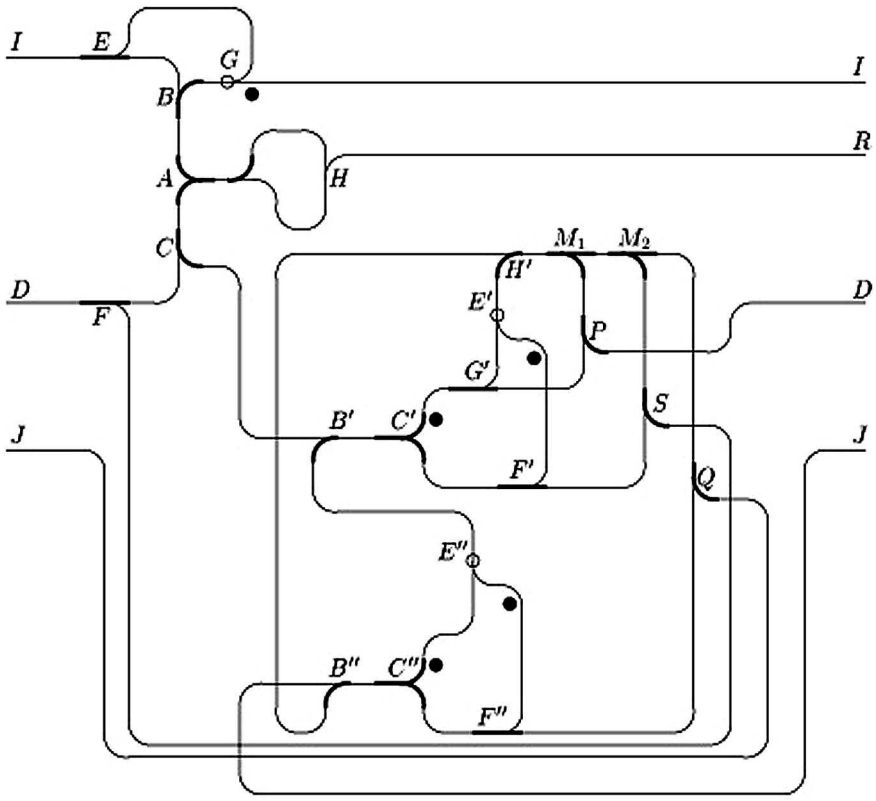


Figure 4. The control circuit for input/output of a register

Let us start by the case of an incrementation

Arriving by the track I in E, the locomotive goes on until A where it indicates to the memory switch to remember the position AB. In A, there is, side by side, a memory switch and a fix switch. The memory

switch remembers whether the current instruction is an incrementation or a decrementation, and the fix switch makes the locomotive enter a loop that goes back to A. When it is again in A, the locomotive is sent to G by the fix switch of B. In G, the flip-flop switch sends the locomotive to the first unit of the register by the track I, because, by default, the position of the flip-flop was that one. Now, the flip-flop switch of G is set to the position GE.

Coming back from the incrementation, by the track R, the locomotive arrives in H, where a fix switch that is always crossed in a passive way sends it on the loop that leads to A. There, the memory switch sends the locomotive back to B, then to G. The flip-flop switch that is in G in position GE sends the locomotive to E where the locomotive goes back to the former track I, in order to perform the next instruction of the program. Now, the switch in G is again set to the default position: later it will be possible for the locomotive to go to the register to increment it as long as needed.

Let us now consider the case of a decrementation and assume that it can be performed.

In that case, the locomotive arrives in A via FC and the memory switch in A will be set to the position AC. When it goes back in A after the loop, the locomotive is sent in C to C' where it enters an elementary circuit. The entrance of that circuit is a memory switch whose default position is indicated by Figure 4. The switch sends the locomotive on the route $G'PM_1H'$. The memory switch of M_1 is now set to the position M_1P and the fix switch of H' sends the locomotive back to the elementary circuit, this time through E'. Consequently, the locomotive will change the position of that circuit from the previous one, namely from $C'G'$ to $C'F'$. This change also indicates that the flip-flop switch in E' was changed. Notice that, as in the case of a unit of a register, the elementary circuit of C' is associated with another elementary circuit. As it goes out from the former circuit through B', the fix switch of that point sends the locomotive to E''. The passage through E'' makes it possible to set the second elementary circuit on the position $C''F''$. Notice that the second circuit is a bit reduced in comparison with the circuit of C': indeed, the second circuit is read

only in the case of a return from the register through the track J.

The locomotive goes out from the second circuit through B'' where the fix switch sends it to H', then M_1. Now, in M_1, the switch memorized the position M_1P. And so, the locomotive goes to P where the fix switch sends it on the track D that leads to the first unit of the circuit.

As we assume that the decrementation was performed, the locomotive goes back through the track R. Consequently, it arrives in H, then in A. In that point, the memory switch sends the locomotive back to C and then, again in the elementary circuit via $B'C'$. In C', the switch has memorized the position $C'F'$. Hence, the locomotive follows the route $C'F'SM_2$ and in M_2, the memory switch is set to the position M_2S. The locomotive follows its way to M_1, setting the memory switch to M_1M_2. Then, the locomotive is again in H' from where it is again sent to the elementary circuit, this time for writing. Both circuits are now set to the default position, namely positions $C'G'$, $C''E''$, which can be easily checked by the positions of the flip-flops at the beginning of the second passage into both circuits. The locomotive goes on to B'' where it is sent back to the route $B''H'M_1$. The switch in M_1 memorized M_1M_2. In M_2, the switch memorized M_2S and so, the locomotive goes to S. There, a fix switch sends it to F where the locomotive takes again the track D, to turn to the next instruction of the program.

It remains to consider the case of a decrementation that cannot be performed because the register is null.

In that case, everything happens as in the previous case until the locomotive arrives in the first unit of the register.

Now, instead of going back by the track R, the locomotive goes back by the track J. It enters the control circuit in B'' and in C'', the switch has memorized the position $C''F''$. In F'', the fix switch sends the locomotive on the route $F''QM_2M_1H'$. In the points M_2 and M_1, the memory switch are respectively set to the positions M_2Q and M_1M_2. From H', the locomotive is sent to both elementary circuits. As in the case of a successful decrementation, the locomotive sets back

the flip-flop and memory switch in E', E'', C' and C'' to their default position, *i.e.* the positions that they had before the locomotive entered the control circuit, before its entrance of the register.

The locomotive goes out from the elementary circuits through B'' where the fix switch sends it back to H'. From there, it goes to M_1, then M_2, as far as the switch in M_1 memorized the position M_1M_2. In M_2, the switch memorized the position M_2Q and so, the locomotive arrives in Q. There, the fix switch sends the locomotive to another track J that performs the jump to the next instruction to be executed.

Consequently, the circuit C that is represented by Figure 4 actually executes the selection of the operations that are described at the beginning of the section **I.3.c**.

1.4 The general setting of the circuit

Considering the circuits that are defined by Figures 3 and 4, we can now give a simplified representation of a register, which is displayed by Figure 5.

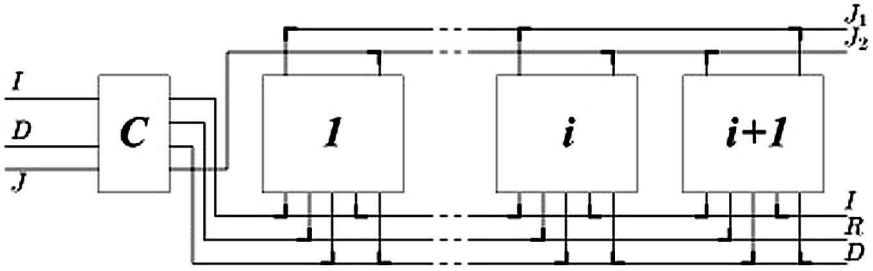


Figure 5. Simplified scheme of the organization of a register

In that representation, each square is denoted by a natural number and it contains two elementary circuits that are connected as in Figure 3. The line of input/output corresponds to the entrances and exits, on the tracks, respectively, I, D, R and J. The circuit that controls the input/output of the register that is displayed on Figure 4 is contained in the square labeled by C.

However, that latter control circuit alone does not allow us to organize the circuit that simulates the program. Indeed, in general, the program may hold *several* incrementation instructions and *several* decrementation instructions that operate on the same register. In our simulation, we have then several tracks of type I and several tracks of type D that arrive to the control C of the register. We have seen how the circuit C is able to send the locomotive back from the track R on the right track I or D on which it entered C. But the circuit C alone cannot send back the locomotive on **the** track I or D through which the locomotive arrived from the program. *A fortiori*, when the locomotive goes back by the track J, the circuit cannot direct the locomotive to the right track to perform the needed next instruction.

We shall separately consider the case of several incrementations and of several decrementations.

4.a *Several incrementations*

That case is easily dealt with by a sequence of memory switches that generalizes the mechanism that is constituted by the switches M_1 and M_2 of Figure 4.

Assume that k incrementation instructions operate on the same given register. We have k tracks, I_1, ..., I_k to arrive at the entrance of circuit C.

These tracks are organized as it is displayed by Figure 6. They are all connected to a common track I by memory switches:

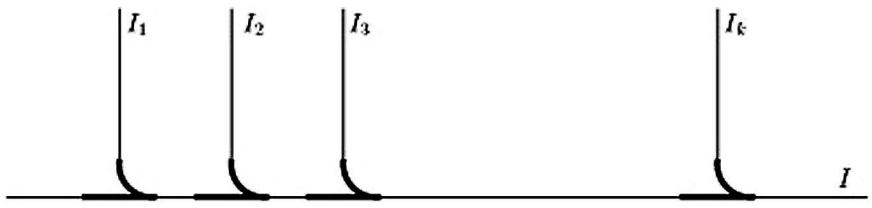


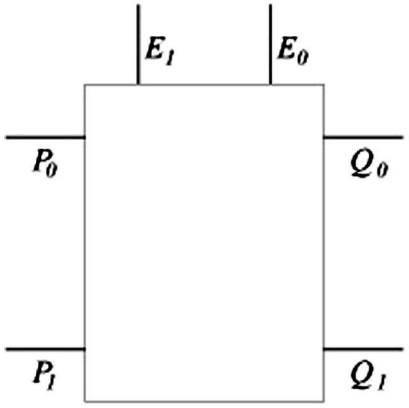
Figure 6. Distributing incrementation instructions on the same register

The proof that this organization is correct is obvious.

4.b *Several decrementations*

In the case of several decrementations, the circuit of Figure 6 cannot solve our problem. Indeed, the return from a decrementation may occur by two different tracks depending on whether the decrementation was performed or not.

In that case, we construct a circuit that is inspired from the circuit that controls the entrance of the register. The rôle of the new circuit is to *mark*, on the way *on* of the locomotive, the tracks D and J that can be taken on the way back of the locomotive. The circuit will use an element that we call *elementary selector*, which works as follows.



entrance		exit
way on	return	exit
E_0	Q_0	E_0
E_0	Q_1	E_1
P_0	Q_0	P_0
P_0	Q_1	P_1

Figure 7. Scheme of the elementary selector

The table of Figure 7 indicates the point of exit from the circuit, depending on the entrance point both for the way on and for the way back.

A solution for such a circuit is provided by Figure 8.

Let us check that the circuit is correct:

First of all, notice that on the way *on*, if the locomotive arrived by P_0, it goes out through Q_0 without visiting the elementary circuits of C, C' and C'', respectively. Denote by **0** the default position of the switches of Figure 8, *i.e.* the positions displayed by the figure. The

elementary circuits all remain in position **0**, by definition. Indeed, the fix switch of U directs the locomotive to S'', then S' and then V, that are on the same track. The locomotive exits on that track by Q_0.

Assume now that the locomotive arrives through E_0. The fix switch of A is crossed in the active sense and so, the locomotive takes the route BCG, as far as the switch in C has memorized the position CG which, by definition, is the default position, **0**. The later route is RM_1HE. Notice that the switch in M_1 will memorize the position M_1R. When the locomotive arrives in E, later on it will go via EC but not through G, then via $BE'F'$, $B'E''F''$ and than via $B''HM_1$. This means that the locomotive sets all three elementary circuits of C, C' and C'' in position **1**. In M_1, the memory switch has memorized the position M_1R and so, the locomotive is directed to R. There the fix switch directs the locomotive to N, then to V by a fix switch. The locomotives takes the track UV starting from V and exits by Q_0.

Consequently, either the locomotive enters the elementary selector by E_0 or by P_0, the exit of the locomotive is always through Q_0. However, in the first case, the elementary circuits are all set to **1**, while in the second case, they all remain to **0**.

Now, consider the case of the return.

First, assume that the locomotive goes back by the point Q_0. It arrives to V from where it is sent to N then to B'.

We have two cases, depending on whether all the three circuits were set to **0** or to **1**. If the position is **0**, which corresponds to an entrance through P_0 on the way on, the locomotive tests that in C' where it arrives after its passage in B'. It is then sent to G', then to U and from there, to the exit through P_0.

According what we already saw, from E, the locomotive will change the position of the three elementary circuits from **1** *to* **0**, *which restores the default position for possible later passages. Once the locomotive arrived in B'', it is directed to H and then to M_1. The switch in M_1 has memorized the position M_1M_2. And so, the locomotive reaches M_2 where the switch has memorized the position M_2S. Consequently, the locomotive reaches S where the fix switch sends it to E_0 through S' and S''.*

If it is the position **1**, which is the case for the three circuits of the selector, it corresponds to an entrance through E_0 on the way on. Again, the locomotive arrives in C' via B', but this time it is sent to F' as far as the switch in C' has memorized the position $C'F'$. Next, the locomotive runs over the route SM_2M_1HE and the switches in M_2 and M_1 memorize the positions M_2S and M_1M_2, respectively.

The switches in A and S'' are fix ones. The stroke that indicates the fixed direction in A is made twice in order to avoid confusion with a memory switch. Now, assume that during the return, the locomotive enters the selector through the point Q_1. In W, that it meets next, it is directed to the third elementary circuit to read the current value shared by the three circuits. Accordingly, it arrives in C'' through B'', and then in G'' or in F'' depending on whether the group of elementary circuits is in position **0** or it is in position **1**.

If position **0** is read, the locomotive follows the route $G''Q$ and meets again the track P_1Q_1; it exits through P_1.

If position **1** is read, the locomotive is now sent to F'' as far as the switch in C'' has memorized the position $C''F''$. Later, the locomotive follows the route TM_2M_1HE, and the switches in M_2 and M_1 memorize the positions M_2T and M_1M_2, respectively.

According what we already saw, the locomotive goes to E and from there, it changes the position of the three elementary circuits from **1** to **0**, which restores the default position for possible later passages. When it arrives in B'', the locomotive is sent to H and then to M_1. The switch in M_1 has memorized the position M_1M_2. The locomotive reaches M_2. There, the switch has memorized the position M_2T. Accordingly, the locomotive reaches T where a fix switch sends it to E_1, and the

corresponding track crosses the track P_0Q_0 a bit on the left of U in Figure 8.

Thus, we checked that the group of circuits that is presented in Figure 8 satisfies the indications that are given in the table of Figure 7.

Now it is possible to see how we can organize the circuit that manages the decrementations that arrive to the same register. Let $D_1, \ldots, D_k$ be the considered decrementations. The circuit makes use of $k-1$ elementary selectors $S_2, \ldots, S_k$. They are associated according to the scheme of Figure 9. The track D_i corresponds to the entrance E_0 of the elementary selector S_i for $i \geq 2$. Similarly, the track J_i corresponds to the exit E_1 of selector S_i. The track D_1 corresponds to the entrance P_0 of selector S_2 and the track J_1 corresponds to the exit P_1 of the same selector. Moreover, for $1 \leq i < k$, the exit Q_α of the selector S_i is connected with the entrance P_α of the selector S_{i+1}, where $\alpha \in \{0, 1\}$. At last, the point Q_0 of the selector S_k is connected with the entrance/exit D of the controller of the register and the point Q_1 is connected with the exit J of the controller.

If the locomotive arrives by the track D_i for $i \geq 2$, it sets the selector S_i to **1**, and it leaves all selectors S_j to **0** for $j < i$ because it did not visit them and, as it exits from S_i by Q_0, it enters all S_j by P_0 for $j > i$ because it leaves also those selectors by their exit Q_0, according to what we saw for the elementary selector. If the locomotive arrives by the track D_1, all selectors remain to **0**. And so, at most one selector is set to **1** when the locomotive enters the register. Consequently, on the way back, as long as the visited selectors are set to **0**, the locomotive goes back to the previous selector. When it arrives to the single selector set to **1**, the one by which it arrived, the table of Figure 7 shows that the exit is performed on the expected track. The table shows that it is also the case even if the locomotive does not find any selector set to **1**. It means that it arrived before by the track D_1.

Accordingly, the locomotive goes on its way to a part of the circuit that corresponds to the execution of the remaining part of the program: either the instruction which follows D_i if the return occurred through the track D_i, or the instruction whose address is given in the instruction

corresponding to D_i if the return occurs through the track J_i.

4.c *The sequencing of the instructions*

Let us remind that the instructions of a register machine are numbered and that they belong to one of the following three types:

inc R_i ; increment the register R_i
dec R_i, **k** ; decrement the register R_i if $R_i > 0$,
; perform **k** if $R_i = 0$
jump k ; turn to performing the instruction numbered with **k**

We shall decide that if the program contains N instructions that belong to one of the above three types, the numbers of the instructions range from **0** up to **N-1**. The halting of the program is given by the instruction **jump N**.

From the principle of grouping the instructions of the program that operate in the same way on the same register, it is possible to define the circuit that deals with a register by the following scheme which combines the circuits that are defined by the previous figures:

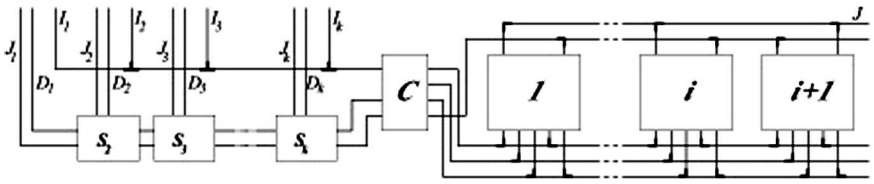


Figure 9. Distributing the operations on the same register

It remains to assemble all the groups of instructions that are combined as in Figure 9 in order to reproduce the scheduling of the program of the simulated register machine.

Remind that, *a priori*, a track is associated to an instruction and that the sense of motion corresponds to the execution. In the case of an operation on a register, incrementation or decrementation, the route of the locomotive can be split into three parts: the part *way on*, the part

execution and the part *return*.

The part *way on* starts from the beginning of the track devoted to the instruction in the scheduling part of the circuit until the entrance in the first unit of the register. And so, we consider the circuit C in that part of the route. Accordingly, the locomotive is on the track I or on the track D, depending on the type of the operation that it performs on the register.

The part *execution* is performed inside the register: from the moment when the locomotive enters the circuit of the first unit until the moment when it takes the track R or the track J to return.

The part *return* leads the locomotive to the starting point of the track that is associated to the instruction that must be performed after the one that was just performed. It is the next instruction of the program if an incrementation or successful decrementation was performed. It is another instructions in the case of an unsuccessful decrementation.

We have dealt with almost all the parts of the route of the locomotive that concerns the execution of an instructions. The scheduling of the instructions that is defined by the program of the register machine is implemented in a particular circuit, the *sequencer*. We shall now explain its principle by illustrating it on a simple program.

Let $C_1, \ldots, C_N$ be the sequence of the instructions of the program of the register machine that we simulate, in the order given by the program. For each register R_i, define $\mathcal{I}_i$ to be the set of the instructions of the program that increment the register and define $\mathcal{D}_i$ to be the set of instructions that decrement it. For the instructions of $\mathcal{I}_i$, the way on and the way back are performed on the same track and the turning to the next instruction is very simply realized as it is shown on figure 10.

For an instruction belonging to $\mathcal{D}_i$, the circuit is more complex and it is represented on Figure 11. The same figure illustrates the case of a jump from instruction C_i to instruction C_j, with $i < j$.

Consider the following sequence of instructions for a machine with four registers that we shall here call x, y, z and w for convenience.

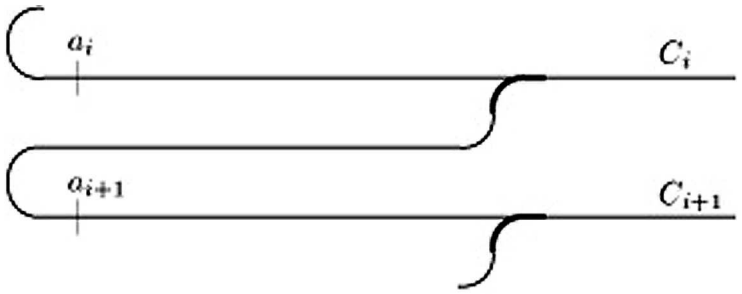


Figure 10. Detail of the sequencer circuit
The points a_i and a_{i+1} are simply marks that allow us to materialize the entrance on the track that is associated to the corresponding instruction.

```
1 :  dec   x, 5          8 :  dec   y, 12
     inc   z                    inc   z
     inc   w                    inc   w
     jump  1                    jump  8
5 :  dec   w, 8          12 : dec   w, 15
     inc   x                    inc   y
     jump  5                    jump  12
                         15 :
```

We easily check that the program adds the contents of the registers x and y into z, using w to save the contents of x and y during the computation. Starting from the initial configuration $(x,\ y,\ 0,\ 0)$, that denote the respective contents of the registers x, y, z and w, the computation leads to the final configuration $(x,\ y,\ x+y,\ 0)$.

The proof that this program is correct is easy and we leave it to the reader. It can be done by induction on x and y successively and on w for the replication of the content of w on x and y, according to the corresponding part of the program.

The circuit that is associated to that program is completely given in Figure 13 except the registers: for them, only finitely many units can be given!

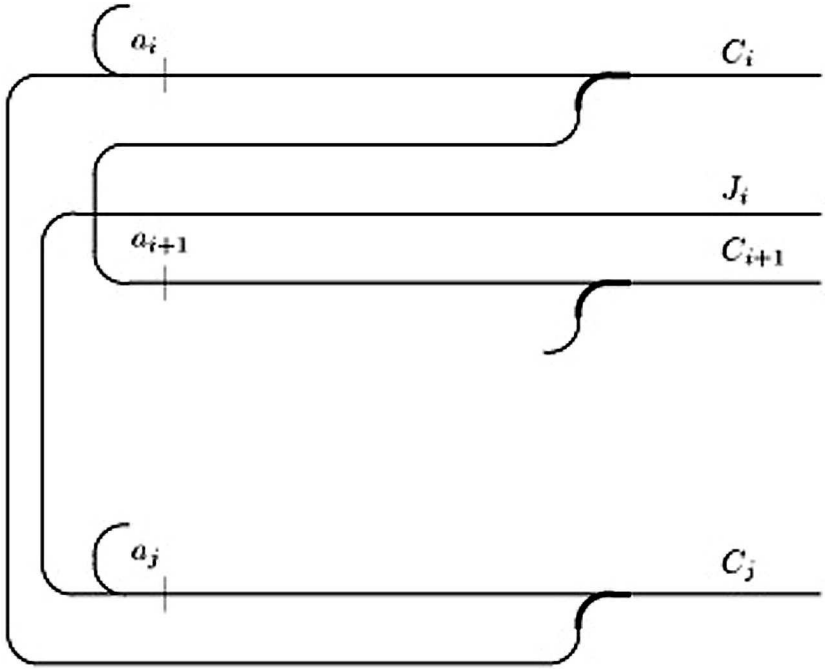


Figure 11. Another detail of the sequencer circuit
In that figure, the instruction C_i is a decrementation that may lead to instruction C_j; that latter one is followed by a jump to instruction C_i in the program.

As here we deal with a particular circuit, we take advantage of it to give a few variants with respect to the previous indications.

As an example, as only incrementations operate on register z, the tracks that correspond to decrementations are stopped: the track D below the register and both tracks J, above it. It could be possible, in an analogous case where instructions of the same type only operate on the register to simplify again the circuit by modifying the unit circuit: instead of two elementary circuits, it would be enough to construct it with a single one, as indicated in Figure 12.

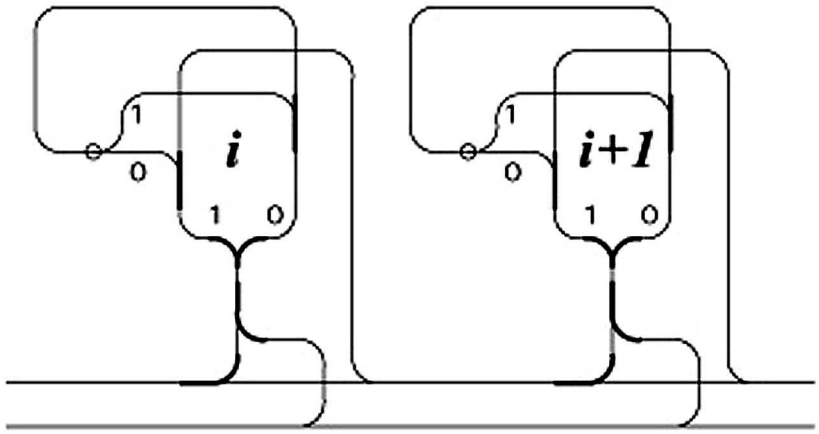


Figure 12. Simplified unit of register for incrementation only

Another remark: as it can be seen on Figure 13, the halting of the computation is obtained by sending the machine back to the buffer that is indicated at the beginning of the track that is associated to the first instruction. We can see that point as the platform of a station which is used as departure, for the first instruction of the program, and as arrival, for the last one.

2 A NP-difficult circuit

As indicated in the introduction, the experience shows a certain relation between the possibility to simulate a Turing machine and the possibility of finite models that generates very complex computations. As an example, let us quote the case of the firing-chip game: in each node of a non directed graph, there is a finite number of chips at disposal. At each time, each node sends one chip to its neighbours if and only if it contains at least as many chips as there are edges that start from the node. A lot of papers were devoted to the computations that can be obtained that way on finite graphs, see [1] for exact references. The growing complexity of the obtained computations inclined the authors of [1]

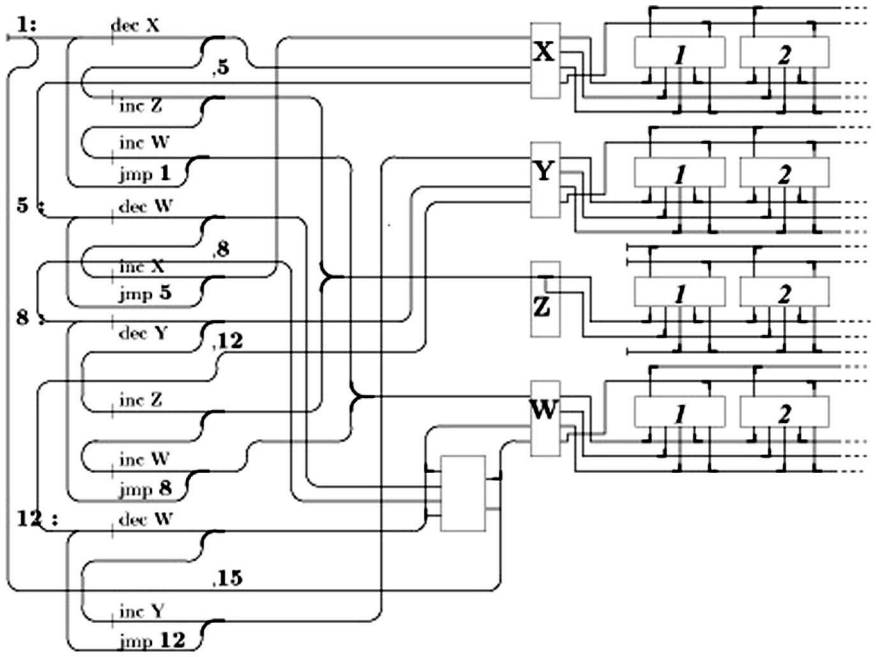


Figure 13. The circuit that is obtained from the program of $x + y$

to prove that there is an infinite graph on which that game simulates a universal Turing machine. Here, we can argue in the *opposite*: as we know how to construct an infinite railway circuit that is able to simulate a universal Turing machine, there must be finite circuits that lead to *difficult* problems. We shall prove this now, by constructing a circuit to which it is possible to associate an NP-complete problem.

3 Simulation of SAT

Let us remind that an instance of 3-SAT consists in a formula of the propositional calculus that is given in conjunctive normal form. This means that the formula is a finite conjunction of clauses, each clause

being a disjunction of the form $y_i \vee y_j \vee y_k$, where y_α, called *literal*, is either x_α, or $\neg x_\alpha$, $x_1, \ldots, x_n$ being the *variables* of the considered instance. An *assignment* is a mapping f from $\{1..n\}$ into $\{0,1\}$ which gives to x_i the value *true* if $f(i) = 1$ and otherwise the value *false*. The problem which consists in finding at least one assignment for which the formula is true, is NP-complete.

Given a formula $F(x_1, \ldots, x_n)$ in normal form, we define a circuit with a certain number of *free* flip-flop switches and with two fixed points A and B such that there is a setting of the free switches such that it is possible to go from A to B if and only if there is an assignment of the variables for which $F(x_1, \ldots, x_n)$ takes the value *true*.

We take again the idea of the circuit that is given in Figure 8.

Indeed, we organize the simulation by a progressive evaluation of the conjunction $C = \bigwedge_{i=1}^{s} \sigma_i$, where σ_i is the i^{th} clause, by the formulas $C_i = \bigwedge_{k=1}^{i} \sigma_k$, for $i = 1, \ldots, s$.

The locomotive arrives at a point of the circuit that correspond to the evaluation of C_{i+1} if and only if the evaluation of C_i was completed with *true*. Consequently, it arrives on the part of the circuit that corresponds to the clause $\sigma_{i+1} = y_{\alpha_1} \vee y_{\alpha_2} \vee y_{\alpha_3}$. The locomotive is then sent to another part of the circuit that represents the value of x_{α_1} under the considered assignment. We shall later examine that part. The locomotive goes back to the part of the circuit devoted to the clause by following a track 0 or a track 1, depending on the value that is read for x_{α_1} and depending on whether y_{α_1} is x_{α_1} or its negation. If the return is done by the track 1, the locomotive is directly sent to the evaluation of the next clause. If the return occurs by the track 0, the locomotive turns to the evaluation of y_{α_2}. For that purpose, the locomotive goes to the part of the circuit associated to x_{α_2}. We proceed as previously. After the reading of x_{α_3}, the locomotive goes to the next clause, only if it goes back by the track 1. If the return is performed by the track 0, there is a clause evaluated to *false* and so, the whole conjunction is false for that assignment. By convention, the locomotive goes back to A in that case.

Finally, when after reading x_{α_3} in the last clause the locomotive uses the track 1, it is then switched to a track that leads directly to B. Consequently, by the construction of the circuit, it is possible to go from A to B if and only if there is an assignment of values for $x_1, \ldots, x_n$ such that the formula $F(x_1, \ldots, x_n)$ is true. In terms of the circuit, the formula $F(x_1, \ldots, x_n)$ is translated by the sequence of fix switches that stay on the line that leads from A to B, which is parallel to the track that goes back to A, see Figure 14.

4 The simulating circuit

The organization of the circuit is the following.

A first track, say I, goes from A to B. It has as many fix switches as there are literals in the formula $F(x_1, \ldots, s_n)$. This is what is suggested by the Figure 14 where a general scheme to represent the simulating circuit is presented.

On that first track from A to B, the fix switches connect to tracks that lead to the control circuits C_1, C_2, ..., C_n of each variable. From the circuit C_i, the locomotive goes through the track 0 to the lecture of x_i. Without modifying that value, the locomotive goes back to the control circuit by the track 0 or the track 1, depending on what was read as an assignment value.

From the circuit C_i, the locomotive comes back to the literal source on the track AB. It takes the same track as on the way on if the read value is *false*, that is symbolized by 0 on the below schemes. It takes an appropriate track if the read value is *true*. That track is symbolized by 1. The return by the track 0 leads to the next literal if the locomotive does not go back to the last literal of the clause. That return leads to A by a track which is parallel to the track AB if the last scanned literal was the last one of the clause. The return by the track 1 leads to the first literal of the next clause. It leads to B if the locomotive came from the last clause.

We shall now have a closer look on each one of the circuits C_i and x_i.

The circuit is illustrated by Figure 15, below.

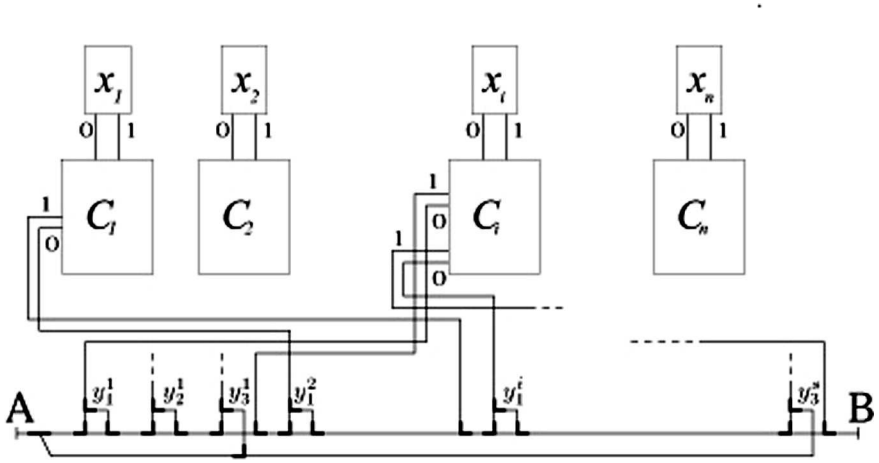


Figure 14. Principle scheme for a circuit that simulates 3-SAT

First, the circuit sends the locomotive to E where there is a flip-flop switch the position of which encodes the assignment value: if the value is *true*, the locomotive is sent on the track 1 and the switch is set to the track 0. The opposite actions happen if the value is *false*. And so, assume that the assignment value is *true*. The locomotive goes through the memory switch of C and then is sent to F by a fix switch. The flip-flop switch of F sends the locomotive back to E and it is set on the other track, that leads to C. The locomotive goes again through E, this time taking the track 0, while the flip-flop is set to **1** which was its initial setting. The locomotive goes back to C and now, the switch of C memorizes the sense **1**, its initial setting and the locomotive reaches F. From F it goes to C again, while the flip-flop is reset to its default position. Now in C, the locomotive is sent on the direction **1** and so it goes later on on the track 1. The reader will easily check that if the assignment value were *false*, the locomotive would exit from the circuit through the track 0 and that the switches would be reset on their initial values.

This return to the initial positions of the switches guarantees that

any reading of an assignment value leaves the value unchanged.

The circuit C_i

It is clear that the circuit C_i is a stack of elementary selectors of the same kind as the ones that we constructed in §I.4.

Tracks **0** and **1** correspond to what was read in the circuit x_i as the assignment value of the variable.

There is no change to what was just said when the literal y_i is x_i. When the literal is $\neg\ x_i$, the return route must be modified after the *exit* from the corresponding selector, because the same assignment value of x_i may be used either for x_i or for $\neg\ x_i$ in different clauses. The right occurrence is marked by the return track that is taken at the exit from the elementary selector that is associated to that clause.

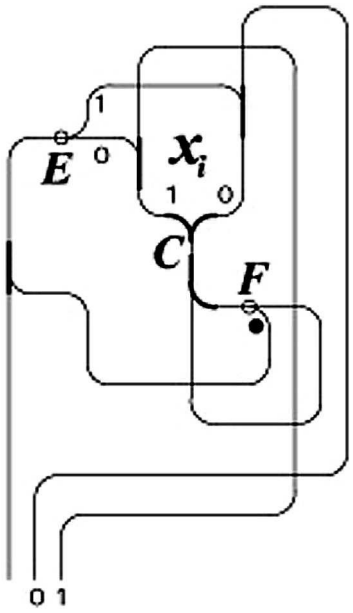


Figure 15. The circuit for reading the assignment value of a variable

A way to obtain that result consists in doing what is represented

in Figure 16. In the figure, the exits from two selectors corresponding to the same C_i circuit are represented. One of them corresponds to a clause where $y_i = x_i$, the other one to a clause where $y_i = \neg\, x_i$.

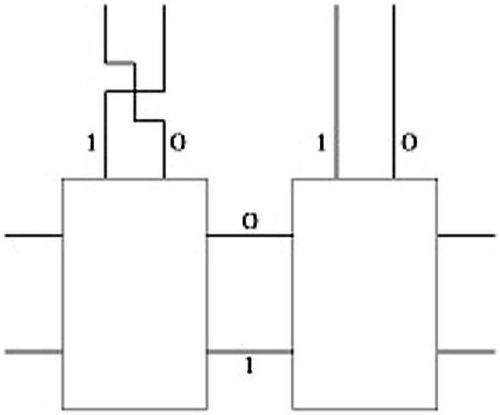


Figure 16. Exits of selectors, depending on the nature of y_i

It is easy to check that these changes correspond to the interpretation that are given to the clauses according to the assignment values of the variables.

The **NP**-*complete problem*

Consider the following problem:

The railway circuit problem:
Instance : A finite circuit that is constituted of rails and switches as it is indicated in the first section of this paper, two points A and B of the circuit and n flip-flop switches of the circuit defined as to be *free*.
Question : Is there a setting of the positions of the free switches such that it makes it possible for a locomotive that starts from A to arrive in B?

As the just constructed circuit reduces that problem to 3-SAT, it

ensues that the problem is NP-complete, because it is not difficult to check that the number of elements that constitute the just constructed circuit is bounded by $P(n)$, where P is a fixed polynomial.

This allows us to state:

Theorem – *The railway circuit problem is an* **NP***-complete problem.*

Acknowledgement

The idea of the first part of this paper arose from a discussion with Alain Colmerauer at the café of Metz railway station while waiting for my train to Paris. Let me here thank Alain for this very enlightening discussion.

References

[1] Goles E., Margenstern M., *Universality of the chip-firing game*, TCS, **172, 1-2**, pp.91–120, 1997.

[2] Minsky M., *Computation : Finite and Infinite Machines*, Prentice Hall (Englewood Cliffs, N-J), 1967.

[3] Stewart Ian, *A Subway Named Turing*, Mathematical Recreations in *Scientific American*, pp.90–92, sept. 1994.

Maurice Margenstern,
L.I.T.A., EA 3097, Université de Metz,
I.U.T. de Metz, Département d'Informatique,
Île du Saulcy, 57045 Metz Cedex, France,
E–mail: *margens@lita.univ – metz.fr*

Received April 10, 2001

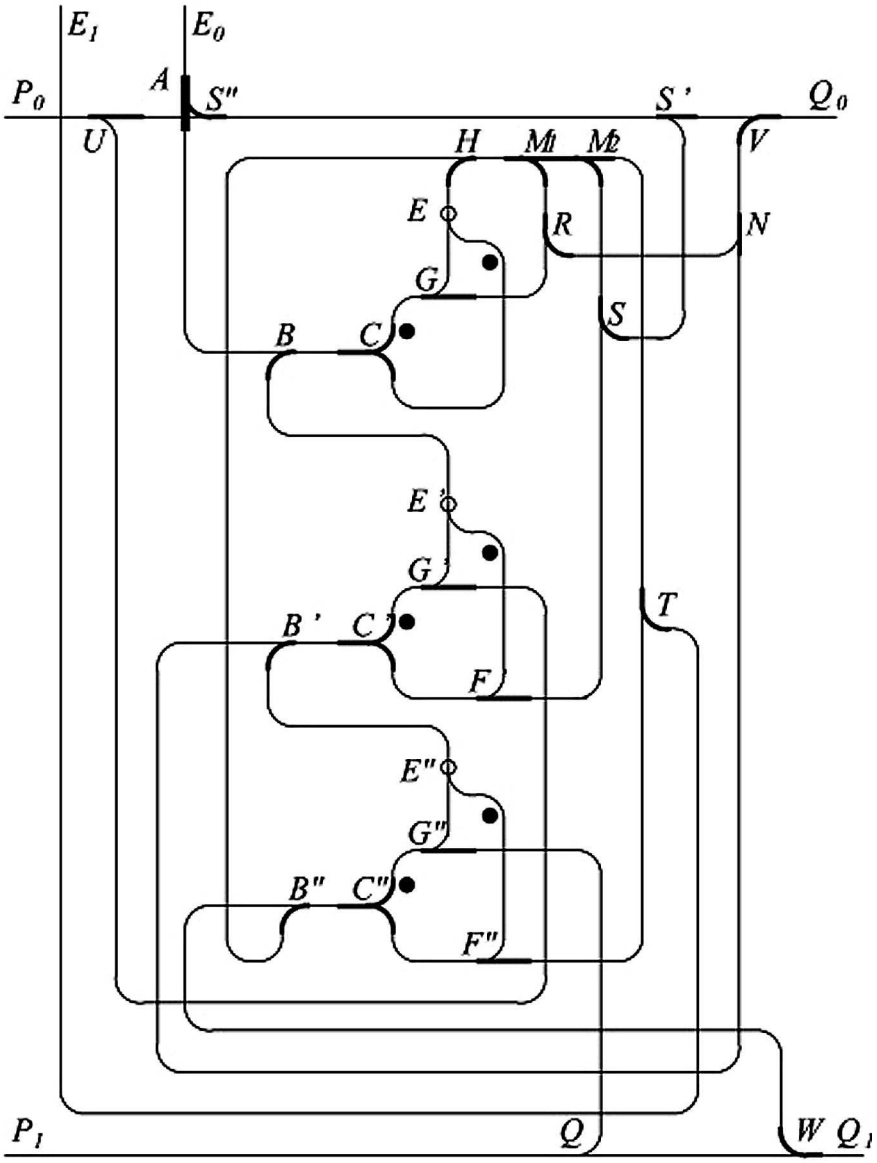


Figure 8. Scheme of the elementary selector