This leads to the following important question that we address in this section: *Is the Invariance Thesis true for all Turing machine models?*

2.1.1. Description of the Turing machine models

In the Turing machine model, the memory consists of a finite collection of tapes divided into tape cells. Each cell is capable of storing a single symbol from the worktape alphabet. For each tape there exist one or more channels (*read-write heads*) connecting the finite control to the tape. Each head is positioned at some tape cell. Several heads may be located at the same cell.

In the program of the Turing machine, various types of instruction have been integrated into a single type: the machine reads the ordered set of symbols from the worktapes which are currently scanned by the heads. Depending on these symbols and the internal state of the finite control, the appropriate instruction will cause the machine to overwrite some or all of the scanned symbols by new ones, move some or all of the heads to an adjacent cell on the corresponding tape and enter a new state in the finite control.

The simplest version is the single-tape Turing machine which can be described by a tuple $M=\langle K, \Sigma, P, q_0, q_f, b, \Delta\rangle$, in which

- K is a finite set of internal states;
- Σ is the tape alphabet;
- P is the program stored in the finite control, consisting of a finite set of quintuples $\langle q, s, q', s', m\rangle \in K\times\Sigma\times K\times\Sigma\times\Delta$; here $\Delta=\{L, 0, R\}$ is the set of possible head moves: Left, No Move or Right; the meaning of a quintuple $\langle q, s, q', s', m\rangle$ is: if the finite control is in state q and the head is scanning symbol s, then write symbol s', perform move m and let finite control enter state q';
- q_0 and q_f are two special elements in K denoting the initial and the final state, respectively;
- b is a special tape symbol called *blank* which represents the contents of a tape cell which has never been scanned by the head.

In the single-tape model there is no special input or output tape. In the initial configuration the input is written on the one available tape and the unique head is positioned over the leftmost input symbol. If one wants the model to produce output, one normally adopts an ad hoc convention for extracting the output from the final configuration. (For example, the output may consist of all nonblank tape symbols written to the left of the head in the final configuration).

If we denote configurations of M in the format $\$\Sigma^*K\Sigma^*\$$, with the state symbol written in front of the currently scanned tape symbol, the transitions between two successive configurations can be described by a simple *context-sensitive grammar*. For example, for every instruction $\langle q, s, q', s', R\rangle$ one includes the production rules $(qsa, s'q'a)$ for every $a\in\Sigma$, together with the rule $(qs\$, s'q'b\$)$ for the blank symbol b. Similar rules encode the behavior of left-moving instructions or instructions where the head does not move. This transformation of the program into a grammar establishes a one-one correspondence between computations of the machine and derivations in the grammar. This syntactic representation of computations opens the way to encode Turing machine computations in a large variety of combinatorial structures, like tilings