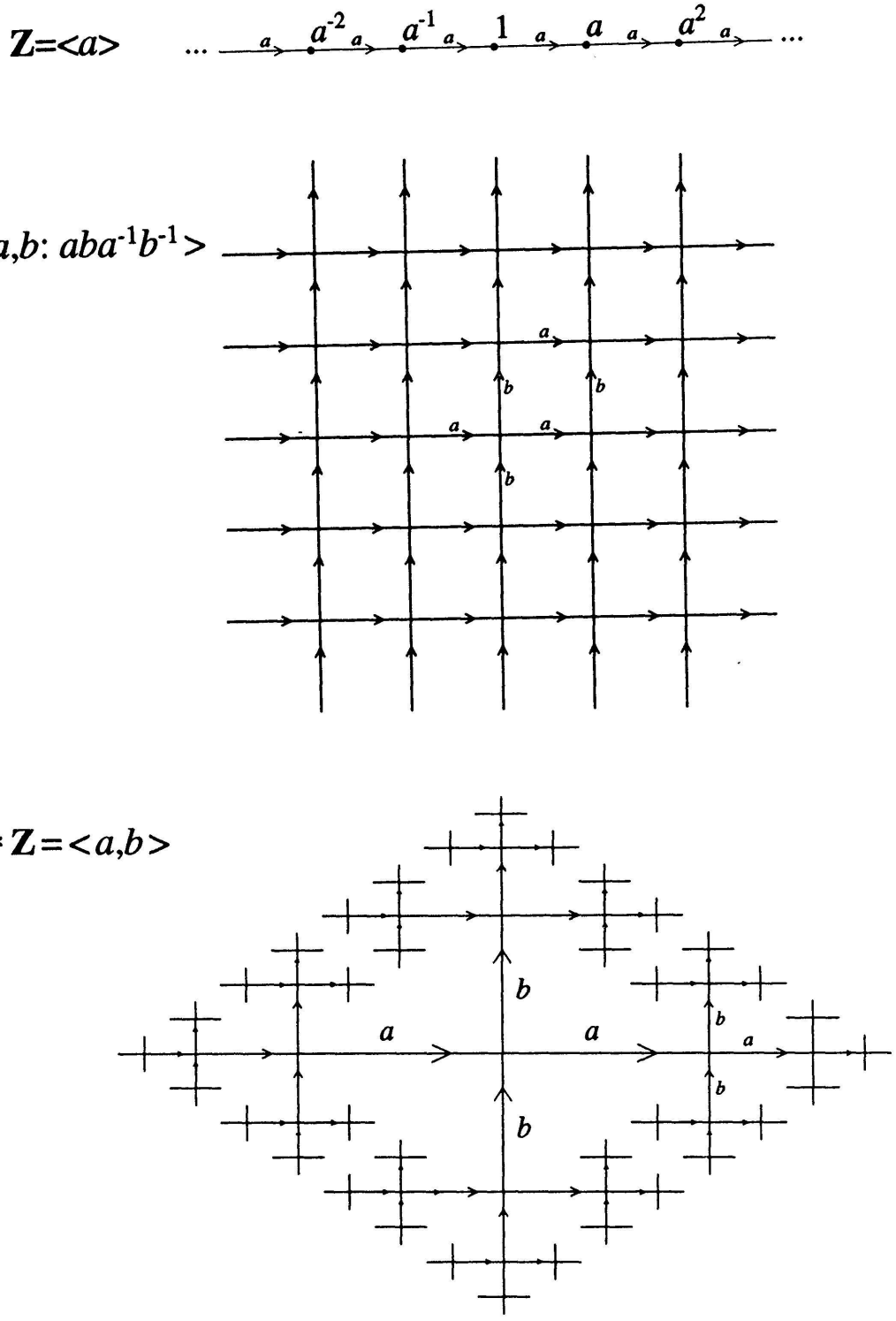

FIGURE 1

Cayley Graphs of $\mathbf{Z}$, $\mathbf{Z} \times \mathbf{Z}$ and $\mathbf{Z} * \mathbf{Z}$ with the standard generators

2. THE BEGINNINGS OF AUTOMATIC GROUPS

The main ideas behind automatic groups began with another beautiful paper of Cannon. In the second decade of this century Dehn solved the word problem for surface groups by using the geometry of the hyperbolic plane. In

1984 Cannon extended Dehn's solution to the word problem to cocompact discrete groups of hyperbolic isometries ([Ca1]). Perhaps the main idea of Cannon's paper is that one can "see" the Cayley graphs of such groups. What does it mean to see the Cayley graph of an infinite group? For example, the Cayley graph of $(\mathbf{Z}, S)$ with $S = \{1\}$ is simply the real line with a vertex at each integer. When drawing a picture of $\Gamma_S(\mathbf{Z})$, we draw only the two-ball, say, and then a trailing line of dots $\cdots$. By this we really mean to repeat the picture of the two-ball in our heads off to infinity, thinking also of the linear recursion $n \mapsto n + 1$. To "see" the Cayley graph should mean to have a picture of some finite ball around the origin and some finite machine which tells us how to piece copies of this ball together out to infinity. Cannon suggested as an open problem that one might "Formalize the notion that a Cayley graph can be described by linear recursion, and devise efficient algorithms for working out that recursion for many examples." The idea is that if such a linear recursion exists, which should happen whenever there is some pattern in the Cayley graph, then we can build a picture of what the group looks like, and from this picture we can construct algorithms to do computations in the group, such as solving the word problem.

The next layer of foundation was provided by Thurston, who gave a formal definition of the "linear recursion" Cannon spoke of. Thurston did this by using finite state automata (FSA for short), the simplest type of machines which have been studied thoroughly by computer scientists for nearly forty years. It seems interesting that Gilman was independently exploring the use of finite state automata for normal forms in groups (see, e.g. [Gi]), although with no (explicit) discussion of geometry. The details of the basic theory of automatic groups were worked out at Warwick by Epstein, Holt and Paterson. The use of finite state automata is partly motivated by their success in both the theory and applications of computer science; most word-processors (including 'vi') construct finite state automata for tasks such as word searches, and many compilers use FSA during lexical and syntactical analysis. In order to understand automatic groups we'll first need to have some understanding of finite state automata.

3. Finite state automata

Given some finite set of letters $\mathscr{A} = \{a_1, \ldots, a_n\}$, we want to pick out a nice subset of the set $\mathscr{A}^*$ of all words in the letters a_i (one can view $\mathscr{A}^*$ as the free monoid generated by the elements of $\mathscr{A}$). A subset $L \subseteq \mathscr{A}^*$ is called a *language*. Informally, a *finite state automaton* W over $\mathscr{A}$ is a finite directed