

Chapter 2

Situational Action Systems

Abstract Situational aspects of action are discussed. The presented approach emphasizes the role of situational contexts in which actions are performed. These contexts influence the course of an action; they are determined not only by the current state of the system but also shaped by other factors as time, the previously undertaken actions and their succession, the agents of actions and so on. The distinction between states and situations is explored from the perspective of action systems. The notion of a situational action system is introduced and its theory is expounded. Numerous examples illustrate the reach of the theory.

In this chapter, structures more complex than elementary action systems are investigated. These are *situational action systems*. Before presenting, in a systematic way, the theory of these structures, we shall illustrate the basic ideas by means of two examples.

2.1 Examples—Two Games

2.1.1 Noughts and Crosses

Noughts and crosses, also called tic-tac-toe, is a game for two players, **X** and **O**, who take turns marking the spaces in a 3×3 grid. Abstracting from the purely combinatorial aspects of the game, we may identify the set of possible states of the game with the set of 3×3 matrices, in which each of the entries is a number from the set $\{0, 1, 2\}$. 0 marks blank, 1 and 2 mark placing Xs and Os on the board, respectively. (It should be noted that many of the 3^9 positions are unreachable in the game.)¹ The rules are well known and so there is no need here to repeat them in detail. The initial state is formed by the matrix with noughts only. The final states are formed by matrices, in which all the squares are filled with 1s or 2s; yet there is neither a column nor a row nor a diagonal filled solely with 1s or solely with 2s (the state of a draw). Neither are there matrices in which the state of ‘three-in-a-row’

¹One may also encode each position by a sequence of length 9 of the digits 0, 1, 2.

is obtained; i.e., exactly one column or one row, or one diagonal filled with 1s or 2s (the victory of either of the players). The direct transition R from one state u to w is determined by writing either a 1 or a 2 in the place of any one appearance of a nought in matrix u . There is no transition from the final states u into any state. (Relation R is therefore not total.) There are only two elementary actions: A_X and A_O . The action A_X consists in writing a 1 in a free square, whereas the action A_O consists in writing a 2 (as long as there are such possibilities). A_X and A_O are then binary relations on the set of states. The above remarks define the elementary action system $\mathbf{M} = (W, R, \{A_X, A_O\})$ associated with noughts and crosses.

Not all the principles of the game are encoded in the elementary system \mathbf{M} . The fact that the actions are performed alternately is of the key importance, with the provision that the action A_X is performed by the player (agent) \mathbf{X} and A_O by \mathbf{O} . We assume that the first move is made by \mathbf{X} . We shall return to discuss this question after the next example is presented.

The game is operated by two agents \mathbf{X} and \mathbf{O} . The statement that somebody or something is an agent, i.e., the doer/performer of a given action, says something which is difficult (generally) to explicate. What matters here is that we show the intimate relations occurring between the agent and the action. If we know that the computer is the agent of, for example, the action A_X (or A_O), then this sentence entails something more: in the wake of it there follows a structure of orders, i.e., a certain program that the computer carries out. It is not until such a programme is established that we can speak, in a legitimate way, of the agency as the special bond between the computer as an agent and the action being performed. When, on the other hand, it is a *person* that is a player (i.e., he is the agent of one of the above-mentioned actions), then in order to establish this bond it is sufficient, obviously, to know that this person understands what he is doing, knows the rules of the game (which he can communicate to us himself), and is actually taking part in the game.

The game of noughts and crosses can also be viewed from the perspective of the theory of automata because the game has well-determined components such as: a finite set of states, a two-element alphabet composed of atomic actions A_X , A_O , the initial state, the set of terminal states, and the indeterministic transition function between states. In consequence, we obtain a finite automaton accepting (certain) words of the form $(A_X A_O)^n$ or $(A_X A_O)^n A_O$, where $n \geq 1$. However, as is easily noticed, the above perspective is useless in view of the goals of the game. We are not interested here in *what* words are accepted by the above-described automaton (since it is well-known that these are actions A_X , A_O which are performed alternately), but *how* the successive actions should be carried out so that the victory (or at least a draw) is secured.

The details above concerning the game noughts and crosses are fully understood by human beings. One can say even more: such detailed knowledge of the mathematical representation of the game is not necessary. It suffices just to have a pencil and a sheet of paper to start the game. Still, if one of the players is to be a computer (a defective creature) seeing that it is not equipped with senses, the above rules are not sufficient

and must be further expanded on in programming language of the computer. It is not hard, though, as a competent programmer can easily write a suitable program on his own.

2.1.2 Chess Playing

We shall use the standard **algebraic notation** (AN) (but in a rather rudimentary way) for recording and describing the moves in the game of chess. Each square of the chessboard is identified by a unique coordinate pair consisting of a letter and a number. The *vertical* rows of squares (called *files*) from White's left (the queenside) to his right (the kingside) are labeled *a* through *h*. The *horizontal* rows of squares (called *ranks*) are numbered **1** to **8** starting from White's side of the board. Thus, each square has a unique identification of the file letter followed by the rank number.

The Cartesian product of the two sets $X := \{a, b, c, d, e, f, g, h\}$ and $Y := \{1, 2, 3, 4, 5, 6, 7, 8\}$, i.e., the set $X \times Y$, is called the *chessboard*. The elements of $X \times Y$ are called *squares*. The *black squares* are the elements of the set

$$\{a, c, e, g\} \times \{1, 3, 5, 7\} \cup \{b, d, f, h\} \times \{2, 4, 6, 8\},$$

and the *white squares* are the elements of the set

$$\{a, c, e, g\} \times \{2, 4, 6, 8\} \cup \{b, d, f, h\} \times \{1, 3, 5, 7\}.$$

The game is played by two players (agents): *White* and *Black*. Each of them has 16 pieces (chessmen) at his disposal. *White* plays with the white pieces while *Black* with the black ones. Any arrangement of pieces (not necessarily all of them) on the chessboard is a *possible position*. We consider only the positions where there are at least two kings (white or black) on the chessboard. Thus, a position on the chessboard is any non-empty injective partial function from the set *White pieces* \cup *Black pieces* into $X \times Y$ such that the two kings belong to its domain. (Not all possible positions appear during the game; some never occur in any game.)

Let W be the set of all possible positions. Instead of the word 'position' we will also be using the term 'chessboard state' or 'configuration'. (The drawings showing the chessboard states are customarily called diagrams.)

The relation R of the direct transition between the chessboard states is determined by the rules of the chessmen's movements. Thus, the two states u and w are in the relation R , i.e., $u R w$, if and only if in the position u one of the players moves a chessman of a proper colour, and w is the position just after the move. The move is made in accordance with the rules. The position w may have as many chessmen as u , or fewer in the case when some chessmen have been taken. So, a chessman's move that is followed by the taking of an opposite colour chessman is recognized as one move.

The first simplification we will is to reject the possibility of both big and small castling. Accommodating castling is something hampered by certain conditions which will not be analyzed here. The second simplification is to omit the replacement of pawns with chessmen on the change line. (A pawn, having moved across the whole board to the change line must be exchanged for the following chessmen: the queen, a castle, a bishop or a knight from the set of spare chessmen.)

The transition $u R w$ can be made by the *White* or *Black* player. In a fixed position u , $\{w \in W : u R w\}$ is then the set of all possible configurations on the chessboard which one arrives at by any side's move undertaken in the state u . The relation R is the join of two relations: R^{Black} and R^{White} , where $R^{White}(u, w)$ takes place if and only if in the position u the *White* player moves, according to the regulations, a white piece which is in the configuration u , and w is the position just following the move. The relation R^{Black} is defined analogously. Thus, $R = R^{White} \cup R^{Black}$. For a given configuration u , the set of all pairs (u, w) such that $R^{White}(u, w)$ takes place ($R^{Black}(u, w)$ takes place, respectively) can be called the set of all moves in the state u admissible for the *White* player (admissible for the *Black* player, respectively).

We adopt here a restrictive definition of the direct transition relation between configurations: in checking situations only the transitions resulting in releasing the opponent's king (if such exist) are admissible. Checking puts the king under the threat of death. When the king is threatened, the danger must be averted by the king's being moved to a square not in any line of attack of any of the opponent's pieces. In other words, in any configuration u , in which the king of a given colour is checked only the transitions from u to the states w , in which the king is released from check are admissible. So, if u is a configuration where, for example, the white king is checked, then $R^{White}(u, w)$ takes place if and only if there exists a white piece's move from u to the position w and the white king is not checked in the position w , i.e., it is not in a position where it could be taken by a black piece. The transition $R^{Black}(u, w)$ is similarly defined in the position u , where the black king is checked. Thus, if one of the kings is checked in the state u , then $u R w$ must be a move that releases the king from check.

There are various ways of selecting atomic actions in the game of chess. One option is to assign to each piece a certain atomic action. Thus, there are as many atomic actions as there are pieces. Each player can thus perform 16 actions with particular pieces. The above division of atomic actions is based on the tacit assumption that pieces retain their individuality in the course of the game. We will be more parsimonious and distinguish only six atomic actions to be performed by each player. Consequently, the atomic actions we shall distinguish refer to the *types* of pieces, and not to individual pieces. According to **AN**, each type of piece (other than pawns) is identified by an uppercase letter. English-speaking players use **K** for king, **Q** for queen, **R** for rook, **B** for bishop, and **N** for knight (since **K** is already used). Pawns are not indicated by a letter, but rather by the absence of any letter. This is due to the fact that it is not necessary to distinguish between pawns for moves, since only one pawn can move to a given square. (Pawn captures are an exception and indicated differently.)

Here is the list of atomic actions performed by the *White* player:

$$K^{White}, Q^{White}, R^{White}, B^{White}, N^{White}, P^{White}.$$

K^{White} is the action any performance of which is a single move of the white knight, Q^{White} are the white queen's moves, R^{White} are the white rooks' moves, B^{White} the bishops' moves, N^{White} the knights' moves, and P^{White} the action any performance of which is a move of an *arbitrary* white pawn on the chessboard.

A similar division of pieces and atomic actions is also adopted for the *Black* player:

$$K^{Black}, Q^{Black}, R^{Black}, B^{Black}, N^{Black}, P^{Black}.$$

A move with a piece includes taking the opponent's piece when performing this move.

Actions are conceived of extensionally. A given atomic action is identified with the set of its possible performances. Thus, if $A \in \{K^{White}, Q^{White}, R^{White}, B^{White}, N^{White}, P^{White}\}$, e.g. $A = K^{White}$, and $u, w \in W$, then $A(u, w)$ takes place if and only if in the position u the *White* player moves the knight (according to the movement regulations for the knight), and w is the chessboard position just after the move. (According to the algebraic notation, each move of a piece is indicated by the piece's uppercase letter, plus the coordinate of the destination square. For example, **Be5** (move a bishop to **e5**), **c5** (move a pawn to **c5**—no letter in the case of pawn moves, remember).

The system

$$(W, R, \{K^{White}, Q^{White}, R^{White}, B^{White}, N^{White}, P^{White}\} \cup \{K^{Black}, Q^{Black}, R^{Black}, B^{Black}, N^{Black}, P^{Black}\}), \quad (2.1.1)$$

where $R = R^{White} \cup R^{Black}$ is an elementary action system operated by two agents *White* and *Black*. The system is not normal. The reason is in the fact that in the positions u in which the king of a given colour is checked, e.g. the white one, the set of all possible performances of the actions $K^{White}, Q^{White}, R^{White}, B^{White}, N^{White}, P^{White}$ in the state u is, in general, larger than the set of admissible transitions from the state u to others, in which the king is no longer under threat. In other words, R^{White} is a proper subset of the union the relations $K^{White} \cup Q^{White} \cup R^{White} \cup B^{White} \cup N^{White} \cup P^{White}$. Analogously, R^{Black} is a proper subset of $K^{Black} \cup Q^{Black} \cup R^{Black} \cup B^{Black} \cup N^{Black} \cup P^{Black}$. Thus, the actions from the above list are not totally performable in some states.

The system (2.1.1) is complete. Although certain configurations u occur in none of the chess games, each direct transition $u R w$ is made by means of a certain atomic action from the above list, i.e., there exists an A such that $u A, R w$. From this remark the completeness of the system follows.

u_0 is the initial configuration accepted in any chess game, i.e., the arrangement of pieces on the chessboard at the start of each game. The proposition $\Phi := \{u_0\}$ is thus the initial condition of each game.

In the game of chess, a few types of final conditions are distinguished. The most two important are where the black king is checkmated and where the white king is checkmated.

Let Ψ_0^{Black} be the set of all possible positions u on the chessboard in which the black king is checked by some white piece. The proposition Ψ_0^{Black} expresses the fact that the black king is checked (but not the fact that the next move must be performed by the *Black* player). Analogously, the set Ψ_0^{White} is defined as the set of positions u in which the white king is checked.

The king is checkmated when he is in a configuration in which he is checked *and* when there is no way of escape nor cannot he in any other way be protected against the threat of being taken. Checkmate results in the end of the game.

So, the checkmate of the black king is the set Ψ^{Black} of all positions u in which the black king is checkmated and none of the actions K^{Black} , Q^{Black} , R^{Black} , B^{Black} , N^{Black} , P^{Black} can be undertaken in order to protect the black king. (As noted above, we are omitting here the possibility of castling as a way to escape checkmate.) This fact can be simply expressed in terms of the relation R^{Black} as follows:

$$\Psi^{Black} := \{u \in \Psi_0^{Black} : \delta_R^{Black}(u) = \emptyset\}.$$

(δ_R^{Black} is the graph of the relation R^{Black} .)

In the analogous way we define the proposition Ψ^{White} which expresses the checkmate of the white king:

$$\Psi^{White} := \{u \in \Psi_0^{White} : \delta_R^{White}(u) = \emptyset\}.$$

Thus, in the game of chess, we distinguish two tasks: (Φ_0, Ψ^{Black}) and (Φ_0, Ψ^{White}) . The task (Φ_0, Ψ^{Black}) is taken by the *White* player. He aims to checkmate the black king. The states belonging to the proposition Ψ^{Black} define then the goal the *White* player wants to achieve. Analogously, (Φ_0, Ψ^{White}) is the task for the *Black* player. His goal is to checkmate the white king.

A stalemate is a position in which one of the players, whose turn comes, cannot move either the king or any other chessman; at the same time the king is not checked. When there is a stalemate on the chessboard, the game ends in a draw.

Let Λ^{Black} denote the set of stalemate positions of the black king, i.e.,

$$\Lambda^{Black} := \{u \in W : u \notin \Psi_0^{Black} \text{ and } \delta_R^{Black}(u) = \emptyset\}.$$

Similarly, Λ^{White} denotes the set of stalemate positions of the white king:

$$\Lambda^{White} := \{u \in W : u \notin \Psi_0^{White} \text{ and } \delta_R^{White}(u) = \emptyset\}.$$

If during the game the position $u \in \Lambda^{Black}$ is reached and the *Black* player is to make a move in this position, the game ends in a draw; similarly for when $u \in \Lambda^{White}$ and the *White* player is to move.

The stalemate positions do not exhaust the set of configurations which give rise to the game ending. A game ends in a draw when both players know that there exist possibilities for continuing the game indefinitely which do not lead to positions of $\Psi^{Black} \cup \Psi^{White}$. In practice, a purely pragmatic criterion is applied which says that the game is drawn after the same moves have been repeated in sequence three times; the game is also limited by the rule that in 50 moves a pawn must have been moved.

In spite of simplifications made in the chess tournament, we have not explored all the rules of the game. We know what actions the players perform but we have not added that they can only do them alternately. The same player is not allowed to make successively two or more moves (except the case of castling). Moreover, the game is started only by the *White* player in the strictly defined initial position. These facts indicate that the description of the game of chess, expressed by means of formula (2.1.1), is oversimplified and it does not adequately reflect the real course of the game. We shall return to this issue in the next paragraph.

The above situational description of the game of chess does not have, obviously, any greater value as regards its usability. For obvious reasons neither the best of players nor any computer is able to take on a victor's strategy through searching all possible configurations on the chessboard that are continuations of the given situation. The phenomenon of combinatorial explosion blocks further calculations (we get very large numbers, even for a small number of steps). At the time of writing, the best computer can search at the most three moves ahead in each successive configuration (a move is one made by *White* and the successive move by *Black*). It is in this fact, as well as the rate at which computers operate, that the advantage the computer has over the human manifests itself. One obvious piece of advice and the most reasonable suggestion offered to the human player is to select a strategy referring based on his experience as a chess-player and the knowledge gathered over the centuries, where this includes a description of the finite set (larger but not too large—within the limits of what a human brain cell remember) of particularly significant games.

2.2 Actions and Situations

The above examples reveal the significant role of situational contexts in which action systems function. These contexts that influence the course of an action are not determined just by the current state of the action system but also by other factors. It is often difficult or even not feasible to specify them all. Moving a black piece to an empty square may be allowed by the relation R^{Black} if only the current arrangement of chessmen is taken into account; the move however will not be permitted when the previous move was also made by the *Black* player for two successive moves by the same player does not comply with the rules of chess. A similar remark applies to tic-tac-toe.

The above examples also show that the performability of an action in a particular state u of the system may depend on the previously undertaken actions, their succession, and so on. In the light of the above remarks a distinction should be made between the notion of a state of the system and the situation of the system. In this chapter this distinction is explored from the perspective of action systems.

The set W of possible states, the relation R of a direct transition between states and the family \mathcal{A} of atomic actions, fully determine the possible transformations of the atomic system, i.e., the transformations (carried out by the agents) that move the system from one state to others. The current situation of the system is in general shaped by a greater number of factors. A given situation is determined by the data from the surroundings of the system. The current state of the system is one of the elements constituting the situation but, of course, not the most important. The agents are involved in a network of mutual relations. The situation in which the agents act may depend on certain principles of cooperation (or hostility) as well. Acceptance of certain norms of action is an example of such interdependencies. Thus, each state of the system is immersed in certain wider situational contexts. The context is specified, among others, by factors, not directly bound up with the state of the system as: the moment when a particular action is undertaken, the place where the system or its part is located at the moment the action is started or completed, the previously performed action (or actions) and their agents, the strategies available to the agents, etc. Not all of the mentioned factors are needed to make up a given situation—it depends on the depth of the system description and the principles of its functioning. These elements constitute the situational setting of a given state.

In a game of chess the relation R of direct transition between configurations on the chessboard defined in Sect. 2.1 as the union $R = R^{White} \cup R^{Black}$ is insensitive to the situational context the players are involved in. The atomic actions $K^{White}, Q^{White}, R^{White}, B^{White}, N^{White}, P^{White}$ or $K^{Black}, Q^{Black}, R^{Black}, B^{Black}, N^{Black}, P^{Black}$ transform some configurations into others. The above purely extensional, ‘input—output’ description of atomic actions abstracts from some situational factors which influence the course of the game. The performability of an atomic action depends here exclusively on the state of the system in which the action is undertaken (Definition 1.4.1). Thus performability is viewed here as a context-free property, devoid of many situational aspects which may be relevant in the description of an action.

We shall outline here a situational concept of action performability, according to which the fact that an action is performable in a given state of the system depends not only on the state and the relation R of direct transition between states but also on certain external factors—the situational context the system is set in.

Performing an action changes the state of the system and at the same time it creates a new situation. A move made by a player in a game of chess changes the arrangement of chessmen on the board. It also changes the player’s situation: the next move will be made by his opponent unless the game is finished. It does not mean that the notion of an action should be revised—as before we identify (atomic) actions with binary relations on the set W of possible states of the system. We shall, however, extend the notion of an elementary action system by enlarging it with new

components: the set S of possible situations, the relation Tr of a direct transition between possible situations, and a map f which to every possible situation s assigns a state $f(s) \in W$. The state $f(s)$ is a part of the situation s —if s occurs then $f(s)$ is the state of the system corresponding to s . These components enable one to articulate a new, ‘situational’ definition of the performability of an action. Thus, we will speak of action performability in a given state of the system with respect to a definite situational context, or shortly, the *performability of an action in a given situation*.

Situations are not investigated here in depth as a separate category. That is, we shall not investigate and develop an ontology of situations but rather limit ourselves to some general comments. The focus is rather on illustrating the role of situations in the action theory in various contexts than outlining a general account of structured situations. The notion of a situation we shall use here is built out of elements taken from automata theory, theory of algorithms, games and even physics and does not fully agree with the notions that occur in the literature.

We shall present here a simplified, ‘labeled’ theory of situations. This theory is convergent with the early formal approaches to logical pragmatics (see e.g. Montague 1970; Scott 1970). On the other hand, this concept is based on some ideas which directly stem from the theory of algorithms.

The ontology of actions adopted in this book shows similarities with the situation calculus applied in logic programming. The situation calculus is a formalism designed for representing and reasoning about dynamic entities (McCarthy 1963; Reiter 1991). It is based on three key ingredients: the actions that can be performed in the world, the situations, and the fluents that describe the state of the world. In Reiter’s approach situations are finite sequences of actions. Here, situations are isolated and form a separate ontological category.

Let S be a set, whose elements will be called *possible* (or *conceivable*) situations.² Each situation $s \in S$ is determined by a system of factors. Their specification depends on the ways of organization and functioning of the action system. A possible situation can include the following components: the state of the system, the location of the system (or its distinguished parts), the agent of each atomic action, the action currently performed on the system, the previously performed action and its agent, and so on. We can roughly characterize any situation s as a sequence of entities

$$s = (w, t, x, \dots), \quad (2.2.1)$$

where $w \in W$ is a possible state of the system, t —time, x —location of the system, etc. To each possible situation (2.2.1) a unique state w is assigned—the first component of the above sequence—which is called the *state of the system in the situation s* . The values of all the other parameters (and strictly speaking the names of the values of

²This sentence is a convenient conceptual metaphor. Situations are constituents of the real world. In this chapter situations are viewed as *mathematical* (or rather *set-theoretical*) representations of these constituents. We therefore speak of sets of conceivable situations.

Of course, any representation of the world of situations and actions does not involve mentioning all of them individually but it rather classifies them as uniform sorts or types. But token situations or actions can be individuated, such as the stabbing of Julius Caesar.

these parameters) defining a given situation form a subsequence of s which is called the *label* of the situation s . In the case of situation (2.2.1), its label is equal to the sequence (t, x, \dots) . Thus, each possible situation s can be represented by an ordered pair

$$s = (w, a),$$

where $w \in W$ is a state of the system and a is a label.

The labels are assumed to form a set which is denoted by V . (It is not assumed that the set V is finite.) Thus, the set S of possible situations is equal to the Cartesian product $W \times V$, i.e.,

$$S := W \times V.$$

Apart from the set S there is a transition relation Tr between situations. If $Tr(s_1, s_2)$ holds, then s_2 is the situation immediately occurring after the situation s_1 ; we also say that the situation s_1 *directly turns* into the situation s_2 . The fact that $Tr(s_1, s_2)$ holds does not prejudice the occurrence of the situation s_1 .

The above remarks enable us to articulate the definition of a situational action system.

Definition 2.2.1 A *situational action system* is a six-tuple

$$\mathbf{M}^s := (W, R, \mathcal{A}, S, Tr, f),$$

where

- (1) the reduct $\mathbf{M} := (W, R, \mathcal{A})$ is an elementary action system in the sense of Definition 1.2.1;
- (2) S is a non-empty set called the set of *possible situations* the action system \mathbf{M} is set in. The set S is also called the *situational envelope* of the action system \mathbf{M} ;
- (3) Tr is a binary relation on S , called the *direct transition relation* between possible situations;
- (4) $f : S \rightarrow W$ is a mapping which to each situation $s \in S$ assigns a state $f(s) \in W$ of the action system \mathbf{M} . $f(s)$ is called the *state of the action system \mathbf{M} corresponding to the situation s* , or simply, the *state of the system in the situation s* . It is therefore unique, for each situation s ;
- (5) the relation R of direct transition between states of the action system $\mathbf{M} = (W, R, \mathcal{A})$ is compatible with Tr , i.e., for every pair $s_1, s_2 \in S$ of situations, if $s_1 Tr s_2$ then $f(s_1) R f(s_2)$. \square

The changes in the situational envelope take place according to the relation Tr . Condition (5) says that the evolution of situations in the envelope is compatible with transformations between the states of the system \mathbf{M} , defined by the relation R . (In the ‘labeled’ setting of situations, that is, when $S = W \times V$, the function f is defined as the projection of $W \times V$ onto W , i.e., $f(s) = w$, for any situation $s = (w, a) \in S$.)

It follows from (5) that the set

$$\{(u, w) \in W \times W : (\exists s, t \in S)(s Tr t \ \& \ f(s) = u \ \& \ f(t) = w)\}$$

is contained in the relation R . The two sets need not be equal. In other words, it is not postulated that for any $s_1 \in S$, $w \in W$ the condition $f(s_1) R w$ implies that $s_1 Tr s_2$ for some $s_2 \in S$ such that $w = f(s_2)$. (For example, this implication fails to hold in the situational model of chess playing defined below.) This shows that, in general, the relation $R \subseteq W \times W$ cannot be eliminated from the description of situational action systems and replaced by the relation Tr .

In the simplest case, possible situations of S are identified with states of the action system $\mathbf{M} = (W, R, \mathcal{A})$, that is, $S = W$ (the label of each situation is the empty sequence), f is the identity map, and $Tr = R$, and the situational action system \mathbf{M}^s reduces to the elementary system \mathbf{M} .

Definition 2.2.1 is illustrated by means of so-called *iterative* action systems; the latter form a subclass of situational systems. Iterative action systems function according to simple algorithms that define the order in which particular atomic actions are performed. A general scheme which defines these systems can be briefly described in the following way. Let $\mathbf{M} = (W, R, \mathcal{A})$ be an elementary action system. It is assumed that S , the totality of possible situations, is equal to the Cartesian product $W \times V$, where V is a fixed set of labels. The function f that assigns to each situation s the corresponding state of the system \mathbf{M} is the projection from $W \times V$ onto W . In order to define the relation Tr of a direct transition between situations we proceed as follows. An action performed in a definite situation changes the state of the system \mathbf{M} and also results in a change of the situational context. The triples

$$(a, A, b), \quad (2.2.2)$$

where a and b are labels in V and $A \in \mathcal{A}$ is an atomic action, are called *labeled actions*. The action (2.2.2) has the following interpretation: the atomic action A performed in a situation with label a changes the state of the system \mathbf{M} in such a way that the situation just after performing this action is labeled by b . Thus, (a, A, b) leads from situations labeled by a to ones with label b while the states of \mathbf{M} transform themselves according to the action A and the relation R .

A non-empty set \mathbf{LA} of labeled actions is singled out. \mathbf{LA} need not contain all the triples (2.2.2) nor need be a finite set. We say that a labeled action $(a, A, b) \in \mathbf{LA}$ transforms a situation s_1 into s_2 if and only if $s_1 = (u, a)$, $s_2 = (w, b)$ and, moreover, it is the case that $u A, R w$.

The set \mathbf{LA} makes it possible to define the transition relation $Tr(\mathbf{LA})$ between situations. Let $s_1 = (u, a)$, $s_2 = (w, b) \in S$. Then we put:

$$(s_1, s_2) \in Tr(\mathbf{LA}) \text{ if and only if } (\exists A \in \mathcal{A})((a, A, b) \in \mathbf{LA} \ \& \ u A, R w). \quad (2.2.3)$$

It follows from (2.2.3) that $(s_1, s_2) \in Tr(\mathbf{LA})$ implies that $f(s_1) R f(s_2)$, i.e., R is compatible with Tr and that the conditions (1)–(5) of Definition 2.2.1 are met. The six-tuple

$$(W, R, \mathcal{A}, S, Tr(\mathbf{LA}), f)$$

is a situational action system.

A finite *run* of situations is as any finite sequence

$$(s_0, s_1, \dots, s_n)$$

of situations such that $(s_i, s_{i+1}) \in Tr(\mathbf{LA})$ for every i , $0 \leq i \leq n - 1$. The situation s_0 is called the *beginning* of the run.

We may also consider infinite (or *divergent*) runs of situations as well. These are infinite sequences of situations

$$(s_0, s_1, \dots, s_n, \dots)$$

such that $(s_i, s_{i+1}) \in Tr(\mathbf{LA})$, for all i . The problem of divergent runs, interesting from the viewpoint of computability theory, is not discussed in this book.

In tic-tac-toe two labels are distinguished: X and O. Thus, $V := \{X, O\}$. A possible situation is a pair $s = (w, a)$, where $a \in V$ and $w \in W$ is a configuration on the board (each configuration being identified with an appropriate 3×3 matrix). The pair $s = (w, X)$ is read: w is the current configuration on board and **X** is to make a move. One interprets the pair (w, O) in a similar way.

The set of labeled actions has two elements: (X, A_X, O) and (O, A_O, X) .³ The action (X, A_X, O) transforms a situation of the form $s_1 = (u, X)$ into a situation $s_2 = (w, O)$. The transformation from s_1 to s_2 is accomplished by performing the action A_X by **X**. This action moves the system from the state u to w . The labeled action (O, A_O, X) is read in a similar way.

The second example is similarly reconstructed as a situational action system. In a game of chess we distinguish two labels, BLACK and WHITE, that is $V := \{\text{BLACK}, \text{WHITE}\}$. A possible chess situation is thus any pair of the form $s = (w, a)$, where $a \in V$ and $w \in W$ is a configuration on the chessboard. The pair $s = (w, \text{WHITE})$ is interpreted as follows: w is the current configuration on the chessboard and *White* is to make a move. The pair (w, BLACK) is read analogously.

The set \mathbf{LA} of labeled atomic actions consists of the following triples:

$$(\text{WHITE}, A^{\text{White}}, \text{BLACK}) \quad (2.2.4)$$

where $A^{\text{White}} \in \{\mathbf{K}^{\text{White}}, \mathbf{Q}^{\text{White}}, \mathbf{R}^{\text{White}}, \mathbf{B}^{\text{White}}, \mathbf{N}^{\text{White}}, \mathbf{P}^{\text{White}}\}$ and

$$(\text{BLACK}, A^{\text{Black}}, \text{WHITE}) \quad (2.2.5)$$

where $A^{\text{Black}} \in \{\mathbf{K}^{\text{Black}}, \mathbf{Q}^{\text{Black}}, \mathbf{R}^{\text{Black}}, \mathbf{B}^{\text{Black}}, \mathbf{N}^{\text{Black}}, \mathbf{P}^{\text{Black}}\}$.

The labeled action (2.2.4) thus transforms situations of the type $s_1 = (u, \text{WHITE})$ into situations $s_2 = (w, \text{BLACK})$. The transformation is accomplished by

³Indeed, instead of the triples (X, A_X, O) and (O, A_O, X) it suffices to take the pairs (A_X, O) and (A_O, X) , since the actions A_X and A_O are already labeled by X and O, respectively (that is, they possess their own names). Defining labeled actions as triples is justified by the fact that elements of the set of atomic actions in elementary action systems are not basically linguistically distinguished through investing them with special names.

performing in the state u an atomic action by *White* so that w is the configuration of the chessboard just after the move. The labeled action (2.2.5) is interpreted in a similar way.

Having defined the set \mathbf{LA} , we then define the relation $Tr(\mathbf{LA})$ according to formula (2.2.3). The function $f : S \rightarrow W$ is defined, as expected, as the projection of S onto $W : f((w, a)) := w$, for all $(w, a) \in S$.

There is only one initial situation $s_0 := (u_0, \text{WHITE})$, where u_0 is the configuration at the outset of the game. Thus, a *game of chess* is a finite sequence of situations (s_0, s_1, \dots, s_n) , $n \geq 0$, such that $s_0 = (u_0, \text{WHITE})$ is the initial situation and $(s_i, s_{i+1}) \in Tr(\mathbf{LA})$, for all i , $0 \leq i \leq n-1$. It follows from the definition of $Tr(\mathbf{LA})$ that successive moves are alternately performed by the players *Black* and *White*. The transition from s_0 to s_1 is accomplished by *White* who starts the game. In turn, the transition from s_1 to s_2 is made by *Black*, and so on. Thus, a game of chess can be represented as an iterative action system, taking into account the simplifications we have made.

A few words about terminal situations. In the light of the above definitions, a game of chess may be *any* long sequence (s_0, \dots, s_n) of a situation starting with the initial situation. A game thus defined need not respect the rules of chess. First, there exist time limitations that do not permit game to go on for too long. (They can be inserted into the scheme of situation presented here by adjoining an additional time parameter.) However, there is a more important reason. A game of chess *has* to finish in the case of checkmating the adversary. The above definition of a game does not take this factor into consideration. To resolve the matter we will define a certain subclass of the class of possible situations. First, we will extend the set of labels with a new label ω , which will be called the *terminal* label. Thus

$$V' := \{\text{BLACK}, \text{WHITE}, \omega\}.$$

We also say that

$$S' := W \times V'.$$

A possible *terminal* situation is any situation of the form (w, ω) , where w is a configuration belonging to the set $\Psi^{\text{Black}} \cup \Psi^{\text{White}}$ i.e., w is a configuration in which either the black or the white king is checkmated.

The set \mathbf{LA} of labeled atomic actions defined by means of the formulas (2.2.4) and (2.2.5) is also extended to the set \mathbf{LA}' by augmenting it with the following actions:

$$(\text{BLACK}, A^{\text{White}}, \omega) \tag{2.2.6}$$

where A^{White} is an atomic action assigned to *White*, and

$$(\text{WHITE}, A^{\text{Black}}, \omega) \tag{2.2.7}$$

where A^{Black} is an atomic action of *Black*.

Note One should carefully distinguish between linguistic interpretations of a game and situational runs of a game. Here ω , BLACK, WHITE are linguistic entities while A^{White} and A^{Black} are binary relation on the set of states. \square

Let $Tr(\mathbf{LA}')$ be the transition relation between situations of S' determined by the labeled actions of \mathbf{LA}' . As is easy to check, the terminal situations defined as above are indeed terminal—if s is terminal then there exists no situation s' such that $Tr(\mathbf{LA}')(s, s')$. The notion of a game of chess is defined similarly as above; that is, as a finite sequence of situations (s_0, s_1, \dots, s_n) , $n \geq 0$, such that $s_0 = (u_0, \text{WHITE})$ is the initial situation and $(s_i, s_{i+1}) \in Tr(\mathbf{LA}')$, for all i . A game is concluded if the situation s_n is terminal. (The definition of a concluded game does not exhaust the set of all situations in which the game is actually concluded. A game can be concluded through being a draw. To take these situations into account, we would have to further extend the notion of a situation and the transition relation between situations.)

The distinction between the meanings of ‘state’ and ‘situation’ is not absolute. When speaking about elementary action systems, we do not always have in mind sharply-distinguished material objects subject to the forces exerted by the agents. The definition of an elementary system distinguishes only certain states of affairs and relations between them; in particular the relation of a direct transition. The selection of one or other set of states and binary relations representing atomic actions greatly depends on the ‘world perspective’ and on a definite perception of the analyzed actions in particular. We may figuratively say that elementary action systems define *what* actions are performed while situational action systems also take into account *how* they are performed. The borderline between the two concepts is fluid. For example, one may modify the scheme of chess presented above so as to include the players in the notion of a state (i.e. a configuration of pieces). Thus, e.g., the fact that *Black* is to make a move would be a component of the current state of the game. Such a step is, of course, possible; it would however complicate the description of the game.

Suppose $\mathbf{M} = (W, R, \mathcal{A})$ is a quite arbitrary elementary action system. If one wants to restrict uniformly all actions in the system to sequences of states of length at most n , it suffices to introduce n labels, being e.g. the consecutive natural numbers (or numerals) $0, 1, \dots, n$, and to define the set of possible situations as the product $S := W \times \{0, 1, \dots, n\}$. The function f assigning to each situation s its unique state is the projection onto the first axis, i.e., $f(s) := w$ for any situation $s = (w, k)$.

A given state may therefore receive $n + 1$ different labels. The (direct) transition relation Tr between situations is defined as follows: for $s = (w, k)$ and $s' = (w', k')$,

$$s Tr s' \Leftrightarrow_{df} w R w' \wedge w \neq w' \wedge k' = k + 1.$$

Each situational transition changes states and increases the label from k to $k + 1$. (The second conjunct excludes reflexive points of R but does not exclude loops of states of longer length from situational transitions. If we want to exclude loops $u R u_1 R \dots u_l R u$ altogether as ‘futile’ computations, the problem is much more complicated and not analysed here.)

$M^s = (W, R, \mathcal{A}, Tr, f)$ is a well-defined situational action system in which $M = (W, R, \mathcal{A})$ is contained.

The situations of the form $s = (w, n)$ are terminal, where w is an arbitrary state. This means that for $s = (w, n)$ there is no situation s' such that $s Tr s'$. Therefore the system halts at $s = (w, n)$ and the work of the action system M ceases as being subject to the organization of M^s . In turn, situations of the form $(w, 0)$ may be treated as initial ones.

Since loops in runs of situations are not excluded, it may happen that there is a sequence of transitions between situations of length ≥ 3 , say $s Tr s_1 Tr \dots Tr s_l Tr s$. In this case the system starts at s and finishes at s (in the same state). But of course we may declare at the outset that R does not admit loops.

The width of the situational envelope of an elementary action system depends on how the system functions. Let us take a look at Example 1.3.3. W_T is here the set of all proofs carried out from T with the help of the rules from a set Θ . Let $w = (\phi_0, \dots, \phi_{n-1}, \phi_n)$ be a fixed proof in W_T . In the simplest case, the situational context of the proof w is constituted by the way the formula ϕ_n , i.e., the last element of the proof w , is adjoined to the shorter proof $u = (\phi_0, \phi_1, \dots, \phi_{n-1})$. The proof w is a result of applying a definite rule $r \in \Theta$ to the some ‘prior’ formulas $\{\phi_j : j \in J\}$, where J is a subset of $\{0, \dots, n-1\}$; that is, to some formulas occurring in u . The situational context of w is therefore represented by means of the triple (u, r, J) , which encodes the above fact. Denoting the pair (r, J) by a , we see that the above situation can be identified with the pair $s = (u, a)$. A deeper description of possible situations would take into account not only the way the proposition ϕ_n was affixed to u forming the proof w , but also, for example, the ways all or some of the sub-proofs of u were formed. Along with changing the width and the depth of the description of possible situations, the description of the relation Tr of a direct transition between possible situations would be modified too.

The second remark concerns current situations of action systems. An action system always finds itself in a definite situation—the *current situation* of the system. The state corresponding to this situation (the state in which the system is) is called the *current state* of the system. It is not possible to single out the separate category of current situations by means of linguistic procedures though it is possible to provide an exhaustive list of attributes determining these situations. Hybrid logic introduces nominals which are objects that signify only one situation, namely the one which actually happens. For example, one can always describe the configuration of pieces which are *now* placed on the chessboard but no linguistic operation is able to fully render the meaning of the word ‘now’. The terms ‘the current situation’ and ‘the current state’ are demonstratives—their proper understanding always requires knowledge of extralinguistic factors.

The third remark concerns the relationship between situations and (possibly infinite) runs of situations. Suppose

$$M^s = (W, R, \mathcal{A}, S, Tr, f)$$

is a situational system. A *possible* (or *hypothetical*) *run* of situations (in M^S) is any function mapping an interval of integers into S . There are thus *a priori* four possible *types* of runs. A run is of type $(-\infty, +\infty)$ if it is indexed by the set of all integers, i.e., it is represented as an infinite sequence without end points

$$(\dots, s_{-m}, s_{-m+1}, \dots, s_0, \dots, s_n, s_{n+1}, \dots), \quad (2.2.8)$$

where $s_k Tr s_{k+1}$, for every integer k .

A run is of type $(-\infty, 0)$ if it is indexed by the non-positive integers, i.e., it is of the form

$$(\dots, s_{-m}, s_{-m+1}, \dots, s_0), \quad (2.2.9)$$

where $s_k Tr s_{k+1}$ for every negative integer k and there does not exist a situation $s \in S$ such that $s_0 Tr s$.

A run is of type $(0, +\infty)$ if it is of the form

$$(s_0, \dots, s_n, s_{n+1}, \dots), \quad (2.2.10)$$

where $s_k Tr s_{k+1}$ for every natural number $k \in \omega$ and there does not exist a situation $s \in S$ such that $s Tr s_0$.

Finally, a run is *finite* (and terminated in both directions) if it is of the form (s_0, \dots, s_n) for some natural number n and there do not exist situations $s, s' \in S$ such that $s Tr s_0$ or $s_n Tr s'$.

Whenever we speak of a run of situations, we mean *any* run falling into one of the above four categories. If a situation s occurs in a run, then the situation s' in the run that directly follows s is called the *successor* of s ; the situation s is then called the *predecessor* of s' .

Proposition 2.2.2 *Let $M^S = (W, R, \mathcal{A}, S, Tr, f)$ be a situational action system. Let s, s' be two possible situations in S . Then $s Tr s'$ holds if and only if there exists a run (of one of the above types) such that s and s' occur in the run and s' is the successor of s .*

The proof is simple and is omitted. □

To each situational system M^S the class R of all possible runs of situations in the system is assigned. It may happen that R includes runs of all 4 types. This property gives rise to a certain classification of situational systems. For example, a system M^S would be of category 1 if R contained only finite runs. There are $2^4 - 1 (= 15)$ possible categories of situational action systems.

The relation Tr of direct transitions between situations can be unambiguously described in terms of runs of situations. More precisely, any situational action system can be equivalently characterized as a quintuple

$$(W, R, \mathcal{A}, S, f) \quad (2.2.11)$$

satisfying the earlier conditions imposed on (W, R, \mathcal{A}) and S, f , for which additionally the class \mathbf{R} of runs of situations from S is singled out. \mathbf{R} is postulated to be the class of all runs of situations admissible for the system (2.2.11). The relation Tr of direct transitions between situations can be then *defined* by the right-hand side of the statements of Proposition 2.2.2.

We conclude this section with remarks on the performability of actions in situational systems. Suppose we are given a situational action system \mathbf{M}^s in the sense of Definition 2.2.1. The fact that A is performable in a definite situation s is fully determined by the relations Tr and R . Accordingly:

If s is a situation and $A \in \mathcal{A}$ is an atomic action, then the act of performing the action A in this situation turns s into a situation s' such that $s Tr s'$ and $f(s) A f(s')$.

Immediately after performing the action A the system $\mathbf{M} = (W, R, \mathcal{A})$ is in the state $f(s')$. This is due to the fact that R is compatible with Tr , i.e., $s Tr s'$ implies that $f(s) R f(s')$.

The above remarks give rise to the following definition:

Definition 2.2.3 Let $\mathbf{M}^s = (W, R, \mathcal{A}, S, Tr, f)$ be a situational action system and let $s \in S$ and $A \in \mathcal{A}$.

- (i) The atomic action A is *performable in the situation s* if and only if there exists a situation $s' \in S$ such that $s Tr s'$ and $f(s) A f(s')$; otherwise A is *unperformable* in s .
- (ii) The action A is *totally performable in the situation s* if and only if it is performable in s and for every state $w \in W$, if $f(s) A w$, then $s Tr s'$ for some situation s' such that $w = f(s')$. □

The performability of an action A in a situation s is thus tantamount to the existence of the direct Tr -transition from s to another situation s' such that the pair $(f(s), f(s'))$ is a possible performance of A (i.e., A is accounted for the transition $s Tr s'$). The second conjunct of the definition of total performability of A in s states that *every* possible performance of A in the state $f(s)$ results in a new situation s' such that $s Tr s'$ (i.e., the relation Tr imposes no limitations on the possible performances of A in $f(s)$).

Similar to the case of elementary action systems, total performability always implies performability. The converse holds for deterministic actions, i.e. actions A which are partial functions on W .

The concept of the situational performability of an atomic action is extended onto non-empty compound actions. Let \mathbf{M}^s be a situational system and $\mathbf{A} \in C\mathcal{A}$ a compound action. \mathbf{A} is *performable in a situation s* if and only if there exists a non-empty string (s_0, \dots, s_n) of situations and a string of atomic actions $A_1 \dots A_n \in \mathcal{A}$ such that $s = s_0 Tr s_1 \dots s_{n-1} Tr s_n$ and $f(s_0) A_1 f(s_1) \dots f(s_{n-1}) A_n f(s_n)$ (i.e., the transition from s to s_n is effected by means of consecutive performances of the actions A_1, \dots, A_n).

The action \mathbf{A} is *totally performable in s* if and only if \mathbf{A} is performable in s and for every possible performance (u_0, \dots, u_n) of \mathbf{A} such that $u_0 = f(s)$ there

exist situations s_0, \dots, s_n with the following properties: $s_0 = s$, $u_i = f(s_i)$ for $i = 0, \dots, n$, and $u_i \text{Tr} u_{i+1}$ for all $i \leq n - 1$.

The second conjunct of the above definition states that for every performance (u_0, \dots, u_n) of \mathbf{A} starting with $u_0 = f(s)$ there exists a run of situations (s_0, \dots, s_n) with $s_0 = s$ such that u_0, \dots, u_n are the states corresponding to s_0, \dots, s_n .

The existence of a situational envelope of an elementary action system radically restricts the possibilities of performing compound actions. In any normal elementary action system \mathbf{M} , every compound action not containing the empty string ε is totally performable (Proposition 1.7.6). However, if the elementary system \mathbf{M} is a part of some situational action system $\mathbf{M}^s = (W, R, \mathcal{A}, S, \text{Tr}, f)$, that is, the reduct (W, R, \mathcal{A}) of \mathbf{M}^s coincides with \mathbf{M} , then the above result is no longer true (if performability is taken in the sense of \mathbf{M}^s).

2.2.1 An Example. Thomson Lamp

Intuitively, a supertask is an activity consisting of an infinite number of steps but taken, as a whole, during a finite time. Examples of modern supertasks resemble ancient paradoxes posed by Zeno of Elea (e.g. Achilles and the tortoise). Supertasks may be identified with *infinite* strings of atomic actions performed in a finite time period.

A Thomson lamp is a device consisting of a lamp and a switch set on an electrical circuit. If the switch is on, then the lamp is lit, and if the switch is off, then the lamp is off. A Thomson lamp is therefore modelled as a simple elementary action system \mathbf{M} with two states and two atomic action A : ‘Switching on the lamp’ and B : ‘Switching off the lamp’. The picture gets more complicated if \mathbf{M} is contained in a situational envelope involving only one additional situational parameter—time. Suppose that:

1. At time $t = 0$ the switch is on.
2. At time $t = 1/2$ the switch is off.
3. At time $t = 3/4$ the switch is on.
4. At time $t = 7/8$ the switch is off.
5. At time $t = 15/16$ the switch is on, etc.

What is the state of the lamp at time $t = 1$? Is it lit or not? We see that the above infinite sequence of alternate atomic actions gives rise to a formulation of non-trivial questions, depending on the selection of the situational envelope.⁴ Solving them requires introducing genuinely infinitistic components into the picture of action theory presented here. These components are introduced in different ways. One option is to endow the *set of states* with a topology or with some order-continuous properties. This option requires infinite, ordered sets of states exhibiting various forms of order-completeness. The other option is to embed the atomic system in a

⁴The above example is taken from Jerzy Pogonowski’s essay *Entertaining Math Puzzles* which can be found at www.glii.uni.opole.pl.

situational envelope endowed with various continuity properties. (This is the case with a Thomson lamp.) A mixture of these two approaches is also conceivable. In Chap. 3 the first option is elaborated in the context of an ordered action system.

2.3 Iterative Algorithms

Let $M = (W, R, \mathcal{A})$ be an elementary action system. An *iterative algorithm* for M is a quintuple

$$D := (W, V, \alpha, \omega, \mathbf{LA}), \quad (2.3.1)$$

where W is the set of states of M , V is a finite set called the set of *labels* of the algorithm, α and ω are designated elements of V , called the *initial* and the *terminal* (or *end*) label of the algorithm, respectively, and \mathbf{LA} is a finite subset of the Cartesian product

$$V \setminus \{\omega\} \times \mathcal{A} \times V \setminus \{\alpha\}.$$

The members of \mathbf{LA} are called *labeled atomic actions* of the algorithm D . Thus, the labeled actions are triples (a, A, b) , where a and b are labels, called respectively the *input* and the *output* label, and A is an atomic action. Since \mathbf{LA} involves only finitely many atomic actions of \mathcal{A} , the members of the set

$$\mathcal{A}_D := \{A \in \mathcal{A} : (\exists a, b \in V) (a, A, b) \in \mathbf{LA}\}$$

are called the *atomic actions of the algorithm* D . \mathbf{LA} may be empty.

The elements of the set $S_D := W \times V$ are called the *possible situations* of the algorithm. If $s = (w, a) \in S_D$, then w is the *state corresponding to* s , and a is the *label of* s . The situations with label α , i.e., the members of $W \times \{\alpha\}$ are (possible) *initial situations* of the algorithm while the members of $W \times \{\omega\}$, i.e., the situations labeled by ω are (possible) *terminal situations* of the algorithm D .

The relation Tr_D of *direct transition* in the algorithm is defined as follows: if $s = (u, a)$, $t = (w, b)$ are possible situations of D , then

$$Tr_D(s, t) \quad \text{if and only if} \quad (\exists A) (a, A, b) \in \mathbf{LA} \ \& \ uA, Rw.$$

Thus, if s is an initial situation, then $Tr(s', s)$ for no situation s' of D . Similarly, if s is terminal, then there does not exist a situation $s' \in S$ such that $Tr(s, s')$.

The relation Tr_D^* , the transitive and reflexive closure of Tr_D , is called the *transition relation* in D . Thus,

$$(s, t) \in Tr_D^* \quad \text{if and only if} \quad (\exists n \geq 0) (s, t) \in (Tr_D)^n.$$

(Thus, in particular, $(Tr_D)^0 = \{(s, s) : s \in S\}$.) *Finite runs* of the algorithm D are sequences of situations

$$(s_0, s_1, \dots, s_n) \quad (2.3.2)$$

such that s_0 is an initial situation of D and $(s_i, s_{i+1}) \in Tr_D$ for all $i, 0 \leq i \leq n-1$. The situations s_0 and s_n are called the *beginning* and the *end* of the run (2.3.2). The run (2.3.2) is *terminated* in D if and only if s_n is a terminal situation of D .

It may happen that for some run (2.3.2) there does not exist a situation s with the property that $Tr_D(s_n, s)$; i.e., there is no possibility of continuation of the run (2.3.2), even if the situation s_n is *not* terminal in D . It is said then that the algorithm D is stuck in the situation s_n . A run with this property is not qualified as terminated in D .

The fact that an algorithm may get jammed in some situations leads us to isolate the so-called integral iterative algorithms. A situation $s = (w, a)$ is *non-terminal* if $a \neq \omega$. An iterative algorithm D is *integral* if the domain of the relation Tr_D coincides with the set of all non-terminal situations; that is, if for every non-terminal situation s there exists a situation s' such that $Tr_D(s, s')$.

If the system $\mathbf{M} = (W, R, \mathcal{A})$ is normal and every atomic action $A \in \mathcal{A}$ is a total function (with domain W) and the iterative algorithm D for \mathbf{M} has the property that for every triple $(a, A, b) \in \mathbf{LA}$ with $b \neq \omega$ there exists a triple $(c, B, d) \in \mathbf{LA}$ such that $b = c$, then D is integral.

Infinite (or *divergent*) *runs* of D are defined as infinite sequences of situations

$$(s_0, s_1, \dots, s_n, \dots)$$

such that s_0 is an initial situation of D and $(s_i, s_{i+1}) \in Tr_D$ for all i . Thus, no terminal situation occurs in a divergent run.

Every elementary action system $\mathbf{M} = (W, R, \mathcal{A})$ with a distinguished iterative algorithm D for \mathbf{M} forms a situational action system. For let

$$\mathbf{M}^s := (W, R, \mathcal{A}, S_D, Tr_D, f),$$

where $S_D (= W \times V)$ is the set of all possible situations of D , Tr_D is the transition relation in D , and $f : W \times V \rightarrow W$ is the projection onto W . \mathbf{M}^s is a situational action system in the sense of Definition 2.2.1.

Each state of W corresponds to a certain initial situation of \mathbf{M}^s . This is not the case in the iterative system associated with chess playing (Sect. 2.2). In a game of chess there is only one initial situation.

To each iterative algorithm (2.3.1) a binary relation Res_D on the set W of states is assigned and called the *resultant relation* of the algorithm D :

$$(u, w) \in Res_D \quad \text{if and only if} \quad ((u, \alpha), (w, \omega)) \in Tr_D^*.$$

Equivalently, $(u, w) \in Res_D$ if and only if there exists a terminated run (s_0, \dots, s_n) such that $s_0 = (u, \alpha)$ and $s_n = (w, \omega)$.

The resultant relation Res_D is a sub-relation of the reach of the elementary action system \mathbf{M} .

Example Let \mathbf{M} be an arbitrary elementary action system, and let (A_1, \dots, A_n) be a fixed non-empty sequence of atomic actions of \mathcal{A} . Let $V := \{0, 1, \dots, n\}$, $\alpha := 0$, $\omega := n$, and $\mathbf{LA} := \{(0, A_1, 1), \dots, (n-1, A_n, n)\}$. Then $D := (W, V, \alpha, \omega, \mathbf{LA})$ is an algorithm for \mathbf{M} . The resultant relation of D is equal to $(A_1 \cap R) \circ \dots \circ (A_n \cap R)$. \square

An iterative algorithm (2.3.1) is *well-designed* if and only if for every labeled action $(a, A, b) \in \mathbf{LA}$ there exists a finite string A_1, \dots, A_n of atomic actions of D and a finite sequence a_0, a_1, \dots, a_n of labels of V such that $a_0 = \alpha$, $a_n = \omega$, $(a_i, A_{i+1}, a_{i+1}) \in \mathbf{LA}$ for all $i \leq n-1$, and (a, A, b) is equal to a triple (a_i, A_{i+1}, a_{i+1}) for some $i \leq n-1$. Intuitively, D is well-designed if every labeled action of \mathbf{LA} is employed in some terminated run of D .

Well-designed algorithms are singled out for technical reasons. Each algorithm D can be transformed into a well-designed algorithm by deleting from \mathbf{LA} those labeled actions (a, A, b) which do not satisfy the above condition. This operation does not change the resultant relation of the algorithm.

We recall that $\text{REG}(\mathcal{A})$ denotes the family of all regular compound actions (over \mathcal{A}), and $\text{REG}^+(\mathcal{A}) := \{\mathbf{B} \in \text{REG}(\mathcal{A}) : \varepsilon \notin \mathbf{B}\}$. The properties of the family $\text{REG}(\mathcal{A})$ are independent of the internal structure of an action system $\mathbf{M} = (W, R, \mathcal{A})$; the cardinality of \mathcal{A} is the only factor that matters. This fact lead us to distinguish, for each $\mathbf{B} \in C\mathcal{A}$, the set $\mathcal{A}_{\mathbf{B}}$ of atomic actions of \mathcal{A} occurring in the strings of \mathbf{B} .

Lemma 2.3.1 *If $\mathbf{B} \in \text{REG}(\mathcal{A})$, then $\mathcal{A}_{\mathbf{B}}$ is finite.*

The lemma follows from the definition of a regular language over a (possibly infinite) alphabet (see Sect. 1.6). \square

Let \mathbf{A} be a family of binary relations on a set W . The *positive Kleene closure* of \mathbf{A} , denoted by $Cl^+(\mathbf{A})$, is the least family \mathbf{B} of binary relations on W which includes \mathbf{A} as a subfamily and satisfies the following conditions:

- (i) $\emptyset \in \mathbf{B}$
- (ii) if $\{P, Q\} \subseteq \mathbf{B}$, then $\{P \circ Q, P \cup Q, P^+\} \subseteq \mathbf{B}$.

The *Kleene closure* of \mathbf{A} , denoted by $Cl^*(\mathbf{A})$, is the least family \mathbf{B} of binary relations on W which includes \mathbf{A} as a subfamily and satisfies

- (i) $\emptyset, E_W \in \mathbf{B}$
- (ii) if $\{P, Q\} \subseteq \mathbf{B}$, then $\{P \circ Q, P \cup Q, P^*\} \subseteq \mathbf{B}$.

The purpose of this section is to determine the principles of the functioning of elementary action systems \mathbf{M} which are regulated by appropriate automata. This will give rise to a class of situational action systems erected on atomic systems.

We henceforth will work with action systems $\mathbf{M} = (W, R, \mathcal{A})$ which are normal and in which the relation R is reflexive. These assumptions imply the total performatibility of all non-empty compound actions of $C\mathcal{A}$ in \mathbf{M} , and of the action ε in particular. (If the reflexivity of R is dropped, one should rather work with the compound actions of $C^+\mathcal{A}$ rather than those of $C\mathcal{A}$ and with the positive Kleene closure Cl^+ .)

Proposition 2.3.2 *Let $M = (W, R, \mathcal{A})$ be a normal elementary action system with reflexive R . Then the following conditions hold:*

- (i) *For every regular action $\mathbf{B} \in \text{REG}(\mathcal{A})$, $\text{Res } \mathbf{B}$, the resultant relation of \mathbf{B} defined as in the formula (1.7.3) of Sect. 1.7, belongs to $Cl(\mathcal{A}_{\mathbf{B}})$;*
- (ii) *Conversely, for every binary relation $Q \in Cl(\mathcal{A})$ there exists a regular action $\mathbf{B} \in \text{REG}(\mathcal{A})$ such that $\text{Res } \mathbf{B} = Q$ and $Q \in Cl(\mathcal{A}_{\mathbf{B}})$.*

Proof (i) We define:

$$\mathcal{P} := \{\mathbf{B} \in C\mathcal{A} : \text{Res } \mathbf{B} \in Cl(\mathcal{A}_{\mathbf{B}})\}.$$

($\mathcal{A}_{\mathbf{B}}$ is a set of binary relations on W , viz, the set of atomic actions occurring in the compound action \mathbf{B} . As R is reflexive, $\text{Res } \mathbf{B}$ is reflexive, for all non-empty \mathbf{B} .)

\mathcal{P} has the following properties:

- (a) *If $\mathbf{B} \in C\mathcal{A}$ is finite then $\mathbf{B} \in \mathcal{P}$; in particular $\emptyset \in \mathcal{P}$;*
- (b) *If $\mathbf{B}, \mathbf{C} \in \mathcal{P}$, then $\mathbf{B} \cup \mathbf{C}$, $\mathbf{B} \circ \mathbf{C}$ and \mathbf{B}^* belong to \mathcal{P} as well.*

((b) directly follows from Lemma 1.7.2 and the remarks following it.) Thus, every regular action of $\text{REG}(\mathcal{A})$ belongs to \mathcal{P} .

(ii) We define:

$$\mathcal{L} := \{Q \subseteq W \times W : (\exists \mathbf{B}) \mathbf{B} \in \text{REG}(\mathcal{A}) \ \& \ Q = \text{Res } \mathbf{B}\}.$$

We claim that $Cl(\mathcal{A}) \subseteq \mathcal{L}$.

$\emptyset \in \mathcal{L}$ and $E_W \in \mathcal{L}$, because $\emptyset = \text{Res } \emptyset$ and $E_W = \text{Res } \varepsilon$. For every atomic action $Q \in \mathcal{A}$, $\mathbf{B} := \{Q\}$ is a regular action on M and, since M is normal, $\text{Res } \mathbf{B} = Q$. Hence, $\mathcal{A} \subseteq \mathcal{L}$. Now assume that $P, Q \in \mathcal{L}$. We shall show that $\{P \circ Q, P \cup Q, P^*\} \subseteq \mathcal{L}$. We have that $P = \text{Res } \mathbf{B}$ and $Q = \text{Res } \mathbf{C}$ for some regular compound actions \mathbf{B} and \mathbf{C} on M . Then, by Lemma 1.7.2, $P \circ Q = \text{Res } \mathbf{B} \circ \text{Res } \mathbf{C} = \text{Res } (\mathbf{B} \circ \mathbf{C})$, $P \cup Q = \text{Res } \mathbf{B} \cup \text{Res } \mathbf{C} = \text{Res } (\mathbf{B} \cup \mathbf{C})$, and, as R is reflexive, $(\text{Res } \mathbf{B})^* = \text{Res } (\mathbf{B}^*)$. As $\mathbf{B} \circ \mathbf{C}$, $\mathbf{B} \cup \mathbf{C}$, and \mathbf{B}^* are regular, this proves that $Cl(\mathcal{A}) \subseteq \mathcal{L}$.

If $Q = \text{Res } \mathbf{B}$ for some regular \mathbf{B} , then by (i), $\text{Res } \mathbf{B} \in Cl(\mathcal{A}_{\mathbf{B}})$, and hence $Q \in Cl(\mathcal{A}_{\mathbf{B}})$. So (ii) holds. \square

An iterative algorithm $D = (W, V, \alpha, \omega, \mathcal{L}\mathcal{A})$ for M is said to *accept a string* $A_1 \dots A_n$ of atomic actions of M if there exists a sequence a_0, \dots, a_n of labels of V such that $a_0 = \alpha$, $a_n = \omega$ and $(a_i, A_{i+1}, a_{i+1}) \in \mathcal{L}\mathcal{A}$, for all $i \leq n - 1$. (Thus D accepts the empty string ε if and only if $\alpha = \omega$. D accepts no strings if and only if $\mathcal{L}\mathcal{A}$ is empty and $\alpha \neq \omega$.)

The *total* (compound) *action* of the algorithm D , denoted by $\mathbf{A}(D)$, is defined as the set of all strings of atomic actions accepted by D .

In particular, for a given algorithm $D = (W, V, \alpha, \omega, \mathcal{L}\mathcal{A})$, if $\mathcal{L}\mathcal{A}$ is empty and $\alpha \neq \omega$, then the total action of D is the empty compound action \emptyset . If $\mathcal{L}\mathcal{A}$ is empty and $\alpha = \omega$, then the total action of D equals ε .

Thus, D is well-designed if and only if $\mathbf{A}_D = \mathbf{A}_{\mathbf{A}(D)}$, i.e., every atomic action of D occurs in some sequence belonging to $\mathbf{A}(D)$.

Proposition 2.3.3 *Let D be an iterative algorithm for an elementary action system $M = (W, R, \mathbf{A})$. Then:*

- (i) *the total action $\mathbf{A}(D)$ of D is regular, and*
- (ii) *the resultant relation of the total action $\mathbf{A}(D)$ (in the sense of formula (1.7.3) of Sect. 1.7) coincides with the resultant relation Res_D of D .*

Proof To prove (i) we shall employ some facts from the theory of formal grammars. Let $D = (W, V, \alpha, \omega, \mathbf{LA})$. We shall treat the sets V and \mathbf{A}_D (the set of atomic actions of D) as disjoint alphabets. V will be the auxiliary alphabet (variables) while \mathbf{A}_D will be the terminal alphabet (terminals). The initial label will be regarded as the start symbol. The set \mathbf{LA} of labeled actions of D specifies, in turn, a certain finite set \mathcal{P} of productions:

$$\mathcal{P} := \{a \rightarrow Ab : (a, A, b) \in \mathbf{LA}\} \cup \{a \rightarrow A : (\exists a \in V) (a, A, \omega) \in \mathbf{LA}\},$$

where ω is the terminal label. The system $\mathbf{G}_D := (\mathbf{A}_D, V, \mathcal{P}, \alpha)$ is, thus, a combinatorial grammar.

Lemma 2.3.4 *The language generated by the grammar \mathbf{G}_D coincides with the total action $\mathbf{A}(D)$ of the algorithm D .*

Proof of the lemma. We first show that $\mathbf{A}(D) \subseteq L(\mathbf{G}_D)$. Let $A_1 \dots A_n \in \mathbf{A}(D)$. Then, for some $a_0, \dots, a_n \in V$ with $a_0 = \alpha$ and $a_n = \omega$, we have that $(a_i, A_{i+1}, a_{i+1}) \in \mathbf{LA}$ for all $i \leq n-1$. This implies that

$$\alpha \Rightarrow A_1 a_1 \Rightarrow A_1 A_2 a_2 \Rightarrow \dots \Rightarrow A_1 \dots A_{n-1} a_{n-1} \Rightarrow A_1 \dots A_{n-1} A_n$$

is a terminated derivation in \mathbf{G}_D . Hence $A_1 \dots A_n \in L(\mathbf{G}_D)$.

To prove the reverse inclusion $L(\mathbf{G}_D) \subseteq \mathbf{A}(D)$, suppose that $A_1 \dots A_n \in L(\mathbf{G}_D)$ and let $z_0 \Rightarrow z_1 \Rightarrow \dots \Rightarrow z_{m-1} \Rightarrow z_m$ be a derivation of $A_1 \dots A_n$ in \mathbf{G}_D , where $z_0 = \alpha$ and $z_m = A_1 \dots A_n$. We show by induction for $i = 1, \dots, n-1$ that $z_i = A_1 \dots A_i a_i$ for some $a_i \in V$. This holds for $i = 1$. Let $i \geq 2$ and assume that $z_j = A_1 \dots A_j a_j$ for all $j \leq i-1$. In particular, $z_{i-1} = A_1 \dots A_{i-1} a_{i-1}$. Since $i \leq n-2$, no production of the form $a \rightarrow A$ can be applied to z_{i-1} (for otherwise we would get a word of the form $A_1 \dots A_{i-1} A$ which is different from $A_1 \dots A_i$; and the string $A_1 \dots A_{i-1} A$ blocks further derivations). So the word z_{i-1} derives z_i by means of a production $a_{i-1} \rightarrow Ab$ in which $A = A_i$.

Since $z_{n-1} = A_1 \dots A_{n-1} a_{n-1}$, we see that z_n must be equal to $A_1 \dots A_n$. So $m = n$, $z_m = z_n$, and the word z_{n-1} derives z_n by means of the production $a_{n-1} \rightarrow A_n$.

It follows from the definition of \mathbf{G}_D that the sequence $a_0 := \alpha, a_1, \dots, a_{n-1}, a_n := \omega$ has the property that $(a_i, A_{i+1}, a_{i+1}) \in \mathbf{LA}$ for all $i \leq n-1$. So $A_1 \dots A_n \in \mathbf{A}(D)$. This concludes the proof of the lemma. \square

\mathbf{G}_D is a right linear-grammar. This fact, in view of Theorem 1.6.3, implies that the language $L(\mathbf{G}_D) (= \mathbf{A}(D))$ is regular. So (i) holds.

(ii) We have to show that $\text{Res}_D = \text{Res } \mathbf{A}(D)$. Let $(u, w) \in \text{Res}_D$, i.e., $((u, \alpha), (w, \omega)) \in (\text{Tr}_D)^+$. Hence there exists a finite sequence of situations s_0, \dots, s_n , where $s_i = (u_i, a_i)$, for all $i \leq n$, such that $s_0 = (u, \alpha)$, $s_n = (w, \omega)$ and, for every $i \leq n-1$, there exists an action $A_{i+1} \in \mathbf{A}(D)$ with the property that $(a_i, A_{i+1}, a_{i+1}) \in \mathcal{P}$ and $u_i A_{i+1}, R u_{i+1}$. It follows that the sequence $A_1 \dots A_n$ belongs to $\mathbf{A}(D)$ and $(u, w) \in (A_1 \cap R) \circ \dots \circ (A_n \cap R)$, i.e., $(u, w) \in \text{Res } \mathbf{A}(D)$.

Conversely, let $(u, w) \in \text{Res } \mathbf{A}(D)$. Hence there exists a sequence $A_1 \dots A_n \in \mathbf{A}(D)$ and a string of states u_0, \dots, u_n such that $u_0 = u$, $u_n = w$, and $u_i A_{i+1}, R u_{i+1}$ for all $i \leq n-1$. Since $A_1 \dots A_n \in \mathbf{A}(D)$, there exists also a sequence of labels a_0, \dots, a_n such that $a_0 = \alpha$, $a_n = \omega$, and $(a_i, A_{i+1}, a_{i+1}) \in \mathbf{LA}$, for all $i \leq n-1$, which means that $((u_i, a_i), (u_{i+1}, a_{i+1})) \in \text{Tr}_D$ for every $i \leq n-1$. Hence $(u, w) \in \text{Res}_D$. This proves that $\text{Res}_D = \text{Res } \mathbf{A}(D)$. \square

Corollary 2.3.5 *Let $D = (W, V, \alpha, \omega, \mathbf{LA})$ be an iterative algorithm for a normal system $\mathbf{M} = (W, R, \mathcal{A})$ with reflexive R . Then Res_D belongs to the Kleene closure of \mathcal{A}_D .*

Proof Use Propositions 2.3.2 and 2.3.3. \square

Regular languages are those accepted by finite automata. Regular compound actions, however, can be also conveniently characterized in terms of total actions of iterative algorithms.

Theorem 2.3.6 *Let $\mathbf{M} = (W, R, \mathcal{A})$ be an elementary action system with reflexive R . For every compound action $\mathbf{B} \in C\mathcal{A}$ the following conditions are equivalent:*

- (i) \mathbf{B} is regular.
- (ii) *There exists a well-designed iterative algorithm D for \mathbf{M} such that the total action of D is equal to \mathbf{B} and $\mathcal{A}_D = \mathcal{A}_{\mathbf{B}}$.*

Proof (ii) \Rightarrow (i) This is an immediate consequence of Proposition 2.3.3.

(i) \Rightarrow (ii) The proof of this part of the theorem consists in, for every regular action $\mathbf{B} \in C\mathcal{A}$, the effective construction of an iterative algorithm D satisfying the conditions mentioned in clause (ii). The proof is by induction on complexity of the regular action \mathbf{B} .

Suppose first that the action \mathbf{B} is finite and non-empty. Let the initial and the terminal labels α and ω be fixed. For each sequence $\underline{A} = A_1 \dots A_n \in \mathbf{B}$ we select a string of distinct labels a_1, \dots, a_{n-1} , and define:

$$\begin{aligned} P_{\underline{A}} &:= \{(\alpha, A_1, a_1), \dots, (a_{i-1}, A_i, a_i), \dots, (a_{n-1}, A_n, \omega)\}, \\ \mathbf{LA} &:= \bigcup \{P_{\underline{A}} : \underline{A} \in \mathbf{B}\}. \end{aligned}$$

Since \mathbf{B} is assumed to be finite, \mathbf{LA} is finite as well.

Let V be the collection of all the labels occurring in the labeled actions of \mathbf{LA} . Then $D = (W, V, \alpha, \omega, \mathbf{LA})$ is a well-designed iterative algorithm for \mathbf{M} . Moreover, it follows from the construction of D that the compound action of D is equal to \mathbf{B} and that $\mathcal{A}_D = \mathcal{A}_\mathbf{B}$.

If \mathbf{B} is empty, it is assumed that $\alpha \neq \omega$ and \mathbf{LA} is empty. It follows that the transition relation Tr_D is empty. Consequently, the action of D is equal to \emptyset .

For the compound action ε , it is assumed that $\alpha = \omega$ and \mathbf{LA} is empty. This implies that the transition relation Tr_D is equal to E_W . Consequently, the action of D is equal to ε .

Now suppose \mathbf{B} and \mathbf{C} are compound actions on \mathbf{M} such that well-designed algorithms D_1 and D_2 for \mathbf{M} have been constructed so that the total action of D_1 is equal to \mathbf{B} and the total action of D_2 is equal to \mathbf{C} . We shall build an algorithm D for \mathbf{M} such that the total action of D is equal to $\mathbf{B} \cup \mathbf{C}$. Let $D_i = (W_i, V_i, \alpha_i, \omega_i, \mathbf{LA}_i)$ for $i = 1, 2$. Without loss of generality we can assume that $\alpha_1 = \alpha_2 = \alpha$ and $\omega_1 = \omega_2 = \omega$, and that the sets $V_1 \setminus \{\alpha, \omega\}$ and $V_2 \setminus \{\alpha, \omega\}$ are disjoint. (Otherwise one can simply rename the labels of V_1 and V_2 .) We then put:

$$D := (W, V_1 \cup V_2, \alpha, \omega, \mathbf{LA}_1 \cup \mathbf{LA}_2).$$

It is easy to see that the total action of D is equal to $\mathbf{B} \cup \mathbf{C}$ and $\mathcal{A}_D = \mathcal{A}_{\mathbf{B} \cup \mathbf{C}}$. Moreover D is well-designed.

We shall now construct an algorithm D° for \mathbf{M} such that the total action of D° is equal to $\mathbf{B} \circ \mathbf{C}$. We put:

$$D^\circ := (W, V_1 \cup V_2 \cup \{c\}, \alpha, \omega, \mathbf{LA}^\circ),$$

where c is a new label adjoined to $V_1 \cup V_2$, and \mathbf{LA} is the ‘concatenation’ of \mathbf{LA}_1 and \mathbf{LA}_2 . To define \mathbf{LA}° formally, we need an auxiliary notion.

A *terminated* sequence of labeled atomic actions of an algorithm $D = (W, V, \alpha, \omega, \mathbf{LA})$ is any finite sequence of triples

$$(a_0, A_1, a_1), (a_1, A_2, a_2), \dots, (a_{n-2}, A_{n-1}, a_{n-1}), (a_{n-1}, A_n, a_n) \quad (2.3.3)$$

of \mathbf{LA} such that $a_0 = \alpha$ and $a_n = \omega$.

For any terminated sequence (2.3.3) of elements of \mathbf{LA}_1 and any terminated sequence

$$(b_0, B_1, b_1), (b_1, B_2, b_2), \dots, (b_{m-2}, B_{m-1}, b_{m-1}), (b_{m-1}, B_m, b_m)$$

of elements of \mathbf{LA}_2 , where $b_0 = \alpha$ and $b_m = \omega$, we define a new sequence

$$\begin{aligned} &(\alpha, A_1, a_1), (a_1, A_2, a_2), \dots, (a_{n-2}, A_{n-1}, a_{n-1}), (a_{n-1}, A - n, c), \\ &(c, B_1, b_1), (b_1, B_2, b_2), \dots, (b_{m-2}, B_{m-1}, b_{m-1}), (b_{m-1}, B_m, \omega), \end{aligned} \quad (2.3.4)$$

where c is the new label adjoined to $V_1 \cup V_2$.

\mathbf{LA}° is, by definition, the set of all labeled atomic actions occurring in all the sequences of the form (2.3.4).

It is easy to show that the total action of D° is equal to $\mathbf{B} \circ \mathbf{C}$, the set of atomic actions of D° is equal to $\mathcal{A}_{\mathbf{B} \circ \mathbf{C}}$, and D° is well-designed.

Having given a well-designed algorithm $D = (W, V, \alpha, \omega, \mathbf{LA})$ for \mathbf{M} such that the total action of D is equal to \mathbf{B} , we define a new well-designed algorithm

$$D^\# = (W, V^\#, \alpha^\#, \omega^\#, \mathbf{LA}^\#).$$

Here $\alpha^\# = \omega^\#$, $V^\# = (V \setminus \{\alpha, \omega\}) \cup \{\alpha^\#, c\}$, where $\alpha^\#$ and c are new labels not occurring in V . Thus, the labels α and ω are removed from V and new labels $\alpha^\#$, c are adjoined. $\mathbf{LA}^\#$ is defined as follows: for every finite number, say k , of terminated non-empty sequences of labeled atomic actions of D :

$$(a_0^j, A_1^j, a_1^j), (a_1^j, A_2^j, a_2^j), \dots, (a_{n_j-1}^j, A_{n_j}^j, a_{n_j}^j) \quad (2.3.5)$$

where $j = 1, \dots, k$, and $a_0^j = \alpha$, $a_{n_j}^j = \omega$ for all j , a new sequence of labeled actions is formed:

$$\begin{aligned} &(\alpha^\#, A_1^1, a_1^1), (a_1^1, A_2^1, a_2^1), \dots, (a_{n_1-1}^1, A_{n_1}^1, c), \\ &(c, A_1^2, a_1^2), (a_1^2, A_2^2, a_2^2), \dots, (a_{n_2-1}^2, A_{n_2}^2, c), \\ &\quad \vdots \\ &(c, A_1^j, a_1^j), (a_1^j, A_2^j, a_2^j), \dots, (a_{n_j-1}^j, A_{n_j}^j, c), \\ &\quad \vdots \\ &(c, A_1^k, a_1^k), (a_1^k, A_2^k, a_2^k), \dots, (a_{n_k-1}^k, A_{n_k}^k, \omega^\#). \end{aligned} \quad (2.3.6)$$

If $k = 0$, (2.3.6) reduces to the sequence $(\alpha^\#, \omega^\#)$, which is equal to $(\alpha^\#, \alpha^\#)$.

$\mathbf{LA}^\#$ is the set of all labeled atomic actions occurring in the sequences (2.3.6).

The total compound action of the algorithm $D^\#$ is equal to \mathbf{B}^* , the iterative closure of \mathbf{B} . Moreover the set of atomic actions of $D^\#$ is equal to the set of all atomic actions occurring in \mathbf{B} .

It follows from the above constructions that for every action $\mathbf{B} \in \text{REG}(\mathcal{A})$ there exists a well-designed iterative algorithm D such that the total action of D is equal to \mathbf{B} and $\mathcal{A}_D = \mathcal{A}_{\mathbf{B}}$. This completes the proof of the theorem 2.3.6. \square

The following corollary (in a slightly different but equivalent form) is due to Mazurkiewicz (1972a):

Corollary 2.3.7 *Let $\mathbf{M} = (W, R, \mathcal{A})$ be a normal action system with reflexive R , and let Q be a relation that belongs to the Kleene closure of \mathcal{A} . Then there exists an iterative algorithm D for \mathbf{M} such that $\text{Res}_D = Q$ and $Q \in \text{Cl}(\mathcal{A}_D)$.*

Proof According to Proposition 2.3.2.(ii), there exists a regular compound action \mathbf{B} on \mathbf{M} such that $Q = \text{Res } \mathbf{B}$ and $Q \in Cl(\mathcal{A}_{\mathbf{B}})$. In turn, in view of Theorem 2.3.6, there exists an iterative algorithm D for \mathbf{M} , whose total action $\mathbf{A}(D)$ is equal to \mathbf{B} and $\mathcal{A}_D = \mathcal{A}_{\mathbf{B}}$. Hence $Q \in Cl(\mathcal{A}_D)$. Since $\mathbf{A}(D) = \mathbf{B}$, Proposition 2.3.3 yields $\text{Res}_D = \text{Res } \mathbf{B} = Q$. \square

Let $\mathbf{M} = (W, R, \mathcal{A})$ be a *finite* elementary action system. In Sect. 1.7, the compound actions $\Phi \boxtimes_i \Psi$ have been defined for all $\Phi, \Psi \subseteq W$ and $i = 1, \dots, 4$. Since these actions are regular, Theorem 2.3.6 implies that $\Phi \boxtimes_i \Psi$ is ‘algorithmizable’ for all Φ, Ψ and $i = 1, \dots, 4$; that is, there exists a well-designed iterative algorithm D for \mathbf{M} such that the total action $\mathbf{A}(D)$ of D is equal to $\Phi \boxtimes_i \Psi$. In particular this implies that the resultant relation of $\Phi \boxtimes_i \Psi$ coincides with Res_D . Speaking figuratively, it means that the ways the task (Φ, Ψ) is implemented and the goal Ψ is reached in the finite system \mathbf{M} is subordinated to a well-designed iterative algorithm. Thus, there exists, at least a theoretical possibility of ‘automatizing’ the tasks (Φ, Ψ) in the system \mathbf{M} . The practical realization of this idea requires adopting realistic assumptions as regards the cardinalities of the sets \mathcal{A} , W and V (the set of labels of the algorithm).

The reach Re_M of an elementary action system \mathbf{M} is equal to the resultant relation of the action \mathcal{A}^* , the set of all non-empty strings of atomic actions. Since \mathcal{A}^* is regular if \mathcal{A} is finite, Theorem 2.3.6 and Corollary 2.3.7 imply that Re_M belongs to the Kleene closure of \mathcal{A} whenever \mathbf{M} is a finite normal elementary system with reflexive R . But this fact can be established directly by appealing to the definition of Re_M .

Let $\mathbf{M} = (W, R, \mathcal{A})$ be an elementary action system. Two compound actions $\mathbf{B}, \mathbf{C} \in C\mathcal{A}$ are *equivalent over \mathbf{M}* if and only if $\mathcal{A}_{\mathbf{B}} = \mathcal{A}_{\mathbf{C}}$ and $\text{Res } \mathbf{B} = \text{Res } \mathbf{C}$, i.e., \mathbf{B} and \mathbf{C} are ‘built up’ with the same atomic actions and the resultant relations of \mathbf{B} and \mathbf{C} are identical.

The facts we have established thus far enable us to draw the following simple corollary:

Corollary 2.3.8 *Let \mathbf{M} be an elementary normal action system with reflexive R . Then for every action $\mathbf{B} \in C\mathcal{A}$ the following conditions are equivalent:*

- (i) \mathbf{B} is equivalent to a regular action;
- (ii) There exists a well-designed algorithm D for \mathbf{M} such that $\mathcal{A}_D = \mathcal{A}_{\mathbf{B}}$ and the resultant relation of \mathbf{B} is equal to Res_D .

Proof (i) \Rightarrow (ii) Suppose \mathbf{B} is equivalent to a regular action \mathbf{C} . By Theorem 2.3.6 there exists an iterative algorithm D for \mathbf{M} such that $\mathcal{A}_D = \mathcal{A}_{\mathbf{C}}$ and the total action $\mathbf{A}(D)$ of D is equal to \mathbf{C} . Hence $\mathcal{A}_D = \mathcal{A}_{\mathbf{B}}$ and $\text{Res } \mathbf{B} = \text{Res } \mathbf{C} = \text{Res } \mathbf{A}(D) = \text{Res}_D$. So (ii) holds.

(ii) \Rightarrow (i) Assume (ii) Since D is well-designed, $\mathcal{A}_{\mathbf{B}} = \mathcal{A}_D = \mathcal{A}_{\mathbf{A}(D)}$. By Proposition 2.3.3, the action $\mathbf{A}(D)$ of D is regular. So \mathbf{B} is equivalent to $\mathbf{A}(D)$ over \mathbf{M} . \square

The above corollary gives rise to the problem of the preservation of regular actions by the relation of equivalence. More specifically, we ask if the following is true in normal action systems \mathbf{M} :

- (*) for any two compound actions $\mathbf{B}, \mathbf{C} \in C\mathcal{A}$, if \mathbf{B} is regular and equivalent to \mathbf{C} over \mathbf{M} , then \mathbf{C} is regular as well.

To answer the above question, we shall make use of the Myhill-Nerode Theorem. Let Σ be an alphabet. For an arbitrary language L over Σ the equivalence relation R_L on the set Σ^* for all non-empty words is defined as follows:

$x R_L y$ if and only if for each word z , either both or neither xz and yz is in L , i.e., for all z , $xz \in L$ if and only if $yz \in L$.

The Myhill-Nerode Theorem states that for an arbitrary language $L \subseteq \Sigma^*$, the relation R_L is of finite index (i.e., the number of equivalence classes of R_L is finite) if and only if L is accepted by some finite automaton (if and only if, by Theorem 1.6.5, L is regular).

Let $\Sigma := \{A, B\}$ and let $L := \{A^n B^n : n = 1, 2, \dots\}$. The language L is not regular. To show this we define $x_n := A^n$, $y_n := B^n$ for all $n \geq 1$. If $m \neq n$ then $x_m x_n \notin L$ and $x_n x_n \in L$. Hence $m \neq n$ implies that $[x_m] \neq [x_n]$, where $[x]$ denotes the equivalence class of x with respect to R_L . This means that the index of R_L is infinite. So L cannot be regular.

Let $\mathbf{M} = (W, R, \mathcal{A})$ be an action system, where $\mathcal{A} = \{A, B\}$, the relations A and B are transitive, $B \subseteq A \subseteq R$ and R is reflexive. The compound action \mathbf{B} is defined in the same manner as the language L as above, that is, $\mathbf{B} := \{A^n B^n : n \geq 1\}$.

The above argument shows that \mathbf{B} is not regular. It is also easy to prove that $\text{Res } \mathbf{B} = A \circ B$. Hence trivially $\text{Res } \mathbf{B}$ belongs to the positive Kleene closure of $\mathcal{A}_B = \{A, B\}$ which implies, by Corollary 2.3.7, that there exists an iterative algorithm D for \mathbf{M} such that $\mathcal{A}_D = \mathcal{A}$ and the resultant relation of D is equal to $\text{Res } \mathbf{B}$. Thus, \mathbf{B} is equivalent over \mathbf{M} to the regular action $\mathbf{A}(D)$ of the algorithm. As \mathbf{B} is not regular, we see that (*) does not hold for \mathbf{B} and \mathbf{C} .

The above result is not surprising because the definition of a regular compound action over an action system \mathbf{M} does not take into account the internal set-theoretic structure of the family of atomic actions of \mathbf{M} .

2.4 Pushdown Automata and Pushdown Algorithms

A grammar $\mathbf{G} = (\Sigma, V, \mathcal{P}, \alpha)$ is *context-free* if each production of \mathcal{P} is of the form $a \rightarrow x$, where a is a variable and x is a string of symbols from $(\Sigma \cup V)^*$.

The adjective ‘context-free’ comes from the fact that every production of the above sort can be treated as a substitution rule enabling to replace the symbol a by the word x irrespective of the context in which the symbol occurs.

A language L is *context-free* if it is generated by some context-free grammar.

Example 2.4.1 Let $L := \{A^n B^n : n = 1, 2, \dots\}$. We showed in Sect. 2.3 that L is not regular. Let us consider the grammar $\mathbf{G} = (\Sigma, V, \mathcal{P}, \alpha)$, where $\Sigma = \{A, B\}$, $V = \{\alpha\}$ and $\mathcal{P} = \{\alpha \rightarrow A\alpha B, \alpha \rightarrow AB\}$. \mathbf{G} is context-free. By applying the first production $n - 1$ times, followed by an application of the second production, we have

$$\alpha \Rightarrow A\alpha B \Rightarrow AA\alpha BB \Rightarrow A^3\alpha B^3 \Rightarrow \dots \Rightarrow A^{n-1}\alpha B^{n-1} \Rightarrow A^n B^n.$$

Furthermore, induction on the length of a derivation shows that if $\alpha \Rightarrow_{\mathbf{G}} z_1 \Rightarrow_{\mathbf{G}} z_2 \dots z_{n-1} \Rightarrow_{\mathbf{G}} z_n$ then $z_n = A^n B^n$ if z_n is a terminal and $z_n = A^n \alpha B^n$ otherwise. Thus $L = L(\mathbf{G})$. \square

$\text{CFL}(\Sigma)$ denotes the family of all context-free languages over Σ . $\text{CFL}^+(\Sigma)$ is the class of all context-free languages over Σ which do not involve the empty word ε ,

$$\text{CFL}^+(\Sigma) := \{L : L \in \text{CFL}(\Sigma) \text{ and } \varepsilon \notin L\}.$$

There are several ways by means of which one can restrict the format of productions without reducing the generative power of context-free grammars. If L is a non-empty context-free language and ε is not in L , then L can be generated by a context-free grammar \mathbf{G} which uses productions whose right-hand sides each start with a terminal symbol followed by some (possibly empty) string of variables. This special form is called *Greibach normal form*. We shall formulate this result in a slightly sharper form.

Let \mathbf{G} be context-free. If at each step in a \mathbf{G} -derivation a production is applied to the *leftmost* variable, then the derivation is said to be *leftmost*. Similarly a derivation in which the rightmost variable is replaced at each step is said to be *rightmost*.

Theorem 2.4.2 (Greibach 1965) *Every context-free language L without ε is generated by a grammar \mathbf{G} for which every production is of the form $v \rightarrow Ax$, where v is a variable, A is a terminal and x is a (possibly empty) string of variables. Furthermore, every word of L can be derived by means of a leftmost derivation in \mathbf{G} .*

The proof of the above theorem is constructive—an algorithm is provided which, for every context free-grammar \mathbf{G} in which no production of the form $v \rightarrow \varepsilon$ occurs, converts it to Greibach normal form. We omit the details. \square

We shall define a broad class of algorithms which comprises iterative algorithms as a special, limit case.

Let $\mathbf{M} = (W, R, \mathcal{A})$ be an elementary action system. A *pushdown algorithm* for \mathbf{M} is a quadruple

$$D := (W, V, \alpha, \mathbf{LA}),$$

where W is, as above, the set of states of \mathbf{M} , V is a finite set called the *stack* alphabet, α is a particular stack symbol called the *start* symbol, and \mathbf{LA} is a finite set of labeled

atomic actions of M , i.e., LA is a finite set of triples of the form

$$(a, A, \beta), \quad (2.4.1)$$

where $A \in \mathcal{A}$, $a \in V$, and $\beta \in V^*$ is a possibly empty string of symbols of V . (The empty string $\varepsilon \in V^*$ is called the *end* symbol.) The labels a and β of (2.4.1) are called the *input* and the *output* label of (2.4.1), respectively.

The set

$$\mathcal{A}_D := \{A \in \mathcal{A} : (\exists a, \beta) (a, A, \beta) \in LA\}$$

is called the set of *atomic actions of the algorithm D*. The set is always finite.

Iterative algorithms may be regarded as a limit case of pushdown algorithms. The former are pushdown algorithms in which the terminal labels β of labeled atomic actions $(a, A, \beta) \in LA$ have length ≤ 1 , i.e., $|\beta| \leq 1$.

The elements of the set $S_D := W \times V^*$ are called the possible *situations* of the algorithm. If $s = (w, \beta) \in S_D$, w is the state corresponding to s and β is the label of the situation s . β is also called the *stack* corresponding to the situation s . A situation s is *initial* if it is of the form (w, α) for some $w \in W$. A situation s is *terminal* if it is labeled by the empty string ε of stack symbols; i.e., the stack in this situation is empty; so terminal situations are of the form (w, ε) , $w \in W$.

A labeled action (a, A, β) transforms a situation $s_1 = (w_1, \gamma_1)$ into a situation $s_2 = (w_2, \gamma_2)$ if and only if $\gamma_1 = a\sigma$, $\gamma_2 = \beta\sigma$ for some $\sigma \in V^*$, and $w_1 A, R w_2$. Thus, s_1 is a situation in which a is the top symbol on the stack and s_2 is the situation which results from s_1 by replacing the top symbol a by the string β and performing the atomic action A so that the system moves from w_1 to the state w_2 .

The transition relation Tr_D in a pushdown algorithm D is defined similarly as in the case of iterative algorithms:

$(s_1, s_2) \in Tr_D$ if and only if there exists a labeled action $(a, A, \beta) \in LA$ which transforms s_1 into s_2 .

The notions of finite, infinite or terminated runs of situations in D are defined in the well-known way. The latter are finite sequences (s_0, \dots, s_n) such that s_0 is an initial situation, $Tr(s_i, s_{i+1})$ for all $i \leq n-1$, and s_n is a terminal situation.

Every elementary normal action system $M = (W, R, \mathcal{A})$ together with a distinguished pushdown algorithm D for M form, in a natural way, a situational action system. (To simplify matters, it is also assumed that R is reflexive.) We put:

$$M^s := (W, R, \mathcal{A}, S_D, Tr_D, f)$$

where S_D and Tr_D are defined as above, and f is the projection from $S_D (= W \times V^*)$ onto W . M^s is easily seen to satisfy the conditions imposed on situational action systems.

The resultant relation Res_D of a pushdown algorithm D is defined as follows:

$$(u, w) \in Res_D \quad \text{if and only if} \quad ((u, \alpha), (w, \varepsilon)) \in (Tr_D)^*.$$

Res_D is a subrelation of the reach of M .

The members of V^* represent possible contents of the stack. A string $a_1 \dots a_n \in V^*$ is a possible state of the stack with a_1 the top symbol of the stack. The strings of V^* are called *control labels* of the algorithm. Every pair (a, β) with $a \in V$ and $\beta \in V^*$ defines the *control relation*, denoted by $a \rightarrow \beta$, on the set V^* :

$$\gamma_1(a \rightarrow \beta)\gamma_2 \quad \text{if and only if} \quad (\exists \sigma \in V^*) (\gamma_1 = a\sigma \ \& \ \gamma_2 = \beta\sigma). \quad (2.4.2)$$

$a \rightarrow \beta$ is in fact a partial function, whose intended meaning is to modify control labels of situations during the work of the algorithm. The functions $a \rightarrow \beta$ enable us to express neatly possible transformations of the situations of the algorithm: a labeled action (a, A, β) transforms a situation $s_1 = (w_1, \gamma_1)$ into $s_2 = (w_2, \gamma_2)$ if and only if $\gamma_1(a \rightarrow \beta)\gamma_2$ and $w_1 A, R w_2$. The functions $a \rightarrow \beta$ are also used in defining a certain compound action on the system M called the total action of the pushdown algorithm D . More specifically, a finite string $A_1 \dots A_n$ of atomic actions of M is said to be *accepted by the algorithm D* if and only if there exists a sequence

$$(a_1, A_1, \beta_1), \dots, (a_n, A_n, \beta_n)$$

of labeled actions of LA and a sequence of control labels $\gamma_1, \dots, \gamma_{n+1} \in V^*$ with the following properties:

- (i) $\gamma_1 = a_1 = \alpha$ (= the start symbol), $\gamma_{n+1} = \varepsilon$ (= the end symbol)
- (ii) $\gamma_i(a_i \rightarrow \beta_i)\gamma_{i+1}$ for all $i \leq n$.

The elements of the string $A_1 \dots A_n$ are therefore consecutive actions of the algorithm leading from initial to terminal situations.

The *total action* of a pushdown algorithm D is defined as the set $\mathbf{A}(D)$ of all non-empty strings of atomic actions accepted by D .

Since \mathcal{A}_D , the set of all atomic actions the algorithm D involves, is always finite, the composite action $\mathbf{A}(D)$ is a language over the alphabet \mathcal{A}_D .

The resultant relation $Res \mathbf{A}(D)$ of the action $\mathbf{A}(D)$ coincides with the resultant relation Res_D of the algorithm D . This is an immediate consequence of the following simple observations:

- (i) If (s_0, \dots, s_n) is a terminated run of situations, where $s_i = (w_i, \gamma_i)$, $i = 0, 1, \dots, n$, then (w_0, w_1, \dots, w_n) is a realizable performance of $\mathbf{A}(D)$.
- (ii) If (w_0, w_1, \dots, w_n) is a realizable performance of $\mathbf{A}(D)$, then there exist labels $\alpha = \gamma_0, \dots, \gamma_n = \varepsilon$ such that $(w_0, \gamma_0), \dots, (w_n, \gamma_n)$ is a terminated run of situations.

The following theorem characterizes the compound actions of pushdown algorithms.

Theorem 2.4.3 *Let $M = (W, R, \mathcal{A})$ be an elementary action system. Then for every non-empty compound action $B \in C\mathcal{A}$ which does not contain the empty word ε the following conditions are equivalent:*

- (i) B is context-free,
- (ii) *There exists a pushdown algorithm D for M such that B is the total action of D .*

Proof (ii) \Rightarrow (i) Assume $B = A(D)$ for some pushdown automaton D for M . The idea of the proof is similar to the one applied in the proof of Proposition 2.3.3. The stack alphabet V is treated here as an auxiliary alphabet, while the members of \mathcal{A}_D are terminals. The label α is the start symbol. To the set LA of labeled actions of D the following finite list \mathcal{P} of productions is assigned:

$$\mathcal{P} := \{a \rightarrow A\beta : (a, A, \beta) \in LA\}.$$

The system

$$G_D := (\mathcal{A}_D, V, \mathcal{P}, \alpha)$$

is then a grammar in Greibach normal form.

It is easy to see that the total action $A(D)$ of D coincides with the set L of words over \mathcal{A}_D that can be derived from α by means of leftmost derivations in G_D . Theorem 2.4.2 implies that $L = L(G_D)$. Hence $B = A(D) = L(G_D)$. Since every language generated by a Greibach grammar is context-free, (i) thus follows.

(i) \Rightarrow (ii) The proof of this part is also easy. Let B be a context-free compound action (over the finite alphabet \mathcal{A}_B). Since B contains only non-empty finite strings of atomic actions, the language B is generated by a grammar

$$G = (\mathcal{A}_B, V, \mathcal{P}, \alpha) \tag{2.4.3}$$

in which every production of \mathcal{P} is of the form $a \rightarrow A\beta$, where $a \in V$, $A \in \mathcal{A}_B$, and β is a possibly empty string of elements of V . (This immediately follows from Greibach Normal Form Theorem 2.4.2; the auxiliary alphabet V is obviously individually suited to the language B .)

Let $D := (W, V, \alpha, LA)$ be the pushdown algorithm for M in which the set LA of labeled actions is defined as follows:

$$LA := \{(a, A, \beta) : a \rightarrow A\beta \text{ is in } \mathcal{P}\}.$$

The total action of the algorithm D is easily seen to be equal to the set of words over \mathcal{A}_B obtained by means of all leftmost derivations in the grammar (2.4.3). But in view of Theorem 2.4.2, the latter set of words is equal to the language (over \mathcal{A}_B) generated by the grammar (2.4.3). Thus, $A(D) = L(G) = B$. \square

Finite automata over some finite alphabet function according to a simple rule. They read consecutive symbols in a word while assuming different states. Formally, this requires assuming a specification of a finite set of states, a next-move function

δ from states and symbols to finite subsets of states, and a distinction between an initial state and the set of final states of the automaton.

Pushdown automata are much more powerful tools equipped with a restricted memory. A pushdown automaton can keep a finite stack during its reading a word. The automaton may use a separate stack alphabet. The ‘next-move’ function indicates, for each state, the symbol read and the top symbol of the stack, the next state of the automaton, as well as the string of stack symbols which will be entered or removed at the top of the stack.

Formally, a *pushdown automaton* (PDA) is a system

$$A = (W, \Sigma, \Gamma, \delta, w_0, \alpha, F) \quad (2.4.4)$$

where

- (i) W is a finite set of *states* of the automaton
- (ii) Σ is an alphabet called the *input alphabet*
- (iii) Γ is an alphabet called the *stack alphabet*
- (iv) w_0 is the *initial state* of the automaton
- (v) $\alpha \in \Gamma$ is a distinguished stack symbols called the *start symbol*
- (vi) $F \subseteq W$ is the set of *final states*
- (vii) δ is a mapping from $W \times \Sigma \cup \{\varepsilon\} \times \Gamma$ to finite non-empty subsets of $W \times \Gamma^*$.

The interpretation of

$$\delta(u, a, Z) = \{(w_1, \gamma_1), (w_2, \gamma_2), \dots, (w_m, \gamma_m)\},$$

where $w_1, \dots, w_m \in W$ and $\gamma_1, \dots, \gamma_m \in \Gamma^*$, is that the PDA in state u , with input symbol a and Z the top symbol on the stack, can, for any i ($i = 1, \dots, m$), enter state w_i , replace the symbol Z by the string γ_i , and advance the input head one symbol. The convention is adopted that the leftmost symbol of γ_i will be placed highest on the stack and the rightmost symbol lowest on the stack. It is not permitted to choose the state w_i and the string γ_j for some $j \neq i$ in one move. In turn,

$$\delta(u, \varepsilon, Z) = \{(w_1, \gamma_1), (w_2, \gamma_2), \dots, (w_m, \gamma_m)\}$$

says that M in the state u , independent of the input symbol being scanned with Z the top symbol on the stack, enters one of the states w_i and replaces Z by γ_i for $i = 1, \dots, m$. The input head is not advanced.

To formally describe the configuration of a PDA at a given instant one introduces the notion of an *instantaneous description* (ID). Each ID is defined as a triple (w, x, γ) , where w is a state, x is a string of input symbols, and γ is a string of stack symbols. According to the terminology adopted in this chapter, each instantaneous description $s = (w, x, \gamma)$ can be regarded as a possible situation of the automaton— w is the state of the automaton in the situation s while the label (x, γ) of the situation

s defines the string x of input symbols read by the automaton and the string γ of symbols on the stack.

If $\mathbf{M} = (W, \Sigma, \Gamma, \delta, w_0, \alpha, F)$ is a PDA, then we write $(u, ax, Z\gamma) \vdash_{\mathbf{M}} (w, x, \beta\gamma)$ if $\delta(u, a, Z)$ contains (w, β) . a is an input symbol or ε . $\vdash_{\mathbf{M}}^*$ is the reflexive and transitive closure of $\vdash_{\mathbf{M}}$. $L(\mathbf{M})$ is the language defined by *final state* of \mathbf{M} . Thus,

$$L(\mathbf{M}) := \{x \in \Sigma^* : (w_0, x, \alpha) \vdash_{\mathbf{M}}^* (w, \varepsilon, \gamma) \text{ for some } w \in F \text{ and } \gamma \in \Gamma^*\}.$$

Theorem 2.4.4 *The class of languages accepted for by PDA's coincides with the class of context free languages.*

The book by Hopcroft and Ullman (1979) contains a proof of the above theorem and other relevant facts concerning various aspects of PDA's. \square

Turing machines can be represented as situational action systems too. The transition relations between possible situations in such systems are defined in a more involved way however.

The characteristic feature of iterative and pushdown algorithms is that they provide *global* transformation rules of possible situations. In the simple case of iterative algorithms, transformation rules are defined by means of a finite set \mathbf{LA} of labeled actions and then by distinguishing the set of all possible pairs of sequences of labels (a_0, \dots, a_n) and atomic actions $A_1 \dots A_n$ such that $a_0 = \alpha$, $a_n = \omega$, and $(a_i, A_{i+1}, a_{i+1}) \in \mathbf{LA}$.

The conjugate sequences (a_0, \dots, a_n) and (A_1, \dots, A_n) define ways of transforming some situations into others. The transition from a situation labeled by a_i to a situation labeled by a_{i+1} is done through performing the action A_{i+1} (for $i = 0, \dots, n - 1$). The rules neither distinguish special states of the system nor do they impose any restrictions on transitions between the states corresponding to situations other than what relation R dictates. To put it briefly, they abstract from the *local* properties of the action system.

Schemes of action which are not captured by the above classes of algorithms can be easily provided. Let us fix an atomic action $A \in \mathcal{A}$. Let $\Phi \subset W$ be a non-empty set of states. The program ‘**while** Φ **do** A ’ defines the way the action A should be performed: whenever the system is in a state belonging to Φ , iterate performing the action A until the system moves out of Φ . (We assume here for simplicity that A is atomic; the definition of ‘**while** – **do**’ programs also makes sense for compound actions.) The program ‘**while** Φ **do** A ’ is identified with the set of all operations of the form

$$u_0 A u_1 A u_2 \dots u_{n-1} A u_n \quad (2.4.5)$$

such that $u_0, \dots, u_{n-1} \in \Phi$ and $u_n \notin \Phi$. (The above program also comprises infinite operations

$$u_0 A u_1 A u_2 \dots u_{n-1} A u_n \dots$$

such that $u_i \in \Phi$ for every $i \geq 0$.)

We claim that the above program is not determined by a pushdown algorithm. This statement requires precise formulation. To this end we define, for every pushdown algorithm $D = (W, V, \alpha, \mathbf{LA})$, the notion of the *program* $Pr(D)$ corresponding to the algorithm D (see Sect. 1.8).

$Pr(D)$ is the set of all finite operations $u_0 A_1 u_1 \dots u_{n-1} A_n u_n$ with $A_1 \dots A_n$ ranging over the total action $\mathbf{A}(D)$ of D . (As mentioned earlier, the notion of a program is usually conceived of as a certain syntactic entity. Here it is identified with its meaning; i.e., as a set of computations.)

Let $\mathbf{M} = (W, R, \{A\})$ be a normal elementary action system in which A is a total unary function on W . Let Φ be a proper non-empty subset of W closed with respect to A . (This means that for every pair of states $u, w \in W$, $u \in \Phi$ and $u A w$ imply that $w \in \Phi$.)

Proposition 2.4.5 *There does not exist a pushdown algorithm D for \mathbf{M} such that*

$$Pr(D) = \mathbf{while} \ \Phi \ \mathbf{do} \ A. \quad (2.4.6)$$

Proof It follows from the definitions of A and Φ that:

$$\begin{aligned} &\text{For any } n \geq 1, \text{ there exist states } u_0, \dots, u_n \text{ such that } u_0 A u_1 \dots u_{n-1} A u_n \\ &\text{and } u_0, \dots, u_n \in \Phi. \end{aligned} \quad (2.4.7)$$

We apply a *reductio ad absurdum* argument. Suppose that (2.4.6) holds for some algorithm D . It is clear that $\mathbf{A}(D)$ is a compound action over the one-element alphabet $\{A\}$. Let $A \dots A$ be a word of $\mathbf{A}(D)$ of length n . As (2.4.6) holds, we have that for *any* string (u_0, \dots, u_n) of states, $u_0 A u_1 \dots u_{n-1} A u_n$ implies that $u_0, \dots, u_{n-1} \in \Phi$ and $u_n \notin \Phi$. But this contradicts (2.4.7). \square

The program ‘**while** Φ **do** A ’ defines a situational system \mathbf{M}^s over \mathbf{M} . (The system \mathbf{M} need not be normal.) The set V of labels consists of two elements, $V := \{a, \omega\}$, where ω is the terminal label and the label a is identified with the name of the action A .

$S := W \times V$ is the set of possible situations. A pair $(u, a) \in S$ has the following interpretation: u is the current state of the system and the action A is performed. Analogously, (u, ω) is read: u is the current state of the system and no action is performed in u . $S_0 := \Phi \times \{a\}$ is the set of initial situations, while the members of $S := (W \setminus \Phi) \times \{\omega\}$ are terminal situations. The relation Tr of the transition between situations is defined as follows:

$Tr(s, t)$ if and only if **either** $s = (u, a), t = (w, a), u \in \Phi, w \in \Phi$ and $u A w$ **or** $s = (u, a), t = (w, \omega), u \in \Phi, w \notin \Phi$ and $u A w$.

Let $f : W \times V \rightarrow W$ be the projection from S onto W . The six-tuple

$$\mathbf{M}^s := (W, R, \{A\}, S, Tr, f)$$

is called the *situational action system associated with the program ‘while Φ do A ’ on M .*

Finite runs of situations are defined in the usual way; they are sequences of situations

$$(s_0, \dots, s_n) \quad (2.4.8)$$

such that $s_0 \in S_0$ and $Tr(s_i, s_{i+1})$ for all $i \leq n - 1$. The run (2.4.8) is terminated if $s_n \in S_\omega$.

Not every run (2.4.8) can be prolonged to a terminated run. It may happen that for some $s_n = (u_n, a)$ there are no states w such that $u_n A, R w$. The program gets jammed in the state u_n .

2.5 The Ideal Agent

A theory of action should distinguish between praxeological and epistemic aspects of action. (This issue is also discussed in the last chapter of this book.) In this work, we will not discuss at length such praxeological elements of action as the cost of action, available means, money, etc. These ‘hard’ praxeological factors are *en bloc* represented by the relation R of direct transition between states. These problems are closely connected with practical reasoning and with the studies on deliberation in particular. As Segerberg (1985, p. 195) points out: “It is well-known how crucially deliberation depends on the faculties and outlook of the agent, but it depends not only on his knowledge, beliefs and values, but also how he perceives the situation, what ways he has of acquiring new situation, his imagination, what action plans he has available and what decision he employs.”

The very nature of agency and the problem of the epistemic status of agents in particular are the most difficult issues that a theory of action has to resolve. In its lexical meaning, the theory typically describes an action as behaviour caused by an *agent* in a particular *situation*. The agent’s *desires* and *beliefs* (e.g. my wanting a glass of water and believing the clear liquid in the cup in front of me is water) lead to bodily behavior (e.g. reaching over for the glass). According to Davidson (1963), the desire and belief jointly cause the action. But Bratman (1999) has raised problems for such a view and has argued that one should take the concept of intention as basic and as not analyzable into beliefs and desires (the Belief-Desire-Intention model of action).⁵

⁵An interesting example which might be discussed with the problem of agency is that of micromanagement where pathological relations hold between individual and collective agents. Roughly, micromanagement is defined as “attention to small details in management: control of a person or a situation by paying extreme attention to small details”. The notion of micromanagement in the wider sense describes social situations where one person (the micromanager) has an intense degree of control and influence over the members of a group. Often, this excessive obsession with the most minute of details causes a direct management failure in the ability to focus on the major details.

We make in this section the assumption that the agents operating a situational action system

$$M^s = (W, R, \mathcal{A}, S, Tr, f)$$

are omniscient in the generous sense. To be more specific, suppose a_1, \dots, a_n is a list of agents operating the system M^s . It is assumed that :

- (1) each agent a_i ($i = 1, \dots, n$) knows all possible states of the system, i.e., the elements of W , and all possible direct transitions between states, i.e., the elements of R ;
- (2) each agent a_i ($i = 1, \dots, n$) knows the elements of S , Tr , and f ;
- (3) each agent a_i ($i = 1, \dots, n$) knows, for every $A \in \mathcal{A}$, not what will in fact happen, but what can happen if the action A is performed, i.e., for every $u \in W$ and every $A \in \mathcal{A}$, the agent a_i knows the elements of the set $\{w \in W : u A w\}$;

(Postulates (1)–(3) thus ascertain that the internal structure of the system (W, R, \mathcal{A}) , as well as the situational envelope are completely transparent to the agents a_1, \dots, a_n .)

- (4) each agent a_i always knows the current situation of the system; in particular the agent knows the current state of the system;

(The postulates (1)–(4), thus, enable each agent a_i to state if a quite arbitrary action $A \in \mathcal{A}$ is performable (or totally performable) in the current situation.)

- (5) if a_i is the agent of an action $A \in \mathcal{A}$ in a situation s and A is performable in s , then a_i knows what *will* happen, if he performs A in s .

The principle (5) thus says that the actions performed by the agent are completely subject to his free will. It is not assumed that to each atomic action A only one agent is assigned, where he is the same in all possible situations. The agents of A may vary from one situation to another. We state however that every action is carried out by at the most only one agent in a given situation (the agent who actually performs the action). Even if the agents are ideal, if the system is operated by more than one agent, they may not foresee the evolution of the system. The agent may not know *who* is going to perform an action in the next move, *what* action the other agent is going to perform and *how* the action will be performed. (The first case is obviously excluded in a game of chess but the other two are not.) It is conceivable that in some situation s there may be many agents who are allowed to perform certain actions. For example, agent a_1 could perform action A_1 or A_2 in s and agent a_2 could perform A_3 , A_4 or A_5 in the same situation s . This may happen in some situational action systems. The relation Tr of direct transition between situations allows for many possible courses of events commencing with s . Whether agent a_1 or agent a_2 is going to make a move in s may be a random matter; the situation s and the relation Tr need not determine this accurately.

Free will, according to its lexical meaning, is the agent's power of choosing and guiding his actions (subject to limitation of the physical world, social environment and inherited characteristics). Free will enables the agent, in a particular situation in which he can act, both to choose an action he would like to perform (if there is more

than one possible action he can perform in this situation) and to be completely free in his way of accomplishing it. In the framework of the formalism accepted here, the fact that an action A is performed according to the agent's free will means only that the moment the action is undertaken (a situation s), the system may pass to any of the states from the set $\{f(s') : s \text{ Tr } s' \text{ and } f(s) A f(s')\}$. The free agent is not hindered from transferring the system to any of the states belonging to the above set (provided that the set is non-empty); his practical decision in this area is the result of a particular value, and so something that in the given situation the agent considers proper.

The notion of an agent should be widely understood; he can be a real man or a group of people (a collective agent); or a man cooperating with a machine or a robot. A detailed analysis of the concept of agent is not necessary for the present investigation. We want, however, this notion to be devoid of any anthropocentric connotation. (This view is not inconsistent with the above postulates (1)–(5) if the meaning of the phrase 'the agent knows' is properly understood.)

Realistic theories of action should be based on more restrictive postulates than those defining the epistemic and praxeological status of ideal agents. For instance, in the game of bridge the players never know the current distribution of cards—the postulate (4) is not true in this case. The agents may not know the members of W , as well as the pairs of states which are in the relation R . The agent of an action may not know all the possible performances of this action. The knowledge of S and Tr may also be fragmentary.

In Part II an approach to action theory is outlined which takes into account the above limitations and assumes that agents' actions are determined and guided by systems of norms. Norms determine what actions in given conditions are permitted and which are not. Agents' actions are then controlled by the norms. The relationship between this concept of action and deontic logic is also discussed there.

A theory of action is the meeting ground of many interests, and therefore many approaches. The formal approach attempts to answer the following question: how should actions be described in terms of set-theoretic entities? Any answer to this question requires, of course, some ontological presuppositions. In this book we propose, in a tentative spirit, a relatively shallow analysis of this subject. First, we assume that there exist states of affairs and processes. Furthermore, we claim that these simple ontological assumptions can be rendered adequately into the language of set theory. This leads to the definition of a discrete system. The set W of states is the mathematical counterpart of the totality of states of affairs, and R , the relation of direct transitions, represents possible processes (i.e., transitions) between states of affairs. Apart from these concepts the category of situations is singled out. It is an open problem whether situations can be faithfully represented by set theory. We do not discuss this issue here because we would have to begin with a general account of what a situation is. Instead, we aim at showing that in some simple cases the theory of sets provides a good framework for the concepts of action and situation.

2.6 Games as Action Systems

In this section, as an illustration of the theory expounded above, a class of idealized, infinite mathematical games will be presented. These games will be modeled as simple, situational action systems.

We recall that the symbols \mathbb{N} and ω interchangeably denote the set of natural numbers with zero. In set theory the set of natural numbers is *defined* as the least non-empty limit ordinal. The elements of ω are also called *finite ordinals*. Thus $0 = \emptyset$ is the least natural number and $n + 1 = \{0, 1, \dots, n\}$, for all $n \in \omega$, according to the standard set-theoretic definition of natural numbers.

If $n \in \omega$, then ${}^n A$ is the set of functions from n to A . Note that ${}^0 A = \{\emptyset\}$. The empty function \emptyset is also denoted by $\mathbf{0}$. The length of the sequence $\mathbf{0}$ is 0. $A = (W, \Sigma, \Gamma, \delta, w_0, \alpha, F)$.

We then define

$${}^{<\omega} A := \bigcup_{n \in \omega} {}^n A$$

If $p \in {}^n A$, we also write $p = (a_0, a_1, \dots, a_{n-1})$, or simply $p = a_0 a_1 \dots a_{n-1}$, where $a_k := p(k)$ for $k = 0, 1, \dots, n-1$. $|p|$ is the length of p . (Formally, $|p|$ coincides with the domain of p .)

We consider the scheme of an infinite game between two players, denoted respectively by I (the first player) and II (the second player).

Let A be a fixed non-empty set, e.g., the set of natural numbers.

- (i) The players alternately choose elements of A . Each choice is a *move* of the game; and each player before making each of his moves is privy to all the previous moves.
- (ii) Infinitely many moves are to be performed;
- (iii) Player I starts the game.

The resulting infinite sequence of elements of A

$$\mathbf{a} = a_0, a_1, \dots, a_n, a_{n+1}, \dots$$

is called a *play* of the game. The even numbered elements of \mathbf{a} , viz., $a_{2n}, n = 0, 1, \dots$ are established by player I while the odd-numbered elements of \mathbf{a} , $a_{2n+1}, n = 0, 1, \dots$ by player II.

Each set Z of infinite sequences of elements of A , $Z \subseteq {}^\omega A$ determines a game $G(Z)$. Z is called the *payoff* of the game $G(Z)$.

A play \mathbf{a} of the game $G(Z)$ is won by player I if it is the case that $\mathbf{a} \in Z$. If $\mathbf{a} \notin Z$ —the play \mathbf{a} is won by II. Thus the game $G(Z)$ is decidable in the sense that for each play \mathbf{a} either I or II wins \mathbf{a} (there are no draws).

Every initial segment $\mathbf{a}[n = a_0, a_1, \dots, a_{n-1}]$ of a possible play \mathbf{a} is called a *partial play*. ${}^{<\omega} A$ is the set of all partial plays.

Let π_n be the projection of ${}^\omega A$ onto the n th axis, that is, $\pi_n(\mathbf{a})$ is the n th term of the sequence \mathbf{a} for all $\mathbf{a} \in {}^\omega A, n \in \omega$.

Suppose that $\pi_n(Z)$ is a proper subset of A for some odd n . Then II wins the game irrespective of the first n moves a_0, a_1, \dots, a_{n-1} made by the players. In the n th step player II selects an arbitrary element $a_n \in A \setminus \pi_n(Z)$. Then for *any* play \mathbf{a} being a continuation of the initial segment a_0, a_1, \dots, a_n it is the case that $\mathbf{a} \in Z$. (For if $\mathbf{a} \in Z$, then $\pi_m(\mathbf{a}) \in \pi_m(Z)$, for all $m \in \omega$, which is excluded.) Thus the play \mathbf{a} is won by II. Intuitively, the winning strategy for II is any function S defined on all possible positions of even length such that $S(p) \in A \setminus \pi_n(Z)$ for any position $p = a_0, a_1, \dots, a_{n-1}$ of length n with n defined as above.

We shall express the above remarks in a more formal setting as follows. We shall limit ourselves to deterministic games, i.e., the ones in for each player in any position there is only one option open to him according to which he continues the game, which means that the strategy available to him is a function.

Definition 2.6.1 Let A be a non-empty set.

1. The set $W := {}^{<\omega}A$ is called the set of *possible positions* or, in the terminology of action theory, it is called the set of *possible states*.
2. Any function $S : W \rightarrow A$ is called a *deterministic strategy* in A .
3. A play in A is any mapping $P : \omega \rightarrow A$. □

Lemma 2.6.2 For any deterministic strategy S in A there exists a unique play P in A such that $P(n) = S(P \upharpoonright n)$ for all $n \in \omega$.

(Note that $P(0) = S(P \upharpoonright 0) = S(P \upharpoonright \emptyset) = S(\mathbf{0})$.)

Proof This is an application of the known Principle of Definability by Arithmetic Recursion. □

Putting $P(n) = a_n$ for all $n \in \omega$, the above lemma says that $a_0 = S(\mathbf{0})$ and $a_n = S(a_0, a_1, \dots, a_{n-1})$ for all n . P is called the play *played according to the strategy* S .

From the perspective of action theory, each strategy S defines a binary relation R_S on the set $W = {}^{<\omega}A$ of states—possible positions. R_S is the relation of direct transition between possible situations. Intuitively, for $u, w \in W$, the fact $u R_S w$ means that w results from u by an application of the strategy S to u and w is the position obtained from the sequence u by adjoining the element $S(u)$ at the end of u . Formally, $u R_S w \Leftrightarrow_{df} w = u, S(u)$.

Since the strategy S is deterministic, R_S is a function from W to W . It follows that (W, R_S) is a deterministic discrete system.

We are interested in a mathematical analysis of two-person games. To represent such games in terms of action systems, we define the operation of the merger of any two strategies.

The definition of the merger of two strategies takes into account the context-sensitive aspect of each play, represented by the situation whose component is not only the finite string of moves made by one player up to some phase of the play, but also the respective moves of his opponent. Formally, this situation is represented by the following mathematical construction. The players I and II have certain strategies

S_1 and S_2 at their disposal. The strategy S_1 defines the elementary action A_I of the player I on the set of states W . A_I is defined as above as the transition relations between positions imposed by the strategy S_1 on the set of positions. Thus, for any positions $p, q \in W$,

$$p A_I q \Leftrightarrow_{df} q = p, S_1(p).$$

Analogously one defines the action of the other player: for $p, q \in W$,

$$p A_{II} q \Leftrightarrow_{df} q = p, S_2(p).$$

It is clear that the relations A_I and A_{II} are both unary functions from W to W .

However, in two-person games the relation of direct transition between states is defined in a more involved way. According to the rules, each play is represented by a sequence

$$a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1}, \dots$$

The moves made by I are influenced not only by his earlier choices a_0, a_1, \dots, a_n but also by the moves made by the second player, represented by the sequence b_0, b_1, \dots, b_n . The choice of a_{n+1} is motivated not only by the moves a_0, a_1, \dots, a_n but also by b_0, b_1, \dots, b_n . In other words, in the context-sensitive strategy available for I the consecutive moves performed by him are determined *not* only by a_0, a_1, \dots, a_n but by the sequence $a_0, b_0, a_1, b_1, \dots, a_n, b_n$. Analogously, in the context-sensitive strategy of II the $(n + 1)$ -th move of II is determined *not* by b_0, b_1, \dots, b_n , but by the sequence $a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1}$. This sequence determines the next move b_{n+1} .

To define properly the relation R of direct transition between states of W we first define the notion of the merger of two strategies.

Definition 2.6.3 Let S_1 and S_2 be two strategies in A . The *merger* of S_1 and S_2 is the unique strategy $S_1 \oplus S_2$ in A such that for every $p \in {}^{<\omega}A$,

$$(S_1 \oplus S_2)(p) := \begin{cases} S_1(p) & \text{if } |p| \text{ is even} \\ S_2(p) & \text{if } |p| \text{ is odd.} \end{cases} \quad \square$$

If $|p| = 0$, i.e., p is empty, $p = \mathbf{0}$, then

$$(S_1 \oplus S_2)(p) = S_1(\mathbf{0}),$$

and if $|p| = 1$, then

$$(S_1 \oplus S_2)(p) = S_2(p).$$

The above definition says that if $p = (a_0, b_0, a_1, b_1, \dots, a_n, b_n)$, then

$$(S_1 \oplus S_2)(p) = S_1((a_0, b_0, a_1, b_1, \dots, a_n, b_n))$$

and if $p = (a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1})$, then

$$(S_1 \oplus S_2)(p) = S_2((a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1})).$$

In particular, $(S_1 \oplus S_2)(\mathbf{0}) = S_1(\mathbf{0}) = a_0$ and $(S_1 \oplus S_2)(\langle a_0 \rangle) = S_2(\langle a_0 \rangle) = b_0$.

Applying Lemma 2.6.2 to the strategy $S_1 \oplus S_2$ we get

Corollary 2.6.4 *For any strategies S_1, S_2 there exists a unique play P in A such that $P(n) = (S_1 \oplus S_2)(P \upharpoonright n)$ for all $n \in \omega$. \square*

It follows from the above corollary and the definition of $S_1 \oplus S_2$ that

$$P(2n) = S_1(P \upharpoonright 2n) \quad \text{and} \quad P(2n+1) = S_2(P \upharpoonright 2n+1)$$

for all $n \in \omega$.

The relation R of direct transition on W is defined as the transition relation determined by the merger $S_1 \oplus S_2$, that is, $R = R_{S_1 \oplus S_2}$. Thus, for any positions $p, q \in W$,

$$p R q \iff_{df} q = p, (S_1 \oplus S_2)(p).$$

We thus arrive at the definition of an elementary action system $\mathbf{M} = (W, R, \{A_I, A_{II}\})$ endowed with two atomic actions. \mathbf{M} is called the action system *corresponding to the two-person game with strategies S_1 and S_2* .

Taking into account the typology of action systems adopted in Chap. 1, Definition 1.2.3, we have:

Proposition 2.6.5 *The system $\mathbf{M} = (W, R, \{A_I, A_{II}\})$ is deterministic and complete.*

Proof Immediate. \square

The system \mathbf{M} is not normal, because neither $A_I \subseteq R$ nor $A_{II} \subseteq R$. It may also happen that the intersection $A_I \cap A_{II}$ is non-empty. Therefore the system need not be separative. One may also easily check that the system is irreversible. This follows from the fact that $p R q$ implies that $|p| < |q|$, for any positions p and q .

The system \mathbf{M} has another property: the set of states W is ordered by the relation \leq , where for $p, q \in W$, $p \leq q$ means that $p = q$ or p is a proper prefix of q . \mathbf{M} is therefore an ordered action system. (Ordered systems are defined in Chap. 3.)

The system \mathbf{M} is operated by two agents: I and II. Though each of them is able to foresee the course of states according to his own strategy, the situation changes when they play together. As a rule each player does not know his opponent's strategy and therefore is unable to predict consecutive positions that will be taken during the game. In other words, each of them knows the action he will perform, because each knows his own strategy. However, they individually may not know the relation of direct transition R because they do not necessarily know the merger $S_1 \oplus S_2$. Consequently, the players need not know the unique play P determined by $S_1 \oplus S_2$.

But during the course of the game, the consecutive positions $P(n)$ of the game are revealed to both of them. Summing up, according to the remarks from the preceding section, each of the agents I and II is an ideal one. However, the rules of the game exclude the possibility that each of them knows the strategy available to his opponent. In this sense they are *not* omniscient.

(The problem of agency in games is investigated by many thinkers—see e.g. van Benthem and Liu (2004). Various “playing with information” games, public announcements and, more generally, verbal actions are not analysed here.)

The above definition of a deterministic strategy is strong from the epistemic viewpoint. So as to continue the game, the strategy S refers to *all* earlier positions occurring in the course of the play determined by S . In other words, to make a successive move in a given position p , the agent *knows* the sequence p ; in particular he knows all the prefixes of p , and therefore he knows all earlier positions in the play determined by S . Mathematically, this means that the strategy of an agent is a function defined on the set of *all* possible positions $W = {}^{<\omega}A$. In practice, the agent is not omniscient and he is able to remember at most two or three earlier moves. It is therefore tempting to broaden the definition of a deterministic strategy by allowing functions S which are defined not on ${}^{<\omega}A$ but merely on a subset of the form nA for some (not very large) n . The organization of the game is then different. For example, for $n = 2$, the play determined by such a new strategy $S : {}^2A \rightarrow A$ is defined by declaring in advance the first two moves a_0 and a_1 . Then successive moves are determined by S ; the function S takes into account merely the last two moves and determines the subsequent move. We may call such functions $S : {}^nA \rightarrow A$ *Fibonacci-style strategies*, by analogy with Fibonacci sequences.

Mathematically, such a broadening of the definition of a strategy does not change the scope of game theory because, as one can easily check, for every Fibonacci-like strategy S there is a “standard” strategy S' such that both S and S' determine the same play. However, this distinction between the standard and Fibonacci-style strategies may turn out to be relevant in various epistemic contexts in which agents with restricted memory resources compete.

The game is not fully encoded in the elementary action system \mathbf{M} . We must also take into account rules (i)–(iii) above that organize the game. They constitute the situational envelope of \mathbf{M} . More specifically, by a *possible situation* we shall understand any element of the set

$$S := W \times \{I, II\}.$$

Thus, S is a set of labeled situations with labels being the elements of $\{I, II\}$. The pair $(\mathbf{0}, I)$ is called the *initial situation*. (The symbol “ S ” also ranges over strategies, but such a double use of this letter should not lead to confusion.)

The relation of transition Tr between situation is defined in accordance with the above requirements (i)–(iii). Thus, if $s_1, s_2 \in S$ and $s_1 = (p, a)$, $s_2 = (q, b)$, then

$s_1 Tr s_2 \Leftrightarrow_{df}$ either p is a position of even length, $a = I, b = II$, and $A_I(p, q)$
or p is a position of odd length, $a = II, b = I$, and $A_{II}(p, q)$.

Intuitively, if p is a position whose length is an even number, then in the situation (p, I) player I performs the action A_I and the situation (p, I) turns into (q, II) , where $q = p, (S_1 \oplus S_2)(p) (= p, S_1(p))$. Analogously, if p is a position whose length is an odd number, then in the situation (p, II) player II performs the action A_{II} and the situation (p, II) turns into (q, I) , where $q = p, (S_1 \oplus S_2)(p) (= p, S_2(p))$.

$f : S \rightarrow W$ is defined as the projection onto the first axis: if $s = (p, a)$, then $f(s) := p$, for all $s \in S$.

The relation R of direct transition between states of the above action system M is compatible with Tr ; that is, for every pair $s_1, s_2 \in S$ of situations, if $s_1 Tr s_2$ then $f(s_1) R f(s_2)$. Indeed, assume $s_1 Tr s_2$. If $s_1 = (p, a)$, and the length of p is even, then $a = I, s_2 = (q, II)$, where $q = p, (S_1 \oplus S_2)(p)$. Hence $p R q$, which means that $f(s_1) R f(s_2)$. The case when $s_1 = (p, a)$, and the length of p is odd, is analogously checked.

It follows from the above remarks that

$$M^S = (W, R, \{A_I, A_{II}\}, S, Tr, f)$$

is a *situational action system*. M^S is the system *associated with* the above game. M^S , together with the initial situation $(0, I)$ being distinguished, faithfully encodes all aspects of the game with one exception, however; the target of the game has not been revealed thus far. In other words, to have the game fully determined, the payoff set must be defined. Below we present some remarks on winning strategies.

Let Z be a subset of ${}^\omega A$ (i.e., Z is a set of infinite functions from ω to A). Let, as above, $S_1 \oplus S_2$ be the merge of strategies S_1 and S_2 and let $P = a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1}, \dots$ be the play determined by $S_1 \oplus S_2$.

Player I *wins the play P for Z* if the infinite sequence P belongs to Z . Player II *wins the play P for Z* if P does *not* belong to Z .

S_1 is a *winning strategy* in Z (for player I) if for *every* strategy S_2 chosen by player II it is the case that the resulting play $P = a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1}, \dots$ determined by $S_1 \oplus S_2$ belongs to Z .

S_2 is a *winning strategy* in Z (for the second player) if for every strategy S_1 chosen by player I it is the case that the play P determined by $S_1 \oplus S_2$ is *not* in Z .

Examples illustrating the above notions below are taken from Błaszczyk and Turek (2007), Sect. 17.3.

Example We assume $A = \omega$. Any two-person play consists in selection of a sequence $a_0, b_0, a_1, b_1, \dots, a_n, b_n, \dots$ of natural numbers.

Let $Z = {}^\omega \omega$. Then every strategy of I is a winning strategy for I, because for all n , each selection of numbers a_0, a_1, \dots, a_n by I guarantees that I wins every play (irrespective of the corresponding moves b_0, b_1, \dots, b_n made by II).

If $Z = \emptyset$, then every strategy of II is a winning strategy for II.

Let us consider a less trivial case: Z is a *countable* subset of ${}^\omega\omega$, i.e., Z is a countable set of infinite sequences z_n , $n \in \omega$. We define the following strategy for the second player II:

$$\begin{aligned} S_2(\mathbf{0}) &:= z_0(0) + 1, \\ S_2(c_0, c_1, \dots, c_n) &:= z_{n+1}(2(n+1)) + 1, \end{aligned}$$

for all $n \in \omega$ and all $(c_0, c_1, \dots, c_n) \in {}^{n+1}\omega$.

S_2 is called the *diagonal strategy*. (Here $z_n = (z_n(0), z_n(1), \dots)$ and $z_n(2(n+1))$ is the value of z_n for $2(n+1)$).

Proposition 2.6.6 *The diagonal strategy is a winning strategy for the second player.*

Proof Let S_1 be an arbitrary strategy of player I. The play P defined by $S_1 \oplus S_2$ is represented by an infinite sequence

$$a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1}, b_{n+1}, \dots$$

where $b_0 = z_0(0) + 1 = a_0 + 1$ and $b_n = z_n(2n) + 1$ for $n = 1, 2, \dots$

It suffices to notice that $a_0, b_0, a_1, b_1, \dots, a_n, b_n, a_{n+1}, \dots$ is not in Z . Indeed, the above sequence can be written as

$$z_0(0) + 1, b_0, z_1(2) + 1, b_1, z_2(4) + 1, b_2, z_3(6) + 1, b_3, \dots, z_n(2n) + 1, b_n, \dots \quad (*)$$

The above sequence differs from the sequences z_n , $n \in \omega$, because if $(*)$ were equal to some z_n , we would have that the element $z_n(2n)$ of z_n would be equal to the element of $(*)$ with index $2n$; i.e., it would be equal to $z_n(2n) + 1$, which is excluded. \square

A set $Z \subseteq {}^\omega\omega$ is said to be *determined* if in the two-player game $G(Z)$ one of the players, I or II, has a winning strategy. According to the above proposition, every countable set $Z \subseteq {}^\omega\omega$ is determined. Determined sets, when taken collectively, do not show regular algebraic properties. For example, one cannot prove that they form a Boolean algebra. In 1976 Martin proved the following theorem:

Theorem *Assume the set theory ZF of Zermelo Fraenkel. Every Borel subset of \mathbb{R} is determined.* \square

The situation changes if one introduces the following set-theoretic axiom:

Axiom of Determinacy (AD) *Every subset of ${}^\omega\omega$ is determined.* \square

It is easy to see that AD is equivalent to the following statement:

For every countably infinite set A , every subset of ${}^\omega A$ is determined. \square

Other games, as e.g. Banach–Mazur games or parity games, can be also represented in a similar manner as situational action systems. (Parity games are played on a coloured directed graphs. These games are history-free determined. This means that if a player has a winning strategy then he has a winning strategy that depends only on the current position in the graph, and not on earlier positions.)

The games presented in this chapter are all qualified as games with perfect information, which means that they are played by ideal agents in the sense expounded in Sect. 2.5. Card games, where each player's cards are hidden from other players, are examples of games of imperfect information.

2.7 Cellular Automata

Cellular automata form a class of situational action systems. Here, we shall only outline the functioning of one-dimensional cellular automata, the latter being a particular case of the general notion. Suppose we are given an infinitely long tape (in both directions) divided into cells. From the mathematical viewpoint, the tape is identified with the set of integers \mathbb{Z} . Cells are, therefore, identified with integers. Each cell has two immediate neighbors, viz. the left and the right adjacent cells. There are two possible states for each cell, labeled 0 and 1. The state of a given cell c , together with the states of its immediate neighbors, fully determines a *configuration* or *possible situation* of the cell c . There are $8 = 2^3$ possible situations of each cell. Possible configurations are written in the following order

$$\underline{1}11, \quad 1\underline{1}0, \quad 10\underline{1}, \quad 100, \quad 0\underline{1}1, \quad 01\underline{0}, \quad 00\underline{1}, \quad 000,$$

where the underlined digit indicates the current state of a particular cell c . The rules defining a cellular one-dimensional automaton specify possible transitions for each such situation. These rules are identified with functions assigning to each situation a state from 0, 1. There are, therefore, $256 = 2^8$ such rules. Each rule individually defines a separate cellular automaton and its action. For instance,

possible situations	$\underline{1}11$	$1\underline{1}0$	$10\underline{1}$	100	$0\underline{1}1$	$01\underline{0}$	$00\underline{1}$	000
new state for center cell	$\underline{0}$	$\underline{1}$	$\underline{1}$	$\underline{0}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{0}$

is a rule which assigns to each possible configuration of the center cell a new state of this cell.

Stephen Wolfram proposed a scheme, known as the *Wolfram scheme*, to assign to each rule a number from 0 to 255. This scheme has become standard (see e.g. Wolfram 2002).

As each rule is fully determined by the sequence forming new states for the center cell, this sequence is encoded first as the binary representation of an integer and then

by its decimal representation. This number is taken to be the rule number of the automaton. For example, the above rule is encoded by 01101110_2 which is 110 in base 10. This is, therefore, Wolfram's rule 110.

In the above infinite model, each cell c has two immediate neighbors: $c - 1$ (the left one) and $c + 1$ (the right one). Each mapping $u : Z \rightarrow \{0, 1\}$ defines the *global state* of an automaton. (One may think of an automaton as a system of conjugate cells subject to a given transformation rule.) Each rule r determines transformations of global states of the automaton. Suppose that the following sequence of states (the second row) have been assigned to the displayed finite segment of the sequence of integers (cells):

$$\begin{array}{cccccccccccccccc} \dots, & -3, & -2, & -1, & 0, & 1, & 2, & 3, & 4, & 5, & 6, & 7, & \dots \\ & & 1, & 1, & 0, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 0, & \dots \end{array}$$

The rule 110 turns the above states assigned to $-2, -1, 0, 1, 2, 3, 4, 5, 6$ to the following sequence

$$\begin{array}{cccccccccccccccc} \dots, & -3, & -2, & -1, & 0, & 1, & 2, & 3, & 4, & 5, & 6, & 7, & \dots \\ \dots & & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 0, & 0, & & \dots \end{array}$$

(The new states of the cells -3 and 7 are not displayed because the initial states assigned to the left neighbour of -3 and the right neighbour of 7 have not been disclosed.) Further iterations of the rule produce new global states.

One may also define an elementary cellular automaton with a finite number of cells, say $1, 2, \dots, n$ by declaring that 1 is the right neighbour of n and n is the left neighbour of 1 . (Of course, $n - 1$ is the left neighbour of n and 2 is the right neighbour of 1 .) It is also natural to take all the n complex roots of degree n of 1 as cells).

Each Wolfram rule r may be regarded as a certain context-dependent substitution. In each sequence of zeros and ones (of the type of integers) it replaces each consecutive element of the sequence by a new one by looking at the left and right immediate neighbours of this element in the sequence and then applying the adopted rule. Wolfram rules act as rules with limited memory. In order to make a successive substitutions in a 0-1 sequence, the rule merely assumes the knowledge of three consecutive elements of the sequence. One may complicate Wolfram rules in various ways such as by taking into account more distant neighbours.

From the algebraic viewpoint, each Wolfram rule may be treated as a ternary operation of the two-element (Boolean) algebra $\{0, 1\}$. In turn, each (infinite) elementary cellular automaton is regarded as an action of this algebra on the uncountable set $\{0, 1\}^Z$ in the way which was explained above.

It is worth noticing that the automaton determined by the rule 110 is capable of universal computation (Cook 2004) and is therefore equivalent to a universal Turing machine.

Suppose a Wolfram rule r has been selected. Each individual cell c may be then regarded as a simple two-state situational action system $\mathbf{M}_c = (W, R, \mathcal{A}, S, Tr, f)$ which we shall describe. Here $W = \{0, 1\}$ is the set of states. Possible situations s

are labeled states, $s = (u; a)$, where u is the state of s and a the label of s . Labels are ordered pairs of states, $a = (u_L, u_R)$, where u_L is a state of the left neighbour of c and u_R is a state of the right neighbour of c . Possible situations may therefore be identified with configurations of c . Thus, if e.g., $0\bar{1}1$ is a configuration of c , then $s = (1; 01)$ is the possible (labeled) situation corresponding to $0\bar{1}1$: 1 is the current state of c and 0 and 1 are the current states of the left and the right neighbour of c respectively. There are, of course, $4 = 2^2$ labels and 8 possible situations of M_c . $Label = \{00, 01, 10, 11\}$ is the set of labels and S is the set of possible situations.

The work of M_c is not, however, organized in the same way as in the case of the situational action systems corresponding to iterative or pushdown algorithms.

Each label $a \in Label$ gives rise to an atomic deterministic action A_a on W . For suppose $u \in W$ and $a = (u_L, u_R)$. Then, (u_L, u, u_R) is a configuration. The rule r assigns to this configuration a unique state w . We then put: $A_a(u) := w$. (The correspondence $a \rightarrow A_a, a \in Label$, need not be one-to-one. It is dependent on the choice of a Wolfram rule.) We, therefore, have at most 4 atomic actions, all being unary functions). We put:

$$\mathcal{A} := \{A_a : a \in Label\}.$$

Rule r determines transitions between states. As each state u is nested in some configuration (there are four such configurations for u), the rule r defines at the most four possible immediates successors of u . Thus,

$u R w$ if and only if for some configuration (u_L, u, u_R) it is the case that r assigns the state w to (u_L, u, u_R) .

Equivalently,

$u R w$ if and only if $w = A_a(u)$ for some $a \in Label$.

It follows that (W, R, \mathcal{A}) is a deterministic and normal action system. (But R need not be a function.)

The function f is a projection assigning to each possible situation $s = (u; a)$ the state u .

It remains to define the transition relation Tr between possible situations. How do situations evolve? Suppose $(u_L, \underline{u}, u_R)$ is a current configuration of the cell c , where \underline{u} is the current state of c . Rule r merely unambiguously determines the next state of c . The clue is that the configuration being the successor of $(u_L, \underline{u}, u_R)$ depends not only on the current states u_L and u_R of the left and right neighbours of c , i.e. the current states of the cells $c - 1$ and $c + 1$, but it is also determined by the current states of farther neighbours, these being the cells $c - 2$ and $c + 2$. Thus, in order to predict at least on move in a trajectory of situations, one has to know a quintuple of states $(u_{LL}, u_L, \underline{u}, u_R, u_{RR})$ with u_{LL} being the current state of $c - 2$, the left neighbour of $c - 1$, and u_{RR} being the current state of $c + 2$, the right neighbour of $c + 1$. Rule r then assigns to the configurations $(u_{LL}, u_L, \underline{u})$ and $(\underline{u}, u_R, u_{RR})$ certain states, say w_1, w_2 , respectively, and it assigns to the configuration $(u_L, \underline{u}, u_R)$ a state

w . Taking this into account, we see that the configuration $(w_1, \underline{w}, w_2)$ is a successor of $(u_L, \underline{u}, u_R)$.

The above remarks enable one to define the relation Tr of direct transitions between possible situations. Note that generally Tr need *not* be a function because the configuration $(w_1, \underline{w}, w_2)$ depends not only on $(\underline{u}, u_R, u_{RR})$ but also on states u_{LL} and u_{RR} . On the other hand, any situation s has at the most four intermediate successors, according to Tr . Each such successor is determined by states of the cells $c - 2$ and $c + 2$.

The relation R of direct transition between states of the above action system (W, R, \mathcal{A}) is compatible with Tr , i.e., for every pair $s_1, s_2 \in S$ of situations, it is the case that if $s_1 Tr s_2$ then $f(s_1) R f(s_2)$. It follows that $M_c = (W, R, \mathcal{A}, S, Tr, f)$ is a well-defined situational action system.

The action systems M_c are identical, for all cells c . The functioning of each system M_c is a kind of a local feedback among M_c and the surrounding systems $M_{c-2}, M_{c-1}, M_{c+1}$ and M_{c+2} , for all cells c . The structure of this feedback is determined by a definite Wolfram rule r .

The evolution of the system of automata $M_c, c \in Z$, (Z —the set of integers) is initialized by declaring a *global initial state* (a *global configuration*) of the system. This is done by selecting a mapping $u : Z \rightarrow \{0, 1\}$. (For example, one may assume that u assigns the zero state to each cell.) The system is set in motion by the action of the Wolfram rule r for the cell $u(0)$ and its neighbours. Then the evolution of the system proceeds spontaneously.

The theory of one-dimensional cellular automata is a particular case of the more general theory of finitely many dimensional automata. We shall restrict our attention to the two-dimensional case because it gives a sufficient insight into the basic intuitions underlying the general definition. A standard way of depicting a two-dimensional cellular automaton is with an infinite sheet of graph paper and a set of rules for the cells to follow. Each square is called a ‘cell’ and each cell has two possible states, black and white. The ‘neighbours’ of a cell are the 8 squares touching it. Mathematically, each cell (or square) is identified with a pair of integers (a, b) . Any pair of the form (c, d) with $c = a \pm 1$ or $d = b \pm 1$ is therefore a neighbour of (a, b) . (All these pairs form the *Moore neighbourhood* of (a, b) . One may also take a non-empty subset of the defined set of pairs (c, d) to form a narrower neighbourhood of (a, b) ; in a particular case one gets the *von Neumann neighbourhood* of (a, b) by taking the pairs (c, d) with $c = a \pm 1$ and $d = b \pm 1$.) For such a cell and its neighbours, there are 512 ($= 2^9$) possible patterns. For each of the 512 possible patterns, the rule table states whether the center cell will be black or white on the next time interval.

It is usually assumed that every cell in the universe starts in the same state, except for a finite number of cells in other states, often called a (global) *configuration*. More generally, it is sometimes assumed that the universe starts out covered with a periodic pattern and only a finite number of cells violate that pattern. The latter assumption is common in one-dimensional cellular automata.

2.8 A Bit of Physics

We have been concerned thus far with *discrete* situational action systems. A quite natural question which can be posed in this context is: what about *continuous* situational action systems? How should we understand them? We shall not dwell on this topic and discuss it thoroughly here. The paradigmatic examples of one-agent continuous situational action systems occupy a central place in classical mechanics and the behaviour of these systems is well described by appropriate differential equations. An analysis of the mathematical apparatus pertinent to classical mechanics might provide some clues which would enable one to build a theory of continuous action systems in a much wider conceptual setting than that provided by physics. We shall now briefly recall some basic facts, which will be very familiar to physicists. Suppose we are given a physical body consisting of finitely many particles whose individual sizes may be disregarded. We may therefore identify the body with a finite set of material points. Hamiltonian mechanics, being a reformulation of classical principles of Newtonian physics, was introduced in the 19th century by the Irish mathematician William Rowan Hamilton. The key role is played by the *Hamiltonian function* \mathcal{H} . The value of the Hamiltonian is the total energy of the system being described. For a closed system, it is the sum of the kinetic and potential energy of the system. The time evolution of the system is expressed by a system of differential equations known as the *Hamiltonian equations*. These equations are used to describe such systems as a pendulum or an oscillating spring in which energy changes from kinetic to potential and back again over time. Hamiltonians are also applied to model the energy of other more complex dynamic systems such as planetary orbits in celestial mechanics.

Each state w of the system is represented by a pair of vectors $w = (\mathbf{p}, \mathbf{q})$ from a vector space \mathbf{V} . To simplify matters, we shall assume that \mathbf{V} is a finitely dimensional, real space, say $\mathbf{V} = \mathbb{R}^n$ for some positive n . (The number n is called the *degree of freedom* of the system.) \mathbf{V} has therefore well-established topological properties. (Topology is introduced by any of the equivalent norms on \mathbf{V} .) The vector \mathbf{q} represents the (generalized) coordinates of the particles forming the system, and \mathbf{p} represents the (generalized) momenta of these particles (conjugate to the generalized coordinates). If \mathbf{V} is endowed with a Cartesian coordinate system, then the vectors \mathbf{p} are “ordinary” momenta. In the simplest case, when the system consists of one particle only, not subject to external forces, one may assume that $\mathbf{q} = (x, y, z)$ is a vector in the three dimensional Euclidean space \mathbb{R}^3 , being the coordinates of the particle, and the vector $\mathbf{p} = (p_x, p_y, p_z)$ representing the momentum of the particle.

$W := \mathbf{V} \times \mathbf{V}$ is the set of possible states of the system. W is called a *phase space*. We define an action system which is operated by one agent. We call him **Hamilton**. He performs one continuous action only. His action is identified with the Hamiltonian function \mathcal{H} . The Hamiltonian $\mathcal{H} = \mathcal{H}(\mathbf{p}, \mathbf{q}, t)$ is a (scalar valued) function and it specifies the domain T of values in which the time parameter t varies. The domain of \mathcal{H} is a subset of $\mathbf{V} \times \mathbf{V} \times T$ but we assume that the Hamiltonian \mathcal{H} makes sense for all $t \in T$. The set T is usually an open interval on the straight line \mathbb{R} (or \mathbb{R} itself).

By a *possible situation* we shall mean any pair $s = (w, t)$, where w is a state and t is real number representing time moments. In other words, possible situations are just members of the set $S := \mathbf{V} \times \mathbf{V} \times T (= W \times T)$. We therefore identify possible situations with triples $s = (\mathbf{p}, \mathbf{q}, t)$, with \mathbf{p} and \mathbf{q} described as above. The members of T play the role of labels of possible situations. We define f to be the projection assigning to each possible situation $s = (w, t)$ the state w .

How does **Hamilton** act? Suppose we are given a quite arbitrary situation $s_0 = (\mathbf{p}_0, \mathbf{q}_0, t_0) \in S$ from the domain of \mathcal{H} . s_0 is called an *initial* situation. The action of **Hamilton** undertaken in the state $(\mathbf{p}_0, \mathbf{q}_0)$ labeled by t_0 carries over the system, for every moment t , from the state $(\mathbf{p}_0, \mathbf{q}_0)$ to a *uniquely* defined state (\mathbf{p}, \mathbf{q}) with label t . In other words, for every $t \in T$, **Hamilton** turns the initial situation s_0 into a uniquely defined situation $s = (\mathbf{p}, \mathbf{q}, t)$. We may therefore say that **Hamilton** assigns to each situation s_0 from its domain a *phase path* in the phase space, i.e., a continuous mapping from T to W . Moreover, this path passes through the state $(\mathbf{p}_0, \mathbf{q}_0)$ at t_0 . It is now quite obvious how to define the transition relation Tr between possible situations. For any pair of situations $s_0 = (\mathbf{p}_0, \mathbf{q}_0, t_0)$, $s = (\mathbf{p}, \mathbf{q}, t) \in S$, it is the case that $s_0 Tr s$ if and only if both situations s_0 and s belong to the domain of \mathcal{H} and the phase path assigned to s_0 passes through the state (\mathbf{p}, \mathbf{q}) at t . In light of the above remarks, the relation Tr is reflexive on the domain of \mathcal{H} . How to compute this path? We shall discuss this issue below.

The action of **Hamilton** is a binary relation $A_{\mathcal{H}}$ defined on the set W . It is assumed that the transition relation R between states is identical with the relation $A_{\mathcal{H}}$. (**Hamilton** is omnipotent because he establishes the rules that govern mechanics.) Therefore the system we define is normal. In light of the above remarks, given a state $(\mathbf{p}_0, \mathbf{q}_0)$ and $t_0 \in T$, the set $f_R(\mathbf{p}_0, \mathbf{q}_0)$ contains in particular the range of the unique phase path assigned to the situation $s_0 = (\mathbf{p}_0, \mathbf{q}_0, t_0)$, provided that s_0 is in the domain of \mathcal{H} . We assume that for every state $(\mathbf{p}_0, \mathbf{q}_0)$, the set $f_R(\mathbf{p}_0, \mathbf{q}_0)$ is the union of co-domains of all possible phase paths assigned to $s_0 = (\mathbf{p}_0, \mathbf{q}_0, t_0)$, for all possible choices of $t_0 \in T$.

It follows from these definitions that the relation R is compatible with Tr ; that is, for any situations $s_0 = (\mathbf{p}_0, \mathbf{q}_0, t_0)$, $s = (\mathbf{p}, \mathbf{q}, t)$, $s_0 Tr s$ implies that $f(s_0) R f(s)$. Indeed, $s_0 Tr s$ means that the phase path assigned to s_0 passes through the point (\mathbf{p}, \mathbf{q}) of the phase space at t . Hence (\mathbf{p}, \mathbf{q}) belongs to the range of this path. Consequently, $(\mathbf{p}, \mathbf{q}) \in f_R(\mathbf{p}_0, \mathbf{q}_0)$, i.e., $f(s_0) R f(s)$.

We have defined a one-agent situational action system

$$\mathbf{M}^s := (W, R, \{A_{\mathcal{H}}\}, S, Tr, f).$$

The class of all such systems \mathbf{M}^s is called the *Hamiltonian class*.

How do we compute the phase paths involved in the above definitions? One assumes that:

- (1) \mathcal{H} possesses the continuous partial derivatives with respect to \mathbf{p}, \mathbf{q} (in the sense of the vector space \mathbf{V}) and t .

- (2) For every initial condition $s_0 = (\mathbf{p}_0, \mathbf{q}_0, t_0)$ belonging to the domain of \mathcal{H} , the phase path assigned to s_0 is a pair of continuous functions $(\mathbf{p}(t), \mathbf{q}(t))$, each function being a map from T to \mathbf{V} , such that $\mathbf{p}(t_0) = \mathbf{p}_0$ and $\mathbf{q}(t_0) = \mathbf{q}_0$. (In the Cartesian coordinate system, the vector function $\mathbf{q}(t)$ describes the trajectories of the particles in space while $\mathbf{p}(t)$ defines their momenta, and hence their velocities.) Moreover, by solving the Hamiltonian equations:

$$\begin{aligned}\frac{d}{dt}\mathbf{p}(t) &= -\frac{\partial}{\partial \mathbf{q}}\mathcal{H}(\mathbf{q}(t), \mathbf{p}(t), t) \\ \frac{d}{dt}\mathbf{q}(t) &= +\frac{\partial}{\partial \mathbf{p}}\mathcal{H}(\mathbf{q}(t), \mathbf{p}(t), t)\end{aligned}$$

with the initial condition $\mathbf{p}(t_0) = \mathbf{p}_0$ and $\mathbf{q}(t_0) = \mathbf{q}_0$, one computes the values of $\mathbf{p}(t)$ and $\mathbf{q}(t)$ at any moment t belonging to T .

Freedom and Enforcement in Action

A Study in Formal Action Theory

Czelakowski, J.

2015, XV, 261 p. 24 illus., Hardcover

ISBN: 978-94-017-9854-9