CASE STUDIES IN SPECIFICATION: FOUR GAMES

Alejandro Teruel

Technical Monograph PRG-30
July 1982

Oxford University Computing Laboratory
Programming Research Group
45 Banbury Road
Oxford OX2 6PE

1. Introduction

This paper is an exercise in specification style. It strives to present formal specifications of some well known games in a concise and elegant manner, using for this purpose simple mathematical tools.

Games are treated as input-output functions which can be best described using a state oriented framework. Inputs to a game are a sequence of *events* and a sequence of valid *choices*. The output is the set of winning players.

Events are typically not controlled by the players and include such items as the result of spinning a coin, throwing dice or picking a card from a shuffled pack. *Choices* on the other hand are made by the player who usually has several alternatives to pick from. Typical *choices* include deciding which counter to move, the position on the board to move it to or the card to throw away. In this paper we do not attempt to describe the decision making process that leads a player to make one choice rather than another, i.e. no descriptions of game playing strategies are given.

Particular attention has been paid to developing the specifications from the commercially available set of rules and to show, by example, how a tighter set of informal rules can be derived from the formal description of the game.

The decision to work on what some readers may feel to be a toy domain is a deliberate one. Games are one of the few domains where time and care have been spent trying to produce informal specifications (i.e. rules) which are clear, unambiguous and consistent. The results are not encouraging: natural language cannot be made sufficiently precise to rule out different interpretations of the same text without becoming too unwieldy. Games are also fun to work with. They are just as ideal a tool to teach the rudiments of specification writing as they are an ideal, yet often neglected tool to teach programming. They can also help develop that sense of aesthetics which is called mathematical maturity.

II. The Basic Framework for Game Descriptions

The framework we will adopt distinguishes:

1. The basic domains, such as the PLAYER, EVENT and CHOICE domains;
2. The set of (reachable) states, STATE;
3. The set of initial states, INITIAL;
4. The set of final states, FINAL;
5. The state transition function, move;
6. The winning players' determination function, winner.

A helpful rule of thumb to determine the state components is that they should contain enough information to allow a game to be postponed for a long period, during which the players forget all about their play, and so forget such typical items as whose turn it was to play, the positions of their counters on the board and the cards held in their hands. Occasionally the framework uses *invariant* predicates on states to characterize the set of reachable states.

In a well-formed game description, initial states and some final states are reachable states.

In its most general form, the state resulting from a move made by one of the players depends on the *previous state* , on the *choice* exercised by the player, and on the *event* which occurred:

$$move: EVENT \rightarrow CHOICE \rightarrow STATE \twoheadrightarrow STATE$$

Once a final state has been reached no further moves are possible:

$$\forall e:EVENT;\ c:CHOICE;\ s:FINAL,\ s \notin dom(move\ e\ \ c)$$

The set of winning players of a game is a subset of the set of players participating in the game. It is specified by the function *winners* which is only defined on final states:

```
winner: STATE ⇸ ℙ(PLAYER)
dom winner = FINAL
```

A game description is therefore a 5-tuple:

```
(STATE, INITIAL, FINAL, move, winner)
```

The functional meaning of a game description is given in appendix A.

III. Applications

III.1 The Game of "Best of Three Tosses" Heads or Tails

The object of specifying this game is to illustrate the concepts defined in the previous section in a very simple example.

The informal rules of the game are the following:

1. The game is for two players.
2. The first player to win two tosses wins the game.
3. Each toss consists of spinning a coin. One of the players calls out either *"heads"* or *"tails"* (but not both!) while the coin is being spun. The coin will come to rest showing either the *"heads"* face or the *"tails"* face. If it shows the face called out by the player, he wins the toss, otherwise his opponent wins it.
4. The players take alternate turns at calling out.

We start by defining the *basic domains:* PLAYER, EVENT, CHOICE. There are two players (rule 1) whom we can name player *a* and player *b* :

PLAYER = { *a* , *b* }

The players have no control over the result of spinning a coin. Therefore the EVENT domain consists of the "values" the coin can show, which rule 3 specifies as:

EVENT = { *heads* , *tails* }

The players choose whether to call out *heads* , or *tails*. Therefore the "values" of their possible calls constitute the CHOICE domain, which in this case happens to be equal to the EVENT domain:

CHOICE = { *heads* , *tails* }

The next step lies in determining the components of a state of this game. To do so it might be helpful to imagine one is calling up a friend who is watching the progress of the game and asks him to report the present situation. The number of tosses each player has won (i.e. the *score*) and whose *turn* it is to call out characterize the game before and after every turn:

```
STATE______________________________________
|   score: PLAYER → 0..2
|   turn: PLAYER
|__________
|   range score ≠ {2}
|___________________________________________
```

In the game we wish to describe at most one player can win two tosses. i.e. we are not interested in reaching or using states in which the score shows each of the two players as having won two tosses. This restriction is expressed by the *invariant:*

$$\mathrm{range}\ score \neq \{2\}$$

which was added to the specification of the class STATE. Appendix B shows how it can be proved that our *invariant* holds for all states that are reachable from an initial state by a sequence of moves. In this paper we will not attempt to show that the invariants given for the states are strong enough to characterize completely the set of reachable states: indeed in the game of Othello (section III.3) we have yet to find the strongest invariant in this sense.

Once the set of states has been defined. the set of initial states and final states can be defined by restricting the set of states appropiately. Note that the *rules* do not explicitly describe the initial state (thus the rules are incomplete). but in such cases it is generally assumed that neither of the players has won any tosses at the beginning of the game (assuming no *handicaps* are set) and so the score is zero zero.

$$\mathrm{INITIAL} \mathrel{\widehat{=}} \mathrm{STATE} \mid \mathrm{range}\ score = \{0\}$$

From rule 2 we can define final states as those states where a player has won two tosses:

$$\mathrm{FINAL} \mathrel{\widehat{=}} \mathrm{STATE} \mid 2 \in \mathrm{range}\ score$$

Still using rule 2. the set of winning players can be defined as those whose score is two:

$$winner \mathrel{\widehat{=}} (\lambda\ \mathrm{FINAL})\ \{p{:}\mathrm{PLAYER} \mid score\ p = 2\}$$

The effect of a move is to update the score. adding a win to the appropiate player. and to update the turn indicator so that the players alternate calling out (rule 4). This is formalized as follows. Let the player who won the toss be called. appropiately enough. the *tosswinner*. If we restrict the *score* component of the state to the tosswinner so:

$$score \upharpoonright \{tosswinner\}$$

we get a function with one element in its domain. the player who won the toss Now to add his win to his score all we need to do is compose the *succesor* function *succ* to this single element function

$$succ \circ score \upharpoonright \{tosswinner\}$$

which gives us the tosswinner's new score. By rewriting the old score with the tosswinner's new score we get the overall new score component for the new state

$$score' = score \oplus succ \circ score \upharpoonright \{tosswinner\}$$

To update the turn component we introduce a function called the *opponent* function which given one of the players returns his opponent, i.e. the other player:

$$\text{opponent}: \text{PLAYER} \rightarrow \text{PLAYER}$$
$$\text{opponent} \triangleq \{a \mapsto b \,,\, b \mapsto a\}$$

This allows us to update the turn component by simply applying the opponent function to the player whose turn it was:

$$\text{turn}' = \text{opponent turn}$$

Taking into account that the tosswinner is the player whose turn it was if the result of spinning the coin *e* is equal to what he called out, c , and that otherwise the tosswinner is the opponent of the player whose turn it is, i.e.

$$\text{e=c} \Rightarrow \text{tosswinner} = \text{turn} \wedge$$
$$\text{e}\neq\text{c} \Rightarrow \text{tosswinner} = \text{opponent turn}$$

we are now in a position to formally define the *move* function:

$$
\begin{aligned}
&\text{move}: \text{EVENT} \rightarrow \text{CHOICE} \rightarrow \text{STATE} \twoheadrightarrow \text{STATE}\\
&\text{move e c} \triangleq\\
&\qquad \pi\ \text{STATE} \mid 2 \notin \text{range score}\\
&\qquad\quad \text{score}' = \text{score} \oplus \text{succ}\circ\text{score} \restriction \{\text{tosswinner}\};\\
&\qquad\quad \text{turn}' = \text{opponent turn}\\
&\qquad \text{where}\\
&\qquad\quad \text{e=c} \Rightarrow \text{tosswinner} = \text{turn}\\
&\qquad\quad \text{e}\neq\text{c} \Rightarrow \text{tosswinner} = \text{opponent turn}\\
&\qquad\quad \text{opponent}: \text{PLAYER} \rightarrow \text{PLAYER}\\
&\qquad\quad \text{opponent} \triangleq \{a \mapsto b \,,\, b \mapsto a\}
\end{aligned}
$$

Finally, a study of the sequence of moves applicable to an initial state shows that the length of the sequence is never greater than three. This is captured by a variant:

```
var: STATE → N
var s ≙ 3 −$\sum_{p:PLAYER}$ score p
```

Variants are functions from states to the natural numbers whose value is smaller for states which are the outcome of a move than for the original states. The existence of a variant for a game implies that the game is finite. i.e. it will end after a certain number of moves.

To recapitulate. the formal specification of the game of "Best of Three Tosses" Heads or Tails is given on the following page.

```
Best-of-Three-Tosses
      PLAYER ≙ {a , b }
      EVENT ≙ {heads , tails }

      CHOICE
           EVENT

      STATE
           score: PLAYER → 0..2
           turn: PLAYER
      ──────────
           range score ≠ {2}

      INITIAL
         STATE | range score = {0}

      FINAL
         STATE | 2 ∈ range score

      move: EVENT → CHOICE → STATE ⇸ STATE
      winner: STATE ⇸ ℙ (PLAYER)
──────────
      move e c ≙
           π STATE | 2 ∉ range score
              score' = score ⊕ succ∘score↾{tosswinner};
              turn' = opponent turn
           where
              e=c ⇒ tosswinner = turn
              e≠c ⇒ tosswinner = opponent turn
              opponent: PLAYER → PLAYER
              opponent ≙ {a ↦b , a ↦b }
      winner ≙ (λ FINAL) {p:PLAYER | score p = 2}
```

III.2 The Game of Snakes and Ladders

The game of Snakes and Ladders (also known as Ropes and Ladders, or Slides and Ladders) is an example of an extreme kind of game: the player never has to make a choice. This will be modelled by omitting the CHOICE domain. The game will also be used to introduce the idea of a board.

III.2.1 Informal Description

0. Snakes and Ladders is played on a board whose aspect we partially reproduce in figure 1:

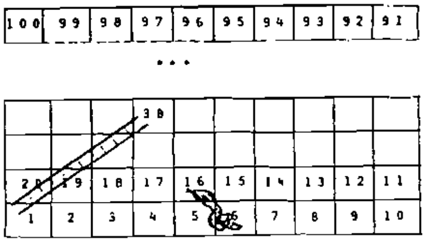

Figure 1: (Partial) Board of Snakes and Ladders.

1. The player who first reaches square 100 is the winner.

2. As many players as there are coloured pieces may compete

3a. The players first throw the dice to decide the order of starting. The player whose score is the highest shall commence first, the next highest will follow and so on.

3b. In the event of a tie in throwing, the players shall again throw to decide who shall precede the other.

4a. Tha game consists of each player throwing the dice and moving his coloured piece the number of squares as thrown.

4b. Any throw bringing a player to a square in connection with the foot of a ladder enables the player to pass upwards to the top of the ladder e.g. square 1 (Meakness) leeds upwards to square 38 (Goodness).

4c. If a throw brings one to a number on which the head of a snake lies, the player must come back to the squara containing the tail of that snake e.g. square 16 leads down to squara 6.

[A list of all ladders and snakes follows in the original rules. We omit it.]

The rules have been taken from Spear's "Junior Compendium of Games" and numbered.

III.2.2 Formalizing the Game

We consider that the informal rules describe two games:

. the "determination of the player order" game;

. the "snakes and ladder board" game (rules 1,2,4a-4c).

The two games are put together in a way which is outside the scope of this paper, in which we will describe only the "snakes and ladders board" game.

Basic domains

Rule 2 allows any number of players to participate:

PLAYER ≙ 0..n **where** n ∈ **N**

At least one player must participate.

Rule 4a clearly indicates that *events* in this game consist of the results of dice throws, thus:

EVENT ≙ 1..6

Snakes and Ladders introduces a new domain, which is the set of positions on the board. Although figure 1 seems to imply the existence of one hundred positions, the ladder of "Meakness" (starting on position number 1) cannot be used unless the player start by placing their counters off the board and bring them on to the board on their first throw. We will need to model the off-board position by incorporating position 0, thus defining:

POSITION ≙ 0..GOAL where GOAL=100

The various snakes and ladders depicted graphically on the board will be dealt with when discussing moves.

States and winner of the game

Two components suffice to describe a game in progress:

. the position reached by each player;

. the turn indicator.

At most one player will ever reach the goal; this is expressed as the invariant in:

$$
\begin{array}{|l|}
\hline
\textrm{STATE} \\
\quad \textrm{is-on}: \textrm{PLAYER} \rightarrow \textrm{POSITION} \\
\quad \textrm{turn}: \textrm{PLAYER} \\
\hline
\quad \#\,(\textrm{is-on} \rhd \{\textrm{GOAL}\}) \leq 1 \\
\hline
\end{array}
$$

Appendix B contains a proof of the invariance of the expression defined above.

Again, there is no rule describing the legal initial states. We have assumed the rules incomplete and completed them by insisting that every player starts on position 0 (i.e. off the board):

$$\textrm{INITIAL} \mathrel{\widehat{=}} \textrm{STATE} \mid \textrm{range is-on} = \{0\}$$

From rule 1 we can deduce that the game goes on until a player reaches the goal position.

$$\textrm{FINAL} \mathrel{\widehat{=}} \textrm{STATE} \mid \textrm{GOAL} \in \textrm{range is-on}$$

The players who have reached that position are the winners:

$$\textrm{winner} \mathrel{\widehat{=}} (\lambda\ \textrm{FINAL})\ \{p:\textrm{PLAYER} \mid \textrm{is-on}\ p = \textrm{GOAL}\}$$

Moves

Snakes and Ladders does not allow the player any choices when moving. Therefore the state a game will be in after a move depends only on the previous state end the result of the dice throw:

```
move: EVENT → STATE → STATE
```

Note that the rules 4a-4c define the effect of a dice throw on the position reached by the player. They only define the effect on the is-on component of a state. This effect can be seen as the *composition* of two functions:

```
displacement ∘ im-effect
```

where im-effect (*immediate effect*) is defined by rule 4a:

"...[move] the piece the number of squares as thrown."

which translates into the following type clause:

```
im-effect: EVENT → POSITION → POSITION
```

while displacement is defined by rules 4b and 4c to be of type:

```
displacement: POSITION → POSITION
```

Rule 4a cannot be interpreted as meaning "move the piece the number of squares *forwards* as thrown." i.e.

```
im-effect' :EVENT → POSITION → POSITION
im-effect' e b ≘ e + b
```

because if the player is on position 98 and a 5 is thrown, there is no position 103 to move to. The rule is not clear on this point: here are some alternative interpretations:

1. count "overshooting" the goal as reaching it:

```
im-effect' e b ≘ min(GOAL,e+b)
```

2. prohibit moves which would overshoot the goal:

```
    im-effect'' ≡ if e+b > GOAL then b else e+b
```

3. move forwards as far as possible, then backwards:

```
  im-effect e b ≡ if e+b > GOAL then 2*GOAL-(e+b)
                          else e+b
```

4. circulate round the board:

```
  im-effect''' e b ≡ (e+b) mod (GOAL+1)
```

We choose to adopt the third interpretation as being the most consistent with rule 4a.

Rules 4b and 4c define a player as being displaced if he encounters the head of a snake or the foot of a ladder. This is nothing more than a function of the form:

```
    displacement: POSITION → POSITION
    displacement ≡ id(POSITION) ⊕ {1 ↦ 38,...rest of
                                  list of snakes and ladders}
```

No rule applies to the effect of a move on the turn component, which hints at the use of the round-robin method we will adopt:

```
     move: EVENT → STATE → STATE
     move e ≙ π STATE | GOAL ∉ range is-on
         is-on' = is-on ⊕
                    displacement ∘ (im-effect e) ∘
                    (is-on†(turn));
         turn' = (turn+1) mod (n+1)
```

Note how only the player whose turn it is is affected with respect to his position on the board.

III.2.3 Formal Specification of "Snakes and Ladders Board" Game

```
Snakes-and-Ladders
|   PLAYER ≙ 0..n where n ∈ ℕ
|   EVENT ≙ 1..6
|   POSITION ≙ 0..GOAL where GOAL = 100
|
|   STATE
|   |  is-on: PLAYER → POSITION
|   |  turn: PLAYER
|   |--------
|   |  # (is-on ↓ GOAL) ≤ 1
|   
|   INITIAL
|   |  STATE | range is-on = {0}
|   
|   FINAL
|   |  STATE | GOAL ∈ range is-on
|   
|   move: EVENT → STATE ⇸ STATE
|   winner: STATE ⇸ ℙ (PLAYER)
|--------
|   move e ≙ π STATE | GOAL ∉ range is-on
|      is-on' = is-on ⊕
|               displacement ∘ (im-effect e) ∘
|               (is-on↾{turn});
|      turn' = (turn+1) mod (n+1))
|   winner ≙ (λ FINAL) {p:PLAYER | is-on p = GOAL}
|   where
|   im-effect: EVENT → POSITION → POSITION
|   im-effect e b ≙ if e+b>GOAL then 2*GOAL-(e+b)
|                         else e+b
|   displacement: POSITION → POSITION
|   displacement ≙ Id(POSITION)
|                  ⊕ { 1↦ 38,16↦6,4↦14,9↦31,
|                     21↦42,28↦84,36↦44,51↦67,
|                     71↦91,80↦100,47↦26,49↦11,
|                     56↦53,62↦19,64↦60,87↦24,
|                     93↦73,95↦75,98↦78}
```

III.2.4 Rewriting the informal Rules

Formal specifications are useful guidelines to refer to when writing informal rules. In the following paragraphs, comments referring to the rules or how they are built are put between square brackets while the rules themselves are left in clear.

[We start by describing the basic domains.]

1. Any number of players can take part.

2. Each player picks a differently coloured counter for the duration of the game.

[The use of dice is so widespread that any description would be superfluous. We assume the board is given so no description of it is necessary.]

[State components are usually not described explicitly so we pass on to the description of initial states.]

Initial set-up

3. Initially all counters are off the board. When they start moving they enter the board through square number one.

[The rules to determine the order of play are now given. These are rules 3a and 3b which have not been modified as no formal description of them has been developed.]

4a. The players now throw the dice to determine the order of starting. The player whose score is the highest shall commence first [i.e. have the first turn], the next highest will follow and so on.

4b. In the event of a tie in throwing, the players shall again throw to decide who shall precede the other.

Goal [Indicates the final state and how to determine the winner]
5. The player whose counter first comes to rest on square number 100 is the winner.

Moves
[The two components are treated separately].
6a. On his turn, a player throws the dice and moves his counter *forwerds* (exception: rule 6b) the number of squares as thrown.
Example: If the player's counter is on square number 3 and he throws a 6, he must move his counter to square number 9.
[Note the use of examples. Examples and diagrams are an important part of informal specifications as they allow a user to check his intuitive understanding of the rules. Sometimes it is easier to introduce a concept by means of examples; however, the golden (meta-)rule is to state concepts and rules *explicitly*. Examples should never substitute explicit rules.]

6b. If the player cannot move his counter forwards the number of squares as thrown because he would overshoot square number 100, then he must move forwards to square 100 and, having reached it, move *backwards* so that the total number of spaces moved is as thrown.
Example: If a player is on square 97 and he throws a 4, he can move his counter forwards only three places (to square 100) and then has to move it backwards one place so that his counter ends up on square 99.

6c. If, as a result of applying rules 6a and 6b, the player moves his counter on to a square which contains the head of a snake or the foot of a ladder, the counter must be pleced on the square which contains the tail of the snake or the top of the ladder respectively. This ends the player's turn.
Example: If a player is on square 3 and throws a 6, he moves his counter to square 9. Square 9 contains the foot of a ladder whose topis contained by square 31. Therefore the player must end his turn by placing his counter

on square 31.

Turns

7. The players teke turns in a round-robin fashion taking into account the order determined initially (see rules 4a and 4b).

III.3 The Game of Othello

Othello, also known as Reversi, is an example of another kind of extreme. Like Go, Checkers and Naughts and Crosses, it is a game of skill in which chance does not intervene. This is modelled by omitting the EVENT domain as we shall show in section III.3.2.

III.3.1 Informal Description

1. The players decide who is to be white and who is to be black. The same colours are kept throughout the game.

2. Each player places two discs of his colour on the board as shown in figure 2:

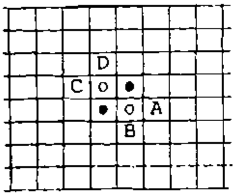

Figure 2

3. Black makes the first move.

4. On every move a disc must be placed next to an opponent's disc, either sideways, lengthways or diagonally. The disc placed *must trap* an opponent's disc (or discs) between the one placed and one already on the board in any direction.

In figure 2, A,B,C,D represent the moves that black can make. If black decides to place his disc in position A, he has captured the white disc E shown in figure 3 and turns it over so that it is black side up as shown in figure 4.

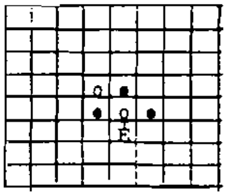

Figure 3

Now it is white's turn. Figure 4 shows the moves that are available (A,B,C).

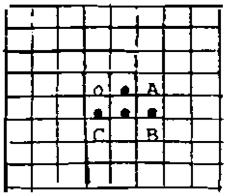

Figure 4

Notes:

A. The number of discs that can be captured in a turn in any or multiple directions is unlimited, e.g. in figure 5 black can capture all the pieces bisected by the arrows by playing in position A, thus arriving at the situation shown in figure 6 and capturing pieces in eight directions in one move, which is the maximum possible.

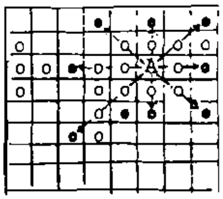

Figure 5

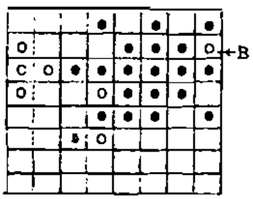

Figure 6

B. A disc or discs can only be captured as a direct result of a move. e.g. In figure 6 white disc B is *not* turned over because it is not captured directly on that move.

C. If it is impossible to capture one of your opponent's men when it is your move, you miss that turn and cannot move again until you can lay e disc that captures at least one of your opponent's discs. Until you can lay such a disc your opponent continues to play, turning over as many of your discs as possible.

6. Play continues as above until:
   a. the board is filled:
   b. one player has no pieces of his colour left on the board:
   c. no further moves as possible (as in note C) for either player.

7. At this time the discs are counted and the player with most of his colour on the board is declared the winner.

Details to be carefully noted

1. It must be emphasized that discs are only turned as a direct result of move, and must be trapped in a diract line (horizontal, vertical, or diagonal). Figures 5 and 6 should make this clear.

2. Discs that are to be turned over must be in a continuous line (you may not skip over an empty square or different colour disc. See figures 7,8.

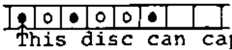
This disc can capture only the white disc next to it.

Figure 7

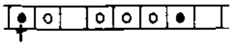
This disc may not be placed. It cannot capture the row.

Figure 8

3. At no time can you place a disc that does not capture at least one opposing disc. When this situation occurs, you miss your turn, and continue to do so until you can make a capture on your move.

4. Once placed a disc is NEVER moved to another square. It may be turned over several times during a game, but it is NEVER moved.

5. If one player runs out of discs, he draws a disc from his opponent's stock. The opponent CANNOT REFUSE to hand over the discs.

6. If an incorrect move is made it may only be corrected BEFORE the opponent's next move

[Rules taken from Peter Pan Playthings' "Rules of Play for Othello". There is a slight difference in numbering and some additional examples have been omitted.]

III.3.2 Formalizing the game

Basic domains

From rule 1. it is clear that Othello is a two player game. We will call them the *black* player and the *white* player:

PLAYER ≘ { *white* ,*black* }

The board is a subset of the Cartesian grid:

POSITION ≘ 0..m × 0..m **where** m = 7

Adjacency relations on the board

It turns out that the concept of a position being *adjacent* to another in a given direction is very useful. We will distinguish eight directions of adjacency which we will name like eight points of the compass. Each direction is a partial function from positions on the board to positions on the board, thus *north* is defined as.

n:POSITION⇸POSITION
n(x,y) ≘ (x,succ y)

and *east* is defined as

e:POSITION⇸POSITION
e(x,y) ≘ (succ x,y)

From these two definitions, and taking *south* as the *inverse* of *north*, *west* as the *inverse* of *east* and forming directions like northeast by going *north* and then going *east*, or viceversa, we can complete the eight adjacency relations by:

s,w:POSITION⇸POSITION
$\mathbf{s} = \mathbf{n}^{-1}$
$\mathbf{w} = \mathbf{e}^{-1}$
DIRECTION ≘ {n,e,s,w,n∘e,n∘w,s∘e,s∘w}

Conventions

A position is *occupied* if and only if a disc is placed on it.

A position *belongs* to a player if and only if a disc of the player's colour is on that position.

A disc *belongs* to the player of the same colour.

States

States have two components:

. the *has* function from positions to players.

. the turn indicator.

Given an occupied position, the *has* function indicates the player to whom it belongs. Unoccupied positions are not mapped, so *has* is a partial function:

$$\mathrm{has} : \mathrm{POSITION} \rightarrow\!\!\!\!\!\!+\;\, \mathrm{PLAYER}$$

The turn indicator has been used before and needs no further introduction:

STATE

$\mathrm{has} : \mathrm{POSITION} \rightarrow\!\!\!\!\!\!+\;\, \mathrm{PLAYER}$

$\mathrm{turn} : \mathrm{PLAYER}$

$\forall b : \mathrm{dom\ has} .\ (\bigcup_{d:\mathrm{DIRECTION}} d \upharpoonright \mathrm{dom\ has})^{*} \upharpoonright \{b\} = \mathrm{dom\ has}$

The invariant points out an interesting characteristic of the game. In any reachable state we can go from any occupied position to any other occupied position by succesive *hops*. Each *hop* goes from an occupied position to an adjacent, occupied position. Formally a hop is described by the relation:

$$\bigcup_{d:\mathrm{DIRECTION}} d \upharpoonright \mathrm{dom\ has}$$

i.e. the union of the eight adjacency relations, each one of them restricted and co-restricted to the set of occupied positions. The set of successions of hops is described by taking the transitive closure over hop:

$$(\cup_{d:DIRECTION}\ d \upharpoonright \text{dom has})^*$$

So the invariant specifies that if we restrict successions of hops to those starting on a given occupied position b, we can still reach any occupied position, i.e.:

$$\forall b:\text{dom has},\ (\cup_{d:DIRECTION}\ d \upharpoonright \text{dom has})^* (\{b\}) = \text{dom has}$$

Rules 2.3 define the initial state as:

$$\text{INITIAL} \triangleq \text{STATE} \mid \text{has} = \{(3,3) \mapsto \textit{black}, (4,3) \mapsto \textit{white}, (3,4) \mapsto \textit{white}, (4,4) \mapsto \textit{black}\}$$
$$\text{turn} = \textit{black}$$

The formal description of final states is postponed until *moves* have been defined.

Moves: the choices open to a player

"On every move a disc must be placed next to an opponent's disc, either sideways, lengthways or diagonally. The disc placed *must trap* an opponent's disc (or discs) between the one placed and one already on the board in any direction..."

[Rule 4]

Unfortunately the informal rules do not define the concepts of "placement" and "trapping" clearly. "Trapping" is not defined explicitly at all: its meaning is conveyed by means of examples, *ad-hoc* notes and "details to be carefully noted"; this clearly violates the meta-rule of explicitly stating concepts used. Our interpretation of rule 4 leads us to the following reformulation:

4'.1 At most one disc is placed on the board on any turn.

4'.2 The player whose turn it is must place one of his discs on an unoccupied position of the board in such a way as to *capture at least one* of his opponent's pieces. To do this, the position on which the disc was placed must form a straight line (either sideways, lengthways or diagonally) with a position already occupied (at the start of the move) by the player whose turn it is. All the positions on that straight line which lie between the two positions occupied by the player *must* belong to the player's opponent and. are said to contain the *captured* pieces.

Example: It is Black's turn to play and (part of) the board is shown by figure 10:

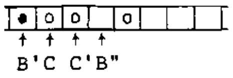

Figure 10

Figure 10 shows that Black can place one of his discs on position B" as all the conditions set forth in the rules are satisfied. i.e.

1. B" is an unoccupied position.
2. B' forms a straight line with B" and B' also belongs to Black.
3. The intermediate positions C.C' on the line from B" to B' all belong to White. Black's opponent.

4'.3 If the player whose turn it is cannot find a position which satisfies 4'.2, he forgoes his turn.

To formalize the precondition to the function *move*we need to formalize the notion of the *opponent* of a given player, and the notion of the set of positions *captured* from a given position. The *opponent* of a player is simply the other player:

```
opponent:PLAYER→PLAYER
opponent ≙ {black↦white white↦black }
```

Toget the set of positions *captured* by placing a disc on position b when in a given state s, certain conditions (stated in rule 4'.2) must be satisfied. First, b must be en unoccupied position.

```
b∉dom has
```

then, b must form a straight line with a position already occupied by the player at the start of the move. Note that a straight line is formed by any position lying in the direction d, so:

```
{(dⁿ b):POSITION |
     b∉dom e.hae ∧
     d∈DIRECTION ∧
     n∈N ∧
     dⁿb∈dom (e.has⩥{s.turn})}
```

gives us the set of positions which form a straight line with the unoccupied position b, and are occupied by the player. Finally, all the intermediate positions on that line must belong to the player's *opponent* so:

```
captured: STATE → POSITION → ℙ (POSITION)
captured STATE b ≙
    {dᵏb:POSITION |
      b∉dom has ∧
      d∈DIRECTION ∧
      n∈N ∧
      k∈1..n-1 ∧
      dⁿb∈dom (hae⩥{turn}) ∧
      ∀i:1..n-1,dⁱb∈dom (has⩥{opponent turn})}
```

There is no EVENT domain in this game, which means that move is a function of type:

$$\text{move: POSITION} \rightarrow \text{STATE} \twoheadrightarrow \text{STATE}$$

The basic domain CHOICE becomes a subset of the domain POSITION defined by move's precondition: the set of captured positions from the chosen position cannot be empty:

$$\text{move } b \equiv \lambda \text{ STATE } \mid (\text{captured STATE } b) \neq \emptyset \ \ldots$$

Moves: the effect of a choice
It is possible that the position chosen to play on can capture positions in more than one direction (see figure 5, page 22, where position A has precisely this property). In this case all captured positions, shown in figure 5 as white discs with arrows through them, become the property of the captor who must replace the discs on the captured positions with discs of his own colour. The effect is shown in figure 6, page 22. Let's state the rule explicitly as:

"All positions captured from a position b, possibly in more than one become the property of the captor who must replace the discs on the captured positions with discs of his own colour."

Formally, given s is in the domain of move(b):

$$(\text{move } b \ s).\text{has} =$$
$$s.\text{has} \oplus \{c \mapsto s.\text{turn} \mid c \in \text{captured } s \ b\} \oplus \{b \mapsto s.\text{turn}\}$$

The effect on the state's second component (the *turn* component) can be described as follows:

"If a player cannot capture any of his opponent's discs no matter where he

plays, he must skip his turn and let his opponent play."
(See Note C or, equivalently, rule 4'.3).

The formal description is:

```
(move b s).turn =
 if valid-choices((move b s).has,opponent(s.turn))= ∅
 then s.turn else opponent(s.turn)
```

Final states

A final state is a state in which the player whose turn it is cannot move, i.e. cannot capture any positions:

```
FINAL = {s:STATE | range (captured s)={∅}}
```

Conditions 6a and 6b of rule 6 are special cases of the third condition:

"Play continues...until...no further moves are possible for either player."

which is what we have formalized.

Winner

Rule 7 states that when a final state is reached:

"...the discs are counted and the player with most of his colour on the board is declared the winner."

which translates into:

```
winner(s:FINAL) =
  {p:PLAYER | card(dom s.has ↓ p)
            > card(dom s.has ↓ opponent(p) )}
```

We will also be using the function *owns* which associates a player with his set of counters:

```
owns: PLAYER → ℙ(COUNTER)
owns p ≙ {(p,i) | i∈1..4}
```

Formalizing the board presents more of a challenge, if only because of the apperent complexity of rule 2b. A careful analysis shows that every player starts on an off-board 'position', moves round 52 squares or positions to the 'outside square of the center row of the arm of his colour of the cross' (which we will denote the GATE position) from where he moves a furher six squares to reach the HOME position. Relative to the start, positions can be numbered as:

```
POSITION ≙ 0..HOME where HOME=58
```

position 0 denoting, as in Snakes and Ladders, the offboard 'position'. The special position GATE marks the last position before the player moves on to the 'homestretch':

```
GATE ≙ 52
```

A function is needed to *map* the players' counters' 'relative' positions with respect to their starting positions to a common 'absolute' position so we can determine when two counters belonging to different players are placed on the same square and 'interfere' with each other

```
map: PLAYER → POSITION → POSITION
map p b ≙ (13*p + b) mod (HOME+1)
```

Once counters are placed on the homestretch or the starting position, by the rules of the game they can no longer interfere with each other, which is why we will never apply *map* to positions outside 1 to GATE.

III.4.2 Formalizing the Game

As with Snakes and Ladders (see section III.2), this paper only formalizes the "board" game and not the game of determining the order of play. This means rule 2a will be ignored.

Basic domains

Rule 1a states that two to four players may take part. We will however slightly generalize the game to allow any number of players (up to four) to take part:

$$\mathrm{PLAYER} \subseteq 0..3$$

An explanation is necessary to show why the domain:

$$\mathrm{PLAYER}' \triangleq 0..np \text{ where } np \in 0..3$$

is inadequate. Some readers may wish to skip this explanation and continue at the next paragraph. When we have only two players participating, two games can be played according to whether the players choose to bring on their counters through adjacent or through opposite arrowed circles. Using the PLAYER' domain only covers one of these games, whereas using the PLAYER domain covers both cases.

The EVENT domain is the set of dice throw results:

$$\mathrm{EVENT} \triangleq 1..maxevent \text{ where } maxevent=6$$

The CHOICE domain is determined by rule 1b, which indicates that each player is assigned four counters, and rule 5 which states that the "...player can use his discretion as to which man he advances or enters."

$$\mathrm{COUNTER} \triangleq \mathrm{PLAYER} \times 1..4$$

3a. Each player throws once each time unless he throws six when he has another throw.

3b. The counters are moved forward as many squares as the dice indicate.

4. When six are thrown different counters may be used for each throw.

5. The player can use his discretion as to which man he advances or enters.

6. Whenever a counter is played into a space already occupied by an opponent it must be sent off the board, but if it happens to get on a square occupied by one of the same colour it is placed on top of it. In fact all counters of one colour may rest in one square.

7. The player is "Home" first is the winner.

The rules have been taken from Spear's "Junior Compendium of Games" and numbered. Rules 2b and 6 have been slightly modified to reflect the simpler board of figure 11. They both conserve the style of the original rules

III.4 The Game of Ludo

The game of Ludo, also known as Parchis, will complete this paper. It uses both tha concept of choice and that of event.

III.4.1 Informal Description

0. Ludo is played on a board whose main features we reproduce in figure 11.

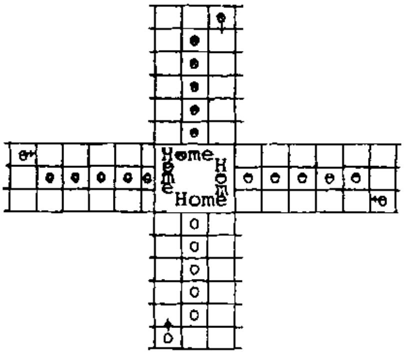

Figure 11: The board in Ludo

1a. Two, three or four players may take part.

1b. Each player is provided with four counters -all of one colour- which are initially set off the board.

2a. The dice is thrown to determine the order of starting and the one who throws the largest number commences the game.

2b. Each player brings his counters on the board by placing them on the square containing the circle and arrow of his colour, and goes all round as indicatad on the board until he arrives back to the outside square of the center row of the arm of his colour of the cross, when he proceeds up the centre row (coloured circles), which takes him to the middle of the board, called "Home".

turn it is. All the positions on that straight line, which lie between the two positions occupied by the player, *must* belong to the player's opponent and are said to contain the *captured* pieces.
Example: See page 27.

4.3 If the player whose turn it is cannot find a position which satisfies 4.2, he forgoes his turn.

5. All positions captured from the position the player chose to play on, possibly in more then one direction, become the property of the captor who must replace the discs on the captured positions with discs of his own colour. Once he has done so his turn is over.
Note that this means that a player can trap positions in up to eight directions in one turn as figure 5 and figure 6 (page 21) should make clear. In figure 5 the Black player decides to place on of his discs on position A: figure 6 shows the state of affairs at the end of Black's turn.

Turns

6. Players alternate unless one of them cannot capture any of his opponent's pieces on his turn in which case he forgoes his turn in favour of his opponent. His opponent then proceeds to play until the original player can capture at least one of his opponent's pieces or until the game comes to an end.

End of the game

7. The game ends when neither player can capture any of his opponent's pieces. At this point the player with more discs of his colour on the board wins the game.

III.3.4 Rewriting the Informal Rules

[Because section III.3.2 restated so many of the original rules, giving examples where necessary, this section will frequently refer to those examples and illustrations instead of presenting them again. Comments, such as this one, will be inserted between brackets.]

On the players

[The first three rules are virtually unchanged from their counterparts in section III.3.1].

1. The players decide who is to be white and who is to be black. The same colours are kept throughout the game.

[The board is assumed given, so no description of it is necessary.]

Initial set-up

2. The initial aspect of the board is given in figure 2 (page 20).
3. Black has the first turn.

Some conventions

A position on the board is *occupied* if and only if a disc has been placed on it.

A position *belongs* to a player if and only if a disc of the player's colour is on the position.

Moves

4.1 At most one disc is placed on the board on any turn.

4.2 The player whose turn it is must place one of his discs on an unoccupied position of the board in such a way as to *capture at least one* of his opponent's pieces. To do this, the position on which the disc was placed must form a straight line (either sideways, lengthways or diagonally) with a position already occupied, at the beginning of the move, by the player whose

```
Othello
    PLAYER ≙ {white ,black }
    POSITION ≙ 0..n where n=7

     STATE
      has: POSITION ⇸ PLAYER
      turn: PLAYER
     ─────────
      ∀b:dom has,(∪d∈DIRECTION d↾dom has)*⦇{b}⦈ = dom has

     INITIAL
     STATE | has={(3,3)↦black ,(4,3)↦white ,
                  (3,4)↦white ,(4,4)↦black };
           turn = black

    FINAL ≙ {s:STATE | range(captured s) = {∅}}
    move: POSITION ⇸ STATE ⇸ STATE
    winner: STATE ⇸ 𝔽(PLAYER)
 ─────────
    move b ≙ (π STATE | captured STATE b ≠ ∅
      has' = has ⊕
           {c↦turn | c∈captured STATE b}⊕{b↦turn};
      turn' = if valid-choice(has',opponent(turn))=∅
             then turn else opponent(turn)
    winner ≙ (λ FINAL){p:PLAYER | #dom has↾p
                               > #dom has↾(opponent p) }
    where
    n,e,s,w:POSITION⇸POSITION
    n(x,y) ≙ (x,succ y)
    e(x,y) ≙ (succ x,y)
    s ≙ n⁻¹
    w ≙ e⁻¹
    DIRECTION ≙ {n,e,s,w,n∘e,n∘w,s∘e,s∘w}
    captured: STATE → POSITION → 𝔽(POSITION)
    captured STATE b ≙
        {dᵏb | b∉dom has ∧
             n∈ℕ ∧
             k∈1..n-1 ∧
             ∀i∈1..n-1, dⁱb∈dom(has↾{(opponent turn)} ∧
             dⁿb∈dom(has↾{turn}))}
    opponent: PLAYER → PLAYER
    opponent ≙ {white ↦black, black ↦white }
```

Variant

A game of Othello always finishes in at most sixty four (i.e., the number of positions on the board) moves. This can be proved by showing that the following variant holds:

```
var(s:STATE) ≙ card(POSITION - dom s.has)
```

Additional Properties

1. If a state is reached in which the player whose turn it is has no valid choices open to him then neither does his opponent. That is, for all reachable states s:

```
s∈FINAL ⇒
range(captured (s.has (opponent s.turn))={∅}
```

2. Once a position is occupied it cannot revert to being unoccupied. Again, for all reachable states s:

```
∀b:POSITION,
(captured s b)≠∅ ⇒ dom s.has ⊆ dom (move b s).has
```

States

Two of the components of the states are similar to ones we have encountered before:

```
on: COUNTER → POSITION
turn: PLAYER
```

The *on* component tells us where each possible counter is placed on the board, by giving its relative position.

The third component is a *derived* component,i.e. one which is, strictly speaking, not necessary but which will allow the state invariant, or other properties, to be expressed more clearly. The *camps* occupied by a player consist of the set of 'absolute' positions on which the player has counters. Rule 6 ("Whenever a counter is played onto a space already occupied by an opponent it must be sent off the board...") originates the invariant: no two counters belonging to different players can occupy the same 'absolute' position on the common part of the board, i.e. their *camps* must be disjoint:

```
STATE________________________________________________
|   on: COUNTER → POSITION
|   turn: PLAYER
|   camps: PLAYER → ℙ(POSITION)
|__________
|   camps p ≙ map p ((on↓1..GATE)⦇owns p⦈)
|   ∀p₁ ,p₂ :PLAYER,p₁ ≠ p₂ ⇒ camps p₁ ∩ camps p₂ = ∅
|____________________________________________________
```

Remember f⦇X⦈ allows us to form a set by applying f to each of the elements of X in the domain of f.

A fourth component could arise by interpreting appropiately the rule of "bonus dice throws" (rule 3a) which states that "...each player throws once each time unless he throws a six, when he has another throw." Does this mean that, a) every time a player throws a six he gets an extra throw, or b) on his turn a player may have up to two dice throws, and receives the second throw only if the result of his first roll was six? The first interpretation leads to a simpler description so we will develop the specification using it.

Rule 1b also states that initially all counters are off the board:

```
INITIAL ──────────────────────────────────────────
│   STATE | range on = {0}
└──────────────────────────────────────────────────
```

A final state is reached when, according to rule 7, a player "...is HOME". The precise meaning of being at home is not made clear but we will take it to mean that a player has his four counters placed on the HOME position:

```
FINAL ────────────────────────────────────────────
│   STATE | ∃p:PLAYER, on(owns p)={HOME}
└──────────────────────────────────────────────────
```

Winner

The winner of the game is the player who places his four counters on the home position:

```
winner: STATE ⇸ ℙ(PLAYER)
winner = (λ FINAL){p:PLAYER | on(owns p)={HOME} }
```

Moves

The description of valid choices for a player is particularly muddled. These are the decisions we adopt:

I. For each dice throw at most one of the player's counters can be moved.

II. A counter cannot be moved if it is *blocked* i.e. if adding the dice throw to its present relative position would cause it to overshoot the home position.

III. Unless all four of the player's counters are blocked a counter must be moved forwards on the player's turn as many spaces as the result of the dice roll.

To make a move, the player whose turn it is must attempt one of three alternatives: *stay-put* *advance* or *start-again*

```
move : EVENT → COUNTER → STATE → STATE
move e c ≙ (stay-put e) ∪ (advance e c) ∪
               (start-again e c)
```

Stay-put can only be attempted when the result of the dicethrow precludes moving any of the player's counters. This happens when all four counters are blocked i.e

```
s.on(owns s.turn) ⊆ HOME-e .. HOME
```

There is no effect on the *on* component of the state and as will happen under any of the alternatives the *next* player will be assigned a turn. We postpone discussion of the *next* function until we have described the *advance* and *start-again* functions:

```
stay-put: EVENT → STATE → STATE
stay-put e ≙
    π STATE | on(owns turn) ⊆ HOME-e .. HOME
      on' = on ;
      turn' = next e turn
```

An *advance* can be carried out when the counter to be moved belongs to the player whose turn it is and, the new position to which it will be taken is either on the homestretch (i.e. between GATE+1 and HOME) or is not *occupied* by any other player (i.e. the new position is not one of another player's camps). The move itself consists of placing the chosen counter on the new position. The new position is determined by adding the result of the dice throw e to the position occupied by the chosen counter at the start of the move i.e. s.on(c). (~turn is the complement of the set consisting of the player whose turn it is with respect to the player domain, i.e. ~turn equals PLAYER − {turn})

```
Let newpos = (on c) + e ;
   occupied = ∪ { camps p | p∈~turn } in
advance: EVENT → COUNTER → STATE ⇸ STATE
advance e c ≙
    π STATE | c∈(owns turn) ∧ newpos≤HOME ∧
                ((map turn newpos)∉occupied ∨
                  newpos>GATE))
      on' = on ⊕ {c↦newpos};
      turn' = next e turn
```

Start-again can be described as a frustrated advance. It attempts to advance, but finds that the new position it would advance to is already *occupied* by another player. The player's chosen counter is placed on the starting position i.e. 0 (again, because of the ambiguity of rule 6, this is just one possible interpretation):

```
start-again: EVENT → COUNTER → STATE ⇸ STATE
start-again e c ≙
    π STATE | c∈(owns turn) ∧
                (map turn newpos)∈occupied ∧
                newpos≤GATE
      on' = on ⊕ {c↦0};
      turn' = next e turn
```

where newpos and occupied are defined as in *advance.*

The second component of the state (i.e. turn) is only affected by having the *next* function applied to it. The turn passes to the same player who moved last if he managed to throw a six, otherwise it passes to the player who follows him in a *round-robin* scheme:

```
next: EVENT → PLAYER → PLAYER
next e p ≙ if e=maxevent then p
              else if p=max PLAYER then min PLAYER
                   else min{p1:PLAYER | p1>p}
```

III.4.3 Formal Specification of the Game of Ludo

```
Ludo
    PLAYER ⊆ 0..3
    EVENT ≙ 1..maxevent where maxevent=6
    COUNTER ≙ PLAYER × 1..4
    POSITION ≙ 0..HOME where HOME=58
    GATE ≙ 52

    STATE
      on: COUNTER → POSITION
      turn: PLAYER
      camps: PLAYER → ℙ(POSITION)
    ──────────
      camps p ≙ map p ((on↓1..GATE)(owns p))
      ∀p1 ,p2 :PLAYER,p1 ≠ p2 ⇒ camps p1 ∩ camps p2 = ∅

    INITIAL
      STATE | range on = {0}

    FINAL
      STATE | ∃p:PLAYER, on(owns p) = {HOME}

    move: EVENT → COUNTER → STATE ⇸ STATE
    winner: STATE ⇸ ℙ(PLAYER)
──────────
    move e c ≙ (stay-put e) ∪ (advance e c) ∪
               (start-again e c)
    winner ≙ (λ FINAL){p:PLAYER | on(owns)={HOME} }
    where
    stay-put: EVENT → STATE ⇸ STATE
    stay-put e ≙
      Ξ STATE | on(owns turn) ⊆ HOME-e .. HOME
        on' = on ;
        turn' = next e turn
```

```
advance: EVENT → COUNTER → STATE ⇸ STATE
advance ≙
  let newpos = (on c)+e ;
     occupied = ∪ {camps p | p∈¯turn} in
  π STATE | c∈(owns turn) ∧ newpos<HOME ∧
            ((map turn newpos)∉occupied ∨
            newpos>GATE))
    on' = on ⊕ {c↦newpos};
    turn' = next e turn
start-again: EVENT → COUNTER → STATE ⇸ STATE
start-again ≙
  let newpos = (on c)+e ;
     occupied = ∪ {camps p | p∈¯turn} in
  π STATE | c∈(owns turn) ∧
             (map turn newpos)∈occupied ∧ newpos<GATE
    on' = on ⊕ {c↦0};
    turn' = next e turn
map: PLAYER → POSITION → POSITION
map p b ≙ (13*p + b) mod (HOME+1)
owns: PLAYER → P(COUNTER)
owns p = {(p,i) | i∈1..4}
next: EVENT → PLAYER → PLAYER
next e p ≙ if e=maxevent then p
           else if p= max PLAYER then min PLAYER
              else min{pl:PLAYER | pl>p}
```

III.4.4 Rewriting the Informal Rules

On players

1. Up to four players may take part.

Initial set-up

2. Initially each player takes four counters of the same colour which will belong to him throughout the game.

3. All counters are initially off the board.

4. [Originally rule 2a, modified] All players throw the dice once and whoever throws the largest number commences the game. If there is a tie for the largest number, the players who tied throw again until the tie is broken and one player is left with the largest number. This player takes the first turn.

Conventions

I. The middle of the board is called 'Home' (see figure 11).

II. Movement proceeds as follows. If the counter to be moved is off the board, it is brought on the board by passing it through the square containing the arrowed circle of its colour. It continues moving in the direction signalled by the arrow until it arrives at the outside square which is exactly six (coloured) squares away from the 'home' position of its colour, when it proceeds up the row of squares containing circles of its colour to the middle of the board.

Examples: Assume it is a player's turn, and that he wishes to move a counter presently off the board. The result of his dicethrow is a three, so he places his counter on the position marked A in figure 12.

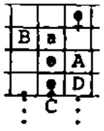

Figure 12: A portion of the Ludo board

Assume that on his next turn the player is still at A and throws a one. Then his counter can be moved from position A to D in figure 12. Finally, assume the player has managed to place one of his counters on B and throws a five. If he wishes to move the counter on B, he must move it to the position pointed at by C.

iii. A counter is said to be blocked if moving it forwards the number of spaces indicated by the dice roll would make it overshoot its "home" position.

On moves

5. On his turn a player moves at most one counter.

6. On his turn a player makes a dice throw and then choses which of his counters to move. Only an unblocked counter can be moved.

7. If the player has at least one unblocked counter he must move it (or one of them, as the case may be) forwards as many squares as the dice roll indicates, following for this matter convention ii. The turn then ends unless rule 8 applies.
If all the player's counters are blocked, his turn ends without any change in the position of his counters.

8. If the player's chosen counter comes to rest on a position occupied by one or more of an opponent's counters, the player's chosen counter is sent off the board and the turn ends. More than one counter of the same player may rest on any square.

On turns

9. Every time the player whose turn it is throws a six, the next turn will be his as well.

10. Should the player whose turn it is not throw a six, the next player in a clockwise feshion has a turn.

On winning

11. The first player to get his four counters on the "home" position is the winner of the game.

IV. Related Work

The framework is based on that used by J.R. Abrial in his course on 'System Specification', although, of course, state oriented mechanisms go back to the misty prehistory of Computer Science. The notation used is introduced in B. Sufrin's papers.

Games have been treated many times, especially in the discipline of Artificial Intelligence, though more emphasis has been put on the development of game playing strategies rather than game specification. K.C. Bowen has introduced an informal taxonomy of game classes which, for example, distinguishes between games played in an automata-like fashion and those played in e 'person-like' fashion. Although his point of view is that of a behavioural scientist interested in decision making processes, that distinction sparked off the analogical idea of extreme kinds of games, lacking either an event domain (cf.Othello, section III.3) or a choice domain (cf. Snakes and Ladders, section III.2). Some of Bowen's other distinctions suggest fruitful lines of research, for example, the difference between games with "open" or "closed" information. Roughly speaking, decisions in "open" games are made with full knowledge of the state of the game, while "closed" games *hide* or even distort part of the information contained in the state. We hope to formalize these ideas in a future paper.

Ranan B. Banerji uses a similar approach to ours: he defines two player games as 5-tuples <S,R,*P*,W,L> where he distinguishes a *P* and an *O* player Then:

1. *S* is the set of states of the game(does not include *turn* as a component);

2. *P* is the subset of states of the game, on which it is *P's* turn to move;

3. *W* is the set of final states where *P* wins;

4. *L* is the set of final states where *P* loses;

5. *R* is the relation that indicates "which of the states are obtainable from which state by an action or a move."

Our framework allows the description of n-player games and so, in this sense, is a little more general than Banerji's.

V. Future Directions of Research

The main thrust of our research is into the use of mathematics as a specification tool in the area of game description. It is our hypothesis that this can be harnessed to serve the following purposes:

1. Introduce mathematical techniques at the school level.

2. Illuminate the use of similar tools in programming.

3. Sharpen the composition of informal rules in a variety of situations.

4. Permit the design, development and implementation of game packages for recreation, education or research.

More research has to be done before the feasibility of any of these purposes can be meaningfully evaluated:

1. A wider range of games needs to be tackled. Suggestions include trying out card games (Patience games, Poker games) and so called "war" games (Diplomacy)

2. The place of rules limiting the information available to the players should be determined. Alternative frameworks could be proposed and evaluated.

3. Interesting properties of games need to be enunciated and proved to hold Certainly both the game designer and the programmer would be attracted by the possibility of proving some kind of completeness or consistency of the rules

4. Proof methods and presentations should be studied and improved. The proofs presented in Appendix B are not going to fire the enthusiasm of a school level student, a programmer, a game designer or any reasonably motivated reader.

5. Relations between informal rules and their formal counterparts need to be looked into more carefully.

6. The use of these specifications in developing other products should also be evaluated. After all, the programmer is interested in obtaining working programs, the game designer in presenting enjoyable game packages...even players are interested in developing winning strategies. Products for consideration should include game state displays, move validators, game playing programs and game consultants.

Acknowledgements

To Cliff Jones who set the Snakes and Ladders problem and encouraged my first, monstruous specification of it;

To my supervisor, C.A.R. Hoare who insisted time and time again that there had to be a simpler way...and pointed me in the right directions;

To J.R. Abrial who waded through umpteen specifications making a comment here, a suggestion there, and demolishing shaky arguments wherever he found them cropping up;

To Bernard Sufrin who convinced me of rewriting extensive parts of this paper.

To my sister Susana who incurred my wrath by telling me she preferred the original rules of Othello to a previous rewritten draft of them, and showed me why;

To the "Fundacion Gran Mariscal de Ayacucho" and the Committee of Vice-chancellors and Principals of the Universities of the United Kingdom for quietly financing my stay at Oxford;

Last but not least, to those friends who had to put up with me, especially in the last, frenzied days of typing and to those friends and family whose letters have been piling up unanswered for quite some time;

MANY THANKS!

Appendix A: The Functional Meaning of a State Oriented Game Description.

If we wish to specify a game called *game1* which is an input-output function of type:

```
game1: seq(EVENT) × seq(CHOICE) ⇸ ℙ(PLAYER)
```

we can build a description:

```
game1' ≙ (STATE,INITIAL,FINAL,move,winner)
```

where as usual INITIAL,FINAL ⊆ STATE,and such that:

```
winner: STATE ⇸ ℙ(PLAYER)
move: EVENT → CHOICE ⇸ STATE ⇸ STATE
game1(es,cs) ≙
  winner(move-repeatedly(es,cs,s:INITIAL))
```

where *move-repeatedly* is a recursive function which repeatedly applies *move* to s and the first element of the event and choice lists es. cs stripping these lists and going from state to state until a final state is reached:

```
move-repeatedly: seq(EVENT) × seq(CHOICE) ⇸ STATE
move-repeatedly(es,cs,s) ≙
  (if s∈FINAL then s
   else
      move-repeatedly(tl es,tl cs,move (hd es)(hd
cs)s)
```

This paper occasionally uses variants of this framework such as those omitting the choice sequence (section III.2), or the avant sequence (section III.3). The reader should be able to convince himself that no new ideas have been introduced in the process.

Appendix B: Some Proofs

B.1 The game of "Best of three tosses" Heads or Tails !

Theorem

The invariant of the state:

$\forall s$:STATE, range s.score $\neq$ {2}

holds.

Proof

```
a. The invariant holds for any initial state.
1.  s∈INITIAL                                ...hypothesis
2.  range s.score = {0}                      ...1,def. of INITIAL
3.  range s.score ≠ {2}                      ...2

b.  Given that the invariant holds for an arbitrary
    non-final state s, then it holds for the state
    reached from s after making a move.
    Let s' = move e c ;
1.  s∈STATE - FINAL ∧ e∈EVENT ∧ c∈CHOICE ∧
    range s.score ≠ {2}                      ...hypothesis
2.  not(∃p:PLAYER,s.score(p) = 2)            ...1,[s∉FINAL]
3a. Assume e=c
4a. s'.score = s.score ⊕ {s.turn↦s.score(s.turn)+1}
                                              ...3a,def. move
5a. s'.score(s.turn)<2 ∧ s'.score(opponent s.turn)<2
                                             ...4a,2
6a. range s'.score ≠ {2}                     ...5a
3b. Assume e≠c
4b. range s'.score ≠ {2}                     ...3b,similar
                                                reasoning
                                                as 4a-6a
7.  ∀s:STATE, range s.score ≠ {2}            ...6a,4b
```

B.2 The game of Snakes and Ladders

Theorem

```
    The invariant:
      ∀s:STATE, #(s.is-on↓GOAL) < 1
holds.
```

Proof

```
a.  The invariant holds for any initial state.
1.  s∈INITIAL                        ...hypothesis
2.  range(s.is-on)={0}               ...1,def. of INITIAL
3.  GOAL∉range(s.is-on)              ...2,GOAL≠0
4.  #(s.is-on↓GOAL) < 1              ...3

b.  Given that the invariant holds for an arbitrary
    non-final state s, it also holds for any state
    s' obtained from s by making a move.
    Let s'=(move e s) in:
1.  s∉FINAL ∧ #(s.is-on↓GOAL)<1
                                     ...hypothesis
2.  GOAL∉range(s.is-on)              ...1
3.  ∀p:PLAYER,s.is-on(p)<GOAL        ...2
4.  s'.is-on=s.is-on⊕
             displacement ∘ im-effect(e) ∘
             (s.is-on↾{turn})        ...def. move
5.  ∀p:PLAYER-{s.turn},s'.is-on(p)<GOAL  ...3,4
6.  s'.is-on(s.turn)<GOAL            ...3,4
7.  #(s'.is-on↓GOAL) < 1             ...5,6
```

The Game of Othello

Theorem

In any reachable state we can go from any occupied position to any other occupied position by successive *hops* . Each *hop* takes us from an occupied position to an adjacent occupied position. Formally:

$$\forall s\text{:STATE};b\text{:dom } s.\text{has},$$
$$(\cup_{d \in \text{DIRECTION}} d \rhd \text{dom } s.\text{has})^{*} (\!| \{b\} |\!) = \text{dom } s.\text{has}$$

Informal Proof

The proof is by induction.

Inductive base : Inspection on initial states.

Inductive step

Assume s is a non-final state in which the invariant holds, and b is any legal choice of position. Let s' be the state reached by playing b when in state s.

The occupied positions in s' are, by definition of move, the occupied positions in s, plus position b. By the inductive hypothesis we can get from any one of the occupied positions in s to any other occupied position in s by successive hops, so we only need to worry about getting to occupied positions from b and viceversa.

For b to be a valid choice,the player must be able to capture a non-empty set of occupied positions from it. By definition this set of captured pieces must include the adjacent position to b, position (d b) for some direction d. Therefore one can hop from b to (d b) , and as (d b) must have been an occupied position in s, by the inductive hypothesis we can get to any other occupied position in s from position (d b), so by adding the hop from b to (d b) to the succession of hops which get us from (d b) to any other occupied position in s, we can get from b to any other occupied position in s', by a succession of hops.

To get to b from any other occupied position in s', all we have to do is to successively hop to the previously mentioned position (d b) and then hop to the adjacent position b. This *inverse* hop can be done as:

$$\forall d{:}\mathrm{DIRECTION},\ d \in \mathrm{DIRECTION} \Rightarrow d^{-1} \in \mathrm{DIRECTION}$$

We leave the reader to prove the validity of this assertion.

Formal Proof

Inductive base

0. By inspection on initial states.

Inductive step

1. $s \notin FINAL$
2. $\forall b{:}\mathrm{dom}\ s.has\ (\bigcup_{d:DIRECTION} d \rhd \mathrm{dom}\ s.has)^{*}\llparenthesis\{b\}\rrparenthesis = \mathrm{dom}\ s.has$

   ...inductive hypothesis
3. $\exists c'{:}POSITION \mid captured\ c'\ s \neq \emptyset$ ...1
4. Let c be any position such that $(captured\ c\ s) \neq \emptyset$,

   i.e. $c = \tau\{c' \mid (captured\ c'\ s) \neq \emptyset\}$

   ...3
5. Let $s' = move\ c\ s$ ...4
6. $(captured\ c\ s) \neq \emptyset \Rightarrow \exists d{:}DIRECTION, (d\ c) \in \mathrm{dom}\ s.has$

   ...def. of captured, properties of corestriction
7. $\exists d{:}DIRECTION, (d\ c) \in \mathrm{dom}\ s.has$ ...6,4
8. $\mathrm{dom}\ s'.has = \mathrm{dom}\ s.has \cup \{c\}$

   ...5, def. of move
9. $(\bigcup_{d:DIRECTION} d \rhd \mathrm{dom}\ s'.has) =$

   $(\bigcup_{d:DIRECTION} d \rhd \mathrm{dom}\ s.has$

   $\bigcup_{d:DIRECTION} \{c \mapsto (d\ c), (d\ c) \mapsto c \mid (d\ c) \in \mathrm{dom}\ s.has\})$

   ...8, $\forall d{:}DIRECTION, d^{-1} \in DIRECTION$,

10. $(\bigcup_{d:DIRECTION} d \rhd \mathrm{dom}\ s'.has)^{*} = \{b \mapsto b' \mid b, b' \in \mathrm{dom}\ s'.has\}$

    ...9,2, properties of *
11. $\forall b{:}\mathrm{dom}\ s'.has\ (\bigcup_{d:DIRECTION} d \rhd \mathrm{dom}\ s'.has)^{*}\llparenthesis\{b\}\rrparenthesis = \mathrm{dom}\ s'.has$

    ...10

B.4 The game of Ludo

Theorem

No player can "trespass" on another. More formally, players' camps are disjoint, i.e. for all reachable states.

```
      ∀p,p1:PLAYER, p≠p1 ⇒ camps p ∩ camps p1 = ∅
```

Proof

```
0.  p≠ p1                                 ...hypothesis
A.  Inductive Base
1.  s∈INITIAL
2.  range s.on = {0}                      ...1,def.INITIAL
3.  ∀p2:PLAYER,s.camps p2 = ∅             ...2,def.camps
4.  camps  p ∩ camps p1 = ∅               ...3

B.  Inductive hypothesis and step
1.  s.camps p ∩ s.camps p1 = ∅ ∧ s∉FINAL
   Let s' = move e c s

   Case I: s' = stay-put e s
   I.1 s'.on = s.on                       ...def.stay-put
   I.2 s'.camps p ∩ s'.camps p1 = ∅       ...I.1,1

   Case II: s' = start-again e c s
   II.1 s'.on = s.on ⊕ {c↦0}     .        ...def.start-again
   II.2 ∀p2:PLAYER,s'.camps p2 ⊆ s.camps p2
                                          ...II.1,def.camps
   II.3 s'.camps p ∩ s'.camps p1 = ∅      ...II.2,1

   Case III: s' = advance e c s
   III.1 s'.on = s.on ⊕ {c↦newpos}        ...def.advance
   III.2 ((map s.turn newpos)∉occupied ∨ newpos>GATE)
                                          ...def.advance
                                             (precondition)
   III.3 newpos>GATE
           ⇒ ∀p2:PLAYER,s'.camps p2 ⊆ s.camps p2
                                          ...III.1,def.camps
```

III.4 newpos>GATE $\Rightarrow$ s'.camps p $\cap$ s'.camps p1 = $\emptyset$
...III.3,1

III.5 c$\in$(owns s.turn) ...def.advance
(precondition)

III.6 s'.camps$\restriction^{-}$s.turn = s.camps$\restriction^{-}$s.turn
...III.1,III.5

III.7 (map s.turn newpos)$\notin\bigcup_{p2^{-}s.turn}$ (s.camps p2)
$\Rightarrow$ s'.camps s.turn $\cap$ $\bigcup_{p2^{-}s.turn}$ (s.camps p2)
= $\emptyset$ ...III.1,III.5

III.8 (map s.turn newpos)$\notin$occupied
$\Rightarrow$ s'.camps p $\cap$ s'.camps p1 =
$\emptyset$ ...III.6,III.7

III.9 s'.camps p $\cap$ s'.camps p1 = $\emptyset$
...III.8,III.4,III.2

References

Abrial, J.R.

'Lecture Notes on System Specification.'

Oxford, 1980 [Unpublished]

Bowen, K.C.

'Resaarch Games: An approach to the study of decision processes.'

Taylor and Francis, 1978

Banerji, Ranan B.

'Artificial Intelligence: A theoretical approach.'

Elsevier North Holland, 1980

Peter Pan Playthings Ltd.

'Rules of Play for Othello.'

[Undated]

Spear's Games

'Spear's Junior Compendium of Games.'

[Undated]

Sufrin, B

'Formal Specification: Notation and Examples.'

[in] 'Tools and Notions for Program Construction'

Cambridge University Press, 1982

Sufrin, B.

'Reading Formal Specifications.'

PRG Monograph #24, Oxford [In preparation]

PRG-18	Elizabeth Fielding *The Specification of Abstract Mappings and their Implementation as B^+-trees*
PRG-19	Dana Scott *Lectures on a Mathematical Theory of Computation*
PRG-20	Zhou Chao Chen and C. A. R. Hoare *Partial Correctness of Communicating Processes and Protocols*
PRG-21	Bernard Sufrin *Formal Specification of a Display Editor*
PRG-22	C. A. R. Hoare *A Model for Communicating Sequential Processes*
PRG-23	C. A. R. Hoare *A Calculus of Total Correctness for Communicating Processes*
PRG-24	Bernard Sufrin *Reading Formal Specifications*
PRG-25	C. B. Jones *Development Methods for Computer Programs including a Notion of Interference*
PRG-26	Zhou Chao Chen *The Consistency of the Calculus of Total Correctness for Communicating Processes*
PRG-27	C. A. R. Hoare *Programming is an Engineering Profession*
PRG-28	John Hughes *Graph Reduction with Super-Combinators*
PRG-29	C. A. R. Hoare *Specifications, Programs and Implementations*
PRG-30	Alejandro Teruel *Case Studies in Specification. Four Games*