Tic-Tac-Toe ADT

1 Abstract Data Types and Design by Contract

1.1 Incompleteness in contracts

Tic-Tac-Toe game is played on a 3-by-3 board, which is initially empty. There are two players: a “cross” player and a “circle” player. They take turns; each turn changes exactly one cell on the board from empty to the symbol of the current player (cross or circle). The “cross” player always starts the game. The rules that define when the game ends and which player wins are omitted from the task for simplicity.

Below you will find an interface view of GAME class representing Tic-Tac-Toe games.

```java
class GAME

create make

feature -- Initialization
make
  -- Create an empty 3-by-3 board
  ensure
cross_turn: next_turn = Cross
end

feature -- Constants
Empty: INTEGER is 0
Cross: INTEGER is 1
Circle: INTEGER is 2
  -- Symbolic constants for players and states of board cells

feature -- Access
next_turn: INTEGER
  -- Player that will do the next turn

item (i, j: INTEGER): INTEGER
  -- Value in the board cell (i, j)
require
  i_in_bounds: i >= 1 and i <= 3
  j_in_bounds: j >= 1 and j <= 3
ensure
    valid_value: Result = Empty or Result = Cross or Result = Circle
end

feature -- Basic operations
put_cross (i, j: INTEGER)
  -- Put cross into the cell (i, j)
require
  cross_turn: next_turn = Cross
  i_in_bounds: i >= 1 and i <= 3
```

1
The contract of this class is incomplete with respect to the game description given above. In which contract elements does the incompleteness reside? Express in natural language what the missing parts of the specification are. Give an example of a scenario that is allowed by the above contract, but should not happen in Tic-Tac-Toe:

1.2 ADT GAME

Create an ADT that describes Tic-Tac-Toe games. The ADT functions should correspond one-to-one to the features of the GAME class above. The axioms of the ADT should be sufficiently complete, overcoming the incompleteness of the class contracts.
TYPES
GAME

FUNCTIONS
- make :
- next_turn :
- item :
- put_cross :
- put_circle :
- Empty :
- Cross :
- Circle :

PRECONDITIONS
P1
P2
P3

AXIOMS
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
1.3 Proof of sufficient completeness

Prove that your specification is sufficiently complete.