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.

```
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
```

```
      j_in_bounds: j >= 1 and j <= 3
      empty: item (i, j) = Empty
    ensure
      cross_put: item (i, j) = Cross
      circle_turn : next_turn = Circle
    end

  put_circle (i, j: INTEGER)
      -- Put circle into the cell (i, j)
    require
      circle_turn : next_turn = Circle
      i_in_bounds: i >= 1 and i <= 3
      j_in_bounds: j >= 1 and j <= 3
      empty: item (i, j) = Empty
    ensure
      circle_put : item (i, j) = Circle
      cross_turn: next_turn = Cross
    end
invariant
  valid_player : next_turn = Cross or next_turn = Circle
end
```

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:

The postcondition of *make* doesn not describe the board cell values. Postconditions of *put_cross* (i, j) and *put_circle* (i, j) do not describe what happens with board cells other than (i, j). For example, after a sequence of calls

```
create game.make
game.put_cross (2, 2)
game.put_circle (1, 1)
```

we expect *game.item* (2, 2) = *Cross*, but according to the contracts also *game.item* (2, 2) = *Empty* and *game.item* (2, 2) = *Circle* are possible.

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

- $make : GAME$
- $next\_turn : GAME \rightarrow INTEGER$
- $item : GAME \times INTEGER \times INTEGER \nrightarrow INTEGER$
- $put\_cross : GAME \times INTEGER \times INTEGER \nrightarrow GAME$
- $put\_circle : GAME \times INTEGER \times INTEGER \nrightarrow GAME$
- $Empty : INTEGER$
- $Cross : INTEGER$
- $Circle : INTEGER$

PRECONDITIONS

P1 $item(g, i, j)$ require $1 \leq i \leq 3$ and $1 \leq j \leq 3$

P2 $put\_cross(g, i, j)$ require $next\_turn(g) = Cross$ and $1 \leq i \leq 3$ and $1 \leq j \leq 3$ and $item(g, i, j) = Empty$

P3 $put\_circle(g, i, j)$ require $next\_turn(g) = Circle$ and $1 \leq i \leq 3$ and $1 \leq j \leq 3$ and $item(g, i, j) = Empty$

AXIOMS

We assume $1 \leq i, j, k, l \leq 3$.

A1 $next\_turn(make) = Cross$

A2 $next\_turn(put\_cross(g, i, j)) = Circle$

A3 $next\_turn(put\_circle(g, i, j)) = Cross$

A4 $item(make, i, j) = Empty$

A5 $item(put\_cross(g, i, j), i, j) = Cross$

A6 $(k \neq i \vee l \neq j) \implies item(put\_cross(g, i, j), k, l) = item(g, k, l)$

A7 $item(put\_circle(g, i, j), i, j) = Circle$

A8 $(k \neq i \vee l \neq j) \implies item(put\_circle(g, i, j), k, l) = item(g, k, l)$

A9 $Empty = 0$

A10 $Cross = 1$

A11 $Circle = 2$

1.3 Proof of sufficient completeness

Prove that your specification is sufficiently complete.

For all terms T there exist resulting terms not involving any functions of the ADT when evaluating $next\_turn(T)$ and $item(T)$.

Once again we assume $1 \leq i, j, k, l \leq 3$.

Induction basis

For all creators above holds.

- $next\_turn(make) \overset{A1}{=} Cross \overset{A10}{=} 1$
- $item(make, i, j) \overset{A4}{=} Empty \overset{A9}{=} 0$

Induction hypothesis

Assume for Tsub being a subterm of T that this is true.

Induction step

- For put_cross:
  - $next\_turn(put\_cross(g, i, j)) \overset{A2}{=} Circle \overset{A11}{=} 2$
  - $$item(put\_cross(g, i, j), k, l) = \begin{cases} \overset{A5}{=} Cross \overset{A10}{=} 1 \text{ if } i = k \wedge l = j \\ \overset{A6}{=} item(g, k, l) \text{ otherwise} \end{cases}$$
- For put_circle:
  - $next\_turn(put\_circle(g, i, j)) \overset{A3}{=} Cross \overset{A10}{=} 1$
  - $$item(put\_circle(g, i, j), k, l) = \begin{cases} \overset{A7}{=} Circle \overset{A11}{=} 2 \text{ if } i = k \wedge l = j \\ \overset{A8}{=} item(g, k, l) \text{ otherwise} \end{cases}$$