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
```
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` does 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

\[
\begin{align*}
\text{create game, make} \\
\text{game.put.cross (2, 2)} \\
\text{game.put.circle (1, 1)}
\end{align*}
\]

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
• \textit{make} : GAME
• \textit{next\_turn} : GAME → INTEGER
• \textit{item} : GAME × INTEGER × INTEGER × INTEGER ↦ INTEGER
• \textit{put\_cross} : GAME × INTEGER × INTEGER × INTEGER ↦ GAME
• \textit{put\_circle} : GAME × INTEGER × INTEGER × INTEGER ↦ GAME
• \textit{Empty} : INTEGER
• \textit{Cross} : INTEGER
• \textit{Circle} : INTEGER

PRECONDITIONS
\textbf{P1} \textit{item}(g, i, j) require 1 ≤ i ≤ 3 and 1 ≤ j ≤ 3
\textbf{P2} \textit{put\_cross}(g, i, j) require \textit{next\_turn}(g) = \textit{Cross} and 1 ≤ i ≤ 3 and 1 ≤ j ≤ 3 and \textit{item}(g, i, j) = \textit{Empty}
\textbf{P3} \textit{put\_circle}(g, i, j) require \textit{next\_turn}(g) = \textit{Circle} and 1 ≤ i ≤ 3 and 1 ≤ j ≤ 3 and \textit{item}(g, i, j) = \textit{Empty}

AXIOMS
We assume 1 ≤ i, j, k, l ≤ 3.
\textbf{A1} \textit{next\_turn}(make) = \textit{Cross}
\textbf{A2} \textit{next\_turn}(\textit{put\_cross}(g, i, j)) = \textit{Circle}
\textbf{A3} \textit{next\_turn}(\textit{put\_circle}(g, i, j)) = \textit{Cross}
\textbf{A4} \textit{item}(make, i, j) = \textit{Empty}
\textbf{A5} \textit{item}(\textit{put\_cross}(g, i, j), i, j) = \textit{Cross}
\textbf{A6} (k \neq i \lor l \neq j) \implies \textit{item}(\textit{put\_cross}(g, i, j), k, l) = \textit{item}(g, k, l)
\textbf{A7} \textit{item}(\textit{put\_circle}(g, i, j), i, j) = \textit{Circle}
\textbf{A8} (k \neq i \lor l \neq j) \implies \textit{item}(\textit{put\_circle}(g, i, j), k, l) = \textit{item}(g, k, l)
\textbf{A9} \textit{Empty} = 0
\textbf{A10} \textit{Cross} = 1
\textbf{A11} \textit{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 $\text{next turn}(T)$ and $\text{item}(T)$. Once again we assume $1 \leq i, j, k, l \leq 3$.

**Induction basis**

For all creators above holds.

- $\text{next turn}(\text{make}) \overset{A_1}{=} \text{Cross} \overset{A_{10}}{=} 1$
- $\text{item}(\text{make}, i, j) \overset{A_4}{=} \text{Empty} \overset{A_9}{=} 0$

**Induction hypothesis**

Assume for $T_{\text{sub}}$ being a subterm of $T$ that this is true.

**Induction step**

- For put\_cross:
  
  - $\text{next turn}(\text{put cross}(g, i, j)) \overset{A_2}{=} \text{Circle} \overset{A_{11}}{=} 2$
  - $\text{item}(\text{put cross}(g, i, j), k, l) = \begin{cases} \overset{A_5}{=} \text{Cross} \overset{A_{10}}{=} 1 & \text{if } i = k \land l = j \\ \overset{A_6}{=} \text{item}(g, k, l) & \text{otherwise} \end{cases}$

- For put\_circle:
  
  - $\text{next turn}(\text{put circle}(g, i, j)) \overset{A_3}{=} \text{Cross} \overset{A_{10}}{=} 1$
  - $\text{item}(\text{put circle}(g, i, j), k, l) = \begin{cases} \overset{A_7}{=} \text{Circle} \overset{A_{11}}{=} 2 & \text{if } i = k \land l = j \\ \overset{A_8}{=} \text{item}(g, k, l) & \text{otherwise} \end{cases}$