## MODULE *TicTacToe*

A specification of Tic-Tac-Toe in the Behavioral Programming style, after Harel et al., *CACM* 2012, `http://www.wisdom.weizmann.ac.il/~harel/papers/Behavioral%20programming%20.pdf`

The idea of Behavioral Programming is that specifications be constructed iteratively and interactively, by gradually adding rules, each specifying a "*b*-thread" (which corresponds to a TLA$^+$ formula, not a TLA$^+$ behavior), and allowing verification at each stage. The rules below do not follow precisely those of Harel, but they follow them in spirit; the variables and definitions below are therefore introduced as needed. The properties defined after each rule can be verified in the model checker before the following rules are defined, thus forming an incremental style of specification.

The goal of this specification is to examine the viability of specifying in the bahvioral programming style in TLA$^+$.

Historical note

In the 1830s (probably, he does not provide a date), having become convinced that "every game of skill is susceptible of being played by an automaton," and after contemplating chess and finding it too taxing, Charles Babbage decided to build a machine that would play Tic-Tac-Toe ("the simplest game with which I am acquainted") against itself, "surrounded with such attractive circumstances that a very popular and profitable exhibition might be produced" that would raise money to fund his Analytical Engine, which would have been, had it been built, the first general purpose computer. Not only was the first computer able to play the game over one hundred years away, Babbage would not have been able to write a formal specification similar to the one below. George Boole's algebra would be invented only some years later, based on Babbage's (and George Peacock's) pioneering work in abstract algebra, and formal logic as we know was forty or fifty years away. Babbage would not have been pleased with the following specification, which would have made the attractive animatronic effects he had planned redundant, as the play tactics always lead to a draw.

(see Charles Babbage, *Passages from the Life of a Philosopher*, 1864)

Conclusions

Rules 1-3, which specify the rules of the game, feel a bit contrived specified in the behavioral way, however, specifying them in this way felt quite easy, allowing to focus on one concept at a time. Rules 4-7, containing the play tactics, are a natural fit for the behavioral style, but in this particular specification, because they have no state or temporal features of their own, would have been just as easily composed in the ordinary specification style. However, one can easily imagine temporal rules, which may benefit from the behavioral style. While the result is not conclusive, I think the style deserves further consideration. Some changes to TLC (based on the comments inline, especially with regards to creating the conjoined specification can make the experience more pleasant, by allowing a more elegant, less tedious way of enabling and disabling some of the rules to examine their effect.

EXTENDS *Naturals*, *FiniteSets*

1. Board: At each step, an $X$ or an $O$ is marked on the board

$\textsc{variable}$ $board$, $pretty\_board$
$v1 \triangleq \langle board, pretty\_board \rangle$

$N \triangleq 3$
$Empty \triangleq$ "-"
$Player \triangleq \{$"X", "O"$\}$
$Mark \triangleq Player$
$Square \triangleq \{Empty\} \cup Mark$

$BoardType \triangleq \land board \in [(1 \,..\, N) \times (1 \,..\, N) \rightarrow Square]$ This is more convenient
$\qquad \land pretty\_board \in [1 \,..\, N \rightarrow [1 \,..\, N \rightarrow Square]]$ Displayed more nicely in TLC output

$Pretty(b) \triangleq [x \in 1 \,..\, N \mapsto [y \in 1 \,..\, N \mapsto b[x, y]]]$

$BoardFull \triangleq \forall i, j \in 1 \,..\, N : board[i, j] \neq Empty$

$Init1 \triangleq \land board = [i, j \in 1 \,..\, N \mapsto Empty]$
$\qquad \land pretty\_board = Pretty(board)$

$Next1 \triangleq \land \exists i, j \in 1 \,..\, N, mark \in Mark : \land board[i, j] = Empty$
$\qquad\qquad \land board' = [board \text{ EXCEPT } ![i, j] = mark]$
$\qquad \land pretty\_board' = Pretty(board')$

$Board \triangleq Init1 \land \Box[Next1]_{v1}$

$TicTacToe1 \triangleq Board$

Properties we can state at this point:
$\textsc{theorem}$ $TicTacToe1 \Rightarrow \Box BoardType$

$OnceSetAlwaysSet \triangleq$
$\quad \forall i, j \in 1 \,..\, N : \Box(\exists mark \in Mark : board[i, j] = mark \Rightarrow \Box(board[i, j] = mark))$
$\textsc{theorem}$ $TicTacToe1 \Rightarrow OnceSetAlwaysSet$

2. *EnforceTurns*: $X$ and $O$ play in alternating turns

$\text{VARIABLE } current,$
$\quad turn$ Necessary for some properties we may wish to state

$v2 \triangleq \langle v1, turn, current \rangle$

$Other(player) \triangleq \text{IF } player = \text{"X" THEN "O" ELSE "X"}$
$Opponent \triangleq Other(current)$

$$TurnType \triangleq \land current \in Player$$
$$\qquad\qquad\quad \land turn \in Nat$$

$$Init2 \triangleq \land turn = 0$$
$$\qquad\qquad \land current = \text{"X"}$$
$X$ starts

$$Next2 \triangleq \land turn' = turn + 1$$
$$\qquad\qquad \land current' = Opponent$$
$$\qquad\qquad \land \exists\, i, j \in 1 \,..\, N : \land board[i, j] = Empty$$
$$\qquad\qquad\qquad\qquad\qquad\qquad \land board'[i, j] = current$$

$EnforceTurns \triangleq Init2 \land \Box[Next2]_{v2}$

$TicTacToe2 \triangleq TicTacToe1 \land EnforceTurns$

Properties we can state at this point:

$\text{THEOREM } EnforceTurns \Rightarrow TurnType$

$Alternating \triangleq \Box[current' \neq current]_{v2}$
$\text{THEOREM } EnforceTurns \Rightarrow Alternating$

3. *DetectWin*: Detect win or draw and end game

$\text{VARIABLE } win$

$v3 \triangleq \langle v2, win \rangle$

$$
\begin{aligned}
Result &\triangleq Player \cup \{\text{"Draw"}\} \\
WinType &\triangleq win \in \{Empty\} \cup Result \\
GameEnd &\triangleq win \in Result
\end{aligned}
$$

$$
\begin{aligned}
Line \triangleq\ & \{[i \in 1 \,..\, N \mapsto \langle i, y \rangle] : y \in 1 \,..\, N\} && \text{horizontal} \\
& \cup \{[i \in 1 \,..\, N \mapsto \langle x, i \rangle] : x \in 1 \,..\, N\} && \text{vertical} \\
& \cup \{[i \in 1 \,..\, N \mapsto \langle i, i \rangle]\} \cup \{[i \in 1 \,..\, N \mapsto \langle i, N - i + 1 \rangle]\} && \text{diagonal}
\end{aligned}
$$

$f \circ g \triangleq [x \in \text{DOMAIN } g \mapsto f[g[x]]]$

$BoardLine(line) \triangleq board \circ line$

$$
\begin{aligned}
Won(player) &\triangleq \exists\, line \in Line : BoardLine(line) = [i \in 1 \,..\, N \mapsto player] \\
NoWin &\triangleq \neg \exists\, player \in Player : Won(player)' \\
StopGame &\triangleq board' = board \quad \text{UNCHANGED } board \text{ – fails } TLC
\end{aligned}
$$

$$
\begin{aligned}
Init3 &\triangleq win = Empty \\
Next3 &\triangleq \lor \land win = Empty \\
&\qquad \land \lor \exists\, player \in Player : Won(player)' \land win' = player \\
&\qquad\quad \lor NoWin \land BoardFull' \land win' = \text{"Draw"} \\
&\qquad\quad \lor NoWin \land \neg BoardFull' \land \text{UNCHANGED } win \\
&\quad\ \lor \land win \in Player \\
&\qquad \land \text{UNCHANGED } win \\
&\qquad \land StopGame
\end{aligned}
$$

$DetectWin \triangleq Init3 \land \Box[Next3]_{v3}$

$TicTacToe3 \triangleq TicTacToe2 \land DetectWin$

Properties we can state at this point:

$\text{THEOREM } DetectWin \Rightarrow WinType$

$GameEndsWhenPlayerWins \triangleq \Box(win \in Player \Rightarrow \Box[board' = board]\_v3)$ (Temporal formulas containing actions must be of fo

$GameEndsWhenPlayerWins \triangleq \Box[(win \in Player \Rightarrow \text{UNCHANGED } board)]_{v3}$ SANY wants parentheses

$\text{THEOREM } TicTacToe3 \Rightarrow GameEndsWhenPlayerWins$

$AtLeast5TurnsToWin \triangleq win \neq Empty \Rightarrow turn \geq 2 * N - 1$

$\text{THEOREM } TicTacToe3 \Rightarrow \Box(AtLeast5TurnsToWin)$

$GameEndsWhenBoardFull \triangleq BoardFull \Rightarrow GameEnd$

$\text{THEOREM } TicTacToe3 \Rightarrow \Box(GameEndsWhenBoardFull)$

4. *AddThirdToWin*: Add third mark to win

So far, we've specified the rules of the game. Now we start adding tactic rules. This one says that if a player has two marks in a line they should place the third to win.

But we run into a problem: the tactics may be contradictory, and prioritization is required. $b$-threads can be prioritized, and we could simulate that mechanism with with maps of boolean functions, but that would be overly clever, especially in a simple specification such as this. Instead, we'll order the rules by their priority, and explicitly model priorities. This means that new rules would need to be inserted in the sequence of rules into their right position.

$$Count(mark, line) \triangleq Cardinality(\{i \in 1 \,..\, N : BoardLine(line)[i] = mark\})$$

$$CanWin(player) \triangleq \exists\, line \in Line : \begin{array}{l} \land\, Count(player, line) = N - 1 \\ \land\, Count(Empty, line) = 1 \end{array}$$

$$MarkLast(line) \triangleq \exists\, i \in 1 \,..\, N : \begin{array}{l} \land\, BoardLine(line)[i] = Empty \\ \land\, board'[line[i]] = current \end{array}$$

$$\begin{array}{lcl} v4 & \triangleq & v3 \\ Init4 & \triangleq & \text{TRUE} \\ Next4 & \triangleq & CanWin(current) \Rightarrow \\ & & \quad \exists\, line \in Line : Count(current, line) = N - 1 \land MarkLast(line) \end{array}$$

$$Priority1 \triangleq CanWin(current)$$

$$AddThirdToWin \triangleq Init4 \land \Box[Next4]_{v4}$$

$$TicTacToe4 \triangleq TicTacToe3 \land AddThirdToWin$$

5. *BlockOpponentFromWinning*: Block the other player if they're about to win

$$\begin{array}{lcl} v5 & \triangleq & v4 \\ Init5 & \triangleq & \text{TRUE} \\ Next5 & \triangleq & CanWin(Opponent) \land \neg Priority1 \Rightarrow \\ & & \quad \exists\, line \in Line : Count(Opponent, line) = N - 1 \land MarkLast(line) \end{array}$$

$$Priority2 \triangleq Priority1 \lor CanWin(Opponent)$$

$$BlockOpponentFromWinning \triangleq Init5 \land \Box[Next5]_{v5}$$

$$TicTacToe5 \triangleq TicTacToe4 \land BlockOpponentFromWinning$$

6. *MarkCenterIfAvailable*: Prefer center square

$CenterSquare \triangleq \langle (N+1) \div 2,\ (N+1) \div 2 \rangle$
$CenterFree \triangleq board[CenterSquare] = Empty$

$v6 \triangleq v5$
$Init6 \triangleq \text{TRUE}$
$Next6 \triangleq (CenterFree \land \neg Priority2) \Rightarrow board'[CenterSquare] = current$

$Priority3 \triangleq Priority2 \lor CenterFree$

$MarkCenterIfAvailable \triangleq Init6 \land \Box[Next6]_{v6}$

$TicTacToe6 \triangleq TicTacToe4 \land MarkCenterIfAvailable$

Properties we can state at this point:

$FirstMarksSquare \triangleq turn = 1 \Rightarrow board[CenterSquare] \neq Empty$
THEOREM $TicTacToe6 \Rightarrow \Box(FirstMarksSquare)$

7. *MarkCornerIfAvailable*: Prefer corner square

$CornerSquares \triangleq \{1,\ N\} \times \{1,\ N\}$
$CornerFree \triangleq \exists\, corner \in CornerSquares : board[corner] = Empty$

$v7 \triangleq v6$
$Init7 \triangleq \text{TRUE}$
$$Next7 \triangleq (CornerFree \land \neg Priority3) \Rightarrow \exists\, corner \in CornerSquares : \begin{array}{l} \land\ board[corner] = Empty \\ \land\ board'[corner] = current \end{array}$$

$Priority4 \triangleq Priority3 \lor CornerFree$

$MarkCornerIfAvailable \triangleq Init7 \land \Box[Next7]_{v7}$

$TicTacToe7 \triangleq TicTacToe6 \land MarkCornerIfAvailable$

Properties we can state at this point:

$SecondMarksCorner \triangleq turn = 2 \Rightarrow \exists\, corner \in CornerSquares : board[corner] \neq Empty$
THEOREM $TicTacToe7 \Rightarrow \Box(SecondMarksCorner)$

The tactics are sufficient to always force a draw
$AlwaysDraw \triangleq (win \notin Player)$
THEOREM $TicTacToe7 \Rightarrow \Box AlwaysDraw$

The conjoined spec. In this particular spec a conjunciton of $WF_{vi}$(Nexti) would work, but as this is not true in general for BP systems, we only specify liveness for the canonical representation.

$TicTacToe \triangleq TicTacToe7$

A mechanical translation of *TicTacToe* into a specification that TLC can handle follows, based on the equivalences $\Box A \land \Box B \equiv \Box(A \land B)$ , $\Box[A]_x \equiv \Box(A \lor \text{UNCHANGED } x)$ and propositional logic equivalences (distributivity of conjunction over disjunction).

In the case of this particular specification, a simpler composition may have sufficed, but I wanted to see how convenient the general mechanical composition would be.

$$
\begin{aligned}
Compose(NextA, UnchA, NextB, UnchB) \triangleq\ & \lor NextA \land NextB \\
& \lor NextA \land UnchB \\
& \lor UnchA \land NextB \\
& \lor UnchA \land UnchB
\end{aligned}
$$

UNCHANGED causes an error, as well as the use of variable sequences, as in $v2' = v2$. If fixed, the previous definition could be made nicer, and the following *Unch* definitions made redundant.

$$
\begin{aligned}
Unch1 &\triangleq board' = board \land pretty\_board' = pretty\_board \\
Unch2 &\triangleq turn' = turn \land current' = current \land Unch1 \\
Unch3 &\triangleq win' = win \land Unch2 \\
Unch4 &\triangleq Unch3 \\
Unch5 &\triangleq Unch4 \\
Unch6 &\triangleq Unch5 \\
Unch7 &\triangleq Unch6
\end{aligned}
$$

$$
\begin{aligned}
Next12 &\triangleq Compose(Next1, Unch1, Next2, Unch2) \\
Unch12 &\triangleq Unch1 \land Unch2 \\
Next123 &\triangleq Compose(Next12, Unch12, Next3, Unch3) \\
Unch123 &\triangleq Unch12 \land Unch3 \\
Next1234 &\triangleq Compose(Next123, Unch123, Next4, Unch4) \\
Unch1234 &\triangleq Unch123 \land Unch4 \\
Next12345 &\triangleq Compose(Next1234, Unch1234, Next5, Unch5) \\
Unch12345 &\triangleq Unch1234 \land Unch5 \\
Next123456 &\triangleq Compose(Next12345, Unch12345, Next6, Unch6) \\
Unch123456 &\triangleq Unch12345 \land Unch6 \\
Next1234567 &\triangleq Compose(Next123456, Unch123456, Next7, Unch7) \\
Unch1234567 &\triangleq Unch123456 \land Unch7
\end{aligned}
$$

$$
\begin{aligned}
vars &\triangleq \langle v1, v2, v3, v4, v5, v6, v7 \rangle \\
Init &\triangleq Init1 \land Init2 \land Init3 \land Init4 \land Init5 \land Init6 \land Init7 \\
Next &\triangleq Next1234567
\end{aligned}
$$

$TicTacToe0 \triangleq Init \land \Box[Next]_{vars} \land \mathrm{WF}_{vars}(Next)$

$Terminates \triangleq win \neq Empty$

THEOREM $TicTacToe0 \Rightarrow \Diamond Terminates$

THEOREM $TicTacToe0 \Rightarrow TicTacToe$ There's a difference in liveness so no $\equiv$