Prove with SPARK: No Math, Just Code

Yannick Moy – SPARK Product Manager – AdaCore
Program Integrity

Correct data initialization

No language error or exception (division by zero, buffer overflow)

Functionality

Correct data flow

Complete lines are removed

Falling piece does not overlap with fallen pieces or board background
type Cell is (Empty, I, O, J, L, S, T, Z);

subtype Shape is Cell range I .. Z;

subtype Three Shape is Cell range J .. Z;

subtype PX_Coord is Integer range -1 .. X_Size - 1;
subtype PY_Coord is Integer range -1 .. Y_Size - 1;

type Direction is (North, East, South, West);

type Piece is record
  S : Shape;
  D : Direction;
  X : PX_Coord;
  Y : PY_Coord;
end record;

Cur_Piece : Piece;

subtype X_Coord is Integer range 1 .. X_Size;
subtype Y_Coord is Integer range 1 .. Y_Size;

type Line is array (X_Coord) of Cell;

type Board is array (Y_Coord) of Line;

Cur Board : Board;
type Action is (Move_Left, Move_Right, Move_Down, Turn_Counter_Clockwise, Turn_Clockwise);

procedure Do_Action (A : Action; Success : out Boolean);

procedure Include_Piece_In_Board;

procedure Delete Complete Lines;
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: analysis of data and information flow ...

no message?
no reads of uninitialized data

Program Integrity
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: analysis of data and information flow ...

no message?
no reads of uninitialized data

Program Integrity

```
procedure Do_Action (A : Action; Success : out Boolean) with
  Global => (Input => Cur_Board, In_Out => Cur_Piece);

procedure Include_Piece_In_Board with
  Global => (Input => Cur_Piece, In_Out => Cur_Board);

procedure Delete_Complete_Lines with
  Global => (In_Out => Cur_Board);
```

no message?
previous + correct data dependencies

Functionality
no message?
previous + no run-time error (division by zero, buffer overflow...)

here: 6 messages on buffer overflows + 4 on scalar ranges
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...

no message?

previous + no run-time error (division by zero, buffer overflow...)

here: 6 messages on buffer overflows + 4 on scalar ranges

Program Integrity

function Within_Bounds (Y : Integer; X : Integer) return Boolean is
  (Y in Y_Coord and then X in X_Coord);

function Within_Bounds (P : Piece) return Boolean is
  (case P.S is when O => Within_Bounds (P.Y, P.X) and then ...)

procedure Include_Piece_In_Board with
  Global => (Input => Cur_Piece, In_Out => Cur_Board),
  Pre   => Within_Bounds (Cur_Piece);

no message!
Functionality

NO_COMPLETE_LINES - Complete lines are removed

```plaintext
147 function Is_Complete_Line (L : Line) return Boolean is
148     (for all X in X_Coord => L(X) /= Empty);
149
150 function No_Complete_Lines (B : Board) return Boolean is
151     (for all Y in Y_Coord => not Is_Complete_Line (B(Y)))
152     with Ghost;
```

Functionality

NO_OVERLAP - Falling piece does not overlap with fallen pieces or board background

```plaintext
155 function Is_Empty (B : Board; Y : Integer; X : Integer) return Boolean is
156     (X in X_Coord and then Y in Y_Coord and then B(Y)(X) = Empty);
157
158 function No_Overlap (B : Board; P : Piece) return Boolean is
159     (case P.S is
160         when O => Is_Empty (B, P.Y, P.X) and then Is_Empty (B, P.Y, P.X + 1) and then
161                 Is_Empty (B, P.Y + 1, P.X) and then Is_Empty (B, P.Y + 1, P.X + 1),
162         when I =>
163             (for all Y in I_Delta =>
164                 (for all X in I_Delta =>
165                     (if Possible_I_Shapes (P.D) (Y, X) then Is_Empty (B, P.Y + Y, P.X + X)))),
166         when Three_Shape =>
167             (for all Y in Three_Delta =>
168                 (for all X in Three_Delta =>
169                     (if Possible Three_Shapes (P.S, P.D) (Y, X) then Is_Empty (B, P.Y + Y, P.X + X)))));
```
State automaton

piece falling

piece blocked

board before clean

board after clean
State automaton

- piece falling
- piece blocked
- board before clean
- board after clean

Ghost code

```plaintext
82< type State is (Piece_Falling, Piece_Blocked, Board_Before_Clean, Board_After_Clean) with Ghost;
87< Cur_State : State with Ghost;

173< function Valid_Configuration return Boolean is
174<   (case Cur_State is
175<     when Piece_Falling | Piece_Blocked =>
176<       No_Overlap (Cur_Board, Cur_Piece),
177<     when Board_Before_Clean => True,
178<     when Board_After_Clean =>
179<       No_Complete_Lines (Cur_Board))
180<   with Ghost;

216< procedure Include_Piece_In_Board with
217< Pre => Cur_State = Piece_Blocked and then
218<   Valid_Configuration,
219< Post => Cur_State = Board_Before_Clean and
220<   Valid_Configuration;
```
no message?
previous + code implements specification
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...

no message?
previous + code implements specification

How hard is it?
Fully proved at level 0 in 11 seconds
...out of 5 levels!
level 0 =
one prover only
≈ 1 sec timeout
no splitting

...on one core!
4 seconds with
multicore