Formal Modeling (WS 2024)
Assignment 1 (June 12, 2024)

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
Wolfgang.Schreiner@risc.jku.at

The result is to be submitted by the deadline stated above ia the Moodle interface of the course as a single archive file in `.zip` or `.tgz` format which contains the following files:

1. A single PDF file with the following contents:
   - a cover page identifying the course, the assignment, and the submitter;
   - a section for each part of the assignment, which contains
   - the deliverables requested for this section with a snapshot of the listing of the corresponding RISCAL file that contains all additions/changes to the skeleton file handed out (nicely formatted and typeset in a fixed-width font of readable size with no line overflows), and
   - optionally any explanations or comments you would like to make.
2. All RISCAL files developed in the assignment.

All assignments only ask to complete the definitions of predicates by formulas in first order logic (operations $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$). You may also use if-then-else and let-in expressions to make the formulas more readable.

Hint: you may annotate arbitrary formulas and terms by the `print` expression (see the RISCAL manual section B.5.15) to understand the derived results. For instance:

```
// result is p(e), prints first e and then p(e) in separate lines
print p(print e)

// result is f(x,y), prints x and y in one line, then f(x,y) in another
print "x:{1}, y:{2}", x, y in print f(x,y)
```

Assignment 1A (40P): Removing Duplicates

$$a : [\overbrace{2, 3, 3, 1, 2, 1}^{N}] \rightsquigarrow b : [\underbrace{\overbrace{2, 1, 3}^{N}}_{n}, 0, 0, 0]$$

We consider the problem of removing duplicates from an array: given an array a of length N, we wish to compute an an array b of length N and an integer n such that b contains at the first n indexes the same elements as a (in other words: every element of a occurs among the first n elements of b and vice versa), in no specific order, but without any duplicates (in other words: the first n elements of b are pair-wise distinct); at the remaining indices, array b contains 0.

The attached RISCAL file `toset.txt` contains a procedure `toset` that solves this problem with the help of an auxiliary procedure `has` that takes an integer a, an integer n, and an element x and returns "true" if x occurs at the first n positions of a. Your task is as follows:

1. Complete the definition of the predicates `hasPred` and `tosetPred` that define the postconditions of the respective procedures.

2. Define in the pop-up window "Other Values" suitable values for the model parameters N and M (e.g., $N = 5$, $M = 3$). For each of the procedures `has` and `toset` perform the following tasks (press in the "Operation" panel the button "Show/Hide Tasks" to open the "Tasks" menu).

3. Validate the specification of the procedure by running (with option "Nondeterminism" switched *on* and option "Silent" switched *off*) the task "Execute specification". Analyze the printed output to investigate which input/output pairs are allowed by your definition. Are all (and only those) pairs printed that you expect?

4. Further validate your specification by checking (with the option "Apply SMT Solver to all Theorems" in the corresponding popup menu) the various conditions listed under "Validate Specification". Are the results as you have expected? If a condition is false, use the option "Show Counterexample" in the corresponding popup menu to derive counterexample values for the formula (whose definition is given by the option "Print Definition"). Explain the results.

5. Run task "Execute Operation" to check whether the procedure indeed satisfies your specification (you can assume that the procedure is correct; any errors indicate deficiencies in the specification).

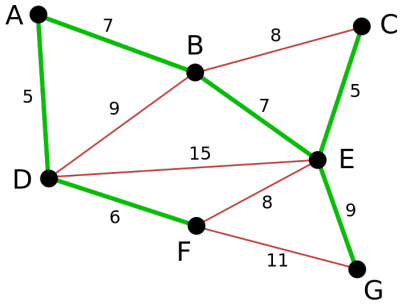
Demonstrate by (a reasonable selection of) the RISCAL output that you indeed have performed above tasks. Interpret the results and judge whether your specifications are adequate.

You only have to perform *one* of the alternatives of assignment 1B.

Assignment 1B (60P, Alternative 1): Minimum Spanning Trees

We consider "Kruskal's Algorithm"[1] for computing the minimum spanning tree of a connected undirected graph whose nodes are numbered $0 \ldots N$ and whose edges are assigned weights $1 \ldots W$. The RISCAL file `Kruskal.txt` contains an implementation of this algorithm as the following procedure:

```
proc kruskal(G:Graph): Set[Edge]
  requires connected(edges(G));
  ensures minstree(result,G);
{
  var S: Set[Edge] = edges(G);
  var T: Set[Set[Node]] = { { v } | v: Node };
  var E: Set[Edge] = ∅[Edge];
  choose e∈S with ∀e0∈S. G[e] ≤ G[e0] do
  {
    S := S\{e};
    choose n1∈e, n2∈e with n1 ≠ n2;
    choose t1∈T, t2∈T with n1 ∈ t1 ∧ n2 ∈ t2;
    if t1 ≠ t2 then
    {
      E := E ∪ {e};
      T := (T \ { t1, t2 }) ∪ { t1 ∪ t2 };
    }
  }
  return E;
}
```



While the algorithm is of its own interest, its details are actually not relevant for this assignment. The main point is that above procedure is annotated with the problem's precondition and postcondition based on predicates `connected(`E`)` (edge set E describes a connected graph) and `minstree(`E`,`G`)` (edge set E describes a minimum spanning tree for graph G). Your task is as follows:

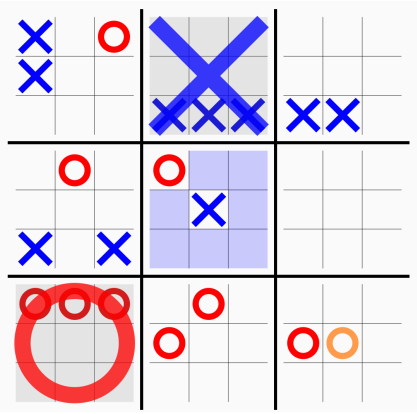
1. Complete the definition of these predicates. The file already contains a skeleton of a suitable definition based on various auxiliary predicates that describe standard notions of graph theory[2].

2. Validate the adequacy of your definition for small values of the model parameters (e.g. $W = 2$ and $N = 3$) in the same way as for assignment 1A (execute the specification itself, check the various validation conditions, and execute the procedure with your pre- and postcondition; to speed up the execution of the procedure, you may switch *off* the execution option *Nondeterminism*).

Again, demonstrate by (a reasonable selection of) the RISCAL output that you indeed have performed above tasks. Interpret the results and judge whether your conditions are adequate.

[1] https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
[2] https://en.wikipedia.org/wiki/Connectivity_(graph_theory)
https://en.wikipedia.org/wiki/Tree_(graph_theory)
https://en.wikipedia.org/wiki/Cycle_(graph_theory)
https://en.wikipedia.org/wiki/Path_(graph_theory)

Assignment 1B (60 Points, Alternative 2): Ultimate Tic-Tac-Toe



We consider the game "Ultimate-Tic-Tac Toe"[3]. The RISCAL file `UltimateTicTacToe.txt` contains a system `UltimateTicTacToe` (together with accompanying auxiliary definitions) whose action `play` plays (depending on the value of constant `all`) some/all possible instances of this game.

1. Complete the definition of the predicate `wins` that determines whether a particular player has won the global board of the game. This predicate should make use of another predicate `winsLocal` that determines whether a player has won a local board of the game.

2. Complete the definition of the predicate `legal` that determines whether a particular move (a choice of a local board and a position in that board) is legal.

Validate your specification by running some/all possible games. A violation of the system invariant indicates that the game has been won by some player; this prints the winning situation and (if the variable `trace` is set to `true`) the moves leading to the situation (thus games resulting in a draw are not printed). Investigate some of the won games whether they were indeed correctly played and the winner was correctly determined.

Explain in your submission one played game in detail and why it was correctly played.

[3] https://en.wikipedia.org/wiki/Ultimate_tic-tac-toe