User Manual

1.1 Execution

The Java Development Kit (JDK) version 8 or later must be installed and the JAVA_HOME variable must be set properly.

1.2 Create a new model

To create a new wRebeca model, from the "File" menu choose the "new" command. Then after modeling the protocol, you can save it through the "save" command. There are two types of model: wRebeca and bRebeca. Static networks are modeled by "bRebeca", while dynamic networks can be modeled by "wRebeca".

To model a protocol by wRebeca, please follow the syntax given in Figure 1.1.

Note that in the main part while instantiating from the defined reactive-classes, node neighbors can be specified which determines the initial topology. In case no neighbor is specified or the defined initial topology is not valid with respect to the given topology constraint, all valid topologies are considered.

1.3 Compiling the Model

Before compiling the model, user should configure the state space builder engine through the "CompileInfo" form. The configuration options are as follows:

- Type of storage (bag or queue)
- Applying reduction or not which means the elimination of τ transitions in a wRebeca model, and applying the counting abstraction in a bRebeca model.
- Output file format, how the generated state space is saved, in an aut, or ACTL file.

```
          Model ::= ReactiveClass+ Main
           Main ::= main {RebecDecl+ ConstraintPart }
        List(X) ::= ⟨X,⟩*X | ϵ
     RebecDecl ::= C R (List(R)) : (List(V));
ConstraintPart ::= constraint {Constraint}
     Constraint ::= ConstrainDec | ! ConstrainDec | and(Constraint , Constraint)
   ConstrainDec ::= con(R , R) | true
  ReactiveClass ::= reactiveclass C { StateVars MsgServer* }
     StateVars ::= statevars { VarDecl* }
     MsgServer ::= msgsrv M(List(T V)) { Statement* }
       VarDecl ::= T V;
     Statement ::= VarDecl | Assign | Conditional | Loop | Broadcast | Multicast | Unicast | break;
        Assign ::= V = Expr;
   Conditional ::= if (Expr) Block else Block
         Block ::= Statement | { Statement* }
          Loop ::= while(Expr) Block
     Broadcast ::= M(List(Expr));
     Multicast ::= multicast ( V , M(List(Expr)));
       Unicast ::= unicast ( Rec , M(List(Expr))) succ : Block unsucc : Block
           Rec ::= self | V
```

Figure 1.1: wRebeca language syntax: Angle brackets (⟨ ⟩) are used as meta-parentheses. Superscript * indicates zero or more times repetition. The symbols C, R, T, M, and V denote the set of classes, rebec names, types, method and variable names, respectively. The symbol $Expr$ denotes an expression, which can be an arithmetic or a boolean expression.

- The number of threads used to create the state space.
- The maximum amount of heap.

After configuring the state space engine, the model would be compiled and the compiling errors would be shown in the "Output" tab, below the source code tab.

1.4 Creating the State Space

After successfully compiling the model, by selecting the state space creation command from the menu, the state space will be created based on the configuration setup made prior to compiling.