APPENDIX A

BNGL SYNTAX

This appendix presents BNGL syntax, including comparments and energy patterns, in Extended Backus-Naur Form (EBNF). EBNF symbols are summarized in Table A1. The BNGL specification listed here is a guideline, and not necessarily authoritative.

Comments are initiated by the `#` character and continue to the end of the line. Line continuation is signified by \, which must be the last non-whitespace character on the line. The following specification assumes that comments and line continuations have been handled in preprocessing. Optional white space is also omitted from the syntax to improve clarity.

The following non-terminals are not defined below, but the standard definitions may be assumed: `Letter`, `Digit`, `Real`, `PositiveInteger`, `NaturalNumber` $(0, 1, \ldots)$, and `String`.

Table A1: Summary of symbols in Extended Backus-Naur Form

description	*symbol*
definition	`=`
concatenation	`,`
termination	`;`
alternation (or)	`|`
optional	`[ ... ]`
one-or-more	`{ ... }`
terminal string	`" ... "`
comment	`(* ... *)`

```
# BNGL syntax with compartments and energy patterns
# (Extended Backus-Naur Form)

WS = {" "|"\t"};
NewLine = {"\n"};

Name = Letter, [{Letter|Digit|"_"}];
State = "~",(Letter|Digit),[{Letter|Digit|"_"}] | "~?";
Bond = "!",{Digit} | "!?"  | "!+";
Tag = "%",(Letter|Digit),[{Letter|Digit|"_"}];
Compartment = "@",Name;
LineLabel = {Digit},WS | Name,":",[WS];

ComponentType = Name, [{"~",State}];

Component = Name, [{"~",State | "!",Bond | "%",Tag}];

MoleculeType = Name, ["(", [ComponentType, [{",",ComponentType}]], ")"];

Molecule =
    Name, [{Tag|Compartment}], ["(",[Component,[{",",Component}]],")"]
  | Name, ["(",[Component,[{",",Component}]],")"], [{Tag|Compartment}];

SimpleMolecule = Name, [{Tag|Compartment}], ["()"]
               | Name, ["()"], [{Tag|Compartment}];

PatternMods = {"$" | "{matchOnce}"};

PatternQuantifier = ("<"|"<="|"=="|">="|">"),NaturalNumber;

Pattern = "0"
        | [{Tag|Compartment},(":"|"::")], [PatternMods],
          Molecule, [{".",Molecule}], [PatternQuantifier];

Species = "0"
        | [{Tag|Compartment},(":"|"::")], [$], Molecule, [{".",Molecule}];

SimpleSpecies = [{Tag|Compartment},(":"|"::")], SimpleMolecule;

RuleModifier =
    "DeleteMolecules" | "MoveConnected" | "TotalRate"
  | "exclude_reactants","(",PositiveInteger,Pattern,[{",",Pattern}],")"
  | "include_reactants","(",PositiveInteger,Pattern,[{",",Pattern}],")"
  | "exclude_products","(",PositiveInteger,Pattern,[{",",Pattern}],")"
  | "include_products","(",PositiveInteger,Pattern,[{",",Pattern}],")";

RateLaw =
    MathExpression
  | "Sat","(",MathExpression,",",MathExpression,")"
  | "MM","(",MathExpression,",",MathExpression,")"
  | "Hill","(",MathExpression,",",MathExpression,",",MathExpression,")"
  | "Arrhenius","(",MathExpression,",",MathExpression,")";
```

```
(*optional whitespace is permitted in definitions below this point*)

UniRule = Pattern,[{"+",Pattern}], "->",Pattern,[{"+",Pattern}],
          WS,RateLaw,[{WS,RuleModifier}];

RevRule = Pattern,[{"+",Pattern}],"<->",Pattern,[{"+",Pattern}],
          WS,RateLaw,",",RateLaw,[{WS,RuleModifier}];

Rule = UniRule | RevRule;

PopulationMap =
    Species, "->", SimpleSpecies, WS, RateLaw,, [{WS,RuleModifier}];

Reaction = NaturalNumber,[{",",NaturalNumber}],
           WS, NaturalNumber,[{",",NaturalNumber}],
           (Real|Real,"*",Name);

Observable =
    ("Molecules"|"Species"), WS, Name, WS, Pattern, [{",",Pattern}];

Group = Name, WS, [NaturalNumber,"*"], NaturalNumber,
        [{",", [NaturalNumber,"*"], NaturalNumber}];

MathExpression = Real | Id | "(",MathExpression,")"
               | UnaryOperator,MathExpression
               | MathExpression,BinaryOperator,MathExpression
               | Id,"(",[MathExpression, [{",",MathExpression}]],")";

ParameterDefn = Id, (WS|"="), MathExpression;

FunctionDefn =
    Id, ["(", [Name,[{",",Name}]], ")"], (WS|"="), MathExpression;

Args = Real | String | Args,",",Args;
     | "{", Name,"=>",Args, [{",", Name,"=>",Args}], "}"
     | "[", (Real|String), [{",", (Real|String)}], "]";

Option = Name, "(",[Args],")",[";"];
Action = Name, "(",[Args],")",[";"];

(* block defintions *)
ParameterBlock = "begin parameters", NewLine,
                 {[LineLabel], ParameterDefn, NewLine},
                 "end molecule types", NewLine;

MoleculeTypeBlock = "begin molecule types", NewLine,
                    {[LineLabel], MoleculeType, NewLine},
                    "end molecule types", NewLine;

CompartmentBlock = "begin compartments", NewLine,
                   {   [LineLabel], Compartment, WS, ("2"|"3"),
                       WS, MathExpression, [WS, Name], NewLine    },
                   "end comparments types", NewLine;
```

```
SpeciesBlock = "begin seed species", NewLine,
               {[LineLabel], Species, WS, MathExpression, NewLine},
               "end seed species", NewLine
             | "begin species", NewLine,
               {[LineLabel], Species, WS, MathExpression, NewLine},
               "end seed species", NewLine;

ObservableBlock = "begin observables", {NewLine},
                  {[LineLabel], Observable, NewLine},
                  "end observables", NewLine;

GroupBlock = "begin groups", NewLine,
             {[LineLabel], Group, NewLine},
             "end groups", NewLine;

EnergyPatternBlock =
    "begin energy patterns", NewLine,
    {[LineLabel], Pattern, [WS], MathExpression, NewLine},
    "end energy patterns", NewLine;

FunctionBlock = "begin functions", NewLine,
                {[LineLabel], FunctionDefn, NewLine},
                "end functions", NewLine;

ReactionRuleBlock = "begin reaction rules", NewLine,
                    {[LineLabel], Rule, NewLine},
                    "end reaction rules", NewLine;

ReactionBlock = "begin reactions", NewLine,
                {[LineLabel], Reaction, NewLine},
                "end reactions", NewLine;

PopulationMapBlock = "begin population maps", NewLine,
                     {[LineLabel], PopulationMap, NewLine},
                     "end parameters", NewLine;

ActionBlock = "begin actions", NewLine,
              {[LineLabel], Action, NewLine},
              "end actions", NewLine;

Block = ParameterBlock      | MoleculesTypeBlock | CompartmentBlock
      | SpeciesBlock        | ObservableBlock    | GroupBlock
      | EnergyPatternBlock  | FunctionBlock      | ReactionRuleBlock
      | ReactionBlock       | PopulationMapBlock | ActionBlock;

Model = [{Option}],
        (   {Block}
          | "begin model", NewLine,
            {Block}
            "end model", NewLine
        ),
        [{Action}];
```