TotemBioNet User Manual

Hélène Collavizza

TotemBioNet User Manual

Hélène Collavizza

University Côte d'Azur, I3S laboratory, UMR CNRS 7271, CS 40121,
06903 Sophia Antipolis Cedex, France
helene.collavizza@univ-cotedazur.fr

18 décembre 2020

`https://gitlab.com/totembionet/totembionet`
Version V2.3.1

Preamble `TotemBionet` *is an ongoing research platform and, as such, there are always parts of it in beta version.* `TotemBionet` *is old-school fashion, with Makefile, no Doker, no Maven ...* Parts that have been added or updated in this version are labeled with *NEW*.

1 Introduction

This manual is a short user manual for `TotemBionet` users. `TotemBionet` is a tool dedicated to *parameter identification* for Biological Regulatory Networks (BRN), using the discrete formalism of René Thomas [Td06]. Given a BRN and some properties on the dynamics of the system, `TotemBionet` outputs all the parameterizations which satisfy these dynamic properties.

`TotemBionet` exploits the complementarity of two approaches : *temporal logic* and *Genetically Modified Hoare Logic*, see [BCK+19]. The benefit is twofold.

On the one hand, the power of expression is reinforced. For the same case study, reachability properties such as multi-stationarity or homeostasis can be expressed by temporal logic, as well as path properties by Hoare logic. On the other hand, performance is improved because the weakest precondition calculated by Hoare logic is used to reduce the search space.

This document is organized as follows. We first recall the main principles of the René Thomas discrete formalization of BRN and state the parameter identification problem. Then we give an overview of `TotemBionet` framework. Then we detail syntax, commands, and other functionalities provided by the platform. We do not enter into the details of the *Genetically Modified Hoare Logic*, the reader is invited to refer to [BCK+19].

2 René Thomas descrete formalization of BRN

The logical modeling defined in [Td06] to describe and analyze the biological regulation networks, relies on a representation of the system in terms of *graphs*. The *static* description of the system is defined by an *influence graph* while the *dynamic* behaviour of the system is a *transition graph*.

2.1 Influence graph

The *influence graph* describes the individual influence of one gene to the others.

Influence graph The **vertices** of the *influence graph* are the genes. An **edge** between v_1 and v_2 means that v_1 has an influence on v_2 : in a certain state, that is to say when the value of v_1 has exceeded a certain *threshold*, v_1 influences the state of v_2.

The **domains** of the variables are *discrete* domains $[\![0, b_v]\!]$ [1] with $b_v \leqslant d^+(v)$ where $d^+(v)$ is the outgoing degree of v.

Influence graph with multiplexe An *influence graph with multiplexe* is defined by $\Sigma = (V \cup M, A)$ where $V \cup M$ is the set of vertices and A is the set of edges , with :

- V, the set of **variables** v of discrete domains $[\![0, b_V]\!]$,
- M, the set of **multiplexes** m with associated multiplexe formulas φ_m. φ_m are recursively defined as follows. Atoms are identifiers of M and atomic formulas of the form $(v \geqslant n)$ with $v \in V$ and $n \in [\![0, b_v]\!]$. Formulas are $\neg\varphi$, $\varphi_1 \Box \varphi_2$ where φ, φ_1 and φ_2 are multiplexe formulas and $\Box$ is either $\wedge$, $\vee$ or $\Rightarrow$.
- $A \subset (M \times V)$, the set of **edges**.

1. $[\![a, b]\!] = \{n \in \mathbb{N} \mid a \leqslant n \leq b\}$.

Parameters The *parameters* associated with the variable v are denoted $K_{v,\omega}$ where ω is a subset of the predecessors of v (i.e. $\omega \subseteq \Sigma^{-1}(v)$ with $\Sigma^{-1}(v)$ the set of predecessors of v in the influence graph).

Parameters allow to represent a combination of influences for a variable v. For example, when v has two predecessors m_1 and m_2, K_v represents the case where v is not influenced at all, K_{v,m_1} the case where v is influenced by m_1, K_{v,m_2} the case where v is influenced by m_2, and K_{v,m_1,m_2} the case where v is influenced both by m_1 and m_2.

2.2 Asynchronous transition graph

In the R. Thomas modelization, the *dynamics* of the system is defined by an *asynchronous transition graph* which defines the next state for each state of the system. The graph is *asynchronous* in the sense that when two transitions are possible, only one is non-deterministically selected to effect the state transition.

State A *state* η of an influence graph is an *interpretation* which associates with each variable v of V a value in $[\![0, b_v]\!]$. We denote ζ the set of states of the system, and η_v the value of v for the interpretation η.

Regulatory network A *regulatory network* $(\Sigma, \mathcal{K})$ is the couple Σ associated with the parameter family $\mathcal{K} = \{K_{v,\omega} | v \in V and\ \omega \subseteq \Sigma^{-1}(v)\}$.

Applicable parameter Let $\eta \in \zeta$ be a state of an influence graph Σ and v a variable in V. The *applicable parameter* of v on state η is the only parameter $K_{v,\omega}$ where ω is the set of variables m_i such as each formula φ_{m_i} associated with edge $(m_i, v), m_i \in \omega$ is true in state η [2].

The *applicable parameter* gives the next state of the variable v in the state η. It is the value towards which v tends, and is sometimes called the *focal value* of v.

Transition graph A *transition graph* associated with the regulation network $(\Sigma, \mathcal{K})$ is defined as follows :

— The set of vertices is the set of possible states ζ

— The transitions between a state $\eta = (\eta_{v_1}, \eta_{v_2}, \ldots, \eta_{v_n})$ and the next state $\eta' = (\eta'_{v_1}, \eta'_{v_2}, \ldots, \eta'_{v_n})$ are such that :

1. There is a unique i such as $\eta_{v_i} \neq \eta'_{v_i}$,
2. Let v_i such that $\eta_{v_i} \neq \eta'_{v_i}$, and $K_{v_i,\omega}$ the applicable parameter of v_i on η. We have $K_{v_i,\omega} \neq \eta_{v_i}$ and if $K_{v_i,\omega} > \eta_{v_i}$ then $\eta'_{v_i} = \eta_{v_i} + 1$, if $K_{v_i,\omega} < \eta_{v_i}$ then $\eta'_{v_i} = \eta_{v_i} - 1$.

The first point expresses that a single variable changes its state at each transition. The second point that the changes are made in steps of one, getting closer to the value of the applicable parameter.

2. True is understood here with the interpretation η for variables and the usual meaning of $\geqslant$ in $\mathbb{N}$.

2.3 Parameter identification

Since the values of variables depend on the value of K, there is not a unique *transition graph* for a given regulatory network, but as many graphs as correct interpretations of the $\mathcal{K}$. A *correct interpretation* of K_v assigns to K_v a value less or equal to b_v.

The *parameter identification problem* consists to find all parameterizations (i.e. correct interpretations of the K) such that some dynamic properties are verified on the associated transition graph. This is the aim of the `TotemBionet` tool.

3 TotemBioNet Overview

`TotemBionet` is the successor to `SMBionet`, written by Adrien Richard [KCRB09]. `SMBionet` has been used to model and identify parameters for many case studies. For example, the production of mucus in *Pseudomonas aeruginosa* [PC03], the control of immunity of bacteriophage lambda [RCB06], the analysis of the apoptosis and tail growth of tadpole [KCRB09], and more recently to support the biologist's expertise in the identification of parameters of a model of energy metabolism regulation [KBT17].

`TotemBionet` integrates `Hoare-fol` written by Maxime Folshette, see [Fol19] and file `doc/Hoare-folTool.pdf`. `Hoare-fol` implements the calculus of the weakest precondition of the *Genetically Modified Hoare Logic* (*GMHL* for short). Inference rules of *GMHL* are applied to calculate the weakest precondition. Then, if a precondition is provided, it is propagated to simplify the weakest precondition using formal simplification rules such as $a \wedge False \Rightarrow False$.

3.1 Properties on the dynamics of the system

Properties on the dynamics of the system may be :

— information on parameters (known values or domains),
— CTL properties,
— FAIRCTL properties. A FAIRCTL property is a CTL property on infinite fair paths. A *fair path* is such that if a vertex v occurs infinitely often in p, then all the edges starting from v occur infinitely often in p. For more details, see the note by Adrien Richard in `doc` directory.
— a HOARE triple $\{P\}$ *trace* $\{Q\}$, where P is an initial operable state, *trace* is an observation trace and Q is a final obervable state.

`TotemBionet` uses these dynamic properties in the following way :

— Information on parameter values are used to reduce the domain of parameters, potentially to only one value,
— When no property is given, all possible parameterizations are computed,
— When many properties are provided, the conjunction of the properties is used,
— When a HOARE triple is provided, the weakest pre-conditon is computed and simplified. If this simplified pre-condition *swp* only contains condi-

tons on parameters, it is used to reduce the state space. Three cases can occur :

1. *swp* has been reduced to $False$. The triplet is inconsistent and there is no model,
2. *swp* is a conjunction of atomic formulas of the form $K \square s$ where $\square$ is a comparison operator and $0 \leq s \leq K_b$ with K_b the maximum bound of parameter K. In the *GMHL*, such a formula characterizes the settings whose dynamics are consistent with the observed triple. In that case, the $Hoare$ formulas are used to reduce the parameter domains,
3. *swp* is a disjunction of formulas. In the *GMHL*, such formulas are obtained when the trace contains existential quantifiers or conditions. In that case, the formulas are evaluated on the fly during the enumeration process : only the parameterizations which satisfy the $Hoare$ formulas are generated.

3.2 `TotemBionet` core

The initialization step builds the internal data structure from a `SMB` file. A `SMB` file contains some blocks to define the BRN and some optional blocks with dynamic properties (see 4). In SMB format, the influence graph is defined as a set of variables and a set of regulations. A *regulation* is an edge from a multiplexe to a variable.

One important point is that the set of parameter variables is deduced from the set of regulations. Let v a variable which is the target of the regulations $R = \{reg_1, reg_2, ..., reg_n\}$. Then the set of parameters K_v for v is the set $\{K_{v:\omega} \ s.t. \ \omega \subseteq \mathcal{P}(R)\}$ where $\mathcal{P}(R)$ the set of parts of R. In the `SMB` syntax, regulations are separated with the symbol :. Possible parameters for v are for example K_v, $K_{v:reg_1}$, $K_{v:reg_1:reg_3}$. For the sake of efficiency, only *effective* parameters are generated. A parameter is *effective* if there is at least a state where the multiplexe of the regulation is valid [3].

If a HOARE block exists, `Hoare-fol` is called to compute the simplified weakest precondition *swp*. If *swp* is a conjunction of atomic formulas (see 2 in subsection 3.1), then domains of parameters are reduced. If not, the flag *dynHoare* is set to indicates that the enumeration process must check the *swp*.

The main loop of `TotemBionet` consists in enumerating all possible parameterizations compatible with the BRN and known information on parameters. By default, the parameterizations respect the Snoussi monotonicity property :

$$\forall v \in V, \quad \forall \omega \subseteq \Sigma^{-1}(v), \quad \forall \omega' \subseteq \Sigma^{-1}(v), \qquad \omega \subseteq \omega' \ \Rightarrow \ K_{v,\omega} \leq K_{v,\omega'}$$

The user can indicate that it doesn't want to use the Snoussi condition on a variable (see 4).

For each parameterization K, (which satisfies *swp* when flag *dynHoare* is on), a non deterministic automatom is built. This automatom is a translation

3. Option `-paras` displays effective and non effective parameters of a network, see 7.

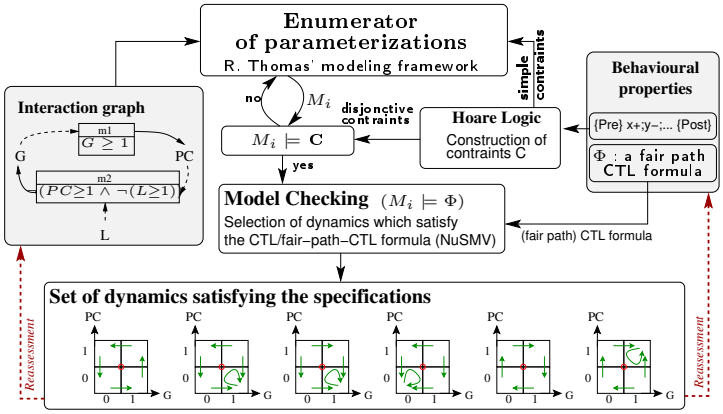

FIGURE 1 – `TotemBionet` processing flow

into a Kripke structure of the asynchronous transition graph associated with K. This automatom and some CTL formulas are provided to the model-checker `NuSMV` [NuS15]. If the CTL property is valid, then the parameterization K is a model. The global `TotemBionet` process is illustrated in Figure 1.

3.3 Running example : mucus production of P. aeruginosa

This example is fully described in [BCS14], we encourage the reader to refer to it for more details. An operon activates several genes : $AlgU$, $MucB$ and some other ones that participate to mucus production. The protein of $AlgU$ is a transcription factor that activates the operon, so the operon activates itself.

Influence graph : We consider the simple influence graph with two variables, $Operon$ to represent the operon and $MucuB$ to represent the production of mucus (see figure 2). The multiplex alg represents the production of alginate $AlgU$ by the operon at the threshold 2. The edge between $Operon$ and $MucuB$ represents the production of $MucB$ by the operon at the threshold 1. The multiplex $free$ represents the absence of the inhibition induced by $MucuB$ at the threshold 1. By convention, we denote $v \geq s$ (or $+s$) an *activation* of variable v at threshold s and $!(v \geq s)$ (or $-s$) an *inhibition* of variable v at threshold s.

Property on the dynamics of the system : [4] It has been established that P. aeruginosa exhibits two attraction basins. The first one produces mucus and the second does not. The production of mucus takes place at a level of operon greater than some unknown threshold. So, necessarily, the system produces mucus if the expression level of operon is equal to its maximal value 2 and it does not if it is equal to 0. We can express the two possible behaviours by the following CTL formulae :

4. This paragraph is a copy/paste of [BCS14] page 21.

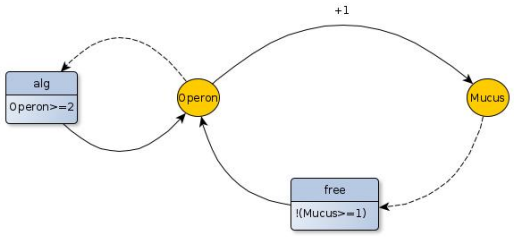

FIGURE 2 – Running example : mucus production

$Operon = 0 \Rightarrow AG(\neg(Operon = 2))$
$Operon = 2 \Rightarrow AG(\neg(Operon = 0))$

The mucus operon example with several modelings and temporal properties can be found in directory examples/mucusOperon.

4 TotemBionet syntax

TotemBionet uses text files in SMB syntax. Using text files is very light, and allows to use system facilities such as diff, grep, ... However, the user can define the BRN using the *yEd* graphical editor. The TotemBionet platform provides a facility for translating *.graphml* format to SMB format (see 6).

We describe here step by step the several parts of the grammar using the running example[5]. Other examples can be found in directory examples.The complete SMB grammar can be found in file TotemBioNet/src/front/smbParser /grammar/RRG.g4.

SMB files structure. A SMB file contains an optional environment variable declaration bloc, a variable declaration block and a regulation block. These 3 blocks define the BRN. The 4 next blocks are about dynamic properties and are all optional. The SMB file is ended by keyword END. The order of the blocks must be respected.

```
prog          : env_var_block?
                var_block
                reg_block
                init_block?
                para_block?
                hoare_block*
                ctl_block*
                'END'

/** Key words for SMB blocks **/
```

5. The syntax for describing the grammar is usual :? means optional, + means at least one and * means zero or many.

```
KENV_VAR    : 'ENV_VAR';
KVAR        : 'VAR';
KINIT       : 'INIT';
KREG        : 'REG';
KPARA       : 'PARA';
KCTL        : ('CTL'|'FAIRCTL');
KHOARE      : 'HOARE';
KPRE        : 'PRE';
KTRACE      : 'TRACE';
KPOST       : 'POST';
COMMENT     : '#' ~('\n')*
```

Variables Variables can be either system variables (e.g. genes) or *environment* variables. An *environment* variable is used to freeze the value of some input, to enforce some particular conditions on the environmenent. *Environment* variables can't be used as target of regulations, thus no parameters are associated with environment variables. As a consequence, the state space explored to enumerate the parameters is reduced.

Domains of variables are intervals denoted *min..max*. The domain of an environment variable is reduced to only one value. The key word `(NS)` idicates that the enumeration process will not use Snoussi conditions for parameters of this variable.

```
ID          : [a-zA-Z][a-zA-Z0-9_]*;
NS          : '(NS)'; /** no Snoussi **/
```

```
env_var_block : KENV_VAR env_var_decl+
env_var_decl  : ID EQ NUM SEMI
var_block     : KVAR var_decl+
var_decl      : ID EQ NUM '..' NUM NS? SEMI
```

```
/** Example: BRN of mucus operon **/
/** There are two variables **/
/** Parameters for Mucus will be enumerated **/
/** without the Snoussi monotonicity conditions **/
VAR
Mucus = 0..1 (NS) ;
Operon = 0..2 ;
```

Regulations A regulation is defined with an identifier, a multiplexe given inside brackets, and one or many target variables. A regulation name can be used to define another regulation. In that case, its multiplexe is used.

```
reg_block     : KREG reg_decl+
reg_decl      : ID '['reg_expr']' CIBLE ID+ SEMI
reg_expr      : reg_expr BOOL_OP reg_expr          # expr_bool_op
              | NEG reg_expr                       # expr_neg
              | '('reg_expr')'                     # expr_brackets
              | ID SEUIL NUM                       # expr_atome
              | ID                                 # expr_mux_name
```

```
/** Example: BRN of mucus operon **/
/** The regulation named Operon_To_Mucus **/
/** has automatically been generated from the .graphml file **/
VAR
Mucus = 0..1 ;
Operon = 0..2 ;

REG
Operon_To_Mucus [(Operon>=1)] => Mucus ;
alg [Operon>=2] => Operon ;
free [!(Mucus>=1)] => Operon ;

END
```

NEW **Initial conditions for model-checking** The INIT block is optional. It can be used to give the initial conditions of model-checking. These conditions are written in the `INIT` block of `NuSMV` file. When there is an `INIT` block, `NuSMV` checks if the CTL formula is true only for states which satisfy the initial conditions. On the contrary, when there is no `INIT` block, `NuSMV` checks if the CTL formula is true for *all* possible initial states.

In the next example, checking formula `AG(!(operon = 2))` with INIT block containing `operon=0;` gives the same result as checking the formula `((operon = 0) -> AG(!(operon = 2)))` with no `INIT` block.

```
/** Example: BRN of mucus operon **/
/** The regulation named Operon_To_Mucus **/
/** has automatically been generated from the .graphml file **/
VAR
Mucus = 0..1 ;
Operon = 0..2 ;

REG
Operon_To_Mucus [(Operon>=1)] => Mucus ;
alg [Operon>=2] => Operon ;
free [!(Mucus>=1)] => Operon ;

# initial conditions for model-checking
INIT
operon=0;

CTL

# steady states for operon
pinit = AX(EX(!(operon = 2)));

# pinit + INIT block is equivalent to p
# p = ((operon = 0) -> AX(EX(!(operon = 2))));

END
```

Parameters The PARA block is optional. It can be used to give some information on parameters. Parameter identifiers for variable v are denoted K_v :r1 :r2 :rn where r_i are the identifiers of regulations whose v is a target.

```
KID         : 'K_'[a-zA-Z][a-zA-Z0-9_]*(':'[a-zA-Z][a-zA-Z0-9_]*)*;
para_block    : KPARA para_decl*
para_decl     : KID '=' NUM ('..' NUM)? SEMI

/** Example: BRN of mucus operon **/
/** With mucus as environment variable and a PARA block **/
ENV_VAR
Mucus = 0;

VAR
Operon = 0..2 ;

REG
alg [Operon>=2] => Operon ;
free [!(Mucus>=1)] => Operon ;

PARA
K_Operon=0..1;
K_Operon:alg=2;

END
```

Hoare The HOARE block allows to specify a *Hoare* triple. The pre-condition is a set of equality relations. The post-condition is a set of logical conditions. The trace may be an assignment, an increment (decrement), a conditional statement, a forall or exist quantifier and an assertion (see [BCK+19] for the details). **WARNING :** conditional statements and assertions have not been intensively tested and used for the moment.

```
hoare_block    : KHOARE hoare_decl+
hoare_decl      : (ID '=')?
                   hoare_pre_decl
                   hoare_trace_decl
                    hoare_post_decl
hoare_pre_decl  : KPRE ':' '{' (ID EQ NUM (',')? )* '}'
hoare_post_decl : KPOST ':' '{' (hoare_assert (',')? )* '}'
hoare_trace_decl: KTRACE ':' hoare_trace SEMI
hoare_trace         : 'Skip'
                    | ID ':=' NUM
                    | ID '+'
                    | ID '-'
                    | hoare_trace SEMI hoare_trace
                    | 'If' hoare_assert 'Then' hoare_trace ('Else' hoare_trace)?
                    | 'While' hoare_assert 'With' hoare_assert 'Do' hoare_trace
                    | 'Forall' '(' hoare_trace  (',' hoare_trace)* ')'
```

```
                    | 'Exists' '(' hoare_trace (',' hoare_trace)* ')'
                    | 'Assert' '(' hoare_assert ')'
                    | '(' hoare_trace ')'
hoare_term          : (NUM|ID|KID)
                    | hoare_term OPER hoare_term
                    | '(' hoare_term ')'
hoare_assert        : hoare_term (EQ|COMP|SEUIL) hoare_term
                    | '(' hoare_assert ')'
                    | hoare_assert (BOOL_OP|IMPL) hoare_assert
                    | NEG hoare_assert
```

NEW Many HOARE blocks can be used (and can be named). When there are many Hoare blocks, they are handled in the following way :

- if all the triplets contain some disjunction, the conjunction of the triplets are checked for each parameterization and if it is satisfied, then the NuSMV automatom is built and checked,
- if one of the triplet is a conjunction, then it is used to reduce the bounds of the parameter variables (and and by the way to optimize the enumeration process),
- if the bound reduction for one of the triplets is incoherent with the bounds obtained for a previous triplet, then the set of Hoare triplets is inconsitent, and there is no model.

```
/** Example: BRN of mucus operon **/
/** With exists Hoare block and CTL **/
VAR
operon = 0 .. 2 ;
mucuB = 0 .. 1 ;

REG
prod [(operon>=1)] => mucuB;
free [!(mucuB>=1)] => operon;
alg [(operon>=2)] => operon;

# Hoare triple
HOARE
PRE : {mucuB=0,operon=1}
TRACE : Exists(mucuB+,mucuB-);operon-;
POST : {operon=0}

CTL
op1 = ((operon = 0) -> AG(!(operon = 2)));
op2 = ((operon = 2) -> AG(!(operon = 0)));

END

/** Case with two inconsistent Hoare triples **/
# Hoare triples
HOARE
```

```
triple_1=
 PRE : {mucuB=0,operon=1}
 TRACE : mucuB+;operon-;
 POST : {mucuB=1,operon=0}

triple_2=
 PRE : {mucuB=0,operon=0}
 TRACE : mucuB+;operon+;operon-;
 POST : {mucuB=1,operon=0}

  /** Result of TotemBioNet **/

*** Using Hoare formulas to reduce parameter domains...
Triple triple_1, set of formulas: [(K_operon<1), (K_mucuB:prod>0)]
   Domains have been successfully reduced.
Triple triple_2, set of formulas: [(K_operon<1), (K_operon>0), (K_mucuB>0)]
Hoare constraint (K_operon>0) is inconsitent with current value of
parameter Koperon. Intervals for K_operon : [0]
   *** Hoare failure
```

Operators We use the notations below for logical operators. The temporal operators follow the usual syntax of CTL logic. A name can be associated with a CTL formula. In that case, this name is used to give an explanation of the error for parameterizations which do not satisfy the property. This is particulary useful when there are many temporal properties (use verbosity level 2 to print this error explanation in the output file).

```
/** Logical operators **/
IMPL        : '->';
CIBLE       : '=>';
COMP        : ('<='|'>'|'<');
EQ          : '=';
SEUIL       : '>=';
SEMI        : ';';
NEG         : '!';
BOOL_OP     : ('&'|'|');
OPER        : ('+'|'-');
/****** CTL ********/
ctl_block     : KCTL ctl_decl+
ctl_decl      : (ID '=')? ctl SEMI
ctl           : ctl IMPL ctl                        # ctl_impl
              | ctl BOOL_OP ctl                     # ctl_bool_op
              | '(' ctl ')'                         # ctl_brackets
              | CTL_PREFIX_1 '(' ctl ')'            # ctl_temp_op1
              | CTL_PREFIX_2 '(' ctl 'U' ctl')'     # ctl_temp_op2
              | NEG ctl                             # ctl_neg_op
              | ID (COMP|EQ|SEUIL) NUM              # ctl_atome
```

```
CTL_PREFIX_1: ('EX'|'AX'|'EF'|'AF'|'EG'|'AG');
CTL_PREFIX_2: ('E'|'A');
```

5 *NEW* Operations on sets of parameterizations using MDDs

For a same influence graph, there could be different dynamic properties depending on the environnement. These properties can be sumarized in "property matrices", as presented in [GBCC21]. In that case, it could be interesting to compute the set of models that satisfy the whole set of properties, which amounts to compute their *intersection*.

Multi-valued Decision Diagrams (MDDs) [KVBSV98] are an extension of well-known BDDs, to represent set of multi-valued variables. MDDs provide an efficient data structure to store set of datas, avoiding redundancy. They also provide useful combination operations : the negation (set of elements not in the set), the conjunction (intersection of the sets) and disjunction (union of the sets).

MDDs may be used to represent sets of parameterizations which validate some dynamic properties of the network (see figure 3. `TotemBionet` uses the MDDs library `https://github.com/colomoto/mddlib` developped by Adrien Naldi in order to study the dynamics of regulatory networks, in particular to study circuits and stable states [NTC07]. MDDs are used for two purposes :

— to compute intersection or union of sets of parameterizations from the outputs of `TotemBionet` (from .csv files),

— to output a "compact" .csv file, where variables that doesn't matter for one configuration are abstracted by their domains (see figure 4). In that case, correct parameterizations are stored in MDDs to obtain an abstract representation which are written in a compact format.

For intersection (or union), the sets of parameterizations can come from properties checked in different environments (use of `ENV_VAR` variables). In that case, some parameters can be effective for some environments and not effective for some others. The intersection is thus calculated for the whole set of parameters, and then the parameters which are not useful in the intersection (i.e. don't care values) are removed.

6 Translation from *yEd* to `SMB` format

`TotemBionet` does not provide a proper graphical user interface but it provides facilities to use *yEd* graph editor (.graphml format) to design the BRN.

The command **.\totembionet -yed input.graphml** translates the file `input.graphml` in `SMB` format into the file `inputFromYed.smb`. This functionnality is a very beta version. It requires many restrictions when editing the *yEd* file, only *yEd* files which follow these conventions are translated. **Warning :** few error messages are provided.

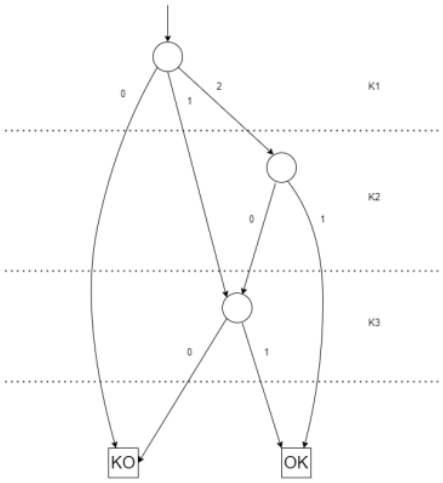

FIGURE 3 – Example of a MDD that represents a set of OK and KO parameterizations.

M	N	O	P	Q	R	S	T	U	V	W	X	Y
K_a:b_ini:sk_act2	K_a:b_ini:en_ini:sk_act2	K_b	K_b:a_act	K_b:m	K_b:a_act:m	K_ep	K_ep:b_act	K_en	K_en:a_ini_en	K_en:ep_act	K_en:sk_ini	K_en:a_ini_en:ep_act
0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
0	0..1	0	0	0	0..1	0	0..1	0	0	0	0	0..1
0	0..1	0	0	0	0..1	0	0..1	0	0	0	1	0..1
0	0..1	0	0	0	0..1	0	0..1	0	0	1	0..1	1
0	0..1	0	0	0	0..1	0	0..1	0	1	0	0	1
0	0..1	0	0	0	0..1	0	0..1	0	1	0	1	1
0	0..1	0	0	0	0..1	0	0..1	0	1	1	0..1	1
0	0..1	0	0	0	0..1	0	0..1	1	1	1	1	1
0	0..1	0	0	1	1	0	0	0	0	0	0	0..1
0	0..1	0	0	1	1	0	0	0	0	0	1	0..1
0	0..1	0	0	1	1	0	0	0	0	1	0..1	1
0	0..1	0	0	1	1	0	0	0	1	0	0	1
0	0..1	0	0	1	1	0	0	0	1	0	1	1
0	0..1	0	0	1	1	0	0	0	1	1	0..1	1
0	0..1	0	0	1	1	0	0	1	1	1	1	1

FIGURE 4 – Example of a compact representation of OK and KO parameterizations.

The requirements expected when using *yEd* editor are given below. The word "property" refers to a property of a component, in the usual sense. The properties can be found int the property view window, after a double-clic on a component.

6.1 Instructions for nodes

- variables are shape nodes, their identifiers are given in the "Text" property,
- identifiers for variables and multiplexes are `SMB` identifiers described in section 4,
- the "Fill Color" property of an environment variables must be"no color"
- multiplexes are "entityRelationship" nodes,
- multiplexes have two information : the multiplexe name at the top and the multiplexe formula at the bottom. The multiplexe name may be a name followed with a long description inside parenthesis. Information in parenthesis is just ignored.

6.2 Instructions for edges

- edges may be "PolyLine" or "ArcLines",
- edges from multiplexes to nodes are lines,
- edges from nodes to multiplexes are *dashed* lines,
- edges between nodes are lines,
- source nodes can be VAR or ENV_VAR variables, target nodes can only be VAR variables,
- a label on an edge from node to node must be the property "Text" of the edge (double clic on the edge, add the label into the property "Text"). The label must be $+n$ or $-n$ where n is an integer. An edge with label $+n$ from node x to y is translated as the regulation $x\_To\_y[(x >= n)] => y$. An edge with label $-n$ from node x to y is translated as the regulation $x\_To\_y[!(x >= n)] => y$. Be careful : the source node is the node where you first clicked when defining the edge in the *yEd* editor.

Some examples can be found in yed directories of several examples.

7 `TotemBionet` commands

Parameter identification : ./totembionet <input>.smb -<options>

- Input : <input>.smb contains influence graph and (possibly) information on parameters, temporal properties, Hoare triple
- Output : <input>.out contains two sets of pameterizations : those that satisfy the properties and the others.
- Option details
  - o name : Generate the output in <name>.out

- verbose n : Verbosity level. 1 : more information is printed (e.g. error explanation, *swp* computed with `Hoare-fol`), 2 : auxilliary files (for `NuSMV` and Hoare) are not deleted.
- csv : Generate also the output in a CSV file <input>.csv (or <name>.csv if option -o is used). *NEW* When combined with verbosity level 2, <input>-MDD.csv is also output, with the compact vision computed from its MDD representation.
- from <parameterSet>.csv *NEW* Enumerate the possible parameterizations from OK models in <parameterSet>.csv. <parameterSet>.csv must be in compatible format which means that it must contains valid parameters in the first line and their domains in the second line (thus it can be the output of a previous call to `TotemBionet`). -from <parameterSet>.csv KO does the enumeration from KO models in <parameterSet>.csv
  When combined with option -csv, the enumeration uses the OK models in <parameterSet>.csv and outputs the checked models in <input>-from.csv where<input>.smb is the input file of `TotemBionet` .

Information on the BRN : ./totembionet <input>.smb -<options>

— Option details

- paras : Print the set of effective parameters
- simu : To start simulation mode (this option calls an option of **SMBioNet** which has not been upgraded, without Hoare, without environment variables, not maintained, see **SMBioNet** user manual in the doc directory)

NEW **MDD Management** : ./totembionet <path> -<MDD-operation>
Path is a path to a directory that contains .csv files output from a previous run of `TotemBionet` on the same influence graph, with different environment variables and properties to check. MDD-operation is either "intersection" or "union". The intersection (resp. union) of the OK models in <path> are computed and written in <path>/intersection.csv (resp. <path>/union.csv). If verbosity level is 2, then the compact version is also computed in <path>/intersection-MDD.csv (resp. <path>/union-MDD.csv). By default, the MDD operation if on OK models. -<MDD-operation> KO allow to build the MDD-operation for KO models.

Translation to json : ./totembionet <input>.smb -json
Translate <input>.smb file into a JSON format in file <input>.json

Translation from *yEd* ./totembionet <input>.graphml -yed
Translate <input>.graphml into <input>FromYed.smb. <input>.graphml contains BRN information built from yEd according conventions listed in 6.

Help : ./totembionet -help
Print help guide.

8 Installation

1. Clone or download the project

   'git clone https ://gitlab.com/totembionet/totembionet.git'

2. Install dependencies

   This project is written in Java and Ocaml. The input file is parsed using antlr4, the CTL properties are checked with NuSMV.

   — java : a JDK is required to install the project, at least version 9.
   — Ocaml : at least version 4.03.0 is required see https ://ocaml.org/docs/install.html
   — NuSMV : a zipped compatible version for linux is provided in './lib/-NuSMV'. You need to unzip it. If you already use another version of NuSMV, you can give its path into the 'Makefile' file.
   — antlr4 : jar files for antlr4 are provided in './lib/antlr4'
   — TotemBioNet also provides a parser from .graphml to `SMB` format. You can download the *yEd* graphical tool here : https ://www.yworks.com/products/yed/download

3. Compile the project

   To compile the project, just enter 'make' in the root directory. Two scripts will be created

   — 'totembionet' which should be used to run TotemBioNet
   — 'ASTtreeViewer' to visualize the AST (Abstract Syntax Tree) of the .smb file

   You can test these programs with

   './totembionet examples/mucusOperon/mucusOperon.smb' to run TotemBioNet

   './ASTTreeViewer examples/mucusOperon/mucusOperon.smb' to view the AST of 'mucusOperon.smb'

   Other examples are provided in the 'examples' directory. *NEW* In particular, benchmarks discussed in [BBCC20] and [GBCC21].

4. Install the project

   Type 'make install' (eventually preceded by a 'sudo') to install the project in the install directory. The install directory is defined at the top of the 'Makefile' in the 'INSTALL_DIR' variable (defaults to '/usr/local/d.TotemBioNet').

   Once installed, eventually update your 'PATH' variable with something like 'PATH=$PATH :<TotemBioNet install directory>' to give access to the command 'totembionet' everywhere.

Nota: `TotemBionet` has been tested and developed under *Ubuntu* platform. Some problems have been found under linux on *Mac* (file names with uppercase and lowercase). The installation procedure has been updated in consequence. If there remains any problem, please contact *Helene.COLLAVIZZA@univ-cotedazur.fr*.

9 Contributors

— Adrien Richard, I3S, who wrote **SMBioNet** and established the constructive proof of the translation from *FAIR-CTL* to *CTL*. `TotemBionet` mainly uses the enumeration process of parameters written by Adrien.
— Maxime Folschette, Centrale Lille, who wrote the Ocaml code that computes the weakest precondition of a Hoare triple
— Sokhna N'deye, Master student at Polytech'Nice Sophia, who wrote the antlr4 parser. She did a very good job !
— Erick Gallésio, Polytech'Nice Sophia, who was co-advisor of the master internship of Sokhna, and wrote the installation scripts.
— Cyril Guillot, Master student at Polytech'Nice Sophia, who studied the MDD library and made the first version of it use in `TotemBionet` .
— Leatitia Gibart (Phd student) provided the examples in examples/bioinfomatics21 and Déborah Boyenval (Phd student) the examples in examples/CMSB202.
— Hélène Collavizza, Polytech'Nice Sophia, Helene.COLLAVIZZA@univ-cotedazur.fr, project manager.

Acknowledgements A big thank you to Laetitia Gibart and Déborah Boyenval, my favorite beta testers.

Références

[BBCC20] Déborah Boyenval, Gilles Bernot, Hélène Collavizza, and Jean-Paul Comet. What is a cell cycle checkpoint ? the TotemBioNet answer. In *CMSB*, pages 362–372, 2020.

[BCK+19] G. Bernot, J.-P. Comet, Z. Khalis, A. Richard, and O. F. Roux. A genetically modified Hoare logic. *Theoretical Computer Science*, 765 :145–157, 2019. https ://doi.org/10.1016/j.tcs.2018.02.003.

[BCS14] Gilles Bernot, Jean-Paul Comet, and El Houssine Snoussi. *Logical Modeling of Biological Systems*, chapter Formal methods applied to gene network modelling, pages 245–289. Bioengineering and health science series. ISTE & Wiley, ISBN 978-1-84821-680-8, 2014.

[Fol19] Maxime Folschette. The Hoare-fol Tool. Technical report, Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France, December 2019.

[GBCC21] L. Gibart, G. Bernot, H. Collavizza, and J.-P. Comet. Totembionet enrichment methodology : Application to the qualitative regulatory network of the cell metabolism. *BioInformatics, to appear*, 2021.

[KBT17] R. Khoodeeram, G. Bernot, and J.-Y. Trosset. *An Ockham Razor model of energy metabolism*, chapter Book chapter in Proc. of the Thematic Research School on Advances in Systems and Synthetic

Biology, pages 81–101. EDP Science publisher, ISBN : 978-2-7598-2116-7, 2017.

[KCRB09] Z. Khalis, J.-P. Comet, A. Richard, and G. Bernot. The SMBioNet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, 3(special issue 1) :15–22, 2009.

[KVBSV98] T. Kam, T. Villa, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Multi-valued decision diagrams : Theory and applications. *Int. J. Multiple-Valued Logic*, 4 :9–12, 1998.

[NTC07] Aurélien Naldi, Denis Thieffry, and Claudine Chaouiya. Decision diagrams for the representation and analysis of logical models of genetic networks. *CMSB 2007*, page 233–247, 2007.

[NuS15] NuSMV Team. *NuSMV Home page*, 2015. http ://-nusmv.fbk.eu/index.html.

[PC03] S. Pérès and J.-P. Comet. Contribution of computational tree logic to biological regulatory networks : example from pseudomonas aeruginosa. In *International workshop on Computational Methods in Systems Biology*, volume 2602 of *LNCS*, pages 47–56, February 24-26, 2003.

[RCB06] A. Richard, J.-P. Comet, and G. Bernot. *Modern Formal Methods and Applications*, chapter Formal Methods for Modeling Biological Regulatory Networks, pages 83–122. Springer, ISBN : 1-4020-4222-1, 2006.

[Td06] R. Thomas and R. d'Ari. Biological feedback. *CRC Press, Inc., pp.316, 1990. hal-00087681*, 2006.