Model building for Parsybone
Version 2.1

Adam Streck
Discrete Biomathematics, FU Berlin

March 19, 2014

Contents

1 Introduction

This is a manual for the Parsybone tool, whose purpose is analysis and reverse engineering of regulatory newtorks and signal transduction networks into discrete models.

This text mainly describes the modeling language and possible modeling approaches. For the compilation details see the `README` of the tool, for execution details invoke the tool with the `--help` switch.

2 Thomas network formalism

To help with understanding of concepts of the formal modeling framework used in Parsybone we demonstrate basic notions on a very simple example of a regulatory network, depicted in Figure 1. This network has two boolean components named A and B, each regulated by both itself and the other component.

The model is a boolean one, which means that there are only two activity levels for each component, roughly corresponding to the situations where its concentration is below threshold or above it. The threshold marks a concentration boundary whose crossing usually causes the component to change its regulatory effect. Boolean component therefore usually works as a switch.

Components of such a network are usually well-know, conversely to their regulatory effects that are hard to obtain from biological measurements [2]. The Parsybone is designed to solve the task of determining those effects or at least to narrow the set of posibilities thus helping with further analysis. Possible regulatory effects of a component are given by the so-called *logical parameters*. Each component has usually several logical parameters that specify towards which activity level the component inclines based on the current activity levels of its regulators e.g. a self-regulating component can stop its production when a certain concentration level is reached.

The user is required to specify the model as a set of components and their interactions. Having such a model, the Parsybone generates all possibile combinations of logical parameters for the model, creating a set of *parametrizations*. This set is usually called

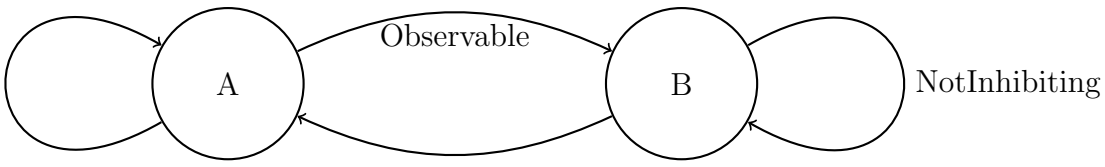

Figure 1: Simple example network with two components.

parametrization space and it can be viewed as a set of all behavioural possibilities for the model. As a second input, the Parsybone takes a specification of some behaviour the model must be able to reproduce. Parametrizations that do not allow such a behaviour are then removed from the set of possibilities.

Apart from basic boolean properties we also allow the model to be created using multiple extensions:

- Multi-valued components, allowing to change the effect of only a subset of its regulatory effects at a time.
- Edge labels, bounding possible effects of the regulations.
- Partial parametrizations, allowing to reduce the parametrization space before the analysis itself.
- Basal values, specifying general case behaviour of components.

These extension allow for more precise model description than the basic boolean network formalism as presented above, but their explanation is beyond scope of this section. In case of furter interest please refer to [3].

3 Modeling

Models are described using an internal modeling language, based on the XML syntax [1], called *PMF* (Parsybone model file). The model is provided within a single PMF file that holds specification of the regulatory network.

For description of desired properties of the dynamical system a second file type, also based on the XML, is used. Predictably the name of the format is PPF (Parsybone property file).

Both the files must abide by the general XML rules and be provided as runtime arguments with their suffixes corresponding to their data type, i.e. with the *.pmf* and *.ppf* suffixes.

3.1 Model example

Every model must be enclosed within a pair `NETWORK` tag. A detailed description of the modeling language is provided later in this section, here we present, as an example, a model file for the network depicted in the *introductory pdf*.

This example model has a quite non-uniform syntax, which has been chosen on purpose to present different possibilities of model description.

```
<NETWORK>
    <SPECIE name="A">
        <REGUL source="B" threshold ="1" label="Observable" />
```

```
        <REGUL source="A" />
    </SPECIE>
    <SPECIE name="B">
        <REGUL source="A" />
        <REGUL source="B" label="NotInhibiting" />
        <CONSTR expr="A:1,B:1 = 1" />
        <CONSTR expr="A:0,B:1 = 1" />
    </SPECIE>
</NETWORK>
```

As can be seen, the model is a structure with two species, both regulated by itself and each other. For the component B, the possible parametrizations space is reduced by requirements that the regulation from A must be observable and that the effect of its self-regulation must be positive, if any. Also, the logical parameter of self-regulation of the component B must always be 1. As a result, the parametrization space is reduced to four possibilities.

3.2 Model property

The main purpose of the tool is picking parametrizations that satisfy some property. The description of this property can be given in one of two possible ways - either as Büchi automaton or as a time series. A time series is merely a Büchi automaton specialization, but as will be explained later the Parsybone is optimized for its usage and provides additional features if the time series is employed. To demonstrate the difference between the two, we present a single property described using each formalism.

This property assures that the model depicted in the complementary *introductory pdf* is able to reproduce a time series composed of the following three measurements:

1. $A = 0$
2. $A \Leftrightarrow B$
3. $A = 1 \wedge B = 0$

Only two out of four parametrizations allow for reproduction of this time series. To obtain them, we can describe the time series either using the Büchi automaton:

```
<AUTOMATON>
    <STATE final="0">
        <EDGE target="0" values="tt" />
        <EDGE target="1" values="A=0" />
    </STATE>
    <STATE>
        <EDGE target="1" values="tt" />
```

```
        <EDGE target="last" values="(A=0 & B=0) | (A=1 & B=1)" />
    </STATE>
    <STATE name="last">
        <EDGE target="last" values="tt" />
        <EDGE target="3" values="A=1 & B=0" />
    </STATE>
    <STATE final="1">
        <EDGE target="3" values="tt" />
    </STATE>
</AUTOMATON>
```

Or using the time series directly:

```
<SERIES>
    <EXPR values="A=0" />
    <EXPR values="(A=0 & B=0) | (A=1 & B=1)" />
    <EXPR values="A=1 & B=0" />
</SERIES>
```

As can be seen, the second method makes the model quite shorter and should be used for description of time series.

3.3 Regulatory network description

- `NETWORK`
  - Occurrence: single, mandatory.
  - Type: pair.
  - Parent: none, is a root node.
  - Description: encloses the definition of a regulatory network.
  - Attributes: none.

- `CONSTRAINT`
  - Occurrence: multiple, optional.
  - Type: solo.
  - Parent: NETWORK.
  - Description: specifies additional static constraint on the parametrization space.
  - Attributes: none.
    1. *type*
       - Occurrence: mandatory.
       - Value: bound_loop.
       - Description: a nature of the constraint, for details see Section 3.7.

- `SPECIE`
  - Occurrence: multiple, mandatory.
  - Type: pair.
  - Parent: NETWORK.
  - Description: defines a single specie.
  - Attributes:
    1. *name*
       * Occurrence: optional.
       * Value: a string containing only digits, letter and underscore.
       * Description: name of the specie under which it will be further addressed.
    2. *max*
       * Occurrence: optional.
       * Value: natural number.
       * Default: 1.
       * Description: maximal activation level this specie can occur in.

- `REGUL`
  - Occurrence: multiple, mandatory.
  - Type: solo.
  - Parent: SPECIE.
  - Description: defines a single incoming regulation of the parent specie.
  - Attributes:
    1. *source*
       * Occurrence: mandatory.
       * Value: name or the ordinal number of a specie.
       * Description: name of the specie that regulates this one.
    2. *threshold*
       * Occurrence: optional.
       * Value: natural number.
       * Default: 1.
       * Description: lowest activation level of the source specie that activates the regulation.
    3. *label*
       * Occurrence: optional.
       * Value: a string, see Sec. 3.6.
       * Default: Free.
       * Description: describes nature of the regulation.

- `CONSTR`
  - Occurrence: multiple.

- Type: solo.
- Parent: SPECIE.
- Description: defines integral constraints on parameters.
- Attributes:
  1. *expr*
     * Occurrence: mandatory.
     * Value: logical formula (see Section 3.8), variables are logical parameters.
     * Description: defines constraints that has to be satisfied by the incoming regulations.

3.4 Büchi automaton description

- `AUTOMATON`
  - Occurrence: single, present if and only if there is no sibling SERIES tag.
  - Type: pair.
  - Parent: none, is a root node.
  - Description: encloses the decription of a Büchi automaton.
  - Attributes: none.

- `STATE`
  - Occurrence: multiple, mandatory.
  - Type: solo.
  - Parent: AUTOMATON.
  - Description: defines a single state of the automaton. The first state in the description is also considered to be the initial state of the automaton.
  - Attributes: none.

  1. *name*
     - Occurrence: optional.
     - Value: string containing letters and numbers.
     - Default: ordinal number of the tag, counting from zero.
     - Description: name of the specie under which it will be further addressed. System also uses its ordinal number (so the first state can be addressed using the name *0*).
  2. *final*
     - Occurrence: optional.
     - Value: Boolean.
     - Default: 0.
     - Description: specifies if the state is final (1) or not (0).

- `EDGE`
  - Occurrence: multiple, mandatory.

  - Type: solo.
  - Parent: STATE.
  - Description: defines an edge leading from the parent state.
  - Attributes:
    1. *target*
       * Occurrence: mandatory.
       * Value: name or the ordinal number of the target state.
    2. *values*
       * Occurrence: mandatory.
       * Value: logical formula (see Section 3.8), variables are atomic propositions.
       * Description: conditions that must be met for the edge to be transitive.
    3. *stable*
       * Occurrence: optional.
       * Value: Boolean.
       * Default: 0.
       * Description: If set to (1), the transition satisfying the edge must be a loop.
    4. *transient*
       * Occurrence: optional.
       * Value: Boolean.
       * Default: 0.
       * Description: If set to (1), the transition satisfying the edge must be between different states.

3.5 Time series description

- `SERIES`
  - Occurrence: single, present if and only if there is no sibling AUTOMATON tag.
  - Type: pair.
  - Parent: MODEL.
  - Description: encloses definition of a time series.
  - Attributes: none.

- `EXPR`
  - Occurrence: multiple, mandatory.
  - Type: solo.
  - Parent: SERIES.
  - Description: a single measurement in the time series.
  - Attributes:
    1. *values*

    * Occurrence: mandatory.
    * Value: logical formula (see Section 3.8), variables of the formula must be atomic propositions.
    * Description: conditions that must be met for the measurement to be reproduced.
2. *stable*
    * Occurrence: optional.
    * Value: Boolean.
    * Default: 0.
    * Description: If set to (1), the transition satisfying the edge must be a loop.
3. *transient*
    * Occurrence: optional.
    * Value: Boolean.
    * Default: 0.
    * Description: If set to (1), the transition satisfying the edge must be between different states.

3.6 Edge label

There are two basic labels:

+ Meaning that the there must be a regulation whose parameter value increases if we add this regulator.

- Has the opposite meaning.

One can compose these labels using a logical formula over $+$ and $-$ or use one of the following predefined descriptions:

- Activating: $+$
- ActivatingOnly: $(+ \wedge \neg -)$
- Inhibiting: $-$
- InhibitingOnly: $(- \wedge \neg +)$
- NotActivating: $\neg +$
- NotInhibiting: $\neg -$
- Observable: $(+ \vee -)$
- NotObservable: $(\neg + \wedge \neg -)$
- Free: $true$

3.7 Static constraints

Constraints that globally restrict the parametrization space based on certain assumptions.

bound_loop

This constraint serves purely to optimization of multi-valued models and should be always used, unless there is a reason not to.

In multi-valued models, a scenario where multiple different parameters spawn the same structure may occur, simply because the values lay outside the activity levels for the respective context. This constraint leaves only a single representative of this behavior.

3.8 Formula construction

A formula is constructed using the following set of recursive rules:

1. Depending on use, any atomic proposition or any kinetic parameter are formulae,
2. tt and ff are formulae representing true or false respectively,
3. if A is a formula, then $\neg A$ a formula,
4. if A, B are formulae, then $(A|B)$ and $(A\&B)$ are formulae representing logical disjunction and conjunction respectively,
5. nothing else is a formula.

Atomic proposition

An atomic proposition is a string of the form: $specie * value$, where:

- *specie* denotes the name or ordinal number of a specie,
- $*$ denotes comparison operator from the set $\{<, >, =\}$,
- *value* denotes a positive integer with which the value of the specie is compared.

Kinetic parameters

A kinetic parameter is a string of the form: $context * value$. For a component v with the set of regulators v^- and $T(v', v)$ the set of thresholds on multi-edge from v' to v the context is a string of the form: $v_1 : t_1, \ldots, v_n : t_n$, where $\{v_1, \ldots, v_n\} = v^-$ and $t_i \in T(v_i, v)$.

ASCII coding

Note that in the model file

- $\neg$ is denoted `!`
- $\wedge$ is denoted `&`
- $\vee$ is denoted `|`

Also, this description is not compliant with the XML standard. It is not necessary for Parsybone, but to increase portability we recommend to use:

- `&` instead &
- `<` instead $<$
- `>` instead $>$

4 Understanding the results

Parsybone computes parametrizations of the network that produce model of the property used for synthesis.

The result is a table that has the columns labeled with individual contexts in the form explained in Sec. 3.8. Each column then describes a single valid parametrization. There are additional columns for data captured during the model checking procedure as well. In the current version of Parsybone these are Cost, Robustness and Witness.

Interpretation of data is not a uniform activity and certainly not a simple one at that. Some tools for partial automation are in development as of now, but here are some general hints.

Parametrization count

There are two obvious extremes - if there is no parametrization produced it means that there is no model satisfying the property - it may be that there is simply an error in the property. If not, one may try to relax the conditions:

- Allow uncertain edges to not have an observable effect (e.g. turning ActivatingOnly into NotInhibiting).
- Add new edges with monotonic effect (NotInhibiting, NotActivating) where one expects possible regulatory effect.
- Relax the behavioral requirements - either by removing values that were rather ambiguous from the expressions or shortening the time series.

If there are on the other hand there is no reduction, one gets reassured that the model is in line with the requirements, however it is not very helpful for the synthesis process. Apart from an obvious path of trying additional properties, one can also refer to additional metrics that create ordering on parametrizations which may or may not be relevant, dependent on the exact nature of the model.

Lastly, in the special case when the size of the parametrization space is dividable by the size of remaining set or if their GCD is quite high, there is a chance that the filtering procedure removed only functions over a specific component. Such a symmetry is likely to be caused by a reduction on functions governing single target only, allowing to observe the resulting set as a product on mutually independent reductions.

Cost

Cost denotes the shortest path that is a witness of the property being satisfied. For some involved formulae this value may not make sense, however in general one is interested in the parametrizations that have low cost. Reasoning behind this value is that each discrete step is something that consumes energy of the system and in general the lower the energy consumption the better. This may not be logical when the reactions are occurring on completely different time-scales, however even in such cases low cost is probably preferable purely by the virtue of simplicity.

Witness

Parsybone can reproduce the steps on a path to accepting states. Note that there are usually many options for non-deterministic choice - to be able to capture the different paths in a reasonable space, we print out only transitions on these paths, meaning that the result is just a set of oriented edges.

What may be of particular interest are *diamond* structures, i.e. cases where there are two states in between which there are path with all the possible orderings on the changes of values. Such a structure suggests components of the network with correlated expression but with no mutual dependencies and may also point out possible reduction point.

Robustness

Last value available to the user is the robustness metric which computes the probability of reaching an accepting state if a random walk of a fixed length is taken.

The greater the value is, the more deterministic is the system in reaching accepting states. Two models with the same cost but different robustness mean that the one that has a smaller robustness has more chances to diverge from its path to the accepting states.

In case we expect the system to be serve only one function, high robustness is welcome, however in systems like pluripotent cells, high non-determinism is more preferable.

Lastly note that our framework is inherently very non-deterministic, therefore the robustness is likely to be very low in general and decrease geometrically as the cost increases.

References

[1] T. Bray, J. Paoli, C. M. Sperberg-Mcqueen, Eve, and F. Yergeau, editors. *Extensible Markup Language (XML) 1.0*. W3C Recommendation. W3C, fourth edition, Aug. 2003.

[2] M. Hecker, S. Lambeck, S. Toepfer, E. van Someren, and R. Guthke. Gene regulatory network inference: Data integration in dynamic models—a review. *Biosystems*, 96(1):86 – 103, 2009.

[3] H. Klarner, A. Streck, D. Safranek, J. Kolcak, and H. Siebert. Parameter identification and model ranking of thomas networks. Technical Report FIMU-RS-2012-03, Masaryk University, 2012.