Petri Nets Semantics of Reaction Rules (RR) A Language for Ecosystems Modelling

Franck Pommereau, Colin Thomas, Cédric Gaucherel

Petri Nets Semantics of Reaction Rules (RR)

A Language for Ecosystems Modelling

Franck Pommereau[1,2][0000−0002−9959−3699], Colin Thomas[1,3][0000−0002−7650−784X], and Cédric Gaucherel[3][0000−0002−4521−8914]

[1] IBISC, Univ. Évry, Univ. Paris-Saclay, 91020 Évry-Courcouronne, France
[2] franck.pommereau@univ-evry.fr
[3] AMAP-INRA, CIRAD, CNRS, IRD, Univ. Montpellier, 34398 Montpellier, France

Abstract. The EDEN framework provides formal modelling and analysis tools to study ecosystems. At the heart of the framework is the *reaction rules* (RR) modelling language, that is equipped with an operational semantics and can be translated into Petri nets with equivalent semantics. In this paper, we formally define the RR language and its semantics, detailing the initial definition from [8] and extending it with a notion of *constraints* that allows to model mandatory events. Then, we consider in turn two classes of Petri nets: *priority Petri nets* (PPN), which are safe place/transition Petri nets equipped with transitions priorities, and *extended Petri nets* (EPN) which are PPN further extended with read arcs, inhibitor arcs, and reset arcs. For each of these classes, we define the translation of an RR system into a Petri net and prove that the state-space generated with the RR operational semantics is equivalent to the marking graph of the Petri net resulting from the translation. We use a very strong notion of equivalence by considering *labelled transition systems* (LTS) isomophism with states and labels matching.

1 Introduction

The framework EDEN has been developed and used for ecological studies for more than five years [4–8, 12]. It provides tools and methods to formally model ecosystems, and analyse them through an interactive method that lets the users explore their models dynamics and draw understanding progressively. Properties of interest include searching the root causes (events, conditions, or states) leading to trajectories of interest, structural stabilities, or collapses. In addition to the published works, EDEN has been used by more than a dozen Master interns in ecology, who modelled and analysed varied ecosystems.

At the heart of EDEN is the *reaction rules* modelling language (RR). An RR system consists of (1) a set of *Boolean variables* representing the functional presence (on) or absence (off) of an entity in an ecosystem, and (2) a set of *actions* representing the possible events that lead to observable changes in the ecosystem (*i.e.* assign new values to variables). A species is functionally present if its presence enables observable effects on the ecosystem, otherwise it is functionally absent. Actions are divided into *constraints* and *rules*, the only difference being that

```
inhabitants:
    Rp+: reproductives
    Wk-: workers
    Sd-: soldiers
    Te-: termitomyces (fungi)
structures:
    Ec-: egg chambers
    Fg-: fungal gardens
resources:
    Wd-: wood
competitors:
    Ac*: ant competitors
```

```
constraints:
  Fg- >> Te-                    # C1
rules:
  Rp+ >> Ec+                    # R1
  Rp+, Ec+ >> Wk+               # R2
  Wk+ >> Wd+, Te+, Fg+, Ec+     # R3
  Wk+, Wd+ >> Sd+, Rp+          # R4
  Wk+, Te+ >> Wd-               # R5
  Wd- >> Wk-, Te-               # R6
  Wk- >> Fg-, Sd-               # R7
  Wk-, Rp- >> Ec-               # R8
  Ac+, Sd- >> Wk-, Rp-          # R9
```

Fig. 1. A toy model of a termite colony, adapted from [8]. Variables are defined in the left column, dispatched into user-chosen sections (that play the role of comments). Each variable is given a name, an initial value (`+` for `on`, `-` for `off`, `*` for both values), and a description. For instance, variable `Rp` is initially `on` and models the reproductives (queen and king). Constraints and rules, collectively referred to as actions, are defined in the right column, each has a left-hand side that corresponds to its guard and a right-hand side that corresponds to its effect. We have named the actions for reference using a comment "# ...". For instance, rule `R9` states that if `Ac` is `on` and `Sd` is `off`, then rule `R9` may be executed, yielding a state in which `Wk` and `Rp` are set to `off`.

the former have the priority over the latter (*i.e.* no rule can be executed if a constraint can). Constraints are useful in particular to model cascading events or transient states. For instance, if a pond dries its inhabitants will rapidly die. Such a situation may be modelled by a constraint, and the state where the pond is dry but its inhabitants are still present has to be transient. RR models can be seen as an analogue for ecology to Boolean networks for systems biology [21], but with important differences that will be underlined later on. An example of an RR system is given in Figure 1. Note that an RR system may have several initial states, for instance this one has two (one with `Ac+` and another with `Ac-`).

The contribution of this paper is graphically summarised in Figure 2. In Section 3, we give a formal definition of RR systems, detailing the initial definition from [8] and enriching it with the notion of constraints as well as with the possibility to have more than one initial state. We then define the semantics of RR systems in terms of *labelled transitions systems* (LTS). At the end of Section 3, we define two transformations on an RR system, *normalisation* and *elementarisation*, and we prove that they generate RR systems whose semantics are equivalent to that of the original RR system. In Section 4, we define the translation of *elementary* RR systems into *priority Petri nets* (PPN: regular Petri nets extended with transitions priorities), and we prove that the nets resulting from this translation yield *marking graphs* (MG) that are equivalent to the semantics of the translated elementary RR system. This corresponds to the right-most column of the diagram depicted in Figure 2. Finally, in Section 5, we define the translation of *normal* RR systems into *extended Petri nets* (EPN: PPN further extended with read-, inhibitor- and reset-arcs), and we prove that

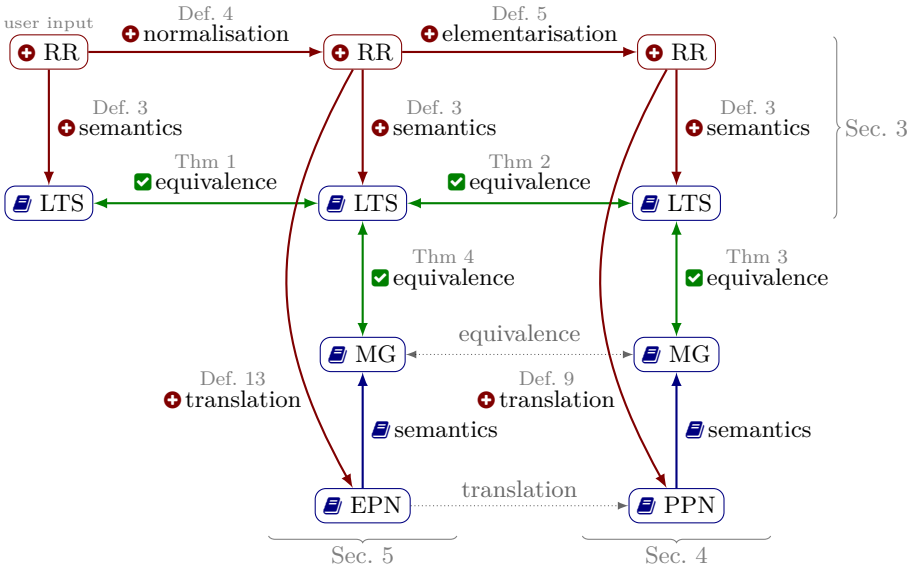

Fig. 2. Visual summary of the paper, where ✪ marks the elements we define, 🗏 those that exist already in the literature, and ☑ those we prove. RR stands for *reaction rules*, LTS for *labelled transition systems*, MG for *marking graph*, EPN for *extended Petri nets*, and PPN for *priority Petri nets*.

the nets resulting from this translation yield marking graphs that are equivalent to the semantics of the translated normal RR system. This corresponds to the middle column of the diagram depicted in Figure 2.

From the proved equivalences, we also have that the marking graph of an EPN translated from a normal RR system is equivalent to the marking graph of the PPN translated from the elementarisation of the normal RR system (upper-most gray dotted edge in the diagram from Figure 2). Finally, note that there exists a translation from EPN to PPN that is well known in the Petri net community[4] and corresponds exactly to what we use in the current paper: safe places are translated into pairs of complementary places, which allows to implement reset and inhibitor arcs as regular arcs, while read arcs are implemented with side-loops.

To start with, Section 2, provides the basic definitions upon which the rest is defined; in particular the definition of *labelled transitions systems* LTS and their equivalence through isomorphism with states and labels matching.

2 Preliminary Definitions

2.1 Boolean and Ternary Truth Valuations

Let $\mathbb{B} \stackrel{\text{df}}{=} \{\bot, \top\}$ be the set of Boolean values and $\mathbb{T} \stackrel{\text{df}}{=} \mathbb{B} \uplus \{\star\}$ be the set of truth values where $\star$ is the unknown value. We define relation $\not\simeq$ on $\mathbb{T}$ as the

[4] We did not find its formal definition and proof of correctness in the literature.

smallest symmetric binary relation such that $\top \not\simeq \bot$. Then, $\simeq$ is defined as the complement of $\not\simeq$, *i.e.*, $a \simeq b$ iff $\neg(a \not\simeq b)$.

Let b be a Boolean expression and x, y two arbitrary values, we note by $\langle b \,?\, x : y \rangle$ the ternary expression whose value is x when b holds else y.

Let V be a finite set of ordered values, an element of $\mathbb{T}^V$ can be written as a vector of values from $\mathbb{T}$ following the order in V. For $v \in V \setminus \{\max(V)\}$ we note by $\mathrm{succ}(v)$ the smallest element of V such that $v < \mathrm{succ}(v)$. Let $x, y \in \mathbb{T}^V$, we define:

- $x[v]$ for $v \in V$ is the value of x at position v;
- $x \simeq y$ iff $x[v] \simeq y[v]$ for all $v \in V$;
- for $a, b \in \mathbb{T}$, $a \triangleleft b \stackrel{\text{df}}{=} \langle b \neq \star \,?\, b : a \rangle$, that is $a \triangleleft b$ has the value of a except when $b \neq \star$ in which case it has the value of b.
- $x \triangleleft y \stackrel{\text{df}}{=} [x[v] \triangleleft y[v] \mid v \in V]$, *i.e.* we extend $\triangleleft$ on vectors component-wise;
- for $a \in \mathbb{T}$ and $v \in V$, we note by $a^{[v]}$ the element of $\mathbb{T}^V$ such that: $a^{[v]}[v] = a$ and $a^{[v]}[v'] = \star$ for all $v' \in V \setminus \{v\}$, i.e., $a^{[v]} \stackrel{\text{df}}{=} [\langle u = v \,?\, a : \star \rangle \mid u \in V]$.

x may be considered as the set of indexes where it valuates to $\top$, *i.e.*, x may be viewed as set $\{v \in V \mid x[v] = \top\}$. Operation $x \triangleleft y$ will be used to compute a new state by applying on a state x the effect y of an executed action. This is why it is defined as x except for some values that are updated as in y. Thus, $\star$ in y at some position v means that the executed action has no effect on variable v.

For example, take $x \stackrel{\text{df}}{=} [\star, \bot, \top]$, $y \stackrel{\text{df}}{=} [\bot, \top, \star]$, and $z \stackrel{\text{df}}{=} [\top, \star, \top]$ with $V \stackrel{\text{df}}{=} [u, v, w]$ we have:

- $x[u] = \star$, $x[v] = \bot$, and $x[w] = \top$;
- $x \simeq z$, but $x \not\simeq y$ because $x[v] = \bot \not\simeq y[v] = \top$;
- $x \triangleleft y = [\bot, \top, \top]$ that is y except on variable w where $y[w] = \star$ and thus we use the value of $x[w]$;
- similarly, we have $y \triangleleft z = [\top, \top, \top]$;
- $\top^{[u]} = [\top, \star, \star]$ and $\bot^{[v]} = [\star, \bot, \star]$.

2.2 Multisets

A multiset m over a domain D is a function $m : D \to \mathbb{N}$ (natural numbers), where, for $d \in D$, $m(d)$ is the number of occurrences of d in the multiset m. We note by D^* the set of all multisets over D. The empty multiset is noted by $\mathbb{0}$ and is the constant function $\mathbb{0} : D \to \{0\}$. Similarly we define $\mathbb{1} : D \to \{1\}$ the unit multiset. A set X may be used as multiset $\mathbb{1}$ over X. A multiset m over D may be naturally extended to any domain $D' \supset D$ by defining $m(d) \stackrel{\text{df}}{=} 0$ for all $d \in D' \setminus D$, which explains why we generally do not need to be precise about multisets domains. If m_1 and m_2 are two multisets over D, we define:

- $m_1 \leq m_2$ iff $m_1(d) \leq m_2(d)$ for all $d \in D$;
- $m_1 + m_2$ is the multiset over D defined by $(m_1 + m_2)(d) \stackrel{\text{df}}{=} m_1(d) + m_2(d)$ for all $d \in D$;
- $m_1 - m_2$ is the multiset over D defined by $(m_1 - m_2)(d) \stackrel{\text{df}}{=} \max(0, m_1(d) - m_2(d))$ for all $d \in D$;

– m_1/m_2 is the multiset over D defined by $(m_1/m_2)(d) \stackrel{\text{df}}{=} \langle m_2(d) = 0?0{:}m_1(d)\rangle$ for all $d \in D$. This operation nullifies m_1 where m_2 is zero. For $D' \subseteq D$ we may use m_1/D' by treating D' as a multiset as explained above.
– for $d \in D$, we note by $d \in m_1$ the fact that $m_1(d) > 0$.

2.3 Labelled Transition Systems

A labelled transition system (LTS) is a tuple $(S, I, A, \rightarrow)$ such that:

– S is the set of states;
– $I \subseteq S$ is the set of initial states;
– A is the set of labels;
– $\rightarrow \subseteq S \times A \times S$ if the set of transitions.

We note by $s \xrightarrow{a} s'$ the fact that $(s, a, s') \in \rightarrow$.

It should be stressed that we use a definition where several initial states are allowed, which will be the case for all our formalisms.

Let $L \stackrel{\text{df}}{=} (S, I, A, \rightarrow)$ and $L' \stackrel{\text{df}}{=} (S', I', A', \rightarrow')$ be two LTS, they are called equivalent through (g, h) iff:

– g is a bijection from S to S';
– h is a function from A' to A;
– $I' = g(I)$;
– for all $x, y \in S$, $x \xrightarrow{a} y$ iff $g(x) \xrightarrow{a'} g(y)$ with $h(a') = a$.

Thus g is an isomophism between the two LTS, and h defines a matching on the labels of the transitions. Assuming that L is an "original" LTS to which we compare a "transformed" LTS L', h maps every action label in L' to the original label it was obtained from. We note by id the identity function that may be later used as g. This definition results in a very strong notion of equivalence that requires matching the states as well as the transitions labels.

2.4 Regular Petri nets

A regular Petri net (RPN) is a tuple (P, T, W) where:

– P is the finite set of places, depicted as circle-shaped nodes;
– T is the finite set of transitions, depicted as rectangle- or square-shaped nodes;
– $W \in ((P \times T) \cup (T \times P))^*$ is the weight of arcs, arcs with non-zero weights are depicted as directed edges, labelled by the weight when it is greater that 1.

A marking m of a RPN is a multiset over P, $m(p)$ is called the marking of place p and is the number of tokens held by p. Tokens are depicted as black bullets $\bullet$ inside p. Given $t \in T$, we define ${}^\bullet t \stackrel{\text{df}}{=} \{p \mapsto W(p, t) \mid (p, t) \in W\} \in P^*$ that is the preset of t, and $t^\bullet \stackrel{\text{df}}{=} \{p \mapsto W(t, p) \mid (t, p) \in W\} \in P^*$ that is the postset of t. Both are multisets of places in which the multiplicity of each place p is the weight of the arc from/to t.

A transition $t \in T$ is enabled at a marking m iff ${}^\bullet t \leq m$. In such a case, t may fire, leading to the marking $m' \stackrel{\text{df}}{=} m - {}^\bullet t + t^\bullet$, which is noted by $m \xrightarrow{t} m'$. The state graph of a RPN (P, T, W) with respect to a set of initial markings $M \subseteq P^*$ is a the smallest LTS $(S, M, T, \rightarrow)$ such that $M \subseteq S$ and, if $m \xrightarrow{t} m'$ in the RPN, then $m' \in S$ and $(m, t, m') \in \rightarrow$ in the LTS as well. This LTS is generally referred to as the marking graph, or the reachability graph, but we call it the state graph to streamline the comparison between our formalisms. Similarly, we have generalised the definition to allow a set of initial markings instead of just one as it is usually the case.

A RPN is safe (or 1-safe) with respect to a set M of initial markings iff for all marking m of its state graph we have $m \leq \mathbb{1}$. In the following, all our nets will be safe so that most multisets in the definitions will be without repetitions (and thus equivalent to sets). But since we will have to prove that our nets are safe, we must state the definitions in the more general context of non-safe Petri nets.

3 Reaction Rules (RR)

3.1 Definition and Syntax

An RR system consists of a set of *Boolean variables* together with *actions* that can change the variables when their values meet the action preconditions. Actions are separated into *constraints* and *rules*, the former having a higher priority.

Definition 1 (RR systems). *An RR system is a tuple $(\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ where:*

- *$\mathcal{V}$ is a finite set of ordered variables;*
- *$\mathcal{I} \subseteq \mathbb{B}^\mathcal{V}$ is the set of initial states;*
- *$\mathcal{C}$ is a finite set of constraints;*
- *$\mathcal{R}$ is a finite set of rules, disjoint from $\mathcal{C}$;*
- *$\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ is the set of actions, and each action is a pair $(\ell, r) \in \mathbb{T}^\mathcal{V} \times \mathbb{T}^\mathcal{V}$.*

RR systems were originally defined in [8] using a concrete syntax that is easy to edit in simple text files. Formalisation was done using much heavier notations that we simplified here thanks to ternary truth values. Figure 3 shows the grammar of the concrete syntax.

In the concrete syntax, using `Ac*`, like in Figure 1, means that we have two initial states, one with `Ac+` and another with `Ac-`. So that initial states are indeed in $\mathbb{B}^\mathcal{V}$. However, the abstract syntax allows more varied initial states than it is possible to define using the concrete syntax. For instance, for two variables, it is possible to have $\mathcal{I} = \{[\top, \bot], [\bot, \top]\}$ which cannot be obtained using the concrete syntax. This is not a problem since we base all the following on the abstract syntax, but rather a practical limitation when using the concrete syntax.

3.2 Operational Semantics

The execution of an RR system is straightforward: starting from a set of initial states, we can reach new states by applying constraints first, then rules from

```
rr ::= { vardecl } [ constraints ] [ rules ]
vardecl ::= word ":" "\n" { varinit ":" line "\n" }
varinit ::= word ( "*" | "+" | "-" )
word ::= /[A-Z][A-Z-9_]*/i
line ::= /[^\n]*/
constraints ::= "constraints:\n" { action }
rules ::= "rules:\n" { action }
action ::= varstate { "," varstate } ">>" varstate { "," varstate } "\n"
varstate ::= word ( "+" | "-" )
```

Fig. 3. Concrete syntax for RR systems in BNF notation. The left-hand side (resp. right-hand side) of an actions correspond to the ℓ (resp. r) part in the definition. A variable that do not appear in one side of an action is assumed to be $\star$ so that the ℓ and r parts of the actions are fully defined using this syntax.

states where no constraint can be applied. Note that we explicitly forbid self-loops, *i.e.* action applications that would not change the state. This contrasts, in particular, with Boolean networks where self-loops are not only allowed but desired and searched for as they usually correspond to stable states of the system of interest (*e.g.* a phenotype of a cell is often modelled as such a stable state). In our setting, such states will be deadlocks.

Definition 2 (RR firing rule). *Let $(\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be an RR system, with $\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ its set of actions. Let $a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A}$ be an action and $s \in \mathbb{B}^{\mathcal{V}}$ be a state. Then:*

1. *Action a is enabled at s iff $s \simeq \ell$.*
2. *If a is a constraint, it can be fired from s yielding a new state $s' \stackrel{\text{df}}{=} s \triangleleft r$ iff it is enabled at s, which is noted by $s \xrightarrow{a} s'$.*
3. *If a is a rule, it can be fired from s yielding a new state $s' \stackrel{\text{df}}{=} s \triangleleft r$ iff it is enabled at s and no constraint in $\mathcal{C}$ is enabled at s, which is noted by $s \xrightarrow{a} s'$.*

The semantics of an RR system is expressed as expected in terms of a LTS obtained by firing actions from the initial states until saturation.

Definition 3 (RR state graph). *Let $(\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be an RR system, with $\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ its set of actions. Its state graph is the smallest LTS $(S, \mathcal{I}, \mathcal{A}, \rightarrow)$ such that $\mathcal{I} \subseteq S$ and, if $s \xrightarrow{a} s'$ in the RR system with $s \neq s'$, then $s' \in S$ and $(s, a, s') \in \rightarrow$.*

Note that we forbid self-loops in the definition of an RR state graph instead of in the definition of actions enabling. Both approaches would be correct but the one we have chosen will simplify the comparison with Petri nets state graphs. From now on, we always consider that we are within a LTS and thus we forbid firing actions when this would create a self-loop.

3.3 Normal and elementary RR systems

As defined above, the actions of an RR system have implicit elements. For instance, taking `A`, `B`, `C` as the variables, when one writes `A+, B- >> C+` in the textual

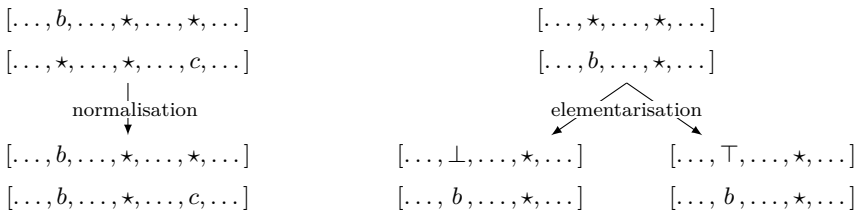

Fig. 4. Illustration of normalisation (left) and elementarisation (right), where $b, c \in \mathbb{B}$. Each action (ℓ, r) is depicted with ℓ drawn above r.

syntax, this corresponds to an action $([\top, \bot, \star], [\star, \star, \top])$ in the definition. Firing this action can be done only from state $[\top, \bot, \bot]$ (otherwise it would be a self-loop), yielding new state $[\top, \bot, \top]$, which results in $A = \top$ and $B = \bot$ while this is not explicit in the action.

An equivalent writing of the same action would be `A+, B- >> A+, B-, C+`, which explicitly specifies which values `A` and `B` get upon firing. This latter version of the action is called *normal*, *i.e.* all of its left-hand side variables appear in the right-hand side, and we show below that any action can be rewritten this way without changing the semantics.

An even more explicit writing of this action would be `A+, B-, C- >> A+, B-, C+`, which clearly states that `C` has to be $\bot$ to fire the action. Such an action, with exactly the same variables on both sides, is called *elementary*. We show below that any normal rule can be rewritten as a set of equivalent elementary rules without changing the semantics. The elementarisation of an action is a one-to-many transformation since there may exist different states from which a normal action can be fired. Consider for instance `A+ >> A+, B+, C+`. It is normal and can be fired whenever $\texttt{A} = \top$ and $(\texttt{B}, \texttt{C}) \neq (\top, \top)$, which corresponds to the three distinct states $[\top, \bot, \bot]$, $[\top, \top, \bot]$, and $[\top, \bot, \top]$ (with the fourth possibility $[\top, \top, \top]$ yielding a self-loop).

Definition 4 (normalisation). *Let $R \stackrel{\text{df}}{=} (\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be an RR system, with $\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ its set of actions. R is called normal iff for all $a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A}$ and for all $v \in \mathcal{V}$ we have: $r[v] = \star \implies \ell[v] = \star$.*

The normalisation of R, noted by $\mathsf{norm}(R)$, is obtained by replacing all its actions (ℓ, r) with $\mathsf{norm}(\ell, r) \stackrel{\text{df}}{=} (\ell, [\langle r[v] = \star \,?\, \ell[v] : r[v] \rangle \mid v \in \mathcal{V}])$.

How $\mathsf{norm}(\ell, r)$ works is illustrated on the left of Figure 4: for each $v \in \mathcal{V}$, if $r[v] = \star$ then it is replaced by $\ell[v]$ (which may be $\star$ also).

Theorem 1. *Let $R \stackrel{\text{df}}{=} (\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be an RR system. Then, R and $\mathsf{norm}(R)$ generate equivalent state graphs.*

Proof. We first note that systems R and $R' \stackrel{\text{df}}{=} \mathsf{norm}(R)$ are defined on the same variables so that every state of R is also a valid state of R' and vice-versa. Moreover, they have the same initial states by definition. Let s and s' be two states of R or R', and let $a \stackrel{\text{df}}{=} (\ell, r)$ be an action of R and $a' \stackrel{\text{df}}{=} \mathsf{norm}(\ell, r) \stackrel{\text{df}}{=} (\ell, r')$.

We will prove that $s \xrightarrow{a} s'$ in R iff $s \xrightarrow{a'} s'$ in R'. As a consequence, starting from the same initial states the LTSs of R and R' are equivalent through (id, h) with $h \stackrel{\text{df}}{=} \{\mathsf{norm}(a) \mapsto a \mid a \in \mathcal{A}\}$.

($\Rightarrow$) assume $s \xrightarrow{a} s'$. By Definition 2 we have $s \simeq \ell$ and $s' = s \triangleleft r$. Since the left-hand side of a' is ℓ, we also have a' enabled by s. So we need to prove that $s' = s \triangleleft r = s \triangleleft r'$, or equivalently, that for all variable v we have $s[v] \triangleleft r[v] = s[v] \triangleleft r'[v]$. From Definition 4, there are three cases:

- if $\ell[v] = \star = r[v]$, we also have $r'[v] = \ell[v]$ and thus $r'[v] = \star = r[v]$, hence the result;
- if $r[v] \neq \star$ then $r'[v] = r[v]$ hence the result;
- if $\ell[v] \neq \star = r[v]$, then $r'[v] = \ell[v]$, and since a is enabled we have $\ell[v] = s[v]$, hence the result.

($\Leftarrow$) assume $s \xrightarrow{a'} s'$. The proof is essentially the same, by exchanging a (resp. r) with a' (resp. r'). □

Definition 5 (elementarisation). *Let $R \stackrel{\text{df}}{=} (\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be an RR system, with $\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ its set of actions. R is called elementary iff for all $a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A}$ and for all $v \in \mathcal{V}$ we have: $\ell[v] = \star \iff r[v] = \star$. Consequently, an elementary RR system is also normal.*

Assuming that R is normal, its elementarisation, noted by $\mathsf{elem}(R)$ is the RR system $(\mathcal{V}, \mathcal{I}, \mathcal{C}', \mathcal{R}')$ where

$$\mathcal{C}' \stackrel{\text{df}}{=} \bigcup_{a \in \mathcal{C}} \mathsf{elem}(a) \quad \textit{and} \quad \mathcal{R}' \stackrel{\text{df}}{=} \bigcup_{a \in \mathcal{R}} \mathsf{elem}(a) \ ,$$

with $a \stackrel{\text{df}}{=} (\ell, r)$, $\mathsf{elem}(a) \stackrel{\text{df}}{=} \mathsf{elem}(\ell, r, \min(\mathcal{V}))$ and $\mathsf{elem}(\ell, r, v)$ defined as:

- $\{(\ell \triangleleft \bot^{[v]}, r), (\ell \triangleleft \top^{[v]}, r)\}$ *if $v = \max(\mathcal{V})$ and $\ell[v] = \star \neq r[v]$;*
- $\{(\ell, r)\}$ *if $v = \max(\mathcal{V})$ and $\neg(\ell[v] = \star \neq r[v])$;*
- $\mathsf{elem}(\ell \triangleleft \bot^{[v]}, r, \mathrm{succ}(v)) \cup \mathsf{elem}(\ell \triangleleft \top^{[v]}, r, \mathrm{succ}(v))\}$ *if $v < \max(\mathcal{V})$ and $\ell[v] = \star \neq r[v]$;*
- $\mathsf{elem}(\ell, r, \mathrm{succ}(v))$ *if $v < \max(\mathcal{V})$ and $\neg(\ell[v] = \star \neq r[v])$.*

How $\mathsf{elem}(\ell, r)$ works is illustrated on the right of Figure 4: for each $v \in \mathcal{V}$ in turn, every $\ell[v] = \star$ such that $r[v] \neq \star$ is replaced by either $\bot$ or $\top$, yielding two new actions.

Theorem 2. *Let $R \stackrel{\text{df}}{=} (\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be a normal RR system. Then, R and $\mathsf{elem}(R)$ generate equivalent state graphs.*

Proof. As with normalisation, R and R' may have the same states and they do have the same initial states. Let s and s' be two states of R or R'. Let $a \stackrel{\text{df}}{=} (\ell, r)$ be an action of R. We will prove that $s \xrightarrow{a} s'$ in R iff $s \xrightarrow{a'} s'$ in R' for some $a' \in \mathsf{elem}(\ell, r)$. As a consequence, starting from the same initial states the LTSs of R and R' are equivalent through (id, h) with $h \stackrel{\text{df}}{=} \{a' \mapsto a \mid a \in \mathcal{A} \wedge a' \in \mathsf{elem}(a)\}$. The right-hand sides of a and all $a' \in \mathsf{elem}(a)$ are the same, so we just need to prove that a is enabled at s iff some $a' \in \mathsf{elem}(a)$ is enabled at s.

($\Rightarrow$) assume $s \xrightarrow{a} s'$. Take $\ell' \stackrel{\text{df}}{=} [\langle \ell[v] = \star \neq r[v] \,?\, s[v] : \ell[v] \rangle \mid v \in \mathcal{V}]$, we have $(\ell', r) \in \mathsf{elem}(a)$ and it is enabled at s because ℓ and ℓ' only differ on positions v where $\ell[v] = \star$ and at these positions we have $\ell'[v] = s[v]$.

($\Leftarrow$) take $a' \stackrel{\text{df}}{=} (\ell', r) \in \mathsf{elem}(a)$ and assume $s \xrightarrow{a'} s'$. As previously, ℓ and ℓ' only differ on positions v where $\ell[v] = \star$, which does not restrict enabling. □

4 Priority Petri nets

Regular Petri nets may be extended with transitions priorities. In our setting, we just need two levels of priorities, so we distinguish a set of urgent transitions whose firing is always preferred above that of non-urgent transitions (hence the used of letter U below). The former will be used to implement constraints while the latter will be used to implement rules.

Definition 6 (PPN). *A priority Petri net (PPN) is a tuple (P, T, W, U) where (P, T, W) is a RPN, called the underlying RPN, and $U \subseteq T$ is the set of urgent transitions.*

Definition 7 (PPN firing rule). *Let (P, T, W, U) be a PPN and m a marking of it. A transition $t \in U$ is enabled at m iff it is enabled at m in the underlying RPN. A transition $t \in T \setminus U$ is enabled at m iff it is enabled at m in the underlying RPN and no $u \in U$ is enabled at m. If t is enabled at m then we may have $m \xrightarrow{t} m - {}^\bullet t + t^\bullet$ just like in the underlying RPN.*

Definition 8 (PPN state graph). *Let (P, T, W, U) be a PPN and M the set of its initial markings. Its state graph is the smallest LTS $(S, M, T, \rightarrow)$ such that $M \subseteq S$ and if $m \xrightarrow{t} m'$ in the PPN with $m \neq m'$, then $m' \in S$ and $(m, t, m') \in \rightarrow$ in the LTS as well. The PPN is safe with respect to M iff for all $m \in S$ we have $m \leq \mathbb{1}$.*

Note that, as for RR systems, we restrict the LTS semantics of PPN to avoid self-loops (hence the condition $m \neq m'$).

Translation from an elementary RR system to a PPN is made by creating a pair of complementary places $p_v^\top$ and $p_v^\bot$ for each variable v and then, each action a gives rise to a transition t_a linked to these places as depicted in Figure 5. The figure also depicts the translation to PPN of four elementary rules, each being depicted separately from the others for the sake of readability.

Another way to avoid self-loops would be to remove from this translation all transitions that do not change the marking (i.e., transitions t such that ${}^\bullet t = t^\bullet$). This is probably the most practical solution but, as discusses later, it will not work for EPN so we prefer to use the same approach for both classes of Petri nets. Moreover, doing so, we guarantee that every action in an elementary RR system has a corresponding transition in the PPN translation.

Definition 9 (elementary RR systems to PPN). *Let $R \stackrel{\text{df}}{=} (\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be an elementary RR system, with $\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ its set of actions. R can be translated to a PPN $\mathsf{ppn}(R) \stackrel{\text{df}}{=} (P, T, W, U)$ and a set of initial markings M as follows:*

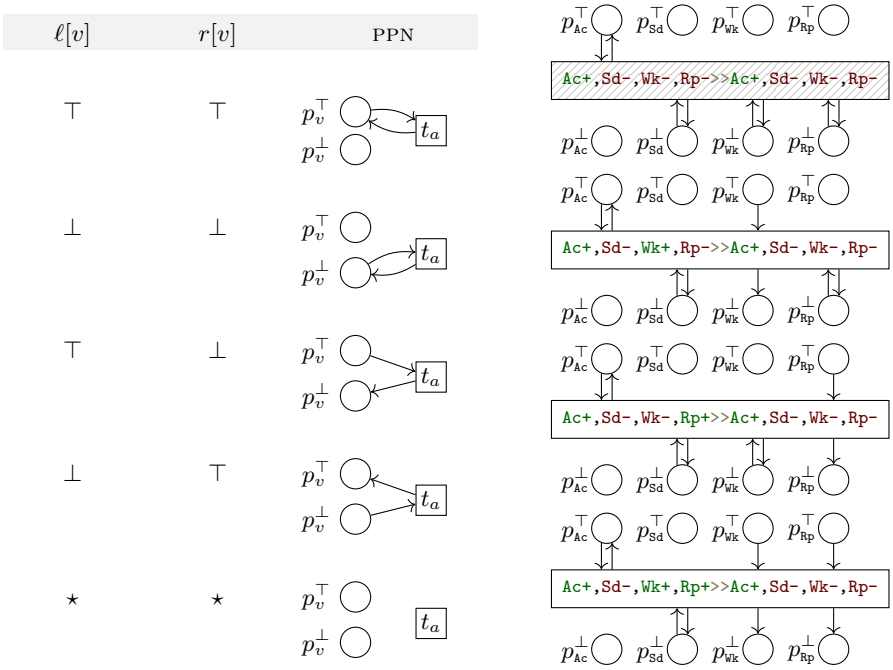

Fig. 5. (Left) Depiction of all the possible relations between a variable v and an action $a \stackrel{\text{df}}{=} (\ell, r)$ of an elementary RR system, and how this is translated to PPN. **(Right)** The four elementary rules resulting from rule `R9` (`Ac+, Sd- >> Wk-, Rp-`) in the termite model of Fig. 1 translated to PPN (presented separately to improve readability). The top-most transition is never fired because it does not change the marking.

- $P \stackrel{\text{df}}{=} \{p_v^\top, p_v^\bot \mid v \in \mathcal{V}\}$;
- $T \stackrel{\text{df}}{=} \{t_a \mid a \in \mathcal{A}\}$;
- $W \stackrel{\text{df}}{=} \{(p_v^{\ell[v]}, t_a), (t_a, p_v^{r[v]}) \mid a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A} \wedge v \in V \wedge \neg(\ell[v] = \star = r[v])\}$;
- $U \stackrel{\text{df}}{=} \{t_a \mid a \in \mathcal{C}\}$;
- $M \stackrel{\text{df}}{=} \{\{p_v^{s[v]} \mid v \in \mathcal{V}\} \mid s \in \mathcal{I}\}$.

Proposition 1. *With the notations from Definition 9, we have that* $\mathsf{ppn}(R)$ *is a safe* PPN *with respect to* M.

Proof. We prove by induction that all the reachable markings are safe and that the pairs of places $p_v^\top$ and $p_v^\bot$ are complementary places (i.e., they together hold exactly one token). For brevity below, we call v-safe such a marking.

(Basis.) Every marking in M is v-safe because for all $s \in \mathcal{I}$ and all $v \in \mathcal{V}$, depending on the value of $s[v]$, we put exactly one token in either $p_v^\top$ or $p_v^\bot$.

(Induction.) Firing any transition from a v-safe marking yields a v-safe marking. Indeed, for all $v \in \mathcal{V}$ and all $a \in \mathcal{A}$, we have the following cases, corresponding to the rows of Figure 5:

– if $\ell[v] = r[v] = \top$ (row 1) then transition t_a has a side loop on $p_v^\top$ which does not change the marking;
– case $\ell[v] = r[v] = \bot$ (row 2) is similar;
– $\ell[v] = \top$ and $r[v] = \bot$ (row 3) then transition t_a removes one token from $p_v^\top$ and puts one in $p_v^\bot$. Since these places are complementary, $p_v^\bot$ is empty before the firing and will hold one token after while $p_v^\top$ holds one token before the firing and will be empty after;
– case $\ell[v] = \bot$ and $r[v] = \top$ (row 4) is symmetric;
– if $\ell[v] = r[v] = \star$ (row 5) then transition t_a is not connected to $p_v^\top$ nor $p_v^\bot$ and thus does not change their markings.

No other arcs exist between t_a and $p_v^\top$ or $p_v^\bot$. □

Theorem 3. *With the notations from Definition 9, we have that R and $\mathsf{ppn}(R)$ generate equivalent state graphs.*

Proof. We prove that the LTS of R and that of $\mathsf{ppn}(R)$ are equivalent through (g, h) with $g \stackrel{\text{df}}{=} \{s \mapsto \{p_v^{s[v]} \mid v \in \mathcal{V}\} \mid s \in S\}$ where S is the set of states of the LTS of R, and $h \stackrel{\text{df}}{=} \{t_a \mapsto a \mid a \in \mathcal{A}\}$. To do so, we prove that $s \xrightarrow{a} s'$ in R iff $g(s) \xrightarrow{t_a} g(s')$ in $\mathsf{ppn}(R)$ with $s \neq s'$ two states of R.

($\Rightarrow$) assume $s \xrightarrow{a} s'$, with $a \stackrel{\text{df}}{=} (\ell, r)$. Since a is enabled, we have $\ell[v] = s[v]$ or $\ell[v] = \star = r[v]$ for all $v \in \mathcal{V}$, and thus ${}^\bullet t_a = \{p_v^{s[v]} \mid v \in \mathcal{V} \wedge \neg(\ell[v] = \star = r[v])\}$ by definition of W. Moreover, we have $g(s) = \{p_v^{s[v]} \mid v \in \mathcal{V}\}$ by definition. Thus ${}^\bullet t_a \leq g(s)$ and t_a is enabled. Take m' the marking such that $g(s) \xrightarrow{t_a} m' \stackrel{\text{df}}{=} g(s) - {}^\bullet t + t^\bullet$, it remains to prove that $m' = g(s')$. From the definitions we have

$$m' = \{p_v^{s[v]} \mid v \in \mathcal{V}\} \tag{1}$$
$$- \{p_v^{\ell[v]} \mid v \in \mathcal{V} \wedge \neg(\ell[v] = \star = r[v])\} \tag{2}$$
$$+ \{p_v^{r[v]} \mid v \in \mathcal{V} \wedge \neg(\ell[v] = \star = r[v])\} \tag{3}$$

and we can consider each $v \in \mathcal{V}$ separately. There are three cases, corresponding to the rows in Figure 5:

– if $\ell[v] = r[v] \neq \star$ (rows 1-2) we also have $s[v] = \ell[v]$ because s enables a, thus one token exists in $p_v^{s[v]}$ at (1), it is removed at (2) and another is added at (3) in $p_v^{\ell[v]}$ so it is marked in m'. Moreover we have $s'[v] = s[v]$ by definition of RR firing, so $g(s')$ has one token in $p_v^{s[v]}$ and none in its complementary place hence the result;
– if $\ell[v] \neq r[v]$ (rows 3-4) then because R is elementary none of them is $\star$. Assume $\ell[v] = \top$ and $r[v] = \bot$ (the other case is symmetric). The token in $p_v^\top$ from (1) is removed at (2), and one token is added to $p_v^\bot$ at (3). Moreover, by definition of RR firing, we have $s'[v] = r[v] = \bot$ so that $g(s')$ has one token in $p_v^\bot$ and none in $p_v^\top$ hence the result;
– if $\ell[v] = \star = r[v]$ (row 5) then place $p_v^{s[v]}$ is left untouched with one token inside because it is not connected to t_a. Moreover we have $s'[v] = s[v]$ by definition of RR firing, so $g(s')$ has one token in $p_v^{s[v]}$ and none in its complementary place hence the result.

($\Leftarrow$) assume $g(s) \xrightarrow{t_a} g(s')$. Since t_a is enabled, we have ${}^\bullet t_a = \{p_v^{\ell[v]} \mid v \in \mathcal{V} \wedge \neg(\ell[v] = \star = r[v])\} \leq g(s) = \{p_v^{s[v]} \mid v \in \mathcal{V}\}$. So, for each v such that $\neg(\ell[v] = \star = r[v])$ we have $\mathbb{0} < \{p_v^{\ell[v]}\} \leq \{p_v^{s[v]}\} \leq \mathbb{1}$ and thus $\ell[v] = s[v]$. Moreover, each v such that $\ell[v] = \star = r[v]$ has no influence on the enabling of a. So a is enabled at s. It remains to show that $s \xrightarrow{a} s'$. Taking $m' = g(s')$, equation (1-3) above still holds and we consider each v separately. There are three cases, corresponding to the rows in Figure 5:

- if $\ell[v] = s[v] = r[v]$ (rows 1-2), one token is removed and another is added from $p_v^{s[v]}$ while $p_v^{\neg s[v]}$ remains empty, thus $s[v] = s'[v]$ which is what firing a with $\ell[v] = s[v] = r[v]$ yields;
- if $\ell[v] = s[v] \neq r[v]$ (rows 3-4), the token in $p_v^{s[v]}$ from (1) is removed at (2) and one is added to $p_v^{\neg s[v]}$ at (3). Because the net is safe, $p_v^{\neg s[v]}$ is empty in $g(s)$ and holds exactly one token in $g(s')$. Firing a from s with $\ell[v] \neq r[v]$ changes the value of v thus $s'[v] = \neg s[v]$, which corresponds to the marking;
- if $\ell[v] = r[v] = \star$ (row 5), token in $p_v^{s[v]}$ from (1) is not removed at (2) and no token is added in $p_v^{s[v]}$ nor $p_v^{\neg s[v]}$ at (3). $g(s')$ has one token in $p_v^{s[v]}$ and $s[v] = s'[v]$, which is what firing a with $\ell[v] = r[v] = \star$ yields. □

5 Extended Petri nets

Priority Petri nets may be further extended with:

- read arcs (depicted as bare edges) that allow to test for the presence of tokens without consuming them (letter Z below is for "Zero tokens consumed");
- inhibitor arcs (depicted with a white dot at the transition side) that allow to test for the absence of tokens in a place (letter H below is for "inHibitor");
- reset arcs (depicted with a black diamond at the transition side) that allow to consume all the tokens from a place, if any (letter F below is for "Flush").

Considering this class rather that PPN allows a translation from RR systems without resorting to elementarisation, only normalisation is needed. On the good side, one action written by the modeller is being translated to one EPN transition and normalisation can be kept invisible. On the bad side, we need to cope with a more complex class of Petri nets for which fewer tools may be available.

Definition 10 (EPN). *An extended Petri net (EPN) is a tuple (S, T, W, U, Z, H, F) where (S, T, W, U) is a PPN, called the underlying PPN, and:*

- *$Z \in (P \times T)^*$ defines the (weighted) read arcs and for $t \in T$ we define ${}^*t \stackrel{\text{df}}{=} \{p \mapsto Z(p,t) \mid (p,t) \in Z\}$ the places from which t reads tokens;*
- *$H \in (P \times T)^*$ defines the (weighted) inhibitor arcs and for $t \in T$ we define ${}^\circ t \stackrel{\text{df}}{=} \{p \mapsto H(p,t) \mid (p,t) \in H\}$ the multiset of places from which t checks the absence of too much tokens;*
- *$F \subseteq (P \times T)$ defines the reset arcs and for $t \in T$ we define ${}^\blacklozenge t \stackrel{\text{df}}{=} \{p \in P \mid (p,t) \in F\}$ the set of places whose marking is reset by t.*

Definition 11 (EPN firing rule). *Let (S, T, W, U, Z, H, F) be an* EPN *and m a marking of it. A transition $t \in T$ is enabled at m iff it is also enabled in the underlying* PPN*, and we have ${}^*t \leq m$ and $m/{}^\circ t < {}^\circ t$. If t is enabled at m then we may have $m \xrightarrow{t} m'$ with $m' \stackrel{\text{df}}{=} m/(P \setminus {}^\blacklozenge t) - {}^\bullet t + t^\bullet$.*

The intuition behind this firing rule is as follows:

- t must be enabled in the underlying PPN, that is: there are enough tokens to be consumed by the regular arcs, and priorities are respected;
- ${}^*t \leq m$ checks that there are enough tokens to be tested by the read arcs. Read arcs are weighted so for instance ${}^*t(p) = 2$ means that two tokens will be tested in p, thus the inequality;
- $m/{}^\circ t < {}^\circ t$ checks that there are not too much tokens with respect to the inhibitor arcs. We consider $m/{}^\circ t$ instead of m because a weight zero on a inhibitor arc corresponds to the absence of such an arc. So, this condition can be read as "for every place p such that there is an inhibitor arc between t and p with weight $w > 0$, p must be marked by less than w tokens";
- $m' \stackrel{\text{df}}{=} m/(P \setminus {}^\blacklozenge t) - {}^\bullet t + t^\bullet$ is similar to $m' \stackrel{\text{df}}{=} m - {}^\bullet t + t^\bullet$ in PPN but instead of computing m' from m, we compute it from m restricted to the places that are not connected to t through a reset arc. In other words, $m/(P \setminus {}^\blacklozenge t)$ is m in which we emptied all the places connected to t through a reset arc.

Definition 12 (EPN state graph). *Let (S, T, W, U, Z, H, F) be an* EPN *and M the set of its initial markings. Its state graph is the smallest* LTS *$(S, M, T, \to)$ such that $M \subseteq S$ and if $m \xrightarrow{t} m'$ in the* EPN *with $m \neq m'$, then $m \in S$ and $(m, t, m') \in \to$ in the* LTS *as well. The* EPN *is safe with respect to M iff for all $m \in S$ we have $m \leq \mathbb{1}$.*

As with PPN, we have restricted the semantics to avoid self-loops in an EPN state graph.

Translation from a normal RR system to an EPN is made by creating one place p_v for each variable v and one transitions t_a for each action a that is connected to each p_v as depicted in Figure 6. An example of such a translation is depicted on the right of the figure.

Definition 13 (normal RR systems to EPN). *Let $R \stackrel{\text{df}}{=} (\mathcal{V}, \mathcal{I}, \mathcal{C}, \mathcal{R})$ be a normal* RR *system, with $\mathcal{A} \stackrel{\text{df}}{=} \mathcal{C} \uplus \mathcal{R}$ its set of actions. R can be translated to a* EPN $\mathsf{epn}(R) \stackrel{\text{df}}{=} (P, T, W, U, Z, H, F)$ *and a set of initial markings M as follows, with reference to Figure 6 displayed at the end of lines:*

- $P \stackrel{\text{df}}{=} \{p_v \mid v \in \mathcal{V}\}$;
- $T \stackrel{\text{df}}{=} \{t_a \mid a \in \mathcal{A}\}$;
- $W \stackrel{\text{df}}{=} \{(p_v, t_a) \mid v \in \mathcal{V} \wedge a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A} \wedge \ell[v] = \top \wedge r[v] = \bot\}$ *(row 2)*
  $+ \{(t_a, p_v) \mid v \in \mathcal{V} \wedge a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A} \wedge \ell[v] \neq \top \wedge r[v] = \top\}$; *(rows 4–5)*
- $U \stackrel{\text{df}}{=} \{t_a \mid a \in \mathcal{C}\}$;
- $Z \stackrel{\text{df}}{=} \{(p_v, t_a) \mid v \in \mathcal{V} \wedge a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A} \wedge \ell[v] = r[v] = \top\}$; *(row 1)*
- $H \stackrel{\text{df}}{=} \{(p_v, t_a) \mid v \in \mathcal{V} \wedge a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A} \wedge \ell[v] = \bot\}$; *(rows 3–4)*
- $F \stackrel{\text{df}}{=} \{(p_v, t_a) \mid v \in \mathcal{V} \wedge a \stackrel{\text{df}}{=} (\ell, r) \in \mathcal{A} \wedge \ell[v] = \star \wedge r[v] \neq \star\}$; *(rows 5–6)*

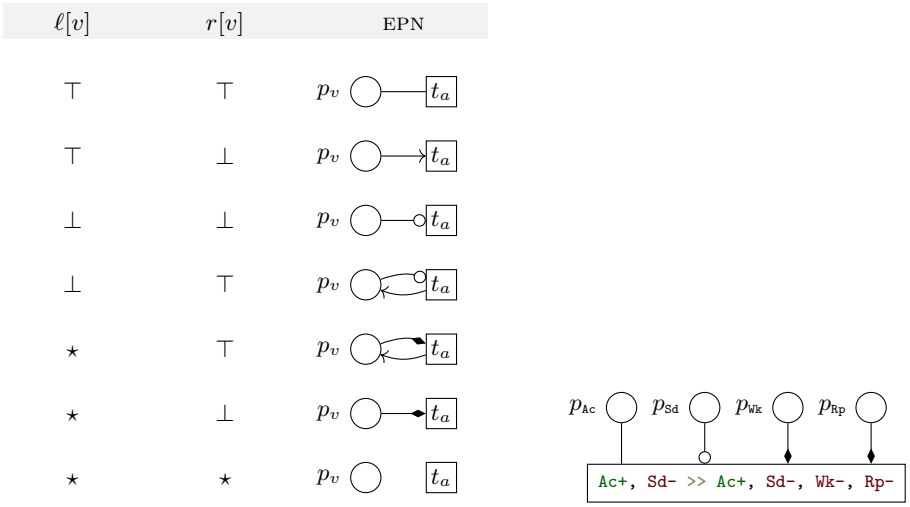

Fig. 6. (Left) Depiction of all the possible relations between a variable v and an action $a \stackrel{\text{df}}{=} (\ell, r)$ of a normal RR system, and how this is translated to EPN. **(Right)** The normal rule resulting from rule `R9` in the termite model of Fig. 1 translated to EPN.

– $M \stackrel{\text{df}}{=} \{\{p_v \mid v \in \mathcal{V} \wedge s[v] = \top\} \mid s \in \mathcal{I}\}$.

Proposition 2. *With the notations from Definition 13, we have that* $\mathsf{epn}(R)$ *is a safe* EPN *with respect to* M.

Proof. We prove by induction that all reachable markings are safe.

(Basis.) Every marking in M is safe because for all $s \in \mathcal{I}$ and all $v \in \mathcal{V}$, depending on the value of $s[v]$, we put at most one token in p_v.

(Induction.) Firing any transition from a safe marking yields a safe marking. Indeed, for all $v \in \mathcal{V}$ and all $a \in \mathcal{A}$, we have the following cases, corresponding to the rows of Figure 6:

– if $\ell[v] = r[v] = \top$ (row 1), only a read arc exists between t_a and p_v, which does not change its marking;
– if $\ell[v] = \top$ and $r[v] = \bot$ (row 2), a token is consumed by t_a from p_v which keeps it safe;
– if $\ell[v] = r[v] = \bot$ (row 3), only an inhibitor arc exists between t_a and p_v, which does not change its marking;
– if $\ell[v] = \bot$ and $r[v] = \top$ (row 4), a token is produced by t_a into p_v but only if it is empty thanks to the inhibitor arc between t_a and p_v;
– if $\ell[v] = \star$ and $r[v] = \top$ (row 5), p_v is emptied by t_a thanks to the reset arc and then one token is produced in p_v;
– if $\ell[v] = \star$ and $r[v] = \bot$ (row 6), p_v is emptied by t_a;
– if $\ell[v] = r[v] = \star$ (row 7), there is no arc between t_a and p_v so its marking is untouched.

No other arcs exist between t_a and p_v. $\square$

Theorem 4. *With the notations from Definition 13, we have that R and $\mathsf{epn}(R)$ generate equivalent state graphs.*

Proof. We prove that the LTS of R and that of $\mathsf{epn}(R)$ are equivalent through (g, h) with $g \stackrel{\text{df}}{=} \{s \mapsto \{p_v \mid v \in \mathcal{V} \wedge s[v] = \top\} \mid s \in S\}$ where S is the set of states of the LTS of R, and $h \stackrel{\text{df}}{=} \{t_a \mapsto a \mid a \in \mathcal{A}\}$. To do so, we prove that $s \xrightarrow{a} s'$ in R iff $g(s) \xrightarrow{t_a} g(s')$ in $\mathsf{epn}(R)$ with $s \neq s'$ two states of R.

($\Rightarrow$) assume $s \xrightarrow{a} s'$, with $a \stackrel{\text{df}}{=} (\ell, r)$. First we prove that t_a is enabled. From Definitions 11 and 13, we must have:

$$
\begin{aligned}
&{}^\bullet t \stackrel{\text{df}}{=} \{p_v \mid v \in \mathcal{V} \wedge \ell[v] = \top \wedge r[v] = \bot\} \leq g(s) \stackrel{\text{df}}{=} \{p_v \mid v \in \mathcal{V} \wedge s[v] = \top\} && (4)\\
&{}^* t \stackrel{\text{df}}{=} \{p_v \in \mathcal{V} \mid \ell[v] = r[v] = \top\} \leq g(s) && (5)\\
&{}^\circ t \stackrel{\text{df}}{=} \{p_v \in \mathcal{V} \mid \ell[v] = \bot\} > g(s)/{}^\circ t && (6)
\end{aligned}
$$

(4) and (5) hold because when a is enabled we must have $s[v] = \top$ for every v such that $\ell[v] = \top$. (6) holds because when a is enabled we must have $s[v] = \bot$ and thus $g(s)(p_v) = 0$ for every v such that $\ell[v] = \bot$. Take m' the marking such that $g(s) \xrightarrow{t_a} m' \stackrel{\text{df}}{=} g(s)/(P \setminus {}^\blacklozenge t) - {}^\bullet t + t^\bullet$. It remains to prove that $m' = g(s')$. From the definitions we have:

$$
\begin{aligned}
m' = \{p_v \mid v \in \mathcal{V} \wedge s[v] = \top \wedge (\ell[v] \neq \star \vee r[v] = \star)\} && (7)\\
- \{p_v \mid v \in \mathcal{V} \wedge \ell[v] = \top \wedge r[v] = \bot\} && (8)\\
+ \{p_v \mid v \in \mathcal{V} \wedge \ell[v] \neq \top \wedge r[v] = \top\} && (9)
\end{aligned}
$$

where $\ell[v] \neq \star \vee r[v] = \star$ corresponds to $p_v \notin {}^\blacklozenge t$ by definition of F in the translation. Then we can consider each $v \in \mathcal{V}$ separately and there are seven cases, corresponding to the rows in Figure 6:

- if $\ell[v] = r[v] = \top$ (row 1), then we have $s[v] = \top$ because a is enabled and $s'[v] = \top$ by definition of RR firing. On the Petri net side, the token in p_v from m is kept at (7) and not removed at (8), and no token is added at (9), so the marking is not changed, hence the result;
- if $\ell[v] = \top$ and $r[v] = \bot$ (row 2), then we have $s[v] = \top$ and $s'[v] = \bot$. Moreover, the token in p_v from m is kept at (7), it is removed at (8), and no token is added at (9), hence the result;
- if $\ell[v] = r[v] = \bot$ (row 3), then we have $s[v] = s'[v] = \bot$. Moreover, there is no token in p_v from m thus none can be kept at (7), and none is added at (9), hence the result;
- if $\ell[v] = \bot$ and $r[v] = \top$ (row 4), then we have $s[v] = \bot$ and $s'[v] = \top$. Moreover, there is no token in p_v from m thus none is kept at (7), and one is added at (9), hence the result;
- if $\ell[v] = \star$ and $r[v] = \top$ (row 5), then we do not know what $s[v]$ is but we have $s'[v] = \top$. Moreover, any token in p_v from m is not kept at (7), and one is added at (9), hence the result;

– if $\ell[v] = \star$ and $r[v] = \bot$ (row 6), then we do not know what $s[v]$ is but we have $s'[v] = \bot$. Moreover, any token in p_v from m is not kept at (7), and none is added at (9), hence the result;
– if $\ell[v] = r[v] = \star$ (row 7), then $s[v] = s'[v]$. Moreover, any token in p_v from m is kept at (7), not removed at (8), and no other token is added at (9), hence the result.

($\Leftarrow$) assume $g(s) \xrightarrow{t_a} g(s')$, and thus relations (4-6). Consider each $v \in \mathcal{V}$ separately, we have four cases to prove that a is enabled at s:

– if $\ell[v] = r[v] = \top$ then from (5) we have $0 < {}^*t(p_v) \leq g(s)(p_v)$ thus $s[v] = \top$;
– if $\ell[v] = \top$ and $r[v] = \bot$ then from (4) we have $0 < {}^\bullet t(p_v) \leq g(s)(p_v)$ and thus $s[v] = \top$;
– if $\ell[v] = \bot$ then from (6) we have $1 = {}^\circ t(v) > g(s)(p_v)$ thus $s[v] = \bot$;
– if $\ell[v] = \star$ then v has no influence on the enabling of a.

It remains to show that $s \xrightarrow{a} s'$, taking $m' = g(s')$, equation (7-9) still holds and we consider each v separately. There are five cases corresponding to the rows in Figure 6:

– if $\ell[v] = r[v]$ (rows 1, 3, and 7), then a possible token in p_v is kept at (7) and not removed at (8) while none is added at (9), and from the definition of RR firing we have $s'[v] = s[v]$, hence the result;
– if $\ell[v] = \top$ and $r[v] = \bot$ (row 2), then one token is consumed from p_v at (8) and none is added at (9) so that $m'(p_v) = 0$. From the definition of RR firing we have $s'[v] = \bot$, hence the result;
– if $\ell[v] = \bot$ and $r[v] = \top$ (row 4), then $m(p_v) = 0$ and one token is added at (9), so that $m'(p_v) = 1$. Moreover, we have $s'[v] = \top$, hence the result;
– if $\ell[v] = \star$ and $r[v] = \top$ (row 5), then no token is copied from m into p_v at (7) and one is added at (9), so that $m'(p_v) = 1$. Moreover, we have $s'[v] = \top$, hence the result;
– if $\ell[v] = \star$ and $r[v] = \bot$ (row 6), then any token in p_v is skipped at (7) and none is added at (9) so that $m'(p_v) = 0$. Moreover, we have $s'[v] = \bot$, hence the result. □

6 Conclusion

We have presented a modelling language for ecosystems called *reaction rules* (RR) that has been developed and used for more than five years. This is a simple rule-based language in which ecological entities are modelled as Boolean variables, and the potential events in the ecosystem are modelled as if-then rules. This language is equipped with an operational semantics expressed in terms of *labelled transitions systems* (LTS). Then, we have proposed two alternative denotational semantics: (1) a translation to Petri nets extended with transitions priorities (PPN) that is obtained through an *elementarisation* of the translated system's rules, and (2) a translation to PPN further extended with read, inhibitor, and reset arcs (EPN) that is obtained trough a *normalisation* of the translated system's rules.

The main result of this paper is to prove that all these semantics are strongly equivalent, which is expressed in terms of the isomorphism of the corresponding LTS, with states and labels matching. We have defined in proofs constructive mappings that can be used in practice to translate one kind of LTS into another. The overall contribution is summarised in Figure 2 page 3.

The main interest of having several consistent semantics is the ability to chose one or another depending on the situation. For example, the operational semantics can be presented in intuitive terms directly on the RR concrete syntax, and thus it is suitable to be explained to ecologists. However, no implementation exists for it so it cannot be used to compute state-spaces. On the other hand, the Petri net semantics allows to use one of the numerous tools readily available for Petri nets. For instance, in [5–8], we have used TINA [1] to compute explicit state-spaces from the PPN semantics of RR since TINA supports transitions priorities. We also have used the EPN semantics in [4, 12] through a translation of extended Petri nets into GAL systems [19, Sec. 5] in order to compute symbolic state-spaces using libDDD and ITS-tools [18, 19]. Another use of the EPN semantics could be through the SNAKES [15] library for interactive simulation.

6.1 Related works

The design of the RR modelling language has been made by computer scientists working together with ecologists, with the goal to provide a language that is both as simple as possible and also sufficiently descriptive for actual use by ecologists. Actually, modelling ecosystems by discrete systems with *if-then* rules was proposed in the early 90's in [16, 17], but using multi-valued variables. These works have then evolved towards cellular automata and, to the best of our knowledge, remained focused on simulation-based analysis.

RR being based on Boolean variables, it may appear similar to Boolean networks that are widely used in systems biology [14, 21]. However, both languages have several important differences. First of all, they greatly differ in the modelling philosophy: Boolean networks are centred onto how each variable is influenced by the others and thus present the system as an interaction network; RR is centred onto the potential events in the system and thus presents the system as a rule-based model. Then, RR allows to express non-determinism at the level of rules while, in Boolean networks, it arises only in the semantics from the update mode of the variables [3]. This is a crucial feature to model ecosystems that often exhibit such non-deterministic behaviours where the same causes (as far as they can be observed) may lead to distinct consequences. Finally, from the LTS perspective, RR is strictly more expressive than Boolean networks. It has been proved that it can generate any LTS based on Boolean variables while Boolean networks cannot generate LTS in which a state has successors with incompatible updates of some variables [20], as for example that depicted in Figure 7. From this it appears that Boolean networks offer a trade-off between modelling complexity and generality, by allowing modellers to focus on the evolution of each variable. RR on the other hand, is focused on the events and may lead to more detailed, and thus more complex, models.

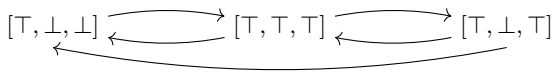

Fig. 7. A LTS on three Boolean variables that can be modelled by an RR system but not by a Boolean network. (This is left to the reader.) This LTS has been taken in the literature in ecology and corresponds to actual observations [9].

6.2 Future works

Several extensions of the RR language will be considered in the future. In particular, we have preliminary results based on an extension with explicit spatial information, which allows to model ecosystems taking into account their "geography". Another extension that is demanded by some users is the ability to have multi-valued variables, for example to represent ecosystems where some species play different roles depending on several thresholds of their population. Finally, we are working on a compact semantics that would remove from the state-space the constraints and the states from which they are executed. This is motivated by the fact that constraints are usually introduced to skip states that are only transient and should be discarded when studying the long term dynamics. While such semantics is quite easy to obtain in explicit state-spaces, it is more tricky for symbolic state-spaces. Moreover, the properties preserved or not by this transformation are still to be precisely characterised.

Another trend of research addresses more particularly the EPN semantics of RR systems. A PhD is in progress about obtaining unfoldings *à la* McMillan [13] for such Petri nets, with the aim to be able to apply in ecology the techniques developed in [2, 10, 11] for a Petri nets semantics of Boolean networks.

References

1. Berthomieu, B., Ribet, P.O., Vernadat, F.: The tool TINA–construction of abstract state spaces for Petri nets and time Petri nets. International journal of production research **42**(14) (2004)
2. Chatain, T., Haar, S., Kolčák, J., Paulevé, L., Thakkar, A.: Concurrency in Boolean networks. Natural Computing **19**(1) (2020)
3. Chatain, T., Haar, S., Paulevé, L.: Boolean networks: Beyond generalized asynchronicity. In: Cellular Automata and Discrete Complex Systems. Springer International Publishing (2018)
4. Cosme, M., Hély, C., Pommereau, F., Pasquariello, P., Tiberi, C., Treydte, A., Gaucherel, C.: Qualitative modeling for bridging expert-knowledge and social-ecological dynamics of an east African savanna. Land **11**(1) (2022). `https://doi.org/10.3390/land11010042`
5. Di Giusto, C., Gaucherel, C., Klaudel, H., Pommereau, F.: Analysis of discrete models for ecosystem ecology. In: Biomedical Engineering Systems and Technologies. Springer (2020)
6. Gaucherel, C., Carpentier, C., Geijzendorffer, I., Noûs, C., Pommereau, F.: Discrete-event models for conservation assessment of integrated ecosystems. Ecological Informatics **61** (2021). `https://doi.org/http://doi.org/10.1016/j.ecoinf.2020.101205`

7. Gaucherel, C., Pommereau, F., Hély, C.: Understanding ecosystem complexity via application of a process-based state space rather than a potential surface. Complexity **2020** (2020). https://doi.org/http://doi.org/10.1155/2020/7163920
8. Gaucherel, C., Pommereau, F.: Using discrete systems to exhaustively characterize the dynamics of an integrated ecosystem. Methods in Ecology and Evolution **10**(9) (2019)
9. Liao, C.: Complexity In The Open Grazing System: Rangeland Ecology, Pastoral Mobility And Ethnobotanical Knowledge In Borana, Ethiopia. PhD Thesis, Cornell University (2016)
10. Mandon, H., Su, C., Haar, S., Pang, J., Paulevé, L.: Sequential reprogramming of Boolean networks made practical. In: International Conference on Computational Methods in Systems Biology. Springer (2019)
11. Mandon, H., Su, C., Pang, J., Paul, S., Haar, S., Paulevé, L.: Algorithms for the sequential reprogramming of Boolean networks. IEEE/ACM transactions on computational biology and bioinformatics **16**(5) (2019)
12. Mao, Z., Centanni, J., Pommereau, F., Stokes, A., Gaucherel, C.: Maintaining biodiversity promotes the multifunctionality of social-ecological systems: holistic modelling of a mountain system. Ecosystem Services **47** (2021). https://doi.org/http://doi.org/10.1016/j.ecoser.2020.101220
13. McMillan, K.L., Probst, D.K.: A technique of state space search based on unfolding. Formal methods in system design **6**(1) (1995)
14. Naldi, A., Monteiro, P.T., Müssel, C., the Consortium for Logical Models, Tools, Kestler, H.A., Thieffry, D., Xenarios, I., Saez-Rodriguez, J., Helikar, T., Chaouiya, C.: Cooperative development of logical modelling standards and tools with CoLoMoTo. Bioinformatics **31**(7) (2015). https://doi.org/10.1093/bioinformatics/btv013
15. Pommereau, F.: SNAKES: a flexible high-level Petri nets library. In: Proceedings of PETRINETS 2015. LNCS, vol. 9115. Springer (2015). https://doi.org/https://dx.doi.org/10.1007/978-3-319-19488-2_13
16. Rykiel, E.J.: Artificial intelligence and expert systems in ecology and natural resource management. Ecological Modelling **46**(1) (1989). https://doi.org/10.1016/0304-3800(89)90066-5
17. Starfield, A.M.: Qualitative, Rule-Based Modeling. BioScience **40**(8) (1990). https://doi.org/10.2307/1311300
18. Thierry-Mieg, Y.: Homepage of ITS-tools. http://lip6.github.io/ITSTools-web
19. Thierry-Mieg, Y.: From Symbolic Verification to Domain Specific Languages. Habilitation thesis, UPMC (2016)
20. Thomas, C.: Model checking applied to discrete models of ecosystems. Master's thesis, ENS Paris-Saclay (2019)
21. Thomas, R.: Boolean formalization of genetic control circuits. Journal of Theoretical Biology **42**(3) (1973). https://doi.org/https://doi.org/10.1016/0022-5193(73)90247-6