Probabilistic Bigraphs

BLAIR ARCHIBALD, University of Glasgow
MUFFY CALDER, University of Glasgow
MICHELE SEVEGNANI, University of Glasgow

Bigraphs are a universal computational modelling formalism for the spatial and temporal evolution of a system in which entities can be added and removed. We extend bigraphs to probabilistic bigraphs, and then again to action bigraphs, which include non-determinism and rewards. The extensions are implemented in the BigraphER toolkit and illustrated through examples of virus spread in computer networks and data harvesting in wireless sensor systems. BigraphER also supports the existing *stochastic bigraphs* extension of Krivine *et al.*, and using BigraphER we give, for the first time, a direct implementation of the membrane budding model used to motivate stochastic bigraphs.

CCS Concepts: •**Theory of computation → Models of computation;** •**Mathematics of computing → Graph algorithms; Probabilistic representations;** •**Software and its engineering → System modeling languages;**

Additional Key Words and Phrases: Bigraphs, Systems Modelling, Probabilistic Modelling, Graph Rewriting

ACM Reference format:
Blair Archibald, Muffy Calder, and Michele Sevegnani. 2022. Probabilistic Bigraphs. 1, 1, Article 1 (January 2022), 28 pages.
DOI: 10.1145/3545180

1 INTRODUCTION

Bigraphical reactive systems (BRSs) [Mil09] are a universal computational modelling formalism for systems that evolve in time and space. They consist of bigraphs, a graph based formalism that models entity relationships, both spatially and through (global) links, and a rewriting framework that allows models to evolve over time via a set of reaction (rewrite) rules. Applying a reaction rule, $L \longrightarrow\!\!\!\!\blacktriangleright R$, replaces an occurrence of bigraph L (in a bigraph) with bigraph R. BRSs can represent a diverse range of phenomena including mixed-reality games [BCRS16], network management [CKSS14], wireless communication protocols [CS14], biological processes [KMT08], cyber-physical security [TPGN18], and indoor environments [WW12].

In practice, the systems we wish to model may be probabilistic, stochastic, or explicitly make non-deterministic choices. Standard BRSs have no notion of the first two concepts, and are implicitly non-deterministic in that if there is a match to L then any rule can be applied.

Previously, Krivine *et al.* [KMT08] extended bigraphs to *stochastic bigraphs*, by associating rates (rather than weights) with reaction rules. We build on that work, utilising similar ideas to create *probabilistic bigraphs* – a discrete variant. We then take the theory further to allow explicit non-determinism with *action bigraphs* that encode Markov decision processes [Bel57], by adding *actions* and *rewards*.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2022 Copyright held by the owner/author(s). XXXX-XX/2022/1-ART1 $
DOI: 10.1145/3545180

For each of the three types of system – probabilistic, stochastic, and action based – we provide an implementation of the theory in the BigraphER toolkit [SC16]. This allows, for the first time, an implementation and analysis of the Krivine et al stochastic bigraph example [KMT08] without requiring a separate PRISM [KNPV09] model.

We make the following contributions:

- we extend standard BRSs with probabilistic reaction rules, to create probabilistic BRSs,
- we extend probabilistic BRSs with non-deterministic actions and reward structures, to create action BRSs,
- we provide an implementation[1], in BigraphER [SC16] of probabilistic, stochastic, and action BRSs,
- we illustrate the new modelling capability through examples of virus spread through computer networks, the membrane budding example of [KMT08], and data harvesting in wireless sensor networks with mobile sinks. Full models are available [ASC].

Outline. The paper is structured as follows. Bigraphs and BRSs are introduced in Sections 2.1 and 2.2, with emphasis on the important notion of matching/occurrence; probabilistic systems are introduced in Section 2.3. In Section 3 we introduce probabilistic BRSs by adding *relative weights* to reaction rules. Section 4 extends probabilistic bigraphs further by adding explicit *actions* that represent non-deterministic choice. We evaluate the approaches through a set of further examples, implemented in an extended BigraphER, in Section 5. We conclude in Section 6 with a discussion of the limitations of the approaches, how they relate to other probabilistic modelling frameworks, and future work.

2 BACKGROUND

2.1 Bigraphs

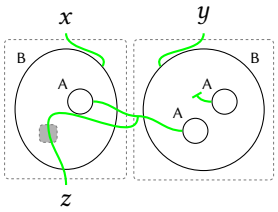
We introduce bigraphs by example, formal definitions can be found elsewhere [Mil09]. Although we restrict ourselves to Milner's original formulation of bigraphs (standard bigraphs), the probabilistic, stochastic, and non-deterministic variants are also applicable to, and implemented for, bigraphs with sharing [SC15] – an extension supporting overlapping spatial regions.

Bigraphs are a universal computational model for representing both the spatial configuration of entities, and their non-spatial interactions. A bigraph consists of two orthogonal structures: the *place graph*, that represents topological space in terms of containment, and the *link graph*, a hypergraph that expresses non-spatial relationships among entities. Each entity has a *type* that determines its (fixed) *arity*, *i.e.* number of links, and whether it is *atomic*, *i.e.* if it cannot contain other nodes.

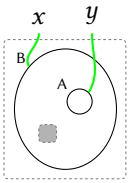
Bigraphs have an equivalent diagrammatic and algebraic notation. Throughout this paper we use the intuitive diagrammatic notation where possible. An example bigraph is shown in Figure 1a. Entities are drawn as (coloured) shapes, with the label, *e.g.* A, B, ... determining the type. Where it is clear from the context we will often omit the labels. Entities may be nested, *e.g.* A is inside B, and non-atomic entities can have any (finite) number of children. The green hyperlinks represent non-spatial links between entities, such as between the two A's in different B's. Entities have fixed arity, so the rightmost A in Figure 1a *must* have a single link, but in this case it is *closed*.

Bigraphs are compositional in nature, that is, we may combine smaller bigraphs to create larger models. To achieve this compositionality, alongside entities, bigraphs may contain *regions*, shown by clear dashed rectangles, which represent adjacent parts of a system; *sites*, shown by filled dashed rectangles, represent abstraction, *i.e.* an unspecified bigraph (possibly the empty bigraph) exists

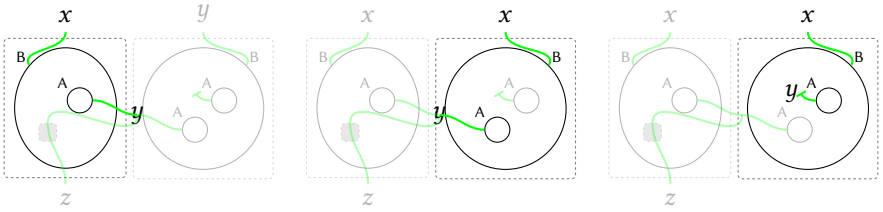
[1]Available online: https://uog-bigraph.bitbucket.io/



(a) $B : \langle 1, \{z\}\rangle \to \langle 2, \{x, y\}\rangle$



(b) $G : \langle 1, \emptyset\rangle \to \langle 1, \{x, y\}\rangle$



(c) Occurrences of G in B. Note that it is possible to rename links as required.

Fig. 1. Example bigraph B (a), pattern bigraph G (b), and occurrences (c).

there; a set of *inner names*, *e.g.* $\{z\}$, allows names to be connected from below; and a set of *outer names*, *e.g.* $\{x, y\}$, allows these links to connect with a wider context. Capabilities to interact with an external environment are recorded formally in the *interface* of a bigraph. For example, in Figure 1a we write $B : \langle 1, \{z\}\rangle \to \langle 2, \{x, y\}\rangle$ to indicate that B has one site and inner name z (written $\langle 1, \{z\}\rangle$) and, two regions and outer names x, y (written $\langle 2, \{x, y\}\rangle$). We use X, Y and K, I, J to denote sets of names and interfaces, respectively.

Composition of two bigraphs $F : K \to I$ and $G : I \to J$ is written

$$G \circ F : K \to J$$

and operates by placing the regions on F inside the sites of G and linking like outer names from F with inner names from G. When the name sets are disjoint, bigraphs may also be combined horizontally by placing regions, which may contain any other bigraph, side-by-side. This is denoted by

$$F_0 \otimes F_1 : \langle m_0 + m_1, X_0 \cup X_1\rangle \to \langle n_0 + n_1, Y_0 \cup Y_1\rangle$$

with $F_i : \langle m_i, X_i\rangle \to \langle n_i, Y_i\rangle$ and $i = 0, 1$.

We write id_X for the *identity* bigraph that maps like-names to like-names. We call bigraphs with no sites or inner names, *e.g.* those that cannot be composed with others, *ground*. In general, we define reactive systems over ground bigraphs, since these represent fully formed models. We use lowercase letters $f, g, g_0, \ldots$ to denote ground bigraphs and uppercase for arbitrary bigraphs (including those that may be ground).

When constructing bigraph models we use *abstract bigraphs*, where entities are identified using their types, *e.g.* an entity A. However, when rewriting models, to identity specific entities, we work instead with *concrete bigraphs*, $\widetilde{F}, \widetilde{G}, \ldots, \widetilde{g}$, where entities and closed links have distinct identifiers, *e.g.* $v, u, e, \ldots$. For a bigraph G, we assign an arbitrary *concretion* $\widetilde{G}$ by giving distinct labels to entities/closed links. We say two concrete bigraphs $\widetilde{F}$ and $\widetilde{G}$ are support equivalent, denoted $\widetilde{F} \simeq \widetilde{G}$, if they are equal under a renaming of entities and links. A bigraph is trivially support equivalent to

itself. An abstract bigraph B is a $\simeq$-equivalence class of concrete bigraphs[2], $B = [\widetilde{B}]_{\simeq}$ with $\widetilde{B}$ an arbitrary concretion of B. An example of support equivalence is in Section 3.

2.2 Bigraphical Reactive Systems

A bigraph represents a system at a single point in time. To encode the dynamics of a system we create a Bigraphical Reactive System (BRS) using a set of reaction rules of the form $L \longrightarrow\!\!\!\!\blacktriangleright R$, where L and R are bigraphs. Intuitively, a BRS operates by finding occurrences of L within a larger model and replacing these with R.

To determine if a bigraph L is present in a bigraph B we need the following definition, which applies to both concrete and abstract bigraphs.

Definition 2.1 (occurrence). A bigraph L occurs in bigraph B if the equation $B = C \circ (L \otimes \mathrm{id}_X) \circ D$ holds for some set of names X and bigraphs C and D. Two occurrences are equal if they differ only by a permutation or a bijective renaming on the composition interfaces; otherwise they are distinct.

The use of the identity bigraph id_X allows links to pass between C and D. An important property is that it is possible to determine an abstract occurrence starting from a concrete one. In other words, a bigraph L occurs in B only if an arbitrary concretion $\widetilde{L}$ occurs in an arbitrary concretion $\widetilde{B}$. In a given bigraph, there may be multiple occurrences of another bigraph. For example, in Figure 1c, bigraph G occurs three times within bigraph B.

In general, the decomposition corresponding to the occurrence of a given bigraph might not be unique. To ensure distinct decompositions, following Krivine *et al.* [KMT08], we introduce the following class of bigraphs.

Definition 2.2 (solid). A bigraph is *solid* if:

- All regions contain at least one node, and all outer names are connected to at least one link.
- No two sites or inner names are siblings
- No site has a region as a parent
- No outer name is linked to an inner name.

This definition is important when determining a suitable probability to apply a rule.

Definition 2.3 (reaction rule). A *reaction rule* R is a pair of bigraphs $\mathsf{R} = (L, R)$, written $L \longrightarrow\!\!\!\!\blacktriangleright R$, where R and L have the same interface and L is solid.

We also say $\mathsf{R} : L \longrightarrow\!\!\!\!\blacktriangleright R$ is *applicable* to g iff L occurs in g. In general, we are interested in applying a reaction rule within a larger bigraph and as such provide the following reaction relation.

Definition 2.4 (reaction relation). Given a reaction rule $\mathsf{R} : L \longrightarrow\!\!\!\!\blacktriangleright R$, the *reaction relation* $\longrightarrow\!\!\!\!\triangleright_{\mathsf{R}}$ over ground bigraphs is defined by

$$g \longrightarrow\!\!\!\!\triangleright_{\mathsf{R}} g' \qquad \text{iff} \qquad g = C \circ (L \otimes \mathrm{id}_X) \circ d \text{ and } g' = C \circ (R \otimes \mathrm{id}_X) \circ d$$

for some bigraph C, ground bigraph d, and set of names X.

Definition 2.5 (bigraphical reactive system (BRS)). A *bigraphical reactive system* is a pair $(\mathcal{B}, \mathcal{R})$, where $\mathcal{B}$ is a set of ground bigraphs and $\mathcal{R}$ is a set of reaction rules defined over $\mathcal{B}$. It has reaction relation

$$\longrightarrow\!\!\!\!\triangleright_{\mathcal{R}} = \bigcup_{\mathsf{R} \in \mathcal{R}} \longrightarrow\!\!\!\!\triangleright_{\mathsf{R}}$$

which will be written $\longrightarrow\!\!\!\!\triangleright$ when $\mathcal{R}$ is understood.

[2]In [Mil09], lean-support equivalence ($\asymp$) is used instead. It corresponds to support equivalence ($\simeq$) after discarding idle links, *i.e.* links connecting zero entities or names.

We indicate the set of reaction rules applicable to g and yielding g' with $\mathcal{R}_{g \twoheadrightarrow g'}$. We also introduce the following notation to count the *concrete occurrences* of a reaction rule R from g to g'

$$\sigma_{\mathsf{R}}[g, g'] = |\{\widetilde{g''} \mid \widetilde{g} \rightarrowtriangle_{\mathsf{R}} \widetilde{g''} \text{ and } \widetilde{g''} \simeq \widetilde{g'}\}|$$

that is, we count how many concrete $\widetilde{g''}$ that are support equivalent to a concretion of g' can be obtained by applying R to a concretion of g.

Definition 2.6 (transition system). A BRS with distinguished initial bigraph $g_0 \in \mathcal{B}$ forms a *transition system* with bigraphs as *states* and state transitions defined by generating all possible rewrites (reactions) in $\rightarrowtriangle$ from g_0 until we hit a fixed point (the set of all states $\mathcal{B}$).

This transition system view is useful for defining probabilistic, stochastic, and non-deterministic BRSs, where the transitions are assigned, for example, specific probabilities.

Finally, we allow states to be labelled by *bigraph predicates* which are also specified as bigraphs [BCRS16]. A state g satisfies a predicate bigraph P if P occurs in g. These can be used to, for example, identify invalid states for use in logical statements when performing verification.

2.3 Probabilistic Models

In the following we assume basic familiarity with probability theory, see for example [Bil12].

Probabilistic systems can be described using Markov models/processes, where the probability/rate of moving to a new state is based (strictly) on the current state [KNP10]. A discrete time Markov chain (DTMC) labels each state transition with a probability $0 \leq p \leq 1$ such that the sum of all outgoing edges from a state is equal to 1. That is, a DTMC draws the *next* state from a probability distribution of all possible states.

Definition 2.7 (Probability Distribution). A probability distribution over a countable set S is a function $\mu : S \to [0, 1]$ satisfying $\sum_{s \in S} \mu(s) = 1$

We use the notation $\mu = [s_0 \mapsto p_0, s_1 \mapsto p_1, \dots]$ to denote the distribution that chooses s_0 with probability p_0, and so on. We assume all other states are chosen with probability 0. To denote a set of probability distributions over S we use $\mathcal{D}_S$, dropping the subscript S if it is clear from the context. For verification purposes, we usually work with *finite* probability distributions with S finite.

Definition 2.8 (Discrete Time Markov Chain (DTMC)). A DTMC is a tuple (S, s_0, P) where S is a set of states, $s_0 \in S$ a distinguished initial state, and $P : S \to \mathcal{D}_S$ is a function assigning to each state $s \in S$ a probability distribution μ_s such that $\mu_s(s') : [0, 1]$ is the transition probability from s to s'.

As distributions cannot be empty, each state $s \in S$ has at least one transition. For terminal states s_t, we have $\mu_{s_t} = [s_t \mapsto 1]$ – the delta distribution.

To model continuous processes, we use continuous time Markov Chains (CTMCs) that assign stochastic *rates*, rather than probabilities, to state transitions.

Definition 2.9 (Continuous Time Markov Chain (CTMC)). A CTMC is a tuple (S, s_0, R) where S is a set of states, $s_0 \in S$ a distinguished initial state, and $R : S \times S \to \mathbb{R}_{\geq 0}$ the transition rate matrix matrix assigning a rate to each pair of states.

A transition between s and s' can only occur if $R(s, s') > 0$, and if so the probability of the transition occurring within time t is modelled as an exponential distribution, *i.e.* $1 - e^{-R(s,s')t}$. Unlike DTMCs, a CTMC allows terminal states where there is a 0 rate of transitioning.

Markov decision processes [Bel57, How60] model decision making in situations with both probabilistic outcomes and non-deterministic decision making. Intuitively, an MDP extends a

DTMC by allowing a *choice* of possible actions at each sate. Unlike a DTMC that provides a single probability distribution per state, the choice of action allows the multiple probability distributions per state.

Definition 2.10 (Markov decision process (MDP)). An MDP is a tuple $(S, s_0, A, Step)$ where S is a set of states, $s_0 \in S$ a distinguished initial state, A a set of *actions*, and $Step : S \to 2^{A \times \mathcal{D}_S}$ a function assigning to each state a set of possible actions with associated probability distributions.

Unlike a DTMC, we allow states with no outgoing transitions, *i.e.* $S \to \emptyset$. When the choice of action for each step is fixed an MDP is a DTMC.

To allow practical analysis of probabilistic models it is useful to define *rewards* associated with being in a particular state.

Definition 2.11 (state reward function). For a DTMC, CTMC, or MDP, a *state reward function* $r_s : S \to \mathbb{R}_{\geq 0}$ assigns to each state a reward. For states where rewards are not required r_s maps the state to 0.

When working with bigraphs we associate rewards with bigraph predicates, allowing state rewards to be defined as simply the sum of the rewards of all matching predicates (0 if no predicates occur).

For MDPs we can also associate a reward for choosing a particular action.

Definition 2.12 (action reward function). An *action reward structure* for an MDP $(S, s_0, A, Step)$ is a function $r_a : S \times A \to \mathbb{R}_{\geq 0}$ that assigns to each state, action pair a reward for performing that particular action. For actions where rewards are not required r_a maps the action to 0.

Although we call these *rewards*, they are often used to model *costs* associated with states/actions.

3 PROBABILISTIC BIGRAPHS

Given a state (bigraph), we want to control the probability of moving into a given *next* state (*i.e.* a bigraph). In other words, we require a DTMC where the states resulting from reactions are drawn from a probability distribution.

Our approach is similar to that of Bournez and Hoyrup [BH03] who consider abstract probabilistic rewrite systems. Here a *weight* is assigned to each rewrite rule which is then normalised based on which rules are applicable to a given state. Other approaches to modelling probabilistic systems are possible, for example, probabilistically determining which entities appear in the right-hand-side of a rule. We discuss these further in Section 6. Our approach allows re-use of existing probabilistic model checking tools such a PRISM [KNP11] or Storm [DJKV17a] for analysis/verification.

3.1 Probabilistic Bigraphical Reactive Systems

A probabilistic BRS adds weights to standard reaction rules to determine transitions probabilities when defining the reaction relation. This is, given a bigraph g_0 with $g_0 \rightarrowtriangle g_1$ and $g_0 \rightarrowtriangle g_2$ (for arbitrary rules) we wish to choose g_1 and g_2 from a probability distribution $\mu = [g_1 \mapsto p_1, g_2 \mapsto p_2]$, *i.e.* choose g_1 with probability p_1.

To account for multiple occurrences of a rule, we do not directly specify probabilities for rules but instead assign a *weight* which is then normalised to a probability.

Definition 3.1 (weighted reaction rule). A *weighted reaction rule* assigns to a reaction rule $L \longrightarrow R$ a *weight* $w \in \mathbb{R}_{\geq 0}$. We write weighted reaction rules as $L \xrightarrow{w} R$.

The *weight* determines how likely a particular rule is to be applied *relative* to all other (applicable) rules. Rules with weight $w = 0$ are never applied. In the following, we write $\rightarrowtriangle_{\mathcal{R}}$ to indicate that a

set of weighted reaction rules $\mathcal{R}$ is treated as a set of standard reaction rules (see Definition 2.5) by dropping all the weights.

Definition 3.2 (total weight). Given a set of weighted reaction rules $\mathcal{R}$, the *total weight from g to g'* is

$$\omega[g, g'] = \sum_{\mathsf{R} \in \mathcal{R}_{g \longrightarrow g'}} w_{\mathsf{R}} \cdot \sigma_{\mathsf{R}}[g, g']$$

where w_{R} is the weight of reaction rule R. Given a set of ground bigraphs $\mathcal{B}$, the *total weight from g* is

$$\omega[g, \_] = \sum_{g' \in \mathcal{B}} \omega[g, g']$$

Definition 3.3 (reaction probability distribution). Given a set of ground bigraphs $\mathcal{B}$, a set of weighted reaction rules $\mathcal{R}$ and $g \in \mathcal{B}$, the *reaction probability distribution from g* is

$$\mu_g = [g_0 \mapsto p_0, g_1 \mapsto p_1, \ldots] \qquad \text{with} \qquad p_i = \frac{\omega[g, g_i]}{\omega[g, \_]}$$

for every $g_i \in \mathcal{B}$ such that $g \rightarrowtriangle_{\mathcal{R}} g_i$. If there are no such g_is, then $\mu_g = [g \mapsto 1]$.

Reaction probability distributions are then used to define a probabilistic reaction relation over ground bigraphs.

Definition 3.4 (probabilistic bigraphical reactive system (PBRS)). A *probabilistic BRS* is a pair $(\mathcal{B}, \mathcal{R})$, where $\mathcal{B}$ is a set of ground bigraphs, and $\mathcal{R}$ is a set of weighted reaction rules. It has *probabilistic reaction relation* defined by

$$g \xrightarrow{p} g' \qquad \text{iff} \qquad \mu_g(g') = p$$

with $g, g' \in \mathcal{B}$.

The correspondence between PBRS and DTMC is as follows:

Lemma 3.5. *A PBRS $(\mathcal{B}, \mathcal{R})$ is a DTMC $(S, s_0, P : S \to \mathcal{D}_S)$.*

Proof. Take $S = \mathcal{B}$, $s_0 = g_0 \in \mathcal{B}$, and $P(s) = \mu_g$ (Theorem 3.3) for $s \in S$, $g \in \mathcal{B}$, and $s = g$. □

From a practical standpoint, the use of weighted reaction rules allows modelling only the *relative* probability a particular rule is executed. Unfortunately, this makes it difficult to specify an *exact* probability between states. Doing so is often impractical, requiring significant effort to control the applicable rules and number of occurrences such that the normalised probabilities are exact. Usually the relative outcomes are what is important, and so far we have not encountered any situation where this is a particular issue (see Section 6.2).

3.2 Example PBRS

Consider a Wireless Sensor Network (WSN) with three sensor nodes (S) and a base-station (BS), as shown in Figure 2. The base station is represented by the rectangle and the three sensors are represented by circles. There is a link between the base station and the sensors. Due to hostile deployment environments, sensors often fail. We model failure using the reaction rule `fail` that marks a[3] sensor as failed (red circle) and unlinks it from the base-station. The rule `recover` allows a failed sensor (red circle) to re-connect with the base-station. `fail` (b) has weight w_f and `recover` (c) has weight w_r. Note that while `fail` and `recover` are behaviourally *inverse* rules, their weights differ.

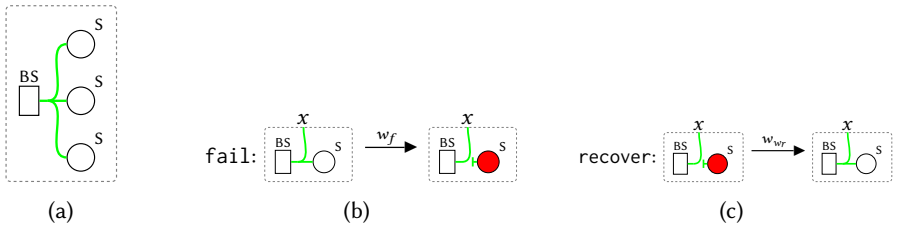


Fig. 2. Wireless sensor networks as bigraphs. (a) Bigraph representing a WSN with base-station BS and three sensors S. (b) Probabilistic reaction rule modelling failure of a sensor. (c) Probabilistic reaction rule modelling recovery of a sensor.

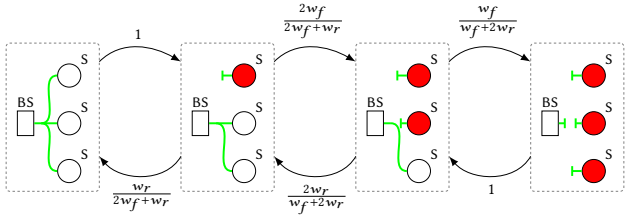


Fig. 3. DTMC for bigraph model of Figure 2. States are bigraphs $g_0 \ldots g_3$ from left to right.

The resulting transition system for this WSN is in Figure 3. From the initial state g_0, we determine the distribution of next states μ_{g_0}. In this case, reaction rule `recover` is not applicable and we can only apply `fail`. It may be surprising that even though there are three sensors, the probability of transitioning to g_1 is $\mu_{g_0}(g_1) = 1$. This is because support equivalent (concrete) bigraphs are combined when computing the states resulting from the application of a reaction rule. To see this more clearly, we show explicitly the three concrete occurrences of `fail` from g_0 to g_1 in Figure 4. The key observation is that through renamings $v_2 \longrightarrow v_1$ and $v_3 \longrightarrow v_1$, these concrete bigraphs are support equivalent and therefore they correspond to the single abstract state g_1. Hence, we have $\sigma_{\texttt{fail}}[g_0, g_1] = 3$, $\omega[g_0, g_1] = 3w_f$, and $\omega[g_0, \_] = 3w_f$, giving the overall reaction probability of 1.

In state g_1, with one failed sensor, we have $\sigma_{\texttt{fail}}[g_1, g_2] = 2$ and $\sigma_{\texttt{con}}[g_1, g_0] = 1$. Normalising this over the total weight $\omega[g_1, \_] = 2w_f + w_r$ we obtain a $\frac{2w_f}{2w_f+w_r}$ probability that another sensor fails, and a probability $\frac{w_r}{2w_f+w_r}$ that the failed sensor recovers.

Importantly, due to support equivalence, transition probability corresponds to the probability that **any** sensor fails rather than the probability that a **particular** sensor fails (*i.e.* $\frac{w_f}{2w_f+w_r}$).

This process of normalising weights to probabilities continues until we obtain the full DTMC as shown.

4 ACTION BIGRAPHS

PBRSs allow a single distribution of possible next states defined over *all* rules $\mathcal{R}$. However, for systems such as controllers, we want actions taken by the controller to affect the possible evolution of the system, *e.g.* by restricting the reaction rules. That is, we want multiple distributions that are

[3]This rule only allows one sensor to fail per rewrite. If concurrent failures are required then additional rules are needed (explicitly matching on n sensors).

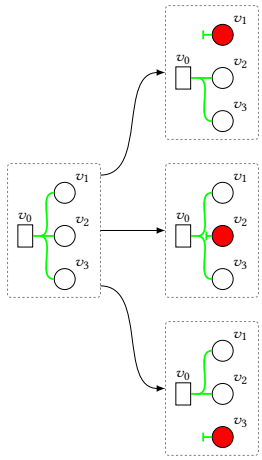


Fig. 4. Concrete occurrences of reaction rule `fail` from g_0 to g_1 (see Figure 2).

determined by the action taken. To this end, we introduce *action bigraphical reactive systems* (ABRS), where the resulting transition system is a Markov decision processes (MDP – Theorem 2.10).

ABRSs extend PBRSs by allowing a *choice* of probability distributions at each step. We call such choices *actions*.

Definition 4.1 (Action). An *action* is a non-empty set of weighted reaction rules (Theorem 3.1) that determines the rewrites that can be performed *if* the action is chosen. We say an action is *applicable* to a bigraph g if at least one rule from the action is applicable to g.

As actions are simply sets of rules, the same reaction rule may appear in multiple actions if required, *e.g.* if two different control actions allow updating of the same state. We use the notation $\restriction_A$ to restrict definitions to consider only rules in A. For example $\omega[g, g_i] \restriction_A$ is the total weight between states g and g_i when considering only rules in $A \subseteq \mathcal{R}$ rather than in $\mathcal{R}$.

To move from weighted to probabilistic rules, we apply, individually for each action, the normalisation procedure from PBRSs, *i.e.*

$$p_i = \frac{\omega[g, g_i] \restriction_A}{\omega[g, \_] \restriction_A}$$

After normalising, we obtain a set of probability distributions; one for each applicable action, allowing us to construct the MDP transition function $Step : S \to 2^{A \times \mathcal{D}_S}$.

We then define an Action BRS as follows.

Definition 4.2 (Action BRS (ABRS)). An *Action BRS* is a triple $(\mathcal{B}, \mathcal{R}, \mathcal{A})$, where $\mathcal{B}$ is a set of (ground) bigraphs, $\mathcal{R}$ is a set of weighted reaction rules over $\mathcal{B}$, and $\mathcal{A} = \{A_0 \subseteq \mathcal{R}, \ldots, A_n \subseteq \mathcal{R}\}$ is a set of actions. It has a *reaction relation* defined by

$$g \xrightarrow{(A_i,\, p)} \triangleright\, g' \qquad \text{iff} \qquad \mu_g(g') \restriction_{A_i} = p$$

for each applicable action $A_i \in \mathcal{A}$ with $g, g' \in \mathcal{B}$.

Just as a PBRS is a DTMC, an ABRS is an MDP:

Lemma 4.3. *An ABRS* $(\mathcal{B}, \mathcal{R}, \mathcal{A})$ *is an MDP* $(S, s_0, A, Step : S \to 2^{A \times \mathcal{D}_S})$.

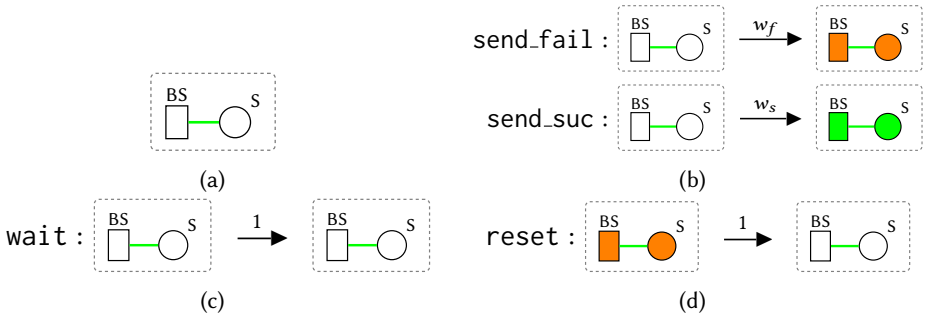


Fig. 5. Non-Deterministic Bigraph Example. (a) Initial bigraph. (b) A_{send} – Attempt send, fails (orange) or succeeds (green). (c) A_{wait} – do nothing. (d) A_{reset} – Get ready to send again.

Proof. Take $S = \mathcal{B}$, $s_0 = g_0 \in \mathcal{B}$, $A = \mathcal{A}$, and for $s = g$ with $s \in S$ and $g \in \mathcal{B}$ define

$$Step(s) = \begin{cases} \{(A_i, \mu_g \restriction_{A_i}), \ldots, (A_n, \mu_g \restriction_{A_n})\} & \text{for all } A_i \in \mathcal{A} \text{ applicable to } g \\ \emptyset & \text{if no action applies to } g \end{cases}$$

□

As with MDPs, we can assign rewards (Theorem 2.12) for choosing a particular action to allow optimisation of decision processes.

Like MDPs, ABRS allow terminal states (the empty set of distributions) if there is no applicable action, however, for practical analysis, *e.g.* in PRISM, we usually require at least one action per state. In a similar manner to PBRS, in the case no *action* applies, we can add an trivial action containing the identity reaction for the current state.

4.1 Example ABRS

As an example, consider the model in Figure 5 representing another simple WSN. In this case, data can be sent between the sensor (S) and base-station (BS), and there is a non-deterministic *choice* whether the sensor should send data or wait.

There are three actions: $A_{send} = \{\texttt{send\_suc}, \texttt{send\_fail}\}$, $A_{wait} = \{\texttt{wait}\}$, and $A_{reset} = \{\texttt{reset}\}$. The resulting Markov decision process is shown in Figure 6. From the initial state g_0 the system has a choice of two actions: A_{send} or A_{wait}. If the system *chooses* to send then the distribution of states is $\mu_{g_0} \restriction_{A_{send}} = [g_1 \mapsto \frac{w_s}{w_s+w_f}, g_2 \mapsto \frac{w_f}{w_s+w_f}]$, while on a wait it is $\mu_{g_0} \restriction_{A_{wait}} = [1 \mapsto g_2]$. In the case the send fails an additional action, A_{reset} reinitialises the state to allow another attempt.

5 EXTENDED BIGRAPHER IMPLEMENTATION AND FURTHER EXAMPLES

BigraphER[4] [SC16] is an open source toolkit for bigraphs. It provides a language for describing bigraphs and reaction rules algebraically (as a `.big` file) and support for simulating the resulting BRS or generating the transition system in PRISM [KNP11] input format, which is also employed by other model checkers. We have extended BigraphER to support standard, probabilistic, stochastic, and action bigraphical reactive systems, which enables quantitative model checking for probabilistic (DTMC), stochastic (CTMC) and action (MDP) BRSs. The extensions are included in the main BigraphER release from version 1.3.4 onwards and are openly available[4].

[4]Available online: https://uog-bigraph.bitbucket.io/

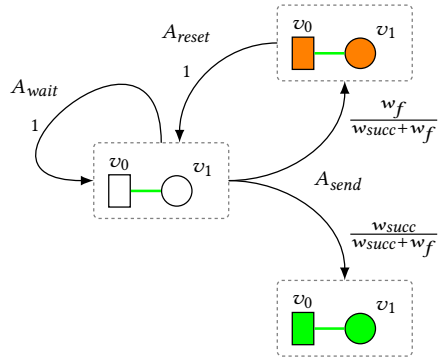


Fig. 6. Markov Decision Process for the example of Figure 5. State g_0 left, g_1 bottom and g_2 top.

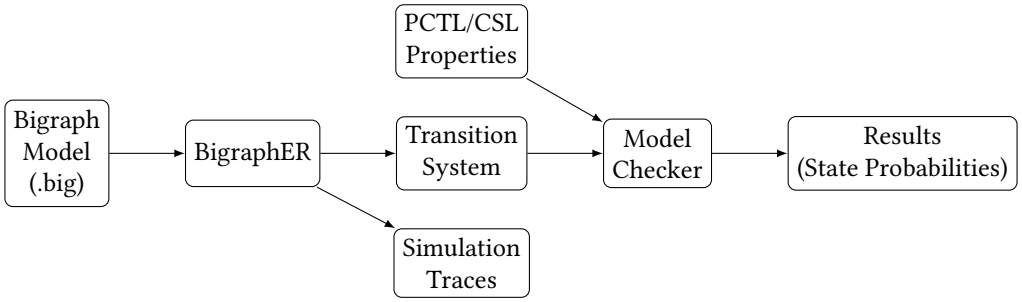


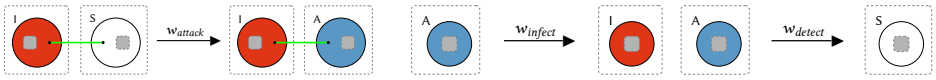
Fig. 7. Overview of probabilistic, stochastic and action BRS analysis.

The export is fully general and requires no additional simplification steps after the transition system in built. A diagrammatic overview is in Figure 7.

The transition system is generated by a breadth first search from a given initial state, new states are generated from successful reaction rule applications (respective priorities etc.). Equivalent states are merged and so it is possible to generate transition systems for some infinite systems. For non-terminating or looping systems we allow a bounded number of states to be generated.

We present examples of probabilistic (DTMC), stochastic (CTMC), and action (MDP) BRSs, showing selected BigraphER implementation details and simulation and PRISM results. Section 5.1 gives a probabilistic BRS model of virus spread through a computer network where firewalls are breached probabilistically, Section 5.2 gives a stochastic BRS model of membrane budding, and in Section 5.3 we give an action BRS that models decisions in wireless sensor networks with mobile sinks: sensors decide whether to send immediately, or wait in the hope the sink moves closer. The model in Section 5.2 follows the example presented in the original definition of Stochastic Bigraphs [KMT08]. But at that time there was no tool support to analysis, instead requiring a manual translation to a PRISM model. Here, we revisit this model and for the first time, giving a direct implementation in bigraphs.

In the following we introduce the syntax of BigraphER by example, highlighting the new language constructs. A full reference is available [SC16]. Model files are available [ASC].



(a) attack (b) infect (c) detect

Fig. 8. Virus spread with entities S = Safe, I = Infected, A = Attacked. (a) Attempt to attack an uninfected neighbour. (b) Firewall fails, node becomes infected. (c) Firewall succeeds.

5.1 Probabilistic Bigraphs – Virus Spread in a Network

Viruses spread through computer networks in a probabilistic manner. The probability of infection may differ between nodes, depending on the effectiveness of firewalls. This example is based on an existing PRISM model [KNPV09].

The problem we address is in an arbitrary computer network, with the infection starting at a specified node, we wish to determine the likelihood of a virus infecting a particular node. We model the following infection process. Infected nodes attack, spreading the virus to uninfected neighbours. All (uninfected) neighbours are equally likely to be targets, *i.e.* the rule has *fixed* weight w_{attack}. A node that has been attacked is either protected by a firewall with weight w_{detect}, or become infected with weight w_{infect}. As we do not directly specify the probabilities, the important measure is the *ratio* of w_{detect} to w_{infect}. This differs from the example PRISM model that adopts the more common approach of setting $p_{detect} = 1 - p_{infect}$.

The bigraph model is in Figure 8. An *infected* node attacks a neighbouring (through the green link) *safe* node (S, clear circle) with weight w_{attack}. An attacked node (A, indigo circle) enters the *attacked* state and from there has a w_{infect} weight of becoming infected (I, red circle), or w_{detect} of the firewall breaking the infection attempt.

Unlike Safe-Infected-Recovered (SIR) models [Het00], infected nodes remain infected for the entire run, *i.e.* there is no recovery, and there is no resistance buildup, *i.e.* if a firewall stops an infection the same node may still be infected later.

5.1.1 Syntax. A model (snippet) in BigraphER syntax is shown in Listing 1. Entities are defined in lines 2-3 using keyword `ctrl` where the number after the equality symbol is the *arity* (number of links). Rewrite rules are specified using keyword `react` and a name followed by a rule in the form B `-->` B' where B, B' are arbitrary bigraphs (written in a form closely resembling the bigraph algebra).

For probabilistic/stochastic rules we extend the notation `-->` to `-[l]->` where `l` is a label for the rule, *e.g.* a weight or a rate (a positive float expression). For example the rule `infect` in line 9 says that a bigraph matching `A` rewrites to `I` with weight `w_infect`.

A PBRS is defined by construct `begin pbrs ... end` which also specifies the initial state, the state predicates (*e.g.* `all_infected` that labels a state with at least 9 `I` entities), and the set of reaction rules (omitted).

5.1.2 Model analysis. Following the original PRISM model, we consider nine nodes connected in a square grid layout (Figure 9b), using BigraphER to export the full (probabilistic) transition system for analysis in PRISM. Figure 9a shows the probability that all nine nodes of the system are infected within the first n timesteps (reactions), *i.e.* the results of checking $\mathcal{P}_{=?}\left[\,\mathrm{F}^{\leq n}\,\texttt{all\_infected}\,\right]$. The predicate `all_infected` is defined on line 13 of Listing 1 matching all nine infected nodes. `big` is a keyword and the iterated operator `par` allows us to place concisely n bigraphs side-by-side inside the same region[5]. We vary the detection weight w_{detect} of the firewalls (5, 10, and 15) and,

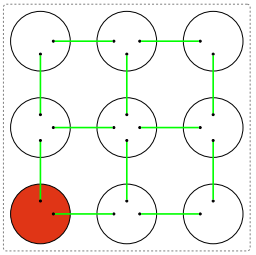
[5]This corresponds to the iterated *merge product* operator in the algebraic form of bigraphs.

Listing 1. Probabilistic rules for virus spread in BigraphER (snippet).

```
# Defines entities with 0 arity (links)
ctrl A = 0;
ctrl I = 0;

float w_infect = 5.0;

# Probabilistic rule with weight w_infect
# Fig Figure 8b
react infect = A -[w_infect]-> I;

# Predicate
# par(n,B) = B | B ... | B (n times)
big all_infected = par(9, I);

begin pbrs
  ...
end
```

Max Steps	w_{detect} 5	10	15
50	$4.47 \cdot 10^{-2}$	$1.22 \cdot 10^{-3}$	$9.59 \cdot 10^{-5}$
100	$6.09 \cdot 10^{-1}$	$8.03 \cdot 10^{-2}$	$1.18 \cdot 10^{-2}$
250	$1 \cdot 10^{0}$	$8.88 \cdot 10^{-1}$	$5.24 \cdot 10^{-1}$
500	$1 \cdot 10^{0}$	$9.97 \cdot 10^{-1}$	$9.88 \cdot 10^{-1}$

(a)



(b)

Fig. 9. (a) Probability of full infection in n steps. (b) Initial topology.

(a) detect_basic (b) detect_advanced

Fig. 10. Modified detect rule accounting for firewall types

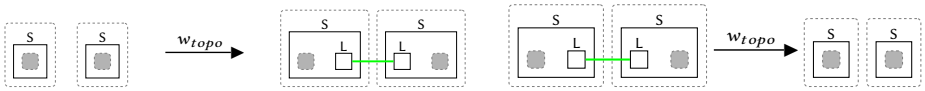
as expected, increasing $w_{dectect}$, *i.e.* adding better firewalls, reduces the probability that all nodes become infected within a given time period.

We now consider *extending* the model in two ways: adding new behaviour and allowing dynamic topologies.

5.1.3 Adding new behaviour. As shown, better firewalls reduce the time to whole system infection. However, this can be prohibitively expensive in a deployed system. Alternatively, we may wish to improve the firewalls on specific nodes. To test this, we add new entities BasicFW (light green squares) and AdvFW (yellow squares) representing a basic and more advanced (and expensive) firewall. We then split the detect rule into two rules, as shown in Figure 10 to account for the type of firewall. Because the sites abstract away internal node structure, the other rules remain the same. The only other change is to place the firewalls in the starting topology – in this case we place advanced firewalls in the nodes of the middle row.

Table 1. Probability of full infection in n steps when using 3 improved firewalls. In all cases $w_{detectBasic} = 5$.

	$w_{detectAdv}$		
Max Steps	5	10	15
50	$4.47 \cdot 10^{-2}$	$1.19 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$
100	$6.09 \cdot 10^{-1}$	$3.12 \cdot 10^{-1}$	$1.67 \cdot 10^{-1}$
250	$1 \cdot 10^{0}$	$9.88 \cdot 10^{-1}$	$9.29 \cdot 10^{-1}$
500	$1 \cdot 10^{0}$	$9.99 \cdot 10^{-1}$	$9.99 \cdot 10^{-1}$



(a) `create_neighbour` (conditional bigraphs unshown) (b) `remove_neighbour`

Fig. 11. Reaction rules for dynamic topologies.

The results of re-running the analysis are in Table 1. As expected, when $w_{detectBasic} = w_{detectAdv}$ the results match the previous experiment. Results like these can be used in a cost-benefit analysis, e.g. when determining how many advanced firewalls to deploy.

An advantage of bigraphs is this extension is more straightforward than the equivalent in PRISM. This is because sites abstract away from internal node structure, whereas the PRISM model has to encode the internal structure explicitly in modules: one for each type of node, *e.g.* corner (2 neighbours), edge (3 neighbours), and middle (4 neighbours). Each module has to be modified to add the new behaviour, and then instantiated as appropriate, in the starting topology.

5.1.4 Dynamic Topologies. The lack of abstraction over node structure in the PRISM model has further implications because model complexity scales with the topology complexity. For example, if we require a node with 5 neighbour links, then we need an additional module.

Because the modules have a fixed neighbourhood linking structure, and there is a fixed number of modules, it is not possible to easily model dynamic topologies, *e.g.* where either the neighbourhood linking structure, or the number of nodes, changes. In bigraphs we have no such constraints: sites allow rules to apply in a variety of specific instances, and nodes can be dynamically added/removed as the model evolves.

We illustrate by extending our virus model to allow dynamic neighbourhood changes. Specifically, we allow neighbour links to be broken or created at any time, provided neither of the nodes is currently under attack or infected, *e.g.* the attacker has (partial) control so can block changes. This is expressed by two new rules: `create_neighbour` and `remove_neighbour`, given in Figure 11. For simplicity, we assume both creating and removing a neighbour link is equally likely by giving them the same shared weight *w_topo*. Further analysis could be performed with different weights if required. Not shown here, we also have conditions on the creation rule expressing what is/is not within the sites, to ensure: i) that no pair of nodes connects with more than one neighbour link, and ii) that there is a maximum of n neighbours per node (other than nodes that *start* with $m > n$ neighbours). For readability we do not show the conditions explicitly here (conditional bigraphs and the BigraphER extension are defined elsewhere [ACS20]).

Note that since topology changes are possible at every step and we only change neighbour links between safe nodes, it is possible that the system never becomes fully infected, *i.e.* it is possible to

Table 2. Simulated (up-to 1000 rewrite step) virus spread model with dynamic topology based on Max neighbours and w_topo. Based on 100 runs per configuration. Mean Inf Steps is the number of non-topology change steps needed to fully infect the system.

Max Neighbours	w_topo	All Inf	Mean Inf Steps
2	0.01	92	129
2	0.1	40	110
2	0.5	21	129
3	0.01	91	127
3	0.1	70	117
3	0.5	63	126
4	0.01	97	125
4	0.1	89	117
4	0.5	92	126

evolve into disconnected networks. This can occur when a safe node is only connected to infected nodes and is attacked before it can reconnect to the safe network (before create_neighbour fires).

Consider analysis of the new model. Given the high number of possible states *i.e.* all possible neighbourhood topologies for all possible node states, instead of performing full model checking we use the simulation capabilities of BigraphER to gain insight into the model. We vary two parameters: i) the weight w_topo assigned to topology changes, and ii) the maximum number of (newly created) neighbours we allow for a node.

Consider the implications of each. First, a higher weighting for topology changes, *i.e.* w_topo, would increase the amount of disruption in the initial topology. This is because more topology changes increase the likelihood of a node being connected to only infected nodes just before an attack, which means greater likelihood of disconnected networks and thus an overall safer system Second, by changing the maximum number of neighbours for a node, we essentially change the width of the network. We expect wider networks (those with more neighbours) to propagate infections more easily and overall be less safe[6].

Table 2 shows how w_topo and max links affect the likelihood of full infection. Results are based on 100 simulations of, up-to 1000 rewrite steps, per configuration. The results are as we expect: for low values of w_topo, *i.e.* less network randomisation, given the initial state is well connected, in most cases (more than 90 from 100) the network is fully infected. As we increase the weight, and so the expected number of new configurations, the likelihood of fully infecting the network lowers (assuming a low enough network width). Likewise, as expected, reducing the maximum number of (new) links we allow for each node, increases the likelihood the network becomes disconnected. The mean infection steps, which include non-topology rewrites only, are fairly constant throughout. This is not unexpected, as a minimal number of steps are required to infect a 9 node connected topology, regardless of the specific setup.

The main point of this extension is not the actual analysis results, but to illustrate the ways in which bigraphs offer advantages over existing formalisms.

5.2 Stochastic Bigraphs – Membrane Budding

Stochastic Bigraphs were first introduced by Krivine, Milner, and Troina [KMT08], and allow a CTMC (Theorem 2.9) as the underlying transition system. Instead of assigning *weights* to reaction

[6]In practice there are key trade-offs between safety and performance to be made.

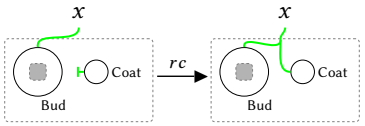


Fig. 12. coat (recreated from [KMT08])

rule (as in PBRS), they assign a *rate* to each reaction rule. Because these rates are defined over rules, not states, similar to PBRS, a normalisation procedure is required to calculate a final exit rate based on the number of occurrences of each (applicable) rule.

Krivine et al [KMT08], present as an example, a stochastic model of *membrane budding* that describes a biological mechanism for particles to move between cells. Coat proteins form on the surface of a membrane and, given enough coat proteins, a bud can form, accept some particles, and finally break off to carry particles out of a cell. Stochasticity derives from rates of coating, particle movement (into/out of the bud), and fission (breaking away).

No implementation of stochastic bigraphs was available when the example was developed, and analysis was given using a hand-coded PRISM model. As BigraphER now supports stochastic bigraphs, we revisit this example to recreate their analysis directly in bigraphs. We show a snippet of the model here, and refer the reader to the original source for full details of the model. Support for stochastic bigraphs is additionally shown by an analysis of the 802.11 CSMA protocol in [CS14]

We focus on the rule `coat` (rule 2 [KMT08]) that allows free Coat proteins to form on a Bud. The rule is shown in Figure 12. Free Coat proteins are distinguished from those already forming on a bud by the use of a closed link. The site abstracts over the particles within the Bud.

The rate *rc* determines how quickly Coats form on the bud. Although it looks constant, because of the normalisation procedure, it in fact depends on the number of free coats, *i.e.* the number of occurrences increases and, in turn, so does the rate the rule applies. This is similar to the example in Section 3.2 where the probability *any* sensor failed was much higher than the probability a specific single sensor failed.

BigraphER code for the `coat` rule is given in Listing 2. Stochastic rules are specified in a similar way to probabilistic rules in BigraphER by placing a float expression between the left and right-hand side of a rule (*i.e.* `rc` on line 13; as described in Section 5.1.1). Similarly, the initial state and the reaction rules of an SBRS are defined by construct `begin sbrs ... end`. The use of `fun big` defines a *family* of bigraphs, in this case one for each value of the parameter n (allowing states to be labelled with the number of particles currently in the bud).

For large models *e.g.* those with 50 free Coat entities, it is time consuming to count occurrences. To overcome this, we use a population model (a *counter abstraction*) that groups free coats etc. into a single entity Coats representing the number bound, and Fcoats representing the number that are still free. This allows `coat` to be alternatively written as shown in Figure 13. Entities such as Coats(c) are *parameterised entities* that can be seen as defining a family of Coats entities, one for each possible value of c. To calculate the number of free coats (Fcoats), we use the constant c_{max} – the total number of coat proteins (free or bound); in our case 50. Coats are defined in the BigraphER language by the `fun ctrl` keywords. Similarly we augment the other rules to, for example, count the number of Proteins in the Bud. In this case we cannot rely on the occurrence count to determine the correct rate, so instead we explicitly scale the rate based on the number of remaining free coats.

For analysis, we define a family of bigraph predicates `particles`(n), with $0 \leq n \leq 40$, that match a Bud with exactly n particles inside, allowing the probability that we end with n particles in a bud

Listing 2. Specifying stochastic rules in BigraphER (snippet).

```
1   ctrl Bud = 1; # Arity 1
2   ctrl Coat = 1;
3   ctrl Gate = 1;
4   ctrl Particle = 0; # No links
5
6   # Coating rate
7   float rc = 1.0;
8
9   # Stochastic reaction rule
10  # Fig Figure 12
11  react coat =
12    Bud{x}.id  | /y Coat{y}
13    -[rc]->
14    Bud{x}.id | Coat{x};
15
16
17  # Predicate used for plotting.
18  # Determines the number of particles that have been transferred from a membrane when the bud
        breaks free
19  # par(n, b) = b | ... | b (n times)
20  fun big particles(n) = Bud{x}.(par(n, Particle));
21
22  begin sbrs
23    ...
24  end
```

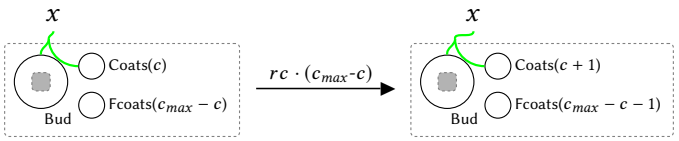


Fig. 13. coat – population model.

(that has broken off) to be determined through the CSL formula: $\mathcal{P}_{=?}[\,\mathbf{F}\,\texttt{particles}(n)\,]$. This is defined in line 20 of Listing 2 by using the `fun` keyword.

We exported the BigraphER model to PRISM, where we successfully reproduced the "Particles in the formed bud" results from the appendix of stochastic bigraphs [KMT08], as shown in Figure 14. This figure shows the number of particles that are in the bud after fission, where rd is the rate of diffusion of particles between the membrane and the bud (rule 3 [KMT08]). As expected, increasing the diffusion rate increases the expected number of particles in the formed bud. The rate of fission depends on the number of coat proteins (more coat proteins implying higher fission rates), and hence for $rc = 2$ the overall effect is to have less expected particles in the bud.

5.3 Action Bigraphs – Mobile Sinks in Wireless Sensor Networks

We use action bigraphs to model a well known decision problem in wireless sensor networks [BT08]. Traditionally, WSNs use multi-hop communication to move data between sensors and fixed sink (base-station). An alternative approach is for a *moving* sink to collect data directly from the sensors. Such an approach can be used, for example, when robots obtain information as they move through a space.

Given limited battery and memory capabilities of a sensor, when the sink moves into range a decision must be made: should the sensor send immediately, with high transmit power, or should it wait in the hope the sink moves closer, risking losing data by exhausting memory if the sink moves out of range before transmission can occur? Such decisions can be modelled as *actions*.

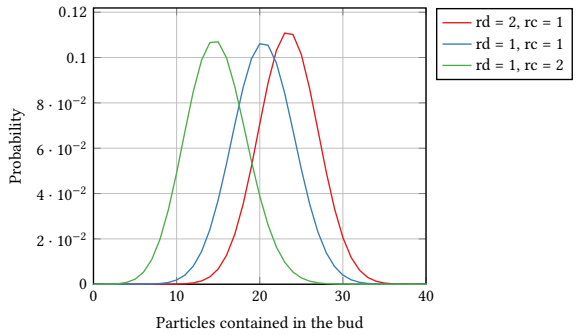
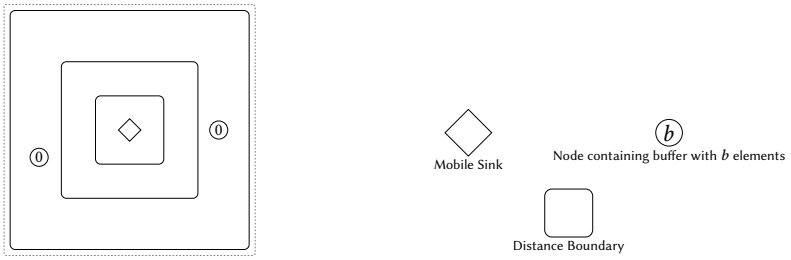


Fig. 14. Reproducing "Particles in the formed bud" figure from [KMT08] directly in BigraphER.



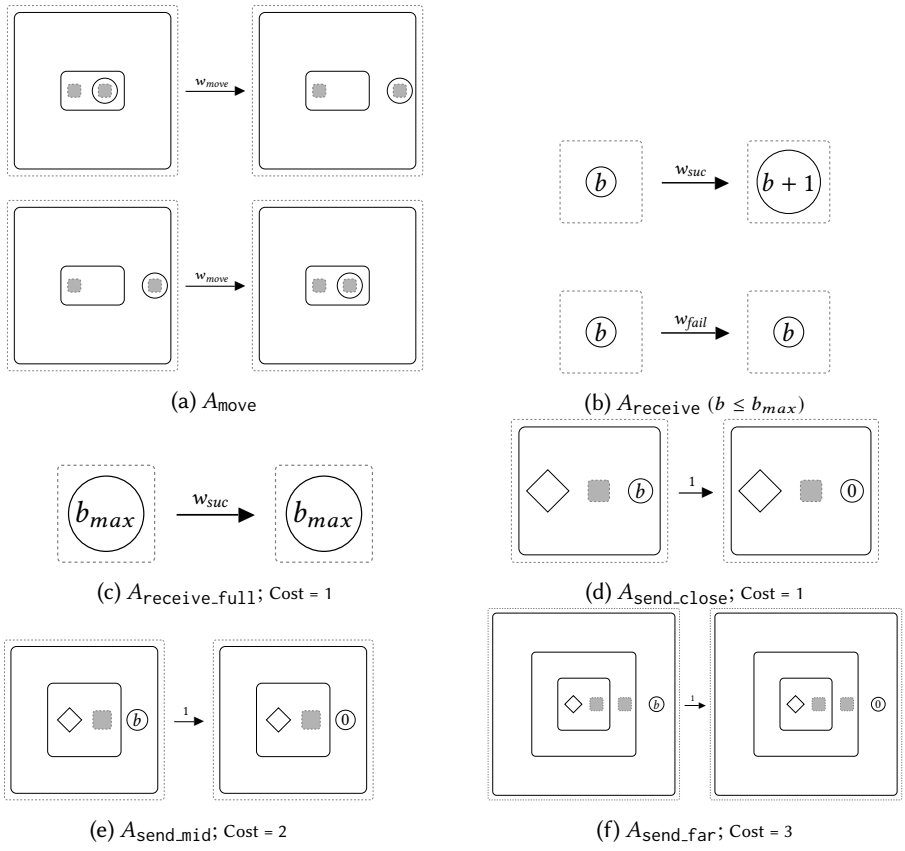
(a) Initial Bigraph (b) Entities

Fig. 15. Send now or later: entities and initial bigraph.

Model entities are shown in Figure 15. There are three types: a (unique) Sink; a sensor Node (that we refer to as nodes) that includes a Buffer with the number of slots filled; and Distance boundaries abstractly representing how close a sensor is to the sink, *e.g.* close, mid-range, or far. Nodes outwith the last Distance entity are considered out-of-range and cannot send. In our example we assume the initial setup given in Figure 15a with 3 distances (close, mid, far), one sink, and two nodes with initially empty buffers. A benefit of the modelling approach is the initial state can be changed, *e.g.* to include nodes with preexisting data, without requiring rule changes.

The model updates in two phases: *movement*, where Nodes (possibly) move between Distance entities, and *act*, where Nodes non-deterministically *decide* whether to send or wait and receive data. Phases are scheduled in a round-robin fashion with each Node taking an action before moving onto the next phase. Phases are modelled using the turn taking technique described in [ACS20] where additional entities (placed in their own parallel region) determine the current phase, *e.g.* Send_Phase, Move_Phase. That is, for a rule $L \longrightarrow\!\!\blacktriangleright R$, we extend to $L \parallel \mathit{PhaseX} \longrightarrow\!\!\blacktriangleright R \parallel \mathit{PhaseY}$ for two phases (possibly equal) X and Y. Sometimes additional rules are added that only change phase, *e.g.* when no other rule applies. For clarity we do not describe the turn-taking in detail, but it is available in the online models [ASC].

Sink movement is modelled by moving Nodes between Distance boundaries, as shown in the reaction rules of Figure 16a. Recall an action is a set of rules (shown here one per line). These rules models movement relative to the Sink, *i.e.* a Node moving to a closer Distance boundary implies a physical Sink move towards the Node. Additional rules (unshown) move Nodes in/out of the outermost Distance boundary.



(a) $A_{\texttt{move}}$ (b) $A_{\texttt{receive}}$ $(b \leq b_{max})$

(c) $A_{\texttt{receive\_full}}$; Cost = 1 (d) $A_{\texttt{send\_close}}$; Cost = 1

(e) $A_{\texttt{send\_mid}}$; Cost = 2 (f) $A_{\texttt{send\_far}}$; Cost = 3

Fig. 16. Send now or later: actions, associated reaction rules (one per line, shown above the action label), and costs.

Non-determinism in the operation phase comes from the decision to send existing data, through one of the $A_{\texttt{send\_close}}$, $A_{\texttt{send\_mid}}$, $A_{\texttt{send\_far}}$ actions, or to receive new data through one of the $A_{\texttt{receive}}$, $A_{\texttt{receive\_full}}$ actions. Using multiple actions to model send/receive allows multiple action reward structures to be used. $A_{\texttt{receive}}$ is shown in Figure 16b and consists of two reaction rules representing receipt or failure of new data with weight $w_{receive}$. When the buffer is full ($b = b_{max}$), $A_{\texttt{receive}}$ is no longer applicable, and instead $A_{\texttt{receive\_full}}$ (Figure 16c) represents a dropped sample and unit cost is incurred.

Actions/rules for sending data in Figures 16d, 16e and 16f. In each case the buffer is fully emptied and a cost is incurred proportional to the distance from the sensor. We assume the cost is constant regardless of how much data is sent *i.e.* data always fits in a single radio packet. A more complex model could account for this through additional actions. Sending data is always successful but out-of-range sensors are unable to send any data, *i.e.* we do not fall back to hop-to-hop communication, and must always do a receive.

5.3.1 Syntax. To specify ABRSs we extend the BigraphER language to include an explicit `actions` declaration within `begin abrs ... end` blocks, as shown in Listing 3 (line 20). Each action consists of an identifier, *e.g.* `receive`, `receive_full`, followed by a set of reaction rules, specified

Listing 3. Specifying actions BigraphER (snippet).

```
ctrl Sensor = 0;
fun ctrl Buffer(x) = 0;
fun ctrl Iden(i) = 0;

float w_suc = 5.0;
float w_fail = 1.0;

# Fig Figure 16b (top)
fun react receive(x,i) =
  Sensor.(Buffer(x) | Iden(i))
  -[w_suc]->
  Sensor.(Buffer(x + 1) | Iden(i))

# Fig Figure 16b (bottom)
fun react receive_fail(x,i) =
  Sensor.(Buffer(x) | Iden(i))
  -[w_fail]->
  Sensor.(Buffer(x) | Iden(i))

begin abrs
  ...
  actions = [
    # Action[cost] = { rules }
    receive = {receive(x,i), receive_fail(x,i)},
    receive_full[1] = {receive(bmax,i)},
  ];
end
```

by their identifiers. We may optionally assign a reward to each action, *e.g.* `receive_full[1]` has reward/cost 1.

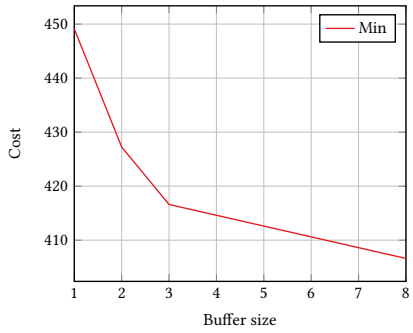
5.3.2 Model Analysis. The model is analysed by exporting the MDP transition system from BigraphER. Unfortunately, PRISM does not support importing action rewards from a transition system, so we encode action rewards through state rewards that are fully supported. To do this we add additional entities (unshown) to the model when actions are taken, *e.g.* a SendClose entity if a sensor sends when the sink is close. These entities can be matched using state predicates to increase the cost. Importantly, this is an implementation detail to overcome tool support, and does not invalidate the theory.

We assume a single sink, two sensors, and 4 distance boundaries: close-range, mid-range, far-range, and out-of-range. Both sensors start in far-range of the sink.

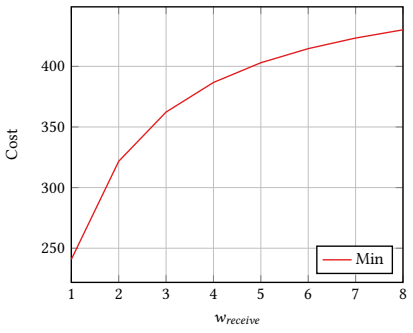
First, consider the effect of increasing the maximum buffer size on the total cost, with the expected result being that an increased buffer size should reduce the overall cost, since sensors can wait longer before deciding to send data. Figure 17a shows how increasing the maximum buffer size effects the minimum possible cost in the first 4000 transitions, using the PCTL formula $\mathcal{R}_{min=?}\left[\,\mathrm{C}^{\leq 4000}\,\right]$. As expected, the minimum cost reduces as the buffer size increases as sensors can now delay sending for longer without incurring a penalty. The relationship is non-linear and we can see that increasing the buffer from 2 to 3 reading is much more beneficial than from 3 to 4.

Second, consider cost reduction by altering the probability that a sensor receives data when not sending – essentially reducing the sampling rate. Figure 17b shows how changing $w_{receive}$ affects the minimum possible cost for a fixed buffer size. As expected, increasing the likelihood of receiving data increases overall costs, as the buffers are likely to fill quickly and readings may be missed.

5.3.3 Arbitrary sized buffers. As with the virus model (Section 5.1), a key benefit of bigraphs is the potential to try out extensions to the model. For example, we could replace the movement system to have the sink move through a dynamic topology (rather than the sensors moving relative



(a) Effect of increasing max buffer size on cost when $w_{receive} = 6$.



(b) Effect of increasing $w_{receive}$ on cost when Buffer size = 4.

Fig. 17. Effects of buffer size and data receive weight on cost.

to the sink). As we have shown topologies previously, we instead focus on a model extension that allows buffers of arbitrary size, different *types* of data within the buffer, and data-dependent send costs.

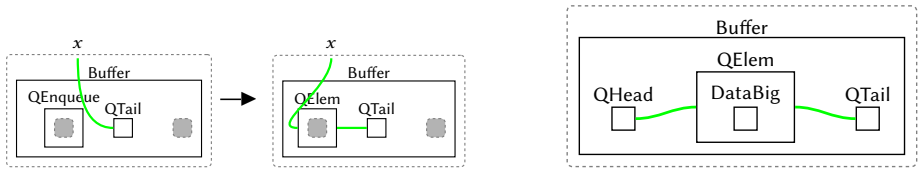
The main extension is the introduction of buffers as a first-in-first-out queue, *i.e.* we send the data in the order we receive it. Data is sent one element at a time, rather than emptying the buffer fully as before[7]. For simplicity, we assume two types of data: *big*, that costs more to send but is received less often, and *small*.

We show a basic bigraphical queue in Figure 18. The queue is modelled using bigraph links (as pointers) between queue elements. Note that queues are parameteried, through the use of sites. Additionally, for the enqueue rule, we make use of BigraphER's instantanous rules feature that means these rewrites do not show up in the final transition system. This feature is helpful for data structure operations that are not pertinent to analysis.

Within the main model, we replace the fixed sized buffers with queues and add new rules such as $A_{\texttt{receive}}$ for receiving data of different sizes. Likewise, we add/modify the send rules so they match explicitly on the head of the buffer queue. This results in new actions, *e.g.* $A_{\texttt{send\_mid\_small}}$. These new actions allow us to vary the reward attached to them based on data size. Key weights relating to data size are $w_{receive\_big}$, $w_{receive\_small}$, and $w_{receive\_fail}$. To illustrate the effects of varying $w_{receive\_big}$, Figure 19 shows the effects for a buffer size of 2 elements, $w_{receive\_small} = 10$ and $w_{receive\_fail} = 1$. As expected, increasing the likelihood of receiving big data increases the expected minimal cost (within 1000 steps).

Simillar to the virus extension, our main point is not the actual results, but to illustrate the ways in which bigraphs offer advantages over existing formalisms. Specifically, applications that use data structures are difficult to model in *e.g.* PRISM, which only supports a few types (integer, boolean, float). Likewise, it is difficult to write models that require control-flow constructs, *e.g.* loops or recursion (a key motivator behind the Modest modelling language [BHH21]). In comparison, Bigraphs allow these to be encoded easily using entities, placement and linking[8].

[7]It is possible to model sending all data at once, in which case we could use multiple counters inside the buffer instead.
[8]We intend to develop a set of basic data structures as a standard library in BigraphER.



(a) Reaction rule to enqueue an element. (b) Example buffer with one element.

Fig. 18. Modelling a Queue in bigraphs.

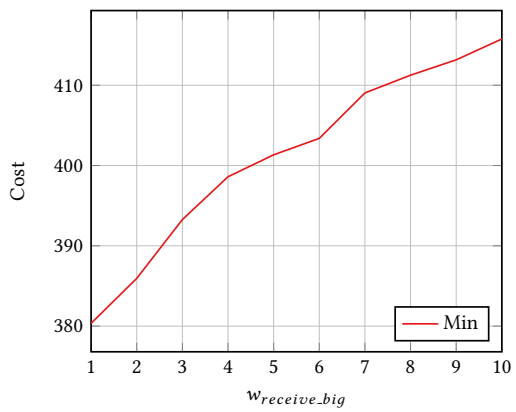


Fig. 19. Effect on minimal expected cost (in 1000 steps) as we increase the weight/probability that big data is received. Weight for small data is fixed at 10.

6 DISCUSSION

6.1 Probabilistic Structure

Our approach is to generate probabilistic *transition systems* from a set of probabilistic reaction rules. An alternative approach is to keep the rewrite semantics unchanged and instead allow the bigraphs themselves to be probabilistic.

For example, given a non-ground bigraph, we could initialise it (probabilistically) with different parameters to simulate probabilistic phenomena andor lack of information. This is the approach of term rewriting systems such as PMaude [AMS06], where rewriting can affect which variables are added in the substitution (the equivalent would be the instantiation of bigraph sites). This scenario can be modelled in probabilistic bigraphs as a set of rules, one for each possible right-hand side – where the set of rules act like a probability distribution.

Another approach, due to Syropoulos [Syr20], is to define the structure of bigraphs (*i.e.* the place and link graphs) in terms of fuzzy sets [Zad65]. An advantage of this formalisation is to allow a succinct representation of families of bigraphs. However, it is still an open question how to define bigraphs dynamics, *i.e.* matching and rewriting, in this setting. This makes it difficult to compare it directly with our approach.

6.2 Weights vs. Exact Probabilities

Our approach to rewriting is based around adding relative *weights* ($w \in \mathcal{R}_{\geq 0}$) to rules that determines how likely a rule is to be applied relative to all other rules that can be applied for a given state, *e.g.* a rule with weight 10 should be 10 times as likely to apply as a rule with weight 1. This

Listing 4. Specifying exact probabilities through actions

```
ctrl A = 0;
ctrl B = 0;
atomic control S1 = 0;
atomic control S2 = 0;
atomic control S3 = 0;

# Specifying exact probabilities of state changes for A
float p_s1_s1 = 0.2;
float p_s1_s2 = 0.6;
float p_s1_s3 = 0.2;

react a_s1_s1 = A.S1 -[p_a_s1_s1]-> A.S1;
react a_s1_s2 = A.S1 -[p_a_s1_s2]-> A.S2;
react a_s1_s3 = A.S1 -[p_a_s1_s3]-> A.S3;

# Exact probabilities for B
float p_b_s1_s1 = 0.3;
float p_b_s1_s2 = 0.7;

react b_s1_s1 = B.S1 -[p_b_s1_s1]-> B.S1;
react b_s1_s2 = B.S1 -[p_b_s1_s2]-> B.S2;

begin abrs
  ...
  actions = [
    # Action[cost] = { rules }
    A_change = {a_s1_s1, a_s1_s2, a_s1_s3},
    B_change = {b_s1_s1, b_s1_s2},
  ];
end
```

approach differs from that of traditional probabilistic modelling languages like PRISM that allow *exact* probabilities to be specified as part of the guarded commands, *e.g.* $p_1 = 0.3$ move to state 1, $p_2 = 0.7$ move to state 2.

The main reason for this difference is in modelling style. In PRISM a system is described as a collection of modules that are then *combined* (composed) to form a single state vector representing the entire system. At this point the probabilities of each module are normalised (with respect to any synchronisation actions) to determine the exact probability the system changes state. In bigraphs, the lack of structured modules means we *only* have the full system as a state, and updates are specified through *decomposition* into smaller matches where the probabilities can then be determined.

It is not immediately clear that one approach is superior to the other. We have found it useful to use weights to answer what-if style questions: "what if the topology changes more often relative to the virus movement?" without necessarily focusing on exact quantitative details.

6.2.1 Using Exact Probabilities in Bigraphs. Although in general it is not possible to assign exact probabilities to system transitions, we can allow limited exact probabilities by using action bigraphs in place of probabilistic bigraphs. In this case the outcome is still a probabilistic system and actions are internal actions, not those you find in the actual domain. This approach is used to define probabilistic graph transformation systems [KG12]. An example showing how to model with exact probabilities is in Listing 4.

The main idea here is to write actions that are constrained such that:

(1) All reaction rules in the action share the same left-hand-side.
(2) The weights assigned to the reaction rules in an action sum to 1 (allowed as weights are any positive real).

For situations where there are, *e.g.* more than one A entity in the system, an additional unique id is needed to ensure distinct matches.

Now, instead of determining all applicable rules in the system in order to normalise, we choose *an* applicable action (non-deterministically) and then within that action no normalisation takes place as the probabilities already sum to 1. This gives an exact probability a specific transition is made. One way to think of this is using actions to decompose the system back into module-like structures. Although specifying exact probabilities is possible, it is much harder to allow PRISM-like module synchronisation without adding additional structure to the bigraph. A version of this approach could be used in the WSN model of Figure 16 to control exact probabilities for $A_{\texttt{receive}}$ if we extend the match for sensor ids (to ensure only one sensor is receiving at a time).

6.3 Performance

In order to take advantage of existing model checkers such as PRISM, we support explicit transition system export. The complexity of this export depends on the system being analysed, and, as with all rewriting, there are no general guarantees the system terminates. Analysis of (some) systems exhibiting infinite behaviours is possible as BigraphER as it merges isomorphic states (up-to support equivalence), *e.g.* as shown in Figure 6, allowing existing states to be revisited.

To show the practical performance of BigraphER for generating probabilistic systems we perform an empirical analysis using the models presented in Section 5. The experimental system consists of an Intel i7-1075H CPU (2.6GHz) and 16GB of RAM, running Linux 5.10.81. We use BigraphER 1.3.4 that features a new matching engine based on an efficient subgraph-isomorphism solver [ABMS21]. Verification uses PRISM version 4.4. We use the model build time (time to generate all states) as reported by BigraphER. To account for random effects in the runtimes, we take an average build time for models that result in the same number of states/transitions, *i.e.* those that differ only in weights.

Table 3 shows the time to generate the transition systems for the models given in Section 5. In all cases BigraphER takes no more than 30s to build the full transition systems, making it a practical tool for model development tasks. The number of states is not directly related to the model build time, this is due to the need to check for isomorphic states (for merging), and this can be particularly costly for highly symmetric models like that in Section 5.1. We expect symmetry breaking techniques [MC06] can substantially improve this.

Once the transition system is generated, verification/analysis of properties is fast, taking less than 0.1s (per property) for virus spread and WSN, and less than 1s per property for Budding. As budding analyses 50 properties (for the different particles contained in the bud; Figure 14) it takes longer to analyse than to build the model. As with all models of this form, we should not try to predict scalability based on previous results as even small changes, *e.g.* the dynamic topology of Section 5.1.4, can exponentially increase the number of states. This is a general problem, not specific to bigraphs.

Overall, with bigraphs there may be trade-offs between expressability/readability and performance, depending on how the modeller employs entities, connections and rules. For example, digrammatic representation may be more important than tractability of exploring the full state space. PRISM models are often more efficient to analyse as they can perform techniques like symbolic analysis and bisimulation reductions etc. on the highly structured (modules/variables/actions) input model. It remains open, and a promising area for future work, if similar techniques can apply to bigraphs models, especially those that reduce the state space.

Table 3. Performance of model generating. Runtimes as reported by BigraphER. States and transitions are those of the final transition system, while occurrences is the total number of rule matches found (as a measure of the total amount of work done). When transitions are less than occurrences then states have been merged due to equivalence checking.

Instance	Build time (s)	States	Transitions	Occurrences
Virus Single Firewall (Section 5.1)	24.78	809	3972	4694
Virus Multi Firewall (Section 5.1.3)	28.84	809	3972	4694
Budding (Section 5.2)	2.86	3570	9486	9486
WSN (Section 5.3)	10.89	1830	4427	9401
WSN Queue (Section 5.3.3)	120.22	4620	12156	30132

6.4 Comparing Probabilistic/Stochastic Bigraphs and PRISM models

Throughout we have compared bigraphs with the PRISM modelling language. We summarise three main observations here[9].

First, the rules based approach allows easy extension of models, such as dynamic topologies (Section 5.1.4) and addition of non-primitive data types (Section 5.3.3). This contrasts with PRISM where the rigid module structure makes it more difficult to model data types and to modify individual processes behaviour, the number of processes, and/or their communication topology.

Second, probabilistic bigraphs employ weights instead (Section 6.2) of *exact* probabilities that are specified in PRISM modules. In the former, probabilities are assigned to pairs of states of the entire system, which requires the additional normalisation procedure to account for all possible (global) transitions. However, PRISM also includes a normalisation procedure to allow multiple modules to be used in parallel, *i.e.* it combines states into a tuple and normalises probabilities [PRI]. This demonstrates the fundamental differences in approach: PRISM takes many smaller parts and combines them to get the total system, whereas probabilistic bigraphs take the total system and splits it into smaller parts. In both cases, normalisation is needed to move between the system and smaller parts.

Finally, there are tradeoffs between bigraphs expressability and performance, and PRISM models are often more efficient to analyse than those constructed from an explicit transition system. The differences depend on the nature of the system being modelled and its representation (*e.g.* use of entities, links etc.). In our experience, as Section 6.3 illustrates, the bigraphs approach is usually efficient enough for practical use.

6.5 Future Work

Additional Probabilistic Process Types. We currently support DTMC, CTMC, and MDP models, however many other probabilistic models are used in practice. For example MDPs have been extended in multiple ways, *e.g.* partially observable MDPs (POMDPs) [Åst65], and these are candidates for further BRS extensions.

In POMDPs, agents cannot directly observe the current state and are instead assigned a set of observations/beliefs allowing decision making in uncertain environments. One possibility for BRSs is to fix the *actual* and *observed* states into two (bigraph) regions, where cross-region links connect the observations to the states they observe. Unfortunately while PRISM allows POMDPs to be specified, there is currently no way to import partially observable models.

[9]The arguments are equally applicable to model checkers such as Storm [DJKV17b].

Another extension of MDPs is Markov automata (MA) [EHZ10, DH13], combining probabilistic branching, non-determinism, *and* exponentially distributed delays. In a bigraph model, the probabilistic states of a MA would be treated as the states of an ABRS, while the Markovian states would behave like the states of a SBRS. However, rewriting for *hybrid states*, *i.e.* states where both non-deterministic choice over probability distributions and a distribution over states are available, requires a new definition.

Bisimulations. Exploring the RPO framework in the presence of probabilistic rules likewise remains an open problem. It is likely RPOs exist for probabilistic rules, however these will have to also account for the normalised probabilities of specific matches (to ensure the contexts are equal).

7 CONCLUSION

Bigraphical reactive systems (BRSs) have proved invaluable for modelling a wide range of systems, both virtual and physical, but are limited in the types of system they can represent. Real-world systems are often probabilistic, stochastic, and feature non-deterministic decisions, and these do not fit within standard BRSs.

We have shown how, by assigning *weights* to standard BRS reaction rules we can model probabilistic systems (probabilistic BRSs – PBRSs), and extend this to support systems that make explicit decision (action) choices (action BRSs – ABRSs).

We have implemented both PBRS and ABRS in BigraphER, an open-source toolkit for working with bigraphs. To support stochastic systems we also implement *stochastic bigraphs* as defined by Krivine *et al.* [KMT08]. We show the new extensions are practical through a set of case study models: virus spread in computer networks, membrane budding in biological systems, and data harvesting in wireless sensor networks. Probabilistic bigraphs have further been applied to model autonomous BDI-agent semantics [ACSX21].

In conclusion, we have successfully extended the capabilities of BRSs and the toolkit BigraphER to model a wider range of systems, whilst preserving the high level nature and flexibility of the bigraph formalism.

ACKNOWLEDGMENTS

This work is supported by the Engineering and Physical Sciences Research Council, under grant EP/N007565/1 S4: Science of Sensor Systems Software, and by PETRAS SRF grant EP/S035362/1 MAGIC. We thank Paulius Dilkas for his work on an early implementation of these ideas.

REFERENCES

[ABMS21] Blair Archibald, Kyle Burns, Ciaran McCreesh, and Michele Sevegnani. Practical bigraphs via subgraph isomorphism. In Laurent D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, pages 15:1–15:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[ACS20] Blair Archibald, Muffy Calder, and Michele Sevegnani. Conditional bigraphs. In Fabio Gadducci and Timo Kehrer, editors, *Graph Transformation - 13th International Conference, ICGT 2020, Held as Part of STAF 2020, Bergen, Norway, June 25-26, 2020, Proceedings*, volume 12150 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2020.

[ACSX21] Blair Archibald, Muffy Calder, Michele Sevegnani, and Mengwei Xu. Probabilistic BDI agents: Actions, plans, and intentions. In Radu Calinescu and Corina S. Pasareanu, editors, *Software Engineering and Formal Methods - 19th International Conference, SEFM 2021, Virtual Event, December 6-10, 2021, Proceedings*, volume 13085 of *Lecture Notes in Computer Science*, pages 262–281. Springer, 2021.

[AMS06] Gul A. Agha, José Meseguer, and Koushik Sen. PMaude: Rewrite-based specification language for probabilistic object systems. *Electr. Notes Theor. Comput. Sci.*, 153(2):213–239, 2006.

[ASC] Blair Archibald, Michele Sevegnani, and Muffy Calder. Probabilistic bigraphs: Models. https://doi.org/10.5281/zenodo.3971680.

[Åst65] Karl J Åström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[BCRS16] Steve Benford, Muffy Calder, Tom Rodden, and Michele Sevegnani. On lions, impala, and bigraphs: Modelling interactions in physical/virtual spaces. *ACM Trans. Comput.-Hum. Interact.*, 23(2):9:1–9:56, 2016.

[Bel57] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[BH03] Olivier Bournez and Mathieu Hoyrup. Rewriting logic and probabilities. In *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Valencia, Spain, June 9-11, 2003, Proceedings*, pages 61–75, 2003.

[BHH21] Yuliya Butkova, Arnd Hartmanns, and Holger Hermanns. A modest approach to markov automata. *ACM Trans. Model. Comput. Simul.*, 31(3), aug 2021.

[Bil12] P. Billingsley. *Probability and Measure*. Wiley Series in Probability and Statistics. Wiley, 2012.

[BT08] Ladislau Bölöni and Damla Turgut. Should I send now or send later? A decision-theoretic approach to transmission scheduling in sensor networks with mobile sinks. *Wireless Communications and Mobile Computing*, 8(3):385–403, 2008.

[CKSS14] Muffy Calder, Alexandros Koliousis, Michele Sevegnani, and Joseph S. Sventek. Real-time verification of wireless home networks using bigraphs with sharing. *Sci. Comput. Program.*, 80:288–310, 2014.

[CS14] Muffy Calder and Michele Sevegnani. Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing. *Formal Asp. Comput.*, 26(3):537–561, 2014.

[DH13] Yuxin Deng and Matthew Hennessy. On the semantics of markov automata. *Information and Computation*, 222:139 – 168, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[DJKV17a] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, pages 592–600, 2017.

[DJKV17b] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. *Proc. of CAV, LNCS*, 10427:592fi?!–600, 2017.

[EHZ10] C. Eisentraut, H. Hermanns, and L. Zhang. On probabilistic automata in continuous time. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*, pages 342–351, 2010.

[Het00] Herbert W. Hethcote. The mathematics of infectious diseases. *Siam Review*, 42(4):599–653, 2000.

[How60] R.A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

[KG12] Christian Krause and Holger Giese. Probabilistic graph transformation systems. In *Graph Transformations - 6th International Conference, ICGT 2012, Bremen, Germany, September 24-29, 2012. Proceedings*, pages 311–325, 2012.

[KMT08] Jean Krivine, Robin Milner, and Angelo Troina. Stochastic bigraphs. *Electr. Notes Theor. Comput. Sci.*, 218:73–96, 2008.

[KNP10] M. Kwiatkowska, G. Norman, and D. Parker. Advances and challenges of probabilistic model checking. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1691–1698, Sep. 2010.

[KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[KNPV09] M. Kwiatkowska, G. Norman, D. Parker, and M.G. Vigliotti. Probabilistic mobile ambients. *Theoretical Computer Science*, 410(12–13):1272–1303, 2009.

[MC06] A. Miller, A. amd Donaldson and M. Calder. Symmetry in temporal model checking. *ACM Surveys*, 38:1–36, 2006.

[Mil09] Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[PRI] PRISM. PRISM manual – parallel composition. http://www.prismmodelchecker.org/manual/ThePRISMLanguage/ParallelComposition. Accessed 4/8/2020.

[SC15] Michele Sevegnani and Muffy Calder. Bigraphs with sharing. *Theor. Comput. Sci.*, 577:43–73, 2015.

[SC16] Michele Sevegnani and Muffy Calder. BigraphER: Rewriting and analysis engine for bigraphs. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, volume 9780 of *Lecture Notes in Computer Science*, pages 494–501. Springer, 2016.

[Syr20] Apostolos Syropoulos. Fuzzy bigraphs. In *Algebraic Techniques and Their Use in Describing and Processing Uncertainty*, pages 157–170. Springer, 2020.

[TPGN18] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. On the interplay between cyber and physical spaces for adaptive security. *IEEE Transactions on Dependable and Secure Computing*, 15(3):466–480,

May 2018.

[WW12] Lisa Walton and Michael F. Worboys. A qualitative bigraph model for indoor space. In Ningchuan Xiao, Mei-Po Kwan, Michael F. Goodchild, and Shashi Shekhar, editors, *Geographic Information Science - 7th International Conference, GIScience 2012, Columbus, OH, USA, September 18-21, 2012. Proceedings*, volume 7478 of *Lecture Notes in Computer Science*, pages 226–240. Springer, 2012.

[Zad65] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.