Nondeterministic Bigraphical Reactive Systems for Markov Decision Processes*

Paulius Dilkas

University of Glasgow, Glasgow, UK

Abstract. In this paper we introduce two extensions of bigraphical reactive systems (BRSs) to handle probabilistic and nondeterministic behaviour, allowing BRSs to be used in discrete-time Markov chain and Markov decision process (MDP) generation. We extend the implementation of an open-source tool for BRSs BigraphER with new syntax for MDP actions and rewards, better visualisation capabilities, and data export options. Furthermore, we introduce a new front-end for working with BRSs based on Jupyter notebooks, allowing the user to quickly visualise bigraphs and reaction rules, simulate execution, or produce a complete transition system, all within the notebook interface. Finally, we demonstrate the modelling capabilities of the new system with a case study on the movement and behaviour of autonomous agents, exploring ideas such as collecting objects, tracking visited locations, hierarchies of spaces, and uncertainty in the topology of space.

Keywords: Bigraphs · Bigraphical reactive systems · Probabilistic semantics · Markov decision process · Spatial models.

1 Introduction

1.1 Markov Decision Processes

Definition 1 (Markov decision process). *For any finite set X, let $Dist(X)$ denote the set of discrete probability distributions over X. A* Markov Decision Process *is a tuple $(S, \overline{s}, A, P, L)$, where: S is a finite set of states and $\overline{s} \in S$ is the initial state; A is a finite set of* actions*; $P : S \times A \to Dist(S)$ is a (partial) probabilistic transition function, mapping state-action pairs to probability distributions over S; $L : S \to 2^P$ is a labelling with atomic propositions.*

Definition 2 (reward structure). *A* reward structure *for an MDP $(S, \overline{s}, A, P, L)$ is a pair (ρ, ι), where $\rho : S \to \mathbb{R}$ is the* state reward function*, and $\iota : S \times A \to \mathbb{R}$ is the* transition reward function*.*

* Supported by organization x.

1.2 Bigraphs

We begin by introducing some notation used in the definitions. Natural numbers are sometimes interpreted as finite ordinals, i.e., $m = \{0, \ldots, m-1\}$. We write $S \uplus T$ to indicate the union of sets known or assumed to be disjoint. For two functions f and g with disjoint domains S and T we write $f \uplus g$ for the function with domain $S \uplus T$ such that $(f \uplus g)|_S = f$ and $(f \uplus g)|_T = g$. The identity function on the set S is written as Id_S. Given a binary relation $rel \subseteq A \times B$, we denote the domain restriction of rel over $S \subseteq A$ by $S \lhd rel$. Similarly, we write $rel \rhd S$ for the range restriction of rel over $S \subseteq B$. We also denote the transitive closure of rel by rel^+. In defining bigraphs we assume that names, node identifiers, edge identifiers, and controls are drawn from four infinite pairwise disjoint sets, respectively $\mathcal{X}, \mathcal{V}, \mathcal{E}$, and $\mathcal{K}$.

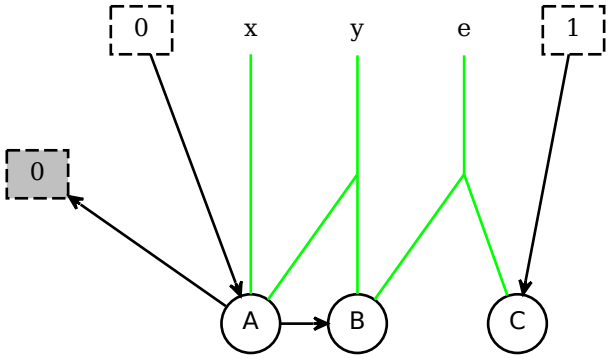

Fig. 1. An example bigraph.

Informally, a bigraph can be thought of as a graph, where a node can be inside another node[1]. However, the pictures tend to become much clearer if we instead visualise the nesting relations between nodes as just a different type of edges. Consider the bigraph in Fig. 1. The white ellipses are the *nodes*. Each node has a type, called *control*, and denoted here by the labels A, B, and C. The green edges are actually hyperedges (can connect any positive number of nodes), and are called *links*. In this example they are given names x, y, and e. Each node has an ordered list of *ports* (not pictured in the diagram) that can be thought of as sockets for links. Nodes of the same control have the same number of ports. The number of ports a node of control K can have is called the *arity* of K and is denoted by $ar(\mathsf{K})$. The two white dashed rectangles represent *regions*, and in this representation are the parents to any otherwise parentless nodes. Grey dashed rectangles are called *sites*, and encode parts of the model that have been

[1] To be specific, in this paper we extend the ideas behind *bigraphs with sharing* [9,10], where a node can be contained in the intersection of several other nodes.

abstracted away. Nodes, sites, and regions are also known as *places*. The directed black arrows between places show the containment relations, e.g., the A node has two children (inner nodes): B and the only site in this bigraph, labelled by the number 0.

Another, more formal way to think about a bigraph is as a pair of graphs: one for describing the placement of nodes, and one for the links between them. This is the approach that will be featured in our formal definitions. We also focus on *concrete* bigraphs, where nodes and links have distinct identifiers.

Definition 3 (concrete place graph with sharing). *A* concrete place graph with sharing

$$F = (V_F, ctrl_F, prnt_F) : m \to n$$

is a triple having an inner interface m and an outer interface n. These index the sites and regions of the place graph, respectively. F has a finite set $V_F \subset \mathcal{V}$ of nodes, a control map $ctrl_F : V_F \to \mathcal{K}$, and a parent relation

$$prnt_F \subseteq (m \uplus V_F) \times (V_F \uplus n)$$

that is acyclic, i.e., $(v, v) \notin prnt_F^+$ for any $v \in V_F$.

We define several operations on place graphs that will be used later.

Definition 4 (composition for place graphs with sharing). *If $F : k \to m$ and $G : m \to n$ are two concrete place graphs with sharing with $V_F \cap V_G = \emptyset$, their composite*

$$G \circ F = (V, ctrl, prnt) : k \to n$$

has nodes $V \coloneqq V_F \uplus V_G$ and a control map $ctrl \coloneqq ctrl_F \uplus ctrl_G$. Its parent relation $prnt \subseteq (k \uplus V) \times (V \uplus n)$ is given by:

$$prnt \coloneqq prnt_G^{\triangleleft} \uplus prnt_{\circ} \uplus prnt_F^{\triangleright},$$

where

$$\begin{aligned} prnt_F^{\triangleright} &= prnt_F \rhd V_F, \\ prnt_G^{\triangleleft} &= V_G \lhd prnt_G, \\ prnt_{\circ} &= (m \lhd prnt_G) \circ (prnt_F \rhd m). \end{aligned}$$

Definition 5 (tensor product for place graphs). *If*

$$G_0 = (V_{G_0}, ctrl_{G_0}, prnt_{G_0}) : m_0 \to n_0$$

and

$$G_1 = (V_{G_1}, ctrl_{G_1}, prnt_{G_1}) : m_1 \to n_1$$

are two concrete place graphs with sharing with $V_{G_0} \cap V_{G_1} = \emptyset$, their tensor product

$$G_0 \otimes G_1 = (V, ctrl, prnt) : m_0 + m_1 \to n_0 + n_1$$

has nodes $V \coloneqq V_{G_0} \uplus V_{G_1}$ *and a control map* $ctrl \coloneqq ctrl_{G_0} \uplus ctrl_{G_1}$*. Its parent relation* $prnt \subseteq [(m_0 + m_1) \uplus V] \times [V \uplus (n_0 + n_1)]$ *is defined as*

$$prnt_{G_0} \uplus prnt_{G_1}^{(m_0,n_0)},$$

where

$$\begin{aligned} prnt_{G_1}^{(m_0,n_0)} =& \{(v, w) \mid (v, w) \in prnt_{G_1} \text{ and } v, w \in V_{G_1}\} \\ &\uplus \{(m_0 + i, w) \mid (i, w) \in prnt_{G_1}, w \in V_{G_1}, \text{ and } i \in m_1\} \\ &\uplus \{(v, n_0 + i) \mid (v, i) \in prnt_{G_1}, v \in V_{G_1}, \text{ and } i \in n_1\} \\ &\uplus \{(m_0 + i, n_0 + j) \mid (i, j) \in prnt_{G_1}, i \in m_1, \text{ and } j \in n_1\}. \end{aligned}$$

Definition 6 (concrete link graph). *A* concrete link graph

$$F = (V_F, E_F, ctrl_F, link_F) : X \to Y$$

is a quadruple having an inner face X *and an outer face* Y*, both finite subsets of* $\mathcal{X}$*, called respectively the* inner *and* outer names *of the link graph.* F *has finite sets* $V_F \subset \mathcal{V}$ *of nodes and* $E_F \subset \mathcal{E}$ *of edges, a control map* $ctrl_F : V_F \to \mathcal{K}$ *and a* link map

$$link_F : X \uplus P_F \to E_F \uplus Y,$$

where $P_F \coloneqq \{(v, i) \mid i \in ar(ctrl_F(v))\}$ *is the set of ports of* F*. Thus,* (v, i) *is the* i*th port of node* v*. We shall call* $X \uplus P_F$ *the* points *of* F*, and* $E_F \uplus Y$ *its* links*.*

Two places with the same parent, or two points with the same link are called *siblings*. A link is *idle* if it has no points. Identities over (inner or outer) name sets are defined by $\mathsf{id}_X = (\emptyset, \emptyset, \emptyset, \mathsf{Id}_X) : X \to X$.

Definition 7 (concrete bigraph with sharing). *A concrete bigraph*

$$F = (V_F, E_F, ctrl_F, prnt_F, link_F) : \langle k, X\rangle \to \langle m, Y\rangle$$

consists of a concrete place graph with sharing $F^P = (V_F, ctrl_F, prnt_F) : k \to m$ *and a concrete link graph* $F^L = (V_F, E_F, ctrl_F, link_F) : X \to Y$*. We write the concrete bigraph with sharing as* $F = (F^P, F^L)$*.*

We use ϵ as a shorthand for interface $\langle 0, \emptyset\rangle$. A bigraph with inner face ϵ is called *ground*.

Definition 8 (solid bigraph). *A bigraph is* solid *if these conditions hold:*

1. *no regions or outer names are idle (a region is called* idle *if it is empty);*
2. *no two sites or inner names are siblings;*
3. *the parent of every site is a node;*
4. *no outer name is linked to an inner name.*

Definition 9 (reaction rule). *A* reaction rule *is a pair*

$$\mathsf{R} = (R : m \to J, R' : m \to J),$$

sometimes written as $R \longrightarrow\!\!\blacktriangleright R'$*, where* R *is the* redex *and* R' *the* reactum*, and* R *is solid. The rule generates all the* ground reaction rules (r, r')*, where* $r = (R \otimes \mathsf{id}_Y) \circ d$ *and* $r' = (R' \otimes \mathsf{id}_Y) \circ d$ *for some discrete ground* parameter $d : \epsilon \to \langle m, Y \rangle$*. The* reaction relation $\longrightarrow\!\!\triangleright_{\mathsf{R}}$ *over ground bigraphs is defined by*

$$g \longrightarrow\!\!\triangleright_{\mathsf{R}} g' \text{ iff } g = D \circ r \text{ and } g' = D \circ r'$$

for some bigraph D *and some ground reaction rule* (r, r') *generated from* R*.*

Definition 10 (bigraphical reactive system (BRS)). *A* bigraphical reactive system *consists of 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 a 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.*

Given a set of reaction rules, we refer to the configurations that a system may adopt as *states*. *Rule priorities* [1] impose a partial ordering on the reaction rules, where a rule of lower priority can be applied only if no rule of higher priority is applicable. Priorities are implemented by assigning each reaction rule to a *priority class*, where all classes are strictly ordered, and reaction rules in the same class have the same priority.

Predicates can be used to generate labels for states (as in Definition 1) or to ease understanding and debugging of the system. They are expressed in a subset of a logic called BiLog [3], allowing any predicate to be encoded as a bigraph [2,9]. Therefore, we can check each state for any satisfied predicates by solving a bigraph matching problem.

2 Extensions to BRSs

We propose two new BRSs: one adds probabilities to reaction rules, the other also groups the rules into actions and supports rewards based on both reaction rules and predicates. The main technicality in both the definitions and the implementation is that after generating a list of applicable ground reaction rules (for a particular action, if appropriate), the probabilities need to be normalised so that they add to 1, as a single reaction rule can sometimes be applied in multiple places, or some rules in an action may not be applicable.

2.1 Probabilistic BRS

Definition 11 (probabilistic reaction rule). *A* probabilistic reaction rule R *is a triple* (R, R', p)*, sometimes written* $R \xrightarrow{p}\!\!\blacktriangleright R'$*, where* (R, R') *is a reaction rule and* $p \in (0, 1]$ *is a probability. Similarly to Definition 9, it generates a set of ground reaction rules of the form* (r, r', p)*.*

Definition 12 (probabilistic bigraphical reactive system (PBRS)). *A* probabilistic bigraphical reactive system *consists of a pair $(\mathcal{B}, \mathcal{R})$, where $\mathcal{B}$ is a set of ground bigraphs and $\mathcal{R}$ is a set of probabilistic reaction rules defined over $\mathcal{B}$.*

Let g, g' be ground bigraphs, and $\{(r_i, r_i', p_i)\}_{i=1}^n$ a set of ground probabilistic reaction rules, where for each r_i, there exists a bigraph D_i such that $g = D_i \circ r_i$. Let $S = \{(r_i, r_i', p_i) \mid g' = D_i \circ r_i'\}$ (for the same D_i), and

$$s = \sum_{i=1}^{n} p_i.$$

Then the reaction relation is defined as

$$g \xrightarrow{p}\!\triangleright_{\mathcal{R}} g' \text{ iff } S \neq \emptyset,$$

where

$$p = \frac{1}{s} \sum_{(r, r', p') \in S} p'.$$

2.2 Nondeterministic BRS

Definition 13 (nondeterministic reaction rule). *Let A be a set of actions. A* nondeterministic reaction rule R *is a tuple (R, R', a, p), where (R, R', p) is a probabilistic reaction rule, and $a \in A$ is an action. We also define a* reaction reward function $r : A \to \mathbb{R}$ *that assigns a reward or cost to each action.*

Definition 14 (nondeterministic bigraphical reactive system (NBRS)). *A* nondeterministic bigraphical reactive system *consists of a pair $(\mathcal{B}, \mathcal{R})$, where $\mathcal{B}$ is a set of ground bigraphs and $\mathcal{R}$ is a set of nondeterministic reaction rules defined over $\mathcal{B}$.*

Let g, g' be ground bigraphs, $a \in A$ an action, and $\{(r_i, r_i', a, p_i)\}_{i=1}^n$ a set of ground nondeterministic reaction rules with action a, where for each r_i, there exists a bigraph D_i such that $g = D_i \circ r_i$. Let $S = \{(r_i, r_i', a, p_i) \mid g' = D_i \circ r_i'\}$ (for the same D_i), and

$$s = \sum_{i=1}^{n} p_i.$$

Then the reaction relation for action a is defined as

$$g \xrightarrow[r(a)]{p}\!\triangleright_{a} g' \text{ iff } S \neq \emptyset,$$

where

$$p = \frac{1}{s} \sum_{(r, r', a, p') \in S} p'.$$

Finally, we can associate a reward with each predicate, allowing us to assign rewards to states in a flexible and semantically meaningful way: the reward of a state is simply the sum of the rewards of all matching predicates (and 0 in case there are none).

3 Implementation

The described extensions have been implemented in an open-source tool for BRSs BigraphER [11]. We refer the interested reader to the paper on BigraphER [11], the related PhD thesis [9], and online resources[2] for further information on the syntax and capabilities of the software. In this section we will introduce the new features of BigraphER by using a simple MDP as an example.

Example 1. Consider an MDP $(S, \overline{s}, A, P, L)$, where $S = \{s_0, s_1, s_2, s_3\}$, $\overline{s} = s_0$, $A = \{a, b, c\}$, and P, L are defined as follows:

$$\begin{aligned} P(s_0, a) &= [s_1 \mapsto 1], \\ P(s_1, b) &= [s_0 \mapsto 0.7, s_1 \mapsto 0.3], \\ P(s_1, c) &= [s_2 \mapsto 0.5, s_3 \mapsto 0.5], \\ P(s_2, a) &= [s_2 \mapsto 1], \\ P(s_3, a) &= [s_3 \mapsto 1], \end{aligned} \qquad \begin{aligned} L(s_0) &= \{initial\}, \\ L(s_1) &= \emptyset, \\ L(s_2) &= \{heads\}, \\ L(s_3) &= \{tails\}. \end{aligned}$$

Furthermore, equip it with a reward structure (ρ, ι), where $\rho(s_2) = 3$, $\iota(s_1, b) = 1$, and both functions are zero everywhere else.

The MDP can be represented as an NBRS with BigraphER code in Listing 1.1. More specifically, reaction rule probabilities are represented as floating-point numbers inside the arrows (e.g. `-[0.7]->`), each action encompasses its reaction rules with **`action`** `actionName` and **`end`**. Rewards can be added to actions simply by inserting an integer enclosed in squared brackets after the action name (e.g. **`action`** `b[1]`). Lastly, predicate rewards have the same format, but are listed on the **`preds`** line of the **`begin nbrs`**—**`end`** block (e.g. **`preds`** = `{heads[3]};`).

BigraphER can construct and visualise the full transition system (see Fig. 2). Each state is represented by a white ellipse (the starting state is highlighted with a thicker border), and each action (per state) is a grey ellipse. The labels inside state ellipses list all predicates that are satisfied by that state. Edges from actions to states are labelled with both the probability and the name of the reaction rule. Transitions from either a fully generated transition system or a simulation, labels from predicates, and state/transition rewards can be exported to the various plain text formats of a probabilistic model checker PRISM [8] for further quantitative analysis.

3.1 Front end

We introduce a new front end to BigraphER based on Jupyter notebooks [6]. The BigraphER kernel is an extension to the OCaml kernel[3] and is available on GitHub[4]. It supports a similar workflow to that of Jupyter notebooks for other

[2] http://www.dcs.gla.ac.uk/~michele/bigrapher.html
[3] https://github.com/akabe/ocaml-jupyter
[4] https://github.com/dilkas/bigrapher-jupyter

```
atomic ctrl S0 = 0;
atomic ctrl S1 = 0;
atomic ctrl S2 = 0;
atomic ctrl S3 = 0;

big initial = S0;
big heads = S2;
big tails = S3;

action a
  react toTemp = S0 -[1.]-> S1;
  react loopH = S2 -[1.]-> S2;
  react loopT = S3 -[1.]-> S3;
end

action b[1]
  react toInit = S1 -[0.7]-> S0
  react loop = S1 -[0.3]-> S1;
end

action c
  react toH = S1 -[0.5]-> S2;
  react toT = S1 -[0.5]-> S3;
end

begin nbrs
  init initial;
  rules = [ {toTemp, toInit,
    loop, toH, toT,
    loopH, loopT} ];
  preds = { initial, heads[3],
    tails };
end
```

Listing 1.1. BigraphER code.

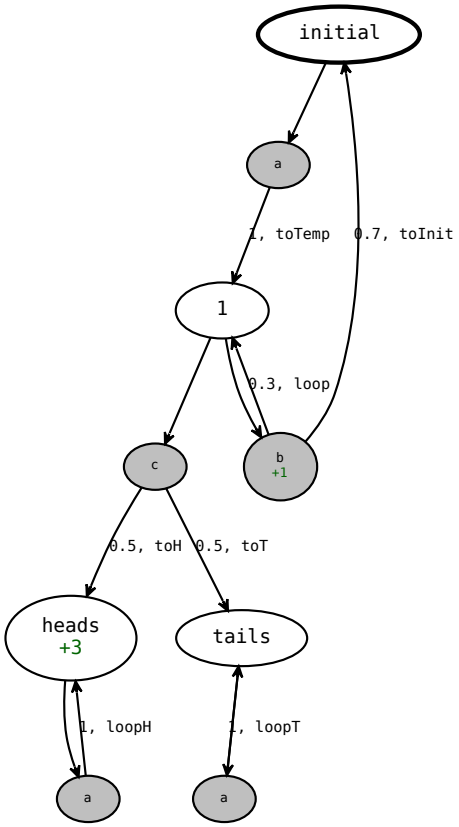

Fig. 2. The full transition system.

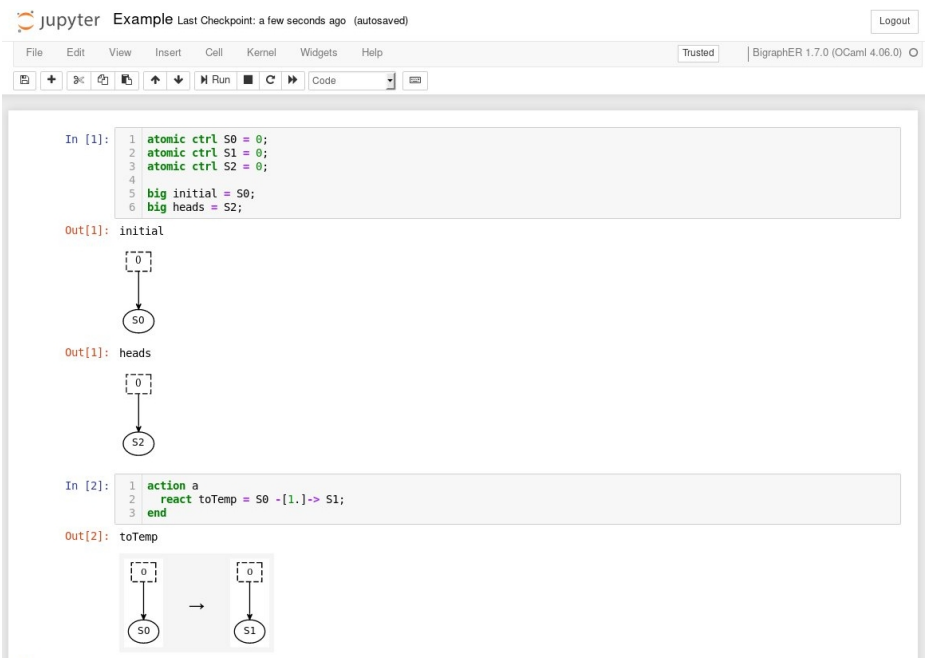

Fig. 3. The BigraphER Jupyter notebook interface with syntax highlighting.

languages, allowing the user to divide the code into multiple *cells* and have the definitions persist in a *buffer* to be used later. By default, the output of each cell displays the graphical representation of all bigraphs and reaction rules defined in that cell (see Fig. 3 for an example). The generated images are stored in a local directory `jupyter-images/`. The kernel also supports a number of *magics*, i.e., custom commands that can be used at the top of a cell:

`%states` produces a state transition diagram (similar to the one in Fig. 2). Mousing over a state shows what that state looks like as a bigraph.

`%simulate n` runs a simulation for `n` steps, when used on non-stochastic models. For stochastic BRSs [7], `n` is interpreted as a floating-point number, representing maximum simulation time. The produced diagram is a single path through the transition system with the same mouse-over functionality.

`%output` displays the full output produced by BigraphER. By default, only errors are displayed in the notebook.

`%ocaml` allows full backward compatibility with the OCaml kernel, including auto-complete and integrated documentation features that have been extended to the BigraphER OCaml API as well as plotting support with Archimedes[5].

`%clear` clears the buffer before interpreting the code from the current cell.

[5] http://archimedes.forge.ocamlcore.org/

4 A Case Study in Autonomous Agents

In order to demonstrate the use of bigraphs in modelling probabilistic processes with nondeterminism, we present two examples of agent movement planning in two dimensions. The examples have different sources of probabilistic behaviour and explore topics such as tracking visited locations and uncertainty about the topology.

4.1 Collecting Objects in a Grid

Our first scenario is concerned with collecting objects in a grid-like two-dimensional world and is inspired by similar scenarios modelled using PRISM [4,5]. The agent starts in the southwest corner of the map (which we call *home*), explores the environment until it collects a specified number of objects, and returns home (while possibly collecting more objects along the way). The model tracks which cells of the grid have been explored, and only the unexplored cells have a constant probability p of rewarding the agent with an object. The desired number of objects can be easily changed, and the map can be customised to make some transitions unavailable (i.e. add walls), as long as the agent can still return home.

```
ctrl Cell = 1;
ctrl Agent = 0;
ctrl Directions = 0;
ctrl Object = 0;

atomic ctrl North = 1;
atomic ctrl East = 1;
atomic ctrl West = 1;
atomic ctrl South = 1;

atomic ctrl Visited = 1;
atomic ctrl Unvisited = 1;
```

Listing 1.2. Controls.

We begin with a list of controls in Listing 1.2. On the right hand side of each equal sign is the number of ports that each node of that control has, while the **atomic** keyword specifies the nodes that cannot contain other nodes. `Cells` represents discrete locations. They are linked to either `Visited` or `Unvisited` (initially, all except one cells are unvisited). `Cells` always contain `Directions`, and one `Cell` contains an `Agent`. The `Agent` may contain an `Object`. In order to represent multiple objects, it is convenient to put the second `Object` inside the first, and so on. `Directions` contains a subset of the four nodes, corresponding to the four possible directions of travel: `North`, `East`, `West`, `South`. We connect neighbouring `Cells` by linking the `West` node of one `Cell` to the `East` node of another (or `North` to `South`).

```
big home = Cell{v}.(Directions.(North{v1}
                              | East{v2})
                  | Agent);
big goal = Agent.Object;
big initial = Visited{v}
           || Unvisited{u}
           # bottom left
           || Cell{v}.(Directions.(North{a} | East{b})
                     | Agent.1)
           # top left
           || Cell{u}.Directions.(East{c} | South{a})
           # bottom right
           || Cell{u}.Directions.(North{d} | West{b})
           # top right
           || Cell{u}.Directions.(West{c} | South{d});
```

Listing 1.3. Predicates and the initial state.

Next, we define two predicates: `home` and `goal` (see Listing 1.3). The former states that the agent is home if it is in the southwest corner of the grid, while the latter specifies a goal of collecting one object. A larger number of objects can be set by nesting `Objects` inside each other, e.g., `Agent.Object.Object`. Additionally, Listing 1.3 defines the initial 2×2 grid, as described previously.

```
action north
  react goNorthToVisited =
        Visited{v}
    || Cell{v}.(Directions.(North{b} | id) | Agent)
    || Cell{v}.Directions.(South{b} | id)
    -[1.0]->
        Visited{v}
    || Cell{v}.Directions.(North{b} | id)
    || Cell{v}.(Directions.(South{b} | id) | Agent)
      @ [0, 2, 1];
  react goNorthToUnvisited =
        Visited{v} || Unvisited{u}
    || Cell{v}.(Directions.(North{b} | id) | Agent)
    || Cell{u}.Directions.(South{b} | id)
    -[1.0-p]->
        Visited{v} || Unvisited{u}
    || Cell{v}.Directions.(North{b} | id)
    || Cell{v}.(Directions.(South{b} | id) | Agent)
      @ [0, 2, 1];
  react northObject =
        Visited{v} || Unvisited{u}
    || Cell{v}.(Directions.(North{b} | id) | Agent)
    || Cell{u}.Directions.(South{b} | id)
```

```
    -[p]->
       Visited{v} || Unvisited{u}
    || Cell{v}.Directions.(North{b} | id)
    || Cell{v}.(Directions.(South{b} | id)
              | Agent.Object)
      @ [0, 2, 1];
end
```

Listing 1.4. The action of going north.

Then, we have an action for each of the four directions, each with three reaction rules: one for going to an already visited cell, one for visiting a cell for the first time, and one for visiting a cell for the first time and finding an object. We will use the rules for going north in Listing 1.4 as an example. When going to an already visited cell, the `Agent` simply traverses a link between the `North` node of its previous `Cell` and the corresponding `South` node. As this is the only reaction rule for this situation, its probability is 1. When going to an unvisited node, we collect an `Object` with probability p, or make the transition without getting an `Object` with probability $1 - p$. Regardless, the `Cell` changes its link from `Unvisited` to `Visited`.

```
react stayHome =
  Cell{v}.(Directions.(North{v1} | East{v2}) | goal)
  -[1.0]->
  Cell{v}.(Directions.(North{v1} | East{v2}) | goal);
```

Listing 1.5. Reaction rule to stay home if a required number of objects is acquired.

Lastly, we have three actions for when the agent acquires the specified number of objects: two for going home west or south (since these are the two directions necessary and sufficient to reach home), and one for staying home. The two actions for going home have two reaction rules each, depending on whether the destination cell is visited or not. They are just like the rules in Listing 1.4, except every mention of `Agent` is replaced with `goal`. The rule to stay home simply says that if the agent has achieved the goal, and is home (characterised by a `Cell` with only two directions: `North` and `East`), then nothing should change (see Listing 1.5). To make the agent start heading home as soon as enough objects are collected, all five reaction rules related to going and staying home are in a higher priority class.

4.2 Layers of Abstraction

Our second model explores two important concepts in modelling the topology of space: nesting and uncertainty. Nesting is the idea that spaces can be inside other spaces: rooms inside floors, floors inside buildings, buildings inside streets, etc. A key characteristic of nested spaces is that there is a separate action (or several) taking you from the outer space to the inner space or vice versa (e.g., using a door to enter or leave a room). In the rest of the paper, whenever we discuss

one space nested inside another, we will refer to the inner space as the *room*. While nesting has been modelled using bigraphs before [12], our approach models each room not as a single entity, but rather as an interconnected collection of smaller spaces, which we continue to call *cells*. Thus, it is not enough to be in the hallway in order to enter a room, one needs to be next to the door. Similarly, after entering a room, the agent is positioned next to the other side of the door.

The other concept explored in this section is uncertainty about the topology itself, i.e., is the door open or closed? The implementation of this idea is tied to that of rooms: the inside of a room is not defined until the outer cell encompassing the room (i.e., the area where the door is located) is entered. Then, the contents of the room are chosen from a finite set of possibilities with a set probability distribution. After that, the room stays fixed. This design decision supports the reality of missions with a reasonably short duration: if the agent finds a locked door, it should not keep revisiting the door hoping for it to suddenly be open.

As the ideas of collecting objects and tracking visited cells have been explored already, they are dropped from this model in order to simplify the implementation. The goal can simply be defined as moving from one location to another. We add one extra control, `Node`. While `Cell` can be imagined as a discrete patch of space (e.g., $1\,\mathrm{m}^2$), `Node` can be thought of as a room. While we stick to two layers (corridors and rooms) in this example, rooms can be freely nested: a room can contain any number of rooms, each of which can contain more rooms.

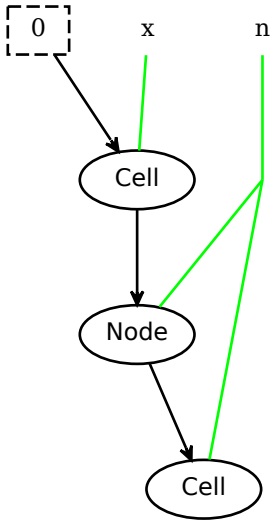

Fig. 4. The bigraphical structure behind nesting.

```
big initial = /x /y /z (
              Cell{x}.(Directions.East{a} | Agent)
```

```
    || Cell{y}.(Directions.(East{b} | West{a})
                | Node{n}.1)
    || Cell{z}.Directions.West{b});
```

Listing 1.6. The initial map: three consecutive cells, the midmost of which has an inner room.

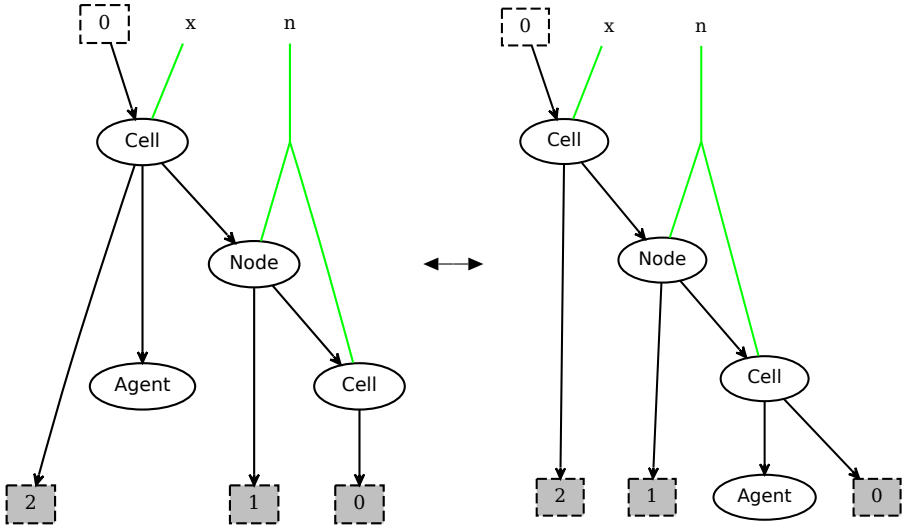

Fig. 5. Left to right: the reaction rule to go inside; right to left: the reaction rule to go outside.

In order to implement nesting, we put a `Node` (representing a room) inside a `Cell`, which has the "door" to that room (see Fig. 4). The `Node` can have its own collection of `Cells`, describing the topology of the room. One of these `Cells` is linked to the `Node`, representing the `Cell` closest to the door from the inside of the room. Thus, both `Nodes` and `Cells` have arity 1 for exactly this linking. `Cell` ports not linked to anything are implemented using closures (e.g. `/x Cell{x}`). The initial map in Listing 1.6 has a corridor of three cells with the agent on the west-most cell and a door to a room in the middle cell. One moves in and out of a room using a pair of actions with reaction rules pictured in Fig. 5.

```
action room
  react openDoor =
    Cell{w}.(Agent | Node{n}.1 | id)
    -[p]-> /x /y /z /a /b /c /d
    Cell{w}.(Agent | Node{n}.(
      # top left
        Cell{n}.Directions.(East{a} | South{b})
      # top right
```

```
      | Cell{x}.Directions.(West{a} | South{c})
      # bottom left
      | Cell{y}.Directions.(North{b} | East{d})
      # bottom right
      | Cell{z}.Directions.(North{c} | West{d}))
      | id);
  react closedDoor =
    Cell{w}.(Agent | Node{n}.1 | id)
    -[1.0-p]-> /x /y /z /b /c /d
    Cell{w}.(Agent | Node{n}.(
      # top left
        Cell{n}.Directions.(South{b})
      # top right
      | Cell{x}.Directions.(South{c})
      # bottom left
      | Cell{y}.Directions.(North{b} | East{d})
      # bottom right
      | Cell{z}.Directions.(North{c} | West{d}))
      | id);
end
```

Listing 1.7. Two ways to generate a room, leaving one door either open or closed.

```
action north
  react goNorth =
       Cell{n1}.(Directions.(North{b} | id)
                | Agent | id)
    || Cell{n2}.(Directions.(South{b} | id) | id)
    -[1.0]->
       Cell{n1}.(Directions.(North{b} | id) | id)
    || Cell{n2}.(Directions.(South{b} | id)
                | Agent | id);
end
```

Listing 1.8. The action of going north.

The room is a 2×2 grid of `Cells`, which is generated when the `Agent` gets to a `Cell` with an empty `Node` (represented by `Node{n}.1`)[6]. In order for the room to be generated before any other action can take place, the two reaction rules in Listing 1.7 are put on a higher priority class. With probability p, the generated room has each `Cell` linked to its two neighbours; and with probability $1 - p$, the top two `Cells` are not linked, possibly forcing the agent to choose a different path. Finally, we have four actions, each with a single reaction rule with probability 1, corresponding to the four possible directions of travel. The

[6] Multiple different rooms can be implemented by having each previously empty `Node` contain a node of different control. This way a different probability distribution (with different possible outcomes) can be assigned to each room.

reaction rules are very similar to those of our previous example, an example of which can be found in Listing 1.8.

5 Conclusion

BIG TODO!

References

1. Baeten, J.C.M., Bergstra, J.A., Klop, J.W., Weijland, W.P.: Term-rewriting systems with rule priorities. Theor. Comput. Sci. **67**(2&3), 283–301 (1989). https://doi.org/10.1016/0304-3975(89)90006-6, https://doi.org/10.1016/0304-3975(89)90006-6
2. Calder, M., Koliousis, A., Sevegnani, M., Sventek, J.S.: Real-time verification of wireless home networks using bigraphs with sharing. Sci. Comput. Program. **80**, 288–310 (2014). https://doi.org/10.1016/j.scico.2013.08.004, https://doi.org/10.1016/j.scico.2013.08.004
3. Conforti, G., Macedonio, D., Sassone, V.: Spatial logics for bigraphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3580, pp. 766–778. Springer (2005). https://doi.org/10.1007/11523468_62, https://doi.org/10.1007/11523468_62
4. Giaquinta, R., Hoffmann, R., Ireland, M., Miller, A., Norman, G.: Strategy synthesis for autonomous agents using PRISM. In: Dutle, A., Muñoz, C.A., Narkawicz, A. (eds.) NASA Formal Methods - 10th International Symposium, NFM 2018, Newport News, VA, USA, April 17-19, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10811, pp. 220–236. Springer (2018). https://doi.org/10.1007/978-3-319-77935-5_16, https://doi.org/10.1007/978-3-319-77935-5_16
5. Hoffmann, R., Ireland, M., Miller, A., Norman, G., Veres, S.M.: Autonomous agent behaviour modelled in PRISM - A case study. In: Bosnacki, D., Wijs, A. (eds.) Model Checking Software - 23rd International Symposium, SPIN 2016, Co-located with ETAPS 2016, Eindhoven, The Netherlands, April 7-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9641, pp. 104–110. Springer (2016). https://doi.org/10.1007/978-3-319-32582-8_7, https://doi.org/10.1007/978-3-319-32582-8_7
6. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., et al.: Jupyter notebooks - a publishing format for reproducible computational workflows. In: Loizides, F., Schmidt, B. (eds.) Positioning and Power in Academic Publishing: Players, Agents and Agendas, 20th International Conference on Electronic Publishing, Göttingen, Germany, June 7-9, 2016. pp. 87–90. IOS Press (2016). https://doi.org/10.3233/978-1-61499-649-1-87, https://doi.org/10.3233/978-1-61499-649-1-87
7. Krivine, J., Milner, R., Troina, A.: Stochastic bigraphs. Electr. Notes Theor. Comput. Sci. **218**, 73–96 (2008). https://doi.org/10.1016/j.entcs.2008.10.006, https://doi.org/10.1016/j.entcs.2008.10.006

8. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_47, https://doi.org/10.1007/978-3-642-22110-1_47
9. Sevegnani, M.: Bigraphs with sharing and applications in wireless networks. Ph.D. thesis, University of Glasgow, UK (2012), http://theses.gla.ac.uk/3742/
10. Sevegnani, M., Calder, M.: Bigraphs with sharing. Theor. Comput. Sci. **577**, 43–73 (2015). https://doi.org/10.1016/j.tcs.2015.02.011, https://doi.org/10.1016/j.tcs.2015.02.011
11. Sevegnani, M., Calder, M.: BigraphER: Rewriting and analysis engine for bigraphs. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9780, pp. 494–501. Springer (2016). https://doi.org/10.1007/978-3-319-41540-6_27, https://doi.org/10.1007/978-3-319-41540-6_27
12. Walton, L., Worboys, M.F.: A qualitative bigraph model for indoor space. In: Xiao, N., Kwan, M., Goodchild, M.F., Shekhar, S. (eds.) Geographic Information Science - 7th International Conference, GIScience 2012, Columbus, OH, USA, September 18-21, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7478, pp. 226–240. Springer (2012). https://doi.org/10.1007/978-3-642-33024-7_17, https://doi.org/10.1007/978-3-642-33024-7_17