

# Graph Abstraction and Abstract Graph Transformations (Amended Version)<sup>\*</sup>

Iovka Boneva<sup>2</sup>   Jörg Kreiker<sup>3</sup>   Marcos Kurbán<sup>4</sup>  
Arend Rensink<sup>1</sup>   Eduardo Zambon<sup>1</sup>

<sup>1</sup> Formal Methods and Tools Group, EWI-INF, University of Twente  
PO Box 217, 7500 AE, Enschede, The Netherlands  
`{rensink,zambon}@cs.utwente.nl`

<sup>2</sup> Institut Universitaire de Technologie de Lille, University of Lille 1  
Park Plaza - Bât A - 40 avenue Halley, 59650 Villeneuve d'Ascq, France  
`iovka.boneva@lifl.fr`

<sup>3</sup> Institut für Informatik (I7), Technische Universität München  
Room 03.11.040, Boltzmannstr. 3, D-85748 Garching bei München, Germany  
`kreiker@in.tum.de`

<sup>4</sup> Former member of Formal Methods and Tools Group, EWI-INF, University of Twente

**Abstract.** Many important systems such as concurrent heap-manipulating programs, communication networks, or distributed algorithms, are hard to verify due to their inherent dynamics and unboundedness. Graphs are an intuitive representation for the states of these systems, where transitions can be conveniently described by graph transformation rules.

We present a framework for the abstraction of graphs supporting abstract graph transformation. The abstraction method naturally generalises previous approaches to abstract graph transformation. The set of possible abstract graphs is finite. This has the pleasant consequence of generating a finite transition system for any start graph and any finite set of transformation rules. Moreover, abstraction preserves a simple logic for expressing properties on graph nodes. The precision of the abstraction can be adjusted according to the properties expressed in this logic that are to be verified.

---

<sup>\*</sup> The main purpose of this amended version is to correct typos, errors and omissions from previous versions of this technical report. We also tried to make the text more clear by rewriting some sentences and adding new figures. There is one major change in terminology: In the previous version of the report the term *shaping* was used to denote a morphism between a graph and a shape, and the term *abstraction morphism* to denote a morphism between two shapes. The usage of these terms were usually misleading and led to confusion. Therefore we swapped their definitions. In the current version of this report we use the term *abstraction morphism* to denote a morphism between a graph and a shape and we write *shape morphism* to indicate a morphism between two shapes.



# Table of Contents

1	Introduction . . . . .	5
1.1	Graph Transformations for System Analysis . . . . .	5
1.2	Contributions . . . . .	5
2	Graphs and Graph Transformations . . . . .	6
3	Graph Abstraction . . . . .	8
3.1	Multiplicities . . . . .	9
3.2	Shapes and Abstraction Morphisms . . . . .	9
3.3	Shape Morphism and Isomorphism of Shapes . . . . .	14
3.4	Neighbourhood Shapes . . . . .	15
4	Canonical Shapes . . . . .	19
4.1	Canonical Names . . . . .	19
4.2	Canonical Representation of Neighbourhood Shapes . . . . .	20
4.3	Canonical Shapes . . . . .	20
5	Shape Transformations . . . . .	22
5.1	Transformations of Shapes . . . . .	22
5.2	Properties of Shape Transformations . . . . .	25
5.3	Using Shape Transformations . . . . .	28
6	Materialisation and Normalisation . . . . .	29
6.1	Definition of the Set of Materialisations . . . . .	29
6.2	Effective Construction of $\mathcal{M}$ . . . . .	31
6.3	Normalisation . . . . .	32
6.4	Back to the Construction of the Abstract Labelled Transition System . . . . .	32
7	A Modal Logic for Graphs and Shapes . . . . .	33
7.1	Syntax of the Logic . . . . .	34
7.2	Satisfaction on Graphs and Shapes . . . . .	34
7.3	Preservation by Abstraction Morphism . . . . .	35
7.4	Preservation and Reflection for Neighbourhood Shaping . . . . .	36
7.5	Relationship between the Logic and Neighbourhood Shaping . . . . .	37
8	Related Work . . . . .	37
9	Conclusion and Further Directions . . . . .	39
	Bibliography . . . . .	40
	Appendices . . . . .	41
A	Proof of Proposition 12 . . . . .	41
B	Proof of Lemma 22 . . . . .	41
B.1	Proof of Statement 1 . . . . .	42
B.2	Proof of Statement 2 . . . . .	43
B.3	Proofs of Lemma 56 and Corollary 57 . . . . .	44
C	Proof of Lemma 23 . . . . .	47
D	Proof of Lemma 28 . . . . .	47
E	Proof of Proposition 50 . . . . .	49
E.1	Preservation . . . . .	50
E.2	Reflection . . . . .	51
E.3	Preservation and reflection . . . . .	52

F Proof of Lemma 44 ..... 52  
G Proof of Lemma 54 ..... 53

## 1 Introduction

Graphs are an important form of representation for the state of a system. Interesting properties of a given state have natural graph-theoretic counterparts. Also, their inherent graphical representation makes them the “lingua franca” of software engineering; they are good to convey ideas back and forth between different communities such as formal verification and specification, software engineering, and even end-users. If we add the concept of graph transformation for modelling transitions between system states, we form a framework that allows people to talk about both the states of a system and how it evolves over time.

This paper presents work carried out in the context of the GROOVE project that seeks to develop such a framework for software verification: states of a software system are represented by graphs and statements of a programming language are given by the semantics of graph transformation rules. As an example, Figure 1 depicts a possible graph representation of a linked-list. Adding a new element to the list consists of creating a new node labelled `Cell` with an associated `Object`-node, and inserting it in the desired place in the list. Removing an element from the list and many other list operations can also be seen as graph transformations.

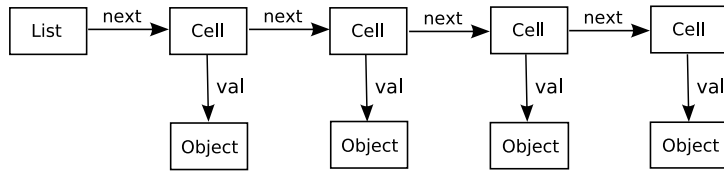
### 1.1 Graph Transformations for System Analysis

A graph transformation rule  $p : L \rightarrow R$  is given by its name  $p$  and a pair of graphs  $\langle L, R \rangle$ , often called left-hand side and right-hand side, respectively. Performing a graph transformation on a graph  $G$  using rule  $p$  can be seen as finding a sub-graph of  $G$  that is isomorphic to  $L$  and replacing it with  $R$ . Systems and system behaviour can be modelled by graphs and graph transformations. Let  $G_0$  be a graph representing an initial state of a system (e.g., the list on Figure 1) and let  $\mathbf{P}$  be a set of transformation rules encoding all possible operations of the system (e.g., operations on lists). We can explore all accessible configurations and evolutions of the system given by  $G_0$  and  $\mathbf{P}$ . This is done by applying all possible transformations from  $\mathbf{P}$  to the start graph  $G_0$  and repeating it iteratively to all graphs resulting from these transformations. This gives rise to a labelled transition system whose states are graphs and whose transitions are applications of graph transformation rules. One can then verify properties, e.g., temporal properties, using the generated transition system. The GROOVE tool [10] allows to construct (finite portions of) such transition systems and to verify temporal properties using CTL and LTL logic.

Problems do arise when approaching this task. One such problem is the possible infinite behaviour of a system which, in most cases, makes it impossible to generate its entire state space. Another problem is memory space: even for a finite state space, each state can be quite big to represent if one does it naively. A usual way to circumvent these two problems is abstraction. In Section 8 we describe several related approaches that exist.

### 1.2 Contributions

In previous work some of the authors proposed abstraction techniques in which graph nodes with similar incoming and outgoing edges [9] or similar direct neighbours [2] are summarised into a single one. Such abstract graphs are sometimes called *shapes* [14,9] and we borrow the same vocabulary here. The number of possible such shapes is bounded. This, combined with a suitable notion of graph transformations for abstract graphs [11], guarantees a finite number of states for a transition system.



**Fig. 1.** Graph representation of a list with four elements. Each Cell contains a pointer to the Object stored into it via a *val*-edge, and possibly a pointer to the next cell via a *next*-edge.

As a first contribution of the paper we introduce a family of *neighbourhood shapes* as a part of a general abstraction mechanism that subsumes previous works. For the abstraction, nodes are grouped if they have similar neighbourhood up to some “radius”  $i$ , parameter of the abstraction. This allows us to have abstractions with different precisions. Additionally, the number of possible neighbourhood shapes is bounded. Moreover, we define graph transformations for our neighbourhood shapes, which allows us to over-approximate system behaviour while keeping a finite state space.

Our second contribution is a logic that goes hand-in-hand with our abstraction method. That is, given a formula describing a property we are interested in, our abstraction method guarantees that *a*) if the formula holds for the original graph, then it holds for the abstracted graph (we call this property *preservation*); and *b*) if the formula holds for the abstracted graph, then it holds for the original one too (we call this *reflection*).

Finally, all these ingredients can be combined to define a fully automatic method which, given an initial graph, a set of graph transformation rules and a set of logic properties on the reachable graphs we are interested in, constructs a finite abstract labelled transition system on which these properties can be verified.

This report is structured as follows. Section 2 introduces graphs and graph transformations. Section 3 defines the general abstraction mechanism as well as neighbourhood shapes. In Section 4 we define canonical shapes, which are a family of shapes including neighbourhood shapes that enjoy the good property of having a unique representation. Then in Section 5 and Section 6 we define transformations on shapes and describe how they can be used for approximating system behaviour into finite labelled transition systems. In Section 7 we introduce a modal logic that is preserved and reflected by the neighbourhood abstraction mechanism. Section 8 describes some related work. Finally, we conclude in Section 9.

## 2 Graphs and Graph Transformations

We are interested in finite graphs whose edges and nodes are labelled from a finite set of labels  $\text{Lab}$ . Formally, we do not associate labels with the nodes of the graph, we use instead special edges whose target is a particular object  $\perp$  not in the set of nodes of the graph. This in particular allows us to have nodes with multiple labels, which shows to be very useful when modelling with graphs. Moreover, we allow multiple edges, *i.e.*, a graph can have several different edges with the same source and target nodes and the same label.

**Definition 1 (Graph).** A graph  $G$  is a tuple  $\langle N_G, E_G, \text{src}_G, \text{tgt}_G, \text{lab}_G \rangle$  where

- $N_G$  is a finite set of nodes;
- $E_G$  is a finite set of edges disjoint from  $N_G$ ;

- $\text{src}_G : E_G \rightarrow N_G$  and  $\text{tgt}_G : E_G \rightarrow N_G \cup \{\perp\}$  with  $\perp \notin (N_G \cup E_G)$  are mappings associating with each edge its source and target nodes, respectively; and
- $\text{lab}_G : E_G \rightarrow \text{Lab}$  is a labelling map for the edges of the graph. ◀

The mapping  $\text{lab}_G$  is extended on nodes to designate the set of labels of a node, i.e.,  $\text{lab}_G(v) = \{\mathbf{a} \in \text{Lab} \mid \exists e \in E_G : \text{src}_G(e) = v, \text{tgt}_G(e) = \perp, \text{lab}_G(e) = \mathbf{a}\}$ .<sup>1</sup> We denote as  $v \triangleright_G^{\mathbf{a}}$  and  $v \triangleleft_G^{\mathbf{a}}$  the set of  $\mathbf{a}$ -outgoing edges and  $\mathbf{a}$ -incoming edges of the node  $v$ , respectively. That is,  $v \triangleright_G^{\mathbf{a}} = \{e \in E_G \mid \text{src}_G(e) = v, \text{lab}_G(e) = \mathbf{a}\}$  and symmetrically for  $v \triangleleft_G^{\mathbf{a}}$ . For a set of nodes  $V$ ,  $V \triangleright_G^{\mathbf{a}}$  (resp.  $V \triangleleft_G^{\mathbf{a}}$ ) is the extension of  $\triangleright_G^{\mathbf{a}}$  (resp.  $\triangleleft_G^{\mathbf{a}}$ ) on sets. Finally, for  $X, Y$  sets of nodes or nodes, we denote  $X \triangleright_G^{\mathbf{a}} Y$  the set of edges labelled  $\mathbf{a}$  and going from  $X$  to  $Y$ , i.e.,  $X \triangleright_G^{\mathbf{a}} Y = X \triangleright_G^{\mathbf{a}} \cap Y \triangleleft_G^{\mathbf{a}}$ . When graph  $G$  is clear from the context, we may omit the subscript  $G$  in  $N_G$ ,  $E_G$ ,  $\text{src}_G$ ,  $\text{tgt}_G$ ,  $\text{lab}_G$ ,  $\triangleright_G^{\mathbf{a}}$ ,  $\triangleleft_G^{\mathbf{a}}$ , and  $\triangleright_G^{\mathbf{a}}$ .

**Definition 2 (Graph Morphism).** *If  $G$  and  $H$  are graphs, a graph morphism  $f : G \rightarrow H$  is a function from  $N_G \cup E_G \cup \{\perp\}$  to  $N_H \cup E_H \cup \{\perp\}$  such that*

- $f$  preserves  $\perp$ , i.e.,  $f(\perp) = \perp$ ,  $f^{-1}(\perp) = \{\perp\}$ ;
- $f$  maps nodes to nodes and edges to edges, i.e.,  $f(N_G) \subseteq N_H$ ,  $f(E_G) \subseteq E_H$ ;
- $f$  is compatible with source and target mappings, i.e.,  $\text{src}_H \circ f = f \circ \text{src}_G$ , and  $\text{tgt}_H \circ f = f \circ \text{tgt}_G$ ; and
- $f$  preserves labels, i.e.,  $\text{lab}_H \circ f = \text{lab}_G$ . ◀

A morphism  $f$  is called *injective* (resp. *surjective*, resp. *bijective*) if it defines an injective (resp. surjective, resp. bijective) map. A bijective morphism is also called an *isomorphism*.

For the sake of clarity, in the sequel of the paper we ignore the node  $\perp$  and simply talk about node labels. It is easy to see that all the proofs can be adapted to this formal definition using the  $\perp$  node.

## Background on Graph Transformations

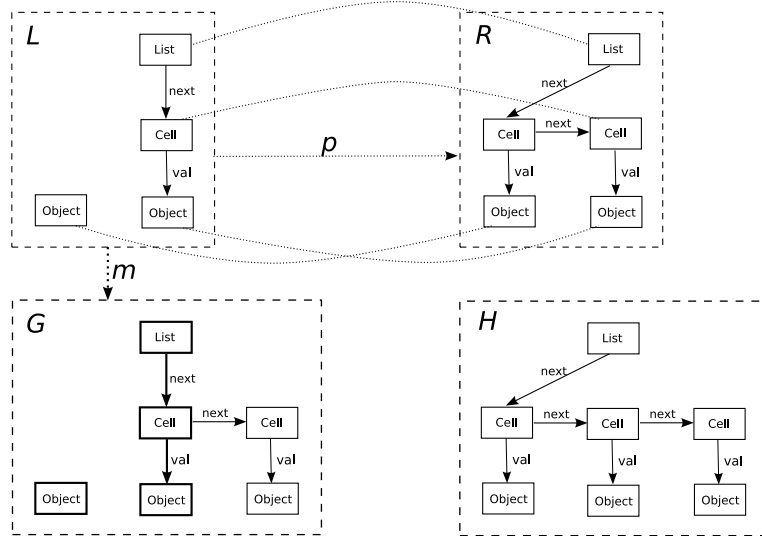
Let us start with some notations for functions. For a set  $A$ , we denote  $\text{id}_A$  the identity function on  $A$ . For two functions  $f, g$ , we denote  $f \cup g$  their union, that is,  $f \cup g$  is the function whose domain is the union of the domains of  $f$  and  $g$  and whose co-domain is the union of the co-domains of  $f$  and  $g$ . The union of functions is defined only if for any  $x$  belonging both to the domains of  $f$  and  $g$ ,  $f$  and  $g$  agree on their value for  $x$ .

**Definition 3 (Transformation Rule).** *A graph transformation rule  $P$  is a pair of graphs  $\langle L, R \rangle$ , called left-hand side and right-hand side respectively. A transformation rule can be seen as the single graph  $L \cup R$ . In this case we distinguish the following sets:*

- $N_P^{\text{del}} = N_L \setminus N_R$  and  $E_P^{\text{del}} = E_L \setminus E_R$  are the elements to be deleted;
- $N_P^{\text{new}} = N_R \setminus N_L$  and  $E_P^{\text{new}} = E_R \setminus E_L$  are the elements to be created;
- $N_P^{\text{use}} = N_L \cap N_R$  and  $E_P^{\text{use}} = E_L \cap E_R$  are the elements that remain unchanged. ◀

The subscript  $P$  is omitted when clear from the context.

<sup>1</sup> Note that  $\text{lab}_G(e)$  is a label for an edge  $e$ , and  $\text{lab}_G(v)$  is a set of labels for a node  $v$ .



**Fig. 2.** Example of a transformation rule  $P = \langle L, R \rangle$  and its application to a graph  $G$  via matching  $m : L \rightarrow G$ . Rule morphism  $p$  is indicated by dotted lines. For the sake of readability, the matching  $m : L \rightarrow G$  is indicated by highlighting its image  $m(L)$  in  $G$ . The host graph  $G$  represents a list with two elements with some additional object in the environment. The application of the rule results in adding a new element at the head of the list.

**Definition 4 (Graph Transformation).** Let  $G$  be a graph and  $P = \langle L, R \rangle$  be a transformation rule such that  $G$  and  $P$  are disjoint. A matching  $m$  for  $P$  into  $G$  is an injective morphism  $m : L \rightarrow G$  satisfying the so called dangling edges application condition: for any edge  $e$  of  $G$ , if  $\text{src}(e) \in m(N^{\text{del}})$  or  $\text{tgt}(e) \in m(N^{\text{del}})$ , then  $e \in m(E^{\text{del}})$ .

If  $m$  is a matching for  $P$  into  $G$ , then the transformation of  $G$  according to  $P$  and  $m$  is the graph  $H$  defined as follows (with  $m' : P \rightarrow G$  the morphism  $m \cup \text{id}_{N^{\text{new}} \cup E^{\text{new}}}$ ):

- $N_H = (N_G \setminus m(N^{\text{del}})) \cup N^{\text{new}}$ ;
- $E_H = (E_G \setminus m(E^{\text{del}})) \cup E^{\text{new}}$ ;
- $\text{src}_H = \text{src}_G \cup m' \circ \text{src}_P$  restricted to  $E_H$ ;
- $\text{tgt}_H = \text{tgt}_G \cup m' \circ \text{tgt}_P$  restricted to  $E_H$ ; and
- $\text{lab}_H = \text{lab}_G \cup \text{lab}_P$  restricted to  $E_H$ .

We write  $G \xrightarrow{P, m} H$  to designate that  $m$  is a matching for  $P$  in  $G$  and  $H$  is the graph resulting from the transformation. ◀

The dangling edges application condition is standard in the so called double push-out approach for graph transformation. It ensures that performing a transformation does not introduce dangling edges (edges without source or target node).

Figure 2 depicts a transformation rule that adds an element to the head of a list. An example application of this rule is also shown.

### 3 Graph Abstraction

In this section, abstract graphs are called shapes. The name “shape” comes from work in shape analysis [14], where abstract graphs are used to represent pointer structures. Any node and



any edge of a given shape may represent several nodes/edges of some *concrete* graph. We want it to carry information on the number of summarised nodes/edges. To define interesting abstractions, it seems necessary that this multiplicity information to be approximate: think for instance about abstracting a list independently of its length. In Section 3.1 we introduce the notion of *multiplicity* for handling approximate information on cardinals of sets. Then, in Section 3.2 we define the shapes that we consider, as well as the abstraction mechanism which is essentially a morphism from a graph to a shape that satisfies some extra conditions.

Shapes may be more or less abstract. In particular, a shape may be abstracted to another shape. This yields a relation between shapes, which we define in Section 3.3. In the same section, we also define isomorphism of shapes and show that isomorphic shapes represent the same sets of concrete graphs.

Finally, in Section 3.4, we define a particular family of shapes called *neighbourhood shapes*. Neighbourhood shapes have several interesting properties that are studied in the rest of the paper.

### 3.1 Multiplicities

A multiplicity is an approximation of the cardinal of a (finite) set. Intuitively, all sets containing strictly more than  $\mu$  elements, for some fixed natural  $\mu$ , are considered to have the same cardinal. This notion of multiplicity was also used in [9].

**Definition 5 (Multiplicity).** *For a natural number  $\mu > 0$ , let  $\mathbf{M}_\mu$  be the set  $\{0, 1, 2, \dots, \mu, \omega\}$  where  $\omega$  is distinct from all natural numbers. The multiplicity with precision  $\mu$  is the function associating with each finite set  $U$  the value  $|U|_\mu$  in  $\mathbf{M}_\mu$  defined by:*

$$|U|_\mu = \begin{cases} |U| & \text{if } |U| \leq \mu, \\ \omega & \text{otherwise.} \end{cases}$$

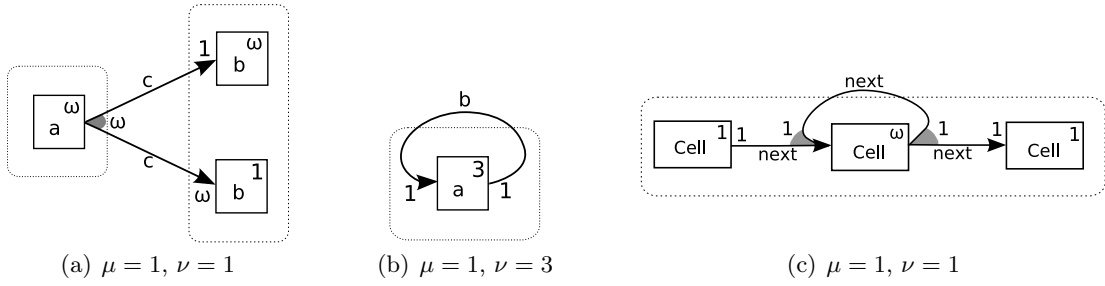
*The value  $|U|_\mu$  is called the  $\mu$ -multiplicity of  $U$ , or simply the multiplicity of  $U$  if  $\mu$  is clear from the context. Elements of  $\mathbf{M}_\mu$  are called multiplicities. We use  $\mathbf{M}_\mu^+$  to denote the set  $\mathbf{M}_\mu \setminus \{0\}$ . ◀*

We extend the usual ordering  $\geq$  over elements of  $\mathbf{M}_\mu$  by defining  $\omega \geq \lambda$  for any  $\lambda$  in  $\mathbf{M}_\mu$ . Sum can also be extended over multiplicities on the expected way: let  $I$  be a finite index set and let  $(\lambda_i)_{i \in I}$  be elements of  $\mathbf{M}_\mu$ . Then  $\sum_{i \in I}^\mu \lambda_i$ , the  $\mu$ -sum of the  $(\lambda_i)_{i \in I}$ , is  $|\bigcup_{i \in I} A_i|_\mu$  where the  $(A_i)_{i \in I}$  are pairwise disjoint sets such that  $|A_i|_\mu = \lambda_i$  for any  $i$  in  $I$ .

In the sequel of the paper, we consider two naturals  $\nu, \mu$ . Whenever their value is not specified, they may have any positive value. The  $\nu$ -multiplicity is used for giving the multiplicity of sets of nodes, and  $\mu$ -multiplicity for giving the multiplicity of sets of edges. In particular, these two numbers are parameters of graph abstractions.

### 3.2 Shapes and Abstraction Morphisms

A shape is a graph together with a *node multiplicity function* that indicates, for each node of the shape, how many concrete nodes it summarises. Moreover, the set of nodes is partitioned into groups. Edges with same source node, and ending into nodes of the same group (or, respectively, edges with the same target node, and starting in nodes of the same group)



**Fig. 3.** Examples of shapes.

cannot be distinguished. Only the number of such edges is indicated by the *edge multiplicity functions* of the shape.

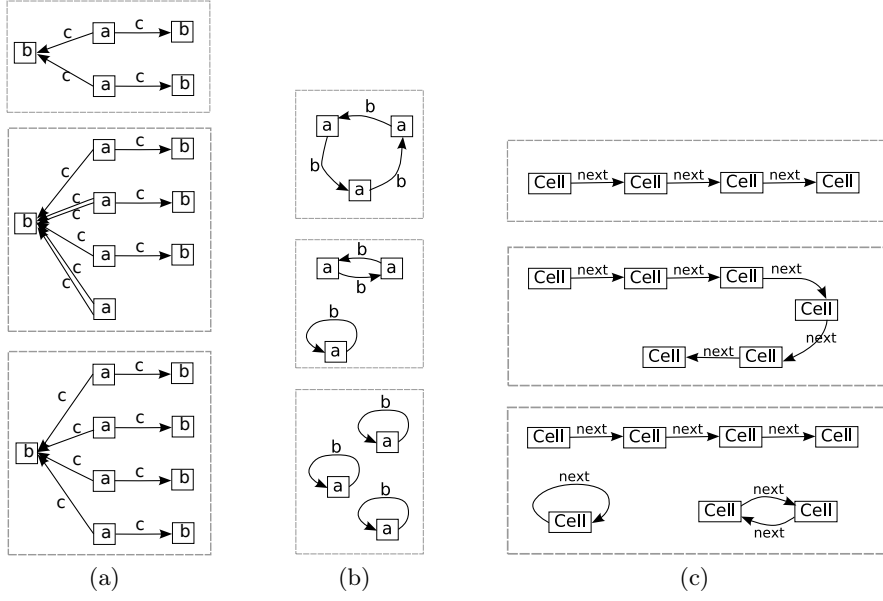
We start by giving a flavour of what a shape is, in the following example.

*Example 6 (Shape).* Figure 3 depicts three shapes as well as values for  $\mu$  and  $\nu$  for these shapes. With each node of each shape is associated a multiplicity from  $\mathbf{M}_\nu^+$ , indicating the number of concrete graph nodes it represents; this is called the *node multiplicity*. The dotted rectangles are delimiting groups of nodes. By definition, this grouping can be arbitrary; in practise it would be defined by some common characteristic (e.g., nodes with same label, nodes with similar neighbourhood, etc). All edges have associated multiplicity information (from  $\mathbf{M}_\mu$ ) in their end points. Sometimes, this multiplicity is shared by several edges, indicated by the grey arc relating them. These are the so-called *outgoing edges multiplicity*, when associated to source of the edge, and *incoming edges multiplicity* when associated to the target. An edge multiplicity intuitively indicates how many of the depicted edges should exist in a concrete graph. One can notice that edges related in one of their end points all have their other end point in the same group of nodes, and all have the same label. Actually, this is the condition for relating edges. To be more precise, according to the formal definition, edge multiplicities are associated with a triple composed of a node, a label and a group of nodes. This is presented in Definition 7.

Let us now explain how one should interpret these example shapes.

- (a). The shape on Figure 3(a) represents a set of bipartite concrete graphs in which **a**-nodes are connected to **b**-nodes by **c**-edges. Each of these graphs has at least two (here  $\omega$  on nodes or edges stands for “two or more”, as  $\nu = \mu = 1$ ) **a**-nodes and at least three ( $\omega$  plus one) **b**-nodes. Moreover, every **a**-node has at least two (i.e.,  $\omega$ ) outgoing **c**-edges going to **b**-nodes. All **b**-nodes except one have only one incoming edge; the remaining **b**-node has at least two incoming edges. See Figure 4(a) for some example concrete graphs.
- (b). The shape on Figure 3(b) represents a set of concrete graphs having three **a**-nodes connected to each other and forming cycles of **b**-edges. See Figure 4(b) for some example concrete graphs.
- (c). The shape on Figure 3(c) represents a set of list-like concrete graphs having **Cell**-nodes connected by **next**-edges. Each of these graphs has at least one acyclic connected component of length four or more with several (possibly zero) cyclic connected components of arbitrary length. See Figure 4(c) for some example concrete graphs. ◀

Before giving the formal definition of a shape, let us fix some notations. Let  $A$  be a set and  $\sim \subseteq A \times A$  be an equivalence relation over  $A$ . For  $x \in A$ , we denote  $[x]_{\sim}$  the equivalence class



**Fig. 4.** Example concrete graphs that can be abstracted to the shapes on Figure 3.

of  $x$  induced by  $\sim$ , i.e.,  $[x]_{\sim} = \{y \in A \mid y \sim x\}$ . We denote  $A/\sim$  the set of equivalence classes in  $A$ , i.e.,  $A/\sim = \{[x]_{\sim} \mid x \in A\}$ . Moreover, if  $\sim$  and  $\sim'$  are two equivalence relations over  $A$ , we write  $\sim \subseteq \sim'$  whenever for all  $x, y \in A$ ,  $x \sim y$  implies  $x \sim' y$ . Note that if  $\sim \subseteq \sim'$ , then any equivalence class for  $\sim$  is included into the equivalence class for  $\sim'$ , that is, for all  $x \in A$ ,  $[x]_{\sim} \subseteq [x]_{\sim'}$ . This means in particular that any equivalence class for  $\sim'$  can be obtained as an union of equivalence classes for  $\sim$ .

Formally, a shape is defined as follows:

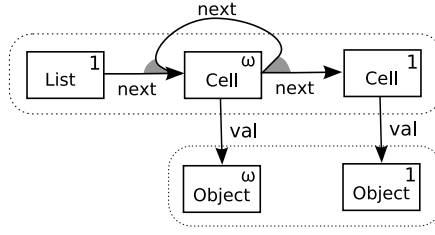
**Definition 7 (Shape).** A shape  $S$  is a structure  $\langle G_S, \simeq_S, \text{mult}_S^n, \text{mult}_S^o, \text{mult}_S^i \rangle$  where

- $G_S = \langle N_S, E_S, \text{src}_S, \text{tgt}_S, \text{lab}_S \rangle$  is a graph;
- $\simeq_S \subseteq N_S \times N_S$  is an equivalence relation on  $N_S$  called the grouping relation of  $S$ ;
- $\text{mult}_S^n : N_S \rightarrow \mathbf{M}_\nu^+$  is a node multiplicity function;
- $\text{mult}_S^o : N_S \times \text{Lab} \times N_S / \simeq_S \rightarrow \mathbf{M}_\mu$  is an outgoing edge multiplicity function; and
- $\text{mult}_S^i : N_S \times \text{Lab} \times N_S / \simeq_S \rightarrow \mathbf{M}_\mu$  is an incoming edge multiplicity function.

Moreover, for any node  $v \in N_S$ , any label  $\mathbf{a} \in \text{Lab}$  and any equivalence class of nodes  $C \in N_S / \simeq_S$ , we require that  $\text{mult}_S^o(v, \mathbf{a}, C) = 0$  if, and only if,  $v \triangleright_{G_S}^{\mathbf{a}} C = \emptyset$ , and  $\text{mult}_S^i(v, \mathbf{a}, C) = 0$  if, and only if,  $C \triangleright_{G_S}^{\mathbf{a}} v = \emptyset$ .  $\blacktriangleleft$

As already mentioned, a shape is a representation of a set of concrete graphs. In this sense, it is an abstract graph. The fact that some concrete graph is *abstracted* to a given shape is determined by the presence of the so called *abstraction morphism*, which is a morphism from the graph to the shape that complies to some additional constraints. We say then that the graph is a *concretisation* of the shape.

**Definition 8 (Abstraction Morphism, Concretisation).** Let  $G$  be a graph and  $S$  be a shape. An abstraction morphism of  $G$  into  $S$  is a graph morphism  $s : G \rightarrow G_S$  such that the following conditions are met:



**Fig. 5.** Example of a shape for a list. All edge multiplicities are equal to one and are omitted.

- for all  $w \in N_S$ ,  $\text{mult}_S^n(w) = |s^{-1}(w)|_v$ ;
- for all  $w \in N_S$ , for all  $\mathbf{a} \in \text{Lab}$ , for all  $C \in N_S / \simeq_S$ , and for all  $v \in s^{-1}(w)$ ,

$$\text{mult}_S^o(w, \mathbf{a}, C) = |v \triangleright_G^{\mathbf{a}} (s^{-1}(C))|_\mu$$

and

$$\text{mult}_S^i(w, \mathbf{a}, C) = |(s^{-1}(C)) \triangleright_G^{\mathbf{a}} v|_\mu.$$

◀

If  $G$  is a graph and  $S$  is a shape such that there exists an abstraction morphism  $s : G \rightarrow S$ , then we say that  $G$  is a *concretisation* of  $S$ . The set of concretisations of a shape  $S$  is denoted  $\text{concr}(S)$ .

*Example 9.* The list structure from Figure 1 is a concretisation for the shape shown in Figure 5. The corresponding morphism maps the **List**-node of the graph to the **List**-node of the shape, the right-most **Cell**-node and the right-most **Object**-node from the graph are mapped to the corresponding right-most nodes from the shape. The remaining **Cell**-nodes and **Object**-nodes from the graph are mapped to the left-hand side such nodes of the shape. ◀

Note that an abstraction morphism is always surjective; this follows from the requirement for the  $\text{mult}_S^n$  function together with the fact that  $\text{mult}_S^n$  maps to non-null multiplicities, by definition of shapes. The requirements on outgoing (resp. incoming) edge multiplicities guarantee in particular that two different nodes  $v, v'$  from a graph  $G$  can be mapped to the same node  $w$  of a shape  $S$  only if  $v, v'$  have the same outgoing (resp. incoming) edges multiplicities with respect to a label and group of nodes.

**Construction of Shapes.** In Definitions 7 and 8, a shape  $S$  is a graph-like structure defined independently on any of its concretisations. A graph  $G$  can be abstracted to a shape  $S$  if there exists a morphism from  $G$  to the graph part of  $S$  satisfying some conditions. In particular, these definitions do not give a hint on how to construct shapes. In the following, we present an alternative, constructive way of defining a shape by providing a graph and two equivalence relations on its nodes.

Let  $G$  be a graph and  $\sim, \equiv \subseteq N_G \times N_G$  be two equivalence relations on the nodes of  $G$  satisfying the following conditions:

- (C1)  $\equiv \subseteq \sim$ , that is, if  $v \equiv v'$ , then  $v \sim v'$ ;

(C2) for any  $v, v'$  nodes of  $G$ , for any  $\sim$ -equivalence class of nodes  $C \in N_G / \sim$  and for any label  $\mathbf{a}$ , if  $v \equiv v'$ , then

$$|v \triangleright_G^{\mathbf{a}} C|_{\mu} = |v' \triangleright_G^{\mathbf{a}} C|_{\mu}$$

and

$$|C \triangleright_G^{\mathbf{a}} v|_{\mu} = |C \triangleright_G^{\mathbf{a}} v'|_{\mu}$$

Let the equivalence relation  $\equiv$  be extended on edges of  $G$  in the following way:  $e \equiv e'$  if  $\text{src}_G(e) \equiv \text{src}_G(e')$ ,  $\text{tgt}_G(e) \equiv \text{tgt}_G(e')$  and  $\text{lab}_G(e) = \text{lab}_G(e')$ .

Consider now the graph  $G_S = \langle N_S, E_S, \text{src}_S, \text{tgt}_S, \text{lab}_S \rangle$  defined by:

- nodes of  $G_S$  are  $\equiv$ -equivalence classes of nodes of  $G$ , i.e.,  $N_S = N_G / \equiv$ ;
- edges of  $G_S$  are  $\equiv$ -equivalence classes of edges of  $G$ , i.e.,  $E_S = E_G / \equiv$ ; and
- for any edge  $[e]_{\equiv}$  in  $E_S$ ,  $\text{src}_S([e]_{\equiv}) = [\text{src}_G(e)]_{\equiv}$ ,  $\text{tgt}_S([e]_{\equiv}) = [\text{tgt}_G(e)]_{\equiv}$  and  $\text{lab}_S([e]_{\equiv}) = \text{lab}_G(e)$ . Note that, due to the definition of  $\equiv$  on edges, the particular choice of  $e$  for  $[e]_{\equiv}$  is not important.

Consider finally the mapping  $s : N_G \cup E_G \rightarrow N_S \cup E_S$  defined by:  $s(v) = [v]_{\equiv}$  and  $s(e) = [e]_{\equiv}$  for any  $v$  in  $N_G$  and any  $e$  in  $E_G$ . The next lemma follows easily from the definitions, so we present it without proof.

**Lemma 10.** 1. *The mapping  $s$ , canonically extended to  $\perp$ , defines a surjective graph morphism from  $G$  into  $G_S$ ; by abuse of notation we denote this morphism  $s$  as well.*

2. *Let*

- $\sim_S \subseteq N_S \times N_S$  be the equivalence relation on nodes of  $G_S$  defined by  $[v]_{\equiv} \sim_S [v']_{\equiv}$  if  $v \sim v'$  for all  $v, v'$  nodes of  $G$ . Due to Condition (C1),  $\sim_S$  is well defined;
- $\text{mult}_S^n : N_S \rightarrow \mathbf{M}_\nu^+$  be the mapping defined by  $\text{mult}_S^n(w) = |s^{-1}(w)|_{\nu}$  for all  $w$  in  $N_S$ ;
- $\text{mult}_S^o, \text{mult}_S^i : N_S \times \text{Lab} \times N_S / \sim_S \rightarrow \mathbf{M}_{\mu}$  be the mappings defined by

$$\text{mult}_S^o([v]_{\equiv}, \mathbf{a}, C) = |v \triangleright_G^{\mathbf{a}} s^{-1}(C)|_{\mu} \quad \text{mult}_S^i([v]_{\equiv}, \mathbf{a}, C) = |s^{-1}(C) \triangleright_G^{\mathbf{a}} v|_{\mu}$$

for all  $v \in N_G$ ,  $\mathbf{a} \in \text{Lab}$  and  $C \in N_S / \sim_S$ . Due to Condition (C2),  $\text{mult}_S^o$  and  $\text{mult}_S^i$  are well-defined.

Then  $S = \langle G_S, \sim_S, \text{mult}_S^n, \text{mult}_S^o, \text{mult}_S^i \rangle$  is a shape and  $s$  is an abstraction morphism. ◀

It follows from this lemma that, given a graph  $G$  and two equivalence relations on the nodes of  $G$  satisfying Condition (C1) and Condition (C2), one can define a shape  $S$  such that there exists an abstraction morphism  $s : G \rightarrow S$ . Note that not all shapes can be defined this way, for two reasons.<sup>2</sup> First, shapes defined as in Lemma 10 necessarily have concretisations, and there exist shapes without concretisations. Second, shapes defined as in Lemma 10 cannot have parallel edges (i.e., edges with the same source and target node, and the same label), whereas shapes may have such parallel edges. Nevertheless, it is the case that any shape admitting concretisations and without parallel edges can be defined<sup>3</sup> by a graph  $G$  and two equivalence relations, as explained.

For a graph  $G$  and equivalence relations  $\sim$  and  $\equiv$  satisfying Condition (C1) and Condition (C2), we define  $\text{shape}(G, \sim, \equiv)$  as the shape described by Lemma 10 and we call  $\text{absMorph}(G, \sim, \equiv)$  the corresponding abstraction morphism.

<sup>2</sup> Actually, there is a third reason which has to do with representation, and that is ignored here. The shapes defined as in Lemma 10 come with their representation: nodes are equivalence classes of nodes of some graph, edges are equivalence classes of edges of some graph, and so on. Thus, two isomorphic, but not equal, graphs would define two different shapes, although intuitively we would consider these two shapes as equivalent. This “equivalence” of shapes is called shape isomorphism and is defined in Section 3.3.

<sup>3</sup> Up to isomorphism; see also Footnote 2.

### 3.3 Shape Morphism and Isomorphism of Shapes

Just like graphs can be abstracted to shapes, shapes can be abstracted (embedded) into (more abstract) shapes. In this section we describe this relation, defined by the presence of the so called *shape morphism* between shapes. Then we show that these morphisms are composable. We also use shape morphisms to define the notion of *isomorphism* between shapes, with the interesting property that isomorphic shapes have the same concretisations. As we will see, these properties allow us to define a partial order on shapes.

**Definition 11 (Shape Morphism).** *Let  $S$  and  $T$  be two shapes. A shape morphism between them is a graph morphism  $f : S \rightarrow T$  that complies to the following axioms:*

1. *for all  $v, v' \in N_S$ :  $v \simeq_S v'$  implies  $f(v) \simeq_T f(v')$ ;*
2. *for all  $w \in N_T$ :  $\text{mult}_T^n(w) = \left( \sum_{v \in f^{-1}(w)} \text{mult}_S^n(v) \right)$ ;*
3. *for all  $w \in N_T$ , all  $\mathbf{a} \in \text{Lab}$ , all  $C \in N_T / \simeq_T$ , and all  $v \in f^{-1}(w)$ , it holds*

$$\text{mult}_T^o(w, \mathbf{a}, C) = \sum_{D \in (f^{-1}(C)) / \simeq_S}^{\mu} \text{mult}_S^o(v, \mathbf{a}, D)$$

and

$$\text{mult}_T^i(w, \mathbf{a}, C) = \sum_{D \in (f^{-1}(C)) / \simeq_S}^{\mu} \text{mult}_S^i(v, \mathbf{a}, D).$$

*When such a morphism exists, we say that  $S$  is a sub shape of  $T$ , and we denote it  $S \sqsubseteq T$ . ◀*

Let us first argue that these axioms are well defined. In the third axiom we are summing up the  $\text{mult}_S^o(v, \mathbf{a}, D)$  and  $\text{mult}_S^i(v, \mathbf{a}, D)$  for all  $D \in (f^{-1}(C)) / \simeq_S$ . It is then necessary that all the triples  $(v, \mathbf{a}, D)$  belong to the domain of  $\text{mult}_S^i$ , that is, it is necessary that any such  $D$  belongs to  $N_S / \simeq_S$ . This is indeed the case due to the first axiom. Let us now make a comparison between abstraction and shape morphisms. The second condition for the shape morphism corresponds to the first condition for the abstraction morphism, but we are summing up node multiplicities instead of simply counting nodes. The third condition on the shape morphism is very close to the second condition for the abstraction morphism, but we are taking into account outgoing and incoming edge multiplicities instead of simply counting edges.

**Proposition 12 (Shape Morphisms are Composable).** *Let  $S$ ,  $T$  and  $U$  be shapes,  $f$  be a shape morphism between  $S$  and  $T$  and  $g$  another such morphism between  $T$  and  $U$ . Then  $g \circ f$  (the function composition of  $f$  and  $g$ ) is a shape morphism between  $S$  and  $U$ . ◀*

*Proof.* See Appendix A. ◻

Let us point out that an abstraction and a shape morphism can also be composed, resulting into an abstraction morphism. The next proposition is presented without proof, but it is not difficult to see that it follows from Proposition 12 and the definition of abstraction morphism.

**Proposition 13 (Abstraction and Shape Morphisms).** *Let  $G$  be a graph and  $S$  and  $T$  be shapes such that there exist an abstraction morphism  $s : G \rightarrow S$  and a shape morphism  $f : S \rightarrow T$ . Then,  $f \circ s : G \rightarrow T$  is an abstraction morphism. ◀*

Shapes that are the sub-shapes of one another are called isomorphic.

**Definition 14 (Isomorphism of Shapes).** *Two shapes  $S$  and  $T$  are isomorphic if there exists an isomorphism  $f : G_S \rightarrow G_T$  such that  $f$  and  $f^{-1}$  are shape morphisms. In this case,  $f$  is called a shape isomorphism. ◀*

It is easy to see from the definitions that if  $f : S \rightarrow T$  is a shape isomorphism, then the grouping relation  $\simeq_T$  is such that  $f(v) \simeq_T f(w)$  if, and only if,  $v \simeq_S w$ ; the node multiplicity function  $\text{mult}_T^n$  is such that  $\text{mult}_T^n(f(v)) = \text{mult}_S^n(v)$ , and analogously for the edge multiplicity functions.

**Lemma 15 (Isomorphism and Concretisations).** *If two shapes  $S$  and  $T$  are isomorphic, then they have the same concretisations. ◀*

*Proof.* Immediately follows from the definitions and Proposition 13. ◻

The inverse is not true. Consider for instance two shapes  $S$  and  $T$  as follows:  $S$  has a single node of multiplicity two and no edges.  $T$  has two nodes, each of multiplicity one, and no edges.  $S$  and  $T$  both have a unique concretisation (up to graph isomorphism) which is the graph with two nodes and no edges, but  $S$  and  $T$  are clearly not isomorphic. Another example are shapes without concretisations, which may have very different underlying graph structures.

**Partial order relation over shapes** Two shapes are considered equivalent if they have the same concretisations; we denote this equivalence relation  $=_{\text{concr}}$ . That is, for all shapes  $S, T$ ,  $S =_{\text{concr}} T$  if, and only if,  $\text{concr}(S) = \text{concr}(T)$ .

**Lemma 16 (Partial Order).** *The sub-shape relation  $\sqsubseteq$  defines a partial order between shapes with respect to the equivalence relation  $=_{\text{concr}}$ . ◀*

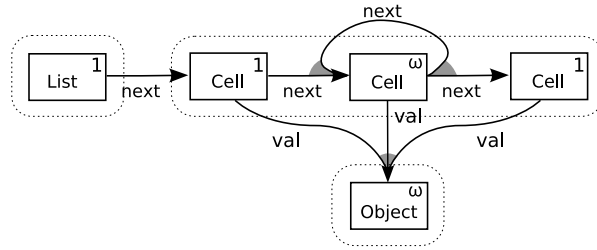
*Proof.*  $\sqsubseteq$  is clearly reflexive; it is antisymmetric for the equivalence relation  $=_{\text{concr}}$ , by definition of isomorphism of shapes and by Lemma 15. Finally,  $\sqsubseteq$  is transitive by Proposition 12. ◻

It is also easy to see that the  $\sqsubseteq$  relation is compatible with the subset relation on concretisations, in the sense that  $S \sqsubseteq T$  implies that  $\text{concr}(S) \subseteq \text{concr}(T)$ . This is an immediate consequence of Propositions 12 and 13. This partial order could be a first step towards a link between our abstraction mechanism and abstract interpretation (see, e.g., [6]). However, it does not allow us to define immediately a Galois connection between graphs and shapes, but between sets of graphs and sets of shapes, as the sub-shape relation is in connection with the subset relation on graphs.

### 3.4 Neighbourhood Shapes

Neighbourhood shapes are a special family of shapes that have several interesting properties, established on the rest of the paper. For the moment, let us only point out the possibility to parametrise the precision of abstraction offered by neighbourhood shapes. Precision of (general) shapes, that we considered up to now, can already be parametrised by the two multiplicities  $\mu$  and  $\nu$ . In a neighbourhood shape, each (abstract) node represents concrete graph nodes that have similar neighbourhood, up to some “radius”  $i$ . This  $i$  is also a parameter of the precision of neighbourhood shapes.

Neighbourhood abstraction (i.e., abstracting into a neighbourhood shape) is always defined for graphs. That is, for any values of the parameters  $\mu$ ,  $\nu$  and  $i$ , and for any graph  $G$ , there



**Fig. 6.** Level one neighbourhood shape of a list. All edge multiplicities are equal to one and are omitted.

exists a neighbourhood shape with the corresponding precision that is a shape for  $G$ . This does not hold for shape morphisms: some shapes can be embedded into a neighbourhood shape with a given precision, but for other shapes this is not possible.

Hereafter, we define neighbourhood abstraction for graphs and shapes, describing the conditions for existence of the latter. For both, we first define the so-called *neighbourhood equivalence* over nodes and edges of a graph (resp. shape) on which the neighbourhood abstraction is based.

### Neighbourhood Shape of a Graph

**Definition 17 (Neighbourhood Equivalence).** *Let  $G$  be a graph. For each natural  $i$ , the  $i$  neighbourhood equivalence relation  $\equiv_i$  between nodes of  $G$  is recursively defined as:*

- $v \equiv_0 v'$  if  $\text{lab}_G(v) = \text{lab}_G(v')$ ;
- $v \equiv_{i+1} v'$  if  $v \equiv_i v'$ , and  $|v \triangleright_G^a C|_\mu = |v' \triangleright_G^a C|_\mu$ , and  $|C \triangleright_G^a v|_\mu = |C \triangleright_G^a v'|_\mu$  for all label  $a$  in  $\text{Lab}$  and for all set of nodes  $C \in N / \equiv_i$ .

*The  $i$ -neighbourhood equivalence relation is extended to edges of  $G$  by  $e \equiv_i e'$  if  $\text{lab}_G(e) = \text{lab}_G(e')$ ,  $\text{src}_G(e) \equiv_i \text{src}_G(e')$ , and  $\text{tgt}_G(e) \equiv_i \text{tgt}_G(e')$ . ◀*

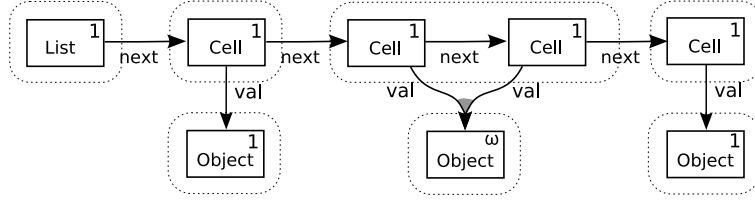
We can now define the family of neighbourhood abstraction morphisms. Two nodes are mapped to the same shape node if they are neighbourhood equivalent up to some radius. The grouping relation is also given by neighbourhood equivalence, but using a smaller radius.

**Definition 18 (Neighbourhood Shape of a Graph, Neighbourhood Abstraction Morphism of a Graph).** *For any natural  $i \geq 1$ , the level  $i$  neighbourhood shape of  $G$  is defined by  $\text{shape}(G, \equiv_{i-1}, \equiv_i)$  and the level  $i$  neighbourhood abstraction morphism of  $G$  by  $\text{absMorph}(G, \equiv_{i-1}, \equiv_i)$ . ◀*

Figures 6 and 7 show respectively the level one and level two neighbourhood shapes of the list from Figure 1, for  $\mu = 1$  and  $\nu = 1$ . Defining the corresponding abstraction morphisms is left to the reader.

The neighbourhood shape of a graph cannot be dissociated from the graph because of its representation: nodes and edges of the shape are sets of nodes and sets of edges of the graph. This situation is not very convenient: we would like to be able to talk about neighbourhood shapes of graphs to designate their properties and not some particular representation, that is, to designate their isomorphism class. Thus, we overload the terms neighbourhood shape and neighbourhood abstraction morphism in the following way. In the sequel, we use the term





**Fig. 7.** Level two neighbourhood shape of a list. All edge multiplicities are equal to one and are omitted.

*neighbourhood shape of graph  $G$*  to designate the isomorphism class of the actual neighbourhood shape of  $G$  in the sense of Definition 18, and we use the term *neighbourhood abstraction morphism of graph  $G$*  for morphisms  $s : G \rightarrow S$  such that  $s = f \circ s'$ , where  $s' : G \rightarrow S'$  is the actual neighbourhood shape of  $G$  and  $f : S' \rightarrow S$  is a shape isomorphism.

### Neighbourhood Shape of a Shape

**Definition 19 (Neighbourhood Equivalence for Shapes).** *Let  $S$  be a shape defined by the tuple  $S = \langle G_S, \simeq_S, \text{mult}_S^n, \text{mult}_S^o, \text{mult}_S^i \rangle$ . For any  $i \geq 0$ , the binary relation  $\sim_i$  over nodes of  $S$  is defined as:*

- $w \sim_0 w'$  if  $\text{lab}_S(w) = \text{lab}_S(w')$ ;
- $w \sim_{i+1} w'$  if  $w \sim_i w'$ ,  $\simeq_S \subseteq \sim_i$  and for all  $C \in N_S / \sim_i$ , and for all labels  $a$ ,

$$\sum_{K \in N_S / \simeq_S \mid K \subseteq C}^{\mu} \text{mult}_S^o(w, a, K) = \sum_{K \in N_S / \simeq_S \mid K \subseteq C}^{\mu} \text{mult}_S^o(w', a, K)$$

and analogously for the incoming edges multiplicity function.

The relation  $\sim_i$  is extended to edges of  $S$  by:  $e \sim_i e'$  if  $\text{src}_S(e) \sim_i \text{src}_S(e')$ ,  $\text{tgt}_S(e) \sim_i \text{tgt}_S(e')$  and  $\text{lab}_S(e) = \text{lab}_S(e')$ . ◀

The requirement  $\simeq_S \subseteq \sim_i$  intuitively says that the grouping relation  $\simeq_S$  should be “finer”, in the sense of grouping less nodes than the  $\sim_i$  relation that we are trying to define. Note that the requirement  $\simeq_S \subseteq \sim_i$  is necessary, as it ensures that any  $K \in N_S / \simeq_S$  is a subset of some  $C \in N_S / \sim_i$ . If this requirement is not fulfilled, then the sums in the definition above are not defined. In this case, the relations  $\sim_j$  for any  $j > i$  are empty.

**Lemma 20.** *Let  $S$  be a shape and  $i \geq 1$ . If the relation  $\sim_i$  over the nodes of  $S$  is not empty, then  $\sim_i$  is an equivalence relation.* ◀

*Proof.* By definition of  $\sim_i$ ,  $\sim_i$  is empty if and only if  $\simeq_S \not\subseteq \sim_{i-1}$ . Now, if  $\simeq_S \subseteq \sim_{i-1}$ , then it is easy to see that  $\sim_i$  is symmetric, reflexive and transitive. ◻

**Definition 21 (Neighbourhood Shape of a Shape, Neighbourhood Shape Morphism of a Shape).** *Let  $S$  be a shape and  $i \geq 1$ . If the relation  $\sim_i$  over the nodes of  $S$  is not empty, let  $T$  be the shape defined as:*

- nodes of  $T$  are  $[v]_{\sim_i}$ , for any  $v$  node of  $N_S$ ;
- edges of  $T$  are  $[e]_{\sim_i}$ , for any  $e$  edge of  $E_S$ ;

- for any edge  $e' = [e]_{\sim_i}$  in  $E_T$  (for  $e \in E_S$ ),  $\text{src}_T(e') = [\text{src}_S(e)]_{\sim_i}$ ,  $\text{tgt}_T(e') = [\text{tgt}_S(e)]_{\sim_i}$  and  $\text{lab}_T(e') = \text{lab}_S(e)$ . By definition of  $\sim_i$  these are well defined;
- $\simeq_T = \sim_{i-1}$ ;
- for any  $w \in N_T$ ,

$$\text{mult}_T^n(w) = \sum_{v \in N_S \mid [v]_{\sim_i} = w}^{\nu} \text{mult}_S^n(v);$$

- for any  $w \in N_T$ , any label  $\mathbf{a}$ , any  $C \in N_T / \simeq_T$ , and some  $v \in N_S$  such that  $[v]_{\sim_i} = w$ ,

$$\text{mult}_T^o(w, \mathbf{a}, C) = \sum_{K \in N_S / \simeq \mid K \subseteq C}^{\mu} \text{mult}_S^o(v, \mathbf{a}, K)$$

and similarly for incoming edges multiplicities.

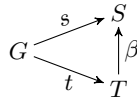
Then  $T$  is called the level  $i$  neighbourhood shape of  $S$ . ◀

Note that the edge multiplicity functions are well defined by definition of  $\sim_i$ . Also note that in the last item of the definition we may pick any  $v$  in the equivalence class  $[v]_{\sim_i} = w$  because all multiplicities sums for elements of the same class are equal. This was checked when  $\sim_i$  was built.

We conclude the section with two properties of neighbourhood shapes and neighbourhood shape morphism that are used in Section 6.

**Lemma 22 (Composition of neighbourhood morphisms).** *Let  $G$  be a graph,  $S, T$  be shapes,  $s : G \rightarrow S$ ,  $t : G \rightarrow T$  be abstraction morphisms, and  $\beta : T \rightarrow S$  be a shape morphism such that  $s = \beta \circ t$ .*

1. *If  $s$  is the neighbourhood abstraction morphism of  $G$ , then  $\beta$  is the neighbourhood shape morphism of  $T$ .*
2. *If  $\beta$  is the neighbourhood shape morphism of  $T$ , then  $s$  is the neighbourhood abstraction morphism of  $G$ .*



*Proof.* See Appendix B. ◻

**Lemma 23 (Common concretisation implies isomorphism).** *If any two neighbourhood shapes have a common concretisation, then they are isomorphic.* ◀

*Proof.* The proof of the lemma uses the canonical representation of neighbourhood shapes, defined in Section 4. Thus, we give it in Appendix C. ◻

## 4 Canonical Shapes

Canonical shapes are a special class of shapes that includes neighbourhood shapes. More precisely, this class is composed of neighbourhood shapes, and of shapes that do not admit concretisations. Canonical shapes have a so called “canonical” representation which is a representation of isomorphism classes of such shapes. This in particular allows us to define a normalised representation of (isomorphism classes of) neighbourhood shapes. Moreover, for each shaping precision (*i.e.*, values for  $\mu$ ,  $\nu$  and the neighbourhood radius  $i$ ), the number of canonical shapes is finite. Additionally, it is decidable whether a shape is (isomorphic to a) canonical shape, and in this case its canonical representation can be computed. All these properties make canonical shapes a good over-approximation of the set of neighbourhood shapes.

### 4.1 Canonical Names

In this section, we introduce the notion of a *canonical name*. Each equivalence class with respect to a neighbourhood equivalence is uniquely identified by such a name. For example, each equivalence class with respect to  $\equiv_0$  contains only nodes having the same labels and is identified by this set of labels. It becomes the canonical name of this equivalence class. Each equivalence relation  $\equiv_i$  comes with a set  $\text{NCan}^i$  of canonical names. A neighbourhood shape can be viewed as a graph whose nodes and edges are canonical names. The notion of a canonical name occurs frequently in literature, for example in [15].

**Definition 24 (Canonical Name).** *The set of level  $i$  canonical node names,  $\text{NCan}^i$ , is defined inductively for  $i \geq 0$ :*

$$\begin{aligned} \text{NCan}^0 &= {}_2\text{Lab} \\ \text{NCan}^{i+1} &= \text{NCan}^i \times (\text{NCan}^i \times \text{Lab} \rightarrow \mathbf{M}_\mu) \times (\text{NCan}^i \times \text{Lab} \rightarrow \mathbf{M}_\mu). \end{aligned}$$

*The set  $\text{ECan}^i$  of level  $i$  canonical edge names is  $\text{ECan}^i = \text{NCan}^i \times \text{Lab} \times \text{NCan}^i$ . ◀*

Let  $G$  be a graph. The mapping  $\text{name}_G^i$  maps nodes and edges of  $G$  to their level  $i$  canonical name as follows. For node  $v$  of  $G$ ,  $\text{name}_G^0(v) = \text{lab}_G(v)$ , and  $\text{name}_G^{i+1}(v) = \langle \text{name}_G^i(v), \text{out}, \text{in} \rangle$  where for each canonical name  $C$  in  $\text{NCan}^i$  and for each label  $\mathbf{a}$  in  $\text{Lab}$  ( $N_C$  stands for the set of nodes  $v'$  such that  $\text{name}_G^i(v') = C$ ),

$$\text{out}(C, \mathbf{a}) = |v \triangleright_G^{\mathbf{a}} N_C|_\mu \qquad \text{in}(C, \mathbf{a}) = |N_C \triangleright_G^{\mathbf{a}} v|_\mu.$$

For edge  $e$  of  $G$ ,  $\text{name}_G^i(e) = \langle \text{name}_G^i(\text{src}(e)), \text{lab}(e), \text{name}_G^i(\text{tgt}(e)) \rangle$ .

*Example 25.* Consider the level zero canonical node name  $C_0 = \{\mathbf{c}, \mathbf{d}\}$  and the level one canonical node name  $C_1 = \langle \{\mathbf{a}\}, \mathbf{0}, \text{in} \rangle$ , where  $\mathbf{0}$  indicates the constant function associating 0 to all elements of its domain, and  $\text{in}(C_0, \mathbf{b}) = 1$ , and  $\text{in}(C', x) = 0$  for all  $C' \neq C_0$  and all  $x \neq \mathbf{b}$ .  $C_0$  is the class of nodes labelled  $\mathbf{c}$  and  $\mathbf{d}$ .  $C_1$  is the class of nodes labelled  $\mathbf{a}$  that have one incoming  $\mathbf{b}$ -edge from a node labelled  $\mathbf{c}$  and  $\mathbf{d}$  and *no more* adjacent nodes. ◀

Note that the number of level  $i$  canonical names is exponentially growing in  $i$ . However, for any  $i$ , this number is bounded in terms of the number of node labels and  $\mu$ .

*Note 26.* For any  $i \geq 0$ , the sets of level  $i$  node and edge canonical names are finite. ◀

The number of different canonical names is growing super-exponentially in  $i$ , that is,  $|\text{NCan}^i| \geq {}^i m = \underbrace{m^{m^{\dots^m}}}_i$ , where  $m = \mu + 2$ . Nevertheless, we believe that in practical cases the number of used different canonical names would not reach this upper bound.

## 4.2 Canonical Representation of Neighbourhood Shapes

There is a quite clear relation between canonical names and the neighbourhood equivalence relation: two nodes (resp. edges) in a graph are  $i$ -neighbourhood equivalent if, and only if, they have the same level  $i$  canonical names. Next lemma easily follows from the definitions thus we present it without proof.

**Lemma 27.** *For any  $i \geq 0$ , any graph  $G$ , any two nodes  $v, v'$  of  $G$  and any two edges  $e, e'$  of  $G$ ,  $v \equiv_i v'$  if, and only if,  $\text{name}_G^i(v) = \text{name}_G^i(v')$ , and  $e \equiv_i e'$  if, and only if,  $\text{name}_G^i(e) = \text{name}_G^i(e')$ . ◀*

In what follows we show that this correspondence gives rise to a canonical representation of neighbourhood shapes. We first introduce the actual representation, and then show that it is canonical, in the sense of uniqueness (up to shape isomorphism).

Let  $G$  be a graph. Consider the triple  $\mathcal{C}_G = \langle \text{name}^i(N_G), \text{name}^i(E_G), \text{mult} \rangle$ , where  $\text{name}^i(N_G)$  and  $\text{name}^i(E_G)$  are the sets of node and edge level  $i$  canonical names of the graph  $G$ , respectively, and  $\text{mult} : \text{name}^i(N_G) \rightarrow \mathbf{M}_\nu^+$  is the function defined for all  $C \in \text{name}^i(N_G)$  by  $\text{mult}(C) = |\{v \in N_G \mid \text{name}_G^i(v) = C\}|_\nu$ . We show that  $\mathcal{C}_G$  is a canonical representation of the isomorphism class of the level  $i$  neighbourhood shape of  $G$ . This provides us with a representation of neighbourhood shapes that is independent of the graphs they were computed from.

**Lemma 28 (Canonical Representation).** *Let  $G, H$  be graphs, and let  $i \geq 1$ . The level  $i$  neighbourhood shapes of  $G$  and  $H$  are isomorphic if, and only if,  $\mathcal{C}_G$  and  $\mathcal{C}_H$  are equal. ◀*

By  $\mathcal{C}_G$  and  $\mathcal{C}_H$  are equal, we mean component-wise equality, that is, equality of the sets of node and edge canonical names and equality of the node multiplicity functions that define them.

*Proof.* The proof is given in Appendix D since it uses results that are introduced later, namely the relation between neighbourhood shape morphisms and the modal logic defined in Section 7. ◻

Thus, by Lemma 28 we know that any isomorphism class of level  $i$  neighbourhood shapes has a canonical representation of the form  $\langle \mathcal{N}, \mathcal{E}, \text{mult} \rangle$ , where  $\mathcal{N} \subseteq \text{NCan}^i$ ,  $\mathcal{E} \subseteq \text{ECan}^i$ , and  $\text{mult} : \mathcal{N} \rightarrow \nu$ . Then the question arises what is the relationship between triples from  $\langle \mathcal{N}, \mathcal{E}, \text{mult} \rangle$  and neighbourhood shapes. This is shown in the next section.

## 4.3 Canonical Shapes

We denote  $\mathcal{CS}_*^i$  the set of triples  $2^{\text{NCan}^i} \times 2^{\text{ECan}^i} \times (\text{NCan}^i \rightarrow \mathbf{M}_\nu^+)$  such that for any  $\langle \mathcal{N}, \mathcal{E}, \text{mult} \rangle \in \mathcal{CS}_*^i$ ,  $\text{dom}(\text{mult}) = \mathcal{N}$ . We will see that some elements of  $\mathcal{CS}_*^i$  define shapes. It is decidable to check, for a given  $\mathcal{C} \in \mathcal{CS}_*^i$ , whether it defines a shape. Moreover, some elements of  $\mathcal{CS}_*^i$  define neighbourhood shapes, but we believe that it is not decidable to know whether an element of  $\mathcal{CS}_*^i$  defines a neighbourhood shape. However, we give a syntactic definition of a subset of  $\mathcal{CS}_*^i$  which contains all neighbourhood shapes.

**From Canonical Names to Shapes.** Let  $\langle \mathcal{N}, \mathcal{E}, \text{mult} \rangle \in \mathcal{CS}_*^i$ , and consider the structure  $S = \langle \langle \mathcal{N}, \mathcal{E}, \text{src}, \text{tgt}, \text{lab} \rangle, \simeq, \text{mult}^n, \text{mult}^o, \text{mult}^i \rangle$ , where  $\text{src}, \text{tgt} : \mathcal{E} \rightarrow \text{NCan}^i$ ,  $\text{lab} : \mathcal{E} \rightarrow \text{Lab}$ ,  $\simeq$  is an equivalence relation in  $\mathcal{N}$ ,  $\text{mult}^n : \mathcal{N} \rightarrow \mathbf{M}_\nu^+$ , and  $\text{mult}^o, \text{mult}^i : \mathcal{N} \times \text{Lab} \times \mathcal{N} / \simeq \rightarrow \mathbf{M}_\mu$  defined as:

- for any  $e = \langle C, \mathbf{a}, C' \rangle$  in  $\mathcal{E}$ ,  $\text{src}_S(e) = C$ ,  $\text{tgt}_S(e) = C'$  and  $\text{lab}_S(e) = \mathbf{a}$ ;
- $\simeq$  is the smallest equivalence relation such that  $C \simeq C'$  if  $C$  and  $C'$  have the same first component. Remind that  $C$  and  $C'$  are level  $i$  node canonical names and their first component is a level  $i - 1$  canonical name;
- $\text{mult}^n = \text{mult}$ ;
- for all  $C \in N_S$ ,  $\mathbf{a} \in \text{Lab}$ , and  $K \in \text{NCan}^{i-1}$ ,  $\text{mult}^o(C, \mathbf{a}, K) = \text{out}_C(K, \mathbf{a})$ , where  $\text{out}_C$  is the function in the second component of  $C$  (remind that  $C$  is a level  $i$  canonical name and  $\text{out}_C : \text{NCan}^{i-1} \times \text{Lab} \rightarrow \mu$ );
- for all  $C \in N_S$ ,  $\mathbf{a} \in \text{Lab}$ , and  $K \in \text{NCan}^{i-1}$ ,  $\text{mult}^i(C, \mathbf{a}, K) = \text{in}_C(K, \mathbf{a})$ , where  $\text{in}_C$  is the function in the third component of  $C$ .

The following lemma identifies the conditions on  $\langle \mathcal{N}, \mathcal{E}, \text{mult} \rangle$  under which  $S$  is a shape.

**Lemma 29.** *If*

1.  $\mathcal{E} \subseteq \mathcal{N} \times \text{Lab} \times \mathcal{N}$ , and
2. for all  $C \in \mathcal{N}$ , all  $K$  in  $\text{NCan}^{i-1}$  and any label  $\mathbf{a}$ ,  $\text{out}_C(K, \mathbf{a}) = 0$  if, and only if,  $\{\langle C, \mathbf{a}, C' \rangle \in \mathcal{E} \mid \pi_1(C') = K\} = \emptyset$  (where  $\pi_1(C')$  denotes the first component of  $C'$ ), and similarly for  $\text{in}_C$ .

then  $S$  is a shape. ◀

*Proof.* The first condition ensures that  $\langle \mathcal{N}, \mathcal{E}, \text{src}, \text{tgt}, \text{lab} \rangle$  is a graph, and the second condition ensures that the edge multiplicity functions of  $S$  are consistent with its graph structure, i.e., an edge multiplicity is positive if, and only if, there is indeed a number of edges in the graph that corresponds to the multiplicity value. □

For  $\mathcal{C} \in \mathcal{CS}_*^i$  satisfying the condition from Lemma 29, we denote  $S_{\mathcal{C}}$  the corresponding shape.

We have now a characterisation of elements of  $\mathcal{CS}_*^i$  that define shapes. In what follows we give some characteristics of elements of  $\mathcal{CS}_*^i$  that represent neighbourhood shapes.

**Definition 30 (Canonical Shape).** *A level  $i$  canonical shape is a shape of the form  $S_{\mathcal{C}}$ , for  $\mathcal{C} \in \mathcal{CS}_*^i$ , and such that  $S_{\mathcal{C}}$  is (isomorphic to) its own level  $i$  neighbourhood shape.* ◀

We denote  $\mathcal{CS}^i$  the set of level  $i$  canonical shapes. Canonical shapes are usually represented as elements of  $\mathcal{CS}_*^i$ , i.e., triples composed of a set of node canonical names, a set of edge canonical names, and a multiplicity function. This is called their *canonical representation*.

**Lemma 31 (Relationship between Neighbourhood Shapes and Canonical Shapes).**

*The following two are equivalent, for all level  $i$  canonical shape  $\mathcal{C}$ :*

1. *The shape  $S_{\mathcal{C}}$  is isomorphic to the neighbourhood shape of some graph  $G$ .*
2. *The shape  $S_{\mathcal{C}}$  admits concretisations.* ◀

*Proof.* The implication  $1 \Rightarrow 2$  is immediate from the definitions. For the implication  $2 \Rightarrow 1$ , let  $\beta : S_{\mathcal{C}} \rightarrow S_{\mathcal{C}}$  be the level  $i$  neighbourhood shape morphism of  $S_{\mathcal{C}}$ . By hypothesis, we know that there exists a graph  $G$  and an abstraction morphism  $s : G \rightarrow S_{\mathcal{C}}$ . Then, by Proposition 13 we know that  $\beta \circ s$  is an abstraction morphism, and by Lemma 22 we deduce that  $\beta \circ s$  is the level  $i$  neighbourhood abstraction morphism of  $G$ . □

That is, shapes that can be obtained by neighbourhood abstraction are exactly canonical shapes that admit concretisations, up to isomorphism. In the following, we are interested at the set  $\mathcal{CS}^i$  as a superset of the set of level  $i$  neighbourhood shapes.

The decidability of checking if a canonical shape is a neighbourhood shape is not known. Note that, according to Lemma 31, it requires the decision on whether a canonical shape admits concretisations.

*Conjecture 32.* It is not decidable whether a shape admits concretisations. ◀

Even if this conjecture is confirmed, it still does not answer the previous question of decidability whether a canonical shape admits concretisations. Our intuition is that the conjecture also holds for canonical shapes.

*Remark 33 (On Isomorphism of Canonical Shapes).* We do not know whether two canonical shapes can be isomorphic without having the same node and edge sets. However, if it could happen, say  $\mathcal{C}$  and  $\mathcal{C}'$  are isomorphic but do not have the same node and edge sets, then necessarily  $\mathcal{C}$  and  $\mathcal{C}'$  are not neighbourhood shapes (i.e., do not have concretisations). Indeed, by Lemma 15, two shapes are isomorphic if, and only if, they have the same concretisations and, by definition, the canonical representation of a neighbourhood shape is unique for its entire isomorphism class. ◀

## 5 Shape Transformations

In this section we define transformations of shapes. We also establish how transformations of shapes are related to transformations of their concretisations. Finally, we discuss on properties of transformations of neighbourhood shapes.

### 5.1 Transformations of Shapes

**Definition 34 (Pre-matching).** Let  $L$  be a graph and  $S$  be a shape. A pre-matching  $p$  of  $L$  into  $S$  is a graph morphism  $p : L \rightarrow G_S$  such that:

1. for all node  $w$  in  $p(L)$ ,  $\text{mult}_S^n(w) \geq |p^{-1}(w)|_\nu$ ,
2. for all label  $\mathbf{a} \in \text{Lab}$ , node  $v \in N_L$ , and edge  $e \in p(v) \triangleright_S^{\mathbf{a}}$ ; it holds (with  $w = \text{tgt}_S(e)$ )

$$\text{mult}_S^o(p(v), \mathbf{a}, [w]_{\simeq_S}) \geq |v \triangleright_L^{\mathbf{a}} p^{-1}(w)|_\mu$$

3. for all label  $\mathbf{a} \in \text{Lab}$ , node  $v \in N_L$ , and edge  $e \in p(v) \triangleleft_S^{\mathbf{a}}$ ; it holds (with  $w = \text{src}_S(e)$ )

$$\text{mult}_S^i(p(v), \mathbf{a}, [w]_{\simeq_S}) \geq |p^{-1}(w) \triangleright_L^{\mathbf{a}} v|_\mu.$$

A pre-matching  $p$  is called *concrete* if  $p$  is an injective morphism and additionally satisfies the following properties:

4. for all node  $v$  in  $p(N_L)$ ,  $\text{mult}_S^n(v) = 1$ ;
5. for all node  $v$  in  $p(N_L)$ , the equivalence class  $[v]_{\simeq_S}$  is the singleton set  $\{v\}$ .
6. for all nodes  $v, w$  in  $p(N_L)$  and for all label  $\mathbf{a} \in \text{Lab}$ ,  $\text{mult}_S^o(v, \mathbf{a}, \{w\}) = |v \triangleright_{G_S}^{\mathbf{a}} w|_\mu = \text{mult}_S^i(w, \mathbf{a}, \{v\})$ . ◀

As shown in the next lemma, the existence of a concrete pre-matching  $p : L \rightarrow S$  guarantees the existence of a matching  $m : L \rightarrow G$  for some graph  $G$  that is a concretisation of  $S$ . A concrete pre-matching  $p$  is a pre-matching whose image can be considered as a concrete “sub-graph” of the shape. That is, nodes in the image of  $p$  are concrete nodes, *i.e.*, with multiplicity one. Let us explain in more detail what the conditions on edges and edge multiplicities are meant for. First, Conditions 2 and 3 guarantee that the actual number of edges can indeed exist in some concretisation, so that an injective morphism from  $L$  into this concretisation can be constructed. Injectiveness of  $p$  guarantees that there are at least as many edges present from  $v$  to  $w$  in  $G_S$  as there are edges from  $p^{-1}(v)$  to  $p^{-1}(w)$  in  $L$  (this for all labels). Finally, Condition 6 guarantees that the actual number of edges present from  $v$  to  $w$  in  $G_S$  is the same that what is required by edge multiplicities. This of course is underspecified if  $\text{mult}^\circ(v, \mathbf{a}, \{w\}) = \omega$ , in which case any number of edges greater or equal to  $\mu + 1$  is correct as soon as this number is greater or equal to  $(p^{-1}(v)) \triangleright_L^{\mathbf{a}} (p^{-1}(w))$  so that it guarantees injectiveness. This underspecified number of edges plays a role in the definition of a concrete shape transformation.

**Lemma 35.** *If  $c : L \rightarrow S$  is a concrete pre-matching from the graph  $L$  to the shape  $S$ , then for any graph  $G$  that is a concretisation of  $S$  with abstraction morphism  $s : G \rightarrow S$ , there exists an injective graph morphism  $m : L \rightarrow G$  such that  $c = s \circ m$ . ◀*

*Proof.* Let  $G$  be a concretisation of  $S$  with corresponding abstraction morphism  $s : G \rightarrow S$ . Note first that for any node or edge  $x \in N_L \cup E_L$ ,  $s^{-1}(c(x))$  is a singleton set. This fact is easily shown using that  $c$  is a concrete pre-matching and that  $s$  is an abstraction morphism. Consider now the mapping  $m : N_L \cup E_L \rightarrow N_G \cup E_G$  defined by  $m(x) = y$  where  $y$  is the unique element of  $s^{-1}(c(x))$ . Thus,  $c = s \circ m$ . The fact that  $m$  is a morphism follows from the fact that  $s$  and  $c$  are morphisms, and injectiveness of  $m$  follows from injectiveness of  $c$  and the fact that  $s$  is a function. ◻

**Definition 36 (Concrete Shape Transformation).** *Let  $P = \langle L, R \rangle$  be a transformation rule and  $S$  be a shape disjoint from  $L$  and  $R$ , and let  $c$  be a concrete pre-matching from  $L$  into  $S$  satisfying the following dangling edges condition: for all edge  $e$  of  $S$ , if  $\text{src}_S(e) \in c(N^{\text{del}})$  or  $\text{tgt}_S(e) \in c(N^{\text{del}})$ , then  $e \in c(E^{\text{del}})$ . Then the transformation of  $S$  according to  $P$  and  $c$  is the shape  $T$  defined by:*

- the graph part of  $T$ , is the graph  $G_T$  such that  $G_S \xrightarrow{P, c} G_T$ ;
- the grouping relation  $\simeq_T$  is defined by
  - for all  $v \in N_S \cap N_T$ ,  $[v]_{\simeq_T} = [v]_{\simeq_S}$ ;
  - for all  $v \in N^{\text{new}}$ ,  $[v]_{\simeq_T} = \{v\}$ ;
- the node-multiplicity function of  $T$  is given by: for all  $v \in N_T$ ,

$$\text{mult}_T^{\mathbf{a}}(v) = \begin{cases} \text{mult}_S^{\mathbf{a}}(v) & \text{if } v \in N_S \cap N_T \\ 1 & \text{if } v \in N^{\text{new}}; \end{cases}$$

- let  $N_{\text{concr}} = c(N^{\text{use}}) \cup N^{\text{new}}$  and  $N_{\text{abstr}} = N_T \setminus c(N^{\text{use}})$ ; thus  $N_{\text{concr}}$  and  $N_{\text{abstr}}$  are disjoint,  $N_T = N_{\text{concr}} \cup N_{\text{abstr}}$  and  $N_S \cap N_T = N_{\text{abstr}} \cup c(N^{\text{use}})$ . Then, for all  $v \in N_T$ ,  $\mathbf{a} \in \text{Lab}$ ,

$C \in N_T / \simeq_T$ , the outgoing edge multiplicity function of  $T$  is given by:

$$\mathbf{mult}_T^o(v, \mathbf{a}, C) = \begin{cases} \left| v \triangleright_{G_T}^{\mathbf{a}} C \right|_{\mu} & \text{if } v \in N_{\text{concr}} \text{ and } C \subseteq N_{\text{concr}}, \\ \mathbf{mult}_S^o(v, \mathbf{a}, C) & \text{if } v \in N_{\text{abstr}} \text{ and } C \subseteq N_{\text{abstr}}, \\ \mathbf{mult}_S^o(v, \mathbf{a}, C) & \text{if } v \in N_{\text{abstr}} \text{ and } C \subseteq c(N^{\text{use}}) \\ & \text{or } v \in c(N^{\text{use}}) \text{ and } C \subseteq N_{\text{abstr}}, \\ 0 & \text{otherwise;} \end{cases}$$

– for all  $v \in N_T$ ,  $\mathbf{a} \in \mathbf{Lab}$ ,  $C \in N_T / \simeq_T$ , the incoming edge multiplicity function of  $T$  is given by:

$$\mathbf{mult}_T^i(v, \mathbf{a}, C) = \begin{cases} \left| C \triangleright_{G_T}^{\mathbf{a}} v \right|_{\mu} & \text{if } v \in N_{\text{concr}} \text{ and } C \subseteq N_{\text{concr}}, \\ \mathbf{mult}_S^i(v, \mathbf{a}, C) & \text{if } v \in N_{\text{abstr}} \text{ and } C \subseteq N_{\text{abstr}}, \\ \mathbf{mult}_S^i(v, \mathbf{a}, C) & \text{if } v \in N_{\text{abstr}} \text{ and } C \subseteq c(N^{\text{use}}) \\ & \text{or } v \in c(N^{\text{use}}) \text{ and } C \subseteq N_{\text{abstr}}, \\ 0 & \text{otherwise.} \end{cases}$$

We write  $S \xrightarrow{P, c} T$  to denote the concrete shape transformation. ◀

In Definition 36 we make some explicit assumptions on the sets  $C$  used in the definitions of the edge multiplicity functions of  $T$ . Let us show that these assumptions hold and thus that  $T$  is well defined.

The first assumption is that for all  $C \in N_T / \simeq_T$  we have  $C \subseteq N_{\text{concr}}$  or  $C \subseteq N_{\text{abstr}}$ , or  $C \subseteq c(N^{\text{use}})$ . Let us show that for all  $v$  node of  $T$ ,  $[v]_{\simeq_T}$  is a subset of one of the sets  $N_{\text{abstr}}$ ,  $N^{\text{new}}$  or  $c(N^{\text{use}})$ . It is sufficient to show, by definition, that  $N_{\text{concr}} = N^{\text{new}} \cup c(N^{\text{use}})$ . If  $v \in c(N^{\text{use}})$ , by hypothesis of  $c$  being a concrete pre-shaping, we know that  $[v]_{\simeq_S} = \{v\}$ , and by definition of  $\simeq_T$ ,  $[v]_{\simeq_T} = [v]_{\simeq_S}$ . If  $v \in N^{\text{new}}$ , then, by definition of  $\simeq_T$  we know that  $[v]_{\simeq_T} = \{v\}$ . Finally, if  $v \in N_{\text{abstr}}$ , by definition of  $\simeq_T$  we have  $[v]_{\simeq_T} = [v]_{\simeq_S} \subseteq N_S$ . Moreover, as stated previously, we know that  $v \not\prec_S w$  for all  $w \in c(N^{\text{new}})$ , thus  $[v]_{\simeq_T} \subseteq N_S \cap c(N^{\text{new}}) = N_{\text{abstr}}$ .

The second assumption we make is that whenever  $C \subseteq N_{\text{abstr}}$  or  $C \subseteq c(N^{\text{use}})$ ,  $C$  is also a set in  $N_S / \simeq_S$  (as it is used as argument of the edge multiplicity functions of  $S$ ). This is the case due to the definition of  $\simeq_T$ , and using the fact that  $N_S \cap N_T = N_{\text{abstr}} \cup c(N^{\text{use}})$ .

Another point to be clarified in Definition 36 is the definition of the value of  $\mathbf{mult}_T^o(v, \mathbf{a}, C)$  when  $v \in N_{\text{concr}}$  and  $C = \{w\} \subseteq N_{\text{concr}}$  (the same for incoming edges multiplicity). This value is defined as the number of edges actually present in the shape (up to  $\mu$ ), and not as some computation involving edge multiplicity functions of  $S$ , as one may expect. This in particular means that the shape  $T$  is not uniquely defined, and depends on the representation of the graph part of  $S$ . However, this non determinism is intended, and guarantees correctness of concrete shape transformation with respect to the corresponding graph transformations when deletion of edges is involved. Consider nodes  $v, w$  in  $c(N^{\text{use}})$  and label  $\mathbf{a}$  with  $\mathbf{mult}_S^o(v, \mathbf{a}, \{w\}) = \mathbf{mult}_S^i(v, \mathbf{a}, \{w\}) = \omega$ , and suppose that rule  $P$  specifies the deletion of  $k$   $\mathbf{a}$ -labelled edges between these nodes. Then  $T$  has  $\omega - k$   $\mathbf{a}$ -labelled edges from  $v$  to  $w$ , and of course this is not uniquely specified, as there may be several multiplicities  $\lambda \in \mathbf{M}_{\mu}$  such that  $\lambda + k = \omega$ .

**Definition 37 (Abstract Shape Transformation).** Let  $P = \langle L, R \rangle$  be a transformation rule,  $S$  be a shape and  $f : L \rightarrow S$  be a pre-matching. We say that  $S$  abstractly transforms into



$T$  according to  $P$  and  $f$ , and we write  $S \xrightarrow{(P,f)} T$ , whenever there exists a shape  $S'$ , a shape morphism  $\beta : S' \rightarrow S$  and a concrete pre-matching  $c : L \rightarrow S'$  such that  $f = \beta \circ c$ , and there exists a shape morphism  $\beta' : T' \rightarrow T$ , where  $T'$  is the shape such that  $S' \xrightarrow{(P,c)} T'$ .

$$\begin{array}{ccccc}
 & & f & & \\
 & \curvearrowright & & \curvearrowleft & \\
 L & \xrightarrow{c} & S' & \xrightarrow{\beta} & S \\
 & & (P,c) \downarrow & & \downarrow (P,f) \\
 & & T' & \xrightarrow{\beta'} & T
 \end{array}$$

◀

## 5.2 Properties of Shape Transformations

In this section we consider a fixed natural  $i \geq 1$ . When we use the terms neighbourhood shape and neighbourhood shape morphism, we mean level  $i$  neighbourhood shape and level  $i$  neighbourhood shape morphism.

**Theorem 38 (A concrete transformation is captured by some abstract one).** *Let  $P = \langle L, R \rangle$  be a transformation rule,  $G, H$  be graphs and  $m : L \rightarrow G$  be a matching such that  $G \xrightarrow{(P,m)} H$ . For any shape  $S$  and abstraction morphism  $s : G \rightarrow S$  such that  $s \circ m$  is a concrete pre-matching, there exists an abstraction morphism  $t : H \rightarrow T$ , where  $T$  is the shape such that  $S \xrightarrow{(P,s \circ m)} T$ .*

$$\begin{array}{ccccc}
 L & \xrightarrow{m} & G & \xrightarrow{s} & S \\
 & & (P,m) \downarrow & & \downarrow (P,s \circ m) \\
 & & H & \xrightarrow{t} & T
 \end{array}$$

◀

*Proof.* Consider the morphism  $t : H \rightarrow T$  defined by  $t(x) = s(x)$  for all  $x$  node or edge of  $G$ , and  $t(x) = x$  for all  $x$  in  $N^{\text{new}} \cup E^{\text{new}}$ . (It is immediate to see from the definitions of graph transformation and concrete shape transformation that  $t$  is indeed a morphism). We show that  $t$  is an abstraction morphism. As in the definition of a concrete shape transformation, we distinguish the sets of nodes  $N_{\text{concr}}$  and  $N_{\text{abstr}}$  in  $T$ , and let  $H'$  be the full<sup>4</sup> sub-graph of  $H$  with nodes  $N_G \setminus m(L)$ . By definition,  $t$  coincides with  $s$  on  $H'$  and  $t$  maps nodes of  $H'$  to nodes in  $N_{\text{abstr}}$  and edges of  $H'$  to edges whose two ends are in  $N_{\text{abstr}}$ . Also, since  $H'$  is a full sub-graph of  $G$ , the multiplicity functions of  $T$  satisfy the requirements of an abstraction morphism when their domain is restricted to  $N_{\text{abstr}}$ . For the node multiplicity function for nodes  $w \in N_{\text{concr}}$ , we know from the definition that  $\text{mult}_T^n(w) = \text{mult}_S^n(w) = 1$  and that  $t^{-1}(w)$  is a singleton set. For the edge multiplicity function  $\text{mult}_T^o(w, \mathbf{a}, C)$  (we consider only  $\text{mult}_T^o$ , by symmetry the same holds for  $\text{mult}_T^i$ ), we distinguish two cases: (i)  $w$  and  $C$  are not both in  $N_{\text{concr}}$ , and (ii)  $w$  and  $C$  are both in  $N_{\text{concr}}$ . For (i), once again pre-images of  $w$  and  $C$  coincide for  $t$  and  $s$ , and also the value of  $\text{mult}_T^o$  and  $\text{mult}_S^o$ . For (ii), remind that  $C$  is a singleton set,  $\text{mult}_T^o(w, \mathbf{a}, C)$  is the actual number of edges in the graph  $G_T$  (up to  $\mu$ ), and by definition  $t$  is an isomorphism in this concrete part.  $\square$

<sup>4</sup> By full sub-graph we mean a sub-graph defined by a subset of the nodes and *all* connecting edges.

**Theorem 39 (A concrete transformation is captured by canonical abstract transformation).** Let  $P = \langle L, R \rangle$  be a transformation rule,  $G, H$  be graphs and  $m : L \rightarrow G$  be a matching such that  $G \xrightarrow{(P,m)} H$ . Let  $S$  be the neighbourhood shape of  $G$  with corresponding neighbourhood abstraction morphism  $s : G \rightarrow S$ , and let  $T$  be the neighbourhood shape of  $H$  with corresponding neighbourhood abstraction morphism  $t : H \rightarrow T$ . Then  $S \overset{(P,f)}{\rightsquigarrow} T$  for some pre-matching  $f$ . ◀

*Proof.* By definition of abstract shape transformation, we need to show that there exist a pre-matching  $f : L \rightarrow S$ , a shape  $S'$ , a shape morphism  $\beta : S' \rightarrow S$ , and a concrete pre-matching  $c : L \rightarrow S'$  such that  $f = \beta \circ c$ , and there exists a shape morphism  $\beta' : T' \rightarrow T$ , where  $T'$  is the shape such that  $S' \xrightarrow{(P,c)} T'$ . Take  $S'$  the trivial shape of  $G$ ,  $T'$  the trivial shape of  $H$ ,  $\beta = s$ ,  $\beta' = t$ ,  $c = m$  and  $f = s \circ m$ . Then the required conditions are satisfied by hypothesis. ◻

**Theorem 40 (Concrete shape transformation vs. graph transformation).** Let  $P = \langle L, R \rangle$  be a production rule,  $S$  be a shape and  $c : L \rightarrow S$  be a concrete pre-matching satisfying the dangling edge condition. For any graph  $G$  concretisation of  $S$  with abstraction morphism  $s : G \rightarrow S$ , there exists a matching  $m : L \rightarrow G$  such that  $c = s \circ m$  and if  $H$  is the graph such that  $G \xrightarrow{P,m} H$ , then there exists an abstraction morphism  $t : H \rightarrow T$ , where  $T$  is the shape obtained by  $S \xrightarrow{P,c} T$ . ◀

*Proof.* The injective morphism  $m : L \rightarrow G$  exists due to Lemma 35. We can define  $m(v) = s^{-1} \circ c(v)$ , because  $s$  is injective on the image of  $c$ . (As a proof assume  $v_1, v_2 \in N_G$  s.t.  $s(v_1) = s(v_2)$  for some  $v \in V_L$  with  $c(v) = s(v_1)$ ). By definition of a shape, we obtain  $\text{mult}_S^n(s(v_1)) = 1$  and thus  $|s^{-1}(v_1)| = 1$  and  $v_1 = v_2$ .)

Let  $H$  be such that  $G \xrightarrow{P,m} H$ . Define the mapping  $t : H \rightarrow T$  defined by

$$t(v) = \begin{cases} v & \text{if } v \in N^{\text{new}} \\ s(v) & \text{otherwise} \end{cases}$$

and analogously on  $E_H$ . Mapping  $t$  is well-defined, because, by the definition of transformation,  $N_H = (N_G \setminus m(N^{\text{del}})) \cup N^{\text{new}}$ , and  $s$  is defined on  $N_G$ . We need to show, that  $t$  is an abstraction morphism, that is:

1.  $t$  is a morphism from  $H$  to  $T$ ;
2. for all  $v \in N_T$  it holds that  $\text{mult}_T^n(v) = |t^{-1}(v)|_\nu$ ;
3. for all  $w \in N_T$ , for all  $\mathbf{a} \in \text{Lab}$ , for all  $C \in N_T / \simeq_T$ , and for all  $v \in t^{-1}(w)$ ,

$$\text{mult}_T^o(w, \mathbf{a}, C) = |v \triangleright_H^{\mathbf{a}} (t^{-1}(C))|_\mu$$

and analogously for incoming edges multiplicities.

*ad 1.* First, we show that  $t(N_H) \subseteq N_T$ . Assume  $t(v) = v' \in t(N_H)$ . There are two cases. If  $v' \in N^{\text{new}}$ , then  $v' = v \in N^{\text{new}} \subseteq N_T$ . Otherwise,  $v' = s(v)$  for  $v \in N_G \setminus s^{-1}(c(N^{\text{del}}))$  ( $\star$ ). Assume  $v' \notin N_T$  but  $v' \in s(N_G)$ . As  $v'$  is not new, it must be the case, due to the definition of  $N_T = (N_S \setminus c(N^{\text{del}})) \cup N^{\text{new}}$ , that  $v' \in c(N^{\text{del}})$ . Hence,  $v \in s^{-1}(c(N^{\text{del}}))$ , contradicting ( $\star$ ). The case for edges is similar.

As a next step, we prove that  $t(\text{src}_H(e)) = \text{src}_T(t(e))$  for an arbitrary edge  $e \in E_H$ . First, assume  $\text{src}_H(e) \in N^{\text{new}}$  implying  $e \in E^{\text{new}}$ . We compute

$$\begin{aligned} t(\text{src}_H(e)) &= \text{src}_H(e) && \text{(Def. of } t) \\ &= \text{src}_S(e) && \text{(Def. transformation and } \text{src}_H(e) \text{ is new)} \\ &= \text{src}_T(e) && \text{(Def. shape transformation)} \\ &= \text{src}_T(t(e)) && \text{(Def. of } t) \end{aligned}$$

In the second case, we have  $\text{src}_H(e) \notin N^{\text{new}}$ , that is  $t(\text{src}_H(e)) = s(\text{src}_H(e))$  yielding another two cases depending on whether or not  $e \in E^{\text{new}}$ . If  $e$  is not new, we have

$$\begin{aligned} s(\text{src}_H(e)) &= s(\text{src}_G(e)) \\ &= \text{src}_S(s(e)) && (s \text{ morphism)} \\ &= \text{src}_T(s(e)) && \text{(Def. transformation)} \\ &= \text{src}_T(t(e)) && \text{(Def. of } t) \end{aligned}$$

If  $e$  is new, we have instead

$$\begin{aligned} s(\text{src}_H(e)) &= s(\text{src}_G(e)) \\ &= \text{src}_S(s(e)) \\ &= \text{src}_T(s(e)) \\ &= \text{src}_T(t(e)) \end{aligned}$$

The cases for edges, target and label mappings are similar.

*ad 2.* Let  $v \in N_T$  be arbitrary. If  $v \in N^{\text{new}}$ , then there is only  $\{v\} = t^{-1}(v)$  and  $\text{mult}_T^n(v) = 1$  by definition of abstract transformations. Assume  $v \notin N^{\text{new}}$ . As  $s$  is an abstraction morphism, we know that  $|s^{-1}(v)|_v = \text{mult}_S^n(v) = \text{mult}_T^n(v)$ , and it suffices to show that  $s^{-1}(v) = t^{-1}(v)$ , which is straightforward from the definition of  $t$ .

*ad 3.* This result follows immediately from the definition of  $\simeq_T$ . By definition of  $\text{mult}_T^o$ , we can either employ the fact that  $s$  is an abstraction morphism or, in case of new edges, none of them are equivalent to either themselves or anything existing before, so all new multiplicities are in fact 1, as defined. This reasoning holds both for source and target multiplicities.  $\square$

**Corollary 41 (Transformation of canonical shapes).** *Let  $P = \langle L, R \rangle$  be a transformation rule,  $S, T$  be canonical shapes and  $f : L \rightarrow S$  be a pre-matching such that  $S \xrightarrow{(P,f)} T$ . Let  $S', T'$  be the shapes,  $c : L \rightarrow S'$  the concrete pre-matching and  $\beta : S' \rightarrow S$  and  $\beta' : T' \rightarrow T$  the shape morphisms that witness  $S \xrightarrow{(P,f)} T$ . Then for any concretisation  $G$  of  $S'$  with abstraction morphism  $s : G \rightarrow S'$ , there exist a matching  $m : L \rightarrow G$  and a graph  $H$  such that  $G \xrightarrow{(P,m)} H$  and  $T$  is (isomorphic to) the neighbourhood shape of  $H$ .  $\blacktriangleleft$*

*Proof.* The matching  $m$  exists by Theorem 40. By the same theorem, we know that there exists an abstraction morphism  $t : H \rightarrow T'$ . Thus,  $\beta' \circ t$  is an abstraction morphism from  $H$  to  $T$ . We can conclude then that  $T$  is a neighbourhood shape (as it has  $H$  as concretisation). By Lemma 23,  $\beta' \circ t$  is the neighbourhood abstraction morphism of  $H$ .  $\square$

### 5.3 Using Shape Transformations

We have seen in the previous section several properties of concrete graph transformations with respect to shape transformations and abstraction morphisms. In this section we informally describe how these results can be used for over-approximating a concrete labelled transition system by an abstract one.

Consider a graph production system  $\langle G_0, \mathbf{P} \rangle$ , where  $G_0$  is the start graph and  $\mathbf{P}$  is a set of graph transformation rules. As briefly described in the introduction, this production system gives rise to a labelled transition system (LTS for short)  $\mathcal{S}$ , on which states are graphs, with start state  $G_0$ , and transitions are applications of graph transformation rules. That is, any state  $G$  of the LTS is a graph that can be derived from  $G_0$  by a final number of applications of graph transformations starting from  $G_0$ . If rule  $P = \langle L, R \rangle$  is applicable in graph  $G$  with matching  $m : L \rightarrow G$  yielding the graph  $H$ , then  $H$  is a state in the LTS and there exists a transition from  $G$  to  $H$  labelled by  $(P, m)$ . A *path starting in state  $G_1$*  in the LTS  $\mathcal{S}$  is a sequence of graph transformation rules  $P_1, \dots, P_k$  such that there exists a sequence of graphs  $G_1, \dots, G_k$  and a sequence of matchings  $m_i : L_i \rightarrow G_i$ , for all  $1 \leq i \leq k - 1$ , such that  $G_i \xrightarrow{P_i, m_i} G_{i+1}$ .

Consider now some fixed positive naturals  $i, \mu, \nu$  defining the precision of the neighbourhood abstraction. Define the LTS  $\mathcal{S}'$  whose states are canonical shapes and whose transitions are abstract shape transformations with:

- states of  $\mathcal{S}'$  are the neighbourhood shapes of states of  $\mathcal{S}$ , in their canonical representation, and initial state is  $S_0$ , the neighbourhood shape of  $G_0$ ;
- transitions of  $\mathcal{S}'$  are the transitions  $S \xrightarrow{P, f} T$  such that there exists a transition  $G \xrightarrow{P, m} H$  in  $\mathcal{S}$ , where  $s : G \rightarrow S$  and  $t : H \rightarrow T$  are the neighbourhood abstraction morphisms of  $G$  and  $H$ , respectively, and  $f = s \circ m$ .

By Theorem 39 we know that transitions in the LTS  $\mathcal{S}'$  indeed correspond to abstract graph transformations. Note also that the LTS  $\mathcal{S}'$  is finite, as there are only a finite number of canonical shapes for fixed  $i, \mu$  and  $\nu$ . Additionally, every path in  $\mathcal{S}$  starting in state  $G$  is also a path in  $\mathcal{S}'$  starting in the neighbourhood shape of  $G$ . Remark that the inverse does not hold, as every state of  $\mathcal{S}'$  may be the neighbourhood shape of several different states in  $\mathcal{S}$ . Therefore,  $\mathcal{S}'$  is a finite over-approximation of  $\mathcal{S}$  with respect to paths and can be used for verifying, e.g., temporal properties on  $\mathcal{S}$ .

Unfortunately, the LTS  $\mathcal{S}'$  cannot be constructed without constructing  $\mathcal{S}$ , which may be infinite. However, we can construct another LTS, denote it  $\mathcal{S}''$ , that is computable and still a finite over-approximation of  $\mathcal{S}$ . The idea is to start from the canonical shape  $S_0$  and construct iteratively all possible abstract transformations. For a fixed state  $S$ , the construction of its outgoing transitions in  $\mathcal{S}''$  can be done in three steps:

**Materialisation:** in order to enumerate and construct all possible abstract transformations of a canonical shape  $S$ , we first have to find and construct witnesses for these transformations (according to Definition 37), i.e., find all rules  $P = \langle L, R \rangle$  and all pre-matchings  $f : L \rightarrow S$  such that there exists a shape  $S'$  less abstract than  $S$  with shape morphism  $\beta : S' \rightarrow S$  and a concrete pre-matching  $c : L \rightarrow S'$  with  $f = \beta \circ c$ . Such shapes  $S'$  are called *materialisations* of  $S$ . Constructing the materialisations is described in Section 6.1 and Section 6.2;

**Transformation:** once we have computed all possible materialisations of the shape  $S$  w.r.t. the graph production system  $\langle G_0, \mathbf{P} \rangle$ , we can perform the actual transformations as concrete shape transformations;

**Normalisation:** applying a concrete shape transformation on some materialisation of canonical shape  $S$  does not necessarily result in a canonical shape. That is, the resulting graph may not be a state of  $\mathcal{S}''$  and therefore the result of the transformation has to be abstracted to a neighbourhood shape. This is called *normalisation* and is described in Section 6.3.

## 6 Materialisation and Normalisation

We define in this section the set of materialisations of a canonical shape  $S$  w.r.t. some pre-matching of a transformation rule. This set of materialisations is finite. In Section 6.2 we briefly describe an algorithm that allows us to construct the set of materialisations and we give some examples.

### 6.1 Definition of the Set of Materialisations

Let us first give a formal definition of what we call a materialisation. In the sequel we consider fixed naturals  $i, \mu, \nu$  defining the precision of a neighbourhood abstraction.

**Definition 42 (Materialisation).** *Given a level  $i$  canonical shape  $S$  and a rule  $P = \langle L, R \rangle$  with pre-matching  $f : L \rightarrow S$ , a materialisation of  $S$  according to  $f$  is a shape  $S'$  such that*

- $S$  is more abstract than  $S'$ , i.e., there exists a shape morphism  $\beta : S' \rightarrow S$ ;
- there exists a concrete pre-matching  $c : L \rightarrow S'$  such that  $f = \beta \circ c$ ;
- let  $T'$  be the shape resulting of the transformation of  $S'$  with  $P, c$ . Then the level  $i$  neighbourhood abstraction morphism of  $T'$  exists. ◀

For any canonical shape  $S$ , rule  $P = \langle L, R \rangle$  and pre-matching  $f : L \rightarrow S$ , we want to construct the set of materialisations  $\mathcal{M}(S, P, f)$  that covers all possible transformations of some concretisation of  $S$ . That is, for any graph  $G$ , concretisation of  $S$ , there exists a shape  $S'$  in  $\mathcal{M}(S, P, f)$  such that  $S'$  is a shape for  $G$ . This set is defined as follows (the first two points coincide with the definition of a materialisation).

**Definition 43 (Set of Materialisations  $\mathcal{M}(S, P, f)$ ).** *For a given level  $i$  canonical shape  $S$  and a rule  $P = \langle L, R \rangle$  with pre-matching  $f : L \rightarrow S$ , the set  $\mathcal{M}(S, P, f)$  is composed of the shapes  $S'$  that satisfy the following (up to shape isomorphism)*

- $S$  is more abstract than  $S'$ , i.e., there exists a shape morphism  $\beta : S' \rightarrow S$ ;
- there exists a concrete pre-matching  $c : L \rightarrow S'$  such that  $f = \beta \circ c$ ;
- let  $S''$  be the shape obtained from  $S'$  as follows: to every node  $v$  in  $c(L)$  of  $S'$  is given an additional, fresh label  $l_v$ . Then shape  $S''$  is a canonical shape. ◀

Elements of the set  $\mathcal{M}(S, P, f)$  are indeed materialisations. The point on which we have to argue is that after transformation, a shape  $S'$  in  $\mathcal{M}(S, P, f)$  admits a level  $i$  neighbourhood shape.

**Lemma 44.** *Let  $S'$  be a shape in  $\mathcal{M}(S, P, f)$ . Then the shape  $T'$  resulting from the transformation of  $S'$  by  $P, c$  admits a level  $i$  neighbourhood shape. ◀*

*Proof. (Sketch)* Let  $S''$  be the shape that witnesses the fact that  $S'$  is a materialisation of  $S$ ; that is,  $S''$  is the same as  $S'$  except that it has fresh labels on the nodes in  $c(L)$ . Consider also the rule  $P'' = \langle L'', R'' \rangle$  obtained from  $P$  by adding fresh labels to all nodes in a way that  $c : L'' \rightarrow S''$  is a concrete matching. That is, fresh labels for  $L''$  and  $c(L)$  in  $S''$  coincide. Then rule  $P''$  can be applied to  $S''$  with matching  $c$ , thus obtaining the graph  $T''$ . It is not difficult to see that shape  $T'$  is like  $T''$  from which the fresh labels have been removed. Then one can show that:

1. if  $T''$  admits a level  $i$  neighbourhood shape, then also does  $T'$ . This is shown in a more general way for a shape  $T'$  obtained from a shape  $T''$  by removing some unique labels. The proof of this result is quite technical and is given in Appendix F;
2. for all  $j \leq i$ ,  $\sim_j$  is defined in  $T''$  and moreover for all node  $v \in T'' \cap S''$ ,  $[v]_{\sim_j}$  in  $S''$  is included into  $[v]_{\sim_j}$  in  $T''$ , whenever this former exists (i.e., whenever  $v \notin N^{\text{new}}$ ).

These two points allow us to conclude that  $T'$  admits a level  $i$  neighbourhood shape. In what follows we sketch a proof for the latter statement. Let us first point out that if  $\sim_j$  is defined on  $T''$ , then  $[v]_{\sim_j} = \{v\}$  in  $T''$  and in  $S''$  for all node  $v$  in  $c(L'') \cup N^{\text{new}}$  because  $v$  has a unique label, and also  $[v]_{\sim_{T''}} = \{v\}$  by definition. Thus, we only have to care about nodes  $v$  not in  $c(L'') \cup N^{\text{new}}$ . Moreover, by definition the grouping relations of  $S''$  and  $T''$  coincide on all nodes in  $N_{S''}$ , the fact that  $\sim_j$  is defined is not a problem as long as  $[v]_{\sim_{j-1}}$  in  $S''$  is included into  $[v]_{\sim_{j-1}}$  in  $T''$ . So let us simply suppose that  $\sim_j$  is defined and argue that if  $v \sim_j v'$  in  $S''$ , then  $v \sim_j v'$  in  $T''$ . Remark that the unique labels in  $c(L'')$  influence the equivalence classes for  $\sim_j$  of the nodes that are in the  $j$ -neighbourhood of  $c(L'')$ . In other words, if  $v \sim_j v'$  in  $S''$ , then either  $v$  and  $v'$  are both far away from  $c(L'')$ , or are both at the same distance from all nodes in  $c(L'')$ . In the first case, it is clear that they are also far away from the nodes  $c(L'') \cup N^{\text{new}}$  in  $T''$  so they remain  $\sim_j$ -equivalent in  $T''$ . In the second case, intuitively  $v$  and  $v'$  are connected exactly in the same way to all the nodes  $c(L'')$ , this is because of the uniqueness of labels of these latter. Now if, e.g.,  $v$  is in the  $j$ -neighbourhood of some of the newly added nodes from  $N^{\text{new}}$ , and thus “influenced” by this new node for its  $\sim_j$  equivalence class, then  $v'$  is influenced in exactly the same way because nodes in  $N^{\text{new}}$  are only connected to nodes in  $c(L'')$ , and because of uniqueness of labels.  $\square$

Remark that the set  $\mathcal{M}(S, P, f)$  is finite. Indeed, it is a set of canonical shapes over the initial set of labels augmented with the fresh labels  $l_v$ , for  $v$  in  $c(L)$ , and the number of different such canonical shapes is finite.

**Lemma 45 (Completeness of the Set of Materialisations).** *Let  $S$  be a neighbourhood shape. For any concretisation  $G$  of  $S$  with corresponding neighbourhood abstraction morphism  $s : G \rightarrow S$ , for a rule  $P = \langle L, R \rangle$ , and for any match  $m : L \rightarrow G$ , there exist a pre-matching  $f : L \rightarrow S$  and a shape  $S'$  in  $\mathcal{M}(S, P, f)$  such that  $f = s \circ m$  and  $H$  abstracts to  $T'$ , where  $H$  and  $T'$  are the graph and the shape such that  $G \xrightarrow{P, m} H$  and  $S' \xrightarrow{P, f} T'$ .  $\blacktriangleleft$*

*Proof.* It is immediate to see that if  $m : L \rightarrow G$  is a matching, then there exists a pre-matching  $f : L \rightarrow S$  such that  $f = s \circ m$ . This holds for all abstraction morphisms  $s : G \rightarrow S$ , and not only for neighbourhood abstraction morphisms. Consider now the set  $\mathcal{M}'(S, P, f)$  as the set of all shapes  $S'$  defined by the first two conditions for  $\mathcal{M}(S, P, f)$ . That is,  $\mathcal{M}'(S, P, f)$  is a possibly infinite over-set of  $\mathcal{M}(S, P, f)$ . In particular, any graph  $G$ , concretisation of  $S$ , is in  $\mathcal{M}'(S, P, f)$  as its trivial shape. In other words,  $\mathcal{M}'(S, P, f)$  is complete in the sense

of the lemma. It is then enough to show that for all shape  $G$  in  $\mathcal{M}'(S, P, f)$ , there exists a shape  $S'$  in  $\mathcal{M}(S, P, f)$  such that  $S'$  is more abstract than  $G$ . This corresponds to showing that the third condition in the definition of  $\mathcal{M}(S, P, f)$  does not remove too much graphs and shapes from the set  $\mathcal{M}'(S, P, f)$ . This is indeed the case because this third condition ensures that materialisations are neither too abstract nor too concrete, but correspond to the level of abstraction of a neighbourhood shape morphism.  $\square$

**Proposition 46 (Minimality of the Set of Materialisations).** *For any concretisation  $G$  of  $S$  with abstraction morphism  $s : G \rightarrow S$ , and any match  $m : L \rightarrow P$  such that  $f = s \circ m$ , there exists a unique shape  $S'$  in  $\mathcal{M}(S, P, f)$  such that  $G$  can be abstracted to  $S'$  with abstraction morphism  $s : G \rightarrow S'$ , and such that  $c = s \circ m$ , where  $c : L \rightarrow S'$  is the concrete pre-matching extracted from  $f$ .  $\blacktriangleleft$*

*Proof.* Let  $S_1$  and  $S_2$  be two shapes in  $\mathcal{M}(S, P, f)$  which both satisfy the conditions of the proposition, with abstraction morphisms  $s_1 : G \rightarrow S_1$  and  $s_2 : G \rightarrow S_2$ , and with concrete pre-matchings  $c_1 : L \rightarrow S_1$  and  $c_2 : L \rightarrow S_2$ . Consider now the canonical shapes  $S'_1$  and  $S'_2$  that witness the fact that  $S_1$  and  $S_2$  are materialisations (according to the third condition in the definition of  $\mathcal{M}(S, P, f)$ .) Consider also the graph  $G'$  obtained from  $G$  by adding the fresh label  $l_v$  to the node  $m(v)$ , this for all  $v$  node of  $L$ . Then it is easy to see that  $s_1 : G' \rightarrow S'_1$  and  $s_2 : G' \rightarrow S'_2$  are abstraction morphisms. Moreover, as  $S_1$  and  $S_2$  are canonical shapes, then necessarily  $s_1$  and  $s_2$  are canonical morphisms. As each graph has a unique neighbourhood shape, necessarily  $S'_1$  and  $S'_2$  are the same canonical shape. By definition of  $S'_1$  and  $S'_2$  it immediately follows that  $S_1$  and  $S_2$  are the same shape, since elements of  $\mathcal{M}(S, P, f)$  are unique up to isomorphism, by definition.  $\square$

## 6.2 Effective Construction of $\mathcal{M}$

In order to effectively construct an abstract labelled transition system, one needs to be able to effectively construct the set of materialisations  $\mathcal{M}(S, P, f)$  for a canonical shape  $S$  and a pre-matching  $f : L \rightarrow S$  for rule  $P = \langle L, R \rangle$ . We give here an algorithm for constructing the set of materialisations.

Intuitively, a materialisation is composed of an abstract part and a materialised part. The abstract part is the initial shape  $S$  or sub-graphs of it. The materialised part is composed of a concrete copy of  $f(L)$  and its neighbourhood of radius  $i$ , where  $i$  is the level of neighbourhood shape morphism. The main idea of the algorithm is to “extract” a concrete copy of  $f(L)$  from  $S$ , remap the matching  $f$  into this concrete part yielding a concrete pre-matching  $c$ , and then modify the obtained structure until it becomes a correct materialisation. Remind that the structure is a materialisation if one can attach fresh names to the nodes in  $c(L)$  and obtain a neighbourhood shape. This intuitively means that in radius  $i$  from the concrete part  $c(L)$ , two nodes of any concrete graph may be grouped together only if they are connected in exactly the same way to all nodes from  $c(L)$  (up to edge multiplicities).

The algorithm starts from the shape  $S$  and iteratively constructs structures that are a kind of pre-materialisations and refines these until they become correct materialisations. This is done by iterating over the following steps:

**extract and connect** for the first iteration, “extract” a concrete copy of  $f(L)$  from the shape, remap  $f$  into this concrete copy yielding a concrete pre-matching  $c(L)$  and associate fresh labels to the nodes in  $c(L)$ . This copy becomes the materialised part that will be widened by

adding new nodes to it during the next iterations. For the second and next iterations, “pull” along the edges that connect the materialised part and the abstract part for extracting new nodes. These nodes become part of the materialised part. Any of these extractions is accompanied by connecting all newly extracted nodes with the abstract part in all possible ways and updating node and edge multiplicities;

**update grouping relation** the grouping relation is updated so that any node that is at distance less than  $i - 1$  from the concrete part  $c(L)$  becomes alone in its equivalence class for the grouping relation. This is necessary because the fresh labels in  $c(L)$  influence the  $\sim_{i-1}$  equivalence relation for these nodes, which should be equal to the grouping relation (in a neighbourhood shape);

**choose nodes and edges** as the previous step acts on the grouping relation by splitting groups, edges that previously had all their start (or end) points in the same group may not be grouped anymore, but still they have an associated common edge multiplicity. The algorithm splits these multiplicity functions in all possible ways so that edges in the materialised part have correct edges multiplicities. This is not done for edges in the abstract part (as it is not always possible).

### 6.3 Normalisation

Applying a shape transformation to a materialised shape does not yield a canonical shape. The role of normalisation is to construct the neighbourhood shape of such unnormalised shapes. Let  $S'$  be a materialisation in  $\mathcal{M}(S, P, f)$ , and  $T'$  be the shape resulting from the concrete shape transformation  $S' \xrightarrow{(P,c)} T'$ , where  $c$  is the concrete pre-matching corresponding to  $f$ . By definition, we know that  $T'$  admits a level  $i$  neighbourhood shape, denote it  $T$ .

### 6.4 Back to the Construction of the Abstract Labelled Transition System

Now all the ingredients for constructing an abstract labelled transition system  $\mathcal{S}''$  (ALTS) are given. Consider the neighbourhood shape  $S_0$  and set of transformation rules  $\mathbf{P}$ . Initially,  $S_0$  is the unique state of the ALTS, and there are no transitions. For any state  $S$  in the ALTS, and for any rule  $P = \langle L, R \rangle$ , we compute all pre-matchings  $f : L \rightarrow S$ . For any pre-matching  $f$ , the set of materialisations  $\mathcal{M}(S, P, f)$  is computed. For all materialisations  $S'$  in  $\mathcal{M}(S, P, f)$ , the actual concrete shape transformation is performed  $S' \xrightarrow{(P,c)} T'$ , where  $c$  is the concrete pre-matching  $c : L \rightarrow S'$  deduced from  $f$ . Finally, the neighbourhood shape  $T$  of  $T'$  is computed. If  $T$  is not a state of the ALTS, then it is added as a state. Then a transition from  $S$  to  $T$  with label  $P, f$  is added to the ALTS.

Note that the ALTS is non deterministic (a state may have several outgoing transitions with the same label), whereas a concrete LTS is always deterministic. This is because one state can have several materialisations for a fixed rule and a fixed pre-matching. In the concrete case, a rule with a matching uniquely define an application of a graph transformation and its result.

The ALTS  $\mathcal{S}''$  constructed this way is an over-approximation of all (concrete) LTS  $\mathcal{S}$  with start graph  $G_0$  and set of rules  $\mathbf{P}$ , for all graphs  $G_0$  with neighbourhood shape  $S_0$ , in the following sense:

For any path  $G_1 \xrightarrow{P_1, m_1} G_2 \xrightarrow{P_2, m_2} \dots \xrightarrow{P_{n-1}, m_{n-1}} G_n$  in  $\mathcal{S}$ , where  $G_i$  are states of  $\mathcal{S}$ ,  $P_i = \langle L_i, R_i \rangle$  are transformation rules and for all  $1 \leq i \leq n - 1$ ,  $m_i : L_i \rightarrow G_i$  are



matchings, there exists a unique path  $S_1 \xrightarrow{P_1, f_1} S_2 \xrightarrow{P_2, f_2} \dots \xrightarrow{P_{n-1}, m_{n-1}} S_n$  in  $\mathcal{S}''$ , where for all  $i$ ,  $S_i$  is the neighbourhood shape of  $G_i$  with corresponding abstraction morphisms  $s_i : G_i \rightarrow S_i$ , and  $f_i = s_i \circ m_i$ .

To show that this property indeed holds, we need to show the following.

1. For any concrete transition  $G \xrightarrow{P, m} H$ , there exists an abstract transition  $S \xrightarrow{P, f} T$ , where  $S$  and  $T$  are the neighbourhood shapes of  $G$  and  $H$ , respectively, and  $f = s \circ m$  for  $s : G \rightarrow S$  the neighbourhood abstraction of  $G$ . This is ensured by Theorem 39.
2. This abstract transition is indeed computed and added as a transition of  $\mathcal{S}''$ . That is, show that a witness for this abstract transformation exists in the set  $\mathcal{M}(S, P, f)$ . This is ensured by the completeness of the set of materialisations (Lemma 45) and by the composition of neighbourhood morphisms for graphs and shapes (Lemma 22). By completeness of the set of materialisations we know that there exists a shape  $S'$  in  $\mathcal{M}(S, P, f)$  that is an abstraction for  $G$ , and that can be transformed to simulate the actual transformation of  $G$ ; let the shape resulting from the transformation be  $T'$ . By composition of neighbourhood shape morphisms, we know that the normalisation of  $T'$  yields the neighbourhood shape of  $H$ .
3. Uniqueness of the path  $S_1 \xrightarrow{P_1, f_1} S_2 \xrightarrow{P_2, f_2} \dots \xrightarrow{P_{n-1}, m_{n-1}} S_n$  is ensured by uniqueness of neighbourhood shapes.

## 7 A Modal Logic for Graphs and Shapes

In this section we define a modal logic with forward and backward modalities and counting that can be interpreted on graphs and on shapes. We show that this logic is preserved and reflected by abstraction and shape morphisms.

Before presenting a formal definition of syntax and semantics of this logic, let us give a flavour of the logic with some examples.

*Example 47.* Consider the graph depicted on Figure 1, representing a list structure. Here are some properties that one could want to express for such structure:

1. any cell has an associated value that is some object, *i.e.*, any **Cell**-node has an outgoing **val**-edge leading to an **Object**-node. This can be expressed by the following formula:

$$\text{Cell} \rightarrow \langle \text{val} \rangle^1 \cdot \text{Object}.$$

The  $\langle \text{val} \rangle^1$  operator is a forward existential modality, indicating the existence of an outgoing **val** edge. With the modality is associated a multiplicity, here 1, which is interpreted as “at least one” outgoing **val**-edge;

2. analogously, any object is the value associated to some cell, *i.e.*, any **Object**-node has an incoming **val**-edge coming from a **Cell**-node. This is expressed by the formula

$$\text{Object} \rightarrow \langle \text{val} \rangle^1 \cdot \text{Cell}.$$

Here,  $\langle \text{val} \rangle^1$  is a backward modality and indicates the existence of at least one incoming **val**-edge;

3. we can go further and express that objects are not shared between different list cells, *i.e.*, every **Object**-node has exactly one incoming **val**-edge coming from a **Cell**-node:

$$\text{Object} \rightarrow ( \langle \text{val} \rangle^1 \cdot \text{Cell} \wedge \neg \langle \text{val} \rangle^2 \cdot \text{Cell} ).$$

Here,  $\neg$  is the negation operator and  $\wedge$  is conjunction. ◀

## 7.1 Syntax of the Logic

Consider a finite set of atomic propositions  $\mathcal{P}$ . A  $\mathcal{L}(\mathcal{P})$  logic formula  $\phi$  is defined by the following syntax:

$$\phi ::= \mathbf{t} \mid \mathbf{p} \mid \neg\phi \mid \phi \vee \phi' \mid \phi \wedge \phi' \mid \mathbf{a}\rangle^\lambda \cdot \phi \mid \langle \mathbf{a}\langle^\lambda \cdot \phi$$

where  $\mathbf{a}$  is a label in  $\text{Lab}$ ,  $\mathbf{p} \in \mathcal{P}$  and  $\lambda$  is an element of  $\mathbf{M}_\mu$ ;  $\mathbf{t}$  stands for the true formula,  $\mathbf{a}\rangle^\lambda \cdot \phi$  and  $\langle \mathbf{a}\langle^\lambda \cdot \phi$  are forward and backward existential modalities, respectively, and  $\neg$ ,  $\vee$  and  $\wedge$  are the usual logical operators<sup>5</sup>.

In Example 47, we used labels as atomic propositions, *i.e.*, the formulae in this example are  $\mathcal{L}(\text{Lab})$  formulae.

The *nesting depth*  $d(\phi)$  of a logic formula  $\phi$  measures the maximal number of nested modalities. It is defined recursively on the structure of  $\phi$  as :  $d(\mathbf{p}) = d(\mathbf{t}) = 0$ ,  $d(\mathbf{a}\rangle^\lambda \cdot \phi) = d(\langle \mathbf{a}\langle^\lambda \cdot \phi) = 1 + d(\phi)$ ,  $d(\neg\phi) = d(\phi)$ ,  $d(\phi \vee \phi') = d(\phi \wedge \phi') = \max(d(\phi), d(\phi'))$  for any  $\mathbf{a}$  in  $\text{Lab}$ . We denote  $\mathcal{L}_i(\mathcal{P})$  the set of logic formulae with nesting depth at most  $i$ .

## 7.2 Satisfaction on Graphs and Shapes

Logic formulae are interpreted in graph nodes. Let  $G$  be a graph and  $\gamma : N_G \rightarrow 2^{\mathcal{P}}$  be a valuation function that associates a set of atomic propositions with any node of  $G$ . For a graph  $G$ , a node  $v$  in  $N_G$ , a valuation  $\gamma$ , and a formula  $\phi$ , the *satisfaction relation*  $G, v, \gamma \models \phi$  is defined recursively on the structure of  $\phi$  by:

- $G, v, \gamma \models \mathbf{t}$ ;
- $G, v, \gamma \models \mathbf{p}$  if  $\mathbf{p} \in \gamma(v)$ ;
- $G, v, \gamma \models \neg\phi$  if  $G, v, \gamma \not\models \phi$ ;
- $G, v, \gamma \models \phi \vee \phi'$  if  $G, v, \gamma \models \phi$  or  $G, v, \gamma \models \phi'$ ;
- $G, v, \gamma \models \phi \wedge \phi'$  if  $G, v, \gamma \models \phi$  and  $G, v, \gamma \models \phi'$ ;
- $G, v, \gamma \models \mathbf{a}\rangle^\lambda \cdot \phi$  if  $|\{e \in v \triangleright^{\mathbf{a}} \mid G, \text{tgt}(e), \gamma \models \phi\}|_\mu \geq \lambda$ ;
- $G, v, \gamma \models \langle \mathbf{a}\langle^\lambda \cdot \phi$  if  $|\{e \in v \triangleleft^{\mathbf{a}} \mid G, \text{src}(e), \gamma \models \phi\}|_\mu \geq \lambda$ .

If  $G, v, \gamma \models \phi$ , we say that  $\phi$  *holds in node*  $v$ . We sometimes omit  $\gamma$  if it is clear from the context. Intuitively, a formula of the form  $\mathbf{a}\rangle^\lambda \cdot \phi$  holds in a node  $v$  if the  $-\mu$ -bounded- number of  $\mathbf{a}$ -labelled edges ( $e$ ) connecting it to some node  $v'$  ( $\text{src}_G(e) = v$  and  $\text{tgt}_G(e) = v'$ ) in which  $\phi$  holds is at least  $\lambda$ . Analogously,  $\langle \mathbf{a}\langle^\lambda \cdot \phi$  holds in  $v$  if the number of  $\mathbf{a}$ -labelled edges connecting some  $v'$  to  $v$  is at least  $\lambda$ .

Back to Example 47 with the definition of satisfaction of the logic in mind, one can notice that in this example the valuation  $\gamma$  is not specified. As we pointed out, the formulae in this example are in  $\mathcal{L}(\mathcal{P})$ . A natural valuation for this logic is the one that associates to each node the set of its labels; *i.e.*,  $\gamma(v) = \text{lab}_G(v)$  for all  $v$  in  $N_G$ , and for any graph  $G$ .

The satisfaction relation is defined for a shape almost in the same manner as it is defined for a graph. The differences are in the way it is defined for a modality formula:

- $S, v, \gamma \models \mathbf{a}\rangle^\lambda \cdot \phi$  if  $\lambda \leq \sum_{C \in X} \text{mult}_S^\circ(v, \mathbf{a}, C)$  where

$$X = \{C \in N_S / \simeq_S \mid \forall w \in C. S, w, \gamma \models \phi\};$$

<sup>5</sup> We explicitly add here the redundant operators  $\mathbf{t}$  and  $\wedge$  because later we will be interested in the logic without negation, which in this case can simply be defined as a syntactical fragment.

–  $S, v, \gamma \models \langle \mathbf{a} \rangle^\lambda \cdot \phi$  if  $\lambda \leq \sum_{C \in X} \text{mult}_S^i(v, \mathbf{a}, C)$  where:

$$X = \{C \in N_S / \simeq_S \mid \forall w \in C. S, w, \gamma \models \phi\}.$$

In the case of shapes, formulae are interpreted a bit differently. We no longer count individual  $\mathbf{a}$ -labelled edges going out of (resp. coming into) a node, but instead sum-up the outgoing (resp. incoming) edge multiplicities associated to that node and to a group of nodes  $C$  such that all nodes in  $C$  satisfy the formula  $\phi$ .

*Example 48.* Back to our list example, the formula  $\text{Cell} \rightarrow \langle \text{Next} \rangle^1 \cdot (\text{List} \vee \text{Cell})$  holds on all nodes of the shape on Figure 5. ◀

### 7.3 Preservation by Abstraction Morphism

Let  $s : G \rightarrow S$  be an abstraction morphism from graph  $G$  to shape  $S$ . We say that  $s$  *preserves* a property  $p$  if whenever  $p$  holds in node  $v$  of  $G$ , it also holds in node  $s(v)$  of  $S$ . Inversely, we say that  $s$  *reflects*  $p$  if whenever  $p$  holds in node  $s(v)$  of  $S$ , it also holds in node  $v$  of  $G$ . One can also talk about preservation and reflection by a shape morphism  $\alpha : S \rightarrow T$ .

Preservation and reflection are very important characteristics. If an abstraction preserves a set of properties, these properties can be verified on the abstract level. If an abstraction reflects a set of properties, then for any characterisation of a shape, the properties also hold for concretisations of the shape. If both preservation and reflection hold, verifying a property on a graph is equivalent to verifying it on the abstract level.

As shown in the next section, neighbourhood abstraction preserves and reflects all properties defined by logic formulae of the corresponding depth. We start in this section with a more general result about preservation and reflection by abstraction, identifying the necessary conditions for it to hold.

In Definition 49 we define what we mean by preservation and reflection of a property, in the most general case. In Proposition 50 we show the result for preservation and reflection for shape morphisms, and in Proposition 52 for abstraction morphisms.

**Definition 49.** Let  $\mathcal{P}$  be a set of atomic propositions,  $S, T$  be shapes,  $\gamma_S : N_S \rightarrow 2^{\mathcal{P}}$  and  $\gamma_T : N_T \rightarrow 2^{\mathcal{P}}$  be valuations, and let  $\mathcal{R}$  be a set of properties such that the satisfaction relations  $S, v, \gamma_S \models p$  and  $T, w, \gamma_T \models p$  are defined for any nodes  $v \in N_S, w \in N_T$  and for any property  $p \in \mathcal{R}$ . We say that  $\alpha : S \rightarrow T$  preserves  $\mathcal{R}$  under  $\gamma_S, \gamma_T$  if for any  $p \in \mathcal{R}$  and for any  $v \in N_S$  we have  $S, v, \gamma_S \models p$  implies  $T, \alpha(v), \gamma_T \models p$ . We say that  $\alpha$  reflects  $\mathcal{R}$  under  $\gamma_S, \gamma_T$  if for any  $p \in \mathcal{R}$  and for any  $v \in N_T$  we have  $T, v, \gamma_T \models p$  implies  $S, w, \gamma_S \models p$  for any  $w \in \alpha^{-1}(v)$ . ◀

In the following we show how, under some conditions on the relationship between  $\alpha, \simeq_T$  and  $\gamma_T$ , the satisfaction of logic formulae of depth one is preserved and/or reflected by  $\alpha$ . For a shape  $T$  and a valuation  $\gamma : N_T \rightarrow 2^{\mathcal{P}}$  we say that  $\simeq_T$  is compatible with  $\gamma$  if for any two nodes  $v, w$  of  $T$ , if  $v \simeq_T w$ , then  $\gamma(v) = \gamma(w)$ . The negation free fragment of  $\mathcal{L}_1(\mathcal{P})$  is the set of  $\mathcal{L}_1(\mathcal{P})$  formulae that do not use the negation operator ( $\neg$ ).

**Proposition 50 (Preservation and Reflection).** Let  $\mathcal{P}$  be a set of atomic propositions,  $S, T$  be shapes,  $\gamma_S : N_S \rightarrow 2^{\mathcal{P}}$  and  $\gamma_T : N_T \rightarrow 2^{\mathcal{P}}$  be valuation functions such that  $\simeq_T$  is compatible with  $\gamma_T$ , and let  $\alpha : S \rightarrow T$  be a shape morphism.

**(preservation)** If  $\alpha$  preserves  $\mathcal{P}$  under  $\gamma_S, \gamma_T$ , then  $\alpha$  preserves the negation free fragment of  $\mathcal{L}_1(\mathcal{P})$  under  $\gamma_S, \gamma_T$ .

**(reflection)** If  $\alpha$  reflects  $\mathcal{P}$  under  $\gamma_S, \gamma_T$ , then  $\alpha$  reflects the negation free fragment of  $\mathcal{L}_1(\mathcal{P})$  under  $\gamma_S, \gamma_T$ .

**(preservation and reflection)** <sup>6</sup> If  $\alpha$  preserves and reflects  $\mathcal{P}$  under  $\gamma_S, \gamma_T$ , then  $\alpha$  preserves and reflects  $\mathcal{L}_1(\mathcal{P})$  (possibly with negation) under  $\gamma_S, \gamma_T$ . ◀

*Proof.* See Appendix E. ◻

This preservation and reflection result can easily be extended to abstraction morphisms (i.e., for  $S$  being a graph and  $\alpha$  an abstraction morphism in the previous proposition). One can define preservation and reflection on the same way for abstraction morphisms as for shape morphisms, as well as the notion of a grouping relation compatible with an abstraction morphism. We only enunciate the result without explicitly giving the definitions.

**Proposition 51.** Let  $\mathcal{P}$  be a set of atomic propositions,  $G$  be a graph,  $S$  be a shape and  $s : G \rightarrow S$  be an abstraction morphism compatible with  $\mathcal{P}$ . ◀

**(preservation)** If  $s$  preserves  $\mathcal{P}$ , then  $s$  preserves any negation-free  $\mathcal{M}(\mathcal{P})$  formula  $\phi$ .

**(reflection)** If  $s$  reflects  $\mathcal{P}$ , then  $s$  reflects any negation-free  $\mathcal{M}(\mathcal{P})$  formula  $\phi$ .

**(preservation and reflection)** If  $s$  preserves and reflects  $\mathcal{P}$ , then  $s$  preserves and reflects any  $\mathcal{M}(\mathcal{P})$  formula  $\phi$ . ◀

*Proof.* It is easy to show for the trivial morphism  $t_G$ . Then the result follows from the composition of shape morphisms and Proposition 50. ◻

## 7.4 Preservation and Reflection for Neighbourhood Shaping

The neighbourhood abstraction enjoys the good properties of preservation and reflection of  $\mathcal{L}(\text{Lab})$  formulae with the appropriate depth.

**Proposition 52 (Preservation and Reflection).** Let  $G$  be a graph and  $S$  be a shape obtained by the level  $i$  neighbourhood abstraction of  $G$ , for some  $i \geq 1$ , with corresponding abstraction morphism  $s : G \rightarrow S$ . Then  $s$  preserves and reflects  $\mathcal{L}_i(\text{Lab})$ . ◀

*Proof. (Sketch)* The proof goes by induction on  $i$ , using Proposition 50 and the fact that  $\mathcal{L}_{i+1}(\text{Lab})$  is equivalent to  $\mathcal{L}_1(\mathcal{R})$  where  $\mathcal{R}$  is the a set properties defined by  $\mathcal{L}_i(\text{Lab})$ . ◻

Thus, for any property  $\phi$  of  $\mathcal{L}(\text{Lab})$  to be verified on a graph  $G$ , one can use the level  $i$  neighbourhood shape of  $G$  for verifying  $\phi$ , where  $i$  is the nesting depth of the formula  $\phi$ . This means that the neighbourhood abstraction provides a graph abstraction mechanism that is parametrised by the properties we want to verify, and that guarantees preservation and reflection of these properties.

<sup>6</sup> Preservation for formulae with negation may seem contradictory with the Morphism Preservation Theorem for finite structures [13]. This theorem states that a first-order formula is preserved by a morphism if, and only if, it is equivalent to an existential positive formula. Some modal logic formulae cannot be expressed in first-order logic without negation (e.g.,  $\neg \mathbf{a}^\lambda \cdot \mathbf{t}$ .) However, in our case, shapes contain information on interpretation of such negated formulae, by means of the multiplicity functions, which explains this apparent contradiction.

*Example 53.* Denote  $G$  the graph on Figure 1 and  $S$  its level one neighbourhood shape (Figure 6), and let  $s : G \rightarrow S$  be the corresponding neighbourhood abstraction morphism. Let  $\phi$  be the formula of nesting depth two  $\text{>next}^1 \cdot \text{>next}^1 \cdot \mathbf{tt}$  (which intuitively expresses that there is a next-path of length two starting from the node). We have that  $S, w_3 \models \phi$ , but  $G, v_4 \not\models \phi$ , and  $s(v_4) = w_3$ . That is,  $\phi$  is not reflected by  $s$ . Consider now  $T$  the level two neighbourhood shape of  $G$ , depicted on Figure 7; the corresponding morphism  $t : G \rightarrow T$  is not difficult to define. One can easily check that the formula  $\phi$  is reflected by  $t$ . ◀

## 7.5 Relationship between the Logic and Neighbourhood Shaping

The modal logic that we presented in this section is tightly related to the neighbourhood equivalence relation and canonical names. We already stated in Lemma 27 that two nodes of a graph are neighbourhood equivalent if, and only if, they have the same canonical name. Here we enhance this characterisation with the logic: two nodes are  $i$ -neighbourhood equivalent if, and only if, they satisfy the same logic formulae of depth  $i$ .

**Lemma 54.** *Two nodes  $v, v'$  of a graph  $G$  are  $i$ -neighbourhood equivalent if, and only if, the same  $\mathcal{L}_i(\text{Lab})$  formulae hold in  $v$  and in  $v'$ .* ◀

*Proof.* See Appendix G. ◻

The following relationship between canonical names and logic formulae is also easy to see: to any canonical name corresponds a logic formula that holds exactly in the nodes having this name.

**Lemma 55.** *For any  $i \geq 1$  and any level  $i$  canonical node name  $C$ , there exists an  $\mathcal{L}_i(\text{Lab})$  formula  $\phi_C$  such that for any graph  $G$  and any node  $v$  of  $G$ ,  $\text{name}^i(G)(v) = C$  if, and only if,  $G, v \models \phi_C$ .* ◀

*Proof. (Idea)* The formula  $\phi_C$  can be effectively constructed by induction on  $i$ . ◻

## 8 Related Work

*Abstract Graph Transformations.* In [9] one of the authors defined a notion of abstract graphs in which abstract graph nodes may summarise an unbounded number of concrete graph nodes. These abstractions are only used for deterministic simple graphs. With the abstract graph are associated constraints on the multiplicities of incoming and outgoing edges for the nodes in a concrete instance. In the same paper are also introduced canonical abstract graphs whose size is bounded and that roughly corresponds to the level one neighbourhood abstraction in the present work. In [11] these canonical abstract graphs are used for transformations.

In [2] are introduced the so called Partner Graph Grammars, suitable formalism for dynamic communication systems. They come with an abstraction mechanism on graphs and an adequate notion of abstract graph transformations. Preservation and reflection are shown for first-order logic without equality and an interesting subclass of abstraction morphisms. Moreover, a CTL-based logic on labelled transition systems is shown to be invariant under abstraction.

*Shape Analysis and 3-Valued Logic.* The work of Sagiv *et al.* on shape analysis [14,15] has resulted in different abstraction mechanisms allowing to finitely abstract structures of unbounded size. In [15] is presented an abstraction framework that can be parametrised by the properties to be preserved; the framework is implemented in the TVLA tool [8,16]. In this work, the authors use logical structures to represent memory states of programs; abstract structures are 3-valued logical structures. Properties on these structures are defined using first-order logic with transitive closure (FO+TC). Dynamics of systems are encoded by updating the sets of predicates associated to the (abstract or concrete) structure. As graphs are logical structures and our modal logic can be encoded into first-order logic, the abstraction mechanism proposed in [15] is more general than ours. Concerning preservation of logical properties, Sagiv’s “embedding theorem” states that any information extracted from an abstract structure via a FO+TC formula  $\phi$  is a conservative approximation of the information extracted from the concrete structure via  $\phi$ . In this sense, our preservation and reflection result is more general than the embedding theorem, but holds for a modal logic and abstraction mechanism that are weaker than FO+TC and abstraction using abstraction predicates. We believe that the benefits of our approach come from the possibility of full automation. A set of graph transformation rules that is given as concrete semantics can be used as it is for the abstract semantics. Moreover, we guarantee to preserve the precision defined by depth  $i$  logic formulae, where  $i$  is the level of abstraction. In TVLA, it may be necessary to define “by-hand” some update-predicates in order to guarantee the required precision. Complexity issues apart, our framework should be easy to integrate into a graph transformation tool such as GROOVE [7].

*Abstract Regular Model Checking.* In regular model checking (see *e.g.*, [5,1]), states of programs are represented as words or trees on finite alphabets, and dynamics are modelled as word or tree transduction. Initial states of a system are represented by a regular (word or tree) automaton  $\text{Init}$ , and bad configurations by a regular automaton  $\text{Bad}$ . Checking whether a bad configuration is reached consists in testing whether the set  $\tau^*(\text{Lang}(\text{Init})) \cap \text{Lang}(\text{Bad})$  is empty, where  $\tau$  is the transduction, and  $\tau^*$  designates a repetition of this transduction. In general the problem is not decidable, as  $\tau^*(\text{Lang}(\text{Init}))$  may not be computable in a finite number of steps. Abstract regular (word or tree) model checking [4,3] proposes a method for over-approximating the set of reachable states  $\tau^*(\text{Lang}(\text{Init}))$  by a set of the form  $\text{Lang}(\tau_\alpha^*(\text{Init}))$ , where  $\alpha$  is an appropriate abstraction function for automata, and  $\tau_\alpha$  is intuitively the (tree or word) transducer  $\tau$  lifted to automata. That is,  $\tau_\alpha$  allows to apply transformations on sets of trees or words, and a parallel may be made with transformation of abstract graphs which actually aims to over-approximate transformations of the set of their concretisations.

*Logic Based Approaches.* Not so closely related but still relevant are several methods for modelling program states – or the memory heap – by relational structures, and operations by instructions modifying these relational structures; these can be qualified as logic-based approaches. For example, in [17] is introduced a fragment of first-order logic with transitive closure for expressing properties on linked data structures, represented as graphs. The logic allows to specify pre-conditions and post-conditions, and to verify loop invariants. Program operations are directly expressed in the logic. We can also cite separation logic (see [12] for an introduction) which has been a very active field of research for the last years.

## 9 Conclusion and Further Directions

We presented a general mechanism for graph abstraction. We also defined graph transformations for abstract graphs that allow us to approximate behaviour of systems defined as graphs and graph transformations. As the number of possible different abstract graphs is finite, this approximation is finite. The construction of an abstract labelled transition system can be fully automatised. Our abstraction mechanism can be parametrised in several ways, thus obtaining different levels of precision. In particular, this can be parametrised in order to preserve and reflect properties on graphs expressed in a modal logic that we also present in this paper. That is, the abstraction guarantees that a (finite) set of properties hold in a graph if, and only if, they hold in its abstraction, and this also holds for all abstract graphs obtained by graph transformations starting from some initial start graph. This gives a parametrisable and fully automatised framework for verifying properties on systems described by graphs and graph transformations.

Our abstraction mechanism possess however some drawbacks which cause poor efficiency. This is related in the precision of the abstraction.

*Precision.* Our abstraction mechanism is not very precise, in the sense that concretisations of the same abstract graph may be very different in their shape and structure (see for example the list-like shape on Figure 3(c) and its concretisations on Figure 4(c)). Obviously, abstraction always leads to loss of precision. However, one could hope that the abstraction method is precise enough for interesting examples such as list and list manipulations.

*Complexity.* The insufficient precision is closely related to performance issues. Performing graph transformations on abstract graphs requires the so called materialisation step, which consists in locally concretising the abstract graph in order to extract a copy of the left-hand side of the graph transformation rule on which the actual transformation can be performed. In our abstraction mechanism, the materialisation part may lead to a very large number of materialisations. Each of this materialisations potentially results into a different application of the transformation rule, and a different result graph. This affects the performance of our algorithm, as we have to explore many different abstract transformations, and many of them may never occur in a concrete labelled transition system. Moreover, as we do not have a procedure for deciding whether an abstract graph admits concretisations (see Conjecture 32), the abstract labelled transition system may contain states that do not correspond to any concrete graph.

In order to improve our mechanism for abstract transformations, we need to improve the precision of the abstraction. This could be done in several ways.

One possible direction to be explored is restricting the graphs on which we apply the method. For instance, deterministic graphs were shown to be a good model for software systems and the memory heap and in our method we can expect that this restriction would decrease the number of possible abstract graphs.

Another improvement would be to increase the precision of the abstraction mechanism. However, one has to be careful with adapting the abstraction mechanism especially if the result of preservation of logic formulae is to be kept. The difficulty comes from the fact that whatever properties one decides to preserve (and make them parameters of the abstraction mechanism), these properties should be possible to update on abstract graphs after a graph transformation. We have several examples of such properties that give an interesting improvement of the

precision, but cannot be updated after transformation. Cyclicity and connectivity are such examples. That is, if a cyclicity or connectivity property is associated to some node or a set of nodes, then after performing an abstract graph transformation we cannot tell whether it still holds. However, the information on existence of cycles of small size may be integrated by using, for instance, hybrid logic.

A third possibility is to restrict the graph transformations that we allow. This is related to our second direction on improving precision. Actually, the abstract transformation mechanism and the abstraction mechanism are closely related, in the sense that properties used for abstraction should be possible to update, or at least conservatively update, after the graph transformation.

## References

1. ABDULLA, P. A., JONSSON, B., MAHATA, P., AND D'ORSO, J. Regular tree model checking. In *Proceedings of 14th International Conference on Computer Aided Verification (CAV'02)* (2002), vol. 2404 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 555–568.
2. BAUER, J. *Analysis of Communication Topologies by Partner Abstraction*. PhD thesis, Universität des Saarlandes, 2006.
3. BOUAIJANI, A., HABERMEHL, P., ROGALEWICZ, A., AND VOJNAR, T. Abstract regular tree model checking. In *Proceedings of the 7th International Workshop on Verification of Infinite-State Systems (INFINITY 2005)* (2006), vol. 149 of *Electronic Notes in Theoretical Computer Science*, Elsevier, pp. 37–48.
4. BOUAIJANI, A., HABERMEHL, P., AND VOJNAR, T. Abstract regular model checking. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV'04)* (2004), no. 3114 in *Lecture Notes in Computer Science*, Springer-Verlag.
5. BOUAIJANI, A., JONSSON, B., NILSSON, M., AND TOUILI, T. Regular model checking. In *Proceedings of 12th International Conference on Computer Aided Verification (CAV'00)* (2000), vol. 1855 of *Lecture Notes in Computer Science*, Springer-Verlag.
6. COUSOT, P. Abstract interpretation. *Symposium on Models of Programming Languages and Computation, ACM Computing Surveys* 28, 2 (June 1996), 324–328.
7. The GROOVE tool. <http://groove.cs.utwente.nl/>.
8. LEV-AMI, T., AND SAGIV, M. TVLA: A System for Implementing Static Analyses. In *Seventh International Static Analysis Symposium (SAS'00)* (2000).
9. RENSINK, A. Canonical graph shapes. In *Programming Languages and Systems — European Symposium on Programming (ESOP)* (2004), D. A. Schmidt, Ed., vol. 2986 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 401–415.
10. RENSINK, A. The GROOVE simulator: A tool for state space generation. In *Applications of Graph Transformations with Industrial Relevance (AGTIVE'03)* (2004), pp. 479–485.
11. RENSINK, A., AND DISTEFANO, D. Abstract graph transformation. In *International Workshop on Software Verification and Validation (SVV)* (2005), *Electronic Notes in Theoretical Computer Science*.
12. REYNOLDS, J. C. Separation logic: A logic for shared mutable data structures. In *17th IEEE Symposium on Logic in Computer Science (LICS 2002)* (2002), IEEE Computer Society, pp. 55–74.
13. ROSSMAN, B. Existential positive types and preservation under homomorphisms. In *20th IEEE Symposium on Logic in Computer Science* (2005), Cambridge University Press, pp. 467–476.
14. SAGIV, M., REPS, T., AND WILHELM, R. Solving shape-analysis problems in languages with destructive updating. *ACM Trans. Program. Lang. Syst.* 20, 1 (Jan. 1998), 1–50.
15. SAGIV, M., REPS, T., AND WILHELM, R. Parametric shape analysis via 3-valued logic. *ACM Trans. Program. Lang. Syst.* 24, 3 (May 2002), 217–298.
16. The TVLA tool. <http://www.cs.tau.ac.il/~tvla/>.
17. YORSH, G., RABINOVICH, A., SAGIV, M., MEYER, A., AND BOUAIJANI, A. A logic of reachable patterns in linked data-structures. In *Proceedings Foundations of Software Science and Computation Structures, (FOSSACS 2006)* (2006), vol. 3921 of *Lecture Notes in Computer Science*, Springer-Verlag.



## A Proof of Proposition 12

**Proposition 12.** *Let  $S, T$  and  $U$  be shapes,  $f$  be a shape morphism between  $S$  and  $T$  and  $g$  another such morphism between  $T$  and  $U$ . Then  $g \circ f$  (the function composition of  $f$  and  $g$ ) is a shape morphism between  $S$  and  $U$ . ◀*

*Proof.* The first two axioms of Definition 11 are not difficultly seen to be met for  $g \circ f$ . For the third one, we only consider the property on the outgoing edges multiplicity function; the one for incoming edges multiplicity follows by symmetry.

Consider a node  $w$  in  $N_U$ , a label  $\mathbf{a}$  and a class of group-equivalent nodes  $C \in N_U / \simeq_U$ . Let also  $v' \in N_T$  be a node such that  $f(v) = v'$  and  $g(v') = w$ . Then, by  $g$  being a shape morphism, we know that

$$\text{mult}_U^{\circ}(w, \mathbf{a}, C) = \sum_{D' \in (g^{-1}(C)) / \simeq_T}^{\mu} \text{mult}_T^{\circ}(v', \mathbf{a}, D').$$

On the other hand, by  $f$  being a shape morphism and by definition of  $v'$ , the right-hand side of this equality can be expanded, giving the following

$$\text{mult}_U^{\circ}(w, \mathbf{a}, C) = \sum_{D' \in (g^{-1}(C)) / \simeq_T}^{\mu} \sum_{D \in (f^{-1}(D')) / \simeq_S}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D).$$

Now, given that any  $D'$  in the inner sum is mapped to one and only one  $D$  (this is a basic consequence of  $f$  being a – total – function), we can combine the two sums into one:

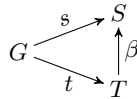
$$\text{mult}_U^{\circ}(w, \mathbf{a}, C) = \sum_{D \in (f^{-1}(g^{-1}(C))) / \simeq_S}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D),$$

which asserts that  $g \circ f$  is a shape morphism between  $S$  and  $U$ . ◻

## B Proof of Lemma 22

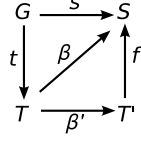
**Lemma 22.** *Let  $G$  be a graph,  $S, T$  be shapes,  $s : G \rightarrow S, t : G \rightarrow T$  be abstraction morphisms, and  $\beta : T \rightarrow S$  be a shape morphism such that  $s = \beta \circ t$ .*

1. *If  $s$  is the neighbourhood abstraction morphism of  $G$ , then  $\beta$  is the neighbourhood shape morphism of  $T$ .*
2. *If  $\beta$  is the neighbourhood shape morphism of  $T$ , then  $s$  is the neighbourhood abstraction morphism of  $G$ .*



For Statement 1, the proof goes as follows (see also Figure 8):

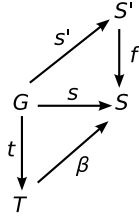
- We first show that the neighbourhood shape morphism of  $T$  exists, let it be the morphism  $\beta' : T \rightarrow T'$ ,



**Fig. 8.** Proof of Statement 1.

- We then define that there exists a morphism  $f : T' \rightarrow S$  which is a shape isomorphism and such that  $\beta = f \circ \beta'$ .

For Statement 2, we show that there exists a morphism  $f : S' \rightarrow S$  such that  $s = f \circ s'$ , where  $s' : G \rightarrow S'$  is the abstraction morphism of graph  $G$ . See also Figure 9.



**Fig. 9.** Proof of Statement 2.

In the following we consider a fixed  $i$  for the abstraction radius.

### B.1 Proof of Statement 1

**The neighbourhood shape morphism of  $T$  exists.** To show that the neighbourhood shape morphism of  $T$  exists, it is enough to show that  $\sim_i$  is defined. By definition,  $\sim_i$  is defined if  $\sim_{i-1}$  is defined and  $\simeq_T \subseteq \sim_{i-1}$ . This is shown in the following lemma.

**Lemma 56.** *Let  $G$  be a graph,  $S, T$  be shapes,  $s : G \rightarrow S$ ,  $t : G \rightarrow T$  be abstraction morphisms, and  $\beta : T \rightarrow S$  be a shape morphism such that  $s = \beta \circ t$ . If  $s$  is the neighbourhood abstraction morphism of  $G$ , then  $\sim_{i-1}$  is defined on the nodes of  $T$ , and  $\simeq_T \subseteq \sim_{i-1}$ . ◀*

Lemma 56 is shown in Section B.3. The proof of this lemma allows us to establish the following corollary. This corollary is used later on for the proof of Statement 1, and also for the proof of Statement 2. The proof of Corollary 57 is also shown in Section B.3.

**Corollary 57.** *Let  $G$  be a graph,  $S, T$  be shapes,  $s : G \rightarrow S$ ,  $t : G \rightarrow T$  be abstraction morphisms, and  $\beta : T \rightarrow S$  be a shape morphism such that  $s = \beta \circ t$ . If at least one of these conditions is verified:*

1.  $s$  is the neighbourhood abstraction morphism of  $G$ ,
2.  $\beta$  is the neighbourhood shape morphism of  $T$ ,

then for any  $0 \leq j \leq i$ , and for all  $v, v' \in N_G$  and all  $e, e' \in E_G$ ,

$$v \equiv_j v' \Leftrightarrow t(v) \sim_j t(v') \quad \text{and} \quad e \equiv_j e' \Leftrightarrow t(e) \sim_j t(e').$$

◀

**The abstraction isomorphism  $f$  exists.** We show here that there exists a morphism  $f : T' \rightarrow S$  which is a shape isomorphism and such that  $\beta = f \circ \beta'$  (see Figure 8). Remind that that  $\beta' : T \rightarrow T'$  is the neighbourhood shape morphism of  $T$ . We start by defining a mapping  $f$  from nodes and edges  $T'$  to nodes and edges of  $S$ , and we show that this mapping is a morphism, it is bijective and  $f$  and  $f^{-1}$  are shape morphisms.

*Define a mapping  $f$  such that  $\beta = f \circ \beta'$ .* Remind that the abstraction morphism  $t$  and the shape morphism  $\beta'$  are surjective. Thus, any node (resp. edge) of  $T'$  can be written as  $t(\beta'(x))$  for some node (resp. edge)  $x$  of  $G$ . Then  $f$  is defined by: for any  $t(\beta'(x))$  node or edge of  $T'$ ,  $f(t(\beta'(x))) = s(x)$ . Then  $\beta = f \circ \beta'$  by definition of  $f$  and using  $s = \beta \circ t$ .

*Show that  $f$  is a morphism.* The fact that  $f$ , defined as previously, is a morphism is not very difficult to deduce using that  $t, \beta, \beta'$  are morphisms.

*Show that  $f$  is a bijection.* Showing that  $f$  is a surjection is easily done using the definition of  $f$  and the fact that  $s$  is a surjection. Let us show that  $f$  is an injection, that is, for any  $t(\beta'(v)), t(\beta'(v'))$  nodes of  $T'$ , if  $t(\beta'(v)) \neq t(\beta'(v'))$ , then  $s(v) \neq s(v')$ . Now,  $t(\beta'(v)) \neq t(\beta'(v'))$  if, and only if, by definition,  $t(v) \not\sim_i t(v')$  and, using Corollary 57, if, and only if,  $v \not\equiv_i v'$ , which is equivalent to  $s(v) \neq s(v')$  by definition of the neighbourhood abstraction morphism  $s$ . The same reasoning can be applied to edges.

*Show that  $f$  and  $f^{-1}$  are shape morphisms.* This is not difficult to show using the following facts:

1. for all nodes  $w, w'$  of  $T'$ ,  $w \simeq_{T'} w'$  if, and only if,  $f(w) \simeq_S f(w')$ . This follows from the fact that  $\simeq_{T'} = \sim_i$  and  $\simeq_S = \equiv_i$ , and using the dependence and characteristics of the abstraction and shape morphisms  $s, t, \beta, \beta', f$ ;
2. for all node or edge  $x$  of  $S$ ,  $s^{-1}(x) = (t \circ \beta' \circ f)^{-1}(x)$ . This fact together with the previous one ensure equality of the multiplicity functions of  $S$  and  $T'$ .

## B.2 Proof of Statement 2

We start by defining  $f$  first as a mapping, then showing that it is indeed a morphism, a bijection, and that  $f$  and  $f^{-1}$  are shape morphisms.

*Define a mapping  $f$  such that  $s = f \circ s'$ .* Remind that  $s' : G \rightarrow S'$  is the neighbourhood abstraction morphism of  $G$ . As  $s'$  is an abstraction morphism, it is surjective, thus any node (resp. edge) of  $S'$  can be written as  $s'(x)$  for  $x$  a node (resp. edge) of  $G$ . Then  $f$  is defined by: for any  $x \in N_G \cup E_G$ ,  $f(s'(x)) = s(x)$ . Thus,  $s = f \circ s'$  by definition.

*Show that  $f$  is a morphism.* Showing that  $f$  is a morphism is not difficult using that  $s'$  and  $s$  are morphisms.

*Show that  $f$  is a bijection.* The morphism  $f$  is surjective as  $s'$  and  $s$  are surjective. For injection, we have to show that for all nodes  $v, v'$  of  $G$ , if  $s'(v) \neq s'(v')$ , then  $s(v) \neq s(v')$  (and the same for edges, but it easily follows). As  $s'$  is the neighbourhood abstraction morphism of  $G$ , we know that  $s'(v) \neq s'(v')$  if, and only if,  $v \not\equiv_i v'$ . On the other hand, as  $s = \beta \circ t$  and  $\beta$  is the neighbourhood shape morphism of  $T$ , we have that  $s(v) \neq s(v')$  if, and only if,  $t(v) \not\sim_i v'$ . Now, by Corollary 57 we know that  $v \not\equiv_i v'$  if, and only if,  $t(v) \not\sim_i v'$ , which shows that  $f$  is injective.

Show that  $f$  and  $f^{-1}$  are shape morphisms. As for Statement 1, it is not difficult to show using the results that have already been established.

### B.3 Proofs of Lemma 56 and Corollary 57

Let us first show Lemma 56, then we argue how its proof is used for deducing Corollary 57.

#### Proof of Lemma 56

**Lemma 56.** *Let  $G$  be a graph,  $S, T$  be a shapes,  $s : G \rightarrow S$ ,  $t : G \rightarrow T$  be abstraction morphisms, and  $\beta : T \rightarrow S$  be a shape morphism such that  $s = \beta \circ t$ . If  $s$  is the neighbourhood abstraction morphism of  $G$ , then  $\sim_{i-1}$  is defined on the nodes of  $T$ , and  $\simeq_T \subseteq \sim_{i-1}$ . ◀*

We show by induction on  $0 \leq j \leq i - 1$  that

IHA( $j$ ):  $\sim_j$  is defined,

IHB( $j$ ): assuming that IHA( $j$ ),  $\simeq_T \subseteq \sim_j$ ,

IHC( $j$ ): assuming IHA( $j$ ) and IHB( $j$ ),  $\forall w \in N_G$ ,  $t([w]_{\equiv_j}) = [t(w)]_{\sim_j}$ .

We start with the following intermediate result.

**Fact 58** *For any  $k \leq i$ , it holds that*

$$\forall v, v' \in N_G, \quad t(v) \simeq_T t(v') \quad \text{implies} \quad v \equiv_k v'$$

◀

*Proof.* The proof of this fact goes as follows.

$$\begin{aligned} t(v) \simeq_T t(v') &\implies (\beta \text{ is a shape morphism}) \\ \beta(t(v)) \simeq_S \beta(t(v')) &\iff (s = \beta \circ t) \\ s(v) \simeq_S s(v') &\iff (s \text{ is the level } i \text{ neighbourhood abstraction morphism of } G) \\ v \equiv_i v &\implies (\text{holds for any } j \geq i) \\ v \equiv_j v' & \end{aligned}$$

□

Going back to the proof of the lemma, using Fact 58 and definition of  $\subseteq$  for equivalence relations, we can see that for IHB( $j$ ) it is enough to show the following

$$\forall v, v' \in N_G, \quad v \equiv_j v' \text{ implies } t(v) \sim_j t(v') \tag{1}$$

*Base case.* For the base case,  $j = 0$ . For IHA(0),  $\sim_0$  is always defined.

For IHB(0), we have to show (1). It is the case that for any nodes  $v, v' \in N_G$ ,  $t(v) \sim_0 t(v')$  if, and only if,  $v \equiv_0 v'$ ; this is because  $\sim_0$  and  $\equiv_0$  only take into account labels for grouping nodes and labels are preserved and reflected by the abstraction morphism. The same argument lets us conclude that IHC(0) also holds.

*General case.* Assume that the induction hypotheses IHA( $k$ ), IHB( $k$ ) and IHC( $k$ ) hold for any  $k < j$ . Let us show that they hold for  $j$ .

For IHA( $j$ ), by definition  $\sim_j$  is defined if  $\sim_{j-1}$  is defined and  $\simeq_T \subseteq \sim_{j-1}$ , thus IHA( $j$ ) follows from IHA( $j-1$ ) and IHB( $j-1$ ).

For IHB( $j$ ), we first establish two intermediary results in Fact 59 and Fact 60.

**Fact 59**  $\forall v, u \in N_G, \forall \mathbf{a} \in \text{Lab}$ , it holds that

$$\sum_{K \in N_T / \simeq_T \mid K \subseteq [t(u)]_{\sim_{j-1}}}^{\mu} \text{mult}_T^{\circ}(t(v), \mathbf{a}, K) = \left| v \triangleright_G^{\mathbf{a}} [u]_{\equiv_{j-1}} \right|_{\mu}.$$

◀

*Proof.* By definition,  $\text{mult}_T^{\circ}(t(v), \mathbf{a}, K)$  is equal to  $\left| v \triangleright_G^{\mathbf{a}} (t^{-1}(K)) \right|_{\mu}$  (for all  $v, \mathbf{a}, K$ ). Note now that the  $K \in N_T / \simeq_T$  summed-up above form a partition of the set  $[t(u)]_{\sim_{j-1}}$ ; this comes from the fact that  $\simeq_T \subseteq \sim_{j-1}$  (IHB( $j-1$ )). Then, by definition of set multiplicity function,

$$\sum_{K \in N_T / \simeq_T \mid K \subseteq [t(u)]_{\sim_{j-1}}}^{\mu} \text{mult}_T^{\circ}(t(v), \mathbf{a}, K) = \left| v \triangleright_G^{\mathbf{a}} (t^{-1}([t(u)]_{\sim_{j-1}})) \right|_{\mu}$$

and it holds for all  $v, u \in N_G$ , and all label  $\mathbf{a}$ . Now by IHC( $j-1$ ),  $[t(u)]_{\sim_{j-1}} = t([u]_{\equiv_{j-1}})$ , therefore  $t^{-1}([t(u)]_{\sim_{j-1}}) = t^{-1}(t([u]_{\equiv_{j-1}})) = [u]_{\equiv_{j-1}}$ .  $\square$

**Fact 60** It holds that

$$\forall v, v' \in N_G, \quad v \equiv_j v' \iff t(v) \sim_j t(v')$$

◀

*Proof.* By definition of the neighbourhood equivalence relations on graphs and shapes,

$$v \equiv_j v' \iff \forall u \in N_G, \quad \forall \mathbf{a} \in \text{Lab}, \quad \left| v \triangleright_G^{\mathbf{a}} [u]_{\equiv_{j-1}} \right|_{\mu} = \left| v' \triangleright_G^{\mathbf{a}} [u]_{\equiv_{j-1}} \right|_{\mu}$$

and

$$t(v) \sim_j t(v') \iff \sim_{j-1} \text{ is defined and } \forall u \in N_G, \quad \forall \mathbf{a} \in \text{Lab},$$

$$\sum_{K \in N_T / \simeq_T \mid K \subseteq [t(u)]_{\sim_{j-1}}}^{\mu} \text{mult}_T^{\circ}(t(v), \mathbf{a}, K) = \sum_{K \in N_T / \simeq_T \mid K \subseteq [t(u)]_{\sim_{j-1}}}^{\mu} \text{mult}_T^{\circ}(t(v'), \mathbf{a}, K).$$

The statement of Fact 60 immediately follows from these definitions and Fact 59.  $\square$

Back to the proof of IHB( $j$ ), we have to show (1). This immediately follows from Fact 58 and Fact 60.

Finally, for IHC( $j$ ), we have to show that  $\forall w \in N_G, t([w]_{\equiv_j}) = [t(w)]_{\sim_j}$ . By definition,  $t([w]_{\equiv_j})$  is the set  $(\forall u \in N_G, \forall \mathbf{a} \in \text{Lab})$

$$\left\{ t(w') \mid \left| w' \triangleright_G^{\mathbf{a}} [u]_{\equiv_{j-1}} \right|_{\mu} = \left| w \triangleright_G^{\mathbf{a}} [u]_{\equiv_{j-1}} \right|_{\mu} \right\}$$

and  $[t(w)]_{\sim_j}$  is the set  $(\forall u \in N_G, \forall a \in \text{Lab})$

$$\left\{ t(w') \left| \sum_{K \in N_T / \simeq_T | K \subseteq [t(w)]_{\sim_{j-1}}}^{\mu} \text{mult}_T^{\circ}(t(w'), a, K) = \sum_{K \in N_T / \simeq_T | K \subseteq [t(w)]_{\sim_{j-1}}}^{\mu} \text{mult}_T^{\circ}(t(w), a, K) \right. \right\}$$

According to Fact 59, these two sets are equal. This finishes the proof for  $\text{IHC}(j)$ , and thus the proof of Lemma 56.

### Proof of Corollary 57

**Corollary 57.** *Let  $G$  be a graph,  $S, T$  be shapes,  $s : G \rightarrow S$ ,  $t : G \rightarrow T$  be abstraction morphisms, and  $\beta : T \rightarrow S$  be a shape morphism such that  $s = \beta \circ t$ . If at least one of these conditions is verified:*

1.  $s$  is the neighbourhood abstraction morphism of  $G$ ,
2.  $\beta$  is the neighbourhood shape morphism of  $T$ ,

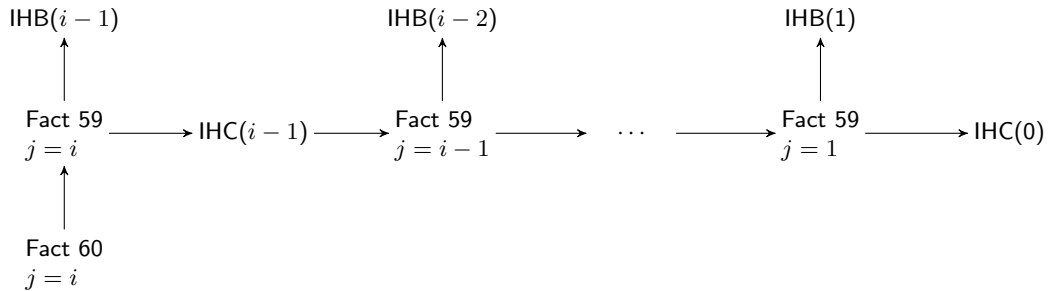
then for any  $0 \leq j \leq i$ , and for all  $v, v' \in N_G$  and all  $e, e' \in E_G$ ,

$$v \equiv_j v' \Leftrightarrow t(v) \sim_j t(v') \quad \text{and} \quad e \equiv_j e' \Leftrightarrow t(e) \sim_j t(e').$$

◀

We only show the corollary for nodes, the statement for edges easily follows from the definitions. One can notice that the statement of this corollary is very similar to what has been shown in Fact 60. However, there are some small differences either in the hypotheses, or in what has been proved. We show how the proof of Fact 60 can be completed for showing the corollary.

The first difference is that Fact 60 is shown for  $j < i$ . However, the proof of Fact 60 can be extended for  $j = i$ . Figure 10 illustrates what needs to be shown for Fact 60 with  $j = i$ . That is, for showing Fact 60 with  $j = i$ , we use Fact 59 with  $j = i$ ; the proof of Fact 59 for  $j = i$  uses  $\text{IHB}(i-1)$  and  $\text{IHC}(i-1)$ . Thus, it is enough to show that  $\text{IHC}(0)$  and  $\text{IHB}(j)$  for all  $0 \leq j \leq i-1$  hold. (Here  $\text{IHC}()$  and  $\text{IHB}()$  denote the induction hypotheses used for proving Lemma 56).



**Fig. 10.** Proof dependence for Fact 60.

Now, if condition 1 of the corollary holds, it corresponds exactly to the hypotheses of Lemma 56. Thus  $\text{IHC}(0)$  and  $\text{IHB}(j)$  for all  $0 \leq j \leq i-1$  can be shown. If condition 2 of the

corollary holds,  $\text{IHB}(j)$  holds for all  $0 \leq j \leq i - 1$ . Remind that  $\text{IHB}(j)$  states that  $\simeq_T \subseteq \sim_j$ , and this is the case because the neighbourhood shape morphism  $\beta$  of  $T$  exists. Moreover, it is immediate that  $\forall w \in N_G, t([w]_{\equiv_0}) = [t(w)]_{\sim_0}$  (that is,  $\text{IHC}(0)$ ) holds under condition 2.

## C Proof of Lemma 23

**Lemma 23.** *If any two neighbourhood shapes have a common concretisation, then they are isomorphic.* ◀

*Proof. (Sketch)* We show that any abstraction morphism to a neighbourhood shape is a neighbourhood abstraction morphism. That is, if  $G$  is a graph,  $s : G \rightarrow S$  is its neighbourhood abstraction morphism and  $t : G \rightarrow T$  is some arbitrary morphism with  $T$  being a neighbourhood shape (i.e., there exists a graph  $H$  with neighbourhood abstraction morphism  $t' : H \rightarrow T$ ), then  $T$  and  $S$  are isomorphic. We consider that  $S$  and  $T$  are given with their canonical repre-

$$\begin{array}{ccc} H & \xrightarrow{t'} & T \\ & \nearrow t & \\ G & \xrightarrow{s} & S \end{array}$$

sentation (see Section 4.2), and we show that  $S$  and  $T$  have the same canonical representation. By Lemma 28, this implies that  $S$  and  $T$  are isomorphic. Let  $v$  be a node of  $G$ . Remind that  $s(v)$  and  $t(v)$  are canonical names. If we show that  $s(v) = t(v)$  (as a canonical name), then it would imply that the set of node canonical names of  $S$  and  $T$  are the same (remind that the morphisms  $s$  and  $t$  are surjective). The same is similarly shown for edges.

A level  $i$  canonical name is of the form  $\langle C, \text{out}, \text{in} \rangle$ , where  $\text{out}$  and  $\text{in}$  are multiplicity functions, and  $C$  is a level  $i - 1$  canonical name. Of course,  $C$  has as first component a level  $i - 2$  canonical name, and so on. Thus, a level  $i$  canonical name contains a level  $j$  canonical name for any  $0 \leq j \leq i$ . In the following we call it its level  $j$  component.

We show that (for any  $v \in N_G$ ):

- for all  $0 \leq j \leq i$ , the level  $j$  components of  $t(v)$  and  $s(v)$  (considered as node canonical names) are the same,
- $s(v)$  and  $t(v)$  have the same  $\text{out}$  and  $\text{in}$  multiplicity functions.

The proof of these two statements goes by a simple induction on  $j$  that we omit here. ◻

## D Proof of Lemma 28

**Lemma 28.** *Let  $G, H$  be graphs, and let  $i \geq 1$ . The level  $i$  neighbourhood shapes of  $G$  and  $H$  are isomorphic if, and only if,  $\mathcal{C}_G$  and  $\mathcal{C}_H$  are equal.*

*Proof.* For the right-to-left direction, we show that  $\mathcal{C}_G$  defines uniquely a shape  $S$  which is isomorphic to the level  $i$  neighbourhood shape of  $G$ . Let  $S = \langle G_S, \simeq_S, \text{mult}_S^n, \text{mult}_S^o, \text{mult}_S^i \rangle$  be the shape defined by:

- $N_S = \mathcal{N}$ ,  $E_S = \mathcal{E}$  and for any  $e = \langle C, \mathbf{a}, C' \rangle$  in  $\mathcal{E}$ ,  $\text{src}_S(e) = C$ ,  $\text{tgt}_S(e) = C'$  and  $\text{lab}_S(e) = \mathbf{a}$ ;

- $\simeq_S$  is the smallest equivalence relation such that  $C \simeq_S C'$  if  $C$  and  $C'$  have the same first component. Remind that  $C$  and  $C'$  are level  $i$  canonical names and their first component is a level  $i - 1$  canonical name;
- $\text{mult}_S^n = \text{mult}$ ;
- for all  $C \in N_S$ ,  $\mathbf{a} \in \text{Lab}$ , and  $K \in \text{NCan}^{i-1}$ ,  $\text{mult}_S^o(C, \mathbf{a}, K) = \text{out}_C(K, \mathbf{a})$ , where  $\text{out}_C$  is the function second component of  $C$  (remind that  $C$  is a level  $i$  canonical name);
- for all  $C \in N_S$ ,  $\mathbf{a} \in \text{Lab}$ , and  $K \in \text{NCan}^{i-1}$ ,  $\text{mult}_S^o(C, \mathbf{a}, K) = \text{in}_C(K, \mathbf{a})$ , where  $\text{in}_C$  is the function third component of  $C$  (remind that  $C$  is a level  $i$  canonical name).

Note that  $S$  is indeed a shape. We do not show this here, but a similar construction is introduced and shown correct in Section 4.3.

Let us now show that  $S$  is isomorphic to the level  $i$  neighbourhood shape of  $G$ . Consider  $T$ , the level  $i$  neighbourhood shape of  $G$ , and consider the function  $f = (f_n, f_e), f_n : N_S \rightarrow N_T, f_e : E_S \rightarrow E_T$  defined by:

- $f_n(C) = \{v \in N_G \mid \text{name}_G^i(v) = C\}$ ;
- $f_e(C) = \{e \in E_G \mid \text{name}_G^i(e) = C\}$ .

Using Lemma 27,  $f_n$  and  $f_e$  are bijections, and it is not difficult to see that  $f$  is a graph morphism, thus a graph isomorphism. Showing that  $f$  is a shape morphism is quite technical, but not difficult, and only uses definitions of  $S$ , of  $T$ , of shape morphism and of neighbourhood shape morphism. This concludes the proof of the right-to-left direction.

Now, for the left-to-right direction, let  $S$  be the level  $i$  neighbourhood shape of  $G$  with neighbourhood abstraction morphism  $s : G \rightarrow S$ , and let  $T$  be the level  $i$  neighbourhood shape of  $H$  with neighbourhood abstraction morphism  $t : H \rightarrow T$ . Suppose that  $S$  and  $T$  are isomorphic with isomorphism  $f : S \rightarrow T$ . We use the modal logic to prove that  $\mathcal{C}_G$  and  $\mathcal{C}_H$  are equal. We use the following results of the modal logic.

- (1) Neighbourhood abstraction morphism preserves and reflects logic formulae (Proposition 52).
- (2) For any graph  $G$ , any two nodes  $v, w$  of  $G$ ,  $v, w$  are level  $i$  neighbourhood equivalent if, and only if,  $v, w$  have the same level  $i$  canonical names, if, and only if,  $v, w$  satisfy the same depth  $i$  logic formulae (Lemma 27 and Lemma 54).
- (3) For any level  $i$  canonical name  $C$ , there exists a representative formula  $\phi_C$  such that in any graph  $G$ , any node  $v$  of  $G$ ,  $\text{name}_G^i(v) = C$  if, and only if,  $G, v \models \phi_C$  (Lemma 55).
- (4) By (1), (2) and (3), we can deduce that for any graph  $G$  and  $i \geq 1$ , if  $S$  is the level  $i$  neighbourhood shape of  $G$  with corresponding abstraction morphism  $s : G \rightarrow S$ , and if  $C = \text{name}_G^i(v)$  for some  $v$  node of  $G$ , then  $S, s(v) \models \phi_C$ .

Let  $\mathcal{C}_G = \langle \mathcal{N}_G, \mathcal{E}_G, \text{mult}_G \rangle$  and  $\mathcal{C}_H = \langle \mathcal{N}_H, \mathcal{E}_H, \text{mult}_H \rangle$ . Remark first that the modal logic cannot distinguish isomorphic structures, and this holds both for graph and shapes. This means that  $(\star)$  if  $v$  is a node of  $S$ , then for any level  $i$  logic formula  $\phi$ ,  $S, v \models \phi$  if, and only if,  $T, f(v) \models \phi$ . Now, let  $C \in \mathcal{N}_G$ , and let  $v$  be a node of  $G$  s.t.  $\text{name}_G^i(v) = C$ . Then, by (3),  $G, v \models \phi_C$ , and by (4),  $S, s(v) \models \phi_C$ . Now, by  $(\star)$  we deduce that  $T, f(s(v)) \models \phi_C$ . By preservation and reflection of the logic (by (1)), we have that for any  $w \in t^{-1}(f(s(v)))$  node of  $H$ ,  $H, w \models \phi_C$ . By (3), we deduce that  $C$  is the level  $i$  canonical name of  $w \in N_H$ , thus  $C \in \mathcal{N}_H$ . That is, we just showed that  $\mathcal{N}_G \subseteq \mathcal{N}_H$ . Symmetrically we can show that  $\mathcal{N}_H \subseteq \mathcal{N}_G$ , and thus  $\mathcal{N}_G = \mathcal{N}_H$ .

It is not difficult to prove that also  $\mathcal{E}_G = \mathcal{E}_H$ , and the same for the multiplicity functions.  $\square$



## E Proof of Proposition 50

**Proposition 50.** *Let  $\mathcal{P}$  be a set of atomic propositions,  $S, T$  be shapes,  $\gamma_S : N_S \rightarrow 2^{\mathcal{P}}$  and  $\gamma_T : N_T \rightarrow 2^{\mathcal{P}}$  be valuation functions such that  $\simeq_T$  is compatible with  $\gamma_T$ , and let  $\alpha : S \rightarrow T$  be a shape morphism.*

(preservation): *If  $\alpha$  preserves  $\mathcal{P}$  under  $\gamma_S, \gamma_T$ , then  $\alpha$  preserves the negation free fragment of  $\mathcal{L}_1(\mathcal{P})$  under  $\gamma_S, \gamma_T$ .*

(reflection): *If  $\alpha$  reflects  $\mathcal{P}$  under  $\gamma_S, \gamma_T$ , then  $\alpha$  reflects the negation free fragment of  $\mathcal{L}_1(\mathcal{P})$  under  $\gamma_S, \gamma_T$ .*

(preservation and reflection): *If  $\alpha$  preserves and reflects  $\mathcal{P}$  under  $\gamma_S, \gamma_T$ , then  $\alpha$  preserves and reflects  $\mathcal{L}_1(\mathcal{P})$  (possibly with negation) under  $\gamma_S, \gamma_T$ . ◀*

The rest of the section is devoted to the proof of this proposition.

Let  $\mathcal{M}(\mathcal{P})$  be the set of logic formulae defined by the symbol  $\phi$  in the following syntax:

$$\begin{aligned} \phi &::= \mathbf{tt} \mid \mathbf{a}^\lambda \cdot \psi \mid \langle \mathbf{a}^\lambda \cdot \psi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg \phi \\ \psi &::= \mathbf{p} \mid \psi \vee \psi \mid \psi \wedge \psi \mid \neg \psi \mid \mathbf{tt} \end{aligned}$$

The set of propositions defined by  $\psi$  above is denoted  $Bool(\mathcal{P})$ . It is not difficult to see that  $\phi$  defines a grammar for  $\mathcal{L}_1(\mathcal{P})$ , thus  $\mathcal{M}(\mathcal{P})$  is exactly the logic  $\mathcal{L}_1(\mathcal{P})$ . We use this grammar in the proof for an induction on the structure of a  $\mathcal{L}_1(\mathcal{P})$  formula.

Let us state Proposition 50 using the previous notations. Let  $\mathcal{P}$  be a set of atomic propositions and  $S, T$  be shapes,  $\gamma_S : N_S \rightarrow 2^{\mathcal{P}}$  and  $\gamma_T : N_T \rightarrow 2^{\mathcal{P}}$  be valuations such that  $\simeq_T$  is compatible with  $\gamma_T$ , and  $\alpha : S \rightarrow T$  be a shape morphism. If  $\alpha$  preserves  $\mathcal{P}$ , then for any node  $v$  in  $N_S$  and for any  $\mathcal{M}(\mathcal{P})$  formula  $\phi$  without negation

$$\text{(preservation)} \quad S, v, \gamma_S \models \phi \text{ implies } T, \alpha(v), \gamma_T \models \phi$$

If  $\alpha$  reflects  $\mathcal{P}$ , then for any node  $v$  in  $N_S$  and for any  $\mathcal{M}(\mathcal{P})$  formula  $\phi$  without negation

$$\text{(reflection)} \quad T, \alpha(v), \gamma_T \models \phi \text{ implies } S, v, \gamma_S \models \phi$$

If moreover  $\alpha$  preserves and reflects  $\mathcal{P}$ , then for any node  $v$  in  $N_S$ , and for any  $\mathcal{M}(\mathcal{P})$  formula  $\phi$  (possibly with negation), both (preservation) and (reflection) hold.

For brevity, we omit the valuations  $\gamma_S$  and  $\gamma_T$ , as they are fixed for each of the shapes.

We first show some preliminary lemmas.

**Lemma 61 ( $\alpha$  preserves / reflects  $Bool(\mathcal{P})$ ).** *If  $\alpha$  preserves  $\mathcal{P}$ , then  $\alpha$  preserves  $Bool(\mathcal{P})$ , that is, for any  $Bool(\mathcal{P})$  formula  $\psi$  without negation and for any node  $v$  in  $N_S$ , if  $S, v \models \psi$ , then  $T, \alpha(v) \models \psi$ .*

*If  $\alpha$  reflects  $\mathcal{P}$ , then  $\alpha$  reflects  $Bool(\mathcal{P})$  without negation, that is, for any  $Bool(\mathcal{P})$  formula  $\psi$  without negation and for any node  $v$  in  $N_S$ , if  $T, \alpha(v) \models \psi$ , then  $S, v \models \psi$ .*

*If  $\alpha$  preserves and reflects  $\mathcal{P}$ , then  $\alpha$  preserves and reflects  $Bool(\mathcal{P})$ , that is, for any  $Bool(\mathcal{P})$  formula  $\psi$  and for any node  $v$  in  $N_S$ ,  $S, v \models \psi$ , if, and only if,  $T, \alpha(v) \models \psi$ . ◀*

*Proof.* The proof is an easy induction on the structure of  $\psi$  that we will omit. ◻

**Lemma 62 ( $\simeq_T$  is compatible with  $Bool(\mathcal{P})$ ).** *If  $\simeq_T$  is compatible with  $\gamma_T$ , for any two nodes  $v, w$  in  $N_T$  and for any formula  $\psi$  in  $Bool(\mathcal{P})$  if  $v \simeq_T w$ , then  $T, v, \gamma_T \models \psi$  if, and only if,  $T, w, \gamma_T \models \psi$ . ◀*

*Proof.* By induction on the structure of  $\psi$ ; the base case for  $\psi = \mathbf{p}$  uses the fact that  $\simeq_T$  is compatible with  $\mathcal{P}$ .  $\square$

We call this property  $\simeq_T$  *is compatible with*  $Bool(\mathcal{P})$ , in the sense that two  $\simeq_T$ -equivalent nodes satisfy the same  $Bool(\mathcal{P})$  formulae.

Let us now go to the proof of Proposition 50. Consider a node  $v$  in  $N_S$  fixed from now on. We first show preservation.

### E.1 Preservation

Assume that  $\alpha$  preserves  $\mathcal{P}$ , and let  $\phi$  be a  $\mathcal{M}(\mathcal{P})$  formula without negation s.t.  $S, v \models \phi$ . We show that  $T, \alpha(v) \models \phi$  and the proof goes by induction on the structure of  $\phi$ . For the base case, either  $\phi = \mathbf{tt}$ , and then the proposition is trivial, or  $\phi$  is in  $Bool(\mathcal{P})$  without negation, in which case the proposition follows from Lemma 61. For the induction step, if  $\phi = \phi_1 \vee \phi_2$  or  $\phi = \phi_1 \wedge \phi_2$ , then, by definition of satisfaction and by induction hypothesis, preservation property easily follow. Let us now show the preservation for  $\phi = \mathbf{a}^\lambda \cdot \psi$  for some  $\psi$  in  $Bool(\mathcal{P})$  without negation. By definition,  $S, v \models \mathbf{a}^\lambda \cdot \psi$  if, and only if,

$$\sum_{D \in X}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D) \geq \lambda$$

and  $T, \alpha(v) \models \mathbf{a}^\lambda \cdot \psi$  if, and only if,

$$\sum_{C \in Y}^{\mu} \text{mult}_T^{\circ}(\alpha(v), \mathbf{a}, C) \geq \lambda$$

where  $X$  and  $Y$  are the sets

$$\begin{aligned} X &= \{D \in N_S / \simeq_S \mid \forall w \in D. S, w \models \psi\} \\ Y &= \{C \in N_T / \simeq_T \mid \forall w \in C. T, w \models \psi\}. \end{aligned}$$

By definition of abstraction,  $\sum_{C \in Y}^{\mu} \text{mult}_T^{\circ}(\alpha(v), \mathbf{a}, C)$  is equal to

$$\sum_{C \in Y}^{\mu} \sum_{D \in (\alpha^{-1}(C)) / \simeq_S}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D) \tag{2}$$

Now, for any two different  $C_1$  and  $C_2$  group-equivalent classes in  $Y$ , the sets of nodes  $\alpha^{-1}(C_1)$  and  $\alpha^{-1}(C_2)$  are different (as  $\alpha$  is functional on  $N_S$ ). Then, by associativity of  $\mu$ -sum, the sum in (2) is equal to

$$\sum_{D \in (\alpha^{-1}(\cup_{C \in Y} C)) / \simeq_S}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D). \tag{3}$$

Consider now the set of nodes

$$Y' = \{w \in N_T \mid T, w \models \psi\}.$$

As  $\simeq_T$  is compatible with  $Bool(\mathcal{P})$  (Lemma 62) and  $\psi$  is a formula in  $Bool(\mathcal{P})$ , we have that if two nodes  $w, w'$  are in some  $C \in N_T / \simeq_T$ , then  $T, w \models \psi$  if, and only if,  $T, w' \models \psi$ . We

deduce that if  $Y'$  contains some node  $w$  of the group-equivalence class  $C$ , then  $C \subseteq Y'$ . Thus,  $Y = Y' / \simeq_T$ , or, equivalently,  $\bigcup_{C \in Y} C = Y'$ . Now,  $\alpha$  being a morphism,  $\alpha^{-1}(Y')$  is the set of nodes  $X'$ :

$$X' = \{w \in N_S \mid T, \alpha(w) \models \psi\}.$$

Therefore, the sum in (3) is equivalent to

$$\sum_{D \in X' / \simeq_S}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D). \quad (4)$$

Now, by preservation of  $Bool(\mathcal{P})$ , we have that  $D \in X$  implies that  $D$  is in the set

$$\{D' \in N_S / \simeq_S \mid \forall w \in D'. T, \alpha(w) \models \psi\}$$

and it is easy to see that then  $D \in X' / \simeq_S$ . That is, the sum in (4) has more components than the sum  $\sum_{D \in X}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D)$ , which by hypothesis we know being greater than  $\lambda$ . As the sum in (4) is equivalent to  $\sum_{C \in Y}^{\mu} \text{mult}_T^{\circ}(\alpha(v), \mathbf{a}, C)$ , we conclude that the latter is greater than  $\lambda$ , thus  $\psi$  is preserved.

## E.2 Reflection

Assume that  $\alpha$  reflects  $\mathcal{P}$ , and let  $\phi$  be a  $\mathcal{M}(\mathcal{P})$  formula without negation s.t.  $T, \alpha(v) \models \phi$ . We show that  $S, v \models \phi$  and the proof goes by induction on the structure of  $\phi$ . For the cases  $\phi = \mathbf{t}$ ,  $\phi$  in  $Bool(\mathcal{P})$ ,  $\phi = \phi_1 \vee \phi_2$  and  $\phi = \phi_1 \wedge \phi_2$ , the proof goes as for preservation.

For  $\phi$  being a modality formula, the proof is close to the proof of preservation. However, there is a particular point on which one has to pay attention, due to the asymmetry in the hypotheses of the proposition, namely, we have a hypothesis for compatibility of  $\simeq_T$  with  $\gamma_T$ , but there is no similar hypothesis for  $\simeq_S$  and  $\gamma_S$ . Therefore, we briefly remind the steps of the proof that are common with the proof for preservation, and then do the remaining part.

Let  $\phi = \mathbf{a}^{\lambda} \psi$  for some  $\psi$  in  $Bool(\mathcal{P})$  without negation. The relation that we have to show is  $\sum_{D \in X}^{\mu} \text{mult}_S^{\circ}(v, \mathbf{a}, D) \geq \lambda$  using that  $\sum_{C \in Y}^{\mu} \text{mult}_T^{\circ}(\alpha(v), \mathbf{a}, C) \geq \lambda$ , where the sets  $X$  and  $Y$  are as for preservation. As for preservation, we can establish that  $\sum_{C \in Y}^{\mu} \text{mult}_T^{\circ}(\alpha(v), \mathbf{a}, C)$  is equal to the sum in (4). It is then enough to show that

$$(\star) \quad \text{if } D \in X' / \simeq_S, \text{ then } D \in X.$$

Consider the set  $X''$

$$X'' = \{w \in N_S \mid S, w \models \psi \text{ and } T, \alpha(w) \models \psi\}.$$

By reflection of  $Bool(\mathcal{P})$ , and  $\psi$  being a  $Bool(\mathcal{P})$  formula, we deduce that  $D \in X' / \simeq_S$  implies  $D \in X'' / \simeq_S$ . Using the definition of  $X$ , one can see that then for  $(\star)$  it is enough to show

$$(\star\star) \quad \text{if } D \in X'' / \simeq_S, \text{ then } D \in N_S / \simeq_S.$$

It remains to show that the set  $X''$  contains only entire classes of nodes for  $\simeq_S$ . That is where the condition  $T, \alpha(w) \models \psi$  in the definition of  $X''$ , which may seem redundant, is used to compensate the asymmetry of the hypotheses pointed out before. Let us show that if  $w \simeq_S w'$ , and  $S, w \models \psi$ , and  $T, \alpha(w) \models \psi$ , then  $S, w' \models \psi$ , which is sufficient for  $(\star\star)$ .

From  $\alpha$  being a shape morphism, we know that  $w \simeq_S w'$  implies  $\alpha(w) \simeq_T \alpha(w')$ . By compatibility of  $\simeq_T$  with  $Bool(\mathcal{P})$  (Lemma 62), it follows that  $T, \alpha(w) \models \psi$  if, and only if,  $T, \alpha(w') \models \psi$ . By hypothesis,  $T, \alpha(w) \models \psi$ , then also  $T, \alpha(w') \models \psi$ . Finally, by reflection of  $\psi$ , we deduce that  $S, w' \models \psi$ .

### E.3 Preservation and reflection

We finally show preservation and reflection in presence of the additional hypothesis that  $\alpha$  preserves and reflects  $\mathcal{P}$ . The proof is very similar as for the previous cases, and goes by induction on the structure of  $\phi$ . More precisely, we show that for any formula  $\phi$ ,  $S, v \models \phi$  if, and only if,  $T, \alpha(v) \models \phi$ . For the base case, if  $\phi$  is  $\mathbf{tt}$  it is trivial, and if  $\phi$  is a formula from  $Bool(\mathcal{P})$ , the result follows from Lemma 61. For the induction, if  $\phi$  is a forward or backward modality formula, the proof goes on the same way that the proof for preservation and reflection. For  $\phi = \phi_1 \vee \phi_2$ , the result is an immediate consequence of the definition of satisfaction. The only remaining case is  $\phi = \neg\phi'$ , for which we use that preservation of the negated formula  $\neg\phi'$  is equivalent to reflection of the sub-formula  $\phi'$ . That is,  $S, v \models \neg\phi$  if, and only if,  $S, v \not\models \phi$  if, and only if, by induction hypothesis,  $T, \alpha(v) \not\models \phi$  if, and only if,  $T, \alpha(v) \models \neg\phi$ .

### F Proof of Lemma 44

The result of the following lemma is used without proof in Lemma 44.

**Lemma 63.** *Let  $U''$  be a shape that admits a level  $i$  neighbourhood shape morphism, and let the shape  $U'$  be obtained from  $U''$  by removing some unique node labels. Then  $U'$  also admits a level  $i$  neighbourhood shape morphism. Moreover, for all node  $v$  in  $N_{U'}$  and for all  $1 \leq j \leq i$ ,  $[v]_{\sim_j}$  in  $U''$  is included into  $[v]_{\sim_j}$  in  $U'$ .  $\blacktriangleleft$*

*Proof.* We denote  $N$  the set of nodes of  $U'$  and  $U''$ , and we denote  $[v]_j^U$  the equivalence class of the node  $v$  for the equivalence relation  $\sim_j$  in the shape  $U$ , where  $U$  may be one of  $U'$  or  $U''$ . We show, by induction on  $j$  for  $j \in 1..i$ , that

$$IH(j) \quad \sim_j \text{ is defined on } U' \text{ and for all } v \text{ in } N, [v]_{\sim_j}^{U''} \subseteq [v]_{\sim_j}^{U'}$$

For the base case,  $IH(0)$  immediately follows from definitions of  $\sim_0$  and the shapes  $U'$  and  $U''$ .

For the induction step, let  $j > 0$ . Remind that  $\sim_j$  is defined in  $U'$  if  $\simeq_{U'} \subseteq \sim_{j-1}$ , or, equivalently, for all node  $v$ ,  $[v]_{\simeq_{U'}} \subseteq [v]_{\sim_{j-1}}^{U'}$ . By definition of  $U''$  we know that  $[v]_{\simeq_{U'}} = [v]_{\simeq_{U''}}$ . As  $U''$  admits a level  $i$  neighbourhood shape morphism, we have  $[v]_{\simeq_{U''}} \subseteq [v]_{\sim_{j-1}}^{U''}$ , and by  $IH(j-1)$ ,  $[v]_{\sim_{j-1}}^{U''} \subseteq [v]_{\sim_{j-1}}^{U'}$ .

Next we have to show that  $[v]_{\sim_j}^{U''} \subseteq [v]_{\sim_j}^{U'}$ . Suppose  $v \sim_j v'$  in  $U''$  and let us show that also  $v \sim_j v'$  in  $U'$ . By definition, this latter holds if, and only if, for all  $C \in N / \sim_{j-1}$ , and for all label  $\mathbf{a}$ ,

$$\sum_{K \in N / \simeq_{U'} \mid K \subseteq C} \text{mult}_{U'}^{\circ}(w, \mathbf{a}, K) = \sum_{K \in N / \simeq_{U''} \mid K \subseteq C} \text{mult}_{U'}^{\circ}(w', \mathbf{a}, K)$$

and analogously for incoming edges multiplicity function. By  $IH(j-1)$ , the set  $C$  is the union of disjoint sets  $C_1, \dots, C_n$  that are all equivalence classes for  $\sim_{j-1}$  in  $U''$ , i.e.,

$$\sum_{K \in N / \simeq_{U'} \mid K \subseteq C} \text{mult}_{U'}^{\circ}(w, \mathbf{a}, K) = \sum_{l \in 1..n} \sum_{K \in N / \simeq_{U''} \mid K \subseteq C_l} \text{mult}_{U'}^{\circ}(w, \mathbf{a}, K)$$

and the same for  $w'$ . This is well defined because the shapes  $U'$  and  $U''$  have the same nodes and edges, the same multiplicities, and the same grouping relation. As  $\sim_j$  is defined on  $U''$ , for any  $C_l$  the equality

$$\sum_{K \in N / \simeq_{U''} \mid K \subseteq C_l}^{\mu} \text{mult}_{U''}^{\circ}(w, \mathbf{a}, K) = \sum_{K \in N / \simeq_{U''} \mid K \subseteq C_l}^{\mu} \text{mult}_{U''}^{\circ}(w', \mathbf{a}, K)$$

holds, thus the equality of the whole sum holds.  $\square$

Consider now the shape  $T''$  as described in the sketch of the proof. We have to show that  $T''$  admits a level  $i$  neighbourhood shape morphism. That is, it is enough to show that  $\sim_j$  is defined in  $T''$  for all  $j \in 1..i$  ( $\sim_0$  being always defined). We show that for all  $v \in N_{S''}$ ,  $[v]_{\sim_j}$  in  $T''$  is equal to  $[v]_{\sim_j}$  in  $S''$ . This is enough as  $\simeq_{T''}$  and  $\simeq_{S''}$  coincide on nodes in  $N_{S''}$ , thus  $\simeq_{S''} \subseteq \sim_j$  in  $S''$  would imply  $\simeq_{T''} \subseteq \sim_j$  in  $T''$ , and nodes in  $N^{\text{new}}$  have also singleton classes for the equivalence relations  $\simeq_{T''}$  and  $\sim_j$  (the latter because they contain fresh labels).

The proof that  $[v]_{\sim_j}$  in  $T''$  is equal to  $[v]_{\sim_j}$  in  $S''$  is quite technical, but does not use any difficult idea. We rather present it in an intuitive way. Suppose that  $v \sim_j v'$  in  $S''$ . Then

1. either  $v$  and  $v'$  are both at distance  $j$  or less from some node in  $c(L'') \cup N^{\text{new}}$  and their equivalence class is influenced by one or more of the fresh labels,
2. or  $v$  and  $v'$  are both far away from  $c(L'') \cup N^{\text{new}}$ .

Remark now that if a node  $w$  is at distance  $d$  from  $c(L'')$  in  $S''$ , then it is at distance at least  $d$  from  $c(L'') \cup N^{\text{new}}$  in  $T''$ . So, such a node may join the equivalence class of some other nodes.

## G Proof of Lemma 54

**Lemma 54.** *Two nodes  $v, v'$  of a graph  $G$  are  $i$ -neighbourhood equivalent if, and only if, the same  $\mathcal{L}_i(\text{Lab})$  formulae hold in  $v$  and in  $v'$ .  $\blacktriangleleft$*

The proof goes by induction on  $i$ . For brevity, we write  $v \models \phi$  instead of  $G, v, \gamma \models \phi$  as the graph  $G$  and the valuation  $\gamma$  are fixed.

*For the base case.* We have  $i = 0$ . For the  $\Rightarrow$  direction, assume that  $v \equiv_0 v'$ . Then, by definition,  $\text{lab}(v) = \text{lab}(v')$ . Let  $\phi$  be a  $\mathcal{L}_0$  formula; by an easy induction on the structure of  $\phi$  one can show that  $v \models \phi$  if, and only if,  $v' \models \phi$ . For the  $\Leftarrow$  direction, assume that it is not the case that  $v \equiv_0 v'$ . Then we easily deduce that there is formula  $\phi$  which holds in one of  $v, v'$  but not in the other; it is sufficient to take  $\phi = \mathbf{a}$  where  $\mathbf{a}$  is a label in  $\text{lab}(v) \cup \text{lab}(v')$  but not in  $\text{lab}(v) \cap \text{lab}(v')$  and we know by assumption that this label exists.

*For the induction.* We have  $i > 0$ . For the  $\Rightarrow$  direction, assume that  $v \equiv_i v'$ . We show that for any  $\mathcal{L}_i$  formula  $\phi$ ,  $v \models \phi$  if, and only if,  $v' \models \phi$  and the proof goes by induction on the structure of  $\phi$ . The only interesting cases are for  $\phi$  being  $\mathbf{a}^\lambda \cdot \phi'$  and  $\langle \mathbf{a} \rangle^\lambda \cdot \phi'$ . Let us show it for  $\mathbf{a}^\lambda \cdot \phi'$ , the other case is symmetrical. So, let  $\phi$  be the formula  $\mathbf{a}^\lambda \cdot \phi'$ . Then, by definition of the satisfaction relation,  $v \models \phi$  if, and only if,  $|v \triangleright^{\mathbf{a}} \cap S_{\phi'} \triangleleft^{\mathbf{a}}|_{\mu} \geq \lambda$  and  $v' \models \phi$  if, and only if,  $|v' \triangleright^{\mathbf{a}} \cap S_{\phi'} \triangleleft^{\mathbf{a}}|_{\mu} \geq \lambda$ , where  $S_{\phi'}$  denotes the set of nodes of  $G$  in which  $\phi'$  holds. We show

that  $|v \triangleright^a \cap S_{\phi'} \triangleleft^a|_\mu = |v' \triangleright^a \cap S_{\phi'} \triangleleft^a|_\mu$ , which allows us to conclude that  $v \models \phi$  if, and only if,  $v' \models \phi$ .

Let  $N / \equiv_{i-1}$  denote the set of equivalence classes of nodes of  $G$  induced by the  $\equiv_{i-1}$  equivalence relation. Let, for any  $C$  in  $N / \equiv_{i-1}$ ,  $F_C$  denote the set of  $\mathcal{L}_{i-1}$  formulae that hold in the nodes in  $C$  :  $F_C = \{\psi \in \mathcal{L}_{i-1} \mid \forall v \in C : v \models \psi\}$ . Then, as  $\phi'$  is an  $\mathcal{L}_{i-1}$  formula and using the induction hypothesis on  $i$ , it is easy to see that  $S_{\phi'}$  is the set  $\bigcup_{C \in N / \equiv_{i-1} \mid \phi' \in F_C} C \triangleleft^a$ . In this case, by distributivity of the multiplicity function over set union, we deduce that  $|v \triangleright^a \cap S_{\phi'} \triangleleft^a|_\mu = \sum_{C \in N / \equiv_{i-1} \mid \phi' \in F_C} |v \triangleright^a \cap C \triangleleft^a|_\mu$ . Now, by assumption we have that  $v \equiv_i v'$ , so by definition of the  $\equiv_i$  equivalence relation we have that for any  $C$  in  $N / \equiv_{i-1}$ ,  $|v \triangleright^a \cap C \triangleleft^a|_\mu = |v' \triangleright^a \cap C \triangleleft^a|_\mu$ . Thus,  $|v \triangleright^a \cap S_{\phi'} \triangleleft^a|_\mu = \sum_{C \in N / \equiv_{i-1}} |v' \triangleright^a \cap C \triangleleft^a|_\mu$  and this last quantity is equal to  $|v' \triangleright^a \cap C \triangleleft^a|_\mu$ .

For the  $\Leftarrow$  direction,

Assume  $v \not\equiv_{i+1} v'$ . If this is the case, we know that there exist a label  $\mathbf{a} \in \mathbf{Lab}$  and an equivalence class  $C \in N_G / \equiv_i$  for which either  $o_v = |v \triangleright^a \cap C \triangleleft^a|_\mu \neq |v' \triangleright^a \cap C \triangleleft^a|_\mu = o_{v'}$  or  $i_v = |v \triangleleft^a \cap C \triangleright^a|_\mu \neq |v' \triangleleft^a \cap C \triangleright^a|_\mu = i_{v'}$ . For the moment, let us assume that there exists a formula  $\psi \in \mathcal{L}_i$  that only one such  $C$  satisfies. Let  $\phi$  be:

$$\begin{aligned} & \rangle \mathbf{a} \rangle^{max(o_v, o_{v'})} . \psi \text{ if } o_v \neq o_{v'} \\ & \langle \mathbf{a} \langle^{max(i_v, i_{v'})} . \psi \text{ otherwise} \end{aligned}$$

As a) no other  $\equiv_i$ -equivalence class satisfies  $\psi$ , and b) all nodes in  $C$  satisfy  $\psi$  (by inductive hypothesis); we can deduce that  $C$  is exactly the set of nodes that satisfies  $\psi$ . Thereby,  $\phi$  is satisfied by only one of  $v$  or  $v'$ . To show that such a  $\psi$  exists, consider the following: for each  $C' \in N_G / \equiv_i$ , s.t.  $C' \neq C$ , we can find a  $\psi_{C'} \in \mathcal{L}_i$  s.t.  $C \models \psi$  and  $C' \not\models \psi$ <sup>7</sup>, also due to the induction hypothesis. So we can take  $\psi$  to be  $\psi = \bigwedge_{C' \in N_G / \equiv_i \mid C' \neq C} \psi_{C'}$

<sup>7</sup> Slight abuse of notation.