Relational Treatment of Term Graphs With Bound Variables

WOLFRAM KAHL, *Department of Computer Science, German Armed Forces University Munich, D-85577 Neubiberg.*
E-mail: kahl@informatik.unibw-muenchen.de

Abstract

We show how and why it makes sense to use a relational formalisation instead of the usual functional one in the treatment of term graphs. Special attention is paid to term graphs with bound variables, that have, to our knowledge, never been formalised with such a generality before.

Besides the novel treatment of term graphs themselves, we present an innovative relational homomorphism concept that for the first time allows to consider terms, resp. term trees as a special case of term graphs and still have the full power of (second-order) substitution available.

Keywords: Term graph rewriting, relational matching, relation calculus, λ-calculus

1 Introduction

Term graphs have probably been introduced into the literature by Wadsworth [42], the inventor of graph reduction, which is an important implementation technique for functional programming languages. Since then, term graphs have been used as an efficient implementation of terms, and various flavours of term graph rewriting have been investigated mainly as implementations of term rewriting. Interestingly, the flavour introduced by Wadsworth already implements λ-calculus, and is therefore concerned with bound variables.

Most of the term graph rewriting approaches studied in the recent literature, as for example in the book [37], however, only implement conventional term rewriting systems [12] without any notion of variable binding, and never approach the expressivity of Klop's "Combinatory Reduction Systems" (CRSs) [28].

An important reason for this is the shift towards categorical, "algebraic" methods of rewriting; graph reduction was defined only operational. For these algebraic approaches a notion of term graph homomorphism is necessary, and although many researchers have stated in their publications that they wanted to extend their respective approaches for being able to cover term graphs with bound variables and graph reduction, no such extensions have been forthcoming.

Only after having developed the first satisfactory approach to algebraic term graph rewriting with bound variables with the expressivity of CRSs in [20, 21], we are able to see the reasons why this result has been delayed so long.

On one hand, a completely new, and far more abstract formalisation of term graphs with different kinds of variables was necessary as a basis for a manageable concept of homomorphisms. Even so, the proofs of the compositionality of the different properties of homomorphisms are extremely complicated. The fact that relation algebraic notation and concepts have been available to the author at the time of undertaking that task has brought a powerful tool to bear on a difficult problem, and we strongly feel that only since our treatment of term graphs with bound variables used relation algebraic notation from the beginning, we have been able to complete the proofs in a way that is still readable and at the same time sufficiently formal to support confidence in the correctness.

On the other hand, the algebraic approach to term graph rewriting itself [34, 26, 27, 30, 4] had to be extended, but that is outside the scope of the present article.

This paper provides a more gentle introduction to the fundamental rationale behind the relational formalisation of term graphs developed in [20, 21], and goes on to extend this formalisation even more into the relational direction, opening up new application possibilities for this approach.

The motivation for our interest in term graph rewriting is due to the development of the graphically interactive program transformation system HOPS[1] [47, 18, 5], which implements second-order rewriting of acyclic term graphs with bound variables.

Overview

After this introductory section, we first present a tutorial introduction to term graphs with bound variables in Sect. 2. There we give a more or less informal introduction to the design decisions that shape our formalisation of term graphs with bound variables.

Section 3 introduces the relation algebraic notation and concepts we are going to use for our formalisation of term graphs.

We then report the basic definitions concerning term graphs from [20, 21] in Sect. 4, however, we also adjust our formalism slightly towards later extensions and most notably provide a completely reworked treatment of intervals. Section 5 builds on this to present a generalisation of a simple version of term graph homomorphisms, now based on *relations* carrying *explicit* information about the images of metavariables, and not only on *functions* with *implicit* information as in [20, 21]. The main result is the presentation of a category of such explicit relations.

In Sect. 6 we show the ease with which the definitions of [20, 21] concerning matching of rule sides can be adapted to the new relational setting.

2 Gentle Introduction to Term Graphs

The purpose of this section is to motivate the way we formalise term graphs with bound variables.

Terms can be considered as trees, i.e., as a special kind of graphs, in a natural way, which we discuss in 2.1. We choose to introduce our view on variable binding and variable identity while still talking about trees, in order to present the unfamiliar aspects in front of a more familiar background.

From trees it is only a small step to directed acyclic graphs (DAGs), taken in 2.2. We discuss some motivation for considering cyclic term graphs in 2.3. In 2.4, we sketch the framework of algebraic rewriting only for providing some background for a few concepts that will crop up later, and in 2.5 we motivate the move from functional to relational matchings from their use in rewriting.

[1] partially supported by the German Research Association DFG

2.1 Term Trees

Terms can be considered as labelled trees[2]. For example, the term $(-a) * (b + b)$ can be considered as the following labelled tree:

$$\begin{tikzcd}
& * \arrow[dl] \arrow[dr] & & \\
- \arrow[d] & & + \arrow[dl] \arrow[dr] & \\
a & b & & b
\end{tikzcd}$$

Strictly speaking, a, b, $*$, $+$ and $-$ are *node labels* of nodes whose identifiers we do not care about, and the edges are labelled with the ordinal number corresponding to the position of their appearance in the drawing among the outgoing edges of their source node. This labelling is necessary because otherwise for example the following two trees would be indistinguishable, i.e., isomorphic (as graphs), while the corresponding terms $a - b$ and $b - a$ are clearly different:

$$\begin{tikzcd}
& - \arrow[dl] \arrow[dr] & & & & - \arrow[dl] \arrow[dr] & \\
a & & b & & b & & a
\end{tikzcd}$$

The idea behind regarding terms as labelled trees is moving the *structure* of terms into the graph level, while leaving the *contents* in the node labels.

With ordinary terms, the only structure to be considered is the immediate constituent relation — or the relation between occurrences of function symbols (*constructors*) and their arguments — together with the argument ordering.

The node labels themselves do not reflect structure, but content.

When we move to the metaterms of CRSs or any kind of terms with bound variables, then, first of all, variable binding has to be considered. In the λ-term $t_1 := \lambda x.\, y$, the variable x is not a subterm of t_1; the only nontrivial subterm is y. Therefore, in the tree representation there should be no node representing x. One way to achieve this was already used by Wadsworth, the inventor of graph reduction [42], who would have represented t_1 by a two-node term graph with the root labelled "λx" and its successor "y":

$$\begin{tikzcd}
\lambda x \arrow[d] \\
y
\end{tikzcd}$$

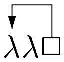
The problem with this approach is, that the variable binding, which is an essential ingredient of the structure of the λ-term, is not reflected in the structure of the graph, but has been coded into the node labels which originally should only carry structure-independent contents.

The same effect results from working with de Bruijn indices. Here, the two nodes of the graph representing t_1 would be labelled "λ" and "y" respectively, whereas the tree for the λ-term $\lambda x.\, x$, or $\lambda 1$ in de Bruijn notation, would have "λ" and "1" respectively — again, the variable binding structure is coded into the node labels:

[2] When we say "tree", we strictly speaking mean "arbourescence" in the graph-theoretic sense, i.e., a directed acyclic graph where every node except one (the root) has exactly one predecessor.

In all these approaches, the binding is *derived* from the node labels and other non-primitive structure elements such as reachability.

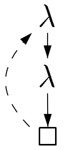
A *primitive binding concept* cleanly separating structure from contents seems to appear only in the foundations of Bourbaki [8].[3] There, "*assemblages*" are considered instead of terms, variables are all denoted by the symbol "□", i.e., there are no variable names, and variable binding is denoted by "*liens*", links drawn above the symbol sequence.[4] An example is



for what usually would be $\lambda\, x\,.\, \lambda\, y\,.\, x$.

From an abstract point of view, the "*liens*" of Bourbaki are an additional graph structure superimposed over the tree structure of terms. When moving from terms to term trees, this additional structure induces a *second kind of edges* between the original term tree nodes.

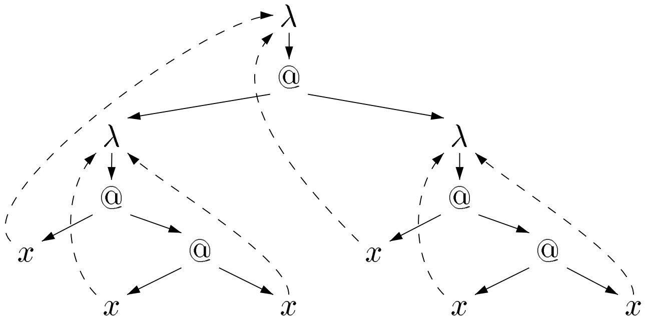
We draw these *binding edges* as dashed, usually curved, edges in our example graphs — only the underlying graph that does not consider binding is still a tree. The above example $\lambda\, x\,.\, \lambda\, y\,.\, x$ then becomes:



This is the approach we want to follow here, and it means that we move the binding structure explicitly onto the graph level, thus introducing a *primitive binding concept.* We shall, however, use the label "x" instead of "□" for bound variables (at least in the λ-calculus).

Trying to represent more λ-terms as enriched term trees like that, one encounters a problem with multiple occurrences of variables. A bound variable that occurs several times is not really a problem, as the following enriched tree shows for the standard fixed point combinator

$$Y = \lambda\, f\,.\, (\lambda\, x\,.\, (f\ \ (x\ x))\ \lambda\, x\,.\, (f\ \ (x\ x)))$$

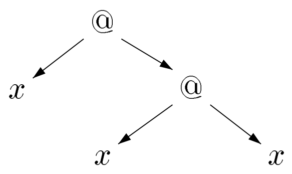


[3] I am grateful to John Staples for this hint.

[4] The rôle assigned to the "*lettres*" in the axioms can be interpreted as the view that, syntactically speaking, there are no free variables — "*lettres*" are something essentially different from the variables denoted by the symbol "□" which is only allowed to occur bound.

Here, variable nodes bound at the same λ-node belong to the same variable, the same way as in Bourbaki, and, as is generally the case in λ-calculus, different bound variables are bound at different λ-nodes.

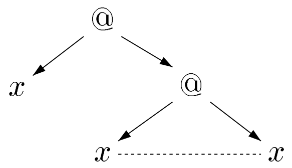
There is, however, corresponding to the absence of unbound occurrences of "□" in Bourbaki, no way to distinguish different free variables — the subterm $(f\ (x\ x))$ of the Y-combinator represented as the subtree



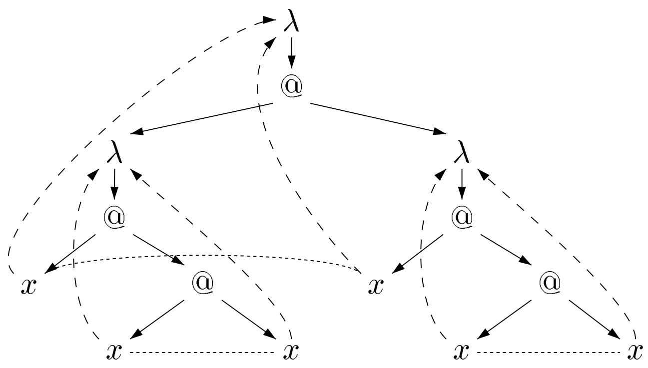
obviously has lost some structure that had not been made explicit before, namely the structure of **variable identity**. If we proceed to add that structure to our graphs, too, we advance beyond Bourbaki in the homogeneity and universality of our approach.

Variable identity obviously has to be a partial equivalence relation on the tree nodes, and has to affect exactly the variable nodes. It can be seen as partitioning the set of all *variable nodes* into equivalence classes that can each be considered as one *variable*.

We shall usually not draw the reflexive part of variable identity, which can be deduced from the labels (i.e., every variable node stands in the relation of variable identity with itself), and only a skeleton of its irreflexive part. In the enriched tree corresponding to $(f\ (x\ x))$, the relation of variable identity thus partitions the three variable nodes into two variables corresponding to f and x respectively, the latter consisting of two variable nodes:



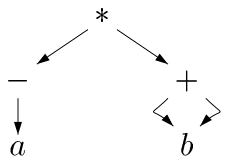
Variable identity should then also be made explicit for bound variables, and our term tree for the fixed point combinator changes to the following:



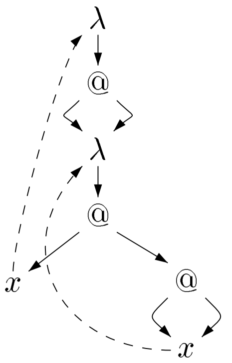
2.2 From Terms to DAGs

Identification of equal subterms is a means of efficiently representing terms in a data structure. This identification replaces several isomorphic subtrees by only one of them and redirects all edges that pointed to one of those subtrees to the selected copy, thus turning the tree into a *directed acyclic graph* (DAG).

Applied to our introductory example, identification of the two b nodes yields:

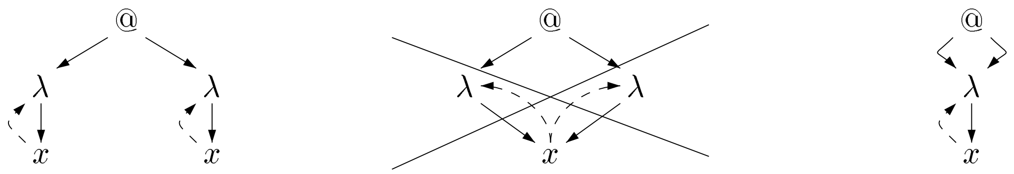


The fixed point combinator offers even more sharing opportunities:



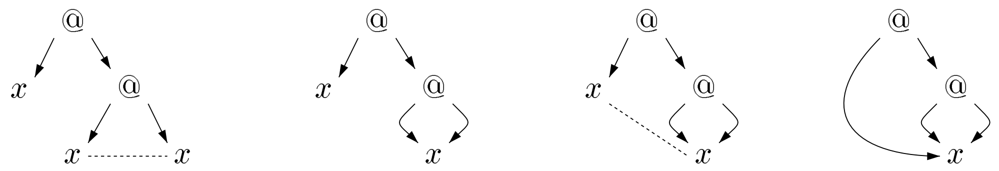
Noticeably, variable identity here becomes trivial again since we always have been able to share different variable nodes.

An important point to be aware of is that bound variables can only be shared together with their binder, making the DAG drawn in the middle illegal:



The graph-theoretic formulation for the condition behind this effect is that binding has to be univalent and that the binder must always *dominate* any bound variables.

For an analogous reason, different free variables may not be shared, either:



Although in this example all four DAGs are legal, the first two correspond to the term $(f\ (x\ x))$ containing two different variables as in the last section, while the last two correspond to the term $(x\ (x\ x))$ with only one variable — these are just different terms, so that there is no legal sharing step leading from one to the other.

In all the examples seen so far, sharing could make the variable identity trivial. This ceases to be the case as soon as we start to consider variables that are not zero-ary. The most prominent example of such variables are the metavariables of combinatory reduction systems.

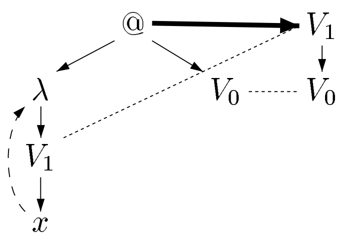
The rule of β-reduction is usually written essentially as

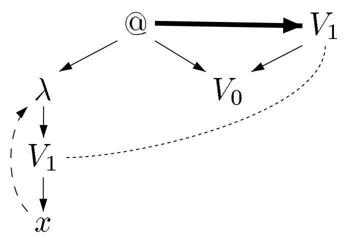
$$(\lambda\, x\,.\, V_1(x))\ V_0 \quad\longrightarrow\quad V_1(V_0)$$

with metavariables V_0 and V_1, which are the parameters of CRS rules. For making its structure more explicit, we also present it in strict prefix notation with an explicit application symbol:

$$@(\lambda\, x\,.\,(V_1(x)), V_0) \quad\longrightarrow\quad V_1(V_0)\ .$$

For representing rules, it will usually be necessary to have graphs (or trees) with more than one root; in the drawings we connect the sources of the rule sides with a thick arrow. As term forest resp. as term DAG the β-rule then takes the following shape:





No further sharing is possible, since the two different V_1-nodes have different successors.

Therefore, variable identity is in general not superfluous in term graphs.

2.3 Cyclic Term Graphs

Cyclic term graphs can be used to code recursion, and this has been done for quite some time in implementations of functional programming languages.

Recently, the topic has become the subject of closer investigation in the context of rational term rewriting systems [25, 17, 11].

Although HOPS always only accepted acyclic term graphs, we drop that restriction for our investigations and always include the possibility of cycles into our considerations. It turns out that this generality forces us to introduce extra concepts in cases where on acyclic graphs there is no need to differentiate. Thus the more general approach rewards us with deeper insight into the underlying structure of variable binding and variable identity.

2.4 Algebraic Term Graph Rewriting

Term graph rewriting has been introduced in the operational definition of graph reduction by Wadsworth [42]; since the development of the algebraic approach to rewriting [14], however, most new developments have come from within this algebraic approach.

The original version of the algebraic approach is founded upon the double-pushout rewriting scheme:

$$\begin{tikzcd} L \arrow[d] & G \arrow[l] \arrow[r] \arrow[d] & R \arrow[d] \\ A & H \arrow[l] \arrow[r] & B \end{tikzcd}$$

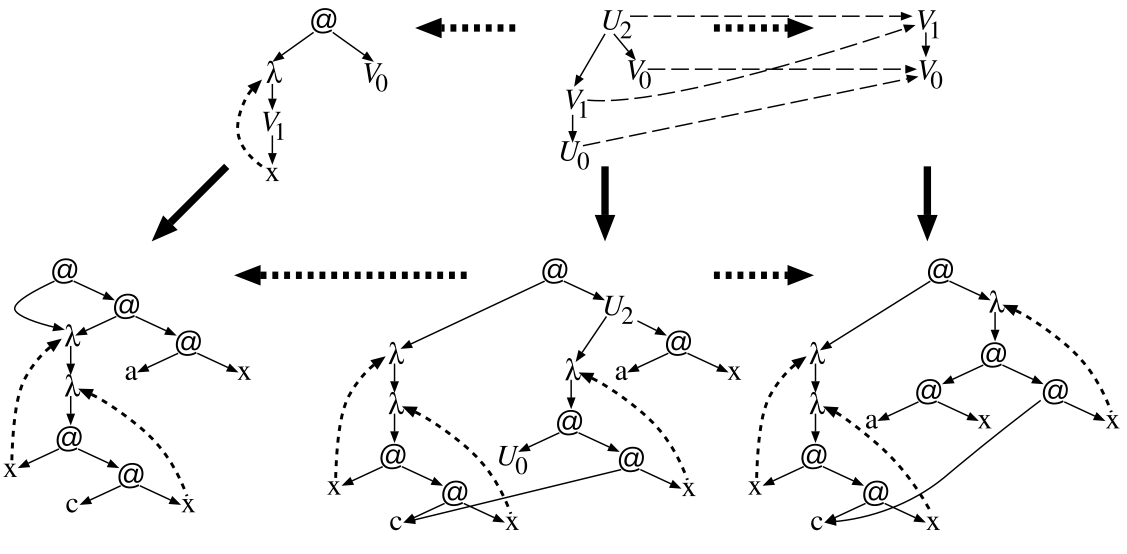
Here a rule $L \longleftarrow G \longrightarrow R$ transfers its rewriting information to an application graph A via two pushouts.

However, all attempts to extend algebraic treatments of simple term graphs to term graphs with bound variables have been futile so far.

Only in [21] a variant of the algebraic approach that makes an algebraic treatment of graph reduction possible was developed, the so-called *fibred approach to rewriting* (see also [23]).

The fibred approach is a *heterogeneous* approach in that the rule arrows and the vertical matching arrows may come from different subclasses of arrows of one underlying category; notably matchings themselves need not form a category in this approach; i.e., the composition of two matchings does not have to be a matching, too.

As an example rewriting step we show an application of the rule of β-reduction (the reduced redex starts at the right successor of the root node):



The schemavariables V_0 and V_1 have already been seen in the term graph rule for β-reduction in 2.2 above; the vertical matching homomorphisms map these schemavariables to *intervals* in their target graphs, i.e., parts of the graph starting at the immediate image node of the schemavariable node and ending before the image nodes of its successors — these intervals have to be restricted in various ways to avoid "*capture of variables*". New are the placeholders U_0 and U_2 in the intermediate graphs; these placeholders suffer almost no restriction as to their images under horizontal arrows.

The fibred rewriting step takes care to duplicate the part of the λ-body that is not "abstractable" in the nomenclature of Wadsworth just by virtue of the restriction posed on the possible vertical arrows.

2.5 Relational Matching

Above and in the literature, term graph morphisms are *total functions* between the node sets of graphs.

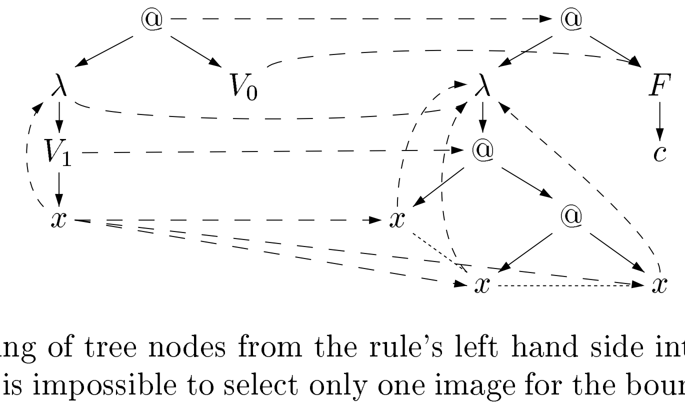
When regarding term graph rewriting as an implementation of term rewriting, however, and accordingly rewriting of our term graphs with bound variables as an implementation of CRSs, then the totality and also the univalence both turn out to induce major shortcomings.

The most important limitation of the approach of [20, 21] is that one term graph rule for β-reduction only implements the β-reduction of λI-calculus, where a bound variable *has to occur* in the body of every λ-abstraction. The λK-calculus, which is the λ-calculus usually employed today, does not know this restriction, so it might seem natural to introduce a special class of partial homomorphisms that solves this problem. Just consider the case of the β-reduction; which, when applied to a redex like $(\lambda x \,.\, y)\, z$, yields a matching of nodes from the rule's left hand side into the redex tree that is not total, but only a *partial function*.

However, we can present an even more general, and, from the point of view of relational methods as well as from the point of view of implementing term rewriting systems, an also far more satisfying solution: Instead of just considering a special class of partial functions as term graph homomorphisms, we can consider a special class of general relations.

Essentially this is based on the following: Term graphs need not always guarantee maximal sharing, and so sometimes a term graph rule is not immediately applicable in a certain term graph while the corresponding term rule is applicable in the corresponding term — on the graph side, a sharing step frequently brings applicability, see e.g. [33], where sharing steps are interleaved with rewriting steps in order to achieve confluence results for (first-order) term graph rewriting systems that are compatible to the behaviour of the corresponding term rewriting systems.

The underlying principle can be better understood when considering term trees instead of conventional terms; just consider the case of the β-reduction in the term tree context, and the following redex:



Here, the matching of tree nodes from the rule's left hand side into the redex tree is not univalent (it is impossible to select only one image for the bound variable because of variable control among other conditions — see Def. 6.2); it is only a *relation*.

This paper strives to find an appropriate concept of *relational matching* for general term graphs and introduces as its main contribution a concept of simple "underlying" term graph morphisms in Sect. 5 before adapting the full matching concepts of [20, 21] to the relational case in Sect. 6.

3 Relation Algebra

For the formal presentation we employ the ever more popular notation of relation algebra; we generally use a style akin to that of [36], but in the notation of [9]. This section summarises the essential basics and a few not-so-trivial constructions. Most of the facts that are stated without proof have been taken from [36].

The most important differences between our treatment and that of [36] are the following:

Vectors in [36] usually are row constants, while we use column constants, the same as in [45, 46], which fit more intuitively into the left-to-right way of denoting relational multiplication.

We also usually provide "typing" for our relations, and for this we use the Z-notation [38] for introducing properties of relations at the same time, for example for univalent relations (partial functions, "$\nrightarrow$") or mappings (total functions, "$\rightarrow$"). The symbol "$\mathbb{1}$" serves to denote an arbitrary but fixed one-element set.

Besides that, we shall use Z-notation for writing quantifications and logical operations on the meta level; for set-comprehension, too, we employ Z-notation, which uses the pattern "$\{$ *signature* | *predicate* $\bullet$ *term* $\}$" instead of the otherwise frequently observed pattern "$\{$ *term* | *predicate* $\}$". So we have for example

$$\{n : \mathbb{N} \mid n < 4 \bullet n^2\} = \{0, 1, 4, 9\} \ .$$

If the predicate is constantly true, then we can also just write "$\{$ *signature* $\bullet$ *term* $\}$" in accordance to common Z usage. Quantification uses the same patterns; here most frequently the predicate is omitted, so we have for example $\forall x : \mathbb{N} \bullet x + 1 > x$. In proofs where space is tight and the context clear, we omit the typing in the signature and just introduce the bound variables, as in $\forall R, S \bullet R \sqcap S \sqsubseteq R$.

For application of a (usually relation-valued) function f to an argument x, instead of the mathematical "$f(x)$" we use: "$f.x$", and we also stick to the convention that this function application has higher syntactical priority than all other operators. This convention together with the notation from [9] that introduces "$;$" for relational composition also opens the way to multi-letter symbols, since juxtaposition now has no meaning.

Finally we normally need the point axiom (Def. 3.4), thus essentially staying in the realm of *concrete relation algebras*. We therefore shall not explicitly mention the use of the point axiom.

Nevertheless we introduce all notation from the point of view of *abstract relation algebra*, since the component free way of arguing usually allows more compact formalisations and proofs, and even when resorting to points a homogeneous style is preserved.

3.1 Elementary Relation Algebra

The basic notations are as follows:

DEFINITION 3.1
A (heterogeneous) **relation algebra** is a self-dual category where the homset from A to B is denoted $A \leftrightarrow B$; for two relations $R : A \leftrightarrow B$ and $S : B \leftrightarrow C$, we denote

their composition with $R;S$, and for an object A of the category, $\mathbb{I} = \mathbb{I}_A$ denotes the *identity relation.*

Furthermore every homset $A \leftrightarrow B$ has to be a complete atomic boolean algebra with *union* “$\sqcup$”, *intersection* “$\sqcap$”, *negation* “$\overline{}$”, *inclusion* “$\sqsubseteq$”, *zero element* $\bot\!\!\!\bot = \bot\!\!\!\bot_{A\leftrightarrow B}$ and *universal element* $\top\!\!\!\top = \top\!\!\!\top_{A\leftrightarrow B}$.

The dual arrow $R^\smile : B \leftrightarrow A$ for an arrow (relation) $R : A \leftrightarrow B$ is also called the *converse* (or *transposed*) *relation*, and the *Dedekind rule*

$$(Q;R \sqcap S) \sqsubseteq (Q \sqcap S;R^\smile);(R \sqcap Q^\smile;S)$$

and the *Tarski rule* $R \neq \bot\!\!\!\bot \;\Rightarrow\; \top\!\!\!\top;R;\top\!\!\!\top = \top\!\!\!\top$ hold. □

The **modal rules** are direct consequences of (and equivalent to) the Dedekind rule:

$$Q;R \sqcap S \sqsubseteq (Q \sqcap S;R^\smile);R \qquad\qquad Q;R \sqcap S \sqsubseteq Q;(R \sqcap Q^\smile;S)$$

Also equivalent to the Dedekind rule the **Schröder equivalences** hold:

$$Q;R \sqsubseteq S \;\Leftrightarrow\; Q^\smile;\overline{S} \sqsubseteq \overline{R} \;\Leftrightarrow\; \overline{S};R^\smile \sqsubseteq \overline{Q}$$

Concepts like univalence can be formulated within relation algebra:

Definition 3.2 (Special heterogeneous relations)
For any $A : U \leftrightarrow V$ we define the following concepts:

$$\begin{array}{llllllll}
A \text{ is } \textit{univalent} & :\Leftrightarrow & A^\smile;A \sqsubseteq \mathbb{I} & \Leftrightarrow & A;\overline{\mathbb{I}} \sqsubseteq \overline{A} & & \\
A \text{ is } \textit{total} & :\Leftrightarrow & \mathbb{I} \sqsubseteq A;A^\smile & \Leftrightarrow & A;\top\!\!\!\top = \top\!\!\!\top & \Leftrightarrow & \overline{A} \sqsubseteq A;\overline{\mathbb{I}} \\
A \text{ is a } \textit{total function} & :\Leftrightarrow & A;\overline{\mathbb{I}} = \overline{A} & \Leftrightarrow & \multicolumn{3}{l}{A \text{ is total and univalent}} \\
A \text{ is } \textit{injective} & :\Leftrightarrow & A;A^\smile \sqsubseteq \mathbb{I} & \Leftrightarrow & \overline{\mathbb{I}};A \sqsubseteq \overline{A} & \Leftrightarrow & A^\smile \text{ is univalent} \\
A \text{ is } \textit{surjective} & :\Leftrightarrow & \mathbb{I} \sqsubseteq A^\smile;A & \Leftrightarrow & \top\!\!\!\top;A = \top\!\!\!\top & \Leftrightarrow & A^\smile \text{ is total} \\
A \text{ is } \textit{bijective} & :\Leftrightarrow & \overline{\mathbb{I}};A = \overline{A} & \Leftrightarrow & \multicolumn{3}{l}{A \text{ is injective and surjective}}
\end{array}$$

□

A univalent relation is also called a *(partial) function* (written $A : U \nrightarrow V$), and a total function *mapping* (written $A : U \to V$). We also use the Z notations for injective partial ($A : U \rightarrowtail\!\!\!\!\!\!\nrightarrow V$), injective total ($A : U \rightarrowtail V$), surjective total ($R : A \twoheadrightarrow B$) and bijective total functions ($R : A \rightarrowtail\!\!\!\!\!\twoheadrightarrow B$).

The following properties will be used in proofs without special mentioning, at most noting, e.g., the univalence of a relation concerned:

Lemma 3.3 (Properties)

$$\begin{array}{lllrcl}
A \text{ univalent} & \Rightarrow & A;\overline{B} \sqsubseteq \overline{A;B}, & A;(B \sqcap C) & = & A;B \sqcap A;C, \\
 & & & B;A \sqcap C & = & (B \sqcap C;A^\smile);A \\
A \text{ total} & \Rightarrow & \overline{A;B} \sqsubseteq A;\overline{B} & & & \\
A \text{ mapping} & \Rightarrow & A;\overline{B} = \overline{A;B} & & & \\
A \text{ injective} & \Rightarrow & \overline{B};A \sqsubseteq \overline{B;A}, & (B \sqcap C);A & = & B;A \sqcap C;A, \\
 & & & A;B \sqcap C & = & A;(B \sqcap A^\smile;C) \\
A \text{ surjective} & \Rightarrow & \overline{B;A} \sqsubseteq \overline{B};A & & &
\end{array}$$

□

A relation v with $\mathbb{T} ; v = v$ is called *vector* and can be seen to denote a subset of its codomain. A vector that is a mapping is called a *point* and can be seen to denote an element of its codomain. For the one-element set $\mathbb{1}$, every relation in $\mathbb{1} \leftrightarrow A$ is a vector and every mapping in $\mathbb{1} \to A$ is a point. We assume throughout the validity of the following additional axiom:

DEFINITION 3.4
The **point axiom** is the following proposition:

"For every relation $R \neq \mathbb{L}$ there are two points x and y such that $x^{\smile} ; y \sqsubseteq R$." □

The two following vector-related laws are elementary for practical proofs, so that we do not need to indicate their use:

LEMMA 3.5 (Domain and Range Restrictions)
For all relations $Q : A \leftrightarrow B$, $R : B \leftrightarrow C$, $S : D \leftrightarrow C$ and $U : A \leftrightarrow E$ the following hold:

$$Q ; R \sqcap \mathbb{T} ; S = Q ;(R \sqcap \mathbb{T} ; S) \qquad \text{and} \qquad Q ; R \sqcap U ; \mathbb{T} = (Q \sqcap U ; \mathbb{T}) ; R \ . \qquad \square$$

Restriction of a relational product by intersection with a vector is therefore equal to the product of the two relations where just the second one is intersected with the vector; that vector has of course to be adapted to its new type by multiplication with a universal relation.

Whenever $R : A \leftrightarrow A$ for an appropriate A, then R is called a *homogeneous relation.*

DEFINITION 3.6 (Special homogeneous relations)

R is *reflexive*	:⇔	$\mathbb{I} \sqsubseteq R$
R is *transitive*	:⇔	$R ; R \sqsubseteq R$
R is *symmetric*	:⇔	$R \sqsubseteq R^{\smile}$
R is an *equivalence relation*	:⇔	R is a reflexive, transitive and symmetric □

For a homogeneous relation $R : A \leftrightarrow A$, its *transitive closure* is denoted by R^+ and its *reflexive transitive closure* by R^*. Using the supremum of sets of relations and Z-style set comprehension we can, for example, write: $R^+ = \bigsqcup\{k : \mathbb{N} \mid k > 0 \bullet R^k\}$.

3.2 Termination

Every homogeneous relation can be considered to be the associated relation of a (simple) directed graph, i.e. the relation that associates one node x with another node y exactly if there is an edge from x to y. Here the question about infinite paths corresponding to infinite processes in transition graphs arises. For a detailed exposition see [36, Section 6.3].

The first concept to be introduced here corresponds to the question about the existence of paths of arbitrary length:

DEFINITION 3.7
A homogeneous relation R is called **progressively bounded**, if

$$\bigsqcup\{k : \mathbb{N} \bullet \overline{R^k ; \mathbb{T}}\} = \mathbb{T} \ . \qquad \square$$

R is therefore progressively bounded if and only if for every node the length of the R-paths leaving that node is upwards bounded, since then for every node a natural number k exists such that there is no path of length k starting in that node. However, there still need not exist infinite paths in a relation that is not progressively bounded

For distinguishing whether there are infinite paths in a graph we need the following concept:

DEFINITION 3.8
The **initial part** $J(R) : \mathbb{1} \leftrightarrow V$ of a homogeneous relation $R : V \leftrightarrow V$ is defined as:

$$J(R) := \bigsqcap\{x : \mathbb{1} \leftrightarrow V \mid \overline{x} = \overline{x} ; R^{\smile}\} = \bigsqcap\{x : \mathbb{1} \leftrightarrow V \mid \overline{x} \sqsubseteq \overline{x} ; R^{\smile}\} \qquad \square$$

The initial part is therefore the vector containing exactly those points from which no infinite paths (according to R) start.

DEFINITION 3.9
A relation R is called **progressively finite**, iff $J(R) = \mathbb{T}$. R is called **regressively finite**, iff $R^{\smile}$ is progressively finite. $\square$

R is therefore progressively finite if and only if there are no infinite paths in R. We also use the following alternative characterisation of progressive finiteness from [36, page 121]:

LEMMA 3.10
A relation $R : V \leftrightarrow V$ is progressively finite if and only if for every vector $v : \mathbb{1} \leftrightarrow V$ the following holds:

$$v \sqsubseteq v ; R^{\smile} \quad \Rightarrow \quad v = \mathbb{L} \qquad \square$$

We now consider equations and inclusions such as $X = A \sqcup B ; X$, and $X \sqsubseteq A \sqcup B ; X$.

$X \sqsubseteq A \sqcup B ; X$ first implies $X \sqcap \overline{B ; \mathbb{T}} \sqsubseteq A$. Through induction we can show that, if B is univalent or bijective, generally

$$X \sqcap \overline{B^j ; \mathbb{T}} \sqsubseteq (\bigsqcup\{k : \mathbb{N} | 0 \leq k < j \bullet B^k\}) ; A$$

holds. Since for progressively bounded B we have: $\bigsqcup\{k : \mathbb{N} \bullet \overline{B^k ; \mathbb{T}}\} = \mathbb{T}$, we obtain:

LEMMA 3.11
If the progressively bounded relation B is univalent or bijective and $X \sqsubseteq A \sqcup B ; X$ holds, then $X \sqsubseteq B^* ; A$ holds, too. $\square$

For the opposite inclusion $X \sqsupseteq A \sqcup B ; X$ trivially $X \sqsupseteq A$ holds, and generally we can prove $X \sqsupseteq (\bigsqcup\{k : \mathbb{N} | 0 \leq k < j \bullet B^k\}) ; A$ for every $j \in \mathbb{N}$. Therefore we have without restriction:

LEMMA 3.12
$X \sqsupseteq A \sqcup B ; X$ implies $X \sqsupseteq B^* ; A$. $\square$

Putting the two together we obtain

LEMMA 3.13
If the progressively bounded relation B is univalent or bijective, then the equality $X = A \sqcup B ; X$ implies $X = B^* ; A$. $\square$

3.3 Relational Specification of Products and Sums

Direct products (of the subcategory of total functions) can neatly be characterised with relation algebraic means:

DEFINITION 3.14 (Direct Product)
For two relations $\pi : P \leftrightarrow A$ and $\rho : P \leftrightarrow B$ we call the triple (P, π, ρ) *direct product* for A and B if the following conditions are fulfilled:

$$\begin{array}{rclcrcl} \pi^{\smile} ; \pi & = & \mathbb{I}_A & \qquad & \rho^{\smile} ; \rho & = & \mathbb{I}_B \\ \pi ; \pi^{\smile} \sqcap \rho ; \rho^{\smile} & = & \mathbb{I}_P & & \pi^{\smile} ; \rho & = & \mathbb{T}_{A,B} \end{array}$$

The surjective mappings π and ρ are called the *natural projections* of this product.□

Since the types we use for our relations usually represent sets and we are restricted to concretely representable relations anyway, we usually employ products of the shape $(A \times B, \pi_{A\times B}, \rho_{A\times B})$ and call this simply $A \times B$. When using for example $\pi_{A\times B}$ without any explanation, then such a product is implied. When the products used can be inferred from the context, we also use plain "π" and "ρ".

For total functions F and G the expression $F ; \pi^{\smile} \sqcap G ; \rho^{\smile}$ corresponds to *function pairing*, the universal arrow in the category theoretic product construction. In relation algebra one can see the names "*fork*" [6] or "*tupeling*" used for this together with the notation "$F \nabla G$". Since we shall need the converse of such an expression about equally often and since $F \triangle G := (F^{\smile} \nabla G^{\smile})^{\smile}$ does not seem to be a very appropriate notation, we rather orient our notation at the direction of writing from left to right instead bottom up as "∇":

DEFINITION 3.15 (Relation Pairings)
For $Q : A \leftrightarrow C$, $R : B \leftrightarrow C$ and $S : A \leftrightarrow D$ we define $\llparenthesis Q, S \rrparenthesis_{C\times D} : A \leftrightarrow C \times D$ and ${}_{A\times B}\llbracket Q, R \rrbracket : A \times B \leftrightarrow C$ as

$$\begin{array}{rclr} \llparenthesis Q, S \rrparenthesis_{C\times D} & := & Q ; \pi^{\smile}_{C\times D} \sqcap S ; \rho^{\smile}_{C\times D} & (\ =\ Q \nabla S\ ) \\ {}_{A\times B}\llbracket Q, R \rrbracket & := & \pi_{A\times B} ; Q \sqcap \rho_{A\times B} ; R & (\ =\ (Q^{\smile} \nabla R^{\smile})^{\smile}\ )\ . \end{array}$$ □

DEFINITION 3.16
The **parallel composition** $(Q \| R) : A \times C \leftrightarrow B \times D$ of two relations $R : A \leftrightarrow B$ and $S : C \leftrightarrow D$ is defined as:

$$(R \| S) := \pi_{A\times C} ; R ; \pi^{\smile}_{B\times D} \sqcap \rho_{A\times C} ; S ; \rho^{\smile}_{B\times D}$$ □

Sometimes we need the natural isomorphisms between products nested to triples, i.e. the bijection $\rightleftarrows : (A \times (B \times C)) \rightarrowtail\!\!\!\!\rightarrow ((A \times B) \times C)$ with

$$\rightleftarrows = \llparenthesis (\mathbb{I}_A \| \pi_{B\times C}), \rho_{A\times(B\times C)} ; \rho_{B\times C} \rrparenthesis$$

and its inverse $\leftrightarrows = \rightleftarrows^{\smile}$. Furthermore, for commuting pairs we use $\bowtie : A \times B \leftrightarrow B \times A$ with $\bowtie := \llparenthesis \rho, \pi \rrparenthesis$.

We also define a few "shuffling" operations we shall frequently need later as functions from relations to relations:

i) $\ltimes : (A \times B \leftrightarrow C) \rightarrowtail\!\!\!\!\rightarrow (A \leftrightarrow B \times C)$ is defined by $\ltimes.R := \pi^{\smile}_{A\times B} ; \llparenthesis \rho_{A\times B}, R \rrparenthesis_{B\times C}$,

ii) $\boxtimes : (A \leftrightarrow B \times C) \rightarrowtail\!\!\!\twoheadrightarrow (A \times B \leftrightarrow C)$ is defined by $\boxtimes.R := {}_{A\times B}\langle\!\langle R, \pi^\smile_{B\times C} \rangle\!\rangle ; \rho_{B\times C}$,

iii) $\boxbslash : (A \times B \leftrightarrow C) \rightarrowtail\!\!\!\twoheadrightarrow (B \leftrightarrow A \times C)$ is defined by $\boxbslash.R := \rho^\smile_{A\times B} ; [\![\pi_{A\times B}, R]\!]_{A\times C}$,

iv) $\boxslash : (A \leftrightarrow B \times C) \rightarrowtail\!\!\!\twoheadrightarrow (B \times A \leftrightarrow C)$ is defined by $\boxslash.R := {}_{B\times A}\langle\!\langle \pi^\smile_{B\times C}, R \rangle\!\rangle ; \rho_{B\times C}$,

v) $\boxslash : (A \times B \leftrightarrow C) \rightarrowtail\!\!\!\twoheadrightarrow (A \leftrightarrow C \times B)$ is defined by $\boxslash.R := \pi^\smile_{A\times B} ; [\![R, \rho_{A\times B}]\!]_{C\times B}$.

A few useful shuffling properties are the following:

LEMMA 3.17

i) $\pi^\smile ; \boxtimes.R = R ; \rho$

ii) $\boxslash.R ; \pi = \pi^\smile ; R$

iii) $\boxtimes.(\boxtimes.(R) ; S) = R ;(\mathbb{I} \| S)$

iv) $\boxbslash.R ; \pi = \rho^\smile ;(\pi \sqcap R ; \mathbb{T})$

v) $\boxbslash.R \sqcap S ; \pi^\smile = \boxbslash.(\langle\!\langle \mathbb{I}, S \rangle\!\rangle ; \mathbb{T} \sqcap R)$

vi) $\boxbslash.R ; \rho = \rho^\smile ; R$

vii) $\langle\!\langle \mathbb{I}, R \rangle\!\rangle ; \mathbb{T} \sqcap \boxslash.S = \boxslash.(S \sqcap R ; \pi^\smile)$

viii) $\rho^\smile ; \boxslash.R = R ; \rho$

ix) $R ; \boxtimes.S = \boxtimes.((R \| \mathbb{I}) ; S)$ □

Finally we present the relation algebraic characterisation of the direct sum — the dual to the direct product:

DEFINITION 3.18 (Direct Sum)
Let $\iota : A \leftrightarrow S$ and $\kappa : B \leftrightarrow S$ be two relations, then we call the triple (S, ι, κ) *direct sum* of A and B, if the following conditions are fulfilled:

$$\begin{array}{rcl@{\qquad\qquad}rcl} \iota ; \iota^\smile & = & \mathbb{I}_A & \kappa ; \kappa^\smile & = & \mathbb{I}_B \\ \iota^\smile ; \iota \sqcup \kappa^\smile ; \kappa & = & \mathbb{I}_S & \iota ; \kappa^\smile & = & \mathbb{L}_{A,B} \end{array}$$

The injective mappings ι and κ are called the *natural injections* of this sum. □

Usually, we use direct sums of the shape $(A + B, \iota, \kappa)$ and do not differentiate the injections of different sums, since these are usually clear from the context.

We still need some notation for the dual to relation pairing:

DEFINITION 3.19
For $R : A \leftrightarrow C$ and $S : B \leftrightarrow C$ we define $[R \mathbin{+\!\!+} S] : (A + B) \leftrightarrow C$ as

$$[R \mathbin{+\!\!+} S] := \iota^\smile ; R \sqcup \kappa^\smile ; S \ .$$ □

3.4 Symmetric Quotients and Element Relations

The symmetric quotient of two relations $Q : X \leftrightarrow Y$ and $R : X \leftrightarrow Z$ in [36] (see also [7]) is denoted by $\mathrm{syQ}(Q, R) : Y \leftrightarrow Z$ and defined as $\mathrm{syQ}(Q, R) = \overline{Q^\smile ; \overline{R}} \sqcap \overline{\overline{Q^\smile} ; R}$.

We prefer to use $[\![A /\!\!/ B]\!] : X \leftrightarrow Z$ for two relations $A : X \leftrightarrow Y$ and $B : Y \leftrightarrow Z$ with

$$[\![A /\!\!/ B]\!] := \overline{A ; \overline{B}} \sqcap \overline{\overline{A} ; B} \ .$$

So we transpose the first argument, and we have $\mathrm{syQ}(A, B) = [\![A^\smile /\!\!/ B]\!]$. The main advantage of our notation is that the types better fit into the flow of the linear notation.

LEMMA 3.20
The symmetric quotient has the following properties (let F be a total function and

Q be a bijective total function):

$$
\begin{array}{rclrclr}
[\![\overline{A}/\!\!/\overline{B}]\!] & = & [\![A/\!\!/B]\!] & F\,;[\![A/\!\!/B]\!] & = & [\![F\,;A/\!\!/B]\!] & \mathbb{I} \sqsubseteq [\![A^{\smile}/\!\!/A]\!] \\
[\![A^{\smile}/\!\!/B^{\smile}]\!] & = & [\![B/\!\!/A]\!]^{\smile} & [\![A/\!\!/B]\!]\,;F^{\smile} & = & [\![A/\!\!/B\,;F^{\smile}]\!] & A\,;[\![A^{\smile}/\!\!/A]\!] = A \\
[\![A^{\smile}/\!\!/\overline{A}]\!] & = & [\![\overline{A^{\smile}}/\!\!/A]\!] & [\![A/\!\!/B]\!] & \sqsubseteq & [\![A\,;C^{\smile}/\!\!/C\,;B]\!] & A\,;[\![A^{\smile}/\!\!/B]\!] \sqsubseteq B \\
[\![A^{\smile}/\!\!/\overline{A}]\!] & = & [\![A^{\smile}/\!\!/\overline{A}]\!]^{\smile} & [\![A/\!\!/Q\,;B]\!] & = & [\![A\,;Q/\!\!/B]\!] & \square
\end{array}
$$

Lemma 3.21 (Cancellation)

i) $A\,;[\![A^{\smile}/\!\!/B]\!] = B \sqcap \mathbb{T}\,;[\![A^{\smile}/\!\!/B]\!]$ and $[\![A/\!\!/B]\!]\,;B^{\smile} = A \sqcap [\![A/\!\!/B]\!]\,;\mathbb{T}$.

ii) If $[\![A/\!\!/B]\!]$ is surjective, then $A^{\smile}\,;[\![A/\!\!/B]\!] = B$.

iii) If $[\![A/\!\!/B]\!]$ is total, then $[\![A/\!\!/B]\!]\,;B^{\smile} = A$. $\square$

Symmetric quotients can be used to define and manipulate element relations and subsets inside relation algebra:

Definition 3.22

The **element relation** $\in : X \leftrightarrow \mathbb{P}(X)$ for a set X is characterised by the following properties:

i) $[\![\in^{\smile}/\!\!/\in]\!] \sqsubseteq \mathbb{I}$

ii) $[\![Q/\!\!/\in]\!]$ is total for every relation $Q : \mathcal{T} \leftrightarrow X$, i.e., $[\![Q/\!\!/\in]\!]\,;\mathbb{T} = \mathbb{T}$. $\square$

When there are several element relations available in one context, we employ the notational convention of using $\in_{\mathbb{P}(X)} : X \leftrightarrow \mathbb{P}(X)$. Since $A \leftrightarrow B$ is the same as $\mathbb{P}(A \times B)$, we have $\in_{A\leftrightarrow B} : (A \times B) \leftrightarrow (A \leftrightarrow B)$.

Lemma 3.23

For every element relation $\in : X \leftrightarrow \mathbb{P}(X)$ and every relation $R : A \leftrightarrow \mathbb{P}(X)$ we have $[\![R\,;\in^{\smile}/\!\!/\in]\!] = R \sqcup (\overline{R\,;\mathbb{T}} \sqcap [\![\mathbb{L}/\!\!/\in]\!])$. $\square$

Definition 3.24

Every relation $R : A \leftrightarrow B$ can be assigned a point $\mathit{RelToPt}.R : \mathbb{1} \to (A \leftrightarrow B)$ representing it in the following way:

$$\mathit{RelToPt}.R := [\![(\mathbb{T}\,;\langle\!|\,\mathbb{I}, R\,|\!\rangle_{A\times B})/\!\!/\in_{A\leftrightarrow B}]\!] \qquad \square$$

3.5 Sequences

Although sequences can be treated entirely within relation algebra through the use of relators and their fixed points (see e.g. [3]), we do not want to burden the reader with those advanced concepts and just provide a simple generic interface to sequences without bothering about ways to define it.

For a set A we denote the set of finite sequences of elements of A as A^*, and we provide the following (generic) relations:

$$
\begin{array}{lcl lcl lcl}
\mathit{empty} & : & \mathbb{1} \to A^* & \mathit{cons} & : & A \times A^* \rightarrowtail A^* & \mathit{conc} & : & A^* \times A^* \to A^* \\
 & & & \mathit{snoc} & : & A^* \times A \rightarrowtail A^* & \mathit{prefix}, \mathit{suffix} & : & A^* \leftrightarrow A^*
\end{array}
$$

Here "*empty*" and "*cons*" resp. "*empty*" and "*snoc*" can be used as constructors, i.e., there ranges partition their target:

$$
\begin{array}{rcl rcl}
\mathbb{T}\,;\mathit{empty} \sqcap \mathbb{T}\,;\mathit{cons} & = & \mathbb{L} & \mathbb{T}\,;\mathit{empty} \sqcap \mathbb{T}\,;\mathit{snoc} & = & \mathbb{L} \\
\mathbb{T}\,;\mathit{empty} \sqcup \mathbb{T}\,;\mathit{cons} & = & \mathbb{T} & \mathbb{T}\,;\mathit{empty} \sqcup \mathbb{T}\,;\mathit{snoc} & = & \mathbb{T}
\end{array}
$$

"*cons*", considered as a constructor functions, constructs a sequence from a first element and a rest sequence, and "*snoc*" constructs a sequence from an initial sequence segment and a last element. "*conc*" stands for concatenation of sequences, and "*prefix*" and "*suffix*" can be defined from "*conc*":

$$\mathit{prefix} = \pi\breve{}_{A^* \times A^*} ; \mathit{conc} \qquad\qquad \mathit{suffix} = \rho\breve{}_{A^* \times A^*} ; \mathit{conc}$$

Let us now look for a metafunction *fold._* that, for every function $F : (A \times B) \nrightarrow A$ and every relation $G : A \leftrightarrow A$, gives us a relation $\mathit{fold}.F : (A \times B^*) \leftrightarrow A$ with

$$\begin{aligned} (\mathbb{I} \| \top ; \mathit{empty}) ; \mathit{fold}.F &= \mathbb{I} \\ (\mathbb{I} \| \mathit{cons}) ; \mathit{fold}.F &= \rightleftarrows ; (F \| \mathbb{I}) ; \mathit{fold}.F \end{aligned}$$

We can deduce the following equation from this:

$$\mathit{fold}.F = (\rho ; \mathit{empty}\breve{} ; \top \sqcap \pi) \sqcup (\mathbb{I} \| \mathit{cons}\breve{}) ; \rightleftarrows ; (F \| \mathbb{I}) ; \mathit{fold}.F$$

Because of $\mathit{cons}\breve{}$, the relation $(\mathbb{I} \| \mathit{cons}\breve{}) ; \rightleftarrows ; (F \| \mathbb{I})$ obviously is progressively-bounded, and with F it is univalent, too. According to Lemma 3.13 we obtain the following solution:

$$\mathit{fold}.F := \left((\mathbb{I} \| \mathit{cons}\breve{}) ; \rightleftarrows ; (F \| \mathbb{I}) \right)^* ; \llbracket\, \mathbb{I}, \mathit{empty}\breve{} ; \top \,\rrbracket$$

A simple application of *fold._* is the definition $\mathit{conc} := \mathit{fold}.\mathit{snoc}$.

4 Relational Formalisation of Term Graphs

The first clear theoretical foundation for higher-order rewriting of term graphs with variable binding has been presented in [21], with details of the proofs in [20]

Building on a novel definition of term graphs with primitive notions of variable binding and variable identity, a concept of homomorphism has been developed avoiding "capture of variables" and catering for multiple instances of metavariables. Rewriting of these term graphs within the algebraic approach required a new extension that is interesting in itself, the *fibred approach* to rewriting. As one result we obtained the first algebraic characterisation of graph reduction.

In this section, we present the central definitions of [21] together with some motivation and a few important results; more details and all proofs are contained in [20].

In 4.1 we present our new class of term graphs with bound variables. 4.2 discusses paths in these term graphs and encapsulation of nodes by binders. 4.3 introduces a new, more abstract treatment of intervals and related concepts. Finally we present correspondences between intervals in 4.4.

4.1 Term Graphs with Bound Variables

We introduce a new class of node- and edge-labelled term graphs that additionally caters for variable binding. For that purpose we first divide the node labels into labels for constant constructors (these include what is usually called function symbols and constant symbols, but also binding constructors—they are called constant only in contrast to variable), for bindable variables and for metavariables in a similar manner as in CRSs. The arity of constructors is taken care of by a relation from node labels to edge labels, which prescribes the leaving edges for every node. In contrast to other term graph formalisms, this relation may prescribe successors for certain variable labels.

We organise the label sets and arity information into an abstract structure:

DEFINITION 4.1

A **graph alphabet** is a tuple $(\mathcal{N}, (k, v_b, v_s, v_p), \mathcal{E}, E)$, where

- $\mathcal{N}$ is the set of *node labels* with a partition into the label vectors k for *constant constructors*, v_b for *object (*or *bindable) variables*, v_s for *schemavariables* and v_p for *placeholder variables*;
- $\mathcal{E}$ is the set of *edge labels*, and
- $E : \mathcal{N} \leftrightarrow \mathcal{E}$ is the *label compatibility* relating every node label to certain edge labels. □

Schemavariables and placeholder variables are subsumed under the name of *metavariables.* In the sequel, we always assume a fixed graph alphabet.

The graphs we are now going to introduce are a variant of conventional term graphs, but with the additional structure of *variable binding* and *variable identity.* Whereas conventional term graphs can be seen as representing the terms used in conventional term rewriting systems, our graphs emulate the metaterms of CRSs.

Edge labels yield an abstraction from the order of successors frequently used in the literature. Edges are not included as primitive objects, but can be obtained as pairs of nodes and edge labels. The binding mechanism and the variable identity, both of which are derived entities in most formalisms, are considered primitive here—a similar primitive binding concept seems to appear only in the foundations of Bourbaki [8]. This is far more abstract than working with de Bruijn indices, for example, and proves extremely beneficial to the investigation of bound variables within term graphs.

Therefore, we design our graphs to include not only node set, labelling and successor function, but furthermore a partial *binding function* from some bindable variable nodes to constant nodes (the binders), and a partial equivalence relation on variable nodes representing *variable identity.* The binding function has to be *dominating* in the graph theoretic sense, i.e., if a bound variable node is reachable from another node, then the latter is either reachable from the node binding the variable, or the binding node lies on every path from the node in question to the variable node. The variable identity must be compatible with the labelling and binding functions. Variable labels are not names—they only distinguish different kinds of variables. Nameless 0-ary metavariables for term graph rewriting have been in use since [40]; nameless bound variables are only possible because of the explicit binding function, and nameless metavariables with successors because of variable identity; thus we eliminate any need for α-conversion.

DEFINITION 4.2

A **graph** is a tuple $(\mathcal{V}, M, S, D, B, W)$, where

- $\mathcal{V}$ is the *node set*,
- $M : \mathcal{V} \to \mathcal{N}$ is the *node labelling* mapping,
- $S : \mathcal{V} \times \mathcal{E} \nrightarrow \mathcal{V}$ is the *successor function* observing label compatibility: every node $n : \mathbb{1} \to \mathcal{V}$ has for every edge label $i : \mathbb{1} \to \mathcal{E}$ contained in $n ; M ; E$ exactly one outgoing edge labelled i: $\pi_{\mathcal{V}\times\mathcal{E}}^{\smile} ;(S ; \mathbb{T} \sqcap \rho_{\mathcal{V}\times\mathcal{E}}) = M ; E$,

- $D : \mathcal{V} \leftrightarrow \mathcal{V}$ is the *associated relation* of this graph with $D = \pi^{\smile}_{\mathcal{V}\times\mathcal{E}} ; S$,
- $B : \mathcal{V} \nrightarrow \mathcal{V}$ is the *binding function* from (some) object variables to constant constructors ($B \sqsubseteq M ; v_b^{\smile} ; \mathbb{T} ; k ; M^{\smile}$) and has to be *dominating*: for every node $n : \mathbb{1} \to \mathcal{V}$ with $n \sqsubseteq \mathbb{T} ; B$ we demand $n ; B^{\smile} ; (D \sqcap \overline{\mathbb{T} ; n})^* \sqsubseteq n ; D^+$,
- $W : \mathcal{V} \leftrightarrow \mathcal{V}$ is the *variable identity*, a partial equivalence relation ($W^{\smile} \sqsubseteq W$, $W ; W \sqsubseteq W$) defined on all variables ($\overline{k} = \mathbb{T} ; W$) that respects labels ($W ; M \sqsubseteq M$) and bindings ($W ; B \sqsubseteq B$) and is trivial on placeholders: $W \sqcap \mathbb{T} ; v_p ; M^{\smile} \sqsubseteq \mathbb{I}$. □

When drawing our graphs, however, we do not depict all this information explicitly. Instead we stick to the intuitive representation already used in Sect. 2 that allows to reconstruct the precise graph definition when necessary. The associated relation has been included in the definition only for convenience.

4.2 Paths and Encapsulation

A sequence of edge labels corresponds to the concept of "positions" of term rewriting or "occurrences" in [43]). Together with a starting node, such a sequence turns into a **path**, and using the successor relation we can follow these paths:

Definition 4.3
For a graph G, the **path function** $Z : \mathcal{V} \times \mathcal{E}^* \nrightarrow \mathcal{V}$ is defined by $Z := Z_G := \mathit{fold}.S$, and the **path relation** as $\mathbb{Z} := \mathbb{Z}_G := (\mathbb{I} \| \mathit{prefix}^{\smile}) ; Z$.

$\underset{+}{\mathbb{Z}} := (\mathbb{I} \| \mathit{prefix}^{\smile} \sqcap \overline{\mathbb{T} ; \mathit{empty}}) ; Z$ relates a concrete path with those nodes it touches after a *nonempty* prefix.

$\underset{\varsigma}{Z} := Z \sqcap \overline{(\underset{+}{\mathbb{Z}} \sqcap \pi_{\mathcal{V}\times\mathcal{E}^*}) ; \mathbb{T}}$ is the restriction of the path function to those concrete paths that do not contain a cycle around their starting point. □

Closely related to the path relations are the following two:

Definition 4.4
The **path prolongation** $P : (\mathcal{V} \times \mathcal{E}^*) \times \mathcal{E} \leftrightarrow \mathcal{V} \times \mathcal{E}^*$ associates with every path in the domain of Z together with an edge label exactly those paths in the domain of Z, that are this one edge label longer:

$$P := [\![\langle\!\langle \mathbb{I}, Z ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}} ; (S ; \mathbb{T} \sqcap \rho_{\mathcal{V}\times\mathcal{E}}) \rangle\!\rangle , \rho^{\smile}]\!] ; \leftrightarrows ; (\mathbb{I} \| \mathit{snoc}) \ .$$

The **path concatenation** $\mathit{Pconc} : (\mathcal{V} \times \mathcal{E}^*) \times \mathcal{E}^* \leftrightarrow \mathcal{V} \times \mathcal{E}^*$ simply concatenates a concrete path with an abstract path: $\mathit{Pconc} = \leftrightarrows ; (\mathbb{I} \| \mathit{conc})$. □

Lemma 4.5
i) $P ; \pi_{\mathcal{V}\times\mathcal{E}^*} \sqsubseteq \pi ; \pi_{\mathcal{V}\times\mathcal{E}^*}$, ii) $\pi^{\smile} ; \pi^{\smile} ; P \sqsubseteq \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}$, iii) $\mathit{Pconc} ; Z = (Z \| \mathbb{I}) ; Z$.

Proof:

$$\begin{aligned}
\text{i) } P ; \pi_{\mathcal{V}\times\mathcal{E}^*} &= [\![\langle\!\langle \mathbb{I}, Z ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}} ; (S ; \mathbb{T} \sqcap \rho_{\mathcal{V}\times\mathcal{E}}) \rangle\!\rangle , \rho^{\smile}]\!] ; \leftrightarrows ; (\mathbb{I} \| \mathit{snoc}) ; \pi_{\mathcal{V}\times\mathcal{E}^*} \\
&\sqsubseteq \pi ; \langle\!\langle \mathbb{I}, Z ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}} ; (S ; \mathbb{T} \sqcap \rho_{\mathcal{V}\times\mathcal{E}}) \rangle\!\rangle ; \leftrightarrows ; \pi \\
&\sqsubseteq \pi ; \mathbb{I} ; \pi^{\smile} ; \pi ; \pi_{\mathcal{V}\times\mathcal{E}^*} \ = \ \pi ; \pi_{\mathcal{V}\times\mathcal{E}^*} \\
\text{ii) } \pi^{\smile} ; \pi^{\smile} ; P &= \pi^{\smile} ; \langle\!\langle \mathbb{I}, Z ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}} ; (S ; \mathbb{T} \sqcap \rho_{\mathcal{V}\times\mathcal{E}}) \rangle\!\rangle ; \leftrightarrows ; (\mathbb{I} \| \mathit{snoc}) \\
&\sqsubseteq \pi^{\smile} ; \mathbb{I} ; \pi^{\smile} ; \leftrightarrows ; (\mathbb{I} \| \mathit{snoc}) \ = \ \pi^{\smile} ; (\mathbb{I} \| \mathit{snoc}) \ \sqsubseteq \ \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}
\end{aligned}$$

iii) $\mathit{Pconc} ; Z = \leftrightarrows ; (\mathbb{I} \| \mathit{conc}) ; \mathit{fold}.S = (\mathit{fold}.S \| \mathbb{I}) ; \mathit{fold}.S = (Z \| \mathbb{I}) ; Z$ □

Since binding nodes dominate their bound variables, there is always at least one acyclic path from a binder to every bound variable. Since these paths are essential for a certain kind of proof, we define:

Definition 4.6
The **binding path relation** $\mathbb{B} : \mathcal{V} \times \mathcal{E}^* \leftrightarrow \mathcal{V}$ for a graph G is defined as

$$\mathbb{B} := \pi_{\mathcal{V}\times\mathcal{E}^*} ; B^{\smile} \sqcap \text{\c{Z}} \qquad \square$$

We have $B = \mathbb{B}^{\smile} ; \pi$.

In the presence of variable binding, we need the concept of free occurrences of variables: If there is a path from any node to a variable node, such that the binding node of the latter—if existent—does not lie on that path, then that variable occurs *free* under the former node. If a variable bound by a binding node occurs free below some other node, then the latter is called *encapsulated* by the binding node.

Definition 4.7
For a graph G, the relation $-\!\!\!\!\mathcal{C} : \mathcal{V} \leftrightarrow \mathcal{V}$ relates object variables to those nodes, below which the variable in question occurs **free**:

$$-\!\!\!\!\mathcal{C} = (W ; \mathbb{T} \sqcap Z^{\smile} \sqcap \overline{B ; \mathbb{Z}^{\smile}}) ; \pi_{\mathcal{V}\times\mathcal{E}^*}$$

The **encapsulation** $C : \mathcal{V} \leftrightarrow \mathcal{V}$ is the relation from the binding nodes to the *encapsulated* ones:

$$C := B^{\smile} ; -\!\!\!\!\mathcal{C} \ .$$

The *encapsulation-free reachability* relation $\Delta : \mathcal{V} \leftrightarrow \mathcal{V}$ holds for nodes x and y exactly when y is reachable from x via a path on which no node encapsulates x:

$$\Delta := (\pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqcap \overline{C^{\smile} ; \mathbb{Z}^{\smile}}) ; Z \qquad \square$$

The encapsulation relation C, which is derived from the concepts of binding and (local) freeness of variables, turns out to be extremely important in the sequel. Later we shall need the requirement that every node is reachable from an unencapsulated node:

Definition 4.8
A graph G is called **encapsulation-free based**, if there is a basis $\beta : \mathbb{1} \leftrightarrow \mathcal{V}$ (that is a vector with $\beta ; D^* = \mathbb{T}$ and $\beta ; D^+ \sqcap \beta \sqsubseteq \overline{\mathbb{I}}$) with $\beta ; C^{\smile} = \mathbb{L}$. $\square$

An interesting and important fact is then the following:

Theorem 4.9 (Acyclic Encapsulation)
Let G be an encapsulation-free based graph, then the encapsulation relation is acyclic, i.e., $C^+ \sqcap \mathbb{I} = \mathbb{L}$ holds.

Proof: Since C is irreflexive, we only need to consider $C^+ ; C$. Let $n \geq 2$ nodes $\alpha_1, \ldots, \alpha_n : \mathbb{1} \to \mathcal{V}$ be given that cyclically encapsulate their successors, i.e., for all $1 \leq i < n$ the inclusion $\alpha_{i+1} \sqsubseteq \alpha_i ; C$ holds, and in addition $\alpha_1 \sqsubseteq \alpha_n ; C$, implying $\alpha_1 \neq \alpha_n$.

Let β be an unencapsulated node from which α_1 is reachable. Since $\alpha_1 \sqsubseteq \alpha_n C$ there is a variable free below α_1 that is bound by α_n, therefore every path from β to α_1 passes α_n, since otherwise β would be encapsulated by α_n. Hence α_n is reachable from β.

Via the nodes α_{n-1} to α_1 the argument can be continued analogously, so that finally every path from β to α_n also must pass α_1. From this, however, it would follow in contradiction to the assumption that there is no finite path from β to α_1. $\square$

A useful consequence of this theorem is that C is regressively finite on encapsulation-free based graphs.

From now on, we shall tacitly restrict all graphs to be encapsulation-free based.

4.3 Intervals

Since we have schemavariables that are to correspond to the metavariables of CRSs, we need a concept that is appropriate to serve for their homomorphic images, i.e., we have to delimit parts of the graph from the top as well as from the bottom, yielding a concept of *inner nodes*. Here, however, we have to be extremely careful in cyclic graphs, where it turns out to be reasonable to let the inner nodes stop short of those nodes that encapsulate the upper border (for an example of these *self-encapsulation borders* see the picture below). Therefore we define two concepts of inner nodes that coincide on acyclic graphs, a "*raw*" version and a refined version; they correspond to the two separate concepts of *intervals* and *sections* in [20, 21]. In the raw view, the inner nodes of an interval are simply those reachable from the upper border via a path that does not touch the lower border. In the refined view, these paths must not touch self-encapsulation borders, either.

DEFINITION 4.10
For a graph G, let $\mathcal{I} := \mathcal{I}_G := \mathcal{V} \times (\mathcal{E} \leftrightarrow \mathcal{V})$ be the type of **intervals**. We defined the *lower border* relation $Lower_G : \mathcal{I}_G \leftrightarrow \mathcal{V}$ by

$$Lower := Lower_G := \rho_{\mathcal{I}_G} ; \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}} ; \rho_{\mathcal{E}\times\mathcal{V}} \quad .$$

For two relations $t : \mathbb{1} \to \mathcal{V}$ and $b : \mathcal{E} \leftrightarrow \mathcal{V}$, the **interval** from t to b can be written as $[t, b] : \mathbb{1} \to \mathcal{I}_G$ and is defined to be $[t, b] := \langle\!\langle t, RelToPt.b \rangle\!\rangle$. □

The raw version of the inner nodes definition assumes that inner nodes are all those nodes which are reachable from the upper border via paths that do not touch the lower border:

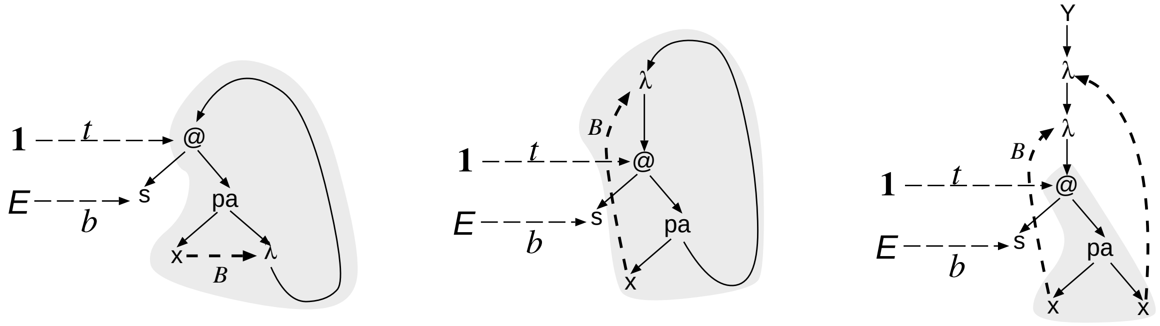
DEFINITION 4.11
$RawInner_G : \mathcal{I}_G \leftrightarrow \mathcal{V}$ associates every interval with its **inner nodes**:

$$RawInner := RawInner_G := (\pi_{\mathcal{I}} ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqcap \overline{Lower ; \mathbb{Z}^{\smile}}) ; Z$$

For some interval $[t, b] : \mathbb{1} \to \mathcal{I}_G$, the vector of its inner nodes is written $\underline{[t, b]} : \mathbb{1} \leftrightarrow \mathcal{I}_G$ with $\underline{[t, b]} := [t, b] ; RawInner$. □

As already mentioned above, this version is not satisfactory in cyclic graphs. Let us consider an example.

The following picture shows on the left two different drawings of the same interval in the same cyclic graph; the λ-node encapsulates the upper border of the interval so that the vector of the inner nodes (in the raw view) of that interval cannot really be treated independently. On the right there is another graph that corresponds to the first one in that the cycle has been broken up by an abstraction that is operated on by a fixed point combinator. Usually the two representations would be considered to be equivalent. But to our theory they are very different. This can be seen most prominently when looking at the "**skeletons**" of the two intervals, where the skeleton of an interval is the vector of those inner nodes that lie on paths from the upper to the lower border (see also Def. 4.14). In the left graph the skeleton contains all inner nodes except the bound variable, while in the right graph in contains only the upper border itself.



For any definition of rewriting, these skeletons require special attention, and so the current state of affairs that two essentially equivalent graphs have so fundamentally differing skeletons is unsatisfactory.

The key concept to the solution of this problem is the following:

Definition 4.12
The **self-encapsulation border** of an interval $[t, b]$ consists of those nodes that lie inside the interval and encapsulate the upper border: $\underline{[t, b]} \sqcap t \mathbin{;} C^{\smile}$. Accordingly we define the relation $SCB_G : \mathcal{I}_G \leftrightarrow \mathcal{V}$ associating intervals with the nodes of their self-encapsulation borders:

$$SCB := RawInner \sqcap \pi_{\mathcal{I}} \mathbin{;} C^{\smile} \qquad \square$$

By constraining the inner nodes of an interval to stop before self-encapsulation borders, we obtain the refined definition of inner nodes that makes the two skeletons in the picture above equal:

Definition 4.13
$Inner : \mathcal{I}_G \leftrightarrow \mathcal{V}$ is defined as

$$Inner \quad := \quad \left(\pi_{\mathcal{I}} \mathbin{;} \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqcap \overline{(Lower \sqcup \pi_{\mathcal{I}} \mathbin{;} C^{\smile}) \mathbin{;} \mathbb{Z}^{\smile}}\right) \mathbin{;} Z$$

For an interval $\mathfrak{a} : \mathbb{1} \to \mathcal{I}_G$, we define the shorthand notation $\underline{\mathfrak{a}} : \mathbb{1} \leftrightarrow \mathcal{V}$ for the *vector of inner nodes*: $\underline{\mathfrak{a}} := \mathfrak{a} \mathbin{;} Inner$. $\square$

From now on, we shall almost always use *Inner* to calculate the inner nodes of intervals — on acyclic graphs, there is no difference, and on cyclic graphs, *RawInner* does not make much sense except under very special circumstances.

The interval $\mathfrak{a}$ is called a **segment**, iff $\underline{\mathfrak{a}}$ is closed under encapsulation, i.e., $\underline{\mathfrak{a}} \mathbin{;} C \sqsubseteq \underline{\mathfrak{a}}$. The *segment vector* $Segment : \mathbb{1} \leftrightarrow \mathcal{I}$ is therefore defined as

$$Segment := \mathbb{T} \mathbin{;} (\mathbb{I} \sqcap \overline{Inner \mathbin{;} C \mathbin{;} \overline{Inner}}) \ ,$$

and we have $\mathfrak{a} \sqsubseteq Segment$ iff $\underline{\mathfrak{a}} \mathbin{;} C \sqsubseteq \underline{\mathfrak{a}}$.

The interval $\mathfrak{a} = [t, b]$ is **coherent** iff the whole lower border has predecessors within the interval: $\mathbb{T} \mathbin{;} b \sqsubseteq \mathbb{T} \mathbin{;} t \sqcup \underline{\mathfrak{a}} \mathbin{;} D$, and the corresponding vector $Coherent : \mathbb{1} \leftrightarrow \mathcal{I}$ of coherent intervals is

$$Coherent := \mathbb{T} \mathbin{;} (\mathbb{I} \sqcap \overline{Lower \mathbin{;} \overline{\pi^{\smile}_{\mathcal{I}} \sqcup D^{\smile} \mathbin{;} Inner^{\smile}}}) \ .$$

The interval $\mathfrak{a} = [t, b]$ is **half-coherent** iff the whole lower border is reachable from the upper border: $\mathbb{T} ; b \sqsubseteq \mathbb{T} ; t ; D^*$, and the corresponding vector $\mathit{HalfCoh} : \mathbb{1} \leftrightarrow \mathcal{I}$ of half-coherent intervals is

$$\mathit{HalfCoh} := \mathbb{T} ;(\mathbb{I} \sqcap \overline{\overline{\pi_{\mathcal{I}} ; D^* ; \mathit{Lower}^{\smile}}}) \ .$$

The following definition is essential for avoiding extra copying of e.g. the bodies of λ-abstractions in β-reductions. Wadsworth enabled efficient graph reduction by clarifying the necessary distinctions [42]. There, the parts outside what we are going to call skeletons are called "abstractable" (since Wadsworth only had bodies of λ-abstractions as intervals, he did not need an independent notion of intervals). The *encapsulation skeleton* is a generalisation of that concept, and the simple skeleton is a restriction we shall need for characterising "acyclic" homomorphisms:

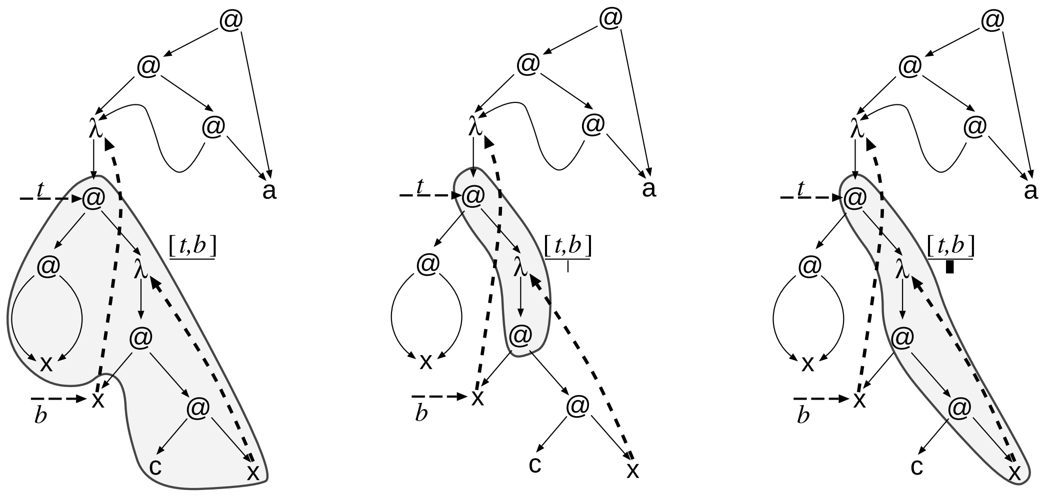
Definition 4.14

The **skeleton** $\underset{\top}{\underline{\mathfrak{a}}} : \mathbb{1} \leftrightarrow \mathcal{V}$ of an interval $\mathfrak{a} = [t, b]$ is defined as the vector comprising all the inner nodes "between" the upper and lower borders of the interval, and the **encapsulation skeleton** $\underset{\blacksquare}{\underline{\mathfrak{a}}} : \mathbb{1} \leftrightarrow \mathcal{V}$ additionally contains all those "depending" from them by (eventually repeated) encapsulation:

$$\begin{aligned} \underset{\top}{\underline{\mathfrak{a}}} &:= \mathbb{T} ; b ; (D^{\smile} \sqcap \mathbb{T} ; \underline{\mathfrak{a}})^+ \\ \underset{\blacksquare}{\underline{\mathfrak{a}}} &:= \mathbb{T} ; b ;(D^{\smile} \sqcap \mathbb{T} ; \underline{\mathfrak{a}}) ; ((D^{\smile} \sqcup C) \sqcap \mathbb{T} ; \underline{\mathfrak{a}})^* \end{aligned}$$

Generalising, we obtain two relations $\mathit{Skel}_G, \mathit{EncSkel}_G : \mathcal{I}_G \leftrightarrow \mathcal{V}$ with

$$\begin{aligned} \mathit{Skel} &:= \llparenthesis \mathit{Lower}, \mathbb{I} \rrparenthesis ; \llparenthesis \llbracket D^{\smile}, \mathit{Inner} \rrbracket, \rho \rrparenthesis^+ ; \pi \\ \mathit{EncSkel} &:= \llparenthesis \mathit{Lower}, \mathbb{I} \rrparenthesis ;\llparenthesis \llbracket D^{\smile}, \mathit{Inner} \rrbracket, \rho \rrparenthesis ; \llparenthesis \llbracket D^{\smile} \sqcup C, \mathit{Inner} \rrbracket, \rho \rrparenthesis^* ; \pi \ . \end{aligned}$$ □



For segments, a more concise definition of the encapsulation skeleton is possible: $\mathit{EncSkel} \sqcap \mathit{Segment}^{\smile} ; \mathbb{T} = \mathit{Skel} ; C^* \sqcap \mathit{Segment}^{\smile} ; \mathbb{T}$.

Finally, we provide an auxiliary relation for paths through intervals:

Definition 4.15

For a graph G, the relation $T_G : \mathcal{I}_G \times \mathcal{E} \leftrightarrow \mathcal{V} \times \mathcal{E}^*$ associates with every interval i and edge label e exactly those paths that start at the upper border of i, and end at a node of the lower border of i for e:

$$T_G := \pi ; \pi_{\mathcal{I}} ; \pi_{\mathcal{V} \times \mathcal{E}^*}^{\smile} \sqcap \boxtimes.(\rho_{\mathcal{I}} ; \in_{\mathcal{E} \leftrightarrow \mathcal{V}}^{\smile}) ; Z^{\smile}$$ □

Note that there is no restriction that those paths should only run over inner nodes here. We now prove a few facts about T that will be useful later:

LEMMA 4.16
i) $\pi^{\smile} \mathbin{;} T = {}_{\mathcal{I}}\{\!\!\{ \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}, \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}} \mathbin{;} \rho \mathbin{;} Z^{\smile} \}\!\!\}$ ii) $T \mathbin{;} Z \sqsubseteq \boxtimes.(\rho_{\mathcal{I}} \mathbin{;} \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}})$

Proof:

$$
\begin{array}{lcll}
\text{i) } \pi^{\smile} \mathbin{;} T & = & \pi^{\smile} \mathbin{;} (\pi \mathbin{;} \pi_{\mathcal{I}} \mathbin{;} \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqcap \boxtimes.(\rho_{\mathcal{I}} \mathbin{;} \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}}) \mathbin{;} Z^{\smile}) & \text{Def. 4.15} \\
& = & \pi_{\mathcal{I}} \mathbin{;} \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqcap \pi^{\smile} \mathbin{;} \boxtimes.(\rho_{\mathcal{I}} \mathbin{;} \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}}) \mathbin{;} Z^{\smile} & \pi \text{ univalent} \\
& = & \pi_{\mathcal{I}} \mathbin{;} \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqcap \rho_{\mathcal{I}} \mathbin{;} \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}} \mathbin{;} \rho \mathbin{;} Z^{\smile} & \text{Lemma 3.17 i)} \\
& = & {}_{\mathcal{I}}\{\!\!\{ \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}, \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}} \mathbin{;} \rho \mathbin{;} Z^{\smile} \}\!\!\} & \text{Def. 3.15}
\end{array}
$$

ii) immediate with the univalence of Z. □

4.4 Correspondences

We have introduced intervals and segments for acting as images of metavariables from the next section on. We have already seen in 2.2 that *parallel metavariables* — according to Def. 4.2 in fact only parallel schemavariables — can occur. Whenever there are several schemavariable nodes connected by the variable identity, we have to compare the image segments of these nodes. One might be tempted to ask for a restricted form of isomorphy, but it turns out that we may demand only *isomorphy up to identification.*

DEFINITION 4.17
$H : \mathcal{V} \leftrightarrow \mathcal{V}$ is a **correspondence** from segment $\mathfrak{a} = [t, b]$ to segment $\mathfrak{b} = [u, c]$, iff:

i) self-encapsulation borders are equal: $\mathfrak{a} \mathbin{;} SCB = \mathfrak{b} \mathbin{;} SCB$,

ii) H covers exactly the inner nodes: $\mathbb{T} \mathbin{;} H^{\smile} = \underline{\mathfrak{a}}$ and $\mathbb{T} \mathbin{;} H = \underline{\mathfrak{b}}$,

iii) H preserves upper borders: $(t \sqcap \overline{\mathbb{T} \mathbin{;} b}) \mathbin{;} H \sqsupseteq u \sqcap \overline{\mathbb{T} \mathbin{;} c}$, and $(u \sqcap \overline{\mathbb{T} \mathbin{;} c}) H^{\smile} \sqsupseteq t \sqcap \overline{\mathbb{T} \mathbin{;} b}$,

iv) H preserves labels: $H \mathbin{;} M \sqsubseteq M$,

v) H preserves successors: $(H \| \mathbb{I}) \mathbin{;} S \sqsubseteq S \mathbin{;} (H \sqcup b^{\smile} \mathbin{;} c \sqcup (\mathbb{I} \sqcap \mathbb{T} \mathbin{;} \mathfrak{a} \mathbin{;} SCB))$,

vi) H preserves bindings on internally bound nodes:

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
H \mathbin{;} B \sqcap \mathbb{T} \mathbin{;} H & \sqsubseteq & B \mathbin{;} H & B \mathbin{;} H & \sqsubseteq & H \mathbin{;} B \\
H^{\smile} \mathbin{;} B \sqcap \mathbb{T} \mathbin{;} H^{\smile} & \sqsubseteq & B \mathbin{;} H^{\smile} & B \mathbin{;} H^{\smile} & \sqsubseteq & H^{\smile} \mathbin{;} B
\end{array}
$$

vii) H respects the distinctness of nodes internally bound to the same constructor: $H^{\smile} \mathbin{;} W \mathbin{;} H \sqcap \mathbb{T} \mathbin{;} H \mathbin{;} B \sqcap B \mathbin{;} B^{\smile} \sqsubseteq W$, and so does $H^{\smile}$,

viii) H respects encapsulation: $H \mathbin{;} H^{\smile} \sqsubseteq \overline{C}$ and $H^{\smile} \mathbin{;} H \sqsubseteq \overline{C}$,

ix) H preserves variables: $H \mathbin{;} W \sqcap \mathbb{T} \mathbin{;} H = W \mathbin{;} H \sqcap H \mathbin{;} \mathbb{T}$, and

x) H stays within the same variable for variables that are not internally bound: $H \sqcap \overline{B \mathbin{;} H \mathbin{;} \mathbb{T}} \sqcap W \mathbin{;} \mathbb{T} \sqsubseteq W$. □

The additional terms in the preservation of successors provide for the lower borders of the segments and for the self-encapsulation borders that are excluded from the inner nodes of the segments, too.

It is relatively easy to see that for a correspondence H, its converse $H^{\smile}$ is also a correspondence. Also the identity relation restricted to the inner nodes of a segment obviously is a correspondence.

For proving composability of correspondences we need a few preparations: A central property essentially following from encapsulation respect is the following commutativity of correspondences with binding paths. This kind of property which is simply stated but hard to prove and heavily relies on the domination of variable binding together with the successor preservation of the relation involved is very typical for term graphs with bound variables.

The proof proceeds via a detailed case analysis and relies heavily on the domination property of the variable binding. Together with the involved definition of binding paths that employs acyclicity this would blow up a fully calculational presentation of the proof to unmanageable size and complexity. We therefore present a "mixed style" proof in conventional meta-level predicate-logic style, introducing variables for points as needed and doing relational calculation where feasible:

Lemma 4.18 (Forwards Commutativity with Binding Paths)
For a graph G, let a correspondence F from segment $\mathfrak{a}$ to segment $\mathfrak{b}$ be given. Then F is preserved over binding paths:

$$(F\|\mathbb{I}) \mathbin{;} \mathbb{B} \sqsubseteq \mathbb{B} \mathbin{;} F \enspace .$$

Proof: Let two points $p_1 : \mathcal{V}\times\mathcal{E}^*$ and $\gamma_2 : \mathbb{1} \to \mathcal{V}$ standing for a path and the end-point of an image, i.e. with $\gamma_2 \sqsubseteq p_1 \mathbin{;} (F\|\mathbb{I}) \mathbin{;} \mathbb{B}$, be given. Let $p_2 := \gamma_2 \mathbin{;} \mathbb{B}^{\smile} \sqcap p_1 \mathbin{;} \rho_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} \rho_{\mathcal{V}\times\mathcal{E}^*}^{\smile}$ be the corresponding intermediate point, which is a binding path leading to γ_2.

Let $\alpha_1, \alpha_2 : \mathbb{1} \to \mathcal{V}$ be the respective starting points of p_1 and p_2, i.e., $\alpha_1 := p_1 \mathbin{;} \pi_{\mathcal{V}\times\mathcal{E}^*}$ and $\alpha_2 := p_2 \mathbin{;} \pi_{\mathcal{V}\times\mathcal{E}^*}$. Then $\alpha_2 = \gamma_2 \mathbin{;} B$ and further $p_2 \mathbin{;} \underset{+}{\not{\mathbb{Z}}} \sqsubseteq \gamma_2 \mathbin{;} \not{C} \sqsubseteq \alpha_2 \mathbin{;} C \sqsubseteq \underline{\mathfrak{b}}$. Therefore we can commute the path function with F in the following, thus obtaining: $\gamma_2 \sqsubseteq p_1 \mathbin{;} (F\|\mathbb{I}) \mathbin{;} (Z \sqcap \pi_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} B^{\smile}) \sqsubseteq p_1 \mathbin{;} Z \mathbin{;} F$.

Let $\gamma_1 := p_1 \mathbin{;} Z$, and we obtain $\alpha_2 \sqsubseteq \gamma_1 \mathbin{;} F \mathbin{;} B \sqcap \mathbb{T} \mathbin{;} F \sqsubseteq \gamma_1 \mathbin{;} B \mathbin{;} F$. This implies $\gamma_1 \mathbin{;} B \neq \mathbb{⊥}$ and $\gamma_1 \mathbin{;} B \sqsubseteq \alpha_1 \mathbin{;} F \mathbin{;} F^{\smile}$. If $\alpha_1 \sqsubseteq \gamma_1 \mathbin{;} \not{C}$, then also $\alpha_1 \sqsubseteq \gamma_1 \mathbin{;} B \mathbin{;} B^{\smile} \mathbin{;} \not{C} = \gamma_1 \mathbin{;} B \mathbin{;} C$, and with $F \mathbin{;} F^{\smile} \sqcap C \sqsupseteq B^{\smile} \mathbin{;} \gamma_1^{\smile} \mathbin{;} \alpha_1 \neq \mathbb{⊥}$ we obtain a contradiction to the encapsulation respect of F.

If otherwise $\alpha_1 \neq \gamma_1 \mathbin{;} B$, then $\gamma_1 \mathbin{;} B$ is on every path from α_1 to γ_1, therefore also on p_1. Let $q_1 : \mathbb{1} \to \mathcal{V} \times \mathcal{E}^*$ be a prefix of p_1 that reaches $\gamma_1 \mathbin{;} B$, i.e. a path with $p_1 \sqsubseteq q_1 \mathbin{;} (\mathbb{I}\|\mathit{prefix})$ and with $q_1 \mathbin{;} Z = \gamma_1 \mathbin{;} B$, and let $q_2 := \{\!\!|\, \alpha_2, q_1 \mathbin{;} \rho_{\mathcal{V}\times\mathcal{E}^*} \,|\!\!\}_{\mathcal{V}\times\mathcal{E}^*}$ then:

$$\alpha_2 \sqsubseteq \gamma_1 \mathbin{;} B \mathbin{;} F = q_1 \mathbin{;} Z \mathbin{;} F \sqsubseteq q_2 \mathbin{;} (F^{\smile}\|\mathbb{I}) \mathbin{;} Z \mathbin{;} F \sqsubseteq q_2 \mathbin{;} Z \mathbin{;} F^{\smile} \mathbin{;} F \enspace .$$

According to the assumptions so far we have $\mathit{empty} \neq q_1 \mathbin{;} \rho_{\mathcal{V}\times\mathcal{E}^*} \neq p_1 \mathbin{;} \rho_{\mathcal{V}\times\mathcal{E}^*}$ and $\alpha_2 \neq \{\!\!|\, \alpha_2, n \,|\!\!\}_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} Z$, and therefore also $\mathbb{⊥} \neq q_2 \mathbin{;} Z \sqsubseteq \alpha_2 \mathbin{;} (C \sqcap F^{\smile} \mathbin{;} F)$, which yields a contradiction to the encapsulation respect of F.

Therefore $\alpha_1 = \gamma_1 \mathbin{;} B$ must hold, and $p_1^{\smile} \mathbin{;} \gamma_2 \sqsubseteq (Z \sqcap \pi_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} B^{\smile}) \mathbin{;} F$ is already shown.

Assuming $\gamma_1 \sqsubseteq \overline{p_1 \mathbin{;} \underset{,}{Z}}$, then there has to exist an edge label sequence $n : \mathbb{1} \to \mathcal{E}^*$ giving rise to a nonempty prefix of p_1 that has α_1 as its end-point, i.e. with $n \neq \mathit{empty}$, $n^{\smile} \mathbin{;} p_1 \mathbin{;} \rho_{\mathcal{V}\times\mathcal{E}^*} \sqsubseteq \mathit{prefix}$ and $\{\!\!|\, \alpha_1, n \,|\!\!\}_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} Z = \alpha_1$. Here again we obtain

$$\begin{array}{rcl} \alpha_2 & \sqsubseteq & \alpha_1 \mathbin{;} F \;=\; \{\!\!|\, \alpha_1, n \,|\!\!\}_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} Z \mathbin{;} F \\ & \sqsubseteq & \{\!\!|\, \alpha_2, n \,|\!\!\}_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} (F^{\smile}\|\mathbb{I}) \mathbin{;} Z \mathbin{;} F \;\sqsubseteq\; \{\!\!|\, \alpha_2, n \,|\!\!\}_{\mathcal{V}\times\mathcal{E}^*} \mathbin{;} Z \mathbin{;} F^{\smile} \mathbin{;} F \end{array}$$

and further $\{\!\!\{ \alpha_2, n \}\!\!\}_{\mathcal{V}\times\mathcal{E}^*} ; Z \sqsubseteq \alpha_2 ;(C \sqcap F^{\smile} ; F)$, again yielding a contradiction to the encapsulation respect of F. Therefore $\gamma_1 \sqsubseteq p_1 ; Z$ and the whole original statement hold. $\square$

Onve established, this property allows nice calculational proofs using it:

Lemma 4.19 (Backwards Commutativity with Binding Paths)
For a graph G, let a correspondence F from segment $\mathfrak{a}$ to segment $\mathfrak{b}$ be given. Then F is preserved over the converse of binding paths starting locally:

$$F ; \mathbb{B}^{\smile} \sqcap \mathbb{T} ; F ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} \sqsubseteq W ; \mathbb{B}^{\smile} ;(F \| \mathbb{I})$$

Proof:

$$\begin{array}{cll}
 & F ; \mathbb{B}^{\smile} \sqcap \mathbb{T} ; F ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} & \\
= & F ; \mathbb{B}^{\smile} \sqcap (F ; B \sqcap \mathbb{T} ; F) ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} & \text{Def. } \mathbb{B}\text{, univalence of } \pi_{\mathcal{V}\times\mathcal{E}^*} \\
\sqsubseteq & F ; \mathbb{B}^{\smile} \sqcap B ; F ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} & F \text{ preserves bindings} \\
= & F ; \mathbb{B}^{\smile} \sqcap B ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} ;(F \| \mathbb{I}) & \forall R \bullet R ; \pi^{\smile} = \pi^{\smile} ;(R \| \mathbb{I}) \\
\sqsubseteq & (F ; \mathbb{B}^{\smile} ;(F^{\smile} \| \mathbb{I}) \sqcap B ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}) ;(F \| \mathbb{I}) & \text{modal rule} \\
\sqsubseteq & (F ; F^{\smile} ; \mathbb{B}^{\smile} \sqcap B ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}) ;(F \| \mathbb{I}) & \text{Lemma 4.18} \\
\sqsubseteq & (F ; F^{\smile} \sqcap B ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} ; \mathbb{B}) ; \mathbb{B}^{\smile} ;(F \| \mathbb{I}) & \text{modal rule} \\
= & (F ; F^{\smile} \sqcap B ; B^{\smile}) ; \mathbb{B}^{\smile} ;(F \| \mathbb{I}) & \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*} ; \mathbb{B} = B^{\smile} \\
\sqsubseteq & W ; \mathbb{B}^{\smile} ;(F \| \mathbb{I}) & F^{\smile} \text{ pres. dist. of local vars.}
\end{array}$$ $\square$

Theorem 4.20 (Composition of Correspondences)
For a graph G, let a correspondence F from $\mathfrak{a}$ to $\mathfrak{b}$ and a correspondence G from $\mathfrak{b}$ to $\mathfrak{c}$ be given, where $\mathfrak{a} = [t, b]$, $\mathfrak{b} = [u, c]$ and $\mathfrak{c} = [v, d]$ are segments. If c is injective, then $F ; G$ is a correspondence, too.

Proof:

i) Equality of self-encapsulation borders: $\mathfrak{a} ; SCB = \mathfrak{b} ; SCB = \mathfrak{c} ; SCB$.

ii) Domain and Range:

$$\mathbb{T} ;(F ; G)^{\smile} = \mathbb{T} ; G^{\smile} ; F^{\smile} = \mathbb{T} ; \underline{\mathfrak{b}} ; F^{\smile} = \mathbb{T} ; \underline{\mathfrak{a}} \quad \text{and} \quad \mathbb{T} ; \underline{\mathfrak{a}} ; F ; G = \mathbb{T} ; \underline{\mathfrak{b}} ; G = \mathbb{T} ; \underline{\mathfrak{c}}$$

iii) Upper borders: $(t \sqcap \overline{\mathbb{T} ; b}) ; F ; G \sqsupseteq (u \sqcap \overline{\mathbb{T} ; c}) ; G \sqsupseteq (v \sqcap \overline{\mathbb{T} ; d})$ and an analogous calculation for the second condition.

iv) Label preservation: $F ; G ; M \sqsubseteq F ; M \sqsubseteq M$

v) Successor preservation: This is where injectivity of c is needed:

$$\begin{array}{cl}
 & (F ; G \| \mathbb{I}) ; S \\
\sqsubseteq & (F \| \mathbb{I}) ; S ;(G \sqcup c^{\smile} ; d \sqcup (\mathbb{I} \sqcap \mathbb{T} ; \mathfrak{b} ; SCB)) \\
\sqsubseteq & S ;(F \sqcup b^{\smile} ; c \sqcup (\mathbb{I} \sqcap \mathbb{T} ; \mathfrak{a} ; SCB)) ;(G \sqcup c^{\smile} ; d \sqcup (\mathbb{I} \sqcap \mathbb{T} ; \mathfrak{b} ; SCB)) \\
= & S ;(F ; G \sqcup b^{\smile} ; c ; c^{\smile} ; d \sqcup (\mathbb{I} \sqcap \mathbb{T} ; \mathfrak{a} ; SCB) ;(\mathbb{I} \sqcap \mathbb{T} ; \mathfrak{b} ; SCB) \sqcup \perp\!\!\!\perp \sqcup \perp\!\!\!\perp \sqcup \perp\!\!\!\perp) \\
= & S ;(F ; G \sqcup b^{\smile} ; d \sqcup (\mathbb{I} \sqcap \mathbb{T} ; \mathfrak{a} ; SCB))
\end{array}$$

The occurrences of $\perp\!\!\!\perp$ result from the fact that inner nodes, lower borders and self-encapsulation borders are mutually exclusive.

vi) Binding preservation: First we have

$$F ; G ; B \sqcap \mathbb{T} ; F ; G \sqsubseteq F ; B ; G = (F ; B \sqcap \mathbb{T} ; F) ; G \sqsubseteq B ; F ; G$$

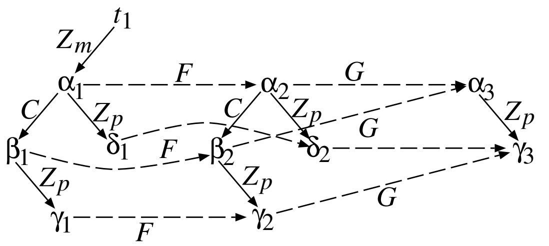
and $B ; F ; G \sqsubseteq F ; B ; G \sqsubseteq F ; G ; B$. The other inclusions are shown analogously.

vii) Preservation of internally bound variables: Let α and β be two points with $\alpha^\smile ; \beta \sqsubseteq G^\smile ; F^\smile ; W ; F ; G \sqcap B ; B^\smile \sqcap \mathbb{T} ; G ; B^\smile$. Then $\alpha ; \mathbb{B}^\smile \neq \perp\!\!\!\perp$, and using the fact that $F^\smile$ and $G^\smile$ are correspondences, too, we can calculate (applying Lemma 4.19 for readability without first deriving the explicit intersections with the respective domains):

$$\begin{array}{rcll}
\alpha ; \mathbb{B}^\smile & \sqsubseteq & \beta ; G^\smile ; F^\smile ; W ; F ; G ; \mathbb{B}^\smile_m & \\
& \sqsubseteq & \beta ; G^\smile ; F^\smile ; W ; F ; W ; \mathbb{B}^\smile ; (G \| \mathbb{I}) & \text{Lemma 4.19} \\
& \sqsubseteq & \beta ; G^\smile ; F^\smile ; W ; W ; F ; \mathbb{B}^\smile ; (G \| \mathbb{I}) & \text{Variable preservation of } F \\
& \sqsubseteq & \beta ; G^\smile ; F^\smile ; W ; W ; \mathbb{B}^\smile ; (F ; G \| \mathbb{I}) & \text{Lemma 4.19} \\
& \sqsubseteq & \beta ; G^\smile ; W ; F^\smile ; \mathbb{B}^\smile ; (F ; G \| \mathbb{I}) & \text{Variable preservation of } F \\
& \sqsubseteq & \beta ; G^\smile ; W ; \mathbb{B}^\smile ; (F^\smile ; F ; G \| \mathbb{I}) & \text{Lemma 4.19} \\
& \sqsubseteq & \beta ; W ; \mathbb{B}^\smile ; (G^\smile ; F^\smile ; F ; G \| \mathbb{I}) \ . & \text{Lemma 4.19}
\end{array}$$

Since $\beta ; W ; \mathbb{B}^\smile ; \pi_{\mathcal{V} \times \mathcal{E}^*} = \beta ; B = \alpha ; B = \alpha ; \mathbb{B}^\smile ; \pi_{\mathcal{V} \times \mathcal{E}^*}$, this implies $\alpha ; \mathbb{B}^\smile \sqsubseteq \beta ; W ; \mathbb{B}^\smile$ and with the univalence of $\mathbb{B}$ we obtain the desired $\alpha \sqsubseteq \beta ; W$.

viii) Encapsulation respect: If $F ; G$ violates the condition of encapsulation respect, then consider a first witness $\alpha_1 \sqsubseteq \mathbb{T} ; (F ; G ; G^\smile ; F^\smile \sqcap C^\smile)$ on a path m from an unencapsulated node t_1. Then contradictions to the encapsulation respect of F and G can be derived by commuting paths to bound and free variables over F and G as sketched in the following picture:



ix) Preservation of variables: analogously to binding preservation.

x) Preservation of non-locally bound variables:

$$\begin{array}{rcl}
F ; G \sqcap \overline{B ; \underline{a}^\smile ; \mathbb{T}} \sqcap W ; \mathbb{T} & = & (F \sqcap \overline{B ; \underline{a}^\smile ; \mathbb{T}} \sqcap W ; \mathbb{T}) ; G \\
& \sqsubseteq & (W \sqcap \overline{\mathbb{T} ; \underline{b} ; B^\smile}) ; G \\
& \sqsubseteq & W ; (G \sqcap \overline{B ; \underline{b}^\smile ; \mathbb{T}} \sqcap W ; \mathbb{T}) \\
& \sqsubseteq & W ; W = W
\end{array}$$

□

This composability together with the convertibility of correspondences and the simple identical correspondences shows that on a fixed graph, segments with injective lower borders together with correspondences form a self-dual category — associativity of the composition of correspondences is preserved from the associativity of relational composition. The existence of a correspondence between segments with injective lower borders therefore is an equivalence relation. Furthermore, correspondences between two segments form a complete lattice; the composition of two minimal correspondences however is not always minimal.

5 Explicit Relations between Term Graphs

Corresponding to substitutions in the context of conventional term rewriting, or valuations in that of CRSs, we need a concept of homomorphisms in graphs. Some places mention that the application of valuations yields homomorphisms between metaterms in CRSs, but there rarely is an explicit definition of such homomorphisms. Actually, for different rôles in the rewriting step, our term graphs require different notions of structure preserving morphisms — see 2.4.

In this section we concentrate on the common substrate of these different kinds of morphisms, on the underlying category that provides the crucial composition definition.

Conventionally, term graph homomorphisms are subject to conditions restricted to the constant nodes—these explicitly include bound variables in the few places where bound variables are mentioned in this context [39, 32]. For the corresponding definition here, the conditions are restricted to at least constant nodes and bindable variables. Of course we have to add conditions for binding and variable identity; we then obtain what we call *embeddings*.

In contrast to the embedding concept of [20, 21], however, we shall give a more general foundation in this paper.

The approach to homomorphisms from one graph G_1 to another graph G_2, that has been pursued in [20, 21] is based on considering total functions between the respective node sets, i.e.,on relations of type $\mathcal{V}_1 \to \mathcal{V}_2$.

For making the essential difference between matching a constant node and matching a metavariable clearer, in this paper we shall consider relations between the node set of G_1 on one hand and either nodes or intervals in G_2 on the other hand.

5.1 Explicit Relations — Definition and Basic Properties

With what has been said above, it is clear that the image type of matchings should be a direct sum, see Def. 3.18. Therefore, we define:

Definition 5.1
An **explicit relation** between two graphs G_1 and G_2 is a relation $\Phi : \mathcal{V}_1 \leftrightarrow (\mathcal{V}_2 + \mathcal{I}_2)$.

i) $\Phi_0 : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ with $\Phi_0 := \Phi ; \iota^{\smile}$ is the **constant part** of Φ.

ii) $\Phi_\vartriangle : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ with $\Phi_\vartriangle := \Phi ;[\mathbb{I} +\!\!+ \pi_{\mathcal{I}_2}]$ is the **immediate image relation** between the node sets underlying the explicit relation Φ; intervals are represented by their upper border.

iii) $\Phi_\perp : \mathcal{V}_1 \leftrightarrow \mathcal{E} \times \mathcal{V}_2$ relates in a certain sense nodes of G_1 that are mapped to intervals in G_2 to the lower borders of those intervals: $\Phi_\perp := \Phi ; \kappa^{\smile} ; \rho_{\mathcal{I}_2} ; \in^{\smile}_{\mathcal{E}\leftrightarrow\mathcal{V}_2}$

Φ is called *half-coherent*, iff $\mathbb{T} ; \Phi ; \kappa^{\smile} \sqsubseteq \mathbb{T} ; \mathit{HalfCoh}$. □

Obviously, $\Phi_\vartriangle$ is a function if and only if Φ is a function, and $\Phi_\vartriangle$ is total if and only if Φ is total. Furthermore, $\Phi_0 \sqsubseteq \Phi_\vartriangle$ always holds. In [20, 21] the starting point for considering term graph homomorphisms was a total function $F : \mathcal{V}_1 \to \mathcal{V}_2$ between the node sets of two graphs. In the setting of this paper, F corresponds to $\Phi_\vartriangle$.

The vector $\mathbb{T} ; \kappa ; \Phi^{\smile}$ is called the vector of **active metavariables** since only these metavariables are related to intervals.

We now list the "obvious" homomorphism conditions for our term graphs with bound variables, generalising the naming conventions in [20, 21], where an embedding is a total function F with all properties of Def. 5.2 holding for its restriction to outside active metavariables. The only difference is that here we had to move preservation of the distinctness of equally bound variables into Def. 6.2 — in the context of univalent embeddings that property is compositional, but not anymore for general relations.

Embeddings preserve structure only outside active metavariables, and there the conditions are similar to conventional homomorphism conditions, only the special circumstances of partiality of the constant part together with the restrictive environment of term graphs in comparison to general graphs require a shape of the conditions that exposes the fact that with term graphs, the structure that is absent is almost as important as the structure that is present:

Definition 5.2
An explicit relation Φ from G_1 to G_2 is an **embedding** iff
i) Φ_0 preserves labels: $\Phi_0^{\smile} ; M_1 \sqsubseteq M_2$,
ii) Φ_0 preserves successors: $(\Phi_0^{\smile} \| \mathbb{I}) ; S_1 \sqsubseteq S_2 ; \Phi_{\vartriangle}^{\smile}$,
iii) Φ_0 preserves bindings: $\Phi_0^{\smile} ; B_1 \sqsubseteq B_2 ; \Phi_0^{\smile}$, and
iv) Φ_0 preserves variables: $\Phi_0^{\smile} ; W_1 \sqsubseteq W_2 ; \Phi_0^{\smile}$. □

Lemma 5.3
$\Phi_0 ; M_2 \sqsubseteq M_1$.

Proof: With the univalence of the labelling we get $\Phi_0^{\smile} ; \overline{M_1} = \Phi_0^{\smile} ; M_1 ; \overline{\mathbb{I}} \sqsubseteq M_2 ; \overline{\mathbb{I}} = \overline{M_2}$, and from this the statement follows with Schröder. □

While the embedding properties are exactly those conditions a function has to fulfill for being appropriate as a simple term graph homomorphism, for relations, and notably explicit relations, there is more that can go wrong. Accordingly we have to restrict explicit relations to an intuitive notion of "appropriateness":

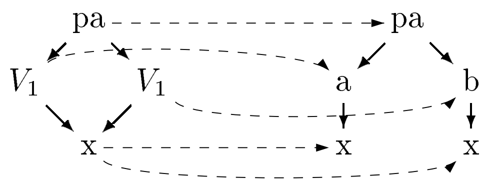
Definition 5.4
An explicit relation Φ from G_1 to G_2 is **appropriate**, iff
i) only metavariables are related to intervals: $\mathbb{T} ; \kappa ; \Phi^{\smile} \sqsubseteq \mathbb{T} ; (v_s \sqcup v_p) ; M_1^{\smile}$; if equality holds, then Φ is called **fully explicit**,
ii) variables related to intervals are disjoint from the constant part:

$$\mathbb{T} ; \kappa ; \Phi^{\smile} ; W_1 \sqsubseteq \overline{\mathbb{T} ; \Phi_0^{\smile}} \ ,$$

iii) the lower borders of image intervals of metavariables are contained in the images of the corresponding successors of the original nodes, i.e.,

$$\Phi_{\perp} \sqsubseteq \boxtimes.(S_1 ; \Phi_{\vartriangle}) \ .$$ □

An appropriate embedding that acts as a counterexample for equality of Def. 5.4 iii) is easily constructed — we only draw $\Phi_{\vartriangle}$:



From the definition of the relational sum (Def. 3.19) the following two properties are obvious from Def. 5.4 i) and ii):

LEMMA 5.5
i) $\Phi_\triangle \sqcap \Phi;\kappa^\smile;\mathbb{T} = \Phi;\kappa^\smile;\pi_{\mathcal{I}}$ ii) $\Phi_\triangle \sqcap \overline{\Phi;\kappa^\smile;\mathbb{T}} = \Phi_0$ □

For every graph G, the explicit relation $\iota : \mathcal{V} \to \mathcal{V} + \mathcal{I}$ has the full identity $\mathbb{I}_\mathcal{V}$ on the nodeset of G as its constant part, and therefore obviously is an appropriate embedding from G to G.

A fully explicit identity on G is

$$\Theta := (\overline{M;(v_s \sqcup v_p)^\smile;\mathbb{T}} \sqcap \iota) \sqcup (M;(v_s \sqcup v_p)^\smile;\mathbb{T} \sqcap \langle\!\!\{ \mathbb{I}, [\![\boxtimes.S /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}}]\!] \}\!\!\rangle_{\mathcal{I}};\kappa) \ .$$

Proof of the appropriateness and embedding properties is trivial.

5.2 Conversion between Plain and Explicit Relations

The reverse direction to $\Phi_\triangle$, that is to calculate $\Phi : \mathcal{V}_1 \leftrightarrow (\mathcal{V}_2 + \mathcal{I}_2)$ from a relation $F : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ between node sets and a vector $p : \mathbb{1} \leftrightarrow \mathcal{V}_1$ of active metavariables, in general can only yield a reliable result if $F \sqcap p^\smile;\mathbb{T}$ (corresponding to $\Phi;\kappa^\smile$) is univalent, and then we could use

$$\Phi := {<}F{>}_p := (F \sqcap \overline{p^\smile;\mathbb{T}});\iota \sqcup \langle\!\!\{ F \sqcap p^\smile;\mathbb{T}, [\![\boxtimes.(S_1;F) /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \}\!\!\rangle_{\mathcal{I}};\kappa$$

If $\Phi;\kappa^\smile$ is univalent, then we obtain the inclusion ${<}\Phi_\triangle{>}_{\mathbb{T};\kappa;\Phi^\smile} \sqsupseteq \Phi$:

$$\begin{array}{cl}
 & {<}\Phi_\triangle{>}_{\mathbb{T};\kappa;\Phi^\smile} \\
= & (\Phi_\triangle \sqcap \overline{\Phi;\kappa^\smile;\mathbb{T}});\iota \sqcup \langle\!\!\{ \Phi_\triangle \sqcap \Phi;\kappa^\smile;\mathbb{T}, [\![\boxtimes.(S_1;\Phi_\triangle) /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
= & \Phi_0;\iota \sqcup \langle\!\!\{ \Phi;\kappa^\smile;\pi_{\mathcal{I}}, [\![\boxtimes.(S_1;\Phi_\triangle) /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
= & \Phi;\iota^\smile;\iota \sqcup \langle\!\!\{ \Phi;\kappa^\smile;\pi_{\mathcal{I}}, [\![\boxtimes.(S_1;\Phi_\triangle) /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
\sqsupseteq & \quad \langle \text{Def. 5.4 iii)} \rangle \\
 & \Phi;\iota^\smile;\iota \sqcup \langle\!\!\{ \Phi;\kappa^\smile;\pi_{\mathcal{I}}, [\![\Phi_\perp /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
= & \quad \langle \text{Def. 5.1 iii)} \rangle \\
 & \Phi;\iota^\smile;\iota \sqcup \langle\!\!\{ \Phi;\kappa^\smile;\pi_{\mathcal{I}}, [\![\Phi;\kappa^\smile;\rho_{\mathcal{I}_2};\in^\smile_{\mathcal{E}\leftrightarrow\mathcal{V}_2} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
= & \quad \langle \Phi;\kappa^\smile \text{ univalent, Lemma 3.23} \rangle \\
 & \Phi;\iota^\smile;\iota \sqcup \langle\!\!\{ \Phi;\kappa^\smile;\pi_{\mathcal{I}_2}, \Phi;\kappa^\smile;\rho_{\mathcal{I}_2} \sqcup ([\![\perp\!\!\!\perp /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}]\!] \sqcap \overline{\Phi;\kappa^\smile;\rho_{\mathcal{I}_2};\mathbb{T}}) \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
= & \quad \left\langle \Phi;\kappa^\smile \text{ univalent, } \langle\!\!\{ R;S, \overline{R;\mathbb{T}} \}\!\!\rangle = \perp\!\!\!\perp \right\rangle \\
 & \Phi;\iota^\smile;\iota \sqcup \Phi;\kappa^\smile;\langle\!\!\{ \pi_{\mathcal{I}_2}, \rho_{\mathcal{I}_2} \}\!\!\rangle_{\mathcal{I}_2};\kappa \\
= & \Phi
\end{array}$$

The equality ${<}\Phi_\triangle{>}_{\mathbb{T};\kappa;\Phi^\smile} = \Phi$ holds if Def. 5.4 iii) is an equality for Φ, as can be seen from the proof.

On the other hand, $({<}F{>}_p)_\triangle = F$ is comparatively easy to see and does not even depend on the univalence of F:

$$
\begin{array}{cl}
 & (<F>_p)_\triangle \\
= & ((F \sqcap \overline{p^\smile \mathbin{;} \mathbb{T}}) \mathbin{;} \iota \sqcup \langle\!\![\, F \sqcap p^\smile \mathbin{;} \mathbb{T}, [\![\boxbslash.(S_1 \mathbin{;} F) /\!\!/ \in_{\mathcal{E} \leftrightarrow \mathcal{V}_2}]\!] \,]\!\!\rangle_{\mathcal{I}} \mathbin{;} \kappa) \mathbin{;} [\mathbb{I} +\!\!+ \pi_{\mathcal{I}_2}] \\
= & (F \sqcap \overline{p^\smile \mathbin{;} \mathbb{T}}) \sqcup \langle\!\![\, F \sqcap p^\smile \mathbin{;} \mathbb{T}, [\![\boxbslash.(S_1 \mathbin{;} F) /\!\!/ \in_{\mathcal{E} \leftrightarrow \mathcal{V}_2}]\!] \,]\!\!\rangle_{\mathcal{I}} \mathbin{;} \pi_{\mathcal{I}_2} \\
= & (F \sqcap \overline{p^\smile \mathbin{;} \mathbb{T}}) \sqcup (F \sqcap p^\smile \mathbin{;} \mathbb{T}) \\
= & F
\end{array}
$$

5.3 Composition of Explicit Relations

Composition of explicit relations that are both not univalent obviously poses difficult problems with respect to the choice of the lower borders of image intervals.

One reason we nevertheless present a first stab at this subject here is that we want to show a few more involved definitions that lead the relational notation to the borderline of what is manageable for human beings.

The key auxiliary function takes an explicit relation and gives as result an induced relation between the paths in the two graphs:

Definition 5.6

For an explicit relation $\Phi : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2 + \mathcal{I}_2$ we define $\Phi_\bowtie : (\mathcal{V}_1 \times \mathcal{E}^*) \leftrightarrow (\mathcal{V}_2 \times \mathcal{E}^*)$ as the smallest relation satisfying the following three conditions:

i) empty paths with their starting points related by the immediate image relation $\Phi_\triangle$ are related by $\Phi_\bowtie$,

ii) if a path in G_1 ending in a node that is treated as a constant by Φ is related to a path in G_2, then the prolongations of these two paths are related by $\Phi_\bowtie$ if their respective endpoints are related by the immediate image relation $\Phi_\triangle$, and

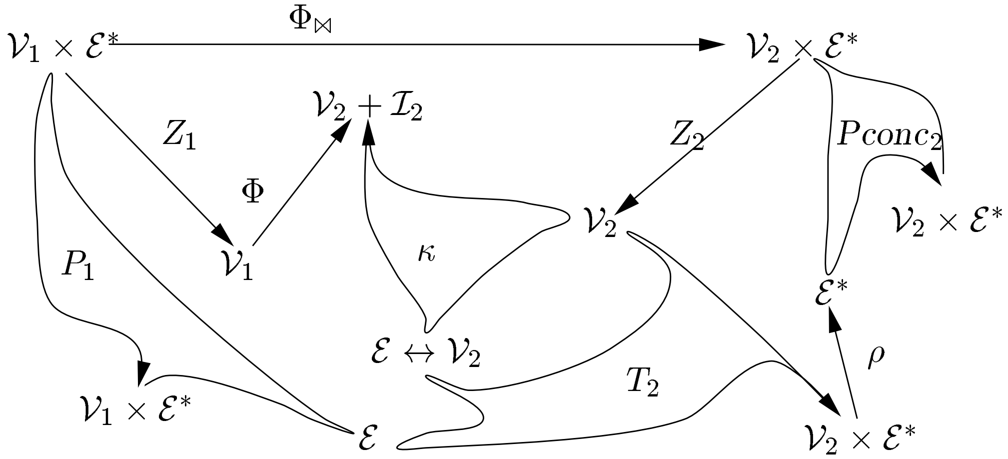
iii) if a path p_1 in G_1 ending in a node v that is treated as a variable by Φ is related to a path p_2 in G_2, then the prolongation of p_1 by some edge label e is related to the catenation of p_2 with those paths that go through the image of v leaving at the lower border for e, if the respective endpoints of the prolonged paths are related by the immediate image relation $\Phi_\triangle$.

Spelled out relationally, these conditions correspond to the following inclusions:

$$
\begin{array}{rcl}
(\Phi_\triangle \,\|\, empty^\smile \mathbin{;} empty) & \sqsubseteq & \Phi_\bowtie \\
P_1^\smile \mathbin{;} (\Phi_\bowtie \sqcap Z_1 \mathbin{;} \Phi \mathbin{;} \iota^\smile \mathbin{;} Z_2^\smile \| \mathbb{I}) \mathbin{;} P_2 \sqcap Z_1 \mathbin{;} \Phi_\triangle \mathbin{;} Z_2^\smile & \sqsubseteq & \Phi_\bowtie \\
P_1^\smile \mathbin{;} \Big((Z_1 \mathbin{;} \Phi \mathbin{;} \kappa^\smile \| \mathbb{I}) \mathbin{;} \langle\!\![\, \pi \mathbin{;} \pi \mathbin{;} Z_2^\smile, T_2 \mathbin{;} \rho \,]\!\!\rangle \sqcap \pi \mathbin{;} \Phi_\bowtie \mathbin{;} \pi^\smile \Big) \mathbin{;} Pconc_2 \sqcap Z_1 \mathbin{;} \Phi_\triangle \mathbin{;} Z_2^\smile & \sqsubseteq & \Phi_\bowtie
\end{array}
$$

Such a relation exists, since these conditions give rise to a continuous (wrt. the inclusion ordering "$\sqsubseteq$") relation functional, of which $\Phi_\bowtie$ is then the least fixed point and can be iteratively approximated, allowing the inductive proofs below. □

For giving a better intuition of how the relation algebraic formalisation of condition (iii) above relates to the prose version, we draw a hypergraph showing all the objects and relations involved in the first term of the relation algebraic formalisation. This diagram can actually be seen as another representation of that term; for a formal treatment of these diagrams and how to use them in proofs see [22].



We shall need two simple properties of $\Phi_{\bowtie}$:

LEMMA 5.7
i) $\Phi_{\bowtie} ; \pi \sqsubseteq \pi ; \Phi_{\triangle}$ ii) $\Phi_{\bowtie} ; Z_2 \sqsubseteq Z_1 ; \Phi_{\triangle}$.

Proof:
i) $(\Phi_{\triangle} \| empty^{\smile} ; empty) ; \pi \quad \sqsubseteq \quad \pi ; \Phi_{\triangle}$

$$\begin{aligned} P_1^{\smile} ; (\Phi_{\bowtie} \sqcap Z_1 ; \Phi ; \iota^{\smile} ; Z_2^{\smile} \| \mathbb{I}) ; P_2 \sqcap Z_1 ; \Phi_{\triangle} ; Z_2^{\smile} \ &\sqsubseteq\ P_1^{\smile} ; \pi ; \Phi_{\bowtie} ; \pi^{\smile} ; P_2 ; \pi \\ \sqsubseteq P_1^{\smile} ; \pi ; \Phi_{\bowtie} ; \pi^{\smile} ; \pi ; \pi \ = \ P_1^{\smile} ; \pi ; \Phi_{\bowtie} ; \pi \ &\sqsubseteq\ P_1^{\smile} ; \pi ; \pi ; \Phi_{\triangle} \ \sqsubseteq \ \pi^{\smile} ; \Phi_{\triangle} \end{aligned}$$

For the third condition of Def. 5.6, the same steps can be applied.
ii) Obvious from Def. 5.6. □

Building on this path relation, we want to find a way that lifts an explicit relation between two graphs to a relation between intervals on the two graphs. The crucial idea is that we want to restrict the lower border nodes for an image of an interval to those images of the corresponding lower border of the original interval that are reachable from the upper border of the image interval by a path that is an image of a path through the original interval, i.e., by a path that is via $\Phi_{\bowtie}$ related to that path through the original interval.

First we define an auxiliary relation that relates an interval to triples of candidates for upper borders together with an edge label for the lower border and an appropriate lower border node that is reachable from the upper border via a path that is related by $\Phi_{\bowtie}$ to a path through the source interval to a lower border node for the same edge label:

DEFINITION 5.8
For an explicit relation $\Phi : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2 + \mathcal{I}_2$ we define $\Phi_{\mathsf{X}} : \mathcal{I}_1 \leftrightarrow \mathcal{V}_2 \times (\mathcal{E} \times \mathcal{V}_2)$ by

$$\Phi_{\mathsf{X}} := \boxslash\!\!\rightarrow . T_1 ; (\Phi_{\bowtie} ; [\![\pi, Z_2]\!] \| \mathbb{I}) ; \trianglelefteq ; (\mathbb{I} \| \times) \qquad \square$$

For later proofs we shall need the following properties:

LEMMA 5.9
i) $\Phi_{\mathsf{X}} ; \pi = \pi^{\smile} ; T_1 ; \Phi_{\bowtie} ; \pi$, ii) $\Phi_{\mathsf{X}} \sqsubseteq \pi_{\mathcal{I}_1} ; \Phi_{\triangle} ; \pi^{\smile}$, iii) $\Phi_{\mathsf{X}} ; \rho \sqsubseteq \rho_{\mathcal{I}_1} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_1} ; (\mathbb{I} \| \Phi_{\triangle})$.

Proof:

i)

$$
\begin{array}{rll}
& \Phi_{\mathsf{X}} ; \pi & \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle \| \mathbb{I}) ; \trianglelefteq ;(\mathbb{I} \| \boxtimes) ; \pi & \text{Def. 5.8} \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle \| \mathbb{I}) ; \trianglelefteq ; \pi & \forall R \bullet (\mathbb{I} \| R) ; \pi = \pi \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle \| \mathbb{I}) ; \pi ; \pi & \trianglelefteq ; \pi = \pi ; \pi \\
= & \boxslash.T_1 ; \pi ; \Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle ; \pi & \forall R, S \bullet (R \| S) ; \pi = \pi ; R \\
= & \pi\breve{} ; T_1 ; \Phi_{\bowtie} ; \pi & \text{3.17 ii)}, \forall R, S \bullet \langle\!| R, S |\!\rangle ; \pi = R
\end{array}
$$

ii)

$$
\begin{array}{rll}
& \Phi_{\mathsf{X}} & \\
\sqsubseteq & \Phi_{\mathsf{X}} ; \pi ; \pi\breve{} & \text{Def. 3.14} \\
= & \pi\breve{} ; T_1 ; \Phi_{\bowtie} ; \pi ; \pi\breve{} & \text{i)} \\
\sqsubseteq & \pi\breve{} ; T_1 ; \pi_{\mathcal{V} \times \mathcal{E}^*} ; \Phi_{\triangle} ; \pi\breve{} & \text{Lemma 5.7 i)} \\
= & {}_{\mathcal{I}}[\!\rangle \pi\breve{}_{\mathcal{V} \times \mathcal{E}^*}, \in\breve{}_{\mathcal{E} \leftrightarrow \mathcal{V}} ; \rho ; Z\breve{} \,\langle\!] ; \pi_{\mathcal{V} \times \mathcal{E}^*} ; \Phi_{\triangle} ; \pi\breve{} & \text{Lemma 4.16 i)} \\
\sqsubseteq & \pi_{\mathcal{I}_1} ; \pi\breve{}_{\mathcal{V} \times \mathcal{E}^*} ; \pi_{\mathcal{V} \times \mathcal{E}^*} ; \Phi_{\triangle} ; \pi\breve{} & \forall R, S \bullet [\!\rangle R, S \langle\!] \sqsubseteq \pi ; R \\
= & \pi_{\mathcal{I}_1} ; \Phi_{\triangle} ; \pi\breve{} & \text{Def. 3.14}
\end{array}
$$

iii)

$$
\begin{array}{rll}
& \Phi_{\mathsf{X}} ; \rho & \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle \| \mathbb{I}) ; \trianglelefteq ;(\mathbb{I} \| \boxtimes) ; \rho & \text{Def. 5.8} \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle \| \mathbb{I}) ; \trianglelefteq ; \rho ; \boxtimes & \forall R, S \bullet (R \| S) ; \rho = \rho ; S \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ;\langle\!| \pi, Z_2 |\!\rangle \| \mathbb{I}) ;\langle\!| \rho, \pi ; \rho |\!\rangle & \trianglelefteq ; \rho ; \boxtimes = \langle\!| \rho, \pi ; \rho |\!\rangle \\
= & \boxslash.T_1 ;(\Phi_{\bowtie} ; Z_2 \| \mathbb{I}) ; \boxtimes & \\
\sqsubseteq & \boxslash.T_1 ;(Z_1 ; \Phi_{\triangle} \| \mathbb{I}) ; \boxtimes & \text{Lemma 5.7 ii)} \\
= & \boxbslash.(T_1 ; Z_1 ; \Phi_{\triangle}) & \forall R, S \bullet \boxslash.R ;(S \| \mathbb{I}) ; \boxtimes = \boxbslash.(R ; S) \\
\sqsubseteq & \boxbslash.(\boxtimes.(\rho_{\mathcal{I}_1} ; \in\breve{}_{\mathcal{E} \leftrightarrow \mathcal{V}_1}) ; \Phi_{\triangle}) & \text{Lemma 4.16 ii)} \\
= & \rho_{\mathcal{I}_1} ; \in\breve{}_{\mathcal{E} \leftrightarrow \mathcal{V}_1} ;(\mathbb{I} \| \Phi_{\triangle}) & \text{Lemma 3.17 iii)} \quad \square
\end{array}
$$

For constructing the lower borders of intervals from Φ_{X} we have to reshuffle the typing so that we can use the symmetric quotient for the transition to relation values, and then we have to reshuffle again for obtaining our desired result type; in addition we have to ensure that only intervals are formed that have an appropriate upper border — the symmetric quotient happily forms empty lower borders for arbitrary unrelated upper border nodes, while only those intervals can be accepted that have image nodes of the original upper border as their upper borders; these then can also have empty lower borders under appropriate circumstances:

Definition 5.10

For an explicit relation $\Phi : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2 + \mathcal{I}_2$ we define $\Phi_{\square} : \mathcal{I}_1 \leftrightarrow \mathcal{I}_2$ by

$$
\Phi_{\square} := \boxbslash.(\llbracket \boxslash.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E} \leftrightarrow \mathcal{V}_2} \rrbracket) \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} ; \pi\breve{}_{\mathcal{I}_2} \qquad \square
$$

With $\Phi_{\square}$ we still have no insurance that there is no intrusion of unwanted lower borders from what could be seen as overlapping interval images, but whenever Φ is univalent we are on the safe side, otherwise $\Phi_{\square}$ still is a plausible starting point e.g. for interactive refinement.

The following properties will be used later:

Lemma 5.11

i) $\Phi_{\square} ; \pi_{\mathcal{I}_2} = \pi_{\mathcal{I}_1} ; \Phi_{\triangle}$, ii) $\Phi_{\square} ; \rho_{\mathcal{I}_2} ; \in\breve{}_{\mathcal{E} \leftrightarrow \mathcal{V}_2} = \Phi_{\mathsf{X}} ; \rho$.

Proof:

i)

$$
\begin{array}{rll}
 & \Phi_{\Box} ; \pi_{\mathcal{I}_2} & \\
= & (\overrightarrow{\boxbslash}.(\llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket) \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} ; \pi_{\mathcal{I}_2}^{\smile}) ; \pi_{\mathcal{I}_2} & \text{Def. 5.10} \\
= & \overrightarrow{\boxbslash}.(\llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket) ; \pi_{\mathcal{I}_2} \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} & \pi_{\mathcal{I}_2} \text{ univalent} \\
= & \rho^{\smile} ;(\pi \sqcap \llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket ; \mathbb{T}) \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} & \text{Lemma 3.17 iv)} \\
= & \rho^{\smile} ;(\pi \sqcap \mathbb{T}) \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} & \text{Def. 3.22 ii)} \\
= & \mathbb{T} \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} & \text{Def. 3.14} \\
= & \pi_{\mathcal{I}_1} ; \Phi_{\triangle} &
\end{array}
$$

ii)

$$
\begin{array}{rll}
 & \Phi_{\Box} ; \rho_{\mathcal{I}_2} ; \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}^{\smile} & \\
= & (\overrightarrow{\boxbslash}.(\llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket) \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} ; \pi_{\mathcal{I}_2}^{\smile}) ; \rho_{\mathcal{I}_2} ; \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}^{\smile} & \text{Def. 5.10} \\
= & \overrightarrow{\boxbslash}.(\{\!| \mathbb{I}, \pi_{\mathcal{I}_1} ; \Phi_{\triangle} |\!\} ; \mathbb{T} \sqcap \llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket) ; \rho_{\mathcal{I}_2} ; \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}^{\smile} & \text{Lemma 3.17 v)} \\
= & \rho^{\smile} ;(\{\!| \mathbb{I}, \pi_{\mathcal{I}_1} ; \Phi_{\triangle} |\!\} ; \mathbb{T} \sqcap \llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket) ; \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}^{\smile} & \text{Lemma 3.17 vi)} \\
= & \rho^{\smile} ;(\{\!| \mathbb{I}, \pi_{\mathcal{I}_1} ; \Phi_{\triangle} |\!\} ; \mathbb{T} \sqcap \llbracket \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} /\!\!/ \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2} \rrbracket ; \in_{\mathcal{E}\leftrightarrow\mathcal{V}_2}^{\smile}) & \text{Lemma 3.5} \\
= & \rho^{\smile} ;(\{\!| \mathbb{I}, \pi_{\mathcal{I}_1} ; \Phi_{\triangle} |\!\} ; \mathbb{T} \sqcap \overleftarrow{\boxslash}.\Phi_{\mathsf{X}}) & \text{Lemma 3.21 iii)} \\
= & \rho^{\smile} ; \overleftarrow{\boxslash}.(\Phi_{\mathsf{X}} \sqcap \pi_{\mathcal{I}_1} ; \Phi_{\triangle} ; \pi^{\smile}) & \text{Lemma 3.17 vii)} \\
= & \rho^{\smile} ; \overleftarrow{\boxslash}.\Phi_{\mathsf{X}} & \text{Lemma 5.9 ii)} \\
= & \Phi_{\mathsf{X}} ; \rho & \text{Lemma 3.17 viii)} \quad \Box
\end{array}
$$

In spite of the incomplete reliability of $\Phi_{\Box}$, we can use it anyway for defining a composition of explicit relations:

Definition 5.12

Let an explicit relation Φ from G_1 to G_2 and another explicit relation Ψ from G_2 to G_3 be given. Then $(\Phi \fatsemi \Psi)$ is an explicit relation from G_1 to G_3 which is defined as follows:

$$(\Phi \fatsemi \Psi) := \Phi ;[\Psi \mathbin{+\!\!\!+} \Psi_{\Box} ; \kappa] \qquad \Box$$

With respect to constant part and immediate image, this definition behaves as can reasonably be expected:

Theorem 5.13

If Φ is an explicit relation from G_1 to G_2 and Ψ another explicit relation from G_2 to G_3, then

i) the constant part distributes over composition: $(\Phi \fatsemi \Psi)_0 = \Phi_0 ; \Psi_0$,

ii) the immediate image distributes over composition: $(\Phi \fatsemi \Psi)_{\triangle} = \Phi_{\triangle} ; \Psi_{\triangle}$. $\Box$

Proof:

i) Distributivity of constant part over composition:

$$
\begin{array}{rcl}
(\Phi \fatsemi \Psi)_0 & = & \Phi ;[\Psi \mathbin{+\!\!\!+} \Psi_{\Box} ; \kappa] ; \iota^{\smile} \;=\; \Phi ;[\Psi ; \iota^{\smile} \mathbin{+\!\!\!+} \Psi_{\Box} ; \kappa ; \iota^{\smile}] \;=\; \Phi ;[\Psi_0 \mathbin{+\!\!\!+} \mathbb{L}] \\
 & = & \Phi ;(\iota^{\smile} ; \Psi_0 \sqcup \kappa^{\smile} ; \mathbb{L}) \;=\; \Phi ; \iota^{\smile} ; \Psi_0 \;=\; \Phi_0 ; \Psi_0
\end{array}
$$

ii) Distributivity of immediate image:

$$
\begin{array}{rll}
 & (\Phi \fatsemi \Psi)_{\triangle} \;=\; (\Phi \fatsemi \Psi) ;[\mathbb{I} \mathbin{+\!\!\!+} \pi_{\mathcal{I}_3}] & \\
= & \Phi ;[\Psi \mathbin{+\!\!\!+} \Psi_{\Box} ; \kappa] ;[\mathbb{I} \mathbin{+\!\!\!+} \pi_{\mathcal{I}_3}] & \\
= & \Phi ;[\Psi ;[\mathbb{I} \mathbin{+\!\!\!+} \pi_{\mathcal{I}_3}] \mathbin{+\!\!\!+} \Psi_{\Box} ; \kappa ;[\mathbb{I} \mathbin{+\!\!\!+} \pi_{\mathcal{I}_3}]] & \\
= & \Phi ;[\Psi_{\triangle} \mathbin{+\!\!\!+} \Psi_{\Box} ; \pi_{\mathcal{I}_3}] & \\
= & \Phi ;[\Psi_{\triangle} \mathbin{+\!\!\!+} \pi_{\mathcal{I}_2} ; \Psi_{\triangle}] & \text{Lemma 5.11 i)} \\
= & \Phi ;[\mathbb{I} \mathbin{+\!\!\!+} \pi_{\mathcal{I}_2}] ; \Psi_{\triangle} \;=\; \Phi_{\triangle} ; \Psi_{\triangle} & \quad \Box
\end{array}
$$

Comparatively easy is then the proof of the embedding properties of the composition:

THEOREM 5.14 (Compositionality of embeddings)
If Φ is an embedding from G_1 to G_2 and Ψ is an embedding from G_2 to G_3, then $(\Phi \mathbin{⨾} \Psi)$ also is an embedding.

Proof: Let $\Xi = (\Phi \mathbin{⨾} \Psi)$, then we have $\Xi_0 = \Phi_0 ; \Psi_0$ according to Theorem 5.13, and the individual embedding conditions are easily verified:

i) Label preservation: $\Xi_0^{\smile} ; M_1 = \Psi_0^{\smile} ; \Phi_0^{\smile} ; M_1 \sqsubseteq \Psi_0^{\smile} ; M_2 \sqsubseteq M_3$,

ii) Successor preservation:

$$(\Xi_0^{\smile} \| \mathbb{I}) ; S_1 = (\Psi_0^{\smile} \| \mathbb{I}) ; (\Phi_0^{\smile} \| \mathbb{I}) ; S_1 \sqsubseteq (\Psi_0^{\smile} \| \mathbb{I}) ; S_2 ; \Phi_{\triangle}^{\smile} \sqsubseteq S_3 ; \Psi_{\triangle}^{\smile} ; \Phi_{\triangle}^{\smile} = S_3 ; \Xi_{\triangle}^{\smile} \ ,$$

iii) Binding preservation: $\Xi_0^{\smile} ; B_1 = \Psi_0^{\smile} ; \Phi_0^{\smile} ; B_1 \sqsubseteq \Psi_0^{\smile} ; B_2 ; \Phi_0^{\smile} \sqsubseteq B_3 ; \Psi_0^{\smile} ; \Phi_0^{\smile} = B_3 ; \Xi_0^{\smile}$,

iv) Variable pres.: $\Xi_0^{\smile} ; W_1 = \Psi_0^{\smile} ; \Phi_0^{\smile} ; W_1 \sqsubseteq \Psi_0^{\smile} ; W_2 ; \Phi_0^{\smile} \sqsubseteq W_3 ; \Psi_0^{\smile} ; \Phi_0^{\smile} = W_3 ; \Xi_0^{\smile}$ $\square$

A little more effort and some restriction are necessary for the proof of appropriateness:

THEOREM 5.15
If Φ is an appropriate embedding from G_1 to G_2 and Ψ is an appropriate embedding from G_2 to G_3, and if $\Phi_0^{\smile} ; \Phi_0 ; \Psi_0 ; \mathbb{T} \sqsubseteq \Psi_0 ; \mathbb{T}$, i.e., if with one image of a node under Φ_0 being a constant for Ψ, all images of that node are constants for Ψ, then $(\Phi \mathbin{⨾} \Psi)$ is appropriate (in the sense of Def. 5.4).

Proof:

i) Only metavariables are related to intervals:

$$\begin{array}{lll}
 & \mathbb{T} ; \kappa ; (\Phi \mathbin{⨾} \Psi)^{\smile} & \\
= & \mathbb{T} ; \kappa ; [\Psi \mathbin{+\!\!\!+} \Psi_{\square} ; \kappa]^{\smile} ; \Phi^{\smile} & \\
= & \mathbb{T} ; \kappa ; (\Psi^{\smile} ; \iota \sqcup \kappa^{\smile} ; \Psi_{\square}^{\smile} ; \kappa) ; \Phi^{\smile} & \text{Def. 3.19} \\
= & \mathbb{T} ; \kappa ; \Psi^{\smile} ; \iota ; \Phi^{\smile} \sqcup \mathbb{T} ; \kappa ; \kappa^{\smile} ; \Psi_{\square}^{\smile} ; \kappa ; \Phi^{\smile} & \text{Distr. of ``;'' over ``}\sqcup\text{''} \\
\sqsubseteq & \mathbb{T} ; (v_s \sqcup v_p) ; M_2^{\smile} ; \Phi_0^{\smile} \sqcup \mathbb{T} ; \kappa ; \Phi^{\smile} & \Psi\text{: Def. 5.4 i), } \Phi\text{: Def. 5.1 i)} \\
\sqsubseteq & \mathbb{T} ; (v_s \sqcup v_p) ; M_1^{\smile} & \Phi\text{: Lemma 5.3 i), Def. 5.4 i)}
\end{array}$$

ii) Variables related to intervals are disjoint from the constant part: with Schröder we obtain

$$\Phi_0 ; \overline{\Psi_0 ; \mathbb{T}} \sqsubseteq \overline{\Phi_0 ; \Psi_0 ; \mathbb{T}} \qquad (*)$$

from $\Phi_0^{\smile} ; \Phi_0 ; \Psi_0 ; \mathbb{T} \sqsubseteq \Psi_0 ; \mathbb{T}$, and then we can calculate:

$$\begin{array}{lll}
 & \mathbb{T} ; \kappa ; (\Phi \mathbin{⨾} \Psi)^{\smile} ; W_1 & \\
= & \mathbb{T} ; \kappa ; [\Psi \mathbin{+\!\!\!+} \Psi_{\square} ; \kappa]^{\smile} ; \Phi^{\smile} ; W_1 & \\
= & \mathbb{T} ; \kappa ; (\Psi^{\smile} ; \iota \sqcup \kappa^{\smile} ; \Psi_{\square}^{\smile} ; \kappa) ; \Phi^{\smile} ; W_1 & \text{Def. 3.19} \\
= & \mathbb{T} ; \kappa ; \Psi^{\smile} ; \iota ; \Phi^{\smile} ; W_1 \sqcup \mathbb{T} ; \kappa ; \kappa^{\smile} ; \Psi_{\square}^{\smile} ; \kappa ; \Phi^{\smile} ; W_1 & \text{Distr. of ``;'' over ``}\sqcup\text{''} \\
\sqsubseteq & \mathbb{T} ; \kappa ; \Psi^{\smile} ; \Phi_0^{\smile} ; W_1 \sqcup \mathbb{T} ; \kappa ; \Phi^{\smile} ; W_1 & \Phi\text{: Def. 5.1 i); } \forall R \bullet R \sqsubseteq \mathbb{T} \\
\sqsubseteq & \mathbb{T} ; \kappa ; \Psi^{\smile} ; W_2 ; \Phi_0^{\smile} \sqcup \overline{\mathbb{T} ; \Phi_0^{\smile}} & \Phi\text{: Def. 5.2 iv)} \\
\sqsubseteq & \overline{\mathbb{T} ; \Psi_0^{\smile}} ; \Phi_0^{\smile} \sqcup \overline{\mathbb{T} ; \Psi_0^{\smile} ; \Phi_0^{\smile}} & \text{Def. 5.4 ii), } \mathbb{T} \sqsupseteq \mathbb{T} ; \Psi_0^{\smile} \\
\sqsubseteq & \overline{\mathbb{T} ; \Psi_0^{\smile} ; \Phi_0^{\smile}} & (*) \\
= & \overline{\mathbb{T} ; (\Phi \mathbin{⨾} \Psi)_0^{\smile}} & \text{Theorem 5.13 i)}
\end{array}$$

iii) Successor consistency of the image intervals:

$$
\begin{array}{rcll}
(\Phi \mathbin{⨾} \Psi)_{0\perp} & = & \Phi ; [\Psi \mathbin{+\!\!+} \Psi_{\Box} ; \kappa] ; \kappa^{\smile} ; \rho_{\mathcal{I}_3} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_3} & \\
& = & \Phi ; [\Psi_{\perp} \mathbin{+\!\!+} \Psi_{\Box} ; \kappa ; \kappa^{\smile} ; \rho_{\mathcal{I}_3} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_3}] & \text{Def. 5.1 iii)} \\
& \sqsubseteq & \Phi ; [\boxslash.(S_2 ; \Psi_{\triangle}) \mathbin{+\!\!+} \Psi_{\Box} ; \rho_{\mathcal{I}_3} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_3}] & \text{Def. 5.4 iii)} \\
& = & \Phi ; \iota^{\smile} ; \boxslash.(S_2 ; \Psi_{\triangle}) \sqcup \Phi ; \kappa^{\smile} ; \Psi_{\Box} ; \rho_{\mathcal{I}_3} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_3} & \text{Def. 3.19}
\end{array}
$$

We now proceed separately for the two parts of the disjunction:

$$
\begin{array}{rll}
& \Phi ; \iota^{\smile} ; \boxslash.(S_2 ; \Psi_{\triangle}) & \\
= & \Phi_0 ; \boxslash.(S_2 ; \Psi_{\triangle}) & \text{Def. 5.1 i)} \\
= & \boxslash.((\Phi_0 \| \mathbb{I}) ; S_2 ; \Psi_{\triangle}) & \text{Lemma 3.17 ix)} \\
\sqsubseteq & \boxslash.(S_1 ; \Phi_{\triangle} ; \Psi_{\triangle}) & \text{with Def. 5.2 ii)} \\
= & \boxslash.(S_1 ; (\Phi \mathbin{⨾} \Psi)_{\triangle}) & \text{Theorem 5.13 ii)}
\end{array}
$$

And for the second part of the disjunction:

$$
\begin{array}{rll}
& \Phi ; \kappa^{\smile} ; \Psi_{\Box} ; \rho_{\mathcal{I}_3} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_3} & \\
= & \Phi ; \kappa^{\smile} ; \Psi_{\mathsf{X}} ; \rho & \text{Lemma 5.11 ii)} \\
\sqsubseteq & \Phi ; \kappa^{\smile} ; \rho_{\mathcal{I}_2} ; \in^{\smile}_{\mathcal{E} \leftrightarrow \mathcal{V}_2} ; (\mathbb{I} \| \Psi_{\triangle}) & \text{Lemma 5.9 iii)} \\
= & \Phi_{\perp} ; (\mathbb{I} \| \Psi_{\triangle}) & \text{Def. 5.1 iii)} \\
\sqsubseteq & \boxslash.(S_1 ; \Phi_{\triangle}) ; (\mathbb{I} \| \Psi_{\triangle}) & \text{Def. 5.4 iii)} \\
= & \boxslash.(S_1 ; \Phi_{\triangle} ; \Psi_{\triangle}) & \forall Q, R, S \bullet \llparenthesis Q, R \rrparenthesis ; (\mathbb{I} \| S) = \llparenthesis Q, R ; S \rrparenthesis \\
= & \boxslash.(S_1 ; (\Phi \mathbin{⨾} \Psi)_{\triangle}) & \text{Theorem 5.13 ii)}
\end{array}
$$
□

A stronger condition than $\Phi_0^{\smile} ; \Phi_0 ; \Psi_0 ; \mathbb{T} \sqsubseteq \Psi_0 ; \mathbb{T}$ is for example that all images of Φ_0 are constants for Ψ, i.e., $\mathbb{T} ; \Phi_0 \sqsubseteq \mathbb{T} ; \Psi_0^{\smile}$. One way to ensure this universally is to restrict appropriate embeddings to be fully explicit, since then

$$
\begin{array}{rcll}
\mathbb{T} ; \Phi_0 & = & \mathbb{T} ; \Phi_0^{\smile} ; \Phi_0 & \\
& \sqsubseteq & \overline{\mathbb{T} ; \kappa ; \Phi^{\smile} ; W_1} ; \Phi_0 & \text{Def. 5.4 ii)} \\
& = & \overline{\mathbb{T} ; (v_s \sqcup v_p) ; M_1^{\smile} ; W_1} ; \Phi_0 & \text{Def. 5.4 i)} \\
& = & \overline{\mathbb{T} ; (v_s \sqcup v_p) ; M_1^{\smile}} ; \Phi_0 & \text{Def. 4.2} \\
& = & \overline{\mathbb{T} ; (v_s \sqcup v_p)} ; M_1^{\smile} ; \Phi_0 & M_1 \text{ mapping} \\
& \sqsubseteq & \overline{\mathbb{T} ; (v_s \sqcup v_p)} ; M_2^{\smile} & \text{Def. 5.2 i)} \\
& = & \overline{\mathbb{T} ; \kappa ; \Psi^{\smile} ; W_2} & \text{Def. 5.4 i)} \\
& = & \mathbb{T} ; \Psi_0^{\smile} & \text{Def. 5.4 i) \& ii)}
\end{array}
$$

Proofs of the identity properties wrt. $(\_ \mathbin{⨾} \_)$ of the fully explicit identity Θ of page 30 are technically involved, but essentially simple; so we do not present them here.

The key for the associativity of $(\_ \mathbin{⨾} \_)$ is the distributivity of the explicit path relation, i.e., $(\Phi \mathbin{⨾} \Psi)_{\bowtie} = \Phi_{\bowtie} ; \Psi_{\bowtie}$, which only is assured to hold if Φ and Ψ are half-coherent. We leave out the lengthy proofs of associativity, too, but altogether we obtain:

Theorem 5.16 (**Rel-TGV**)
The term graphs of Def. 4.2 together with fully-explicit appropriate half-coherent embeddings form a category **Rel-TGV**. □

If attention is restricted to graphs with only 0-ary metavariables and without bindable variables, and at the same time to total functions, the morphisms of **Rel-TGV** exactly correspond to the homomorphisms to be found in the term graph literature.

However, our approach to term graph morphisms is novel first in that it extends to term graphs with bound variables and with metavariables with successors, and second that it is generalised to relations.

This category **Rel-TGV** can now be used as the underlying category for a heterogeneous algebraic definition of term graph rewriting as sketched in 2.4. This then allows for more flexibility with the choice of the horizontal and vertical arrows than the old approach based only on total functions.

6 Homomorphy

In this section we propose a selection for the vertical arrows in the rewriting scheme; i.e., for the matching homomorphisms from the rule sides into application graphs.

In 6.1 we introduce *fittings* that specialise embeddings to avoid "variable capturing", and *homomorphisms* that make sure that different nodes of one schemavariable are matched onto equivalent graph segments in the target.

While [20, 21] only discussed total functions for this purpose, we base the discussion here on the "*explicit relations*" of the last section. The presentation of fittings and homomorphisms does not get noticeably more complicated, and the extra effort pays back through the added generality.

In 6.2 we make a brief excursion discussing what categories of such homomorphisms still exist, and in 6.3 we give a sufficient condition for a homomorphism to be determined uniquely by the images of the base (or root) of its source.

6.1 Transfer of the CRS Redex Concept to Term Graphs

In CRSs, the (meta-)term resulting from applying a valuation to a metaterm is sometimes called a kind of homomorphic image. If one was to specify this kind of homomorphism explicitly, greatest attention had to be paid to what is commonly called "*capture of variables*". Embeddings turn out to be far too weak to encompass this condition that *capturing of variables is forbidden.*

As binding and variable identity are primitive notions here, in contrast to the situation in λ-calculi, CRSs and HORSs [44, 31], where they are derived concepts, we have to be far more explicit about "variable capturing". When starting from embeddings, we have to introduce additional restrictions that mostly are concerned with the interplay between the morphisms and the inner nodes of the metavariable's image intervals. Therefore we first introduce a few more abbreviations:

Definition 6.1
For an explicit relation $\Phi : \mathcal{V}_1 \leftrightarrow (\mathcal{V}_2 + \mathcal{I}_2)$ between two graphs G_1 and G_2 we shall use the following abbreviations:

i) $\Phi_\cup : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ with

$$\Phi_\cup := (\overline{M_1 ; v_p ; \mathbb{T}} \sqcap \Phi ; [\mathbb{I} +\!\!+ \mathit{Inner}]) \sqcup (M_1 ; v_p ; \mathbb{T} \sqcap \Phi ; [\mathbb{I} +\!\!+ \mathit{RawInner}])$$

is the **image closure** of Φ — note that for place holder variables we use the raw view concerning inner nodes. (We shall not need place holders in this paper; they play a special rôle in the rewriting process — see 2.4. We make the distinctions just for completeness' sake.)

ii) $\Phi_{\Cup} : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ with $\Phi_{\Cup} := \Phi_{\cup} \sqcap \Phi ; \kappa^{\smile} ; \mathbb{T}$ is the restriction of the image closure to active metavariables, the **active image closure**.

iii) $\Phi_{\Diamond} : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ with $\Phi_{\Diamond} := \Phi ;[\mathbb{I} + \mathit{Skel}]$ is the **skeleton** of Φ.

iv) $\Phi_{\blacklozenge} : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ with $\Phi_{\Diamond} := \Phi ;[\mathbb{I} + \mathit{EncSkel}]$ is the **encapsulation skeleton** of Φ.□

Here it already becomes obvious why we chose to use explicit relations between two graphs rather than simple relations between the two node sets: In the latter case, when switching from functions to relations, the definition of image closure and skeletons would raise problems. Consider a metavariable that has multiple images, and at least one of the successors has images, too. Then, what we would want to understand as multiple image intervals of that metavariable might overlap, and an appropriate definition of the image closure would be impossible. At first, it would seem to be a sensible restriction to demand that the relation is univalent on metavariables with successors that have images; in this case, the definition from [20, 21] carries over without any problems.

However, in the general case this would be an unnatural restriction, and, as we shall see in the following, working with explicit relations does not complicate matters for functions, and we can use explicit relations to obtain a smooth transition to relations.

An additional source of complication is the fact that we do not want to restrict ourselves to acyclic graphs or to rooted graphs. Nevertheless we have been able to formulate a set of conditions that together prohibit variable capturing and that can be preserved by composition. An embedding fulfilling all these conditions is called a *fitting.*

Definition 6.2
A **fitting** is an appropriate embedding Φ from G_1 to G_2 where

- only *schemavariables* are active: $\mathbb{T} ; \kappa ; \Phi^{\smile} \sqsubseteq v_s ; M_1^{\smile}$,
- D_1 restricted to active metavariables, $\Phi ; \kappa^{\smile} ; \mathbb{T} \sqcap D_1$, is progressively finite (a sanity condition for ensuring what is called "consistency" in [20, 21]),

and Φ

- is coherent: for every active schemavariable its image is coherent, i.e.,

$$\mathbb{T} ; \Phi ; \kappa^{\smile} \sqsubseteq \mathit{Coherent}_2 \ ,$$

- (properties wrt. binding)

  strictly preserves binding: $\Phi_0^{\smile} ; B_1 = B_2^{\smile} ; \Phi_0^{\smile}$,
  is *conforming:* active schemavariables are mapped to segments:

$$\mathbb{T} ; \Phi ; \kappa^{\smile} \sqsubseteq \mathit{Segment}_2 \ ,$$

  respects encapsulation: Φ_0 does not identify a binder with encapsulated nodes:

$$\Phi_0 ; \Phi_0^{\smile} \sqsubseteq \overline{C_1} \ ,$$

  has *encapsulation unity:* if a node not encapsulated in G_1 has an image under $\Phi_{\triangle}$ that is encapsulated by another node, then the images of all nodes in G_1 are reachable from the latter (via nonempty paths):

$$\overline{\mathbb{T} ; C_1} ; \Phi_{\triangle} ; C_2^{\smile} \sqsubseteq \Phi_{\triangle} ; D_2^{+\smile} \ ,$$

is *encapsulation consistent:* for any two unencapsulated nodes in G_1, no variable in G_2 occurs free under one of their image nodes, but only bound under the other (while reachable over a non-selfencapsulating path):

$$(\overline{C_1^{\smile} ; \mathbb{T}} \sqcap \overline{\mathbb{T} ; C_1}) \sqcap \Phi_{\vartriangle} ; (\overline{\not{C}_2^{\smile}} \sqcap \Delta_2) ; \not{C}_2 ; \Phi_{\vartriangle}^{\smile} \;=\; \perp\!\!\!\perp \;,$$

controls bindings: the node binding the image of a free variable may not be contained in the image closure of F: $\Phi_0 ; B_2 \sqsubseteq B_1 ; \Phi_0 \sqcup \overline{\mathbb{T} ; \Phi_{\cup}}$,

- (properties wrt. variable identity)

  strictly preserves variables: $W_1 ; \Phi_0 = \Phi_0 ; W_2$,
  preserves the distinctness of equally bound variables: $\Phi_0 ; W_2 ; \Phi_0^{\smile} \sqcap B_1 ; B_1^{\smile} \sqsubseteq W_1$,
  respects free variables: $\Phi_0 ; W_2 ; \Phi_0^{\smile} \sqcap \overline{B_1 ; \mathbb{T}} \sqsubseteq W_1$ and
  controls variables: no node of a variable occurring free within the image of a schemavariable is the image of a bindable variable.

$$(\Phi_{\Cup} \sqcap \Phi_{\vartriangle} ; \not{C}_2^{\smile}) ; W_2 \sqsubseteq \overline{\mathbb{T} ; \Phi_0} \;.$$

If a fitting is a mapping, it is called **functional fitting**. □

The condition of encapsulation unity is necessary for avoiding that different sources of G_1 are "torn apart" by Φ, i.e., put into disjoint binding contexts. Encapsulation consistency serves essentially the same purpose as the restriction to spine-acyclic redexes in [1], but is more general and more fine-grained. In acyclic graphs it is a consequence of encapsulation unity.

So far we have not taken care of multiple instances of schemavariables. But we have already introduced the tool for comparing image segments in 4.4. The existence of a correspondence between the image segments of the nodes of a parallel schemavariable is a good basis for finally defining *homomorphisms*:

DEFINITION 6.3
A **homomorphism** is a fitting Φ, for which for every two segments in the target graph that are images of nodes contained in the same active schemavariable, i.e., for every two segments which stand in the relation $\kappa ; \Phi^{\smile} ; W_1 ; \Phi ; \kappa^{\smile}$, there is a correspondence between these two segments. □

At least under certain circumstances care has also to be taken to deal with non-univalent lower borders of intervals; these should usually be share-able. The appropriate concept from [20, 21] is that of "resolution", but we leave it aside for the time being since we shall not give the details of the rewriting step anyway.

6.2 Composition

Although we have seen in 2.4 that no category of fittings is needed for the kind of rewriting we have in mind, it is still interesting to see whether the homomorphism concept as defined above gives rise to a category, since that would be in support of our claim that the homomorphism conditions we stipulate are natural, i.e., give rise to an appropriate concept of "structure preserving mappings" of our term graphs with bound variables.

Finding a category of homomorphisms is far more complicated than for embeddings. First of all we need an additional condition to forbid the introduction of cycles in the target that had not been present in the source:

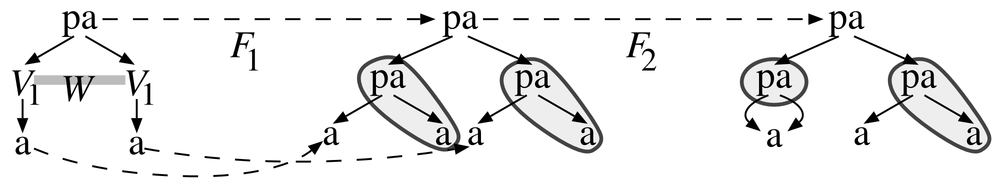
Definition 6.4
An explicit relation is called **acyclic**, iff $\Phi_{\Diamond} ; \Phi_{\cup}^{\smile} \sqcap D_1^{+} \sqsubseteq \mathbb{I}$. □

Now the problem is that acyclicity is necessary to be able to prove the fitting properties for the composition of two functional fittings (under the additional assumptions that either schemavariables are at most unary or all graphs are CRS-patterns — otherwise coherence is the first property to fail under composition), but this composition itself is not generally acyclic again.

So we need additional conditions to ascertain that, and we could either restrict ourselves to acyclic graphs or to functional fittings without schemavariables in the images of active schemavariables ("definitive" fittings, related to "ground substitutions").

According to this scenario, we obtain four kinds of categories of functional fittings; the proofs are essentially the same, but extremely involved and cover some 20 pages in [20].

All these fitting categories can be restricted to categories of homomorphisms, but without the restriction to CRS-patterns additional pretty strong conditions are necessary to prohibit situations like in the following picture, where the lower border of one schemavariable image has been identified into what should be its inner nodes, thus disrupting the existence of a correspondence:



(For a treatment that can cope with this kind of situation, in addition to the *node* relations that currently are used for homomorphisms, also *edge* relations would be necessary.)

One of the contributions of [21] is, however, to have shown that such a category of homomorphisms is not required for the rewriting mechanism, so that all these restrictions do not really hurt — see also 2.4. (A similar insight can be found in [4].)

Nevertheless, in being able to define a category of functional homomorphisms between term graphs with bound variables, avoiding the capture of bound variables and ensuring corresponding images for parallel schemavariables, we have shown that our homomorphisms indeed satisfactorily *preserve the structure* of our term graphs.

A problem that remains to be solved is how to find such homomorphisms, which can, in general, be pretty hard. We now turn to a special case that can be useful even in automated contexts.

6.3 Unique Encapsulation

The problem whether there is a relational homomorphism Φ of a term graph rewriting rule's left-hand side into a given application graph, limited to the case where the images of the rule sources are given, is decidable and finitary on finite graphs, i.e., it has mostly finitely many solutions.[5] However, in several contexts, such as automatic reduction, one would prefer a unitary matching problem, i.e. a matching problem with at most one solution.

[5] On finite DAGs [16] gives this result.

Under certain circumstances Φ is in fact determined by the images of the rule sources in the application graph. An example is the condition imposed on left-hand rule sides in CRSs [28], namely that all successors of every schemavariable are different bound variables.

The condition of *unique encapsulation*, a version of which is employed in HOPS, is strictly weaker than that. This condition is aimed at the fact that for uniquely determining the image of a schemavariable it suffices (in first approximation) to have a parallel schemavariable node encapsulated by some binding node, when the paths from the tip to that binder, from there to the parallel schemavariable node in question and from that node to the bound variable are already determined.

In earlier versions of HOPS, this is still relatively simple in view of the restriction to at most unary schemavariables; in its full generality we have to formulate a complex recursion over the path structure, and simple encapsulation is not sufficient for schemavariables with more than one successor.

We consider a graph G together with a vector $\beta : \mathbb{1} \leftrightarrow \mathcal{V}$ containing an encapsulation free basis of G. Let $\mathbb{Y} : \mathcal{V} \times \mathcal{E} \leftrightarrow \mathcal{V} \times \mathcal{E}^*$ be the relation between edges and concrete paths containing them, defined by

$$\mathbb{Y} := (Z^{\smile} \| \pi^{\smile} ; cons) ; \unlhd ; (\mathbb{I} \| conc) \quad .$$

When $s : \mathbb{1} \leftrightarrow \mathcal{V} \times \mathcal{E}$ is a vector of edges starting at schemavariables, i.e., $s ; \pi_{\mathcal{V}\times\mathcal{E}} \sqsubseteq v_s$, then let $\delta.s : \mathcal{V} \leftrightarrow \mathcal{V}$ be the reachability restricted to paths that do not lead over any edges starting in schemavariables except those contained in s:

$$\delta.s := (\overline{\overline{\mathbb{T} ; s \sqcup \overline{\mathbb{T} ; v_s ; M^{\smile}}} ; \mathbb{Y}} \sqcap \pi^{\smile}_{\mathcal{V}\times\mathcal{E}^*}) ; Z$$

We then define $\omega.s : \mathbb{1} \leftrightarrow \mathcal{V}\times\mathcal{E}$ as the extended vector of edges leaving schemavariables, that employs more encapsulations with the help of s; here the central idea informally proposed above is formalised:

$$\omega.s := s \sqcup \mathbb{T} ; \beta ; \delta.s ; \Big((\delta.s \sqcap \mathbb{T} ; v_s ; M^{\smile}) ; \pi^{\smile}_{\mathcal{V}\times\mathcal{E}} \sqcap B^{\smile} ; (\delta.s)^{\smile} ; S^{\smile} \Big) ; (W \| \mathbb{I})$$

With

$$\Omega := \bigsqcup \{ j : \mathbb{N} \bullet \omega^j . \perp\!\!\!\perp \}$$

we now obtain the vector of all edges starting in schemavariables that are uniquely determined by encapsulations in G. We call G **uniquely encapsulated** *from* β iff Ω comprises all edges starting in schemavariables, i.e., iff

$$\Omega = \mathbb{T} ; v_s ; \{\!| M^{\smile}, E |\!\} \quad .$$

Now the following result can be obtained:

PROPOSITION 6.5

Under unique encapsulation of a rule's left-hand side (from its source vector β) the matching problem becomes unitary thanks to the correspondences between the images of parallel schemavariables, at least if the left-hand side only allows matchings with sharp resolution, i.e., no two successors of one schemavariable are unifiable, and if the bound variables can be uniquely determined, i.e., $B ; B^{\smile} \sqcap M ; M^{\smile} \sqsubseteq W$ holds. $\square$

Returning to the transition from simple node set relations to explicit relations (see 5.2), we can put unique encapsulation to another use. A special case also guaranteeing the validity of $<\Phi_\triangle>_{\top;\kappa;\Phi^\smile} = \Phi$ for general Φ is *unique encapsulation* of G_1.

If G_1 is uniquely encapsulated, then a refined definition of $\Phi_\Box$ (Def. 5.10) could make use of that fact in order to prevent unwanted interference, and only this way it would be possible to ensure (again via additional restrictions just as in the case of functions) that the composition of two relational homomorphisms is again a relational homomorphism.

7 Conclusion

This paper has two main aspects: On one hand the contribution to the theory of term graphs and term graph rewriting, on the other hand a case study on application of relational calculus.

In the first aspect we heavily draw on the foundations laid in [20, 21], and extend our work there towards substituting relations for functions in the central rôles.

A first outlook on another use of our formalisation of second order term graphs has been presented in [19]; there we use them for typing of functional programming languages with *polytypic programming* features.

A point to note is that although the explicit treatment of lower interval borders allows for more differentiations than the traditional node functions, this is not yet the full generality we would like to achieve. The ultimate freedom for intervals is reached when lower borders relate edge labels to *sets of edges ending in the same node* instead of just these nodes as in this paper — we postponed this move for the sake of the brevity of presentation.

One important direction for further research would be to extend the approach to be able to tackle the term graph representation systems used for optimal reduction in the λ-calculus such as [39, 41, 29, 24, 15, 2].

In the second aspect of this paper we stress the usefulness of relational notation and calculus — we strongly feel that the results in [20, 21] and in this paper would not have been achieved without.

Although some concepts require a second thought for understanding, the relational formalisation in general still can serve as a support to understanding through its preciseness and high conciseness. A few times, even far more verbose explanation would probably not easily succeed to make the exposed concepts as clear and as usable in proofs as the relational expression.

Furthermore, wherever we succeeded to provide short formalisations, even if the underlying definitions are themselves more complicated, concise and calculational proofs are enabled, as can be seen in quite a few places here and in [20]. Sometimes, however, the conciseness of the definitions cannot hide the fact that the subject matter is complicated, with accordingly lengthy proofs.

But we also show some limitations; more complex subject matters such as in 5.3 can lead to expressions that are difficult to create, even more difficult to transform and manipulate, and that cloud the concepts involved through the heavy use of tuple manipulation combinators.

The problems can however be seen as mostly a feature of the linearity of the notation; a graphical notation such as that of [22] would help immensely — we have seen an example graph after Def. 5.6. One task for the future will definitely be to help this kind of graphical calculus to find acceptance and to create tools for it; we are certain that the use of relational methods could gain considerable impetus from the presence of supporting tools for graphical exposition and proofs and from general acceptance of presentation of relational expressions as graphs instead of as linear terms.

Summarising, we have employed the calculus of relations as a useful tool for a difficult application, and we usefully generalised that application by supplanting relations for functions, all in all making a strong case for using relations as the basis for formal argument.

References

[1] Zena M. Ariola and Jan Willem Klop. Cyclic lambda graph rewriting. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 416–425, Paris, France, 4–7 July 1994. IEEE Computer Society Press.

[2] Andrea Asperti and Cosimo Laneve. Interaction Systems II, The Practice of Optimal Reductions. Technical Report UBLCS-93-12, Laboratory for Computer Science, University of Bologna, 1993.

[3] Roland C. Backhouse, Peter J. de Bruin, Grant Malcom, Ed Voermans, and Jaap van der Woude. Relational catamorphisms. In Bernd Möller, editor, *Constructing Programs From Specifications, Proceedings of the IFIP TC2 Working Conference on Constructing Programs From Specifications*, pages 319–371. North-Holland, 1991.

[4] Richard Banach. Term graph rewriting and garbage collection using opfibrations. *Theoretical Computer Science*, 131(1):29–94, August 1994.

[5] Arne Bayer and Wolfram Kahl. The **H**igher-**O**bject **P**rogramming **S**ystem "HOPS". In Bettina Buth and Rudolf Berghammer, editors, *Systems for Computer-Aided Specification, Development and Verification*, Bericht Nr. 9416, pages 154–171. Universität Kiel, 1994.

[6] Rudolf Berghammer, Armando M. Haeberer, Gunther Schmidt, and Paulo A. S. Veloso. Comparing two different approaches to products in abstract relation algebra. In G. Scollo, editor, *Proc. 3rd Int. Conf. Algebraic Methodology and Software Technology, Enschede, June 21 – June 25*, Berlin-Heidelberg-New York-Tokyo, 1993. Springer.

[7] Rudolf Berghammer, Gunther Schmidt, and Hans Zierer. Symmetric quotients and domain constructions. *Information Processing Letters*, 33(3):163–168, 1989.

[8] Nicolas Bourbaki. *Éléments de Mathématique, Livre 1: Théorie des Ensembles*. Hermann, 1970.

[9] Chris Brink, Wolfram Kahl, and Gunther Schmidt, editors. *Relational Methods in Computer Science*. Advances in Computing. Springer-Verlag, Wien, New York, 1997. ISBN 3-211-82971-7.

[10] Volker Claus, Hartmut Ehrig, and Grzegorz Rozenberg, editors. *Graph-Grammars and Their Application to Computer Science and Biology, International Workshop*, volume 73 of *Lecture Notes in Computer Science*, Bad Honnef, November 1978. Springer-Verlag.

[11] Andrea Corradini and Frank Drewes. (Cyclic) term graph rewriting is adequate for rational parallel term rewriting. Draft, 1995.

[12] Nachum Dershowitz and Jea-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier Science Publishers B. V., 1990.

[13] Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors. *Graph-Grammars and Their Application to Computer Science, 4th International Workshop*, volume 532 of *Lecture Notes in Computer Science*, Bremen, Germany, March 1990. Springer-Verlag.

[14] Hartmut Ehrig, M. Pfender, and H. J. Schneider. Graph grammars: An algebraic approach. In *Proc. IEEE Conf. SWAT'73*, pages 167–180, Iowa City, 1973.

[15] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *20th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 15–26, Charleston, South Carolina, January 1993. acm press.

[16] G. P. Huet and B. Lang. Proving and applying program transformations expressed with second-order patterns. *Acta Informatica*, 11:31–55, 1978.

[17] Paola Inverardi and Marisa Venturini Zilli. Rational rewriting. In Igor Prívara, Branislav Rovan, and Peter Ružička, editors, *Mathematical Foundations of Computer Science 1994, 19th Internatioanl Symposium, MFCS '94, Košice, Slovakia, August 1994, Proceedings*, volume 841 of *LNCS*, pages 433–442. Springer-Verlag, 1994.

[18] Wolfram Kahl. Can functional programming be liberated from the applicative style? In Bjørn Pehrson and Imre Simon, editors, *Technology and Foundations, Information Processing '94, Proceedings of the IFIP 13th World Computer Congress, Hamburg, Germany, 28 August – 2 September 1994, Volume I*, volume A-51 of *IFIP Transactions*, pages 330–335. IFIP, North-Holland, 1994.

[19] Wolfram Kahl. Aspects of typed term graphs. In Tiziana Margaria, editor, *Kolloquium Programmiersprachen und Grundlagen der Programmierung, Adalbert Stifter Haus, Alt Reichenau, 11.–13. Oktober 1995*, Bericht MIP-9519, pages 104–109. Universität Passau, Fakultät für Mathematik und Informatik, December 1995. URL: http://inf2-www.informatik.unibw-muenchen.de/kahl/papers/typed.ps.gz.

[20] Wolfram Kahl. Kategorien von Termgraphen mit gebundenen Variablen. Technischer Bericht 9503, Fakultät für Informatik, Universität der Bundeswehr München, September 1995. 191 pages.

[21] Wolfram Kahl. *Algebraische Termgraphersetzung mit gebundenen Variablen*. Reihe Informatik. Herbert Utz Verlag Wissenschaft, München, 1996. ISBN 3-931327-60-4; also doctoral dissertation at Fakultät für Informatik, Universität der Bundeswehr München.

[22] Wolfram Kahl. Algebraic graph derivations for graphical calculi. In Fabrizio d'Amore, Paola G. Franciosa, and Alberto Marchetti-Spaccamela, editors, *Graph Theoretic Concepts in Computer Science, 22nd International Workshop, WG '96, Caddenabbia, Italy, June 1996, Proceedings*, volume 1197 of *LNCS*, pages 224–238. Springer-Verlag, 1997.

[23] Wolfram Kahl. A fibred approach to rewriting — how the duality between adding and deleting cooperates with the difference between matching and rewriting. Technical Report 9702, Fakultät für Informatik, Universität der Bundeswehr München, D-85577 Neubiberg, May 1997.

[24] Vinod Kathail. *Optimal Interpreters for Lambda-Calculus Based Functional Languages*. PhD thesis, MIT, May 1990.

[25] J.R. Kennaway, J.W. Klop, M.R. Sleep, and F.J. de Vries. *The Adequacy of Term Graph Rewriting for Simulating Term Rewriting*, chapter 12, pages 157–170. In Sleep et al. [37], 1993.

[26] Richard Kennaway. On "On Graph Rewritings". *Theoretical Computer Science*, 52:37–58, 1987.

[27] Richard Kennaway. Graph rewriting in some categories of partial morphisms. In Ehrig et al. [13], pages 490–504.

[28] Jan Willem Klop. Combinatory reduction systems. Mathematical Centre Tracts 127, Centre for Mathematics and Computer Science, Amsterdam, 1980. PhD thesis.

[29] John Lamping. An algorithm for optimal lambda calculus reduction. In *17th Annual ACM Symposium on Principles of Programming Languages*, pages 16–30, San Francisco, California, January 1990. acm press.

[30] Michael Löwe. Algebraic approach to graph transformation based on single pushout derivations. Technical Report 90/05, TU Berlin, 1990.

[31] Tobias Nipkow. Higher-order critical pairs. In *Logics in Computer Science*, pages 342–349, 1991.

[32] Peter Padawitz. Graph grammars and operational semantics. *Theoretical Computer Science*, 19(1):117–141, 1982.

[33] Detlef Plump. *Evaluation of Functional Expressions by Hypergraph Rewriting*. Dissertation, Universität Bremen, 1993.

[34] J. C. Raoult. On graph rewritings. *Theoretical Computer Science*, 32:1–24, 1984.

[35] *Rewriting Techniques and Applications: 4th international conference, RTA 4*, volume 488 of *LNCS*, Berlin, 1991. Springer.

[36] Gunther Schmidt and Thomas Ströhlein. *Relations and Graphs, Discrete Mathematics for Computer Scientists*. EATCS-Monographs on Theoretical Computer Science. Springer Verlag, 1993.

[37] M.R. Sleep, M.J. Plasmeijer, and M.C.J.D. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice.* John Wiley, 1993.

[38] J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice Hall International Series in Computer Science. Prentice Hall, 1989.

[39] John Staples. A graph-like lambda calculus for which leftmost-outermost reduction is optimal. In Claus et al. [10], pages 440–455.

[40] John Staples. Computation on graph-like expressions. *Theoretical Computer Science*, 10:171–185, 1980.

[41] John Staples. Optimal evaluations of graph-like expressions. *Theoretical Computer Science*, 10:297–316, 1980.

[42] Christopher Peter Wadsworth. *Semantics and Pragmatics of the Lambda Calculus.* D.Phil. thesis, Oxford University, September 1971.

[43] D. A. Wolfram. Rewriting, and equational unification: the higher-order cases. In RTA '91 [35], pages 25–36.

[44] D. A. Wolfram. Rewriting, and equational unification: the higher-order cases. In *Rewriting Techniques and Applications: 4th international conference, RTA 4*, volume 488 of *LNCS*, pages 25–36, Berlin, 1991. Springer.

[45] Hans Zierer. *Programmierung mit Funktionsobjekten: Konstruktive Erzeugung semantischer Bereiche und Anwendung auf die partielle Auswertung.* Dissertation, Technische Universität München, Fakultät für Informatik, 1988. Report TUM-I8803.

[46] Hans Zierer. Relation-algebraic domain constructions. *Theoretical Computer Science*, 87:163–188, 1991.

[47] Hans Zierer, Gunther Schmidt, and Rudolf Berghammer. An interactive graphical manipulation system for higher objects based on relational algebra. In Gottfried Tinhofer and Gunther Schmidt, editors, *Proc. 12th International Workshop on Graph-Theoretic Concepts in Computer Science*, LNCS 246, pages 68–81, Bernried, Starnberger See, June 1986. Springer-Verlag.

Received 15th February 1996