On Graph Reasoning

Renata de Freitas
Institute of Mathematics, UFF, Niterói, Brazil

Paulo A. S. Veloso
Systems and Computer Engineering Program, COPPE/UFRJ, Rio de Janeiro, Brazil

Sheila R. M. Veloso
Systems and Computer Engineering Department, UERJ, Rio de Janeiro, Brazil

Petrucio Viana*
Institute of Mathematics, UFF, Niterói, Brazil

Abstract

In this paper we study the (positive) graph relational calculus. The basis for this calculus was introduced by S. Curtis and G. Lowe in 1996 and some variants, motivated by their applications to semantics of programs and foundations of mathematics, appear scattered in the literature. No proper treatment of these ideas as a logical system seems to have been presented. Here, we give a formal presentation of the system, with precise formulation of syntax, semantics, and derivation rules. We show that the set of rules is sound and complete for the valid inclusions, and prove a finite model result as well as decidability. We also prove that the graph relational language has the same expressive power as a first-order positive fragment (both languages define the same binary relations), so our calculus may be regarded as a notational variant of the positive existential first-order logic of binary relations. The graph calculus, however, has a playful aspect, with rules easy to grasp


*corresponding author

Email address: petrucio@cos.ufrj.br (Petrucio Viana)
and use. This opens a wide range of applications which we illustrate by applying our calculus to the positive relational calculus (whose set of valid inclusions is not finitely axiomatizable), obtaining an algorithm for deciding the valid inclusions and equalities of the latter.

**Key words:** Completeness, Decidability, Expressive power, Graph calculus, Relational Calculus

1. Introduction

We present \( +\mathcal{RG} \), the (positive) relational calculus with graphs. The basis for the graph relational calculus was introduced by S. Curtis and G. Lowe [6]. They exemplified its strong expressive power, claimed soundness of their inference rules and left completeness as an open problem. Although Curtis and Lowe give motivation and examples in [6], no proper treatment of these ideas as a logical system seems to have been presented.

The main issues addressed in this paper concern a formal presentation of the logical system \( +\mathcal{RG} \), i.e. a set of rules to derive graphs that is sound and complete with respect to the valid inclusions between graphs, and a characterization of the graph relational language compared to a first-order positive fragment in the sense that both languages define the same binary relations.

Our formulation of the graph calculus leads to the following improvements on some former treatments of relations by means of graphs: a proper treatment of the union operator by the introduction of the notion of a slice of a graph; a more elaborated definition of homomorphism enabling both precise formulation and use of the homomorphism rule in proofs; a set of rules equivalent to the homomorphism rule providing a better understanding of it; a normal form for proofs resembling the familiar one in classical propositional logic; an analysis establishing the precise relationship among the positive relational calculus, the graph calculus and a positive fragment of the first-order language of binary relations. Despite being a notational variant of the latter, our graph calculus has a playful aspect, with rules easier to grasp and to use. Also, in contrast to the algebraic approach to relations, whose elements are relational terms, the graph approach deals with relational terms and points. This leads to a pictorial and smoother environment for relational reasoning. Such an approach opens a wide range of applications and provides contributions to the areas of algebraic logic, algebraic semantics, theoretical
computer science, and model theory. It also has some important practical consequences since it deals with relational formalisms that are widely applicable, cf. [6, 4, 13] and Section 1.1. We illustrate this aspect by using $+RG$ to prove the valid inclusions and equalities of the positive relational calculus, a (non-finitely axiomatizable) decidable fragment of the Tarski relational calculus [22].

The paper is structured as follows. In Section 2, we present the syntax and semantics of the relational formalism based on graphs. In Section 3, we provide a set of rules to transform a graph into another one and prove that it is sound and complete with respect to the valid inclusions between graphs. In Section 4, we characterize the expressive power of the graph language in terms of the first-order language of binary relations. In Section 5, we apply the graph calculus to the positive fragment of the relational formalism presented in [21, 20], proving the decidability of this system. Finally, in Section 6, we conclude with some specific themes and questions.

1.1. Related work

Some variants of the graph calculus, motivated by applications to semantics of programs and foundations of mathematics, appear scattered in the literature. In particular, D. Cantone et al. [5, 8] deal with some questions about expressive power. They present two techniques designed for translating existentially quantified conjunctions of dyadic literals, i.e. graphs in our sense, into relational terms containing only occurrences of composition, intersection and converse. The use of such techniques is illustrated by examples and their algorithmic complexity is studied. D. Dougherty and C. Gutiérrez [7, 12] consider the equational theory of binary relations in a language containing only composition, converse, and the lattice operations. They define a presentation of relation expressions as graphs in a manner very similar to ours. Using this, they are able to define a notion of reduction which is confluent and strongly normalizing and induces a notion of computable normal form for terms. P.J. Freyd and A. Scedrov [11] define a fragment of the graph calculus and apply it to allegories. They use the representation of relations as graphs to prove some non-finite axiomatizability results. C. Brown and G. Hutton [4, 13] present an approach for the introduction of projections and parallelism into the graph calculus. They use their calculus to specify programs formally. In particular, they derive deterministic programs from specifications by means of formal manipulations of non-deterministic programs.
2. Syntax and semantics

The graph relational language uses familiar relational concepts. Its construction is based on the positive relational language, $+\text{RC}$, whose basic syntactical and semantical concepts are essentially those of [21, 22, 20] without complementation and empty relation.

The terms of $+\text{RC}$, typically denoted by $R, S, T$, are generated from the set of relational variables $\text{RVAR} = \{r_i : i \in \omega\}$ by applying the relational operators $E, I, T, \cap, \cup, \circ$, according to the following grammar:

$$R ::= r_i \mid E \mid I \mid R^T \mid R \cap S \mid R \cup S \mid R \circ S.$$  

The meaning $[R]_M$ of a term $R$ in a model $M$ are defined as in the relational case, excluding references to the empty relation and to complementation. Formally, a model $M$ is a structure $M = (M, r^M_i)_{i \in \omega}$, where $M \neq \emptyset$ and $r^M_i \subseteq M \times M$. Given a model $M$, symbols $E$ and $I$ are interpreted, respectively, as the relations $M \times M$ and $\{(a,b) \in M \times M : a = b\}$; symbols $T, \cap, \cup, \circ$ as the conversion, intersection, union, and composition of relations, respectively.

Now, we present a relational language $+\text{RG}$, based on graphs. $+\text{RG}$ is designed to represent relations using graphs of a special kind. Its language has two kinds of expressions, slices and graphs. Slices are (directed pseudo multi) graphs having a distinguished pair of nodes and arcs labeled by terms of $+\text{RC}$, and a graph consists of slices.

Formally, we fix a set $\text{INOD} = \{x_n : n \in \omega\}$ of individual nodes, typically denoted by $x, y, z, u, v, w$. A slice is a structure $S = (N, A, x, y)$, where $N$ is a finite non-empty set of nodes, $A \subseteq N \times \Sigma^+ \times N$ is a finite set of labeled arcs ($\Sigma^+$ is the set of all $+\text{RC}$ terms), $x, y$ are, not necessarily distinct, distinguished nodes in $N$. The pair $(x, y)$ is called the distinguished pair of $S$. Given a term $R$ of $+\text{RC}$ and nodes $u, v$, the arc $(u, R, v)$ is denoted by $uRv$ and nodes $u$ and $v$ are called the end points of $uRv$. A positive relational graph, or simply a graph, is a finite non-empty set of slices, which may share nodes. We identify a slice and a graph having only this slice. Figure 1 shows three one-slice graphs. In figures, we indicate the first node of the distinguished pair of a slice by $-$ and the second one by $+$. Figure 2 shows a two-slice graph. The $+\text{RG}$ inclusions and equalities are expressions of the forms $G \sqsubseteq H$ and $G \equiv H$, respectively.

Now we present the semantics of $+\text{RG}$. Given a set, considered as universe, a graph defines a binary relation on it, according to some conditions
on its slices. The label of an arc represents a constraint associated to the relation defined by the label. A path from a node to another one represents a constraint associated to the composition of the corresponding relations. Two paths with the same start and end points and sharing no nodes represent a constraint associated to the intersection of the corresponding relations. Each graph represents a constraint associated to the union of the relations corresponding to its slices.

Formally, consider a slice \( S = (N, A, x, y) \) and a model \( \mathfrak{M} = (M, (r_i^{\mathfrak{M}}))_{i \in \omega} \). An assignment for \( S \) in \( \mathfrak{M} \) is a function \( g : N \to M \) such that \( (gu, gv) \in [R]^{\mathfrak{M}} \) whenever \( uRv \in A \). The meaning of \( S \) in \( \mathfrak{M} \) is the set \( [S]^{\mathfrak{M}} = \{(gx, gy) \in M \times M : g \text{ is an assignment for } S \text{ in } \mathfrak{M}\} \). The meaning of a graph \( G \), denoted by \( [G]^{\mathfrak{M}} \), is the union of the meanings of its slices.

We define general notions of inclusion and equivalence for graphs according to the relations they represent as follows. Let \( G \) and \( H \) be graphs of \( +\mathcal{R}G \). We say that \( G \) is included in \( H \) or that the graph inclusion \( G \subseteq H \) is valid, denoted by \( \models G \subseteq H \), when \( [G]^{\mathfrak{M}} \subseteq [H]^{\mathfrak{M}} \), for every model \( \mathfrak{M} \); we say that \( G \) and \( H \) are equivalent or that the graph equality \( G \equiv H \) is valid, denoted by \( \models G \equiv H \), when \( [G]^{\mathfrak{M}} = [H]^{\mathfrak{M}} \), for every model \( \mathfrak{M} \). Obviously, \( G \) and \( H \) are equivalent iff they include each other, i.e. \( G \) is included in \( H \).
3. Graph relational calculus

We shall now present a graph relational calculus, i.e. a set of transformation rules for deriving a relational graph from another. Each rule involves the application of a local transformation, leaving the rest of the graph untouched.

The main idea behind the choice of the rules is to define a normal form for the graph language and use it to establish that \( G \sqsubseteq H \) is valid by executing the following two major steps. First, reduce the graphs \( G \) and \( H \) to their simple normal forms \( \text{SNF}_G \) and \( \text{SNF}_H \), respectively. This is accomplished by using the Elimination/Introduction rules (Table 1). Second, verify whether or not \( \text{SNF}_H \) can be obtained from \( \text{SNF}_G \) by structural transformations (Tables 2 and 3). We will also show (in Section 3.2) that the structural transformations are equivalent to just one rule: the Graph Cover rule \( \text{GCv} \) (Table 2).

As we will see (in Section 3.3), to obtain the completeness result we just need to decide whether or not one can apply one single instance of the Graph Cover rule on \( \text{SNF}_G \) to obtain \( \text{SNF}_H \), this will settle whether \( \models G \sqsubseteq H \).

3.1. Graph relational rules

To state the graph rules, we use the node substitution notation \( \frac{v}{u} \) for replacing \( v \) by \( u \), which we extend naturally to pairs and triples as well as sets, e.g. for a set \( A \) of arcs, we put \( A \frac{v}{u} := \{ a \frac{v}{u} R b \frac{v}{u} : aRb \in A \} \).

The operational rules in Table 1 and the structural rules \( \text{Spl} \) and \( \text{ErN} \) in Table 3 can be applied in both directions. Each one of these rules is an abbreviation for two rules: downward and upward. The structural rules \( \text{Spl}^{-} \), \( \text{Spl}^{+} \), \( \text{Spl}^{--} \), \( \text{ErA} \), and \( \text{AdS} \) can be applied only in the downward direction. The rules in Table 1 allow the elimination (downwards) and the introduction (upwards) of the operators. Each rule involves the application of the local transformation specified in the rule, leaving the rest of the graph untouched. The meaning of the graph is to remain unchanged. We will explain each rule in the downward direction. Soundness of each rule will follow from these explanations.

The operational rules are given in Table 1. Rule \( \text{Unv} \) allows one to erase an arc labeled by \( E \) from a slice (see Figure 3). Rule \( \text{Idn} \) allows one to erase an arc \( uRv \) and a node \( u \) from a slice, renaming nodes and redirecting arcs accordingly (Figure 4). Rule \( \text{Cnv} \) allows one to replace an arc \( uR^Tv \) by arc
Table 1: Elimination/Introduction rules for transforming graphs

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unv</strong></td>
<td>$G \cup {(N, A \cup {uEv}, x, y)}$ [ $G \cup {(N, A, x, y)}$</td>
</tr>
<tr>
<td><strong>Idn</strong></td>
<td>$G \cup {(N, A \cup {ulv}, x, y)}$ [ $G \cup {(N^u_n, A^v_n, x^u_n, y^v_n)}$</td>
</tr>
<tr>
<td><strong>Cnv</strong></td>
<td>$G \cup {(N, A \cup {uR^T v}, x, y)}$ [ $G \cup {(N, A \cup {vRu}, x, y)}$</td>
</tr>
<tr>
<td><strong>Int</strong></td>
<td>$G \cup {(N, A \cup {uR \cap Sv}, x, y)}$ [ $G \cup {(N, A \cup {uRv, uSv}, x, y)}$</td>
</tr>
<tr>
<td><strong>Cmp</strong></td>
<td>$G \cup {(N, A \cup {uR \circ Sv}, x, y)}$ [ $G \cup {(N \cup {w}, A \cup {uRw, wSv}, x, y)}$ if $w \notin N$</td>
</tr>
<tr>
<td><strong>Uni</strong></td>
<td>$G \cup {(N, A \cup {uR \sqcup Sv}, x, y)}$ [ $G \cup {(N, A \cup {uRv}, x, y), (N, A \cup {uSv}, x, y)}$</td>
</tr>
</tbody>
</table>
\( vRu \) in a slice. Rule \textit{Int} allows one to replace an arc \( uR \sqcap Sv \) by arcs \( uRv \) and \( uSv \) in a slice. Rule \textit{Cmp} allows one to replace in a slice an arc \( uR \circ Sv \) by arcs \( uRw \) and \( wSv \), with a new node \( w \). Rule \textit{Uni} allows one to replace in a graph a slice \( T \) having an arc \( uR \sqcup Sv \) by two other slices \( T_R \) and \( T_S \), obtained from \( T \) by replacing the arc \( uR \sqcup Sv \) by arcs \( uRv \) and \( uSv \), respectively.

For instance, consider graphs \( G_1 \) and \( G_2 \) (Figure 1 in Section 2); graph \( G_2 \) is obtained from graph \( G_1 \) by downward applications of \textit{Int} and \textit{Cmp}. Also, graph \( G_5 \) (Figure 5) is obtained from \( G_4 \) (Figure 2 in Section 2) by upward applications of \textit{Unv}, \textit{Cmp} and \textit{Uni}.

Some comments on the upward versions of \textit{Idn} and \textit{Cmp} may be useful. Rule \textit{Idn}, in the upward direction, allows one to transform a graph \( G \cup \{S\} \) into a graph \( G \cup \{(N,A \cup \{uRv\},x,y)\} \), provided that \( S \) is obtained from \( (N,A,x,y) \) by means of the substitution operation, replacing \( v \) by \( u \). Notice that there is an implicit nondeterminism in the upward version of \textit{Idn}, since

\[
\begin{array}{c}
- \quad x \\
\text{(r \circ E)} \sqcup s
\end{array}
\]

\[
\begin{array}{c}
- x \\
\quad y
\end{array}
\]
there may be distinct slices identified by the substitution operation. Rule \( \text{Cmp} \), in the upward direction, allows one to transform a graph \( G \cup \{(N \cup \{w\}, A \cup \{uRw, wSv\}, x, y)\} \) into a graph \( G \cup \{(N, A \cup \{uR \circ Sv\}, x, y)\} \), provided that \( w \not\in N \). Notice that, since \( (N, A \cup \{uR \circ Sv\}, x, y) \) is a slice, \( w \) can not be an end point of an arc in \( A \) when \( w \not\in N \).

When applying an Elimination/Introduction rule, one may choose between erasing or not the arc that determines the application of the rule. Figure 6 illustrates such an upward application of rule \( \text{Cnv} \) that does not erase the determining arc.

We now introduce the concepts of homomorphism and of graph cover. Given slices \( S = (N, A, x, y) \) and \( T = (N', A', x', y') \), by a homomorphism \( \phi : T \to S \) we mean a function \( \phi : N' \to N \) preserving distinguished nodes and labels at arcs, i.e. such that \( \phi x' = x \), \( \phi y' = y \), and \( \phi uR \phi v \in A \), for all \( uRv \in A' \). We say that \( T \) covers \( S \) or \( S \) is covered by \( T \), denoted by \( S \leftarrow T \), iff there exists a homomorphism \( \phi : T \to S \). Now, given graphs \( G \) and \( H \), we say that \( H \) covers \( G \) or \( G \) is covered by \( H \), denoted by \( G \leftarrow H \), iff for each slice \( S \) of \( G \) there exists a slice \( T_S \) in \( H \) such that \( T_S \) covers \( S \).

**Example 1.** Consider graphs \( G_2 \) and \( G_4 \) (Figures 1 and 2, respectively, in Section 2). Function \( \phi \), given by \( \phi x = x \) and \( \phi y = y \), is a homomorphism from the left slice of \( G_4 \) to the slice of \( G_2 \). Thus \( G_2 \leftarrow G_4 \).

The downward Graph Cover rule \( \text{GCv} \) allows one to replace a graph by a graph covering it. Soundness of the Graph Cover rule (if \( G \leftarrow H \), then \( \models G \subseteq H \)) follows from the fact that homomorphisms transfer assignments by composition. Indeed, given a model \( \mathfrak{M} \), consider \( (a, b) \in [G]_{\mathfrak{M}} \). Then \( (a, b) \in [S]_{\mathfrak{M}} \), for some slice \( S = (N, A, x, y) \) of \( G \), i.e. there is an assignment \( g \) for \( S \) in \( \mathfrak{M} \) such that \( gx = a \) and \( gy = b \). Since \( H \) covers \( G \), there exist a slice \( T = (N', A', x', y') \) of \( H \) and a homomorphism \( \theta : T \to S \). This induces
Table 2: Graph Cover rule.

\[
\text{GCv } \frac{G}{H} \quad \text{if } G \leftarrow H
\]

the composite assignment \( g\theta \) for \( T \) in \( \mathcal{M} \). So \( g\theta x' = a \) and \( b = g\theta y' \), whence \((a, b) \in [T]_\mathcal{M} \subseteq [H]_\mathcal{M}\).

The structural rules are given in Table 3. Rules \( \text{Spl} \) (in the downward direction) \( \text{Spl}^- \), \( \text{Spl}^+ \), and \( \text{Spl}^{--} \) allow one to add inside a slice a new node \( u' \) having adjacent to it the same arcs that are adjacent to a node \( u \) of the slice. In \( \text{Spl} \) the resulting slice has the same distinguished nodes as the original slice. In \( \text{Spl}^- \) and \( \text{Spl}^+ \), the new node \( u' \) is one of the distinguished nodes of the resulting slice. The soundness of \( \text{Spl}^- \) and \( \text{Spl}^+ \) does not require the distinguished nodes of the original slice to be distinct. In fact, we need this particular case to obtain a complete system. In \( \text{Spl}^{--} \), the resulting slice has distinguished pair \((u', u')\). As remarked before, rules \( \text{Spl}^- \), \( \text{Spl}^+ \), and \( \text{Spl}^{--} \) can only be applied downwards. These rules use a function denoted by \( \text{splt}_{u'}^u \), split \( u \) with \( u' \), transforming sets of arcs, defined by

\[
\text{splt}_{u'}^u A = A \cup \{u'Rw : uRw \in A\} \cup \{wRu' : wRu \in A\} \cup \{u'Ru', uRu', u'Ru : uRu \in A\}.
\]

Rule \( \text{Spl} \), in the upward direction, allows one to transform a graph \( G \cup \{S\} \) into a graph \( G \cup \{(N,A,x,y)\} \), provided that \( S \) is obtained from \((N,A,x,y)\) by means of a \( \text{splt}_{u'}^u \) operation, for \( u' \not\in N \). Notice that, since \((N,A,x,y)\) is a slice, \( u' \) can not be an end point of an arc in \( A \) when \( u' \not\in N \).

Graph \( G_6 \) (Figure 7) is obtained from graph \( G_2 \) (Figure 1) by a downward application of \( \text{Spl} \) (using \( \text{splt}_{y}^v \)). Figure 8 illustrates an application of \( \text{Spl}^- \).

Rule \( \text{ErN} \) allows erasing an isolated node that is not distinguished in a slice. A node is isolated when it is not linked to another node by an arc. Rule \( \text{ErA} \) allows erasing an arc in a slice. Rule \( \text{AdS} \) allows addition of slices to a graph. As remarked before, rules \( \text{ErA} \) and \( \text{AdS} \) can only be applied downwards.

3.2. Graph relational derivations

The notion of derivation is defined as usual. Since each one of our single-conclusion rules has a single premise, our derivations are linear. Hence, our
**Table 3: Structural Rules.**

### Two-way Structural Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transformation</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spl</td>
<td>( G \cup {(N, A, x, y)} ) to ( G \cup {(N \cup {u}, \text{split}_u A, x, y)} )</td>
<td>if ( u \not\in N )</td>
</tr>
<tr>
<td>ErN</td>
<td>( G \cup {(N, A, x, y)} ) to ( G \cup {(N - {u}, A, x, y)} )</td>
<td>if ( u ) is isolated and ( u \not\in {x, y} )</td>
</tr>
</tbody>
</table>

### One-way Structural Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transformation</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spl(^-)</td>
<td>( G \cup {(N, A, x, y)} ) to ( G \cup {(N \cup {u}, \text{split}_u A, u', y)} )</td>
<td>if ( u' \not\in N )</td>
</tr>
<tr>
<td>Spl(^+)</td>
<td>( G \cup {(N, A, x, y)} ) to ( G \cup {(N \cup {u}, \text{split}_u A, x, u')} )</td>
<td>if ( u' \not\in N )</td>
</tr>
<tr>
<td>Spl(^--)</td>
<td>( G \cup {(N, A, x, x)} ) to ( G \cup {(N \cup {u}, \text{split}_u A, u', u')} )</td>
<td>if ( u' \not\in N )</td>
</tr>
<tr>
<td>ErA</td>
<td>( G \cup {(N, A, x, y)} ) to ( G \cup {(N - {uRv}, A, x, y)} )</td>
<td></td>
</tr>
<tr>
<td>AdS</td>
<td>( G \cup {S} ) to ( G \cup {S} \cup {S'} )</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 7: Graph \( G_6 \).**
system contrasts with the majority of derivation systems which provides tree-
like derivations.

A graph derivation is a sequence $G_0, R_1, G_1, \ldots, G_{i-1}, R_i, G_i, R_{i+1}, \ldots, R_n, G_n$, also displayed as $G_0 \xrightarrow{R_1} G_1 \xrightarrow{R_2} \cdots \xrightarrow{R_{i-1}} G_{i-1} \xrightarrow{R_i} G_i \xrightarrow{R_{i+1}} \cdots \xrightarrow{R_n} G_n$, where

1. $G_0, G_1, \ldots, G_{i-1}, G_i, \ldots, G_n$ are +RG graphs,
2. $R_1, \ldots, R_i, \ldots, R_n$ are graph rules in Tables 1, 2 and 3,
3. graph $G_i$ is obtained from graph $G_{i-1}$ by application of rule $R_i$, which we denote by $G_{i-1} \xrightarrow{R_i} G_i$, for each $i \in \{1, \ldots, n\}$.

**Example 2.** An example of graph derivation is

$$G_1 \xrightarrow{\text{Int}} G' \xrightarrow{\text{Cmp}} G_2 \xrightarrow{\text{GCv}} G_4 \xrightarrow{\text{Unv}} H' \xrightarrow{\text{Cmp}} H'' \xrightarrow{\text{Uni}} G_5,$$

which can be summarized as follows

$$G_1 \xrightarrow{\text{Int,Cmp}} G_2 \xrightarrow{\text{GCv}} G_4 \xrightarrow{\text{Unv,Cmp,Uni}} G_5.$$

Some examples of graph derivations appear in Figures 11, 12 and 13 in Section 5; see also [9, 10].

Given graphs $G$ and $H$, we say that $H$ is derivable from $G$, denoted by $G \vdash H$, when there exists a graph derivation $G_0, R_1, G_1, \ldots, R_n, G_n$ such that $G = G_0$ and $G_n = H$, and we call $G$ and $H$ inter-derivable, denoted by $G \dashv H$, when $G \vdash H$ and $H \vdash G$. We say that a graph inclusion $G \subseteq H$ is provable, denoted by $\vdash G \subseteq H$, when $G \vdash H$.

Our set of rules is not minimal: some rules are derivable from others. For instance, rules $\text{Spl}^-$ and $\text{Spl}^+$ are derivable, one can simulate $\text{Spl}^+$ as

$$G \cup \{(N, A, x, y)\} \xrightarrow{\text{Spl}}$$

$$G \cup \{(N \cup \{u'\}, \text{split}_{u'} A, x, y)\} \xrightarrow{\text{GCv}}$$

$$G \cup \{(N \cup \{u'\}, \text{split}_{u'} A, x, u')\}.$$
Also, Example 1 (in Section 3.1) shows a graph cover \(G_2 \leftarrow G_4\); so we have a graph derivation \(G_2 \xrightarrow{GCv} G_4\) (cf. Example 2). This application of rule GCv to derive \(G_4\) from \(G_2\) can be simulated as follows

\[
G_2 \xrightarrow{Spl} G_6 \xrightarrow{ErA^4} G_7 \xrightarrow{ErN} G_3 \xrightarrow{AdS} G_4,
\]

where \(G_6\) and \(G_7\) are the graphs in Figure 9.

**Proposition 1.** Rule GCv is equivalent to the set of rules in Table 3.

**Proof.** It is easy to see that the structural rules are instances of GCv. For example, the downward direction of rule Spl is an instance of the rule GCv with the homomorphism \(\phi : (N \cup \{u'\}, \text{splt}_{u'}^u A, x, y) \rightarrow (N, A, x, y)\), given by \(\phi u' := u\) and \(\phi v := v\), for \(v \in N\). Now, for the upward direction of Spl, take \(\phi : (N, A, x, y) \rightarrow (N \cup \{u'\}, \text{splt}_{u'}^u A, x, y)\) as the inclusion map. In either direction, for the unaffected slices of \(G\), take the homomorphism as the identity map. Conversely, assume that \(H\) covers \(G\), i.e. \(G \leftarrow H\). Each slice of \(G\) can be transformed into a slice of \(H\), by first using rules Spl, ErA, and ErN and then using AdS to add the slices of \(H\) not obtained in the first step. \(\square\)

3.3. Soundness, completeness and decidability

We now examine soundness and completeness of +RG. For completeness, we will implement the following strategy. First, we show that every graph can be transformed into an equivalent one in a normal form, by applications of rules in Table 1. Second, we show that the inclusion of graphs in normal
form can be decided by testing graph covering. The combination of these steps will provide completeness.

Soundness is clear.

**Proposition 2 (Soundness).** If $\vdash G \sqsubseteq H$ then $\models G \sqsubseteq H$.

**PROOF.** The assertion follows from the soundness of each rule (cf. Section 3.1). □

Let $S$ be a slice of $+\text{RG}$. We say that $S$ is *simple* if all its arcs are labeled by relational variables. Let $G$ be a graph of $+\text{RG}$. We say that $G$ is *simple* if all its slices are simple. A *simple normal form* of $G$ is a simple graph $H$ of $+\text{RG}$ that can be obtained from $G$ by applications of the elimination rules; in this case we write $H = \text{SNFG}$. For instance, $G_2 = \text{SNFG}_1$ (Figure 1 in Section 2). Clearly, $G$ and $\text{SNFG}$ are inter-derivable: $G \vdash \text{SNFG}$. These ideas lead to the next lemma guaranteeing the first step of the strategy.

**Lemma 1 (Normal form).** Any $+\text{RG}$ graph $G$ can be effectively transformed into a unique simple normal form $\text{SNFG}$.

**PROOF.** By induction one eliminates all the relational operators in $G$. □

The second step mentioned above can be established by constructing a finite canonical model. Given a simple slice $S = (N,A,x,y)$, its canonical model is $\mathfrak{S} := (N,r^{\mathfrak{S}})_{i \in \omega}$, where $r^{\mathfrak{S}} := \{(u,v) \in N \times N : ur_{i}v \in A\}$, for $i \in \omega$. Notice that $(x,y) \in [S]_{\mathfrak{S}}$, as the identity is an assignment for $S$ in $\mathfrak{S}$.

The next result gives the basic properties of the canonical model.

**Proposition 3.** Let $S = (N,A,x,y)$ be a simple slice and $\mathfrak{S}$ be the canonical model of $S$. Given any simple slice $T = (N',A',x',y')$,

1. each assignment $g$ for $T$ in $\mathfrak{S}$ with $gx' = x$ and $gy' = y$ is a homomorphism $g : T \rightarrow S$;
2. if $(x,y) \in [T]_{\mathfrak{S}}$, then $S$ is covered by $T$, i.e. $S \leftarrow T$.

**PROOF.** The first assertion holds because $g$ preserves arcs from $T$ to $S$ by the construction of the canonical model (if $ur_{i}v \in A'$, then $(gu,gv) \in r^{\mathfrak{S}}_{i}$, whence $ur_{i}v \in A$). The second assertion follows from the first one (assuming $(x,y) \in [T]_{\mathfrak{S}}$, there is an assignment $g$ for $T$ in $\mathfrak{S}$ with $gx' = x$ and $gy' = y$, so we have a homomorphism $g : T \rightarrow S$). □
From these results, we can obtain the following connections.

**Theorem 1.** Given graphs $G$ and $H$ of $+\text{RG}$, the following assertions are equivalent.

1. Inclusion $G \sqsubseteq H$ is valid, i.e. $\models G \sqsubseteq H$.
2. For every slice $S$ of $\text{SNFG}$, one can effectively determine a slice $T_S$ of $\text{SNFH}$ that covers $S$, i.e. $S \leftarrow T_S$.
3. $\text{SNFG}$ is covered by $\text{SNFH}$, i.e. $\text{SNFG} \leftarrow \text{SNFH}$.
4. Inclusion $G \sqsubseteq H$ is provable, i.e. $\vdash G \sqsubseteq H$.

**Proof.** We will show (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (1).

(1) $\Rightarrow$ (2) Each slice $S \in \text{SNFG}$ gives rise to a canonical model $\mathfrak{S}$. Now $\models \text{SNFG} \sqsubseteq \text{SNFH}$, as $\models G \sqsubseteq H$. So, since $(x, y) \in [S]_{\mathfrak{S}} \subseteq [\text{SNFG}]_{\mathfrak{S}}$, one can find a slice $T_S \in \text{SNFH}$ such that $(x, y) \in [T_S]_{\mathfrak{S}}$, whence Proposition 3 yields $S \leftarrow T_S$.

(2) $\Rightarrow$ (3) By definition (cf. Section 3.1).

(3) $\Rightarrow$ (4) We have $G \vdash \text{SNFG}$, $\text{SNFG} \xrightarrow{G\text{cc}} \text{SNFH}$ and $\text{SNFH} \vdash H$.

(4) $\Rightarrow$ (1) By soundness (cf. Proposition 2). □

The preceding result yields the decidability of some problems related to $+\text{RG}$ as well as the completeness of normal derivations.

**Corollary 1 (Decidability).** The following graph problems are decidable:

1. the validity of inclusions (whether $\models G \sqsubseteq H$ or not);
2. graph derivability (whether $G \vdash H$ or not).

**Proof.** Immediate from Lemma 1 and the equivalences (1) $\Leftrightarrow$ (3) $\Leftrightarrow$ (4) in Theorem 1, i.e. $\models G \sqsubseteq H$ iff $\text{SNFG} \leftarrow \text{SNFH}$ iff $G \vdash H$. □

An algorithm for either one of the above problems is as follows (Table 4):

1. First, convert the given graphs $G$ and $H$ to their respective simple normal forms $\text{SNFG}$ and $\text{SNFH}$, by applying elimination rules in Table 1.
2. Next, decide whether or not $\text{SNFG}$ is covered by $\text{SNFH}$. This is possible since we deal only with finite graphs.
Table 4: Algorithm for deciding whether \( \models G \sqsubseteq H \), or \( \vdash G \sqsubseteq H \).

<table>
<thead>
<tr>
<th>step 1</th>
<th>step 2</th>
<th>step 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G \vdash )</td>
<td>SNFG</td>
<td>( H \vdash )</td>
</tr>
<tr>
<td>elimination</td>
<td>cover?</td>
<td>elimination</td>
</tr>
</tbody>
</table>

A normal graph derivation is one with at most one application of the Graph Cover rule following eliminations and preceding introductions, if any (Figure 5). For instance, in Example 2 (in Section 3.2), we summarized the major steps of a graph derivation as follows

\[
G_1 \xrightarrow{\text{Int; Cmp}} G_2 \xrightarrow{\text{GCv}} G_4 \xrightarrow{\text{Unv; Cmp; Uni}} G_5.
\]

This summary displays elimination section Int; Cmp (\( G_1 \vdash G_2 \)), cover section GCv (\( G_2 \vdash G_4 \)), and introduction section Unv; Cmp; Uni (\( G_4 \vdash G_5 \)).

More precisely, we call a graph derivation \( G_0, R_1, G_1, \ldots, R_n, G_n \) normal when there exists \( i < j \leq n \) such that

1. \( R_1, \ldots, R_i \) are elimination (downward) rules,
2. \( R_j, \ldots, R_n \) are introduction (upward) rules, and
3. either \( j = i + 1 \) or else \( j = i + 2 \) and \( R_{i+1} \) is GCv.

As observed, our main result also yields the completeness of normal derivations.

**Corollary 2 (Completeness of normal derivations).** Given \( G \) and \( H \) graphs of \( +\text{RG} \), \( \models G \sqsubseteq H \) iff there exists a normal derivation of \( H \) from \( G \).

**Proof.** The assertion follows from the equivalence (1) \( \Leftrightarrow \) (3) in Theorem 1.

\( \square \)

4. **Expressive power**

In this section, we characterize the expressive powers of \( +\text{RC} \) and \( +\text{RG} \) by comparing them with first-order logic. Given a non-empty set \( M \), a graph
Table 5: Normal form for graph derivations in $+\text{RG}$.

$G \downarrow$ elimination of operators (Table 1)
$G' \downarrow$ (at most) one application of $\text{GCv}$ (Table 2)
$H' \downarrow$ introduction of operators (Table 1)
$H$

defines a binary relation on $M$, according to constraints imposed to its arcs. These constraints are obtained by the labeling of arcs with relational variables. The global definition of that relations are expressed by means of sequentialization (composition), parallelism (intersection), choice (union) and reversion (converse) of arcs and identification (identity) of nodes. Composition may be viewed as a restrict form of existential quantification, intersection and union are related to the usual connectives of conjunction and disjunction, and converse is closely related to the interchange of variables. So, the following version of first-order language seems to be quite adequate to be used as a yardstick in measuring the expressive power of the graph relational language.

Let $\text{Ivar} = \{x_i : i \in \omega\}$ be a set of individual variables, typically denoted by $x, y, z$, and $\text{Rvar} = \{r_i : i \in \omega\}$ be a set of relational symbols, typically denoted by $r, s, t$. The formulas of $+\exists\text{FOL}(R)$, typically denoted by $\varphi, \psi$, are defined according to the following grammar:

$$\varphi ::= xry \mid x \approx y \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \exists x \varphi.$$ 

We freely use all the syntactic notions, properties and conventions of first-order languages when restricted to $+\exists\text{FOL}(R)$. We denote by $\bar{x}, \bar{y}, \bar{z}$ arbitrary finite sequences of individual variables. In particular, any of $\bar{x}, \bar{y}, \bar{z}$ can be empty. We use set theoretical notation when referring to $\bar{x}, \bar{y}, \bar{z}$ in an usual way.

The semantics for $+\exists\text{FOL}(R)$ is just the first-order one restricted to the positive language. So, the models for $+\text{RC}$, $+\text{RG}$ and $+\exists\text{FOL}(R)$ are the
same. This simplifies the comparison of the expressive powers of these formalisms.

We will now characterize the expressive powers of $+\text{RC}$ and $+\text{RG}$ in terms of two fragments of $+\exists\text{FOL}(R)$. Let $+\exists\text{FOL}(R)^{xy}$ consist of the formulas of $+\exists\text{FOL}(R)$ having at most $x$ and $y$ as free variables, and let $+\exists\text{FOL}(R)^{xyz}$ consist of the formulas of $+\exists\text{FOL}(R)^{xy}$ having at most $x, y$ and $z$ as variables.

The next result parallels the analogous one for the Tarski relational formalism [22]. To prove it, we adapt some ideas from [2], which defines translations from $+\text{RC}$ to $+\exists\text{FOL}(R)^{xyz}$, and back. Our translations do not use renaming of variables.

For a formula $\varphi$ whose only free variables are among $x$ and $y$ we employ the usual abbreviation $\mathcal{M} \models \varphi[m,n]$ for $\mathcal{M}, \beta \models \varphi$ for any first-order assignment $\beta$ in $M$ such that $\beta x = m$ and $\beta y = n$.

**Proposition 4.**
1. For each term $R$ of $+\text{RC}$ there exists a formula $\varphi_R$ of $+\exists\text{FOL}(R)^{xy}$ such that:

   $$(m, n) \in \llbracket R \rrbracket_{\mathcal{M}} \text{ iff } \mathcal{M} \models \varphi_R[m,n],$$

   for every model $\mathcal{M}$ and individuals $m, n \in M$.

2. Conversely, for each formula $\varphi$ of $+\exists\text{FOL}(R)^{xyz}$ there exists a term $R_\varphi$ of $+\text{RC}$ such that:

   $$\mathcal{M} \models \varphi[m,n] \text{ iff } (m, n) \in \llbracket R_\varphi \rrbracket_{\mathcal{M}}.$$ 

   for every model $\mathcal{M}$, individuals $m, n \in M$.

**Proof.** (Outline) To translate from terms to formulas, we define six auxiliary functions, $FT_{ab}, FT_{ba}, \ldots, FT_{cb}$, translating terms of $+\text{RC}$ into formulas of $+\exists\text{FOL}(R)^{xyz}$ as follows. Let $a, b \in \{x, y, z\}$ be two distinct individual variables. The forward translation function associated with the individual variables $a$ and $b$, $FT_{ab}$, is defined recursively by the following rules. In the last two clauses, $c \in \{x, y, z\}$ is the individual variable distinct from $a$ and $b$.

\[
\begin{align*}
FT_{ab}E & \colon= a \approx a \land b \approx b, \\
FT_{ab}I & \colon= a \approx b, \\
FT_{ab}R & \colon= arb, \\
FT_{ab}R^T & \colon= FT_{ba}R, \\
FT_{ab}R \land S & \colon= FT_{ab}R \land FT_{ab}S, \\
FT_{ab}R \lor S & \colon= FT_{ab}R \lor FT_{ab}S, \\
FT_{ab}R \circ S & \colon= \exists c(FT_{ac}R \land FT_{cb}S).
\end{align*}
\]
Finally, we define the forward translation of $R$ as the expression $FTR := FT_{xy}R$.

We prove by induction on terms the proper syntactical and semantical behaviors of the auxiliary translation functions $FT_{ab}, \ldots, FT_{cb}$, leading to the conclusion that $FT$ is a translating function from terms of $\mathcal{+RC}$ into formulas of $\mathcal{+\exists FOL}(R)_{xy}^z$, having occurrences of exactly $x$ and $y$ free, that preserves the meaning of terms:

1. $FTR \in \mathcal{+\exists FOL}(R)_{xy}^z$, for every term $R$ of $\mathcal{+RC}$;
2. $(m, n) \in \llbracket R \rrbracket_M$ iff $\mathcal{M} \models FTR[m, n]$, for every model $\mathcal{M}$ and individuals $m, n \in M$.

To translate from formulas to terms, we need some more work. First, we define six auxiliary fragments, $\text{NICE}^{ab}$, $\text{NICE}^{ba}$, $\ldots$, $\text{NICE}^{cb}$, of $\mathcal{+\exists FOL}(R)_{xy}^z$, by mutual recursion as follows. Given two distinct individual variables $a, b \in \{x, y, z\}$, the formulas of $\text{NICE}^{ab}$ are the positive existential first-order formulas given by the following rules:

1. Formulas $a \approx a$, $a \approx b$, $b \approx a$, $b \approx b$, $ara$, $arb$, $bra$, and $brb$ are formulas of $\text{NICE}^{ab}$;
2. If $\varphi$ and $\psi$ are formulas of $\text{NICE}^{ab}$, then $(\varphi \land \psi)$ and $(\varphi \lor \psi)$ are formulas of $\text{NICE}^{ab}$;
3. If $\varphi$ is a formula of $\text{NICE}^{ac}$ and $\psi$ is a formula of $\text{NICE}^{cb}$, then $\exists c(\varphi \land \psi)$ is a formula of $\text{NICE}^{ab}$.

Second, we define a backward translation function, $BT_{ab}$, associated with the individual variables $a, b$, recursively by the following rules, applied to the formulas of $\text{NICE}^{ab} \cup \text{NICE}^{ba}$, having free occurrences of $a$ or $b$:

\begin{align*}
BT_{ab}ara & := (r \land I) \circ E, \\
BT_{ab}arb & := r, \\
BT_{ab}bra & := r^\top, \\
BT_{ab}brb & := E \circ (r \land I), \\
BT_{ab}a \approx a & := E \\
BT_{ab}a \approx b & := I \\
BT_{ab}b \approx a & := I \\
BT_{ab}b \approx b & := E \\
BT_{ab}(\varphi \land \psi) & := BT_{ab}\varphi \land BT_{ab}\psi, \\
BT_{ab}(\varphi \lor \psi) & := BT_{ab}\varphi \lor BT_{ab}\psi, \\
BT_{ab}\exists c(\varphi \land \psi) & := \begin{cases} BT_{ac}\varphi \circ BT_{cb}\psi, & \text{if } \exists c(\varphi \land \psi) \in \text{NICE}^{ab}, \\
BT_{ac}\psi \circ BT_{cb}\varphi, & \text{otherwise.} \end{cases}
\end{align*}
Observe that, since \( \varphi \) belongs to \( \text{NICE}^{ab} \cup \text{NICE}^{ba} \), in the rules corresponding to the operators \( \land, \lor, \exists \), we have:

- For \( \varphi \land \psi \), formulas \( \varphi \) and \( \psi \) both belong to \( \text{NICE}^{ab} \) or both belong to \( \text{NICE}^{ba} \), and in both cases, by recursion, \( BT_{ab} \) applies. A similar remark holds for \( \varphi \lor \psi \).

- For \( \exists c (\varphi \land \psi) \), if \( \exists c (\varphi \land \psi) \) belongs to \( \text{NICE}^{ab} \), then formulas \( \varphi \) and \( \psi \) belong, respectively, to \( \text{NICE}^{ac} \) and to \( \text{NICE}^{cb} \). In this case, by recursion, \( BT_{ac} \) applies to \( \varphi \) and \( BT_{cb} \) applies to \( \psi \). Otherwise, formula \( \exists c (\varphi \land \psi) \) belongs to \( \text{NICE}^{ba} \) and the formulas \( \varphi \) and \( \psi \) belong, respectively, to \( \text{NICE}^{bc} \) and to \( \text{NICE}^{ca} \). So, in this case, by recursion, \( BT_{ac} \) applies to \( \psi \) and \( BT_{cb} \) applies to \( \varphi \).

Hence, \( BT_{ab} \) is well defined.

Third, the **backward translation** of \( \varphi \in \text{NICE}^{xy} \cup \text{NICE}^{yx} \) to terms is defined by \( BT \varphi := BT_{xy} \varphi \).

We prove by induction on formulas the proper syntactical and semantical behaviors of the auxiliary translation functions \( BT_{ab}, \ldots, BT_{cb} \), leading to the conclusion that \( BT \) is a translating function from formulas \( \varphi \) of \( \text{NICE}^{xy} \cup \text{NICE}^{yx} \) such that \( \text{free} \varphi = \{ x, y \} \) into terms of \( +RC \) that preserves the meaning of formulas:

1. \( BT \varphi \) is a term of \( +RC \), for every formula \( \varphi \) of \( \text{NICE}^{xy} \cup \text{NICE}^{yx} \);
2. \( \mathfrak{M} \models \varphi [m, n] \) iff \( (m, n) \in [BT \varphi]_{\mathfrak{M}} \), for any model \( \mathfrak{M} \) and individuals \( m, n \in M \).

To translate the whole first-order fragment \( +\exists \text{FOL}(R)_{xy} \) it suffices to observe that \( BT \) can, in fact, translate any \( \land, \lor \) combination of formulas in \( \text{NICE}^{xy} \cup \text{NICE}^{yx} \) whose members \( \varphi \) satisfy \( \text{free} \varphi \subseteq \{ x, y \} \), and that every formula \( \varphi \) of \( +\exists \text{FOL}(R)_{xy} \) is equivalent to a \( \land, \lor \) combination of formulas in \( \text{NICE}^{xy} \cup \text{NICE}^{yx} \), having occurrences of the same variables and free variables as \( \varphi \). \( \Box \)

Now, we show that the graph language and the positive existential first-order fragment define the same relations in any model \( \mathfrak{M} \). This result is based on the fact that the disjunctive normal form of formulas of \( +\exists \text{FOL}(R) \) are very close to graphs of \( +\text{RG} \) in simple normal form.

We assume that neither \( x \) nor \( y \) are nodes of the graph language.
**Theorem 2.** Let $G$ be a graph of $+\text{RG}$. Then there exists a formula $\varphi_G$ of $+\exists\text{FOL}(R)^{xy}$ such that:

$$(m,n) \in [G]_{\mathcal{M}} \text{ iff } \mathcal{M} \models \varphi_G [m,n],$$

for every model $\mathcal{M}$ and individuals $m, n \in M$.

**Proof.** Let $\text{SNFG} = (N_j, A_j, x_j, y_j)_{j \in J}$ be the simple normal form of a graph $G$ of $+\text{RG}$ and define the formula $\varphi_G$ in the following way. Take:

$$\varphi_j := \begin{cases} 
\exists N_j - x_j (x \approx y \land \bigwedge_{urv \in A_j} urv \, x, x_j) & \text{if } x_j = y_j, \\
\exists N_j - x_j - y_j (\bigwedge_{urv \in A_j} urv \, x, y, x_j, y_j) & \text{otherwise}, 
\end{cases}$$

for each $j \in J$, and put $\varphi_G = \bigvee_{j \in J} \varphi_j$. Here, both $N_j - x_j$ and $N_j - x_j - y_j$ are viewed as ordered sequences of individual variables. Let $\mathcal{M}$ be a model and $m, n$ be individuals in $M$. We can prove $(m,n) \in [G]_{\mathcal{M}}$ iff $\mathcal{M} \models \varphi_G [m,n]$. An exhaustive case analysis shows that $\varphi_G$ is a formula of $+\exists\text{FOL}(R)^{xy}$ satisfying the required conditions. □

**Theorem 3.** Let $\varphi$ be a formula of $+\exists\text{FOL}(R)^{xy}$. Then there exists a graph $G_\varphi = (N_j, A_j, x, y)_{j \in J}$ of $+\text{RG}$, and:

$$\mathcal{M} \models \varphi [m,n] \text{ iff } (m,n) \in [G_\varphi]_{\mathcal{M}},$$

for every model $\mathcal{M}$ and individuals $m, n \in M$.

**Proof.** Let $\varphi$ be a formula of $+\exists\text{FOL}(R)^{xy}$. We may assume that $\varphi$ is:

$$\bigvee_{i=1}^{m} \exists x_i \varphi_i,$$

where each $\exists x_i \varphi_i$, $1 \leq i \leq m$, satisfies the following conditions:

- $\varphi_i$ is a conjunction of pairwise distinct atomic formulas;
- all variables in $x_i$ occur in $\varphi_i$.
Figure 10: Graph corresponding to term \( R \).

- If \( u \approx v \) is a sub-formula of \( \varphi_i \), then \( u, v \in \{ x, y \} \);
- \( x, y \not\in \text{bound} \varphi \).

Now, for each \( i, 1 \leq i \leq m \), define \( N_i = \text{Ivar} \varphi_i \),

\[
A_i = \{ uIv : u \approx v \text{ occurs in } \varphi_i \} \cup \{ urv : urv \text{ occurs in } \varphi_i \},
\]

and:

\[
S_i := \begin{cases} 
(N_i + x + y, A_i, x, y) & \text{if } \text{free}\exists x_i \varphi = \emptyset, \\
(N_i + y, A_i, x, y) & \text{if } \text{free}\exists x_i \varphi = \{ x \}, \\
(N_i + x, A_i, x, y) & \text{if } \text{free}\exists x_i \varphi = \{ y \}, \\
(N_i, A_i, x, y) & \text{if } \text{free}\exists x_i \varphi = \{ x, y \}.
\end{cases}
\]

Take \( G_\varphi = \{ S_i \}_{i \in \{1, \ldots, m\}} \). Let \( M \) be a model and \( m, n \) be individuals in \( M \).
We can prove \( M \models \varphi [m, n] \) iff \( (m, n) \in [G_\varphi]_M \). An exhaustive case analysis
shows that \( G_\varphi \) is a graph of \(+\text{RG}\) satisfying the required conditions. \( \square \)

5. Applications to the positive relational calculus

An inclusion of \(+\text{RC}\) is an expression of the form \( R \sqsubseteq S \), where \( R, S \) are
terms of \(+\text{RC}\). An inclusion \( R \sqsubseteq S \) is valid, denoted by \( \models R \sqsubseteq S \), when
\([R]_M \subseteq [S]_M \) for every model \( M \). An equality of \(+\text{RC}\) is an expression
of the form \( R = S \). An equality of \(+\text{RC}\) is valid, denoted by \( \models R = S \), when
\( \models R \sqsubseteq S \) and \( \models S \sqsubseteq R \). In this section, \(+\text{RG}\) will be used to decide
valid inclusions and, consequently, valid equalities of \(+\text{RC}\). The idea is to
correspond to each term \( R \) a graph \( G_R \) and to derive an inclusion \( R \sqsubseteq S \) by
using the corresponding graphs \( G_R \) and \( G_S \). The success of this approach
relies on the following. Let \( R \) be a term of \(+\text{RC}\). The graph corresponding
to \( R \) is defined as \( G_R = (\{ x, y \}, \{ xRy \}, x, y) \) and depicted as in Figure 10.
Lemma 2. For every model $\mathfrak{M}$, we have $[R]_{\mathfrak{M}} = [G_R]_{\mathfrak{M}}$.

PROOF. The equality follows from the fact that $xRy$ is the only arc in $G_R$. □

The next result shows that the graph calculus $+\mathcal{RG}$ is a complete deductive calculus for $+\mathcal{RC}$.

Theorem 4. For terms $R, S$ of $+\mathcal{RC}$, we have:

1. $\models R \subseteq S$ iff $G_R \vdash G_S$;
2. $\models R = S$ iff $G_R \vdash G_S$ and $G_S \vdash G_R$.

PROOF. (1) $\models R \subseteq S$ iff, for every $\mathfrak{M}$, $[R]_{\mathfrak{M}} \subseteq [S]_{\mathfrak{M}}$, iff (Lemma 2), for every $\mathfrak{M}$, $[G_R]_{\mathfrak{M}} \subseteq [G_S]_{\mathfrak{M}}$, iff (Corollary 2) $G_R \vdash G_S$.

(2) Follows from (1) and the fact that $\models R = S$ iff $\models R \subseteq S$ and $\models S \subseteq R$. □

Corollary 3. The Validity Problems for inclusion and equalities of $+\mathcal{RC}$ are decidable.

PROOF. To decide an inclusion $R \subseteq S$ of $+\mathcal{RC}$ apply the normal form for proofs algorithm from $+\mathcal{RG}$ given in Table 5 to the graphs $G_R$ and $G_S$. □

The homomorphism from the normal form of $G_S$ into the normal form of $G_R$ can be easily determined, according to the following heuristics, which works to all examples given in this paper. To each node $u$ associate a sequence $(r_1, \ldots, r_m; s_1, \ldots, s_n)$ of relational variables, where $r_1, \ldots, r_m$ is the possibly empty sequence of labels of arcs entering in $u$ and $s_1, \ldots, s_n$ is the also possibly empty sequence of labels of arcs leaving $u$. Now, given nodes $u$ from $\text{SNFG}_R$ and $u'$ from $\text{SNFG}_S$, with corresponding sequences $(r_1, \ldots, r_m; s_1, \ldots, s_n)$ and $(r'_1, \ldots, r'_k; s'_1, \ldots, s'_l)$, respectively, try to map $u'$ to $u$ when $\{r'_1, \ldots, r'_k\} \subseteq \{r_1, \ldots, r_m\}$ and $\{s'_1, \ldots, s'_l\} \subseteq \{s_1, \ldots, s_n\}$.

An interesting feature of $+\mathcal{RG}$ is that its rules are easy to grasp and use. For instance, the work of R. Lyndon [15, 16] yields that the inclusion $R \subseteq S$, where $R$ and $S$ are, respectively, the following $+\mathcal{RC}$ terms:

$$a \sqcap (((b \circ c) \sqcap d) \circ (e \sqcap (f \circ g)))$$
Figure 11: Graph proof of Lyndon’s inclusion.
and
\[
b \circ (((b^T \circ a) \cap (c \circ e)) \circ g^T) \cap (c \circ f) \cap (b^T \circ ((a \circ g^T) \cap (d \circ f))) \circ g,
\]

although valid is not derived in the relational formalism, from the Tarski axioms [21]. For more details, cf. [17]. Within the graph calculus, this inclusion can be proved, as shown in Figure 11.

Besides being powerful and playful, proofs in \(+\text{RG}\) tend to be simpler than proofs conducted inside the equational formalism. For instance, consider the equality:
\[
(r \circ s) \cap t = (r \circ ((r^T \circ t) \cap s)) \cap t,
\]
that plays an important role in the investigations on the relational semantics of programs pursued in [3]. It is easy to see that (1) is valid and we found two (semi-)equational proofs of it, one using the modular law
\[
r \cap (s \circ t) \subseteq s \circ ((s^T \circ r) \cap t)
\]
and an indirect one based on the Tarski axioms for relational algebras.

The first proof has two parts. First, from \(r \circ ((r^T \circ t) \cap s) \subseteq s\) we directly have \((r \circ ((r^T \circ t) \cap s)) \cap t \subseteq (r \circ s) \cap t\), by the monotonicity of composition. Second, since the modular law warrants \(t \cap (r \circ s) \subseteq (r \circ ((r^T \circ t) \cap s)) \cap t\), we have:
\[
(r \circ s) \cap t = (t \cap (r \circ s)) \cap t = (r \circ ((r^T \circ t) \cap s)) \cap t.
\]
Notice that the proof presented above is simple due to the close resemblance between the modular law and the result we want to establish.

The second proof also has two parts. First, we have:
\[
(r \circ s) \cap t = (r \circ (E \cap s)) \cap t = (r \circ ((r^T \circ t) \cup r^T \circ t \cap s)) \cap t = ((r \circ ((r^T \circ t \cap s)) \cap t) \cup ((r \circ (r^T \circ t \cap s)) \cap t).
\]
Also, we have \(\emptyset = (t^T \circ r) \cap \overline{t^T \circ r} = (r \circ \overline{r^T \circ r}) \cap t^T\). And since \(r^T \circ t \cap s \subseteq \overline{r^T \circ t},\) we obtain \(r \circ (r^T \circ t \cap s) \subseteq r \circ \overline{r^T \circ t}\) proving that \((r \circ (r^T \circ t \cap s)) \cap t \subseteq \emptyset\). Notice that, although the equality belongs to \(+\text{RC}\), the proof presented above uses laws on complementation and the empty set, making a detour through the full \text{RC}.
Figure 12: Another example of a graph proof.
Figure 13: Graph proof of the modular law.
A proof in $+\text{RG}$ of the left to right inclusion associated to (1) is shown in Figure 12. Observe that all steps occurring in it are reversible. So we have, in fact, a graph proof of (1).

Another aspect of $+\text{RG}$ is that it can also be used as a heuristic tool to help finding and generalizing some valid inclusions of $+\text{RC}$. For instance, consider the modular law, used in one of the proofs of the equality (1) given above. The proof in $+\text{RG}$ displayed in Figure 13 shows that (2) is valid. Besides, consider the graph $H$ obtained from $G$ by rule $\text{Cnv}$ (Figure 14).

From $H$, we may build terms based on walks, as follows. To each walk starting at the distinguished node $x$ and ending at the node $u$: $x \rightarrow R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow \cdots \rightarrow R_k \rightarrow u$

we associate the term $R_1 \circ R_2 \circ \cdots \circ R_k$. To a set of such walks $C_1, C_2, \ldots, C_m$, we associate the term $C_1 \sqcap C_2 \sqcap \cdots \sqcap C_m$. Analogously, we associate terms $S_1 \circ S_2 \circ \cdots \circ S_l$ to walks starting at $u$ and ending at the distinguished node $y$ and terms $D_1 \sqcap D_2 \sqcap \cdots \sqcap D_n$ to sets of such walks. One can prove, using $+\text{RG}$, that $r \sqcap (s \circ t) \subseteq (C_1 \sqcap \ldots \sqcap C_m) \circ (D_1 \sqcap \ldots \sqcap D_n)$, obtaining a far reaching generalization of (2).

6. Perspectives

We have given a formal treatment to $+\text{RG}$, a relational calculus based on graphs, presenting soundness and completeness results for the valid inclusions and obtaining a finite model property. This is done in a general framework, obtaining as corollaries completeness and decidability of related graph calculi. We have also compared the expressive power of $+\text{RG}$ to a first-order language fragment showing that both define the same binary relations.
One may regard $+\mathbf{RG}$ as a notational variant of the positive existential first-order logic of binary relations. This perspective leaves open the possibility of developing positive first-order logic as graph calculus. Besides interesting from the philosophical perspective, this development may be profitable since, as we illustrate in this paper, the graph rules are playful, making the graph calculus a powerful pedagogical tool.

The non-finite axiomatizability of the valid inclusions of $+\mathbf{RC}$ is a consequence of a general result of H. Andrèka [1] which states that no algebra of relations whose set of basic operations contains $\cap$, $\cup$ and $\circ$ is finitely axiomatizable. This does not preclude infinite axiomatizations and, in fact, the existence of a set of positive axioms follows from a result of B.M. Schein [18] which states that any algebra of relations (satisfying some very general conditions) is axiomatizable by a (recursive) set of universal axioms. To the best of our knowledge, no explicit infinite set of axioms to $+\mathbf{RC}$ has been exhibited. Paralleling the results of B. Jónsson [14], we believe that to describe such a set in simple terms is a challenging task. The quest for axiomatizability of $+\mathbf{RC}$ and some of its subreducts is one of the problems stated in [19]. Of course, since the set of valid equalities of $+\mathbf{RC}$ is decidable, it can be used as a set of equational axioms and the algorithm that decides the valid inclusions of $+\mathbf{RC}$, can be viewed as a rough presentation of this set. We hope to use intrinsic properties of $+\mathbf{RG}$ to obtain more perspicuous ways to describe it.

As a final remark, we should mention that in this paper we have considered just a weak form of completeness. Proving from hypotheses is a major improvement of $+\mathbf{RG}$, which will lead to a graph derivation system adequate for the whole relational formalism of the Tarski relational calculus: complementation can be defined in set-theoretical terms, as usual. Since the equational theory of the Tarski relational calculus is undecidable [22], such a development will necessarily lead us to a very complex system. It is easy to extend $+\mathbf{RG}$ to deal with the empty relation. Extending $+\mathbf{RG}$ to deal with derivations from hypothesis, thereby resulting in an undecidable calculus, involves a much more elaborated work [23].

One can establish a parallel between $+\mathbf{RG}$ and a graph calculus for derivations from hypotheses, on one side, and propositional and first-order logic, on the other side. Propositional calculus is decidable, easy to grasp and largely applicable. L. Kálmar’s proof of weak completeness for propositional calculus provides a nice decision procedure for valid propositions. Our system, as illustrated in this paper, has a wide range of applications and is easily handled. In contrast, the intellectual complexity involved in proving

29
strong completeness for a graph calculus would be more like that involved in Henkin’s completeness proof.

We expect that a good understanding of $+RG$ will pave the way to investigate important fragments of the whole relational formalism.

**Acknowledgments.** We would like to thank the anonymous referees for their helpful suggestions, contributing to the improvement of the quality of our paper.

**References**


