A graphical representation of relational formulae with complementation*

Domenico Cantone, Marianna Nicolosi Asmundo
Università di Catania; {cantone|nicolosi}@dmi.unict.it
Andrea Formisano
Università di Perugia; formis@dmi.unipg.it
Eugenio G. Omodeo
Università di Trieste; eomodeo@units.it

Abstract

We study translations of dyadic first-order sentences into equalities between relational expressions. The proposed translation techniques (which work also in the converse direction) exploit a graphical representation of formulae in a hybrid of the two formalisms. A major enhancement relative to previous work is that we can cope with the relational complement construct and with the negation connective.

Complementation is handled by adopting a Smullyan-like uniform notation to classify and decompose relational expressions; negation is treated by means of a generalized graph-representation of formulae in $\mathcal{L}^+$, and through a series of graph-transformation rules which reflect the meaning of connectives and quantifiers.

KEY WORDS: Algebra of binary relations, Quantifier elimination, Graph transformation.

Contents

Introduction

The possibility to exploit map calculus for mechanical reasoning can be grasped from [24], where Tarski and Givant show how to reformulate most axiomatic systems of Set Theory as equational theories based on a relational language devoid of quantifiers. Map calculus [16] cannot represent *per se* an alternative to predicate calculus. As for expressive power, it corresponds in fact to a fragment of first-order logic endowed with only three individual variables and with binary predicates only. As for deductive power, it is semantically incomplete; that is, there are semantically valid equations which are not derivable within it. Moreover, predicate logic has acquired such an unquestioned status of *de facto* standard as to make one reluctant to adopt the map formalism in its stead, in spite of the greater conciseness of the latter. Nonetheless, map calculus can be applied in synergy with predicate calculus, inside theorem provers or proof assistants, as an inferential engine to be used in the activities of proof-search and model building [13, 17, 1]. In fact, thanks to its pure equational and algebraic character (as well as to the absence of quantification), the mechanization of map calculus can benefit from well-established specific

*This research is partially supported by GNCS-10, GNCS-11, and PRIN-08 projects, and by grants 2009.010.0336 and 2010.011.0403. We would like to thank the anonymous referees for their useful suggestions.

proof-technology developed through several decades of scientific research. The availability of cross-translation algorithms between predicate logic and map calculus comforts one in foreseeing combined approaches to (first-order) theorem proving. In this frame of mind, the reasoning activity should develop by switching between two different levels. On the one hand, deduction can proceed at a '*higher level*' by exploiting well-known proof-technology for first-order logic. On the other hand, the first-order level might invoke the '*lower level*' equational reasoner.

In developing the needed translation techniques (and to increase readability), it is useful to design algorithms allowing one to represent map formulae in a visually alluring way, so as to exploit the immediate perspicuity of graphics.

The translation of formulae in both directions, between predicate logic and map calculus, has been addressed in [4, 5], where an algorithm for translating formulae of dyadic predicate logic into map calculus and an algorithm for converting map expressions into a graphical representation have been presented. Both algorithms are based on suitably defined graphs and are in fact specified by means of graph-transformation rules. One of them is designed to treat existentially quantified conjunctions of literals, the other to treat map expressions involving the constructs of relational intersection, composition, and conversion.

In this paper the techniques introduced in [4, 5] are extended, so as to treat formulae which involve the negation connective and expressions involving the relational complement construct. This allows us to get a graphical representation of any map expression, and to process any formula of dyadic predicate logic with the aim of getting an equivalent map equation. This goal is not always achieved: the algorithm which we will present sometimes fails to find the sought translation even if it exists. This apparent drawback, which also affected the earlier versions of the algorithm, stems from an unsurmountable limiting result [22], namely the fact that no algorithm can establish in full generality whether a given first-order sentence in $n+1$ variables is logically equivalent to some other sentence in n variables.

The enhanced techniques in this paper have been obtained by extending Smullyan's unifying notation, originally devised for formulae of predicate logic, to cover map expressions too. Moreover, we enrich the notion of directed multigraph associated in [5] with formulae and map expressions, so that:

- the multigraph is not necessarily connected: it is partitioned into disjoint subgraphs, its *components*, which, in their turn, are not necessarily connected;
- nodes are labeled with sets of variables (instead of with single variables);
- a relation $\rightsquigarrow$ is introduced between edges and components of the multigraph (intuitively, such a relation associates each disjunctive subformula with subgraphs representing the complements of its disjuncts).

Organization of the paper. Section 1 introduces the two languages to be treated, namely the deductive formalism $\mathcal{L}^\times$ for algebraic logic and Tarski's extension $\mathcal{L}^+$ of traditional dyadic first-order predicate logic with the constructs of $\mathcal{L}^\times$; moreover, we review here the basic toolkit for syntactic manipulation (syntax tree, occurrence, extended Smullyan's classification of formulae, etc.). Section 2 offers a way of representing formulae in $\mathcal{L}^+$ through specialized multigraphs. Section 3 pinpoints a number of meaning-preserving transformation rules for such multigraphs; and Sections 4 and 5 provide algorithms which, by applying these rules in a rather rigid order, translate $\mathcal{L}^\times$ into the traditional sublanguage of $\mathcal{L}^+$ and, conversely (but only in favorable cases), translate formulae of $\mathcal{L}^+$ into $\mathcal{L}^\times$. Section 6 relates the translation techniques proposed in this paper with others, found in the literature.

1 The languages $\mathcal{L}^\times$ and $\mathcal{L}^+$

$\mathcal{L}^\times$ is an equational language devoid of variables where one can state properties of dyadic relations, *maps*, over an unspecified, yet fixed, *domain $\mathcal{U}$ of discourse*. Its basic ingredients are three *constants* $\mathbf{0}$, $\mathbf{1}$, ι; a collection of *map letters* $\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3, \ldots$; dyadic constructs $\cap$, $\cup$, **;** of map *intersection*, map *union*, and map *composition*; and the monadic constructs $\overline{}$ and $\smile$ of map *complementation* and *conversion*. (Further defined constructs, such as relational sum †, can be introduced, e.g. by putting $P \dagger Q =_{\mathrm{Def}} \overline{\overline{P};\overline{Q}}$.) A *map expression* is any term P, Q, R,... built up from this signature in the usual manner.[1] A *map equality* is a writing of the form $Q{=}R$, where both Q and R are map expressions.

Once a nonempty domain $\mathcal{U}$ has been fixed, the map constants $\mathbf{0}$, $\mathbf{1}$, and ι are always interpreted by putting: $\mathbf{0}^\Im =_{\mathrm{Def}} \emptyset$, $\mathbf{1}^\Im =_{\mathrm{Def}} \mathcal{U}^2 =_{\mathrm{Def}} \mathcal{U}{\times}\mathcal{U}$, and $\iota^\Im =_{\mathrm{Def}} \{[a,a] : a \in \mathcal{U}\}$. A specific interpretation $\Im$, based on $\mathcal{U}$, is determined by associating subsets $\mathsf{p}_1^\Im, \mathsf{p}_2^\Im, \mathsf{p}_3^\Im, \ldots$ of $\mathcal{U}^2$ with the map letters p_i. Then, on the basis of the usual evaluation rules:

$$\begin{array}{rll}
(Q{\cap}R)^\Im & =_{\mathrm{Def}} & \{\, [a,b] \in Q^\Im \;:\; [a,b] \in R^\Im \,\} \\
(Q{\cup}R)^\Im & =_{\mathrm{Def}} & \{\, [a,b] \in \mathcal{U}^2 \;:\; [a,b] \in Q^\Im \text{ or } [a,b] \in R^\Im \,\} \\
(Q;R)^\Im & =_{\mathrm{Def}} & \{\, [a,b] \in \mathcal{U}^2 \;:\; \text{some } c \text{ exists s.t. } [a,c] \in Q^\Im \text{ and } [c,b] \in R^\Im \,\} \\
(\overline{Q})^\Im & =_{\mathrm{Def}} & \{\, [a,b] \;:\; [a,b] \in \mathcal{U}^2 \setminus Q^\Im \,\} \\
(Q^\smile)^\Im & =_{\mathrm{Def}} & \{\, [b,a] \;:\; [a,b] \in Q^\Im \,\}\;,
\end{array}$$

[1] To improve readability, in writing expressions we often adopt implicit priorities. Namely, we assume that constructs are ordered in decreasing priority as follows: $\overline{}$, $\smile$, **;**, †, $\cap$, $\cup$.

each map expression P comes to designate a specific map $P^{\Im}$, and each equality $Q{=}R$ turns out to be either true or false. Two expressions Q and R are *equivalent* if for every interpretation $\Im$ it holds that $Q^{\Im} = R^{\Im}$.

The language $\mathcal{L}^+$ is a variant of a first-order dyadic predicate language. Let *Var* be a collection of symbols of variables (ranging over $\mathcal{U}$). An *atomic formula* of $\mathcal{L}^+$ has either the form xPy or the form $Q{=}R$, where x, y are individual variables in *Var* and Q, R stand for map expressions. Given an atomic formula F of the first form, $Map(F)$ denotes the map expression P of F. (For instance, $Map(xR;\overline{Q}y) = R;\overline{Q}$.) Propositional connectives and existential/universal quantifiers are employed as usual, save that we treat $\wedge$ and $\vee$ as connectives of variable arity.

We assume as known the notions of: syntax tree of a *well-formed expression* of $\mathcal{L}^+$ (in short, *wfe*), literal, (immediate) subformulae of a given formula, sentence, and so on.[2] Precise definitions can be found in [8, 12].

It is convenient to assume that the individual variables in *Var* are arranged in a sequence $\langle \ldots, \mathtt{x}_{-2}, \mathtt{x}_{-1}, \mathtt{x}_0, \mathtt{x}_1, \mathtt{x}_2, \ldots \rangle$ whose two subsequences $Var^- = \langle \mathtt{x}_{-1}, \mathtt{x}_{-2}, \ldots \rangle$ and $Var^+ = \langle \mathtt{x}_1, \mathtt{x}_2, \ldots \rangle$ are used as repositories of bound and free variables in formulae of $\mathcal{L}^+$, respectively. We reserve the variable $\mathtt{x}_0$ for a special rôle, to be explained in Section 5. Variables indexed by an odd (resp., even) negative number will play the role of existentially (resp., universally) quantified variables. When it is not necessary to insist on such conventions we will use the metavariables $x, y, z, \ldots$, that, as mentioned above, stand for generic variables in *Var*.

We define a variable assignment $\mathbf{a} : Var^+ \rightarrow \mathcal{U}$ to be a mapping associating elements of $\mathcal{U}$ with free variables. The interpretation $\Im$, introduced above to interpret map expressions, can be extended to Boolean connectives and quantifiers as usual to define the semantics of formulae of $\mathcal{L}^+$. We write $(\varphi)^{\Im,\mathbf{a}}$ to denote the Boolean value resulting from the application of the interpretation $\Im$ and of the variable assignment $\mathbf{a}$ to a given formula φ of $\mathcal{L}^+$. We say that φ is satisfied by an interpretation $\Im$ and a variable assignment $\mathbf{a}$ if $(\varphi)^{\Im,\mathbf{a}}$ is true; φ is satisfied by an interpretation $\Im$, that is $(\varphi)^{\Im}$ is true, if $(\varphi)^{\Im,\mathbf{a}}$ is true, for every assignment $\mathbf{a}$. Two formulae φ and ψ are logically equivalent if $(\varphi)^{\Im}$ is true if and only if $(\psi)^{\Im}$ is true, for every interpretation $\Im$.

1.1 A deductive apparatus for $\mathcal{L}^\times$

Following [24], it is possible to introduce an inferential apparatus for $\mathcal{L}^\times$. To this aim, a collection $\Lambda^\times$ of equality schemes is chosen as *logical axioms*. Figure 1 shows a possible choice for $\Lambda^\times$. Given a collection $\mathtt{E}$ of map equalities, let $\Theta^\times(\mathtt{E})$ be the smallest collection of equalities which both fulfills the inclusion

$$\Lambda^\times \cup \mathtt{E} \cup \{\, P{=}P \ : \text{P is a map expression} \,\} \subseteq \Theta^\times(\mathtt{E})$$

and enjoys the following closure property: *When $P{=}Q$ and $R{=}S$ both belong to $\Theta^\times(\mathtt{E})$, and R occurs in Q and/or in P, then any equality obtainable from $P{=}Q$ by replacement of some occurrences of R by an occurrence of S belongs to $\Theta^\times(\mathtt{E})$.*

An equation $Q{=}R$ is said to be derivable from $\mathtt{E}$, if $Q{=}R$ belongs to $\Theta^\times(\mathtt{E})$.

In principle, this notion of derivability can be mechanized by exploiting any equational theorem prover. An approach of this kind is proposed in [18, 17].

As mentioned, the availability of an inferential machinery for $\mathcal{L}^\times$ calls for cross-translation algorithms between predicate logic and map calculus. This combination enables one to design efficient first-order theorem provers based on a relational core inference-engine. In this frame of mind, [14] and [18] propose a viable instrumentation of equational set-reasoning. The next example emphasizes this point.

Example 1 *In this example we provide an equational proof of a set-theoretical result holding in any (weak) set theory that satisfies the axiom of regularity [25]. This axiom states that:* 'Every nonempty set x contains an element y which is disjoint from x' *and can be formulated in predicative calculus as follows:*

$$(\forall x)(\exists m)\Big((\neg(\exists y)y \in x) \vee \Big(m \in x \wedge \neg(\exists y)(y \in m \wedge y \in x)\Big)\Big).$$

In Section 5 (Example 7) we will obtain this relational translation of the axiom of regularity: $\mathbf{1};(\overline{\mathbf{1};\in \cap \mathbf{1};(\in \cap \overline{\in^{\smile};\in})}) = \mathbf{1}$. *The theorem we want to prove states that:* 'Under regularity, any nonempty transitive set has the empty set as element'. *A set s is said to be transitive if it contains all the members of its members, i.e., if it holds that $(\forall x)((\exists y)(x \in y \wedge y \in s) \rightarrow x \in s)$. Such a property can be rendered in $\mathcal{L}^\times$ by means of the expression: $\iota\cap\overline{\overline{\in^{\smile}}};\in;\in$. Hence, proving the theorem amounts to showing that the inclusion $(\mathbf{1};\in\cap\iota\cap\overline{\overline{\in^{\smile}}};\in;\in) \subseteq (\overline{\mathbf{1};\in};\in)$ belongs to $\Theta^\times(\{\mathbf{1};(\overline{\mathbf{1};\in \cap \mathbf{1};(\in \cap \overline{\in^{\smile};\in})}) = \mathbf{1}\})$.*[3] *A proof of this fact has been reported, for instance, in [18]. Figure 2 lists the chain of equalities leading to such result.* □

1.2 Occurrences

A wfe E *occurs* within another wfe F at *position* ν, where ν is a node in the syntax tree T_F for F, if the subtree of T_F rooted at the node ν is identical to the syntax tree for E. In such a case, we also say that the node ν is an *occurrence* of E (and also an occurrence of the lead symbol of E) in F and that the path from the root of T_F to ν is its *occurrence path*.

An occurrence of E within F can be conveniently coded by a sequence over the set $\mathbb{N}_+$ of all positive integers, representing the positions within its siblings of each node in the occurrence path. Specifically, the set *Pos* of the *positions* in (the syntax tree of) any wfe F can be defined recursively as follows:

[2] The definitions of syntax tree, occurrence, and position adopted in this paper are based on the connectives and quantifiers of predicate logic. Map expressions occurring in formulae of $\mathcal{L}^+$ are regarded as meta-expressions and their internal structure is ignored.

[3] Here we are using $P{\subseteq}Q$ as a shorthand notation for $P{\cap}\overline{Q}{=}\mathbf{0}$.

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
P\cup Q & = & Q\cup P & P;\iota & = & P \\
P\star(Q\star R) & = & (P\star Q)\star R & P^{\smile\smile} & = & P \\
\overline{\overline{P}\cup Q}\cup\overline{\overline{P}\cup\overline{Q}} & = & P & (P\star Q)^{\smile} & = & Q^{\smile}\star P^{\smile} \\
(P\cup Q);R & = & (P;R)\cup(Q;R) & (P^{\smile};\overline{P;Q})\cup\overline{Q} & = & \overline{Q} \\
\mathbf{1} & = & \iota\cup\overline{\iota} & \mathbf{0} & = & \overline{\mathbf{1}} \\
P\cap Q & = & \overline{\overline{P}\cup\overline{Q}} & P\dagger Q & = & \overline{\overline{P};\overline{Q}}
\end{array}
$$

with $\star \in \{\cup, ;\}$

Figure 1: An axiomatic system for $\mathcal{L}^{\times}$

$$
\begin{array}{rl}
\mathbf{1};\in\cap\,\iota\cap\overline{\overline{\in^{\smile}};\in;\in} & = \ \mathbf{1};(\in\cap\overline{\in^{\smile};\in})\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in} \\
& = \ (\mathbf{1};\in\cup\overline{\mathbf{1};\in});(\in\cap\overline{\in^{\smile};\in})\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in} \\
& = \ (\mathbf{1};\in;(\in\cap\overline{\in^{\smile};\in})\cup\overline{\mathbf{1};\in};(\in\cap\overline{\in^{\smile};\in}))\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in} \\
& = \ \mathbf{1};\in;(\in\cap\overline{\in^{\smile};\in})\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in}\cup\overline{\mathbf{1};\in};(\in\cap\overline{\in^{\smile};\in})\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in} \\
& = \ \mathbf{0}\cup\overline{\mathbf{1};\in};(\in\cap\overline{\in^{\smile};\in})\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in} \\
& = \ \overline{\mathbf{1};\in};(\in\cap\overline{\in^{\smile};\in})\cap\iota\cap\overline{\overline{\in^{\smile}};\in;\in} \ \subseteq \ \overline{\mathbf{1};\in};(\in\cap\overline{\in^{\smile};\in}) \ \subseteq \ \overline{\mathbf{1};\in};\in
\end{array}
$$

where $P\subseteq Q$ is a shorthand notation for $P\cap\overline{Q}=\mathbf{0}$

Figure 2: Equational proof for Example 1.

(1) The empty word λ is in $Pos(F)$;

(2) if F is an atomic formula xRy, where $x, y \in Var$, then $Pos(F) = \{\lambda, 1, 2\}$;

(3) if $F = \varphi_1 \wedge \cdots \wedge \varphi_n$ or $F = \varphi_1 \vee \cdots \vee \varphi_n$ and $\pi \in Pos(\varphi_i)$, for some $i \in \{1, \ldots, n\}$, then $i.\pi \in Pos(F)$;

(4) if $F = \neg\psi$, or $F = (\forall x)\psi$, or $F = (\exists x)\psi$ and $\pi \in Pos(\psi)$, then $1.\pi \in Pos(F)$.

Given any wfe F, the occurrences at given positions of subformulae or subterms of F in F are determined as follows. We put $F|_{\lambda} = F$. In case F is an atomic formula xRy, we put $F|_1 = x$ and $F|_2 = y$. If $F = \varphi_1 \circ \cdots \circ \varphi_n$, with $\circ \in \{\wedge, \vee\}$, we put $F|_{i.\pi} = \varphi_i|_{\pi}$, for $i \in \{1, \ldots, n\}$. Finally, if $F = \neg\psi$, or $F = (\forall x)\psi$, or $F = (\exists x)\psi$, we put $F|_{1.\pi} = \psi|_{\pi}$. Thus, the label of a node ν at position π in the syntax tree T_F of a *wfe* F (denoted $lbl(F, \pi)$) is the lead symbol of $F|_{\pi}$.

We indicate by P^F_E the collection of all positions $\pi \in Pos(F)$ such that $F|_{\pi} = E$. If $|P^F_E| = 1$, where $|\cdot|$ denotes the cardinality operator, we may use π^F_E to denote the position of the unique occurrence of E in F. We write P_E and π_E in case F is clear from the context.

It is convenient to establish an ordering $\prec$ over the set $Pos(\varphi)$ of positions in a formula φ such that for any $\pi_1, \pi_2 \in Pos(\varphi)$ and $n_1, n_2 \in \mathbb{N}_+$

- if $\pi_1 = \pi_2 \cdot \eta$ for some sequence η of positive integers, then $\pi_1 \prec \pi_2$;
- if $\pi_1 = \pi \cdot n_1 \cdot \pi'$, $\pi_2 = \pi \cdot n_2 \cdot \pi''$, and $n_1 < n_2$, then $\pi_1 \prec \pi_2$.

Plainly, $\prec$ is a well-ordering. Thus, we can define an operation min that selects from any nonempty set $X \subseteq Pos(\varphi)$ its minimum relative to the ordering $\prec$.

An occurrence ν of a wfe E within a formula F is *positive* if its occurrence path deprived of its last node contains an even number of nodes labeled by the negation symbol $\neg$. Otherwise, the occurrence is said to be *negative*.

Let F be a wfe, π a position in F, and let E be a formula if $F|_{\pi}$ is a formula, and a term otherwise. Also, let $F[\pi/E]$ be the wfe obtained from F by replacing $F|_{\pi}$ at position π by E, so that we have $F[\pi/E]|_{\pi}{=}E$. In case $\pi{=}\eta.n$ is not an element of $Pos(F)$ but $\pi' = \eta.(n{-}1)$ is, $F[\pi/E]$ adds a new subformula E to F and a new position π to $Pos(F)$ (plainly, in this case $lbl(F, \eta)$ must have a variable arity).

Given two wfes E and F, we write $F = F[E]$ to stress that the occurrences of E in F play a significant rôle. Moreover, if E' is another formula, by $F[E/E']$ we denote the wfe resulting from F when *each* occurrence of E in F is replaced by a distinct copy of E'.

Example 2 *Consider for instance the formula* $F = xPy \wedge zQw$. *We have that* $F[2/xRy] = xPy \wedge xRy$. *Position* 3 *does not belong to* $Pos(F)$, *but position* 2 *does (in fact* $F|_2 = zQw$*); thus* $F[3/wSy] = xPy \wedge zQw \wedge wSy$. $\square$

1.3 Uniform notation for formulae and relational expressions

We adopt Smullyan's unifying notation [23] to classify and decompose formulae of $\mathcal{L}^{+}$. Formulae are partitioned into four categories: conjunctive, disjunctive, universal, and existential formulae (called α-, β-, γ-, and δ-formulae, respectively). In particular, δ-formulae are those of the form $(\exists x)\varphi$ and $\neg(\forall x)\varphi$, whereas γ-formulae are those of the form $(\forall x)\varphi$ and $\neg(\exists x)\varphi$. Although either of the connectives $\wedge, \vee$ is clearly redundant in the triad $\neg, \wedge, \vee$, treating the two on a par is a key for streamlining the translation method; especially so in a blended context

α	α_1	$\dots$	α_n
$\varphi_1 \wedge \dots \wedge \varphi_n$	φ_1	$\dots$	φ_n
$\neg(\varphi_1 \vee \dots \vee \varphi_n)$	$\neg\varphi_1$	$\dots$	$\neg\varphi_n$
$\neg\neg\varphi$		φ	

β	β_1	$\dots$	β_2
$\varphi_1 \vee \dots \vee \varphi_n$	φ_1	$\dots$	φ_1
$\neg(\varphi_1 \wedge \dots \wedge \varphi_n)$	$\neg\varphi_1$	$\dots$	$\neg\varphi_n$

Table 1: α-formulae (left), β-formulae (right), and their components.

Conjunctive atoms, α-atoms	$x\alpha y = xR{\cap}Sy$	$x\alpha_1 y = xRy$	$x\alpha_2 y = xSy$	$(\wedge)$
	$x\alpha y = x\overline{R{\cup}S}y$	$x\alpha_1 y = x\overline{R}y$	$x\alpha_2 y = x\overline{S}y$	$(\neg\vee)$
	$x\alpha y = x\overline{\overline{R}}y$	$x\alpha_1 y = xRy$		$(\neg\neg)$
Disjunctive atoms, β-atoms	$x\beta y = xR{\cup}Sy$	$x\beta_1 y = xRy$	$x\beta_2 y = xSy$	$(\vee)$
	$x\beta y = x\overline{R{\cap}S}y$	$x\beta_1 y = x\overline{R}y$	$x\beta_2 y = x\overline{S}y$	$(\neg\wedge)$
Atoms of type δ^α	$x\delta^\alpha y = xR;Sy$	$x\delta_0^{\alpha_1} z = xRz$	$z\delta_0^{\alpha_2} y = zSy$	$(\exists\wedge)$
	$x\delta^\alpha y = x\overline{R\dagger S}y$	$x\delta_0^{\alpha_1} z = x\overline{R}z$	$z\delta_0^{\alpha_2} y = z\overline{S}y$	$(\neg\forall\vee)$
	where z is existentially quantified ($\delta^\alpha \equiv (\exists z)\delta_0^\alpha(z) \equiv (\exists z)(\delta_0^{\alpha_1}(z) \wedge \delta_0^{\alpha_2}(z))$)			
Atoms of type γ^β	$x\gamma^\beta y = x\overline{R;S}y$	$x\gamma_0^{\beta_1} z = x\overline{R}z$	$z\gamma_0^{\beta_2} y = z\overline{S}y$	$(\neg\exists\wedge)$
	$x\gamma^\beta y = xR\dagger Sy$	$x\gamma_0^{\beta_1} z = xRz$	$z\gamma_0^{\beta_2} y = zSy$	$(\forall\vee)$
	where z is universally quantified ($\gamma^\beta \equiv (\forall z)\gamma_0^\beta(z) \equiv (\forall z)(\gamma_0^{\beta_1}(z) \vee \gamma_0^{\beta_2}(z))$)			
Atoms of type κ	$x\kappa y = xR^{\smile}y$	$y\kappa_1 x = yRx$	$x\kappa y = x\overline{R^{\smile}}y$	$y\kappa_1 x = y\overline{R}x$

Table 2: Classification of atomic formulae of $\mathcal{L}^+$.

where the relational constructs of complementation, intersection, and union must be handled too. Analogous uniformity considerations justify our parallel treatment of existential and universal quantifiers on the one hand, and of map composition and Peircean sum $\dagger$ on the other.

Given a δ-formula δ, the notation $\delta_0(x)$ will be used to denote the formula φ, if δ is of the form $(\exists x)\varphi$, or to denote the formula $\neg\varphi$, if δ is of the form $\neg(\forall x)\varphi$. We will refer to $\delta_0(x)$ as *the instance of* δ and to x as *the quantified variable of* δ. Likewise, for any γ-formula γ, $\gamma_0(x)$ denotes the formula φ or $\neg\varphi$, according to whether γ has the form $(\forall x)\varphi$ or $\neg(\exists x)\varphi$, respectively. We allow generalized n-ary α- and β-formulae. To each of them, one can associate its components as shown in Table 1. In general, map expressions, occurring as atomic formulae of $\mathcal{L}^+$, possess an internal structure. For the purpose of representing it with graphs, it is helpful to extend Smullyan's notation to relational constructs. We classify and decompose atomic formulae as shown in Table 2, by exploiting these axiom schemata [24]:

$$(\forall x)(\forall y)(xA{\cup}By \equiv xAy \vee xBy) \qquad (\forall x)(\forall y)(x\overline{A}y \equiv \neg xAy)$$
$$(\forall x)(\forall y)(xA;By \equiv (\exists z)(xAz \wedge zBy)) \qquad (\forall x)(\forall y)(xA^{\smile}y \equiv yAx).$$

Remark 1 In the rest of the paper, without loss of generality, we assume that all formulae are written in *standardized* form. Namely, we assume that all quantifiers are moved inward so as to minimize their scope (but, without rewriting the quantified subformulae). Moreover, we impose that bound variables are renamed so that, for each two positions π_1, π_2, of distinct quantifications $\mathsf{Q}\,\mathtt{x}_{-i}$ and $\mathsf{Q}\,\mathtt{x}_{-j}$ in a formula φ, it holds that $\pi_1 \prec \pi_2$ iff $j > i$. Note that, as a consequence, distinct occurrences of quantifiers in φ always bind different variables (in Var^-).

Example 3 *Considering Remark 1, the formula*

$$(\forall x)(\exists m)\Big((\neg(\exists y)y \in x) \vee \Big(m \in x \wedge \neg(\exists y)(y \in m \wedge y \in x)\Big)\Big)$$

(cf., Example 1) can be rewritten in a standardized *form as follows*

$$(\forall \mathtt{x}_{-2})\Big((\neg((\exists \mathtt{x}_{-4})\,\mathtt{x}_{-4} \in \mathtt{x}_{-2})) \vee (\exists \mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \neg(\exists \mathtt{x}_{-6})(\mathtt{x}_{-6} \in \mathtt{x}_{-5} \wedge \mathtt{x}_{-6} \in \mathtt{x}_{-2}))\Big),$$

where x and m are rewritten as $\mathtt{x}_{-5}$ and $\mathtt{x}_{-2}$, resp., whereas $\mathtt{x}_{-4}$ and $\mathtt{x}_{-6}$ replace the two uses of the variable y. Note, moreover, that the quantification $(\exists\,\mathtt{x}_{-5})$ is moved inward since $\mathtt{x}_{-5}$ does not occur in the first disjunct. □

2 Graphical representation of formulae of $\mathcal{L}^+$

In this section we extend the techniques of [4, 5] for representing map expressions as well as identities of the form $P = \mathbf{1}$ and, more generally, formulae of $\mathcal{L}^+$, by means of directed multigraphs. Our extension calls into play the negation connective $(\neg)$ and the relational complement construct $(\overline{})$, which lie well beyond the scope of the original proposals.

We will make use of (labeled) directed multigraphs allowing multiple edges and self-loops. More specifically, a *directed multigraph* $G = (V, (E, m))$ consists of a set of nodes V, a set of edges $E \subseteq V \times V$, and a *multiplicity function* $m : E \to \mathbb{N} \setminus \{0\}$. Labels are associated with nodes and edges by means of two labeling functions, *lNode* $: V \to X$ and *lEdge* $: E \to Y$, respectively (for some fixed sets X and Y).

Let $G = (V, (E, m))$, $G_1 = (V_1, (E_1, m_1)), \dots, G_n = (V_n, (E_n, m_n))$ be $n + 1$ directed multigraphs such that $V_1, \dots, V_n$ are nonempty, $V_1 \cup \dots \cup V_n = V$, $V_i \cap V_j = \emptyset$ when $i \neq j$, $E_1 \cup \dots \cup E_n = E$, and $m((u, v)) = m_i((u, v))$ iff $(u, v) \in E_i$. Then $\mathcal{S} = \{G_1, \dots, G_n\}$ is said to be a *partition of* G, and $G_1, \dots, G_n$ are said to be the *components* of G. In case none of the components G_i admits a partition other than itself, $\mathcal{S}$ is said to be the *most refined partition of G into components*, and $G_1, \ldots, G_n$ are said to be the *most refined components* of G.

Let φ be a formula of $\mathcal{L}^+$. We can assume φ to be constructed out of atomic formulae of the form xPy. In fact, any equality atom $Q = R$ can be rewritten as $x\mathbf{1};((Q \cup R) \cap \overline{\overline{Q} \cap \overline{R}});\mathbf{1}y$. We say that φ is represented by the labeled directed multigraph $G_\varphi = (V_\varphi, (E_\varphi, m_\varphi))$ if the following conditions hold:

(1) The labeling function $\mathit{lNode} : V_\varphi \rightarrow (\mathrm{pow}(\mathit{Var}(\varphi)) \cup \{\{x\} : x \in \mathit{Var}^-\})$ associates sets of variables of φ with nodes of G_φ. In particular, the nodes $v \in V_\varphi$ such that $\mathit{lNode}(v) \cap \mathit{Var}^- \neq \emptyset$ are called the *bound nodes* of G_φ, and the nodes $v \in V_\varphi$ such that $\mathit{lNode}(v) \subseteq \mathit{Var}^+$ are called the *free nodes* of G_φ.

(2) The labeling function $\mathit{lEdge} : E_\varphi \rightarrow (\mathrm{pow}(\mathcal{L}^\times) \cup \{\{\psi\} : \psi \in \mathcal{L}^+\})$ associates with each edge either a set of map expressions or a singleton containing a disjunctive formula of $\mathcal{L}^+$ devoid of quantifiers. For each edge $(u, v) \in E_\varphi$, its multiplicity is $m_\varphi((u, v)) =_{\mathrm{Def}} |\mathit{lEdge}((u, v))|$.

(3) The multigraph is endowed with a relation $\rightsquigarrow$ induced over the most refined components of G_φ such that G_i is said to be *in relation $\rightsquigarrow$ with* G_j, and we write $G_i \rightsquigarrow G_j$, if there is an edge (u, u') of G_i and two vertices v, v' in V_j such that $\mathit{lNode}(v) \subseteq \mathit{lNode}(u)$, $\mathit{lNode}(v') \subseteq \mathit{lNode}(u')$, and either $\mathit{lEdge}((u, u'))$ contains a β-atom (resp., γ^β-atom) ψ and G_j represents a formula logically equivalent to the complement of ψ, or $\mathit{lEdge}((u, u'))$ contains a β-formula with a component ψ such that G_j represents a formula equivalent to the negation of ψ.[4] In such cases, we also say that ψ is in relation $\rightsquigarrow$ with G_j. The most refined components, G_j, of G_φ such that there is no G_i in relation $\rightsquigarrow$ with G_j are called *top components* of G_φ.

(4) Each most refined component G_i is annotated with a variable sign_{G_i} assuming either value '+' or '−'. If $G_i \rightsquigarrow G_j$, $\mathsf{sign}_{G_j} = \mathsf{opp}(\mathsf{sign}_{G_i})$, where opp is such that $\mathsf{opp}(+) = -$ and $\mathsf{opp}(-) = +$. Every top component G_i has $\mathsf{sign}_{G_i} = +$.

(5) In a graph G_φ every most refined component represents either a subformula of φ or its negation. Let χ be a subformula of φ occurring positively (resp., negatively) in φ, and let ξ be the subformula obtained from χ by dropping the quantifiers in χ (for instance, for $\chi = (\forall \mathtt{x}_{-2})(\mathtt{x}_1 R \mathtt{x}_{-2} \vee \mathtt{x}_{-2} S \mathtt{x}_2)$, we have $\xi = (\mathtt{x}_1 R \mathtt{x}_{-2} \vee \mathtt{x}_{-2} S \mathtt{x}_2)$). Then, χ is represented in G_φ as follows:

   – If ξ is an atomic formula xPy, G_χ consists of an edge (u, v) with $\mathit{lNode}(u) = \{x\}$, $\mathit{lNode}(v) = \{y\}$, and $\mathit{lEdge}((u, v)) = \{P\}$ (resp., $\mathit{lEdge}((u, v)) = \{\overline{P}\}$), if $\mathsf{sign}_{G_\chi} = +$. Otherwise, $\mathit{lEdge}((u, v)) = \{\overline{P}\}$ (resp., $\mathit{lEdge}((u, v)) = \{P\}$).

   – If ξ is an α-formula (resp., β-formula), G_χ is a multigraph having as components the multigraphs representing the components of ξ, if their sign is '+'. Otherwise, G_χ is a graph with only one edge (u, v), such that $\mathit{lEdge}((u, v)) = \{\neg\xi\}$, $\mathit{lNode}(u)$ is the set of the left variables in the atomic formulae of ξ, and $\mathit{lNode}(v)$ is the set of the right variables in the atomic formulae of ξ.

   – If ξ is a β-formula (resp., α-formula), G_χ consists of just one edge (u, v) such that $\mathit{lEdge}((u, v)) = \{\xi\}$, $\mathit{lNode}(u)$ is the set of the left variables in the atomic formulae of ξ, and $\mathit{lNode}(v)$ is the set of the right variables in the atomic formulae of ξ, if $\mathsf{sign}_{G_\chi} = +$; otherwise G_χ is a multigraph having as components the multigraphs representing the components of $\neg\xi$.

(6) Let χ be a subformula of φ. If G_χ has as components the multigraphs representing χ or its negation, and $\mathsf{sign}_{G_\chi} = +$ (resp., $\mathsf{sign}_{G_\chi} = -$), nodes labeled with the same singleton set $\{z\}$ can be identified, provided they are free nodes or when z is an existential (resp., universal) variable. Moreover, if there is a component of G_χ, G_i, representing a β-atom (resp., γ^β-atom) ψ, and a component G_j representing a formula ψ' equivalent to ψ (obtained from ψ by applying the axioms of Section 1.3) and ψ, ψ' occur in χ in the same conjunction, G_j can be removed from G_χ and the component $\tilde{G}_j$ representing the complement of ψ' is introduced by putting $G_i \rightsquigarrow \tilde{G}_j$ and $\mathsf{sign}_{\tilde{G}_j} = \mathsf{opp}(\mathsf{sign}_{G_i})$.

A multigraph G_ψ can be identified with the formula ψ it represents. For this reason, its meaning is defined to be $(\psi)^{\Im,\mathbf{a}}$. Two representation graphs G_φ and G_ψ are said to be *equivalent* if they have the same meaning for every interpretation $\Im$ and for every variable assignment $\mathbf{a}$. In case the formula φ is atomic, say xPy, with a slight abuse of notation we say that G_φ represents P.

A multigraph G_φ, representing a formula φ, is in *simple form* if (a) each of its edges is labeled with either a β-atom, or a γ^β-atom, or a β-formula, or a map letter, or the complement of a map letter, (b) every component of G_φ representing a β-formula is in relation $\rightsquigarrow$ with the components representing the complement of each of its disjuncts, and (c) every component of G_φ representing a β-atom or a γ^β-atom is in relation $\rightsquigarrow$ with the component representing its complement.

Example 4 *Let $\varphi_1 = x\overline{P} \cap Qy$. G_{φ_1} is the directed multigraph depicted in Figure 3 having $V_{\varphi_1} = \{u_0, v_0\}$, $E_{\varphi_1} = \{(u_0, v_0)\}$, $m_{\varphi_1}((u_0, v_0)) = 1$, $\mathit{lEdge}((u_0, v_0)) = \{\overline{P} \cap Q\}$, $\mathit{lNode}(u_0) = \{x\}$ and $\mathit{lNode}(v_0) = \{y\}$.*

[4] The intuition behind the relation $\rightsquigarrow$ is that an edge labeled with either a β-formula, or a β-atom, or a γ^β-atom 'calls' other graph components which, taken together, represent the dual of the formula. The decomposition is done according to Table 1.

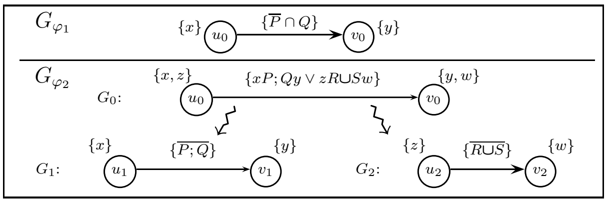


Figure 3: The multigraphs G_{φ_1} and G_{φ_2} associated with $\varphi_1 = x\overline{P} \cap Qy$ and $\varphi_2 = xP;Qy \vee zR\cup Sw$, resp. (see Example 4).

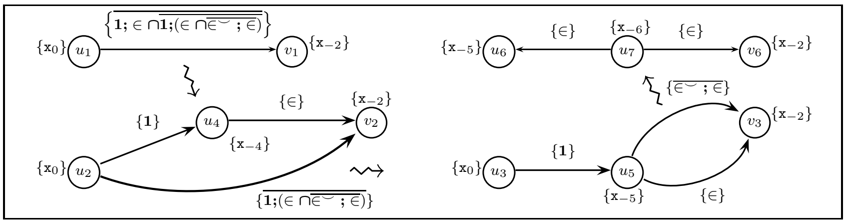


Figure 4: Graphical representation for Example 4.

Let $\varphi_2 = xP;Qy \vee zR\cup Sw$. G_{φ_2} is the multigraph with components G_0, G_1, and G_2 shown in Figure 3. The component G_0 is in relation $\rightsquigarrow$ with both G_1 and G_2. In particular, $lNode(u_0) = \{x, z\}$ includes both $lNode(u_1) = \{x\}$ and $lNode(u_2) = \{z\}$. $lNode(v_0) = \{y, w\}$ includes both $lNode(v_1) = \{y\}$ and $lNode(v_2) = \{w\}$.

Figure 4 shows a representation of the atom $\mathtt{x}_0\mathbf{1}; \in \cap\overline{\mathbf{1};(\in \cap\overline{\overline{\in^{\smile};\in}})}\mathtt{x}_{-2}$. Each complemented subexpression is in relation $\rightsquigarrow$ with a component of the multigraph. (For clarity we split the multiple edge between nodes u_5 and u_3 into two arcs.) □

3 Graph-transformation rules

In [5] some graph-transformation rules have been introduced which are meaning preserving, in the sense that they transform a given multigraph G_ψ into a multigraph $G_{\psi'}$ such that ψ' is equivalent to ψ. Let us adapt them to the more general context we are analyzing in this paper:

(1) If u and v belong to the same more refined component, $\mathbf{1}$ can be added to or removed from $lEdge((u, v))$.

(2) An edge (u, v) with $lEdge((u, v)) = \{P\}$ can be replaced by an edge (v, u) with $lEdge((v, u)) = \{Q\}$, if either $P \equiv Q^{\smile}$, or $Q \equiv P^{\smile}$, or $P \equiv Q \equiv \iota$.

(3) If there are two expressions $\alpha_1, \alpha_2 \in lEdge((u, v))$, for two nodes u and v, it is possible to replace them by a single relational expression $\alpha \equiv \alpha_1 \cap \alpha_2$, i.e., $lEdge((u, v))$ can be updated to $(lEdge((u, v)) \setminus \{\alpha_1, \alpha_2\}) \cup \{\alpha\}$. Analogously, any relational expression $\overline{\overline{P}} \in lEdge((u, v))$ can be replaced by the expression P, i.e., $lEdge((u, v))$ can be updated to $(lEdge((u, v)) \setminus \{\overline{\overline{P}}\}) \cup \{P\}$. In both cases the converse replacement is also possible.

(4) If (u, v) is an edge of a most refined component G_i labeled with a β-atom or a γ^β-atom, say xPy, and Q is equivalent to $\overline{P}$, a component G_j representing xQy such that $G_i \rightsquigarrow G_j$ can be added or removed.

(5) If (u, v) is an edge of a most refined component with sign '+' (resp., '−') such that $lEdge((u, v)) \supseteq \{\delta^\alpha\}$, a new bound node s labeled with a new existentially (resp., universally) quantified variable can be introduced together with two edges (u, s) and (s, v) such that $lEdge((u, s)) = \{\delta_1^\alpha\}$ and $lEdge((s, v)) = \{\delta_2^\alpha\}$. Then, $lEdge((u, v))$ is updated to $lEdge((u, v)) \setminus \{\delta^\alpha\}$. Conversely, let (u, s), (s, v) be the only edges of a most refined component of sign '+' (resp., '−') involving the node s, and such that $lEdge((u, s)) = \{\delta_1^\alpha\}$ and $lEdge((s, v)) = \{\delta_2^\alpha\}$. If s is a bound node labeled with a singleton $\{z\}$ and z is an existential (resp., universal) variable, these edges can be removed and $lEdge((u, v))$ is updated to $lEdge((u, v)) \cup \{\delta^\alpha\}$.

(6) Let $lNode(v)$ and $lNode(u)$ be singletons. If $lEdge((u, v)) = \{\iota\}$, where either $lNode(v) = lNode(u)$, or any of v and u is a bound node of degree 1, the edge (u, v) can be removed. If either $lNode(v) = lNode(u)$, or any of v and u is a new bound node, the edge (u, v) can introduced with $lEdge((u, v)) = \{\iota\}$.

(7) An isolated node may be removed.

The rules just presented are applied to define the tactics used in the graph-fattening algorithm of Section 4 which constructs a multigraph in simple form representing the internal structure of a map expression, and in the graph-thinning algorithm of Section 5 which tries to construct a quantifier-free formula equivalent to the input formula together with its representation graph.

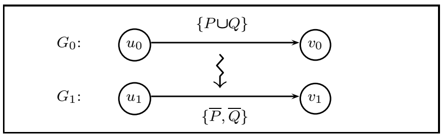


Figure 5: Result of the graph-fattening algorithm applied to $P \cup Q$.

Correctness of the rules (1), (3), (5), (6), and (7) can be verified as in [5], and rule (2) can be proved correct by applying an axiom of Section 1.3. To prove correctness of rule (4), assume G_i represents $\varphi = \psi[xR \cup Sy]$ (resp., $\psi[xR \dagger Sy]$). Then $G_i \rightsquigarrow G_j$ represents $\varphi' = \psi[xR \cup Sy \wedge (xRy \vee xSy)]$ (resp., $\psi[xR \dagger Sy \wedge (\forall z)(xRz \vee zSy)]$). By the axioms of Section 1.3, φ and φ' are logically equivalent, so that G_i and $G_i \rightsquigarrow G_j$ are equivalent. Hence rule (4) is meaning preserving.

4 The graph-fattening algorithm

We now present an algorithm that allows one to graphically represent the internal structure of a map expression P (i.e., of a formula xPy of $\mathcal{L}^+$). This extends the one presented in [5], unable to deal with map complementation and map union.

Let us consider the map expression P as an atomic formula xPy of $\mathcal{L}^+$. The multigraph G representing xPy has only one edge (s_0, s_1) with $lEdge((s_0, s_1)) = \{P\}$. The nodes s_0 and s_1, called *source* and *sink*, respectively, represent in G the two arguments of xPy (hence, we have $lNode(s_0) = \{x\}$ and $lNode(s_1) = \{y\}$).

The graph-fattening algorithm takes as input G and proceeds nondeterministically and recursively by selecting one of the tactics listed below. After a finite number of steps, the input graph G is transformed into an equivalent, maximally expanded multigraph (in simple form) which unwinds the internal structure of P. Equivalence between the input and the output graph can be checked by noticing that all the tactics considered are derived from the graph transformation rules of Section 3 and, therefore, they are meaning preserving.

The construction can proceed recursively, by obtaining the subgraph representing an expression as combination of the graphs representing its subexpressions. In this way one starts by generating the components relative to each map letter. Then the tactics (2)–(5) are applied until all constructs in the given expression have been considered.

(1) G consists of a single edge from s_0 to s_1, such that $lEdge((s_0, s_1)) = \{P\}$;

(2) P is of type κ and G, s_1, s_0 (with source and sink exchanged) represents κ_1;

(3) P is of type δ^α, the disjoint graphs G', s_0, s_2' and G'', s_2'', s_1 represent $\delta_0^{\alpha_1}$ and $\delta_0^{\alpha_2}$, respectively. Then G, s_0, s_1 is obtained from G' and G'' by 'gluing' together s_2' and s_2'' to form a single node; if $\mathsf{sign}_G = +$ (resp., $\mathsf{sign}_G = -$), s_2' is labeled with a singleton containing an existential (resp., universal) variable;

(4) P is of type α, the disjoint graphs G', s_0', s_1' and G'', s_0'', s_1'' represent α_1 and α_2, respectively. Then G is obtained from G' and G'' by gluing s_0'' to s_0' to form s_0, and s_1'' to s_1' to form s_1. In case $P = \overline{\overline{\alpha_1}}$, G, s_0, s_1 coincide with G', s_0', s_1', representing α_1;

(5) P is either of type β or γ^β, the graph G', s_0, s_1 consists of a single edge from s_0 to s_1 such that $lEdge((s_0, s_1)) = \{P\}$, the graph G'', s_0'', s_1'' represents an atomic formula of type α or δ^α whose complement is equal to P, and $G' \rightsquigarrow G''$. Then G, s_0, s_1, representing P, is the graph with components G' and G''.

Example 5 *The multigraph resulting from the application of the graph-fattening algorithm to the map expression $P \cup Q$ is shown in Figure 5.*

Figure 4 shows the maximally expanded multigraph produced by the algorithm for the atomic formula

$$\mathtt{x}_0\overline{\mathbf{1};\in\cap\overline{\mathbf{1};(\in\cap\overline{\in^{\smile}\,;\in})}}\mathtt{x}_{-2}.$$

(For clarity, the multiple edge between nodes u_5 and u_3 is split into two distinct arcs.) □

Q_1	Q_2	Translation
$\exists$	$\exists$	$\mathbf{1};R;\mathbf{1} = \mathbf{1}$
$\exists$	$\forall$	$\mathbf{1};R = \mathbf{1}$

Q_1	Q_2	Translation
$\forall$	$\exists$	$R;\mathbf{1} = \mathbf{1}$
$\forall$	$\forall$	$R = \mathbf{1}$

Table 3: Relational translation of $(\mathsf{Q}_1\, x)(\mathsf{Q}_2\, y)(xRy)$, depending on the quantifiers Q_1 and Q_2.

5 The graph-thinning algorithm

Our aim in what follows is to determine, out of a given formula φ of $\mathcal{L}^+$, an equivalent quantifier-free formula ψ. In case $\psi = xRy$, where $x, y \in Var^+$, we say that R is a map translation of φ. If x, y are both in Var^-, depending on the quantifiers bounding x, y, we can obtain an equality equivalent to φ as shown in Table 3. A translation can be obtained also when just only one among x, y is free. For instance, $(\exists x)xRy$ can be rendered as $z\mathbf{1};Ry$, where z is a new free variable.

A simpler version of this problem has been analyzed and solved in [5], by designing an algorithm (called *graph-thinning algorithm*) that seeks a quantifier-free formula of $\mathcal{L}^+$ equivalent to a given existentially quantified conjunction φ of literals of the form xPy. According to its original specification, such an algorithm contains as a preliminary step the construction of a labeled multigraph G_φ, its normalization (i.e., elimination of loop edges), the fusion of multiple edges, and the application, up to stabilization, of two rules named bypass and bigamy.

The extension of the graph-thinning algorithm we introduce in this paper does no longer resort to a preliminary construction of G_φ. Instead, it transforms the input formula φ directly, by operating on its positions with the purpose of deriving, at the same time, both ψ and G_ψ. As said, the output formula ψ is required to be of the form xRy. Note that our algorithm may fail in achieving its goal. This does not mean that φ does not admit a quantifier-free translation in $\mathcal{L}^+$, it simply witnesses the incompleteness of the algorithm in solving a problem which is, in fact, undecidable [22, 24].

Let us denote by $\mathcal{P}$ the set of all the positions that one must analyze to derive the formula ψ and to construct the graph G_ψ. $\mathcal{P}$ is initially set equal to $Pos(\varphi)$, the set of all the positions in φ. We construct a sequence of formulae $\psi^{(0)}, \psi^{(1)}, \ldots$ and a sequence of multigraphs

$$G_\psi^{(0)} = (V_\psi^{(0)}, (E_\psi^{(0)}, m_\psi^{(0)})),\ G_\psi^{(1)} = (V_\psi^{(1)}, (E_\psi^{(1)}, m_\psi^{(1)})),\ \ldots$$

such that for a certain index $k \in \mathbb{N}$, $\psi^{(k)} = \psi$ and $G_\psi^{(k)} = G_\psi$. To construct the two sequences we start by putting $\psi^{(0)} = \varphi$, and by defining $G_\psi^{(0)}$ as the graph having $V_\psi^{(0)} = E_\psi^{(0)} = \{\}$. Then, $\psi^{(i+1)}$ is obtained from $\psi^{(i)}$, and $G_\psi^{(i+1)}$ is obtained from $G_\psi^{(i)}$, by extracting the $\prec$-minimal position n left in $\mathcal{P}$ and performing one of the following transformations, depending on the form of $\psi^{(i)}|_n$, for $i \in \{0, 1, \ldots\}$.

(1) If $\psi^{(i)}|_n$ is a variable, let $\psi^{(i+1)} = \psi^{(i)}$ and $G_\psi^{(i+1)} = G_\psi^{(i)}$.

(2) If $\psi^{(i)}|_n$ is an atomic formula xRy, let $\psi^{(i+1)} = \psi^{(i)}$. Then, construct a graph $G_n = (E_n, V_n)$, with distinguished nodes u_n, v_n, $E_n = \{(u_n, v_n)\}$, and $V_n = \{u_n, v_n\}$. Put $lNode(u_n) = \{\psi^{(i)}|_{n.1}\} = \{x\}$, $lNode(v_n) = \{\psi^{(i)}|_{n.2}\} = \{y\}$, and $lEdge((u_n, v_n)) = \{R\}$. If $\psi^{(i)}|_n$ occurs positively (resp., negatively) in $\psi^{(i)}$, put sign_{G_n}=+ (resp., sign_{G_n}=−). $G_\psi^{(i+1)}$ is obtained from $G_\psi^{(i)}$ by introducing in $G_\psi^{(i)}$ the component G_n, that is $V_\psi^{(i+1)} = V_\psi^{(i)} \cup V_n$ and $E_\psi^{(i+1)} = E_\psi^{(i)} \cup E_n$.

(3) If $\psi^{(i)}|_n = (\exists x)\chi$ and x does not occur in χ, then we put $\psi^{(i)}|_n = \psi^{(i)}|_{n.1}$ and $\psi^{(i+1)} = \psi^{(i)}$. Otherwise, if x occurs only once in χ, and it occurs in a subformula of the form xRw (resp., wRx), then we replace in χ such subformula with $\mathtt{x}_0\mathbf{1};Rw$ (resp., $wR;\mathbf{1}\mathtt{x}_0$). In $G_{n.1}$, the expression R labelling the edge is replaced by $\mathbf{1};R$ (resp., $R;\mathbf{1}$) whereas $\mathtt{x}_0$ replaces x in the node label. Finally, we put $\psi^{(i)}|_n = \psi^{(i)}|_{n.1}$ and $\psi^{(i+1)} = \psi^{(i)}$. In both cases, the multigraph $G_\psi^{(i+1)}$ results from $G_\psi^{(i)}$ by calling $Rename(G_{n.1}, n)$, which renames the component $G_{n.1}$ as G_n. (The procedure *Rename* is listed in Section A.1.)

(4) Let $\psi^{(i)}|_n = \chi_1 \wedge \ldots \wedge \chi_k$, where $k > 1$. Each χ_i is either an atomic formula, or an α-formula, or a β-formula. $\psi^{(i+1)}$ and $G_\psi^{(i+1)}$ are obtained by executing the following steps.

Step 1: The first step eliminates from $\psi^{(i)}|_n$ every χ_j which is an α-formula by attaching all the components of χ_j directly to $\psi^{(i)}|_n$. This operation is called *collapse* of χ_j in $\psi^{(i)}|_n$ (its implementation is described in the App. A). After the application of this step, the components of $\psi^{(i)}|_n$, $\chi_1, \ldots, \chi_m$, can be either atomic formulae or β-formulae. Because of the ordering $\prec$ defined on $\mathcal{P}$, the corresponding graphs $G_{n.1}, \ldots G_{n.m}$ have already been constructed. Thus, G_n is the graph having $G_{n.1}, \ldots, G_{n.m}$ as most refined components. For an example of collapse step, see Figure 7 and Example 6.

Step 2: The *normalization* step eliminates from $\psi^{(i)}|_n$ atomic formulae with identical left and right arguments, and removes the corresponding 'loop edges' from G_n. (The details of the normalization step are described in Section A.3, App. A). A graphical illustration of the normalization step for the formula $\mathtt{x}_1R\mathtt{x}_2 \wedge \mathtt{x}_1Q\mathtt{x}_1$ is given in Figure 6 (see also Example 6).

Step 3: The *fusion* operation replaces in $\psi^{(i)}|_n$ occurrences of atomic formulae involving the same variables (as xRy and xSy, for instance) with a single atomic formula having as map expression the intersection of the map expressions of the considered formulae (i.e., $xR\cap Sy$). This, in G_n, amounts to replacing single edge components $G_{n.j}, G_{n.k}$, labeled with a map expression and such that $lNode(u_{n.j}) \cup lNode(v_{n.j}) = lNode(u_{n.k}) \cup lNode(v_{n.k})$, with a unique component whose edge is labeled with the relational intersection of the map expressions labeling their edges. (More details on the fusion step are given in Section A.4, App. A.)

Step 4: The *bypass* and *bigamy* rules, and then the normalization and fusion operations, are repeatedly applied to $\psi^{(i)}|_n$ and G_n till stability is reached, that is, until $\psi^{(i)}|_n$ and the graph G_n cannot be modified anymore. Then $\psi^{(i+1)} = \psi^{(i)}$ and $G_\psi^{(i+1)} = G_\psi^{(i)}$. Let us briefly introduce the bypass and the bigamy rules (see Sections A.5 and A.6 of App. A for the corresponding pseudo-code). Let $\psi^{(i)}|_n$ occur positively (resp., negatively) in $\psi^{(i)}$. The bypass rule can be applied to any two atomic formulae of $\psi^{(i)}|_n$, say

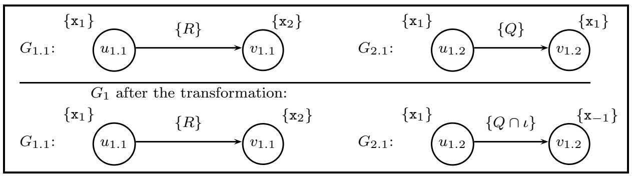


Figure 6: An application of the normalization step.

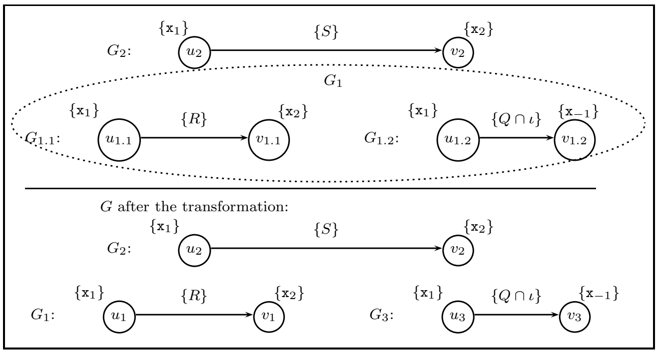


Figure 7: A description of an application of the collapse step.

xRz and zSy, that share an existential (resp., universal) variable z that does not occur elsewhere in $\psi^{(i)}$, in case no universal (resp., existential) quantifier is in the scope of the quantifier binding z. If the preconditions of the rule are satisfied, the two formulae are replaced by an atomic formula $xR;Sy$. The multigraph G_n is modified accordingly, by eliminating the components representing xRz and zSy, and introducing a component representing $xR;Sy$. An example of application of the bypass rule is illustrated in Figure 9.

The bigamy rule can be applied if there is a quantified variable x occurring only once in $\psi^{(i)}$, and specifically in $\psi^{(i)}|_n$. If x is existentially quantified and occurs in $\psi^{(i)}|_n$ as the left (right) argument of an atomic formula, say xRy (resp., yRx), if there is in $\psi^{(i)}|_n$ an atomic formula wSy or ySw, and if no universal quantifier lies in the scope of the quantifier of x, then $w\mathbf{1}x$ (resp., $x\mathbf{1}w$) can be added to $\psi^{(i)}|_n$. The multigraph G_n is modified accordingly by introducing a component representing the new atomic formula. On the other hand, if x is universally quantified, the occurrence of xRy (resp., yRx) in $\psi^{(i)}|_n$ is simply replaced by $w\mathbf{0} \dagger Ry$ (resp., $y\mathbf{0} \dagger Rw$). Let k be the position of the conjunct of $\psi^{(i)}|_n$ we are considering, then the labels of the left (right) node and of the edge of the component $G_{n.k}$ are modified accordingly. An example of application of the bigamy rule is shown in Figure 10. Note that the bigamy rule may cause (in the existential case) a growth of the formula at hand. This growth, however, enables a subsequent application of the bypass rule (involving the variable x) and of the fusion step.

(5) If $\psi^{(i)}|_n = \neg\chi$, we can distinguish the following cases:

(a) If χ is an atomic formula xRy, the negation in front of χ is removed, R is complemented, $\psi^{(i)}|_n$ is set equal to $x\overline{R}y$, and $\psi^{(i+1)} = \psi^{(i)}$. G_n is a single edge multigraph such that $E_n = \{(u_n, v_n)\}$, $V_n = \{u_n, v_n\}$, $lNode(u_n) = \{x\}$, $lNode(v_n) = \{y\}$, $lEdge((u_n, v_n)) = \{\overline{R}\}$, and $\mathsf{sign}_{G_n} = \mathsf{opp}(\mathsf{sign}_{G_{n \cdot 1}})$. Finally, $G_\psi^{(i+1)} = (G^{(i)} \setminus \{G_{n.1}\}) \cup \{G_n\}$.

(b) If χ is a conjunction, $\psi^{(i+1)} = \psi^{(i)}$ and $G_\psi^{(i+1)} = G_\psi^{(i)} \cup \{G_n\}$, where G_n is obtained as follows. $V_n = \{u_n, v_n\}$ and $E_n = \{(u_n, v_n)\}$, u_n is labeled with the set of left variables of the atomic formulae in χ, v_n is labeled with the set of right variables of the atomic formulae in χ, $lEdge((u_n, v_n)) = \{\neg(\chi)\}$. Each multigraph $G_{n.1.i}$, representing a component χ_i of χ, is renamed as $G_{n.i}$ (see procedure *Rename* in Section A.1, App. A). Then G_n is put in relation $\rightsquigarrow$ with all of them and $\mathsf{sign}_{G_n} = \mathsf{opp}(\mathsf{sign}_{G_{n \cdot i}})$.

(c) Double negation and complementation are treated in the usual way, simplifying the formula χ and the multigraph representing it.

(6) If $\psi^{(i)}|_n = \chi_1 \vee \ldots \vee \chi_k$, we put $\psi^{(i)}|_n = \neg(\neg\chi_1 \wedge \ldots \wedge \neg\chi_k)$ and $\mathcal{P} = \mathcal{P} \cup \{n, n.1, n.1.1, \ldots n.1.k\}$. Moreover,

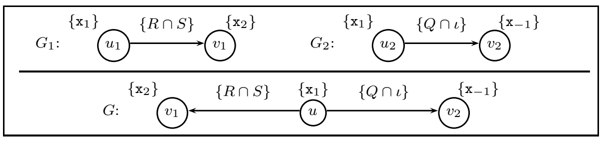


Figure 8: Two alternatives for the multigraph $G_\psi^{(7)}$, both representing the formula $\psi^{(7)} = \mathtt{x}_1(R \cap S)\mathtt{x}_2 \wedge \mathtt{x}_1(Q \cap \iota)\mathtt{x}_{-1}$.

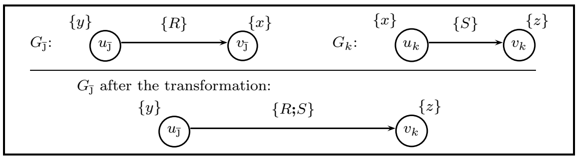


Figure 9: An application of the bypass rule.

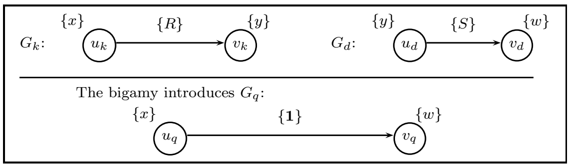


Figure 10: An application of the bigamy rule.

we apply the procedure *Rename* to each graph $G_{n.i}$ obtaining $G_{n.1.i.1}$, for $i \in \{1,\dots,k\}$. Now the minimal position to be processed is $n.1.1$ (a negation, thus falling in the previous cases).

(7) If $\psi^{(i)}|_n = (\forall x)\chi$, we put $\psi^{(i)}|_n = \neg(\exists x)\neg\chi$ and $\mathcal{P} = \mathcal{P} \cup \{n, n.1, n.1.1\}$. The minimal position to be processed is $m = n.1.1$, $\psi^{(i)}|_m$ is a negation of a formula and therefore it is treated as outlined in case (5).

Example 6 *To better illustrate the steps introduced in item (4), let us consider the formula $\varphi = (\mathtt{x}_1 R \mathtt{x}_2 \wedge \mathtt{x}_1 Q \mathtt{x}_1) \wedge \mathtt{x}_1 S \mathtt{x}_2$. If we disregard the positions of the variables in φ, the set of positions of φ is $\{\lambda, 1, 2, 1.1, 1.2\}$. According to the ordering $\prec$, after the construction of the components $G_{1.1}$ and $G_{1.2}$ relative to the subformulae $\varphi|_{1.1} = \mathtt{x}_1 R \mathtt{x}_2$ and $\varphi|_{1.2} = \mathtt{x}_1 Q \mathtt{x}_1$ (item 2 of the graph-thinning algorithm), the subformula $\varphi|_1 = \mathtt{x}_1 R \mathtt{x}_2 \wedge \mathtt{x}_1 Q \mathtt{x}_1$ is analyzed and the normalization step is applied to $\mathtt{x}_1 Q \mathtt{x}_1$. As a result we have $\psi^{(4)}|_1 = \mathtt{x}_1 R \mathtt{x}_2 \wedge \mathtt{x}_1 (Q \cap \iota) \mathtt{x}_{-1}$, where $\mathtt{x}_{-1}$ is a new existentially quantified variable (cf., the function newOddVar in the procedure Normalize in Section A.3) and the corresponding component G_1 is a multigraph constituted of two components, $G_{1.1}$ and $G_{1.2}$, as shown in Figure 6. After the construction of the graph representing $\psi^{(4)}|_2 = \mathtt{x}_1 S \mathtt{x}_2$, the formula $\psi^{(5)}|_\lambda = \psi^{(5)} = (\mathtt{x}_1 R \mathtt{x}_2 \wedge \mathtt{x}_1 (Q \cap \iota) \mathtt{x}_{-1}) \wedge \mathtt{x}_1 S \mathtt{x}_2$ is analyzed. Through an application of the collapse step, $\psi^{(6)} = \mathtt{x}_1 R \mathtt{x}_2 \wedge \mathtt{x}_1 S \mathtt{x}_2 \wedge \mathtt{x}_1 (Q \cap \iota) \mathtt{x}_{-1}$, $G_{1.1}$ and $G_{1.2}$ are renamed to G_1 and G_3, respectively. $G_\psi^{(6)}$ is a multigraph with components G_1, G_2, and G_3. A graphical description of the collapse step is given in Figure 7. Next, the application of the Fusion procedure to $\psi^{(6)}$ yields the formula $\psi^{(7)} = \mathtt{x}_1 (R \cap S) \mathtt{x}_2 \wedge \mathtt{x}_1 (Q \cap \iota) \mathtt{x}_{-1}$ and the multigraph $G_\psi^{(7)}$ with components G_1 and G_2 shown in Figure 8. Notice that, the two nodes u_1 and u_2 can be merged together, so as to form a single node u labeled by $\{x\}$. This allows one to obtain an alternative graph representation of $\psi^{(7)}$. Such a multigraph G is shown in Figure 8.* □

Example 7 *We illustrate now a complete execution of the graph-thinning algorithm applied to the first-order formulation ψ of the regularity axiom [25]*

$$\begin{aligned}\psi \quad = \quad & (\forall \mathtt{x}_{-2})\Big((\neg((\exists \mathtt{x}_{-4})\, \mathtt{x}_{-4} \in \mathtt{x}_{-2})) \vee \\ & (\exists \mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \neg(\exists \mathtt{x}_{-6})(\mathtt{x}_{-6} \in \mathtt{x}_{-5} \wedge \mathtt{x}_{-6} \in \mathtt{x}_{-2}))\Big),\end{aligned}$$

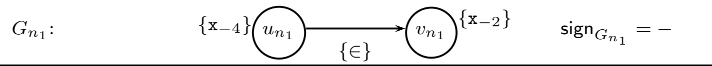
where we renamed the bound variables as described in Section 1 (cf., Remark 1 and Examples 1 and 3). For simplicity, let us ignore the applications of rule (1) of the thinning algorithm. Hence, except for variables positions, the $\prec$-minimal position to be considered is $n_1 = 1.1.1.1$, corresponding to the subformula $\mathtt{x}_{-4} \in \mathtt{x}_{-2}$. By rule (2) we obtain $\psi^{(1)} = \psi$ and $G_\psi^{(1)} = G_{n_1}$, as depicted in Figure 11.

The complete sequence of transformations performed by the thinning algorithm are reported in Figures 11 and 12: for the i-th step of the algorithm we indicate the analyzed position n_i, the corresponding subformula $\psi|_{n_i}$, the rule applied by the algorithm, and the resulting $\psi^{(i)}$ and $G_\psi^{(i)}$. The sequence of steps yields the desired translation: $(\forall\, \mathtt{x}_{-2})\,(\mathtt{x}_0 \overline{\mathbf{1}; \in \cap \overline{\mathbf{1};(\in \cap \overline{\in^{\smile}; \in})}}\, \mathtt{x}_{-2})$. □

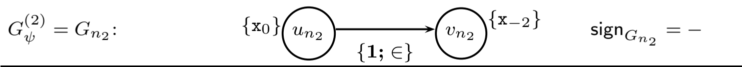
5.1 Termination and correctness of the graph-thinning algorithm

Termination of the graph-thinning algorithm can be checked by making the following considerations. The number of subformulae analyzed and the number of intermediate multigraphs constructed to produce the final formula and multigraph are finite. This is so because the set of positions of the input formula $Pos(\varphi)$ is finite and each position can be extracted a finite number of times. Moreover, it is easy to verify that each possible operation to transform an intermediate formula and an intermediate multigraph are finite too. In doing this one can benefit from considering the pseudo-code reported in the Appendix. In particular, the procedures described by the rules (1)–(3) and (5)–(7) of the algorithm, originate a finite number of steps because they involve the construction of a new component of the multigraph with only one edge (i.e., rules (2), (5.a), and (5.b)), the elimination/introduction of a (finite) component in the multigraph (i.e., rules (5.a), and (5.b)), a recursive renaming of the components of

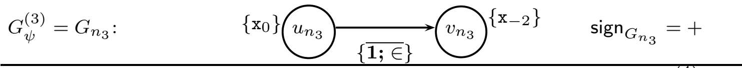
$n_1 = 1.1.1.1$. $\psi|_{n_1} = \mathtt{x}_{-4} \in \mathtt{x}_{-2}$. By rule (2), we have $\psi^{(1)} = \psi$ and $G^{(1)}_\psi = G_{n_1}$:



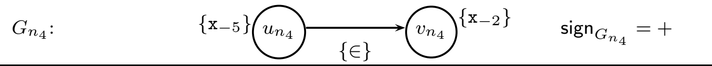
$n_2 = 1.1.1$. $\psi^{(1)}|_{n_2} = (\exists\,\mathtt{x}_{-4})\mathtt{x}_{-4} \in \mathtt{x}_{-2}$. By rule (3), since $\mathtt{x}_{-4}$ occurs once in $\psi^{(1)}|_{n_2.1}$, we have $\psi^{(2)} = (\forall\,\mathtt{x}_{-2})(\neg \mathtt{x}_0 \mathbf{1;} \in \mathtt{x}_{-2} \vee (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \neg(\exists\,\mathtt{x}_{-6})(\mathtt{x}_{-6} \in \mathtt{x}_{-5} \wedge \mathtt{x}_{-6} \in \mathtt{x}_{-2})))$



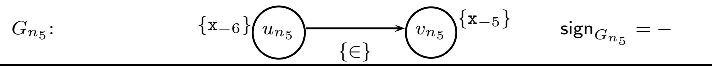
$n_3 = 1.1$. $\psi^{(2)}|_{n_3} = \neg\, \mathtt{x}_0 \mathbf{1;} \in \mathtt{x}_{-2}$. By rule (5.a), we have $\psi^{(3)} = (\forall\,\mathtt{x}_{-2})(\mathtt{x}_0 \overline{\mathbf{1;} \in} \mathtt{x}_{-2} \vee (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \neg(\exists\,\mathtt{x}_{-6})(\mathtt{x}_{-6} \in \mathtt{x}_{-5} \wedge \mathtt{x}_{-6} \in \mathtt{x}_{-2})))$



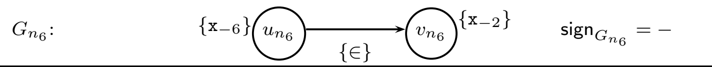
$n_4 = 1.2.1.1$. $\psi^{(3)}|_{n_4} = \mathtt{x}_{-5} \in \mathtt{x}_{-2}$. By rule (2), we have $\psi^{(4)} = \psi^{(3)}$ and $G^{(4)}_\psi = G_{n_3}, G_{n_4}$.



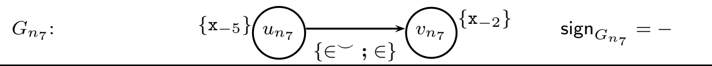
$n_5 = 1.2.1.2.1.1.1$. $\psi^{(4)}|_{n_5} = \mathtt{x}_{-6} \in \mathtt{x}_{-5}$. By rule (2), we have $\psi^{(5)} = \psi^{(4)}$ and $G^{(5)}_\psi = G_{n_3}, G_{n_4}, G_{n_5}$.



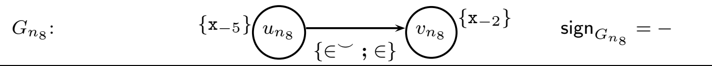
$n_6 = 1.2.1.2.1.1.2$. $\psi^{(5)}|_{n_6} = \mathtt{x}_{-6} \in \mathtt{x}_{-2}$. By rule (2), we have $\psi^{(6)} = \psi^{(5)}$, $G^{(6)}_\psi = G_{n_3}, G_{n_4}, G_{n_5}, G_{n_6}$.



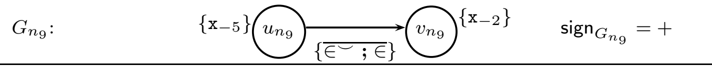
$n_7 = 1.2.1.2.1.1$. $\psi^{(6)}|_{n_7} = \mathtt{x}_{-6} \in \mathtt{x}_{-5} \wedge \mathtt{x}_{-6} \in \mathtt{x}_{-2}$. By rule (4.bypass), we have $G^{(7)}_\psi = G_{n_3}, G_{n_4}, G_{n_7}$ and $\psi^{(7)} = (\forall\,\mathtt{x}_{-2})(\mathtt{x}_0 \overline{\mathbf{1;} \in} \mathtt{x}_{-2} \vee (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \neg(\exists\,\mathtt{x}_{-6})(\mathtt{x}_{-5} \in^{\smile} \mathbf{;} \in \mathtt{x}_{-2})))$



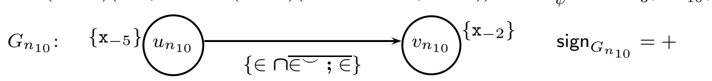
$n_8 = 1.2.1.2.1$. $\psi^{(7)}|_{n_8} = (\exists\,\mathtt{x}_{-6})\mathtt{x}_{-5} \in^{\smile} \mathbf{;} \in \mathtt{x}_{-2}$. By rule (3), since $\mathtt{x}_{-6}$ does not occur in $\psi^{(7)}|_{n_8.1}$, $\psi^{(8)} = (\forall\,\mathtt{x}_{-2})(\mathtt{x}_0 \overline{\mathbf{1;} \in} \mathtt{x}_{-2} \vee (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \neg \mathtt{x}_{-5} \in^{\smile} \mathbf{;} \in \mathtt{x}_{-2}))$ and $G^{(8)}_\psi = G_{n_3}, G_{n_4}, G_{n_8}$, with:



$n_9 = 1.2.1.2$. $\psi^{(8)}|_{n_9} = \neg\, \mathtt{x}_{-5} \in^{\smile} \mathbf{;} \in \mathtt{x}_{-2}$. By rule (5.a), we have $\psi^{(9)} = (\forall\,\mathtt{x}_{-2})(\mathtt{x}_0 \overline{\mathbf{1;} \in} \mathtt{x}_{-2} \vee (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \mathtt{x}_{-5} \overline{\in^{\smile} \mathbf{;} \in} \mathtt{x}_{-2}))$ and $G^{(9)}_\psi = G_{n_3}, G_{n_4}, G_{n_9}$, with:



$n_{10} = 1.2.1$. $\psi^{(9)}|_{n_{10}} = \mathtt{x}_{-5} \in \mathtt{x}_{-2} \wedge \mathtt{x}_{-5} \overline{\in^{\smile} \mathbf{;} \in} \mathtt{x}_{-2}$. By rule (4.fusion), we have $\psi^{(10)} = (\forall\,\mathtt{x}_{-2})(\mathtt{x}_0 \overline{\mathbf{1;} \in} \mathtt{x}_{-2} \vee (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \cap \overline{\in^{\smile} \mathbf{;} \in} \mathtt{x}_{-2}))$ and $G^{(10)}_\psi = G_{n_3}, G_{n_{10}}$, with:



$n_11 = 1.2$. $\psi^{(10)}|_{n_{11}} = (\exists\,\mathtt{x}_{-5})(\mathtt{x}_{-5} \in \cap \overline{\in^{\smile} \mathbf{;} \in} \mathtt{x}_{-2})$. By rule (3), since $\mathtt{x}_{-5}$ occurs once in $\psi^{(10)}|_{n_{11}.1}$, we have $\psi^{(11)} = (\forall\,\mathtt{x}_{-2})(\mathtt{x}_0 \overline{\mathbf{1;} \in} \mathtt{x}_{-2} \vee \mathtt{x}_0 \mathbf{1;}(\in \cap \overline{\in^{\smile} \mathbf{;} \in})\mathtt{x}_{-2})$ and $G^{(11)}_\psi = G_{n_3}, G_{n_{11}}$, with:

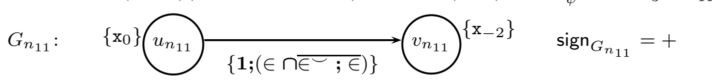


Figure 11: A run of the thinning algorithm for Example 7.

a (finite) multigraph by the procedure *Rename* (i.e., rules (3), (5.b)). The steps described in item (4), namely collapse, normalization, fusion, bypass, and bigamy, cause the execution of a finite number of steps as well. In fact, considering the pseudo-code in the Appendix, observe that their loops are executed a finite number of times only. Moreover, each single instruction of their code describes an operation (analogous to the ones mentioned above) that can be executed in a finite number of steps.

Correctness of the graph-thinning algorithm is stated by the following theorem:

Theorem 1 *Let φ be a formula of $\mathcal{L}^+$. If the graph-thinning algorithm terminates successfully with input φ yielding as output the quantifier-free formula ψ of $\mathcal{L}^+$ and the multigraph G_ψ, then*

1. *φ and ψ are logically equivalent, that is $(\varphi)^{\Im}$ is true iff $(\psi)^{\Im}$ is true;*
2. *the multigraph G_ψ represents ψ.*

Proof: We prove the first part of the theorem by showing that the property

'$\psi^{(i)}$ and $\psi^{(i+1)}$ are logically equivalent, for every $i \in \{0, \ldots, k-1\}$'

is an invariant for the algorithm. The proof is by induction on i, and by case distinction since it has to take into account all the local transformations that can be performed on the intermediate formulae by the procedures of the graph-thinning algorithm. Each transformation step is justified by the axiom schemata of Section 1.3 together with some other well-known laws of first-order logic. In particular, if $\psi^{(i)}|_n$ is a variable (rule (1) of the algorithm), an atomic formula (rule (2)), or a negated conjunction (rule (5.b)), $\psi^{(i)}|_n = \psi^{(i+1)}|_n$, and thus the property holds. If $\psi^{(i)}|_n$ is an existential formula (rule (3)), equivalence is preserved because when the quantified variable does not occur in $\psi^{(i)}|_n$, the quantifier can simply be removed. On the other hand, if the quantified

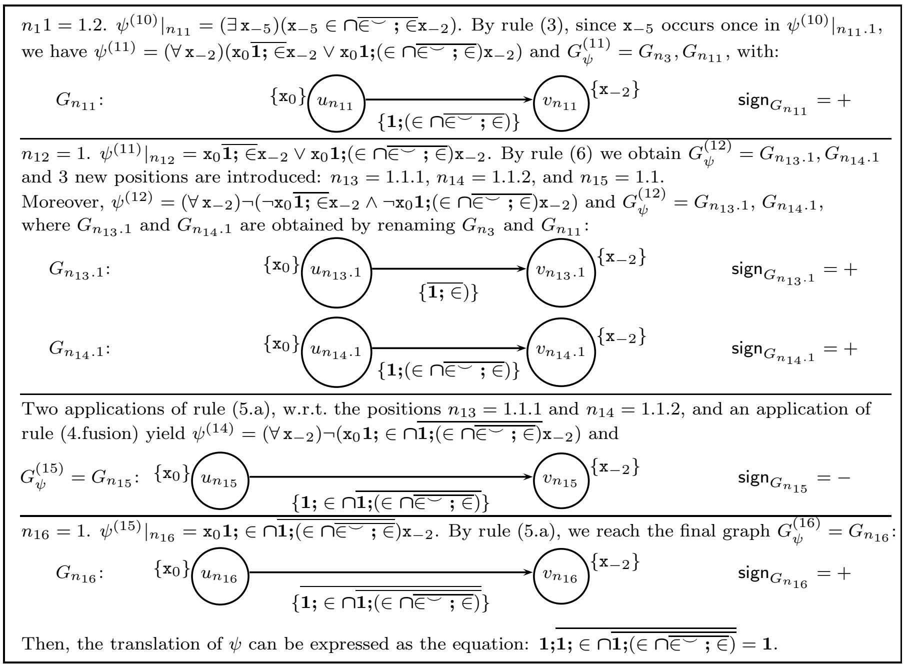


Figure 12: A run of the thinning algorithm (see also Figure 11 and Example 7).

variable occurs just once in $\psi^{(i)}|_n$, $\psi^{(i+1)}|_n$ is obtained from $\psi^{(i)}|_n$ by applying an instance of the axiom schema $(\forall x)(\forall y)(xR;\mathbf{1}y) \equiv (\exists z)(xRz \wedge z\mathbf{1}y)$. Analogously, if $\psi^{(i)}|_n$ is a negated atom (rule (5.a)), the property holds by applying one of the axioms of Section 1.3, whereas, if it is a double negated formula (rule (5.c)) by elementary rules of propositional logic. If $\psi^{(i)}|_n = \chi_1 \vee \cdots \vee \chi_n$, rule (6) is applied and $\psi^{(i+1)} = \psi^{(i)}[n/\neg(\neg\chi_1 \wedge \cdots \wedge \neg\chi_n)]$. Clearly, since $\chi_1 \vee \cdots \vee \chi_n \equiv \neg(\neg\chi_1 \wedge \cdots \wedge \neg\chi_n)$, the property holds. If $\psi^{(i)}|_n = (\forall x)\chi$, the proof is carried out as for case (6) and we do not report it. If $\psi^{(i)}|_n = \chi_1 \wedge \cdots \wedge \chi_n$, we apply rule (4). In this case we have to show correctness of the four steps of rule (4). This amounts to considering the procedures collapse, normalization, fusion, bypass, and bigamy. We prove that equivalence is preserved for the fusion step only. The proof is similar in the other cases. The fusion step selects in $\psi^{(i)}|_n$ atomic conjuncts of the forms xRy and xSy and replaces them with the conjunct $xR \cap Sy$. By the axioms of Section 1.3, the resulting formula is equivalent to $\psi^{(i)}|_n$ and thus the property is satisfied. To prove that the second part of the theorem holds, it suffices to verify that G_ψ is constructed so as to fulfill the conditions described in Section 2 and defining the graph representation of a formula. □

6 Related Work

During the last decades, several authors proposed and investigated graphical techniques applied to map calculus. Generally the proposed approaches support some form of inference mechanism, possibly exploited in specific contexts. An example is given in [2, 3] where the authors develop relational-based methods to formalize, design, and verify hardware circuits. This goal is achieved by developing a correspondence between two representations of circuits, one based on pictorial devices and another relying on the relational framework. Consequently, a natural relational semantics for pictures/circuits is easily obtained. High-level operations on pictures/circuits are rendered through transformation rules that ultimately reflect the semantics of the map constructs.

In [10, 11], a graphical representation of expressions by means of *diagrams* is proposed. In this case, only the fragment of $\mathcal{L}^\times$ not involving complementation (and excluding union and relational sum †, as well) is considered. A notion of reduction for the expressions of this language is given in terms of morphisms between diagrams. The paper also provides interesting results regarding normalization and decidability properties. The graphical framework proposed in [11] is closely related to the well-known class of series-parallel networks [9]. In a sense, the properties of the diagrammatic framework of [11] reflect analogous properties of those networks.

Another graphical calculus enabling representation and visual reasoning on mathematical formulae is proposed in [6]. Also in this case, the fragment of language taken into account is devoid of complementation. The proposed treatment of composition and intersection of maps, largely coincides with the one of [5, 15], that we have enhanced above. Actually, the graph-transformation rules described in [6] have a large overlap with the ones of our graph-fattening algorithm (when restricted to intersection, composition, and conversion).

The positive fragment of $\mathcal{L}^\times$ (where, complementation is banned) is also the object of the investigation carried out in [7]. In this case, the framework described in [6] is enriched by admitting expressions involving the union construct. Similarly to our proposal, admitting union forces one to consider a collection of graphs in place of a single graph, to represent an expression. The authors of [7] also propose a set of rewriting rules (*graph relational rules*) that support a notion of derivability within the graphical framework. The expressive power of the graph relational calculus is shown to capture the class of formulae of positive existential first-order logic with (at most) two free variables. This result is, clearly, in line with [24]. Finally, we mention [20, 21] that provide a general approach to the graphical calculi of relations introduced by [2] and [6], by resorting to algebraic graph-rewriting techniques.

Conclusions and Future Work

We have enhanced preexisting algorithms for translating dyadic first-order logic into map calculus, which rely on a specific graph representation of map expressions. As an outcome, we are able to deal with map expressions and with formulae containing the relational complement construct and the negation connective.

The first algorithm, graph-fattening, constructs a maximally expanded multigraph representing the internal structure of a given map expression. To do this, it decomposes the expression by a Smullyan-like unifying notation for map expressions. When applied to a map expression, it provides a representation which better conveys its meaning. The second algorithm, graph-thinning, translates formulae of dyadic first-order logic into map expressions. It works bottom-up (w.r.t. the structure of the formula), to build up a graph which represents the output formula, and by transforming the input formula destructively.

We plan to further improve the bypass and bigamy rules, making them more liberal in the case of nested quantifiers, and to incorporate into that algorithm rules that exploit semantical information such as functionality of maps (for instance, the knowledge that an expression P denotes a single-valued relation). Such rules could enable an otherwise unachievable translation.

A first attempt in the implementation of a proof-assistant based on the graph representation of map expressions has been done in [19]. In that case the algorithms described in [5, 15] have been implemented on top of the attributed graph-transformation system Agg. In order to validate the approach described in this paper, an implementation of the new algorithms is due (either as a stand-alone tool or in integration with standard theorem provers for first-order logic). These are interesting and challenging topics for further research.

References

[1] Belinfante, J.G.F. (2000). Gödel's algorithm for class formation. In D.McAllester, ed, *Proc. of CADE'00*, LNCS 1831, pages 132–147.

[2] Brown, C. and Hutton, G. (1994). Categories, allegories and circuit design. In *Proc. of 9th IEEE Symp. on Logic in Computer Science*, pp. 372-381. IEEE Computer Society Press.

[3] Brown, C. and Jeffrey, A. (1994). Allegories of circuits. In *Proc. of Logic for Computer Science*, pages 56-68.

[4] Cantone, D., Cavarra, A., and Omodeo, E. G. (1997). On existentially quantified conjunctions of atomic formulae of $\mathcal{L}^+$. In M. P. Bonacina and U. Furbach, eds, *Proc. of International Workshop on First-Order Theorem Proving (FTP97).*

[5] Cantone, D., Formisano, A., Omodeo, E. G., and Zarba, C. G. (2003) Compiling dyadic first-order specifications into map algebra. Theoretical Computer Science 293(2):447-475.

[6] Curtis, S. and Lowe, G., (1996) Proofs with graphs. Science of Computer Programming 26:197-216.

[7] de Freitas, R., Veloso, P.A.S., Veloso, S.R.M., Viana, P. (2009). On graph reasoning. Information and Computation, (207):228-246.

[8] Dershowitz, N. and Jouannaud, J.-P. (1990). Rewrite systems. In *Handbook of Theoretical Computer Science, Vol. B.: Formal Models and Semantics (B)*, pp.243-320.

[9] Duffin, R. J. (1965). Topology of Series-Parallel Graphs. J. of Mathematical Analysis and Applications, (10):303-318.

[10] Dougherty, D. and Gutiérrez, C. (2000). Normal forms and reduction for theories of binary relations. In L. Bachmair, ed., *Proc. of Rewriting Techniques and Applications*, LNCS, 1833.

[11] Dougherty, D. and Gutiérrez, C. (2006). Normal forms for binary relations. Theoretical Computer Science, 360:228-246.

[12] Fitting, M. C. (1996). First-order Logic and Automated Theorem Proving, Graduate Texts in Computer Science. Springer-Verlag, New York, 2nd edition.

[13] Formisano, A., Nicolosi Asmundo, M. (2006). An efficient relational deductive system for propositional non-classical logics. J. Applied Non-Classical Logics, 16(3-4):367-408.

[14] Formisano, A. and Omodeo, E. G., (2000). An equational re-engineering of set theories. In R. Caferra and G. Salzer, eds., *Selected Papers from Automated Deduction in Classical and Non-Classical Logics*, LNCS, 1761, pages 175-190.

```
procedure Rename(G_n, k)
  if (u_n, v_n ∈ V_n) then  // Rename the distinguished nodes of G_n
     u_n := u_k; v_n := v_k;
  endif;
  if (¬isAtom(ψ^(i)|_n)) then
     Φ_n := {l ∈ ℕ : l ∈ Pos(ψ^(i)|_n)};
     while (Φ_n ≠ ∅) do
           j := extractMin(Φ_n);
           G_{k.j} := Rename(G_{n.j}, k.j);
           G_ψ^(i) := (G_ψ^(i) \ {G_{n.j}}) ∪ G_{k.j};
     endwhile;
  endif;
end procedure
```

Figure 13: The procedure *Rename*.

[15] Formisano, A., Omodeo, E. G., and Simeoni, M. (2001). A graphical approach to relational reasoning. In W. Kahl, D. L. Parnas, and G. Schmidt, eds., *Proc. of RelMiS 2001 (ETAPS 2001).* Electronic Notes in Theoretical Computer Science, 44(3), Elsevier Science.

[16] Formisano, A., Omodeo, E. G., and Temperini, M. (2000). Goals and benchmarks for automated map reasoning. J. of Symbolic Computation, 29(2):259-297.

[17] Formisano, A., Omodeo, E. G., and Temperini, M. (2001) Instructing equational set-reasoning with Otter. In R. Goré, A. Leitsch, and T. Nipkow, eds, *Proc. of IJCAR'01.*

[18] Formisano, A., Omodeo, E. G., and Temperini, M. (2001) Layered map reasoning: An experimental approach put to trial on sets. In A. Dovier, M.-C. Meo, and A. Omicini, eds, *Declarative Programming*, ENTCS, 48:1–28, Elsevier.

[19] Formisano, A. and Simeoni, M. (2001). An AGG application supporting visual reasoning. In L. Baresi and M. Pezzè, eds., *Proc. of GT-VMT'01 (ICALP 2001).* Electronic Notes in Theoretical Computer Science, 50(3), Elsevier Science.

[20] Kahl, W., (1997). Algebraic graph derivations for graphical calculi. In F. d'Amore, P. G. Franciosa and A. Marchetti-Spaccamela, eds., *Proc. of Graph Theoretic Concepts in Computer Science, WG '96*, LNCS, 1197, pages 224-238.

[21] Kahl, W., (1999). Relational matching for graphical calculi of relations. Information Sciences, (119):253-273.

[22] Kwatinetz, M. K. (1981) *Problems of expressibility in finite languages.* PhD thesis, University of California, Berkeley.

[23] Smullyan, R. M. (1995) *First-order Logic.* Dover Publications, New York.

[24] Tarski, A. and Givant, S. (1987). *A formalization of Set Theory without variables*, vol. 41 of Colloquium Publications. American Mathematical Society.

[25] von Neumann, J. (1925). Eine Axiomatisierung der Mengenlehre. J. für die reine und angewandte Mathematik, 154:219–240. English translation in J. van Heijenoort, ed., *From Frege to Gödel: a source book in mathematical logic, 1879–1931.* Harvard University Press, 1977.

A Details of the graph-thinning

A.1 The procedure *Rename*

The procedure *Rename* is illustrated in Figure 13. It works recursively by renaming the top component of G_n, if any, and all its subcomponents.

A.2 The procedure *Collapse*

For every χ_j which is a conjunction (α-formula), the procedure *Collapse* illustrated in Figure 14 adds all the conjuncts of χ_j directly to $\psi^{(i)}|_n$, and renames the labels of the relative multigraphs and distinguished nodes, according to the new positions in which χ_j occurs. The process of renaming the distinguished nodes of every component of $G_{n.j}$ (and of their subcomponents) is performed by the recursive procedure *Rename* of Figure 13.

A.3 The procedure *Normalize*

The normalization step is illustrated in Figure 15 and works as follows. For every conjunct χ_j in $\psi^{(i)}|_n$ (line 1) that is an atomic formula of type xRx (lines 4 and 5), it substitutes in $\psi^{(i)}|_n$ the occurrence of xRx with $x(R \cap \iota)x'$, where x' is a new existentially quantified variable introduced by means of the function *newOddVar*. Then the component $G_{n.j}$ of G_n is modified (lines 8 and 9) by labeling its edge with $\{(R \cap \iota)\}$ and its nodes with $\{x\}$ and $\{x'\}$, respectively.

procedure *Collapse*(n)
$\Phi_n := \{l | l \in Pos(\psi^{(i)}|_n) \cap \mathbb{N}\}$; // Positions of the conjuncts of $Pos(\psi^{(i)}|_n)$
$m := |\Phi_n|$;
while $(\Phi_n \neq \emptyset)$ do
    $j := extractMin(\Phi_n)$;
    if $(isAlpha(\psi^{(i)}|_{n.j}))$ then // if it is an α-formula
        $\Psi_n := \{k \in \mathbb{N} : k \in Pos(\psi^{(i)}|_{n.j})\}$;
        //replace $\psi^{(i)}|_n.j$ in $\psi^{(i)}|_n$ with its first conjunct
        $\psi^{(i)} := \psi^{(i)}[n.j/\psi^{(i)}|_{n}.j.1]$;
        $\Psi_n := \Psi_n \setminus \{1\}$;
        $G^{(i)}_\psi := G^{(i)}_\psi \setminus \{G_{n.j}\}$; // modify the multigraph
        $G_{n.j} := Rename(G_{n.j.1}, n.j)$;
        $G^{(i)}_\psi := G^{(i)}_\psi \cup \{G_{n.j}\}$;
        while $(\Psi_n \neq \emptyset)$ do
        // enlarge $\psi^{(i)}|_n$ adding, as new conjuncts,
        // the other conjuncts of $\psi^{(i)}|_{n.j}$, and modify the multigraphs
            $k := extractMin(\Psi_n)$;
            $\psi^{(i)} := \psi^{(i)}[n.((k-1)+m)/\psi^{(i)}|_{n.j.k}]$;
            $G_{n.k} := Rename(G_{n.j.k}, n.((k-1)+m))$;
            $G^{(i)}_\psi := (G^{(i)}_\psi \setminus G_{n.j.k}) \cup G_{n.((k-1)+m))}$;
        endwhile;
        m := m + (k-1);
    endif;
endwhile;
end procedure

Figure 14: The procedure*Collapse*.

procedure *Normalize*(n)
$\Phi_n := \{l \in \mathbb{N} : l \in Pos(\psi^{(i)}|_n)\}$;
while $\Phi_n \neq \emptyset$ do
    $j := extractMin(\Phi_n)$;
    if $(isAtom(\psi^{(i)}|_{n.j}))$ then
      if $(\psi^{(i)}|_{n.j.1} = \psi^{(i)}|_{n.j.2})$ then
        $\psi^{(i)}|_{n.j.2} := newOddVar(\psi^{(i)})$;
        $\psi^{(i)} := \psi^{(i)}[n.j/(\psi^{(i)}|_{n.j.1}(Map(\psi^{(i)}|_{n.j}) \cap \iota)\psi^{(i)}|_{n.j.2})]$;
        $lEdge(u_{n.j}, v_{n.j}) := \{Map(\psi^{(i)}|_{n.j})\}$;
        $lNode(v_{n.j}) := \{\psi^{(i)}|_{n.j.2}\}$;
      endif;
    endif;
endwhile;
end procedure

Figure 15: The normalization procedure.

A.4 The procedure *Fusion*

The procedure *Fusion* is depicted in Figure 16. For every atomic conjunct of $\psi^{(i)}|_n$, the procedure searches within $\psi^{(i)}|_n$ for all the atoms sharing the same variables as arguments, and merges them (lines 12-21). For each pair of conjuncts, it operates as follows: it modifies 'in place' in $\psi^{(i)}|_n$ the first conjunct (the one indexed by $\bar{\jmath}$ in the code), it removes the second conjunct (the one indexed by $\bar{k}$) from $\psi^{(i)}|_n$, and it shifts all the remaining conjuncts one position to the left (lines 22-27). The corresponding multigraph components are treated in an analogous way: the first component (indicated by the index $\bar{\jmath}$) is modified, then the second one (indicated by the index $\bar{k}$) is eliminated, and finally the remaining components are suitably renamed (because the corresponding positions in the formula have been shifted).

A.5 The procedure *Bypass*

Let us describe the main steps of the procedure *Bypass* (Figure 17). First the collection Φ_n of the positions of conjuncts of $\psi^{(i)}|_n$ is determined. For each position j of an atomic conjunct of $\psi^{(i)}|_n$, the procedure determines the set $BoundVar(\psi^{(i)}|_{n.j})$ of all the variables in $Var(\psi^{(i)}|_{n.j}) \cap Var^-$ that occur exactly twice in $\psi^{(i)}$ and occur only in atomic conjuncts of $\psi^{(i)}|_n$ (lines 3-5).

For every variable x in $BoundVar(\psi^{(i)}|_{n.j})$, if either x is existentially quantified, $\psi^{(i)}|_n$ occurs positively in $\psi^{(i)}$, and no universal quantifier is in the scope of the quantifier binding x, or x is universally quantified, $\psi^{(i)}|_n$ occurs negatively in $\psi^{(i)}$, and no existential quantifier is in the scope of the quantifier binding x (this test is performed in line 7), the code in lines 8-40 is executed. The position k of the other conjunct in $\psi^{(i)}|_n$ having x as argument is determined in line 9. The bypass operation is then performed on the two conjuncts (lines 10-29). In particular, since x occurs as one of two arguments in each one of the two disjuncts, four cases are possible. As an example, let us describe the first of them (lines 11-13). If $\psi^{(i)}|_{\bar{\jmath}} = yRx$ and $\psi^{(i)}|_k = xSz$, the two graphs $G_{\bar{\jmath}}$ and G_k are merged into a new component replacing $G_{\bar{\jmath}}$ that has only one edge labeled with $\{R;S\}$ and the two nodes labeled with $\{y\}$ and $\{z\}$, respectively (this 'new' $G_{\bar{\jmath}}$ is depicted in Figure 9). The other component, G_k, is removed (line 30) from $G^{(i)}_n$. At the same time, the formula $\psi^{(i)}$ is coherently modified in line 33. Since in processing the two conjuncts, the one in position k has been removed, the remaining sequence of conjuncts

```
procedure Fusion(n)
  Φ_n := {l ∈ ℕ : l ∈ Pos(ψ^(i)|_n)}; pp_1 := {};
  while Φ_n ≠ ∅ do
        j := extractMin(Φ_n); pp_1 := pp_1 ∪ {j}
        ȷ̄ := n.j;
        if isAtom(ψ^(i)|_ȷ̄) then
          Expr_ȷ̄ := Map(ψ^(i)|_ȷ̄);
          S_n := Φ_n; pp_2 := {};
          while S_n ≠ ∅ do
                k := extractMin(S_n);
                if (Var(ψ^(i)|_{n.k}) = Var(ψ^(i)|_ȷ̄)) then
                  k̄ := n.k;
                  if (ψ^(i)|_{k̄.1} = ψ^(i)|_{ȷ̄.1}) then
                    Expr_ȷ̄ := Expr_ȷ̄ ∩ Map(ψ^(i)|_k̄);
                  else
                    Expr_ȷ̄ := Expr_ȷ̄ ∩ Map(ψ^(i)|_k̄)^⌣;
                  endif;
                  lEdge((u_ȷ̄, v_ȷ̄)) := {Expr_ȷ̄};
                  G_n := G_n \ {G_k̄};
                  x_1 := extract(lNode(u_ȷ̄));
                  x_2 := extract(lNode(v_ȷ̄));
                  ψ^(i) := ψ^(i)[ȷ̄/x_1 Expr_ȷ̄ x_2];
                  Ψ := S_n;
                  while (Ψ ≠ ∅) do // 'shift' remaining conjuncts of ψ^(i)|_n
                      h := extractMin(Ψ);   ψ^(i) := ψ^(i)[n.(h−1)/ψ^(i)|_{n.h}];
                      G_{n.(h−1)} := Rename(G_{n.h}, n.(h−1));
                      G_ψ^(i) := (G_ψ^(i) \ G_{n.h}) ∪ G_{n.(h−1)};
                  endwhile;
                  Φ_n := {l ∈ ℕ : l ∈ Pos(ψ^(i)|_n)} \ pp_1;
                  S_n := Φ_n \ pp_2;
                else pp_2 := pp_2 ∪ {k};
                endif;
          endwhile;
        endif;
  endwhile;
end procedure
```

Figure 16: The multiple-edge elimination procedure.

is compacted in lines 34-38 (this might involve renaming of their distinguished nodes). Finally, the set of the positions of conjuncts of $\psi^{(i)}|_n$ is updated in line 39.

A.6 The procedure *Bigamy*

The *Bigamy* procedure (see Figure 18) is applied to every bound node of G_n with just one adjacent edge. In terms of $\psi^{(i)}|_n$, it applies to every quantified variable x that occurs only once in $\psi^{(i)}$. For simplicity, in Figure 18 we provide a simplified procedure that deals only with the case of an existentially quantified variable x, occurring in $\psi^{(i)}|_n$ as a left argument of an atomic formula. The other cases are treated likewise. Let y be the variable occurring as right argument in the atom where x occurs. In line 11 the procedure determines (if any) a second occurrence of y in another atom of $\psi^{(i)}|_n$. Let d be the position of the determined atom. Then a new component G_q, to be added to the multigraph, is created. G_q is made of a single edge (u_q, v_q). The labels of the two new nodes are assigned in lines 16-21. In particular, one of them must be $\{w\}$ and the other one must be $\{x\}$, where w is one of the two variables of the atom in position d (the other variable being y). The label of the edge is set in line 23 to be the universal map $\{\mathbf{1}\}$. Finally, the new component is added to G_n, whereas the formula $\psi^{(i)}|_n$ is updated by conjoining the literal $w\mathbf{1}x$

procedure *Bypass*(n)
$\Phi_n := \{l \in \mathbb{N} : l \in Pos(\psi^{(i)}|_n)\}$;
while $\Phi_n \neq \emptyset$ do
  $j := extractMin(\Phi_n)$; $\bar{\jmath} := n.j$; $Expr_{\bar{\jmath}} := Map(\psi^{(i)}|_{\bar{\jmath}})$;
  if $isAtom(\psi^{(i)}|_{\bar{\jmath}})$ then
    $BoundVar(\psi^{(i)}|_{\bar{\jmath}}) := \{x \in (Var(\psi^{(i)}|_{\bar{\jmath}}) \cap Var^-) : |P_x^{\psi^{(i)}}| = |P_x^{\psi^{(i)}|_n} \cap (\mathbb{N} \cdot \mathbb{N})| = 2\}$;
    for every $x \in BoundVar(\psi^{(i)}|_{\bar{\jmath}})$ do
      if $((\exists j \in \mathbb{N} : x = \mathtt{x}_{-(2j+1)}) \wedge OccPositively(\psi^{(i)}|_n, \psi^{(i)}) \wedge NoScope(\forall, \mathsf{Q}_x)) \vee$
        $((\exists j \in \mathbb{N}^+ : x = \mathtt{x}_{-2j}) \wedge \neg OccPositively(\psi^{(i)}|_n, \psi^{(i)}) \wedge NoScope(\exists, \mathsf{Q}_x))$ then
        $p := extract(\{s : s \in P_x^{\psi^{(i)}|_n}, Father(s) \neq \bar{\jmath}\})$; // second occurrence of x
        $k := Father(p)$;
        if $(x = \psi^{(i)}|_{\bar{\jmath}.2})$ then // if one atom is of type yRx
          if $(p = k \cdot 1)$ then // and the other one of type xSz
            $Expr_{\bar{\jmath}} := (Expr_{\bar{\jmath}}; Map(\psi^{(i)}|_k))$;
            $lNode(v_{\bar{\jmath}}) := \psi^{(i)}|_{k \cdot 2}$;
          else// if the other one is of type zSx
            $Expr_{\bar{\jmath}} := (Expr_{\bar{\jmath}}; Map(\psi^{(i)}|_k)^{\smile})$;
            $lNode(v_{\bar{\jmath}}) := \psi^{(i)}|_{k \cdot 1}$;
          endif;
          $lEdge((u_{\bar{\jmath}}, v_{\bar{\jmath}})) := Expr_{\bar{\jmath}}$;
        else // if one atom is of type xRy
          if $(p = k \cdot 1)$ then // and the other one of type xSz
            $Expr_{\bar{\jmath}} := (Expr_{\bar{\jmath}}^{\smile}; Map(\psi^{(i)}|_k))$;
            $lNode(v_{\bar{\jmath}}) := \psi^{(i)}|_{k \cdot 2}$;
          else// if the other one is of type zSx
            $Expr_{\bar{\jmath}} := (Expr_{\bar{\jmath}}^{\smile}; Map(\psi^{(i)}|_k)^{\smile})$;
            $lNode(v_{\bar{\jmath}}) := \psi^{(i)}|_{k \cdot 1}$;
          endif;
          $lEdge((u_{\bar{\jmath}}, v_{\bar{\jmath}})) := Expr_{\bar{\jmath}}$;
          $lNode(u_{\bar{\jmath}}) := \psi^{(i)}|_{\bar{\jmath} \cdot 2}$;
        endif;
        $G_n := G_n \setminus \{G_k\}$; // removal of G_k (it has been merged into $G_{\bar{\jmath}}$)
        $\Psi := \{s \in \Phi_n : s > k\}$;
        $x_1 := extract(lNode(u_{\bar{\jmath}}))$; $x_2 := extract(lNode(v_{\bar{\jmath}}))$;
        $\psi^{(i)} := \psi^{(i)}[\bar{\jmath}/x_1 Expr_{\bar{\jmath}} x_2]$;
        while $(\Psi \neq \emptyset)$ do // 'shift' remaining conjuncts of $\psi^{(i)}|_n$
          $h := extractMin(\Psi)$; $\psi^{(i)} := \psi^{(i)}[n.(h-1)/\psi^{(i)}|_{n.h}]$;
          $G_{n.(h-1)} := Rename(G_{n.h}, n.(h-1))$;
          $G_\psi^{(i)} := (G_\psi^{(i)} \setminus G_{n.h}) \cup G_{n.(h-1)}$;
        endwhile;
        $\Phi_n := \{l \in \mathbb{N} : l \in Pos(\psi^{(i)}|_n)\}$;
      endif;
    endfor;
  endif;
endwhile;
end procedure

Figure 17: The procedure *Bypass*. It takes a position n as parameter and acts on $\psi^{(i)}$ and $G_\psi^{(i)}$ by modifying $\psi^{(i)}|_n$ and G_n.

procedure $Bigamy(n)$
$\Phi_n := \{j \in \mathbb{N} : j \in Pos(\psi^{(i)}|_n)\}$;
$m := |\Phi_n|$;
while $\Phi_n \neq \emptyset$ do
    $l := extractMin(\Phi_n)$;
    $\bar{l} := n.l$
    if $isAtom(\psi^{(i)}|_{\bar{l}})$ then
        $BoundVarSingle(\psi^{(i)}|_{\bar{l}}) := \{x \in Var(\psi^{(i)}|_{\bar{l}}) \cap Var^- : |P_x^{\psi^{(i)}}| = |P_x^{\psi^{(i)}|_n}| = 1\}$;
        foreach $x \in BoundVarSingle(\psi^{(i)}|_{\bar{l}})$ do
            if $(\exists h \in \mathbb{N} : x = \mathtt{x}_{-(2h+1)}) \wedge NoScope(\forall, \mathsf{Q}_x)$ then
            $p := extractMax(P_x^{\psi^{(i)}|_n})$;
                if $(p = \bar{l}.1 \wedge \exists d \in \Phi_n : isAtom(d) \wedge (\psi^{(i)}|_{\bar{l}.2} = \psi^{(i)}|_{d.1} \vee \psi^{(i)}|_{\bar{l}.2} = \psi^{(i)}|_{d.2}))$ then
                    $q := m + 1$;
                    $V_q := \{u_q, v_q\}$; $E_q := \{(u_q, v_q)\}$;
                    if $(\psi^{(i)}|_{\bar{l}.2} = \psi^{(i)}|_{d.1})$ then
                        $\psi^{(i)}|_n := \psi^{(i)}|_n \wedge \psi^{(i)}|_{d.2}\mathbf{1}\psi^{(i)}|_{\bar{l}.1}$;
                        $lNode(u_q) := \psi^{(i)}|_{d.2}$;
                        $lNode(v_q) := \psi^{(i)}|_{\bar{l}.1}$;
                    elseif $(\psi^{(i)}|_{\bar{l}.2} = \psi^{(i)}|_{d.2})$ then
                        $\psi^{(i)}|_n := \psi^{(i)}|_n \wedge \psi^{(i)}|_{d.1}\mathbf{1}\psi^{(i)}|_{\bar{l}.1}$;
                        $lNode(u_q) := \psi^{(i)}|_{d.1}$;
                        $lNode(v_q) := \psi^{(i)}|_{\bar{l}.1}$;
                    endif;
                    $lEdge((u_q, v_q)) := \{\mathbf{1}\}$;
                    $G_n := G_n \cup \{G_q\}$;
                    $m := m + 1$;
                endif;
            endif;
        endfor;
    endif;
endwhile;
end procedure

Figure 18: The procedure *Bigamy*.