Electronic Notes in Theoretical Computer Science 44 No. 3 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume44.html 22 pages

A graphical approach to relational reasoning [1]

Andrea Formisano [2]

Dipartimento di Matematica e Informatica, Università di Perugia

Eugenio G. Omodeo [3]

Dipartimento di Matematica e Pura ed Applicata, Università di L'Aquila

Marta Simeoni [4]

Dipartimento di Informatica, Università Ca' Foscari *di Venezia*

Abstract

Relational reasoning is concerned with relations over an unspecified domain of discourse. Two limitations to which it is customarily subject are: only dyadic relations are taken into account; all formulas are equations, having the same expressive power as first-order sentences in three variables. The relational formalism inherits from the Peirce-Schröder tradition, through contributions of Tarski and many others.

Algebraic manipulation of relational expressions (equations in particular) is much less natural than developing inferences in first-order logic; it may in fact appear to be overly machine-oriented for direct hand-based exploitation.

The situation radically changes when one resorts to a convenient representation of relations based on labeled graphs. The paper provides details of this representation, which abstracts w.r.t. inessential features of expressions.

Formal techniques illustrating three uses of the graph representation of relations are discussed: one technique deals with translating first-order specifications into the calculus of relations; another one, with inferring equalities within this calculus with the aid of convenient diagram-rewriting rules; a third one with checking, in the specialized framework of set theory, the definability of particular set operations. Examples of use of these techniques are produced; moreover, a promising approach to mechanization of graphical relational reasoning is outlined.

Key words: Formalized reasoning, algebra of dyadic relations, labeled multigraphs, Peircean existential diagrams, graph transformation.

1 Background

The graphs, he wrote, "put before us moving pictures of thought." They render the structure "literally visible before one's very eyes." In doing this they free the structure from all the "puerilities about words" with which so many English logical works are strewn. "Often not merely strewn with them," he adds, "but buried so deep in them, as by a great snowstorm, as to obstruct the reader's passage and render it fatiguing in the extreme." (From [20],[5] p.56)

Refined designs of the arithmetic of dyadic relations and related research, constitute the most traditional and lasting effort to bridge first-order predicate reasoning with purely equational reasoning [39,41].

Several axiomatizations of the algebra of relations are available (see, among others, [9,28,36,35,38]); our own, shown in Fig. 1, is conceived with the aim of providing support to theorem-proving activities based on a state-of-the-art proof assistant. These axioms will be left 'behind the scene' after this section, because the authors envision a diagrammatic approach to relational reasoning in place of a merely logical or algebraic symbolic manipulation system.[6]

The equalities in the fixed initial endowment of axioms, describing the full variety of dyadic relations over a generic domain $\mathcal{U}$ of discourse, are called *logical* axioms.

Further axioms, added to the logical ones, lead to algebraic characterizations of specific domains and data structures: at varying degrees of mathematical abstraction, one has general classes and sets, hereditarily finite sets, trees, nested or flat lists, lines subjected to editing, etc.—cf. [6,18]. These, which are called *proper* axioms, state the properties of context-specific relations, e.g., `car` and `cdr` in the case of lists, and membership, $\in$, in the case of sets. Typically, proper axioms are *ground*, i.e. devoid of the variables that occur in the logical axioms;[7] usually they involve, in addition to the standard constructs of relation algebra, the symbol(s) characteristic of the application, e.g. $\in$.

The intended meaning of the axioms in Fig. 1 is as follows: one refers to a nonempty domain $\mathcal{U}$, and thinks that the variables (P, Q, and R) occurring

[1] This research was partially funded by the Italian CNR (coordinated project log(SETA)); by MURST PGR-2000; by the EC TMR Network GETGRATS; and by Esprit Working Group APPLIGRAPH.

[2] Email:`formis@dipmat.unipg.it`

[3] Email:`omodeo@univaq.it`

[4] Email:`simeoni@dsi.unive.it`

[5] The author quoted by Martin Gardner is the American philosopher Charles Sanders Peirce (1839–1914), whose work on existential graphs (cf. [26,24]) is likely to have a kinship with ours, presented in this paper and in [6,8].

[6] Due to space limitations, the authors had to indulge to 'puerilities about words' much more in drawing up the article than in the paper-and-pencil preparatory work.

[7] A non-ground axiom could be the algebraic rendering of what would be an axiom *scheme* in a specification based on first-order predicate logic.

symbol :	$=$	$\subseteq$	$\perp\!\!\!\perp$	$\mathbb{T}$	$\mathbb{I}$	$\in$	$\cap$	$\triangle$	$;$	$\smile$	$\overline{}$	$-$	$\cup$
degree :	2	2	0	0	0	0	2	2	2	1	1	2	2
priority :	1	1					5	3	6	7		2	2

$P \cup Q \ \equiv_{\mathrm{Def}}\ (P \triangle Q) \triangle (P \cap Q)$	$P - Q \ \equiv_{\mathrm{Def}}\ P \triangle (P \cap Q)$
$\not{P} \ \equiv_{\mathrm{Def}}\ \overline{P} \ \equiv_{\mathrm{Def}}\ P \triangle \mathbb{T}$	$P \subseteq Q \ \equiv_{\mathrm{Def}}\ P \cap Q = P$
$\mathsf{funcPart}(P) \ \equiv_{\mathrm{Def}}\ P - (P ; \overline{\mathbb{I}})$	$\mathsf{tot}(P) \ \equiv_{\mathrm{Def}}\ P \triangle \overline{P ; \mathbb{T}}$
$\mathsf{Func}(P) \ \equiv_{\mathrm{Def}}\ \mathsf{funcPart}(P) = P$	$\mathsf{Total}(P) \ \equiv_{\mathrm{Def}}\ P ; \mathbb{T} = \mathbb{T}$

$$
\begin{aligned}
P \cap Q &= Q \cap P \\
(P \cap (Q \triangle R)) \triangle (P \cap Q) &= P \cap R \\
(P \star_1 Q) \star_1 R &= P \star_1 (Q \star_1 R) \\
\mathbb{I} ; P &= P \\
P^{\smile\smile} &= P \\
(P \star_2 Q)^{\smile} &= Q^{\smile} \star_2 P^{\smile} \\
(P \cup Q) ; R &= (Q ; R) \cup (P ; R) \\
(P^{\smile} ; (R - (P ; Q))) \cap Q &= \perp\!\!\!\perp \\
P &\subseteq \mathbb{T}
\end{aligned}
$$

$\star_1 \in \{\triangle, \cap, ;\}$ and $\star_2 \in \{\cap, ;\}$

Figure 1. Operators and axioms for relation algebras.

in the axioms range over all subsets of the Cartesian square $\mathcal{U}^2 =_{\mathrm{Def}} \mathcal{U} \times \mathcal{U}$. The following designation rules recursively extend an interpretation $\Im$ from the immediate subexpressions of a given expression to the expression itself:

$$
\begin{aligned}
&\perp\!\!\!\perp^{\Im} =_{\mathrm{Def}} \emptyset, \qquad \mathbb{T}^{\Im} =_{\mathrm{Def}} \mathcal{U}^2, \qquad \mathbb{I}^{\Im} =_{\mathrm{Def}} \{[a,a] : a \text{ in } \mathcal{U}\}; \\
&(Q \cap R)^{\Im} =_{\mathrm{Def}} \{ [a,b] \text{ in } \mathcal{U}^2 : a\, Q^{\Im}\, b \text{ and } a\, R^{\Im}\, b \}; \\
&(Q \triangle R)^{\Im} =_{\mathrm{Def}} \{ [a,b] \text{ in } \mathcal{U}^2 : a\, Q^{\Im}\, b \text{ if and only if not } a\, R^{\Im}\, b \}; \\
&(Q ; R)^{\Im} =_{\mathrm{Def}} \{ [a,b] \text{ in } \mathcal{U}^2 : \text{there exist } c\text{s in } \mathcal{U} \text{ for which } a\, Q^{\Im}\, c \text{ and } c\, R^{\Im}\, b \}; \\
&(Q^{\smile})^{\Im} =_{\mathrm{Def}} \{ [b,a] \text{ in } \mathcal{U}^2 : a\, Q^{\Im}\, b \}.
\end{aligned}
$$

Any non-logical symbol, such as $\in$, in the language of the algebra of relations, is not constrained in the least by the above rules. One must hence assign explicitly an interpretation to each such symbol (e.g., a value $\in^{\Im} \subseteq \mathcal{U}^2$, in short $\in$, must be given to the sign $\in$), before the relation on $\mathcal{U}$ that corresponds to each ground expression of the language becomes uniquely determined.

The proper axioms that one adds to the logical axioms are to reflect one's conception of the meaning of the non-logical symbols (e.g., in the case of $\in$

they must state that $\mathcal{U}$ is a hierarchy of nested sets over which $\in$ behaves as membership).

Remark 1.1 The law,[8] . $P^{\smile} ; (R \triangle R \cap P;Q) \cap Q = \perp\!\!\!\perp$, is closely related to the law $P^{\smile} ; \overline{P;Q} \cup \overline{Q} = \overline{Q}$ known from the work of Peirce and Schröder (see, for instance, [36,41]). Our adoption of a different set of primitive constructs motivated the introduction of this law which, moreover, permits a concise characterization of the constant $\perp\!\!\!\perp$ by means of a single axiom.

2 A graphical representation for dyadic relations

It is useful (cf. [5]) to represent a relational expression P, an identity $P = \mathbb{T}$ or, more generally, an existentially quantified conjunction φ of literals, by a so-called *existential graph*, which is a directed multi-graph whose edges are labeled by expressions of the algebra of relations. To see the most immediate way of doing this, let us assume that φ is composed by atoms of the form xPy, where x and y are individual variables (ranging over the domain $\mathcal{U}$ of discourse), and P is a relation of the kind discussed in Sec. 1. (Equality atoms $Q = R$ have been rewritten already in the form $x\overline{\mathbb{T};(Q\triangle R);\mathbb{T}}y$, negative literals in the form $x\overline{Q}y$; moreover, free variables may occur in φ intermixed with existentially quantified variables.) A directed multi-graph G_φ representing φ is built up so that:

1) G_φ has a node ν_x for each distinct variable x occurring in φ;

2) for each literal xPy in the conjunction φ, there is a labeled edge $[\nu_x, P, \nu_y]$ leading from node ν_x to node ν_y; and

3) the nodes of G_φ are subdivided into two sets: the ones that correspond to the existential variables in φ, called *bound* nodes, and all remaining nodes.

A chain of transformations can then be applied to any graph obtained in this standard fashion, by the following rules, which manifestly preserve the meaning of the graph (these rules are graphically shown in Fig. 2 where we used black circles to represent bound nodes, and white circles to represent the remaining ones):

(1) An edge $[\nu, \mathbb{T}, \nu']$ can be removed or created between nodes ν, ν'.

(2) An edge $[\nu, P, \nu']$ can be converted into $[\nu', Q, \nu]$ where either $P \equiv Q^{\smile}$ or $Q \equiv P^{\smile}$ or $P \equiv Q \equiv \mathbb{I}$.[9]

[8] We are now getting rid of redundant parentheses by exploiting the conventions on priority introduced in Fig. 1. The priorities we adopt for the Boolean constructs reflect the (abstract) algebraic traditional approach, where $\triangle$ and $\cap$ act as additive and multiplicative operators of Boolean algebras, respectively (cf. [27], pp.208–211). On the other hand, w.r.t. Peircean constructs, we inherit the well established convention adopted, for instance, in [21]

[9] Primarily, this 'conversion' is intended as an edge-replacement rule; however, it could also be intended in the sense that $[\nu', Q, \nu]$ is added to the graph without $[\nu, P, \nu']$ being removed.

(3) Two edges $[\nu, P, \nu']$ and $[\nu, Q, \nu']$ can be replaced by the single edge $[\nu, P \cap Q, \nu']$, and conversely.

(4) If $[\nu, P, \nu']$ and $[\nu', Q, \nu'']$ are the only edges involving the bound node ν', then they can be replaced by the single edge $[\nu, P ; Q, \nu'']$; conversely, an edge $[\nu, P ; Q, \nu'']$ can be replaced by two edges $[\nu, P, \nu']$ and $[\nu', Q, \nu'']$ where ν' is a new bound node.

(5) An edge $[\nu, \mathbb{I}, \nu']$, where either ν' is a bound node with degree 1,[10] or $\nu' \equiv \nu$, can be deleted; conversely, an edge $[\nu, \mathbb{I}, \nu']$ where either ν' is a new bound node or $\nu' \equiv \nu$ can be created.

(6) An edge $[\nu', Q, \nu]$ (respectively, $[\nu, Q, \nu']$) can be replaced by an edge $[\nu', Q, \nu'']$ (resp., $[\nu'', Q, \nu']$) when there is an edge $[\nu, \mathbb{I}, \nu'']$ distinct from $[\nu', Q, \nu]$ (resp., from $[\nu, Q, \nu']$).

(7) An isolated bound node can be deleted or created.

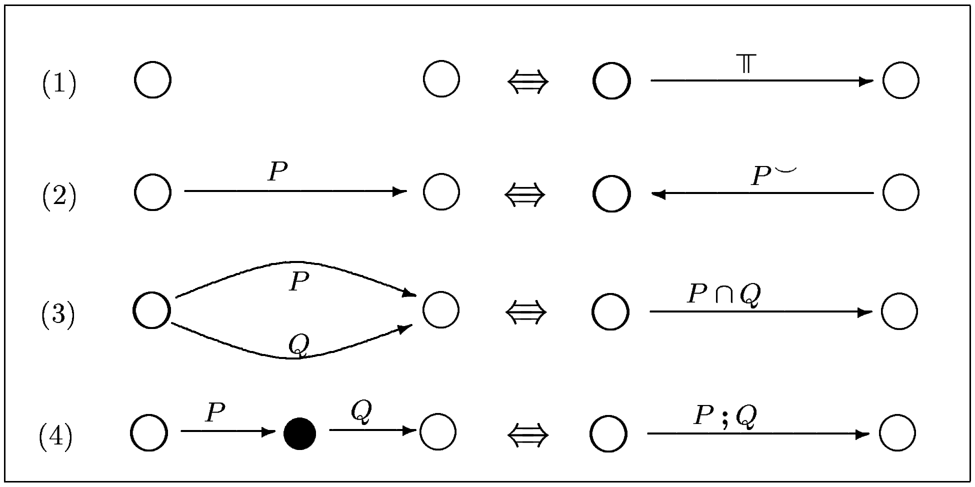


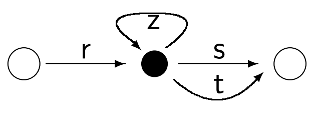
Figure 2. Some simple rewriting rules

These basic actions can be packaged into relatively complex transformation rules, tactics, and even algorithms of some sophistication, which preserve the meaning of the representation. At the lowest level one may place, e.g.: a rule that shifts, in a single move, several edges attached to one extreme of an edge labeled $P \cap \mathbb{I}$ to the other extreme; a rule that converts $[\nu, P \cap \mathbb{I}, \nu]$ (resp., $[\nu, P ; \mathbb{I}, \nu']$) into $[\nu, P, \nu]$ (resp., $[\nu, P, \nu']$); one that converts $[\nu, P ; Q^{\smile}, \nu]$ into $[\nu, P \cap Q, \nu']$ where ν' is new and bound; etc.

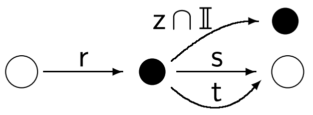
At a slightly higher level, one can eliminate multiple labeled edges $[\nu, P, \nu']$ sharing the same two endpoints ν, ν', through systematic use of action 3, thus reducing the multi-graph to a graph proper. At the same level, one can eliminate all loop-edges $[\nu, P, \nu]$ by introducing, for each of them, a new bound node ν' along with an edge $[\nu, P \cap \mathbb{I}, \nu']$.

Example 2.1 Consider the following graphs:

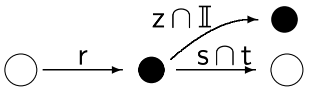
[10] As usual, the degree of a node is the number of edges incident on it.



(a)



(b)



(c)

The graph (a) can be transformed first into the loop-free multi-graph (b) and then into the graph (c). □

A further level up, one has an algorithm for associating a planar (multi-)graph G to a given expression P of the calculus of relations. Two designated nodes s_0 and s_1, named *source* and *sink*, will represent the two arguments of P, and every node distinct from these two will be regarded as being bound.

Algorithm. (Graph fattening) Given P, one proceeds non-deterministically to construct G, s_0, s_1, as follows: either

- G consists of a single edge, labeled P, leading from s_0 to s_1; or
- P has the form $Q^{\smile}$, and G, s_1, s_0 (with source and sink interchanged) represents Q; or
- P is of the form $Q{;}R$, the disjoint graphs G', s_0, s'_2 and G'', s''_2, s_1 represent Q and R respectively, and one obtains G by combination of G' with G'' by 'gluing' s''_2 onto s'_2 to form a single node; or
- P is of the form $Q \cap R$, the disjoint graphs G', s'_0, s'_1 and G'', s''_0, s''_1 represent Q and R respectively, and one obtains G from G' and G'' by gluing s''_0 onto s'_0 to form s_0 and by gluing s''_1 onto s'_1 to form s_1.

(The name of this algorithm refers to the possible choice of resorting to the first alternative only when no other alternative is viable, so that the 'fattest' possible graph is obtained.) □

As an additional convention related to this algorithm, one can either [11]

∀∀: label both s_0 and s_1 by ∀, to convert a representation G, s_0, s_1 of P into a representation of the equality $P = \mathbb{T}$ (which corresponds to the first-order sentence $(\forall x)(\forall y)(xPy)$); or

∀∃: label the source by ∀ and the sink by ∃, to represent the statement $\mathsf{Total}(\,P\,)$, which is a short for $P{;}\mathbb{T} = \mathbb{T}$ (i.e., $(\forall x)(\exists y)(xPy)$); or

∃∃: label both s_0 and s_1 by ∃, to represent the inequality $P \neq \mathbb{\perp}$, which is a short for the equality $\mathsf{Total}(\,\mathbb{T}{;}P\,)$ (i.e., $(\exists x)(\exists y)(xPy)$); or

¬∃∃: label the source by ¬∃ and the sink by ∃, to represent the equality $P = \mathbb{\perp}$ (i.e., $\neg(\exists x)(\exists y)(xPy)$).

[11] Graphs with source and sink labeled ∃ and ∀ respectively will not be treated, and they do not seem to fit well in our framework. Currently, we see them as unstable structures that immediately decay into ∀∃-graphs (by interchange of the source with the sink), with considerable loss of information.

Thus, for example, the following graph states that f fulfills both $\mathbb{I} \subseteq \mathsf{f};\mathsf{f}$ and $\mathsf{f}\cap\mathbb{I} = \mathbb{\perp}$:

$$\begin{tikzcd}
& \bullet \arrow[dr, "\mathsf{f}"] & & \\
\textcircled{$\forall$} \arrow[ur, "\mathsf{f}"] \arrow[rr, "\mathbb{I}"] & & \bullet \arrow[r, "\mathbb{I}\cap\overline{\mathsf{f}}"] & \textcircled{$\exists$}
\end{tikzcd}$$

As the example shows, we represent bound nodes by black circles. An edge is usually represented in drawings by a solid arrow, with the label written next to it. *When the arrow is dotted, the associated label $\overline{P}$ is denoted simply as P.* Simple graph-rewriting rules related to this convention are:

$$\boxed{\begin{tikzcd}
(1) & \bigcirc \arrow[r, "P"] & \bigcirc & \Leftrightarrow & \bigcirc \arrow[r, dotted, "\overline{P}"] & \bigcirc \\
(2) & \bigcirc \arrow[r, "\overline{P}"] & \bigcirc & \Leftrightarrow & \bigcirc \arrow[r, dotted, "P"] & \bigcirc
\end{tikzcd}}$$

Remark 2.2 Obvious rules enabling one to displace source and sink are:

– In a graph of type $\exists\exists$ or $\neg\exists\exists$, the roles of source and sink can indifferently be played by any two distinct nodes;

– in a graph of type $\forall\forall$, source and sink can swap their roles;

– in a graph of type $\forall\exists$, the role of sink can be played by any node distinct from the source, e.g.,

$$\begin{tikzcd}
\textcircled{$\forall$} \arrow[r, "P"] \arrow[rr, bend right, "R"'] & \bullet \arrow[r, "Q"] & \textcircled{$\exists$} & \Leftrightarrow & \textcircled{$\forall$} \arrow[r, "P"] \arrow[rr, bend right, "R"'] & \textcircled{$\exists$} \arrow[r, "Q"] & \bullet
\end{tikzcd}$$

□

The above discussion does not address issues related to the operators $\triangle$ and $\cup$. Concerning these constructs, we only marginally mention two rules to which it will at times useful to resort (perhaps tacitly):

(8) Suppose there is an edge $[\nu, P, \nu']$. Then any edge $[\nu, P, \nu'']$ with $\nu' \not\equiv \nu''$ can be replaced by $[\nu, \mathsf{tot}(\,P\,), \nu'']$, i.e. by $[\nu, P\triangle\overline{P;\mathbb{T}}, \nu'']$, and conversely.

(9) If $[\nu', P\cup Q, \nu]$, $[\nu, R, \nu'']$ are the only edges involving the bound node ν, they can be replaced by an edge $[\nu', P;R\cup Q;R, \nu'']$; and conversely.

3 Translating first-order logic into the calculus of relations

As is well known (cf. [41]), a sentence α of dyadic first-order logic can be translated into a ground (relational) equation if and only if α is logically equivalent to a sentence involving at most three distinct variables. This characterization of translatable sentences is, alas, not very useful in practice: establishing whether a given α belongs to this collection is in fact an undecidable problem (cf. [33]). Notwithstanding, conservative translation techniques can be devised to partially solve the problem. One such technique, originally described in [18], is recalled here for ease of the reader:

Algorithm. (Graph thinning) An existentially quantified conjunction φ of literals of the form xPy is given (cf. Sec. 2). The goal is to find a quantifier-free conjunction —or simply an atom, if there are at most two free variables in φ— equivalent to φ. Initially, a directed and labeled multi-graph G_φ representing φ by the usual conventions is built up, then it is normalized by elimination of loop-edges, and finally it is rendered a graph by fusion of multiple edges between the same nodes (cf. Example 2.1).

This G_φ and its labels will be manipulated as stated below, with the aim of eliminating as many bound nodes as possible. This elimination (which represents the elimination of existential quantifiers from φ) is performed by repeatedly applying two graph-transformation rules (see also Fig. 3):

bypass rule. Let ν be a bound node with degree 2 and let $[\nu', P, \nu]$ and $[\nu, Q, \nu'']$ be the edges adjacent to it, suitably re-oriented (by rule 2. of Sec. 2) so that the former enters and the latter leaves ν. Then the node ν and its edges are removed, and a new labeled edge, $[\nu', P;Q, \nu'']$, takes their place in the graph. If an edge with endpoints ν', ν'' existed already, then, after being re-oriented to comply with the orientation $[\nu', \nu'']$, it gets fused with the new edge by the rule 3 of Sec. 2.

bigamy rule. The rule applies to a bound node ν having just one adjacent edge. Let ν, ν' be the endpoints of this edge, and assume there exist a node $\nu'' \not\equiv \nu$ and an edge with endpoints ν', ν''. Then, the bigamy rule behaves as if there were an edge $[\nu, \mathbb{T}, \nu'']$ labeled $\mathbb{T}$, performing bypass of the node ν.

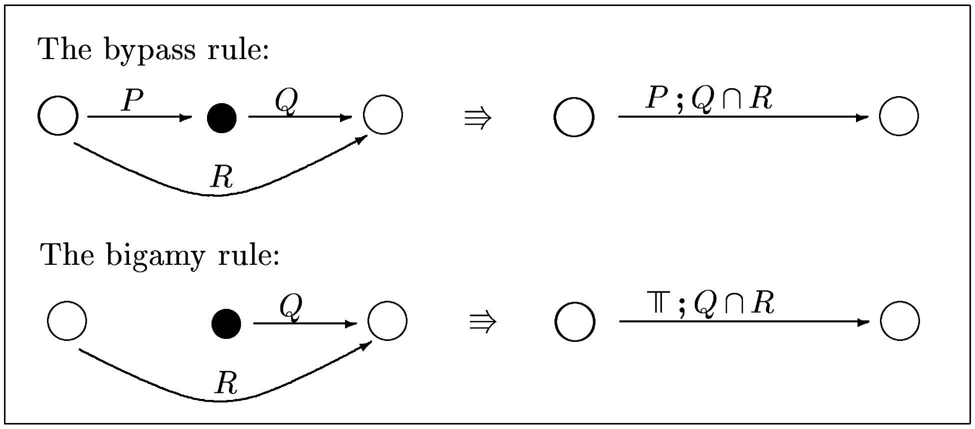


Figure 3. The bypass rule and the bigamy rule

The process ends when no further applications of the above rules can be carried out. If the resulting graph has no bound nodes of degree greater than 1, the sought conjunction can be directly read off the graph, else we have a failure. □

It has been proved that this algorithm has complexity $\mathcal{O}(|N_\varphi| \log |N_\varphi|)$

(where N_φ is the set of all nodes of G_φ), and that a crucial confluence property holds: the order in which bypass and bigamy actions are performed is immaterial [6].

In Sec. 6 we will briefly report on a 'graphical' implementation of this algorithm.

Let us see through two examples how the above-outlined algorithm works:

- the first example will refer to the theory of natural numbers with successor operation s (cf. [15,18]).
- the other example will refer to a theory of nested sets.

Example 3.1 Rules for evaluating $2^U = V$ and $2\,U = V$ in the theory of successor are:

$\mathsf{twoTo}(0, \mathsf{s}\,0)$
$\mathsf{twice}(\mathsf{s}\,X, \mathsf{s}\,\mathsf{s}\,Y) \leftarrow \mathsf{twice}(X, Y)$
$\mathsf{twice}(0, 0)$
$\mathsf{twoTo}(\mathsf{s}\,X, V) \leftarrow \mathsf{twoTo}(X, Z),\ \mathsf{twice}(Z, V)$

We begin by rectifying (cf. [42]) these Horn clauses into

$U\ \mathsf{twoTo}\ V \leftarrow U\ \mathsf{z}\ W,\ U\ \mathsf{s}\ V$
$U\ \mathsf{twoTo}\ V \leftarrow X\ \mathsf{s}\ U,\ X\ \mathsf{twoTo}\ Z,\ Z\ \mathsf{twice}\ V$
$U\ \mathsf{twice}\ V \leftarrow U\ \mathsf{z}\ V$
$U\ \mathsf{twice}\ V \leftarrow X\ \mathsf{s}\ U,\ Y\ \mathsf{s}\ W,\ W\ \mathsf{s}\ V,\ X\ \mathsf{twice}\ Y$

where z represents the predicate $\{[0, 0]\}$.
Through graph-thinning, one easily obtains corresponding inclusions:

$$\mathsf{z} \,;\, \mathbb{T} \cap \mathsf{s} \subseteq \mathsf{twoTo}, \qquad \mathsf{z} \subseteq \mathsf{twice},$$

$$\mathsf{s}^{\smile} \,;\, \mathsf{twoTo} \,;\, \mathsf{twice} \subseteq \mathsf{twoTo}, \qquad \mathsf{s}^{\smile} \,;\, \mathsf{twice} \,;\, \mathsf{s} \,;\, \mathsf{s} \subseteq \mathsf{twice},$$

Fig. 4 zooms in on the details of the translation of the two clauses regarding twoTo.
Then we can condense the inclusions into equalities:

$$\mathsf{twoTo} = \mathsf{z} \,;\, \mathbb{T} \cap \mathsf{s} \bigtriangleup \mathsf{s}^{\smile} \,;\, \mathsf{twoTo} \,;\, \mathsf{twice};$$

$$\mathsf{twice} = \mathsf{z} \bigtriangleup \mathsf{s}^{\smile} \,;\, \mathsf{twice} \,;\, \mathsf{s} \,;\, \mathsf{s}.$$

(We are replacing, e.g., the inclusion $\mathsf{z} \,;\, \mathbb{T} \cap \mathsf{s} \cup \mathsf{s}^{\smile} \,;\, \mathsf{twoTo} \,;\, \mathsf{twice} \subseteq \mathsf{twoTo}$ by the former equality by the closed-world assumption [40] and by virtue of the disjointness of the operands of $\cup$.) □

Example 3.2 In set theory, the possibility to build the pair

$$\big\{\, Y \setminus \{X\},\ Y \cup \{X\} \,\big\}$$

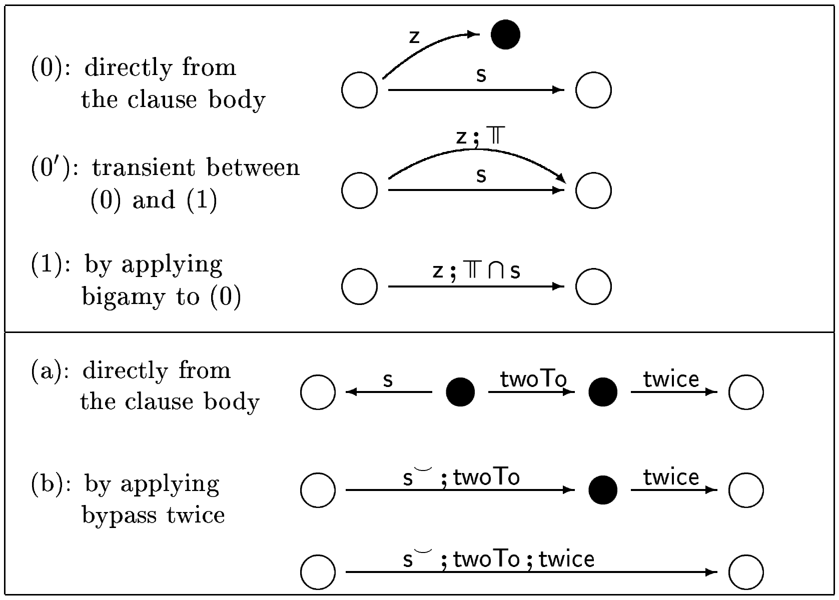
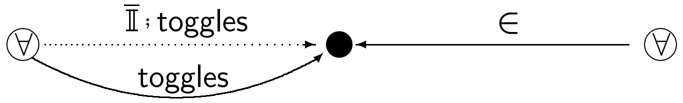

(0): directly from the clause body
z
s
(0′): transient between (0) and (1)
z ; 𝕋
s
(1): by applying bigamy to (0)
z ; 𝕋 ∩ s
(a): directly from the clause body
s
twoTo
twice
(b): by applying bypass twice
s˘ ; twoTo
twice
s˘ ; twoTo ; twice

Figure 4. Steps of the translation of the clauses defining twoTo (cf. Example 3.1)

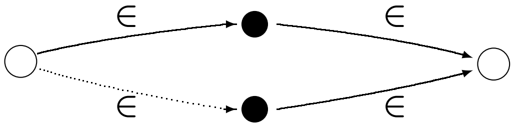
out of given sets X and Y can be formulated by the sentence

$$(1)\qquad \exists\, d\,\Big(\,Y{\in}d{\wedge}\forall\, u\,(\,u{=}X{\leftrightarrow}\exists\, v\,\exists\, w\,(\,u{\in}v{\in}d{\wedge}u{\notin}w{\in}d\,)\,)\,\Big)\,.$$

This statement is rendered by the graph



inside which one has the following ‘grafting’ of toggles:



Here the expressions on the dotted edges must be complemented, so that two bypass actions give us that toggles $\equiv_{\mathrm{Def}}$ $\in;\in\cap\notin;\in$. Then the translation of (1) turns out to be $(\,\mathsf{toggles} - \overline{\mathbb{I}}\,;\mathsf{toggles}\,)\,;\in^{\smile}\,=\,\mathbb{T}$. □

4 Inferring equalities within the calculus of relations

Deriving new equalities from proper axioms, and from laws already known, can to a large extent be viewed as a graph-rewriting activity. W.l.o.g. (cf. Fig. 5(1),(2)), let us assume momentarily that only existential graphs of the two types ∀∀ and ¬∃∃, respectively called *positive* and *negative* graphs, are exploited to represent premises and conclusions, known laws, theses, etc.

From this perspective, inference mechanisms are seen as graph-rewriting rules and techniques: in forward reasoning, they are used to transform premises into conclusions; in backward reasoning, to reduce theses —‘goals’, as they are often called— into simpler goals and, ultimately, into known and perhaps obvious facts.

As a basic principle, it is legitimate to replace a positive goal by a more demanding one, and a negative goal by one less demanding. E.g., new labeled edges can be added at will to a positive goal, whereas edges can be removed from a negative goal. Solving the new goal, although not necessarily equivalent to solving the previous goal, will in fact suffice for the purpose. Quite often a negative premise represents an inclusion $P \subseteq Q$; therefore, if a subgraph of a positive goal matches the part of the premise which represents Q, then it can be replaced by the part representing P; in a negative goal, on the opposite, Q may replace P (cf. Fig. 5(6)).

The following rule is conceived of in the same frame of mind:

- Let a $\forall\forall$-premise G be fully decomposable into subgraphs G_0, G_1 such that the source ν is an articulation node between the two, and the sink ν' belongs to G_1. Then one can infer any $\forall\exists$-graph obtained from G by gluing the sink ν' onto any node of G_0 (even onto ν), and by choosing as new sink any node distinct from the source (cf., e.g., Fig. 5(5)).

Without entering into further detail on such generalities, we limit ourselves to specifying a few inference rules in the form of graph-rewriting rules in Fig. 5 (cf. also Figures 6 and 7).

Example 4.1 The authors assessed the power of the above-proposed approach by means of a thorough set-theoretic case-study (cf. [7]): under weak set-theoretic axioms, including (1) of Example 3.2 and the extensionality axiom $\overline{\in^{\smile} ; \notin} - \notin^{\smile} ; \in \subseteq \mathbb{I}$ (stating that 'sets are the same whose elements are the same'), they proved that specific relations $\boldsymbol{\lambda}, \boldsymbol{\rho}$ designate conjugated 'quasi-projections', in the sense which will be clarified in Sec. 5. The proof was certified by an automated (first-order) theorem-prover whose autonomy was not so high as to exempt the authors from providing many hints. The graphical approach was consistently and quite effectively exploited to obtain the proof outline needed to guide the prover. As a matter of fact, heuristic insight in devising proofs within the calculus of relations played a crucial role. $\Box$

5 Defining operations over sets

Set-abstraction terms are, by common usage, expressions of the form $\{ t \mid \varphi \}$, where t and φ are a term and a formula in a first-order language suitable for the formalization of set theories. To simplify matters, let us assume here that t be a variable u, and that at most one variable x distinct from u may occur free in φ.

Suppose the only basic constructs available (in addition to the standard first-order endowment of logical symbols, inclusive of =) are the membership relator $\in$ and two monadic function symbols $\boldsymbol{\lambda}, \boldsymbol{\rho}$. At the very least, the set-theoretic axioms must to ensure that the four laws

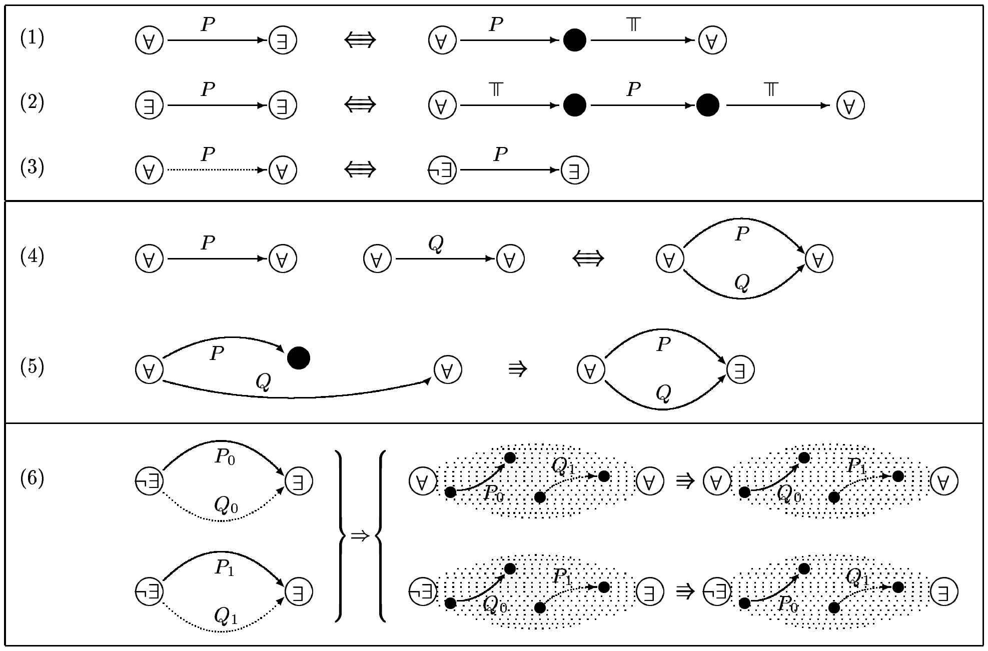


Figure 5. Graphic representation of various inference macros

(Pair) $\boldsymbol{\lambda}^{\smile};\boldsymbol{\rho} = \mathbb{T}$, $\mathsf{Func}(\boldsymbol{\lambda})$, $\mathsf{Func}(\boldsymbol{\rho})$, $\in;\ni = \mathbb{T}$.

hold, where $\ni \;\equiv_{\text{Def}} \in^{\smile}$. (We may summarize $\textbf{(Pair)}_{1,2,3}$ by saying that $\boldsymbol{\lambda}, \boldsymbol{\rho}$ are conjugated quasi-projections.) Further axioms of the Zermelo-Fraenkel theory ZF (cf. [32]), such as *extensionality* (cf. Example 4.1), will be brought into play as the opportunity will arise.

The meaning of $\{u \mid \varphi\}$ is conveyed by the double implication $u \in \{u \mid \varphi\} \leftrightarrow \varphi$, generalizable into $s \in \{u \mid \varphi\} \leftrightarrow \varphi[s/u]$. One cannot admit that $\{u \mid \varphi\}$ always designates a set; e.g., by assuming that $\{u \mid u \notin u\}$ exists, one would incur the well-known Russell's antinomy

$$\{u \mid u \notin u\} \in \{u \mid u \notin u\} \leftrightarrow \{u \mid u \notin u\} \notin \{u \mid u \notin u\}.$$

On the other hand, one can peacefully assume that $\{u \mid u \in x \wedge \psi\}$ always designates a set.

The issue which will be addressed in this section is: given a formula φ (within which x generally occurs free, along with u) how can one recognize that an abstraction term $\{u \mid \varphi\}$ designates a set, for every x? Otherwise stated: how can one establish, for such a given φ, the totality of the relation holding between x and y iff $\forall u(\varphi \leftrightarrow u \in y)$? When this relation is total, then by extensionality there will be exactly one y corresponding to each x, and hence the 'abstractor' $\{u \mid \varphi\}$ will define an operation over the universe of all sets: this is why the question we have raised deserves some interest (see also [22], and [2]).

We find it comfortable to address the question in the framework of our calculus, where the role of abstraction terms will be played by relational expressions of the form

$$\mathcal{F}(P) \;\equiv_{\text{Def}}\; \partial(P) - \overline{P};\in,$$

in which $\partial(\,P\,) \equiv_{\text{Def}} \overline{P;\not\in}$.
Thus we have

$$\begin{aligned} x\,\mathcal{F}(\,P\,)\,y &\leftrightarrow (\forall u\,(\,x\,P\,u \rightarrow u{\in}y\,) \wedge \forall u\,(\,u{\in}y \rightarrow x\,P\,u\,)\,) \\ &\leftrightarrow \{\,u \mid x\,P\,u\,\}{=}y. \end{aligned}$$

Tricks to prove the totality of relations

By studying numerous cases, we have discovered the following fundamental tactic rules to obtain equalities of the form $\mathsf{Total}(\,T\,)$, i.e. $T;\mathbb{T} = \mathbb{T}$, from the axioms of Fig. 1 enhanced by the *split rule*[12] $P = \mathbb{L} \vee \mathbb{T};P;\mathbb{T} = \mathbb{T}$ and by proper axioms—set-theoretic axioms in our privileged scenario of case-studies.

T0: $\mathsf{tot}(\,T\,)$, i.e. $T \triangle \overline{T;\mathbb{T}}$, is total for any relational expression T.

T1: If there is a Q such that both $Q{-}T = \mathbb{L}$ and $\mathsf{Total}(\,Q\,)$ are derivable, then $\mathsf{Total}(\,T\,)$ holds.

T2: If there is a Q such that $\mathsf{Total}(\,T;Q\,)$ is derivable, then $\mathsf{Total}(\,T\,)$ holds.

T3: If there are P, Q, R_0, R_1 such that $T = P;R_0 \cap Q;R_1$, $\mathsf{Total}(\,P\,)$, $\mathsf{Total}(\,Q\,)$, and $R_0;R_1^{\smile} = \mathbb{T}$ are derivable, one can conclude that $\mathsf{Total}(\,T\,)$.

T4: It can be assumed that either $\mathsf{Total}(\,P\,)$ or $\mathsf{Total}(\,\overline{\mathbb{T};P^{\smile}}\,)$ holds, for any relational expression P. □

These rules, depicted in Fig. 6, with the aid of similar ones related to inclusion —see Fig. 7—, easily yield additional tactic rules, e.g.,

T0′: Both $\mathbb{T}$ and $\mathbb{I}$ are total.

T3′: By singling out P, R such that $T = P;R$, $\mathsf{Total}(\,P\,)$, and $\mathsf{Total}(\,R\,)$ are derivable, one can conclude that $\mathsf{Total}(\,T\,)$.

T4′: Either $\mathsf{Total}(\,\overline{Q}\,)$ or $\mathsf{Total}(\,Q^{\smile}\,)$ can be assumed, for any relational expression Q.

T5′: When $T \cap T^{\smile} = \mathbb{L}$ is known to hold, one can conclude that $\mathsf{Total}(\,\overline{T}\,)$.

□

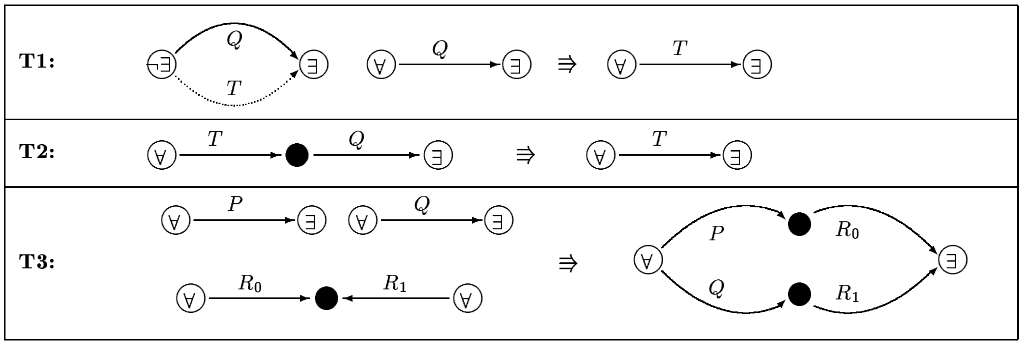


Figure 6. Inference rules for totality of relations

[12] Although inessential, the split rule plays at times a useful technical role.

Examples 5.1

(i) To obtain $\mathsf{Total}(\in)$ from **(Pair)**, we reduce it to $\mathsf{Total}(\in;\mathbb{T})$ through **T2**, then to $\mathsf{Total}(\in;\ni)$ through **T1**, in view of the obvious equality $\in;\ni-\in;\mathbb{T}=\perp\!\!\!\perp$. Since $\in;\ni=\mathbb{T}$ holds by $\mathbf{(Pair)}_4$, we conclude as desired with **T0′**.

(ii) To obtain $\mathsf{Total}(\partial(P\cup Q))$ from $\mathsf{Func}(P)$, $\mathsf{Func}(Q)$, and $\in;\ni=\mathbb{T}$, we reduce it to $\mathsf{Total}(\mathsf{tot}(P);\in\cap\,\mathsf{tot}(Q);\in)$ through **T1**, after verifying the inclusion $\mathsf{tot}(P);\in\cap\mathsf{tot}(Q);\in-\partial(P\cup Q)=\perp\!\!\!\perp$. Since both $\mathsf{Total}(\mathsf{tot}(P))$ and $\mathsf{Total}(\mathsf{tot}(Q))$ hold by **T0**, the desired goal is reached through **T4**, by $\in;\ni=\mathbb{T}$. As a special case, we get $\mathsf{Total}(\partial(\boldsymbol{\lambda}\cup\boldsymbol{\rho}))$ from $\mathbf{(Pair)}_{2,3,4}$.

(iii) To derive $\mathsf{Total}(\partial(\boldsymbol{\lambda};\ni\cup\boldsymbol{\rho}))$ from **(Pair)** taken along with the sum-set axiom of ZF, which is statable as $\mathsf{Total}(\partial(\ni;\ni))$ (cf. [17]), we reduce it to $\mathsf{Total}((\mathsf{tot}(\boldsymbol{\lambda});\in\cap\mathsf{tot}(\boldsymbol{\rho});\in;\in);\partial(\ni;\ni))$ through **T1**, after verifying the inclusion $(\mathsf{tot}(\boldsymbol{\lambda});\in\cap\mathsf{tot}(\boldsymbol{\rho});\in;\in);\partial(\ni;\ni)-\partial(\boldsymbol{\lambda};\ni\cup\boldsymbol{\rho})=\perp\!\!\!\perp$. Since $\mathsf{Total}(\mathsf{tot}(\boldsymbol{\lambda}))$ and $\mathsf{Total}(\mathsf{tot}(\boldsymbol{\rho}))$ hold by **T0**, and $\mathsf{Total}(\in)$ holds (by (1) above), we get $\mathsf{Total}(\mathsf{tot}(\boldsymbol{\rho});\in)$ by **T3′**, and hence
$$\mathsf{Total}(\mathsf{tot}(\boldsymbol{\lambda});\in\cap\mathsf{tot}(\boldsymbol{\rho});\in;\in),$$
thanks to **T3** and to $\in;\ni=\mathbb{T}$. We can conclude as desired, by **T3′**.

(iv) If we assume P and $\mathcal{F}(T)$ to be total, then from the axiom $\mathbf{(Pair)}_1$ we get the totality of $(P;\boldsymbol{\lambda}^{\smile}\cap\boldsymbol{\rho}^{\smile});\mathcal{F}(T)$ by exploiting the known fact $\boldsymbol{\rho}^{\smile}=\mathbb{I};\boldsymbol{\rho}^{\smile}$ and resorting to **T0′**, **T3**, and **T3′**.

(v) To see how **T4′** is derived from earlier tactics, we can proceed as follows. We know from **T4** that either $\mathsf{Total}(\overline{Q})$ or $\mathsf{Total}\left(\overline{\mathbb{T};\overline{Q}^{\smile}}\right)$ holds. If it is the latter alternative that holds, then $\overline{\mathbb{I};\overline{Q}^{\smile}}$ holds by **T1**, thanks to the easily verified inclusion $\overline{\mathbb{T};R}-\overline{\mathbb{I};R}=\perp\!\!\!\perp$. Since $\overline{\mathbb{I};\overline{Q}^{\smile}}=\overline{\overline{Q}^{\smile}}=\overline{\overline{Q^{\smile}}}=Q^{\smile}$, we conclude with **T4′**.

(vi) To see how **T5′** is derived from earlier tactics, notice that either $\mathsf{Total}(\overline{T})$ or $\mathsf{Total}(T^{\smile})$ holds by **T4′**. If it is the latter alternative that holds, the tactic **T1** gives us the totality of $\overline{T}$ anyhow, from the assumption $T\cap T^{\smile}=\perp\!\!\!\perp$ which yields $T^{\smile}-\overline{T}=\perp\!\!\!\perp$. As a special case, we get the totality of $\notin$ in ZF, by the consequence $\in\cap\ni=\perp\!\!\!\perp$ of the postulated well-foundedness of membership. □

A strategy to define operations on sets

Let us now focus on the following *subset axioms* (also known as 'separation' axiom scheme) of ZF:

(S) $$\mathsf{Total}(\mathcal{F}(\boldsymbol{\lambda};\ni\cap\boldsymbol{\rho};S)).$$

This states that for every ordered pair $x=\langle x_0,x_1\rangle$ there exists a set $\{u\in x_0 \mid x_1\,S\,u\}$. We will discuss in the ongoing a versatile *proof strategy* for verifying theses of the form $\mathsf{Total}(\mathcal{F}(R))$. The strategy consists in singling

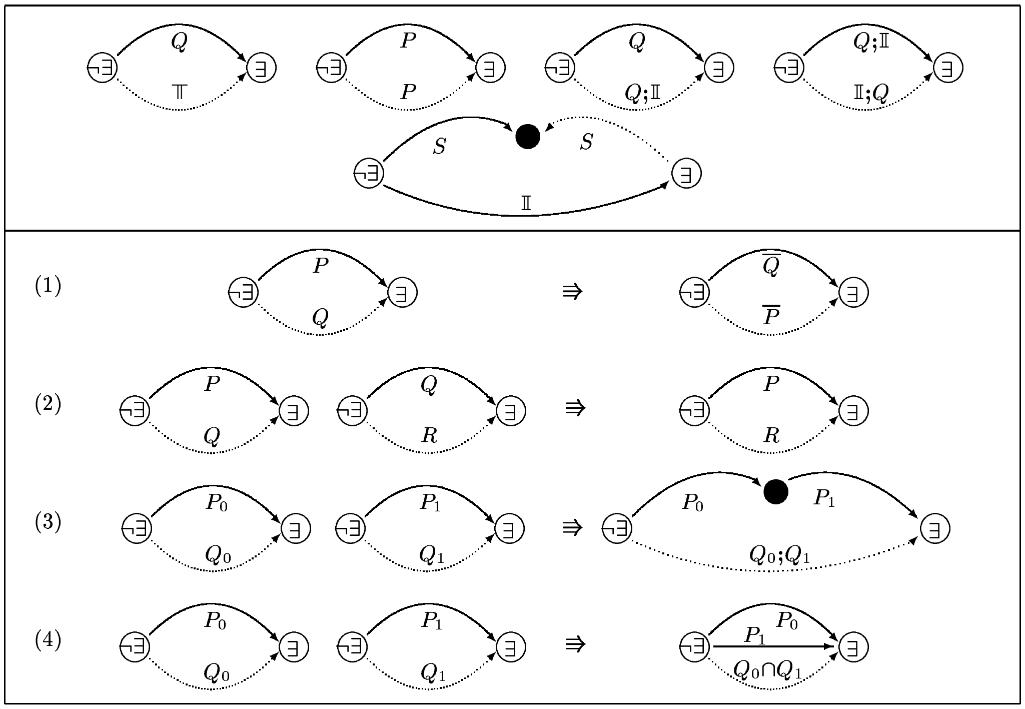
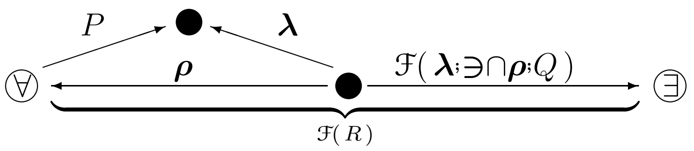


Figure 7. Five axioms and four inference rules for inclusion of relations

out relational expressions P, Q such that both of

$$\mathsf{Total}(\,P\,) \quad \text{and} \quad \mathcal{F}(\,R\,) \;=\; (\,P{;}\boldsymbol{\lambda}^{\smile}\cap\boldsymbol{\rho}^{\smile}\,)\,{;}\,\mathcal{F}(\,\boldsymbol{\lambda}{;}\ni\cap\boldsymbol{\rho}{;}Q\,)$$

are equalities easily derivable from the axioms. Graphically, the decomposition of $\mathcal{F}(\,R\,)$ together of the totality thesis involved in it, can be rendered as follows:



The soundness of this strategy under **(S)** ensues from the analysis carried out in Example 5.1(iv).

For the choice of Q, in tuning the strategy to different situations, we will adopt one of the tactic rules below. One of them (the most obvious, and first in the list that follows) turns out to work in the totality of cases; the others —when applicable— are syntactically simpler:

***Q*∞ :** Put $Q \equiv R$.

Q1: If $R \equiv T_0\cap T_1$, and P is fixed so as to fulfill $P{;}\ni{-}T_0 = \perp\!\!\!\perp$, put $Q \equiv T_1$.

Q2: If $R \equiv T_0{;}\ni\cap T_1$, and a P of the form $\mathsf{tot}(\,T_0\,)$ is taken, put $Q \equiv T_0{;}\mathbb{T}\cap T_1$.
□

When no specific indication is given on the tactic for choosing Q, the choice **Q∞** is understood.

Let us now come to tactic rules for choosing P in our proof strategy regarding ‘equalities’ of the form $\mathsf{Total}(\,\mathcal{F}(\,R\,)\,)$.

P1: Single out a (total) P such that the equality $P\cap R;\not\in = \perp\!\!\!\perp$, or (equivalently) $R\cap P;\not\ni = \perp\!\!\!\perp$ is derivable.

P2: Single out a P and a T such that the equalities $\mathsf{Total}(\,P\,)$, $P-\partial(\,T\,) = \perp\!\!\!\perp$ and $R-T = \perp\!\!\!\perp$ are derivable.
(The explanation why this works is that $\mathsf{Total}(\,\partial(\,T\,)\,)$, and hence that $\partial(\,T\,)\cap R;\not\in = \perp\!\!\!\perp$, follows from $\mathsf{Total}(\,P\,)$; thus we fall under the tactic **P1**).

Specializations of the latter tactic, simply consist in either

P2.a: taking a $P \equiv \partial(\,T\,)$ such that the equalities $\mathsf{Total}(\,\partial(\,T\,)\,)$ and $R-T = \perp\!\!\!\perp$ both are derivable; or

P2.b: taking a $P \equiv \mathcal{F}(\,T\,)$ such that the equalities $\mathsf{Total}(\,\mathcal{F}(\,T\,)\,)$ and $R-T = \perp\!\!\!\perp$ both are derivable.

Examples 5.2

(i) The existence of the null set (i.e., devoid of elements) can be stated as $\mathsf{Total}(\,\mathcal{F}(\,\perp\!\!\!\perp\,)\,)$. This can be proved without any particular strategy—indeed, **(S)** with $S \equiv \perp\!\!\!\perp$ directly supplies the desired thesis.

(ii) The totality, $\mathsf{Total}(\,\mathcal{F}(\,\mathbb{I}\,)\,)$, of the operation $x \mapsto \{x\}$, can be proved by taking $P \equiv \in$ (cf. Example 5.1(i)) by the tactic **P1**: in fact $\in\cap\mathbb{I};\not\in = \perp\!\!\!\perp$ obviously holds.

(iii) In order to prove that $\mathsf{Total}(\,\mathcal{F}(\,\mathsf{funcPart}(\,Z\,);\ni\cap T\,)\,)$ holds, it suffices to exploit the tactic **Q2** by taking $P \equiv \mathsf{tot}(\,\mathsf{funcPart}(\,Z\,)\,)$ and $Q \equiv \mathsf{funcPart}(\,Z\,);\mathbb{T}\cap T$.

(iv) Let us postulate that both $\mathsf{Total}(\,\partial(\,\ni;\ni\,)\,)$ and $\mathsf{Total}(\,\partial(\,\overline{\not\ni;\in}\,)\,)$ hold. These are weak formulations of the *sum-set* axiom and of the *power-set* axiom. By virtue of **(S)**, their stronger versions can be proved by the tactic **P2.a** (by taking $T \equiv \ni;\ni$ and $T \equiv \overline{\not\ni;\in}$) which yields $\mathsf{Total}(\,\mathcal{F}(\,\ni;\ni\,)\,)$ and $\mathsf{Total}(\,\mathcal{F}(\,\overline{\not\ni;\in}\,)\,)$, respectively.
Clearly, this argument can be applied whenever it is the case that $\mathsf{Total}(\,\partial(\,R\,)\,)$ holds and we want to prove that $\mathsf{Total}(\,\mathcal{F}(\,R\,)\,)$ holds too.

(v) Any attempt to prove that $\mathsf{Total}(\,\mathcal{F}(\,\mathbb{T}\,)\,)$, i.e., the existence of a set comprising every set as a member, must fail. In fact, from the existence of this omnicomprehensive set, the existence of the antinomic Russell’s set would follow: the latter could, in fact, be decomposed as
$$\mathcal{F}(\,\mathbb{T}(\,\mathbb{I}-\ni\,)\,) = (\,\mathcal{F}(\,\mathbb{T}\,);\boldsymbol{\lambda}^{\smile}\cap\boldsymbol{\rho}^{\smile}\,)\,;\,\mathcal{F}(\,\boldsymbol{\lambda};\ni\cap\mathbb{T};(\,\mathbb{I}-\ni\,)\,)\,,$$
where the totality of the third $\mathcal{F}(\,\cdot\,)$ ensues from the example in (3) above. □

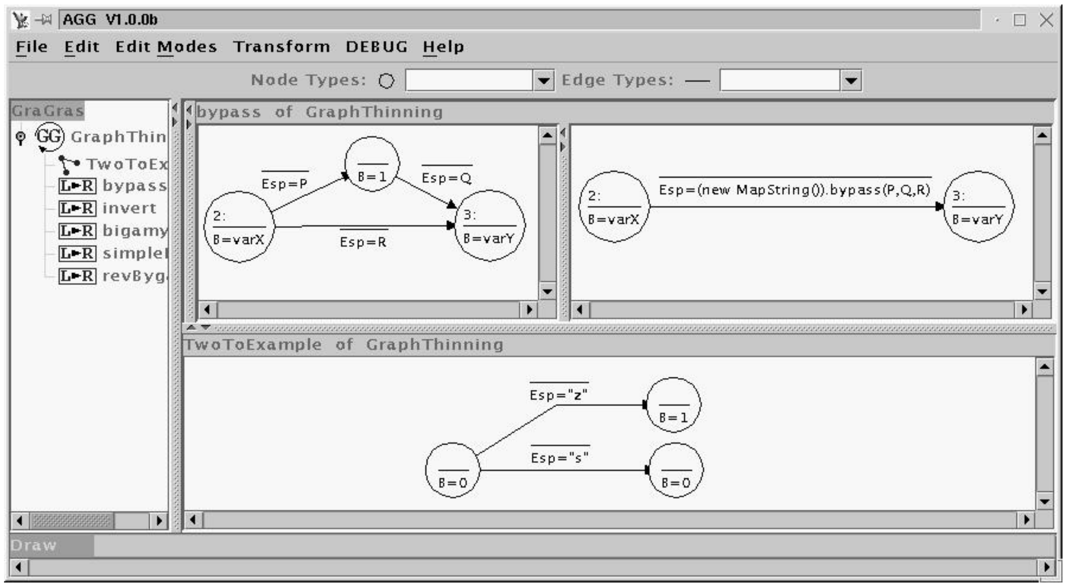


Figure 8. The AGG user-interface displaying the 'graph thinning' grammar

6 Towards a graphical relational reasoner

In this section we outline current activity aimed at putting to trial simple graphical techniques for relational reasoning. A more detailed description of this approach is provided in [19] by means of a number of worked examples.

The main goal of ours is implementing the graphical techniques described so far on top of an automated tool for algebraic transformation developed at the TU Berlin, namely AGG (Attributed Graph Grammar, see [16] for a detailed description).

Applications based on graph-transformations are described by Agg graph-grammars; the latter consist of a start graph (initializing the system) and a set of graph-rewriting rules describing the transformations which can be performed. The Agg environment supports visual manipulation of such graphs and rules. The start graph, as well as the graphs of the rules, may be attributed by Java objects (i.e., instances of Java classes, either loaded from standard libraries or user-defined) and expressions which are evaluated during rule applications. This allows for a powerful combination of visual and textual programming.

Agg has a formal foundation based on the single-pushout approach to graph transformation as introduced in [34]. The approach owes its name to the algebraic construction used to define the basic derivation step, which is modeled by a pushout in the category of graphs and partial graph morphisms (cf. [14]).

The user interacts with the Agg environment through a graphical user interface which provides several visual editors, and an interpreter: the graph-transformation machine. Fig. 8 shows the main window of the Agg user-interface. On the left, the current graph-grammar is visualized: the user can browse and inspect the rules and the start graph. The selected graph or rule is shown in the corresponding editor window, on the right: the upper editor displays the left- and right-hand sides of the rule, while the lower one displays the graph. A special attribute editor pops up whenever an object is selected for attribution.

Once a graph grammar has been formalized, Agg allows one to apply the graph transformation rules by providing two basic mechanisms: a) the user can select and apply one-by-one the rules; or b) Agg itself can perform automatically a complete run of transformations. In the latter case, the rules are selected following their order in the grammar; each single rule is repeatedly applied to the current graph, as much as possible, before the next one is taken into consideration.

The graph thinning algorithm has been implemented on top of Agg by specifying the graph grammar described in Sec. 3. In particular, Fig. 8 shows the Agg formalization of the bypass rule (cf. Fig. 3) and the start graph corresponding to the graph (0) in Fig. 4. The status of being a bound node is rendered in Agg by means of an attribute of the nodes (here, `B=1` means that the corresponding node is bound). Attribution of arcs is exploited to manage labeling by relational expressions (the name of the attribute is `Esp`, in Fig. 8, while expressions are represented by Java strings).

Consider the left-hand side of the displayed rule. Whenever the value of an attribute has to be accessed for further use —e.g. to perform further elaboration in order to instantiate the attributes of the right-hand side during rule application— a variable is employed (in our example the variables are `varX`, `varY`, `P`, `Q`, and `R`). Notice that the value of `Esp` in the right-hand side of the bypass rule is obtained by calling a suitable Java method.

Our implementation of the graph thinning algorithm works well in practice. Actually, the translations described in Sec. 3 were among those obtained by exploiting Agg.

At the present time, Agg can be employed profitably as a proof-assistant or just as a semi-automated theorem prover for graphical reasoning. This is so because the default strategy provided for rule selection/application does not permit easy implementation of the standard search methods commonly exploited in theorem proving. As a matter of fact, the realization of a depth-first iterative deepening strategy combined with a best-first heuristic search (see [31]) is one of the challenging goals of our current research.

We plan to accomplish this intent also in collaboration with the research group that developed Agg, which is currently implementing a parser for Agg graph grammars.

7 Related work

Several approaches to the automation of relational reasoning have been proposed. Various tools supporting algebraic logic exist already. We would like to mention, at least, RALF, Libra, and RELVIEW. RALF is basically a graphical interactive proof assistant and proof checker: it allows the user to manipulate relation-algebraic formulas mainly by using substitution of equals for equals, weakening and strengthening (cf. [25]). The RELVIEW system (cf. [1]) offers a support for relational computation: assuming finiteness of domains and relations, it offers explicit and extensional representation of concrete relations and provides efficient implementation of the basic relational constructs. The Libra language (Lazy Interpreter of Binary Relational Algebra [13]) is a general-purpose programming language based on the algebra of dyadic relations that offers immediate support to program specification.

A few graphical approaches to relational calculus have been proposed too. For instance, relational methods are exploited in [3,4] to tame the problem of circuit design. This goal is achieved by developing a pictorial representation of relational terms and by providing a (relational) semantics for pictures. High-level operations on pictures/circuits are rendered by transformation rules that ultimately correspond to the axioms/laws of the calculus.

In [12,23], a graphical representation by means of *diagrams* is proposed for all term-expressions of an equational theory of dyadic relations which does not involve complementation. A notion of reduction is given in terms of morphisms between diagrams. Normalization and decidability properties for this graphical framework are also provided.

Another graphical calculus for representation of and visual reasoning on mathematical formulas is proposed in [11]. In this approach the treatment of $\cap$ and $;$ essentially coincides with ours; moreover, [11] introduces a set of graphical tranformation rules which turns out to have a large overlap with the rules exploited in our Graph-fattening algorithm; however, [11] does not deal explicitly with complementation and adopts a different treatment of inclusion.

The work of Kahl (cf. [29,30]) provides a more general approach to the graphical calculi of relations introduced by [3] and [11], by resorting to algebraic graph-rewriting techniques and concepts [10,14,34].

Acknowledgments

We are grateful to Olga Runge, Gabriele Taentzer, and Thorsten Schultzke for useful discussions and suggestions about the Agg system. We would like to thank Andrea Corradini, Wolfram Kahl, James Lipton, and Antonino Salibra for giving us many suggestions and useful references.

References

[1] Behnke, R., R. Berghammer and P. Schneider, *Machine support of relational computations: The Kiel RELVIEW system*, Tech. Rep. Bericht Nr. 9711, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, Kiel, Germany (1997).

[2] Belinfante, J. G. F., *Computer proofs in Gödel's class theory with equational definitions for composite and cross*, Journal of Automated Reasoning **22** (1999), pp. 311–339.

[3] Brown, C. and G. Hutton, *Categories, allegories and circuit design*, in: *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, Paris, France, 1994, pp. 372–381.

[4] Brown, C. and A. Jeffrey, *Allegories of circuits*, in: *Proc. Logic For Computer Science* (1994), pp. 56–68.

[5] Cantone, D., A. Cavarra and E. G. Omodeo, *On existentially quantified conjunctions of atomic formulae of $\mathcal{L}^+$*, in: M. P. Bonacina and U. Furbach, editors, *Proceedings of the FTP97 International workshop on first-order theorem proving*, 1997, pp. 45–52, RISC-Linz Report Series No.97-50.

[6] Cantone, D., A. Formisano, E. G. Omodeo and C. G. Zarba, *Compiling dyadic first-order specifications into map algebra*, in: *Proceedings, of the 16th Twente Workshop on Language Technology—2nd AMAST Workshop Algebraic Methods in Language Processing (AMILP 2000), TWLT 16*, University of Twente, 2000.

[7] Chiacchiaretta, A., A. Formisano and E. G. Omodeo, *Benchmark #1 for equational set theory*, in: Giornata "Analisi Sperimentale di Algoritmi per l'Intelligenza Artificiale", Roma, 1999.

[8] Chiacchiaretta, A., A. Formisano and E. G. Omodeo, *Map reasoning through existential multigraphs*, Tech. Rep. 05/00, Dipartimento di Matematica Pura ed Applicata, Università di L'Aquila (2000).

[9] Chin, L. H. and A. Tarski, *Distributive and modular laws in relation algebras*, University of California Publications in Mathematics **1** (1951), pp. 341–384, new series.

[10] Corradini, A., U. Montanari, F. Rossi, H. Ehrig, R. Heckel and M. Löwe, *Algebraic approaches to graph transformation I: Basic concepts and double pushout approach*, in: Rozenberg [37] pp. 163–246.

[11] Curtis, S. and G. Lowe, *Proofs with graphs*, Science of Computer Programming **26** (1996), pp. 197–216, mathematics of program construction, Kloster Irsee, 1995.

[12] Dougherty, D. and C. Gutiérrez, *Normal forms and reduction for theories of binary relations*, in: L. Bachmair, editor, *Rewriting Techniques and Applications, 11th International Conference, RTA2000, Norwich, UK, July 2000, Proc.*, LNCS **1833** (2000), pp. 95–109.

[13] Dwyer, B., *LIBRA: a Lazy Interpreter of Binary Relational Algebra*, Tech. Rep. 95-10, Department of Computer Science University of Adelaide, (1995).

[14] Ehrig, H., R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner and A. Corradini, *Algebraic approaches to graph transformation II: Single pushout approach and comparison with double pushout approach*, in: Rozenberg [37] pp. 247–312.

[15] Enderton, H. B., "A Mathematical Introduction to Logic," Academic Press, New York and London, 1972.

[16] Ermel, C., M. Rudolf and G. Taentzer, *The* AGG *approach: Language and environment*, in: H. Ehrig, G. Engels, H.-J. Kreowski and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation, vol. 2: Applications, Languages and Tools*, World Scientific, Singapore, 1999 .

[17] Formisano, A. and E. G. Omodeo, *An equational re-engineering of set theories*, in: R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, LNCS 1761 (LNAI) (2000), pp. 175–190.

[18] Formisano, A., E. G. Omodeo and M. Temperini, *Goals and benchmarks for automated map reasoning*, Journal of Symbolic Computation **29** (2000), special issue. M.-P. Bonacina and U. Furbach, editors.

[19] Formisano, A. and M. Simeoni, *Graphs and maps: rewriting techniques at work*, Tech. Rep. TU-Berlin 2001-01, Technische Universität Berlin, (2001).

[20] Gardner, M., "Logic machines and diagrams," The Harvester Press, 1982, 2^{nd} edition edition.

[21] Givant, S., *Tarski's Development of Logic and Mathematics based on the Calculus of Relations*, in H. Andréka, J. D. Monk, and I. Németi editors, *Algebraic Logic*, Colloquia Mathematica Societatis János Bolyai, vol. 54, pp. 189-216, North Holland, 1991.

[22] Gödel, K., "The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis with the Axioms of Set Theory," Princeton University Press, Princeton, New Jersey, 1940.

[23] Gutiérrez, C., "The arithmetic and geometry of allegories," Ph.D. thesis, Wesleyan University, Middletown, CT (1999).

[24] Hammer, E. M., *Peirce's Logic* (1999), in: E. N. Zalta, C. Allen, and U. Nodelman, editors, *Stanford Encyclopedia of Philosophy*, Stanford University, World Wide Web URL: http://plato.stanford.edu/.

[25] Hattensperger, C., R. Berghammer and G. Schmidt, *RALF - A relation-algebraic formula manipulation system and proof checker. Notes to a system demonstration*, in: M. Nivat, C. Rattray, T. Rus and G. Scollo, editors, *AMAST '93*, Workshops in Computing (1994), pp. 405–406.

[26] Houser, N., D. D. Roberts and J. V. Evra, editors, "Studies in the Logic of Charles Sanders Peirce," Indiana University Press, 1997.

[27] Jacobson, N., “Lectures in abstract algebra: I. basic concepts,” The University series in Higher Mathematics, Van Nostrand, 1951.

[28] Jónsson, B., *Varieties of relation algebras*, Algebra Universalis **15** (1982), pp. 273–298.

[29] Kahl, W., *Algebraic graph derivations for graphical calculi*, in: F. d’Amore, P. G. Franciosa and A. Marchetti-Spaccamela, editors, *Graph Theoretic Concepts in Computer Science, WG ’96*, LNCS **1197** (1997), pp. 224–238.

[30] Kahl, W., *Relational matching for graphical calculi of relations*, Information Sciences **119** (1999), pp. 253–273.

[31] Korf, R. E., *Depth-first iterative-deepening: An optimal admissible tree search*, Artificial Intelligence **27** (1985), pp. 97–109.

[32] Krivine, J.-L., “Introduction to axiomatic set theory,” Reidel, Dordrecht. Holland, 1971.

[33] Kwatinetz, M. K., “Problems of expressibility in finite languages,” Ph.D. thesis, University of California, Berkeley (1981).

[34] Löwe, M., *Algebraic approach to single-pushout graph transformation*, Theoretical Computer Science **109** (1993), pp. 181–224.

[35] Maddux, R. D., *The origin of relation algebras in the development and axiomatization of the calculus of relations*, Studia Logica **50** (1991), pp. 421–455.

[36] Maddux, R. D., *Relation-algebraic semantics*, Theoretical Computer Science **160** (1996), pp. 1–85.

[37] Rozenberg, G., editor, “Handbook of Graph Grammars and Computing by Graph Transformation. vol. I: Foundations,” World Scientific, 1997.

[38] Schmidt, G. and T. Ströhlein, “Relations and graphs,” Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, 1993.

[39] Schröder, E., “Vorlesungen über die Algebra der Logik (exakte Logik), vol.1–3” B. Teubner, Leipzig, 1891–95, [Reprinted by Chelsea Publishing Co., New York, 1966.].

[40] Shepherdson, J. C., *Negation as failure: A comparison of Clark’s completed data base and Reiter’s closed world assumption*, Journal of Logic Programming **1** (1984), pp. 51–79.

[41] Tarski, A. and S. Givant, “A formalization of Set Theory without variables,” Colloquium Publications **41**, American Mathematical Society, 1987.

[42] Ullman, J. D., “Database and Knowledge-base Systems, vol.1,” Principles of Computer Science **49**, Computer Science Press, Stanford University, 1988.