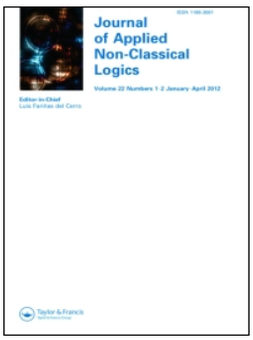

Journal of Applied Non-Classical Logics

ISSN: 1166-3081 (Print) 1958-5780 (Online) Journal homepage: https://www.tandfonline.com/loi/tncl20

An efficient relational deductive system for propositional non-classical logics

Andrea Formisano & Marianna Nicolosi-Asmundo

To cite this article: Andrea Formisano & Marianna Nicolosi-Asmundo (2006) An efficient relational deductive system for propositional non-classical logics, Journal of Applied Non-Classical Logics, 16:3-4, 367-408, DOI: 10.3166/jancl.16.367-408

To link to this article: https://doi.org/10.3166/jancl.16.367-408

An efficient relational deductive system for propositional non-classical logics

Andrea Formisano* — **Marianna Nicolosi-Asmundo****

* *Dipartimento di Informatica*
Università di L'Aquila
via Vetoio, loc. Coppito
I-67010 L'Aquila (Italy)
formisano@di.univaq.it

** *Dipartimento di Matematica e Informatica*
Università di Catania
viale A. Doria 6
I-95125 Catania (Italy)
nicolosi@dmi.unict.it

ABSTRACT. *We describe a relational framework that uniformly supports formalization and automated reasoning in varied propositional modal logics. The proof system we propose is a relational variant of the classical Rasiowa-Sikorski proof system. We introduce a compact graph-based representation of formulae and proofs supporting an efficient implementation of the basic inference engine, as well as of a number of refinements. Completeness and soundness results are shown and a Prolog implementation is described.*

KEYWORDS: *Rasiowa-Sikorski proof systems, relational reasoning, non-classical logics.*

1. Introduction

In the last decades, relational formalization of many non-classical propositional logics has been systematically and rigorously studied (see, for instance, [ORŁ 91, ORŁ 92, ORŁ 94, ORŁ 96]). Long-standing research and well established results indicate that standard relational structures can be considered as a common core supporting representations of many non-classical propositional logics.

This result suggests a uniform relational approach to non-classical automated reasoning which is complementary and alternative to common *ad hoc* direct inference methods (cf., e.g., [OHL 01, GOR 99]). The idea is to exploit a relational proof sys-

tem to verify that the relational rendering of a modal formula is logical consequence of the relational counterpart of some modal axioms.

In the relational framework several alternatives for automated deduction can be considered as viable. For instance, to mention some of the approaches proposed in the literature, we have tableaux systems [HEN 80], Gentzen-style systems [SCH 82, MAD 83], systems *à la* Rasiowa-Sikorski [ORŁ 97], display calculus [GOR 96], and equational proof systems [FOR 01b, FOR 01c].

This paper mainly focuses on Rasiowa-Sikorski systems [RAS 63]. More specifically, the research we are to describe consists in the realization of a relational Rasiowa-Sikorski system suitable to uniformly support modal reasoning for any relationally expressible propositional logic.

The basic constituent of a Rasiowa-Sikorski system is a collection of inference rules. Analogously to tableaux systems [DAG 99b], given a theorem to be proved, (the search for) a proof is developed through repeated applications of some *decomposition rules*. As a result, the solution of a problem is reduced to the solutions of syntactically simpler sub-problems. The proof is completed whenever a successful condition becomes satisfied. Success situations are detected by means of some *closure rules* that, together with the decomposition rules, characterize the proof system. Dually to tableaux, Rasiowa-Sikorski systems prove that a formula is valid, instead of unsatisfiable. For this reason they are sometimes referred to as *dual-tableaux* systems.

Let $\mathcal{L}$ be a non-classical logic that can be translated into the relational calculus. Proving a modal theorem φ of $\mathcal{L}$ amounts to prove the validity of its relational formulation $t_{\mathcal{L}}(\varphi)$ (i.e., that the relational expression $t_{\mathcal{L}}(\varphi)$ actually denotes the universal relation).

Different translations are needed to deal with different logics in order to suitably reflect, in the relational framework, the proper axioms that characterize each specific logic. This usually corresponds to identify a collection $\mathcal{C}$ of constant relations to be interpreted as relational counterparts of modal accessibility relations. Proper modal axioms are then rendered by imposing relational axioms which restrain the admitted interpretations of the constants in $\mathcal{C}$. In the context of a Rasiowa-Sikorski system this is dealt with by enriching the collection of inference rules of the system. In particular suitable specific decomposition rules or refined closure rules are added to the basic inference system.

Handling such issues is not always an easy task to tackle. In fact several syntactical and semantical aspects need to be taken into account. With regard to this, the duality result between tableaux and Rasiowa-Sikorski systems constituted a helpful support. In fact, in developing the system we are going to describe, we fruitfully adapted and combined several techniques and proof-strategies independently developed for tableaux systems, as well as a compact representation for relational expressions akin to Decision Diagrams. Such techniques, to the best of our knowledge, have never been combined together in the realization of a relational deductive framework.

Further, the choice of a suitable graph-based representation of formulae and proofs allows us to enrich the basic inferential apparatus by embedding in the system specific mechanisms to deal with semantical aspects, such as the handling of the equality relation as well as the properties of interpreted relational constants.

The paper is structured as follows. We first briefly introduce the relational framework, common to all the relational renderings of modal logics. Then, in Section 2.2, we illustrate the relational formalizations of various non-classical propositional logics. In Section 2.3 a syntactical, schematic classification of relational formulae in Smullyan's style is presented, to be used in Section 3 in order to uniformly and concisely describe the core of the relational deductive apparatus. Section 4 introduces a graph-based representation of proofs and formulae, alternative to the classical tree-based one. In Section 5 soundness and completeness of the relational Rasiowa-Sikorski proof system are proved for both the representations. Section 6 introduces some refinements in the graph-based representation, mainly aimed at efficiently dealing with equality and interpreted constants. Finally, in Section 7 we draw our conclusions, give some hints for future research, together with a brief discussion on related work.

2. A relational logic for propositional non-classical logics

In this section we briefly describe a generic relational logic based on algebras of relations constituting a common framework in which the relational rendering of many propositional modal logics is embedded (several examples of such logics are given in Section 2.2).

In such a framework, the logics are treated in a homogeneous way and share basic features: formulae and accessibility relations are represented as binary relations; relational constructs represent extensional and intensional modal operations; and the deductive apparatus corresponds to a relational calculus.

In order to describe such relational structure let us start by recalling some preliminary notions (see also [BRI 97, MAD 91]). Let $\boldsymbol{\mathcal{D}}$ be a non empty set. The full algebra of binary relations over $\boldsymbol{\mathcal{D}}$ is the structure

$$Re(\boldsymbol{\mathcal{D}}) = (\wp(\boldsymbol{\mathcal{D}} \times \boldsymbol{\mathcal{D}}), \overline{\ }, \cap, \cup, U, Z, ;, \dagger, ^{\smile}, I, D).$$

The elements of $Re(\boldsymbol{\mathcal{D}})$ are the subsets of $\boldsymbol{\mathcal{D}} \times \boldsymbol{\mathcal{D}}$. Among them we distinguish U (the universal relation), Z (the empty relation), I (the identity relation) and D (the diversity relation), whereas the operations (constructs) of $Re(\boldsymbol{\mathcal{D}})$ are $\cap$ (intersection), $\cup$ (union), ; (relational product), $\dagger$ (relational sum) and $^{\smile}$ (conversion). Further, $(\wp(\boldsymbol{\mathcal{D}} \times \boldsymbol{\mathcal{D}}), \overline{\ }, \cap, \cup, U, Z)$ is a Boolean algebra (being U and Z the top and bottom elements, respectively) and $(\wp(\boldsymbol{\mathcal{D}} \times \boldsymbol{\mathcal{D}}), ;, ^{\smile}, I)$ is a monoid with involution [BRI 97] (being I the identity element and $^{\smile}$ the involution). Notice that, for the sake of simplicity, we are considering as primitive, or *basic*, all the relational constructs $\overline{\ }, \cap, \cup, ;, \dagger$, and $^{\smile}$ as well as the constants U, Z, I, D. An alternative

possibility could consist in introducing a minimal set of primitive constructs (such as $\overline{\ }, \cap, ;, ^{\smile}, I$, for instance) and in defining the remaining ones in term of these (by putting, for example, $A \dagger B =_{\text{Def}} \overline{\overline{A}; \overline{B}}$).

Occasionally, we will enrich such a basic structure with a collection of relational constants (representing, as we will see, accessibility relations) and with a set of *non-standard* relational constructs not definable in terms of the basic ones. A distinction between basic and non-standard relational constructs can be established by classifying a construct as basic if its semantics can be expressed through a first-order sentence in three variables. Accordingly, we will call non-standard those constructs that are not expressible in three variables (cf. Section 3.1.1, and also [TAR 87]). We recall that establishing the 3-variable expressibility of a sentence is, in general, an undecidable problem [TAR 87, KWA 81]. Nevertheless, techniques can be designed to treat particular classes of formulae [CAN 03].

Let $\mathcal{L}$ be a non-classical logic and $Rel\mathcal{L}$ its relational rendering. In what follows we introduce the syntax and the semantics of $Rel\mathcal{L}$, whereas in Section 3 we develop the corresponding deductive apparatus.

2.1. *Syntax and semantics of* $Rel\mathcal{L}$

Let $\mathcal{V}$ be a set of individual variables (denoted by u, w, x, y, z, or occasionally by e, t, possibly subscribed), $\mathcal{R}$ a set of relational variables (P,Q,R,...), and $\mathcal{C}$ a set of relational constants. A *relational expression* is any term generated by the symbols in $\mathcal{R} \cup \mathcal{C} \cup \{U, Z, I, D\}$ and the relational constructs. Further (defined) basic constructs, can be introduced as usual, e.g.:

$$\begin{array}{rcl@{\qquad}rcl} P{-}Q & =_{\text{Def}} & P \cap \overline{Q}, & P{+}Q & =_{\text{Def}} & \overline{(Q \cup P){-}(Q \cap P)} \\ Q \backslash R & =_{\text{Def}} & \overline{\overline{R}; Q^{\smile}} & R/P & =_{\text{Def}} & \overline{P^{\smile}; \overline{R}} \end{array}$$

Given a binary relational construct $\circ$, its dual $\diamond$ is defined as $P \diamond Q =_{\text{Def}} \overline{\overline{P} \circ \overline{Q}}$ (and analogously for monadic constructs different from complementation).

The collection of all the relational expressions is denoted by $\mathcal{E}$.

A *relational equation* is a writing of the form $R{=}Q$, with $R, Q \in \mathcal{E}$. Shorthand notations for equalities of special kind are also possible, e.g.: $P \sqsubseteq Q \leftrightarrow_{\text{Def}} P{-}Q{=}Z$.

A *relational formula* is a writing of the form $x\varphi y$ where $\varphi \in \mathcal{E}$ is a relational expression and $x, y \in \mathcal{V}$ are individual variables. If $\varphi \in \mathcal{R} \cup \mathcal{C} \cup \{U, Z, I, D\}$, then $x\varphi y$ is said to be an *atomic* relational formula. Any atomic relational formula $x\varphi y$ and its complement $x\overline{\varphi}y$ are *literals*. A formula is *compound* if it is not a literal. The *leading* construct of a compound formula $x\varphi y$ is the dual of the construct $\circ$ if $\varphi = \overline{\psi \circ \phi}$ or $\varphi = \overline{\circ\psi}$. While, it is the construct $\circ$ if $\varphi = \psi \circ \phi$ or $\varphi = \circ\psi$.

Relational formulae are interpreted by means of semantic structures of the kind $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ where $\boldsymbol{\mathcal{D}}$ is a non-empty set and $\boldsymbol{\mathcal{I}}$ is a function assigning:

– A binary relation over $\boldsymbol{\mathcal{D}}$ to each relation symbol in $\mathcal{R} \cup \mathcal{C} \cup \{U, Z, I, D\}$. In particular, U is interpreted as the universal relation, I as the identity, Z as $\overline{U}$, and D as $\overline{I}$. Relational variables are interpreted as right-ideal relations. As usual a relation R is said to be right-ideal if $R; U = R$ holds (see [ORŁ 94] for details).

– The intended interpretation to the primitive and defined constructs. (Considering, as usual, $\cup$ and $\dagger$ to be the duals of $\cap$ and ;, respectively.)

We call such structures, *models* of $Rel\mathcal{L}$. An evaluation, or *assignment*, A in $\boldsymbol{\mathcal{M}}$ is a function $A : \mathcal{V} \to \boldsymbol{\mathcal{D}}$. A relational formula $x\varphi y$ of $Rel\mathcal{L}$ is *satisfied* by $\boldsymbol{\mathcal{M}}$ and A if $(x^A, y^A) \in \varphi^{\boldsymbol{\mathcal{I}}}$. In this case we write $\boldsymbol{\mathcal{M}}, A \models x\varphi y$. The formula $x\varphi y$ is satisfied (true) in $\boldsymbol{\mathcal{M}}$ if and only if for every evaluation A in $\boldsymbol{\mathcal{M}}$, it holds that $\boldsymbol{\mathcal{M}}, A \models x\varphi y$. A formula $x\varphi y$ of $Rel\mathcal{L}$ is *valid* if it is satisfied in every model of $Rel\mathcal{L}$.

2.2. *Relational formalizations of modal logics*

By following [ORŁ 94, FOR 06], in this section we describe the relational formalizations of a number of propositional non-classical logics. For each logic we present a syntax-directed translation. The target of such translation is always the algebra of relations and for each logic, a specific collection $\mathcal{C}$ of constant relations is chosen. Such constants are often subject to a set of axioms restraining their admitted interpretations. In this section φ, ψ, χ denote propositional sentences. As before, $\mathcal{R}$ is the collection of all relational variables.

Mono-modal logics

This is the basic translation of (propositional) modal formulae into relational terms originated in [ORŁ 91]. In this case $\mathcal{C} = \{\mathsf{r}\}$. The propositional connectives and the necessity operator are so translated:

$$\begin{array}{llll} t(\neg\,\psi) & =_{\text{Def}} & \overline{t(\psi)} & \qquad t(\psi\,\&\,\chi) =_{\text{Def}} t(\psi) \cap t(\chi) \\ t(\Diamond\,\psi) & =_{\text{Def}} & \mathsf{r}\,;\, t(\psi) & \qquad t(p_i) =_{\text{Def}} p_i' \end{array}$$

where p_i' is the relational variable in $\mathcal{R}$ that uniquely corresponds to the propositional variable p_i. Similar rules are introduced for the other customary propositional connectives.

Multi-modal logics

These logics correspond to multi-modal frames consisting of a relational system where $\mathcal{C}$ enjoys closure properties with respect to relational constructs. Modalities are then of the form $[R]$ and $\langle R\rangle$, where $R \in \mathcal{E}$ (cf. [ORŁ 94]). The translation of modal operators is the same as in the case of mono-modal logics. The differences between operators are articulated in terms of the properties of the corresponding accessibility relations.

Lattice-based modal logics

Lattice-based modal logics have the operations of disjunction and conjunction and, moreover, each of them includes a modal operator which can be either a possibility $\Diamond$, or necessity $\Box$, or sufficiency $\boxdot$, or dual sufficiency operator $\diamondsuit$ (see also [ORŁ 05, DÜN 04, JÄR 05]). Notice that, since negation is not available in these logics, both in the possibility–necessity and in the sufficiency–dual-sufficiency pair, neither operator is expressible in terms of the other. We can also consider mixed languages with any subset of these operators. For all of these logics we have $\{\leqslant_1, \leqslant_2\} \subseteq \mathcal{C}$. Moreover, $\leqslant_1$ and $\leqslant_2$ are assumed to be reflexive and transitive and such that $\leqslant_1 \cap \leqslant_2 = I$. The translation of disjunction and conjunction is as follows:

$$t(\psi \vee \chi) \quad =_{\text{Def}} \quad \overline{\leqslant_1; \overline{\leqslant_2; (t(\psi) \cup t(\chi))}} \qquad t(\psi \,\&\, \chi) \quad =_{\text{Def}} \quad t(\psi) \cap t(\chi).$$

As regards the alternative modalities, Table 1 summarizes their translations. Each modality is modeled in the relational framework by means of a pair of constant relations subject to suitable axioms (also displayed in Table 1).

Table 1. *Lattice-based modalities: translations and axioms for constant relations*

possibility $\Diamond$	$t(\Diamond\chi) =_{\text{Def}} \overline{\leqslant_1; \overline{S_\Diamond; \leqslant_2; t(\chi)}}$	$\leqslant_1^\smile; R_\Diamond; \leqslant_1^\smile \sqsubseteq R_\Diamond$ $\leqslant_2; S_\Diamond; \leqslant_2 \sqsubseteq S_\Diamond$	$R_\Diamond \sqsubseteq S_\Diamond; \leqslant_1^\smile$ $S_\Diamond \sqsubseteq \leqslant_2; R_\Diamond$
necessity $\Box$	$t(\Box\chi) =_{\text{Def}} \overline{R_\Box; \overline{t(\chi)}}$	$\leqslant_1; R_\Box; \leqslant_1 \sqsubseteq R_\Box$ $\leqslant_2^\smile; S_\Box; \leqslant_2^\smile \sqsubseteq S_\Box$	$R_\Box \sqsubseteq \leqslant_1; S_\Box$ $S_\Box \sqsubseteq R_\Box; \leqslant_2^\smile$
sufficiency $\boxdot$	$t(\boxdot\chi) =_{\text{Def}} \overline{R_\boxdot; \leqslant_2; t(\chi)}$	$\leqslant_1; R_\boxdot; \leqslant_2 \sqsubseteq R_\boxdot$ $\leqslant_2^\smile; S_\boxdot; \leqslant_1^\smile \sqsubseteq S_\boxdot$	$R_\boxdot \sqsubseteq \leqslant_1; S_\boxdot$ $S_\boxdot \sqsubseteq R_\boxdot; \leqslant_1^\smile$
dual sufficiency $\diamondsuit$	$t(\diamondsuit\chi) =_{\text{Def}} \overline{\leqslant_1; \overline{S_\diamondsuit; \overline{t(\chi)}}}$	$\leqslant_1^\smile; R_\diamondsuit; \leqslant_2^\smile \sqsubseteq R_\diamondsuit$ $\leqslant_2; S_\diamondsuit; \leqslant_1 \sqsubseteq S_\diamondsuit$	$R_\diamondsuit \sqsubseteq S_\diamondsuit; \leqslant_2^\smile$ $S_\diamondsuit \sqsubseteq \leqslant_2; R_\diamondsuit$

Logics of knowledge and information

We briefly recall three modal logics described in [DEM 02]. The first one is a logic with a knowledge operator $\mathbf{K}$. In this case $\mathcal{C} = \{\mathsf{r}\}$, and $\mathbf{K}$ is subject to the following translation rule:

$$t(\mathbf{K}\varphi) \quad =_{\text{Def}} \quad \overline{\mathsf{r}; \overline{t(\varphi)}} \cup \overline{\mathsf{r}; t(\varphi)}.$$

The second logic is the logic of non-deterministic information (NIL) [DEM 02, Section 7.2]. It is a multi-modal logic with three modalities, determined by the relations of informational inclusions ($\leqslant$ and $\geqslant$) and similarity (σ). Hence $\{\leqslant, \geqslant, \sigma\} \subseteq \mathcal{C}$ such that $\leqslant$ is reflexive and transitive and such that $\leqslant \;=\; \geqslant^\smile$. Moreover, σ is reflexive and symmetric, and it holds that $\geqslant; \sigma; \leqslant \sqsubseteq \sigma$.

The last logic of this group is the information logic (IL) [DEM 02, Section 7.3]. It is characterized by three modal operators corresponding to the relations of indiscernibility ($\equiv$), forward inclusion ($\leqslant$), and similarity (σ). Then $\{\equiv, \leqslant, \sigma\} \subseteq \mathcal{C}$ such that $\equiv$ is an equivalence relation, $\leqslant$ is reflexive and transitive, and σ is reflexive and symmetric. Moreover, both $\leqslant^{\smile}; \sigma \sqsubseteq \sigma$ and $\leqslant \cap \leqslant^{\smile} = \equiv$ hold.

Intuitionistic logic

The translation of propositional intuitionistic logic is based on the following rules, where $\leqslant \in \mathcal{C}$ denotes a reflexive and transitive constant relation.

$$\begin{array}{rcl@{\qquad\qquad}rcl} t(\psi \to \chi) & =_{\text{Def}} & \overline{\leqslant; (t(\psi) \cap \overline{t(\chi)})} & t(\psi \,\&\, \chi) & =_{\text{Def}} & t(\psi) \cap t(\chi) \\ t(\neg\, \psi) & =_{\text{Def}} & \overline{\leqslant; t(\psi)} & t(\psi \vee \chi) & =_{\text{Def}} & t(\psi) \cup t(\chi). \end{array}$$

Temporal logics

We consider here the relational formalization of temporal logics given in [ORŁ 95]. The modalities (referring to states in the future or in the past) are: $\mathtt{G}\phi$ (interpreted as "ϕ will be always true in the future"); $\mathtt{F}\phi$ ("ϕ will be true sometime in the future"); $\mathtt{H}\phi$ ("ϕ has been always true in the past"); $\mathtt{P}\phi$ ("ϕ was true sometime in the past"); $\phi\, \mathtt{U}\, \chi$ (Until: "there will be a moment in the future when χ is true and from now *until* then ϕ will be true"); $\phi\, \mathtt{S}\, \chi$ (Since: "there has been a moment in the past when χ was true and *since* then ϕ has been true"); $\mathtt{X}\phi$ ("ϕ will be true in the *next* moment in time"). Moreover, we can consider the two operators $\mathtt{R}$ (Release, dual of $\mathtt{U}$) and $\mathtt{T}$ (Trigger, dual of $\mathtt{S}$).

In this context, relational representations of temporal formulae are expressed by considering an accessibility relation r that (together with its converse $\mathsf{r}^{\smile}$) links time instants. The relational translations of the modalities are as follows:

$$\begin{array}{rcl@{\qquad\qquad}rcl} t(\mathtt{G}\phi) & =_{\text{Def}} & \overline{\mathsf{r}; \overline{t(\phi)}} & t(\mathtt{H}\phi) & =_{\text{Def}} & \overline{\mathsf{r}^{\smile}; \overline{t(\phi)}} \\ t(\mathtt{F}\phi) & =_{\text{Def}} & \mathsf{r}; t(\phi) & t(\mathtt{P}\phi) & =_{\text{Def}} & \mathsf{r}^{\smile}; t(\phi) \\ t(\phi\, \mathtt{U}\, \chi) & =_{\text{Def}} & t(\phi)\, \mathtt{U}\, t(\chi) & t(\phi\, \mathtt{S}\, \chi) & =_{\text{Def}} & t(\phi)\, \mathtt{S}\, t(\chi) \\ t(\mathtt{X}\phi) & =_{\text{Def}} & t((\phi \,\&\, \neg\, \phi)\, \mathtt{U}\, \phi) & & & \end{array}$$

Notice that in translating the modal operators $\mathtt{U}$ and $\mathtt{S}$ we introduced two new relational constructs (denoted, for simplicity, by the same symbols). Similar translations can be described for $\mathtt{R}$ and $\mathtt{T}$. Observe that these constructs are non-standard (in the sense of Section 2, page 369) since they cannot be defined in terms of the basic relational constructs. As an example we give the intended interpretation of $\mathtt{U}$ (see [ORŁ 95] for a more detailed treatment): $P\mathtt{U}Q$ designates the binary relation consisting of all pairs $\langle u, v\rangle$ such that there exists t such that $\langle u, t\rangle$ belongs to the accessibility relation $\mathsf{r}^{\mathfrak{S}}$, $\langle t, v\rangle$ belongs to $Q^{\mathfrak{S}}$, and for all w, if $\langle u, w\rangle \in \mathsf{r}^{\mathfrak{S}}$ and $\langle w, t\rangle \in \mathsf{r}^{\mathfrak{S}}$ then $\langle w, v\rangle \in P^{\mathfrak{S}}$. (The interpretation of $\mathtt{S}$ is analogous, with respect of $\mathsf{r}^{\smile}$.) We will discuss more on this aspect in Sections 3.1.1 and 4.3.

Other modal logics

Other modal logics for which it is possible to give a relational formalization are, among others: the logics with specification operators [JIF 86, ORŁ 88], the logics with Humberstone operators [HUM 83], the logics with sufficiency operators [GOR 90, DÜN 01]. In particular, following the semantics developed in [JIF 86], the operators of the weakest prespecification ($\backslash$) and the weakest postspecification ($/$) are modeled with residuals (cf. page 370). The Humberstone operators are the modal operators of possibility and necessity determined by the complement of an accessibility relation. It follows that their translation can easily be derived from the translation of the mono-modal operators. The sufficiency ($\boxdot$) and dual sufficiency ($\diamondsuit$) operators receive the following relational translation: $t(\boxdot\phi) =_{\text{Def}} \overline{\mathsf{r}; t(\phi)}$ and $t(\diamondsuit\phi) =_{\text{Def}} \mathsf{r}; \overline{t(\phi)}$, (with $\mathcal{C} = \{\mathsf{r}\}$). A treatment of these logics can be found also in [FOR 06].

2.3. *Classification of relational formulae*

In order to illustrate the deductive system and the related proof techniques and heuristics in a more concise and clear way, we introduce a classification of relational formulae with respect to their leading construct. A basic (non-standard) relational formula is a formula having a leading construct which is basic (non-standard). Basic relational formulae can be further classified by taking inspiration from Smullyan's uniform notation for first-order logic [SMU 95]. In fact, they can be grouped into five categories according to the first-order characterization of the relational constructs (cf. Table 2).

Table 2. *Classification of basic relational formulae*

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

If the leading construct is extensional (i.e., $\cup$, $\cap$) then, in analogy with Smullyan's notation, the relational formula can be classified as an α- or a β-formula. It is

an α-formula if its leading construct is the intersection, $\cap$ (or an analogous construct of "conjunctive nature", such as difference or complemented union). Conversely, it is classified as β-formula if its leading construct is the union, $\cup$ (or another one of "disjunctive nature"). Formulae of the form $x\overline{\overline{R}}y$ are classified as β-formulae.

Observe that, the first-order formulation of a formula with leading operator ; (such as $xR;Sy$) is of the form $(\exists z)(xRz \wedge zSy)$, where a conjunction occurs in the scope of an existential quantifier. Dually, formulae with † as leading operator (such as $xR\dagger Sy$) present first-order equipollents of the form $(\forall z)(xRz \vee zSy)$, where a disjunction is universally quantified. From such perspective, ; and † could be seen as compound operators where an existential quantifier is combined with a conjunction and a universal quantifier with a disjunction. Thus, in analogy with Smullyan's uniform notation, where existentially (universally) quantified formulae are classified as δ-formulae (γ-formulae), we classify relational formulae with leading operator ; (†) as δ^{α}-formulae (γ^{β}-formulae).[1] Complemented Peircean constructs are classified in an analogous way whereas formulae having the conversion, $^{\smile}$, as leading construct are denoted as κ-formulae.

Table 3. *Basic decomposition rules*

$$\frac{x\alpha y}{x\alpha_1 y | x\alpha_2 y} \qquad \frac{x\beta y}{\begin{array}{c} x\beta_1 y \\ [x\beta_2 y] \end{array}} \qquad \frac{x\gamma^{\beta} y}{\begin{array}{c} x\gamma_0^{\beta_1} w \\ w\gamma_0^{\beta_2} y \end{array}}$$

$$\frac{x\delta^{\alpha} y}{x\delta_0^{\alpha_1} z, x\delta^{\alpha} y | z\delta_0^{\alpha_2} y, x\delta^{\alpha} y} \qquad \frac{x\kappa y}{x\kappa_1 y}$$

3. The relational deductive apparatus

In this section we introduce a Rasiowa-Sikorski deductive system for the relational calculus. We describe its basic inference rules (decomposition and closure rules) as well as some efficiency issues mainly related to equality handling.

3.1. *A Rasiowa-Sikorski proof system for relational logics*

The development of a proof in usual Rasiowa-Sikorski systems proceeds by systematically decomposing the (disjunction of) formula(e) to be proved till a tautological

1. In what follows we feel free to use notations as δ-formula in place of δ^{α}-formula (and similarly for δ-expression, δ-rule, and so on), to denote formulae, expressions, etc., having ; as leading construct.

condition is detected through a closure rule (examples of such systems are described in [RAS 63, ORŁ 96]). Analogously to [ORŁ 96], the relational proof system we present here, relies upon a collection of decomposition rules for basic relational formulae and upon a closure rule which takes care of axiomatic sequences such as xRy, $x\overline{R}y$, and xUy (defined in Section 3.3). The basic decomposition rules have been defined according to the classification of formulae given in Section 2.3 and are illustrated in Table 3.

The α- and β-rules are used to decompose conjunctive and disjunctive formulae, respectively. The square brackets in the β-rule indicate that the second component, namely $x\beta_2 y$ may or may not be present. This to care of the situation where $\beta = \overline{\overline{\psi}}$, with ψ a relational expression in $\mathcal{E}$. The δ^α-rule decomposes δ^α-formulae, whereas γ^β-formulae are expanded by the γ^β-rule. The variable z introduced in the δ^α-rule is chosen among the individual variables already occurring in the proof. On the other hand, the individual variable w in the γ^β-rule is new to (the current branch of) the derivation. Notice that such decomposition rules reflect the duality of Rasiowa-Sikorski systems and tableaux systems [GOL 06].

The set of decomposition rules described above regards only basic relational formulae. However it can be enlarged with suitable specific rules to treat the decomposition of non-standard formulae and/or to deal with the properties of particular constants such as the equality relation or other relations, representing some accessibility relations.

Specific rules for non-standard formulae are treated in Sections 3.1.1 and 4.3, whereas the introduction of specific decomposition rules for relational constants such as, for instance, the equality, requires a bit of care and some discussion. In fact from the point of view of the efficiency, it is not always convenient to introduce a new decomposition rule.

The addition of new decomposition rules constitutes an effectively viable choice whenever such rules satisfy the following *sub-formula property*: "a relational specific-rule has the sub-formula property if and only if every formula in the consequences is a proper sub-formula of some formulae in the premises". A relational calculus has the *sub-formula property* if and only if each of its decomposition rules has the sub-formula property. A calculus with the sub-formula property can be easily converted into an efficient proof procedure. Moreover, this property guarantees the termination of the procedure [GOR 99][2].

A significant example of a relational proof system involving an interpreted constant $C \in \mathcal{C}$ and not satisfying the sub-formula property, is described in [DÜN 00]. Such a system has been designed to formalize the relational reasoning for contact algebras, where the relation C models the contact relationship in the context of qualitative

2. The basic decomposition rules in Table 3 have the sub-formula property with the exception of the δ^α-rule. The "problem" cannot be circumvented and is strictly related to the undecidability of the relational logic.

geometry. In the following sections we will describe some techniques to overcome difficulties related to rules of this kind (cf. Section 6).

It is also evident that the specific rules for equality, introduced for instance in [ORŁ 95], do not satisfy the sub-formula property. Nonetheless in place of these decomposition rules, one can add suitable axiomatic sequences to the closure rule to permit deductions in presence of equality and to recover the sub-formula property of the remaining part of the calculus.

In general, specific rules not fulfilling the sub-formula property can be avoided only if there exists an alternative way of expressing the underlying semantics or proper axioms (for instance, in presence of specific constant relations and of axioms restraining their interpretation).

In this frame of mind, our variant of relational Rasiowa-Sikorski system differs from the one described in [ORŁ 95] (among others). The main difference is in the treatment of equality and equality axioms. Indeed, instead of introducing a set of specific decomposition rules to fully handle the identity relation, we handle it by combining a refined closure phase together with a suitable simplification rewriting process (to be described in Section 6.2).

Notice that an alternative approach, not antithetic, would consists in adding the relational rendering of the proper axioms directly as part of the formula to be proved. We will consider a third viable approach in Section 6.2.

3.1.1. *Non-standard constructs*

The decomposition rules in Table 3 constitute a common core of any relational Rasiowa-Sikorski system. As mentioned, there are logics whose relational translation may involve intensional operators not expressible by means of basic relational constructs. For instance, in temporal logics we introduced new constructs, namely $\mathtt{U}$ and $\mathtt{S}$, as counterpart of the Until and the Since modal operators (page 373). In order to manipulate these new constructs some *ad hoc* decomposition rules are introduced. In particular, [ORŁ 95] proposes the following rules for $\mathtt{U}$ and $\mathtt{S}$:

$$\frac{xP\mathtt{U}Qy}{xrt, xP\mathtt{U}Qy \mid tPy, xP\mathtt{U}Qy \mid x\bar{r}u, u\bar{r}t, uPy, xP\mathtt{U}Qy}$$

$$\frac{xP\mathtt{S}Qy}{trx, xP\mathtt{S}Qy \mid tPy, xP\mathtt{S}Qy \mid u\bar{r}x, t\bar{r}u, uPy, xP\mathtt{S}Qy}$$

where t is chosen among the individual variables occurring in the proof and u is an individual variable new to the current branch of the derivation. (Notice that the above decomposition rules originate three branches. Nonetheless, it is easy to surrogate such triple-branching by a binary tree.)

The introduction of these rules can be justified by considering their frame-based semantics. For instance, in the case of the Until operator we have the following (binary) first-order formulation:

$$(\forall x\, y)(x\, P\mathtt{U}Q\, y) \equiv (\forall x\, y)(\exists t)(xrt \wedge tQy \wedge (\forall u)((xru \wedge urt) \supset uPy)).$$

This translates in the above decomposition rule. An analogous argument applies to the Since operator (as well as to Release and Trigger).

3.2. *The proof construction*

A Rasiowa-Sikorski *derivation tree* (or simply *derivation*) $\mathcal{D}$ for a disjunction of relational formulae S is represented as a binary ordered tree whose nodes are labeled by disjunctions of formulae.[3] We call *branch* of $\mathcal{D}$ any maximal path in $\mathcal{D}$. More formally:

DEFINITION 1. — *Let S be a disjunction of relational formulae of $Rel\mathcal{L}$. A Rasiowa-Sikorski derivation $\mathcal{D}$ for S is recursively defined as follows.*

The tree with only one node labeled with S, is a derivation for S. Let $\mathcal{D}$ be a derivation for S, θ a branch of $\mathcal{D}$, N the leaf-node of θ. Then, the tree obtained from $\mathcal{D}$ by applying a decomposition rule, as illustrated by items 1-6 below, is a derivation for S. Let φ be a formula in N.

1) If φ is a β-formula $x\beta y$, add $(N\setminus\{x\beta y\})\cup\{x\beta_1 y, x\beta_2 y\}$ as a successor of N;

2) If φ is a κ-formula, $x\kappa y$, add $(N \setminus \{x\kappa y\}) \cup \{y\kappa_1 x\}$ as successor of N;

3) If φ is an α-formula $x\alpha y$, add $(N\setminus\{x\alpha y\})\cup\{x\alpha_1 y\}$ and $(N\setminus\{x\alpha y\})\cup\{x\alpha_2 y\}$ as left and right successors of N, respectively;

4) If φ is a γ^β-formula $x\gamma^\beta y$, add $(N \setminus \{x\gamma^\beta y\}) \cup \{x\gamma_0^{\beta_1} w, w\gamma_0^{\beta_2} y\}$ as successor of N, where w is a variable new to θ;

5) If φ is a δ^α-formula $x\delta^\alpha y$, add $N \cup \{x\delta_0^{\alpha_1} z\}$ and $N \cup \{z\delta_0^{\alpha_2} y\}$ as left and right successors of N, where z is chosen among the variables occurring in $\mathcal{D}$;

6) If φ is a non-standard formula, extend θ according to the corresponding specific decomposition rule.

3.3. *Closure phase*

The closure of a branch is determined by applying a closure rule. For instance, we mentioned that a branch is declared closed whenever its leaf-node contains a pair of the form xRy and $x\overline{R}y$. The following definition summarizes, in more generality, such closure condition by handling the properties of equality.

DEFINITION 2. — *A node N is* closed *if one of the following conditions holds:*

1) N contains the formulae $x_1 D x_2, \ldots, x_{h-1} D x_h$, $y_1 D y_2, \ldots, y_{\ell-1} D y_\ell$, $x_1 \overline{R} z$, $x_h R y_\ell$, with $z = y_\ell$ (for some $h, \ell \geq 1$). In case R is a right-ideal atomic relation, then it is not required that $z = y_\ell$ holds.

3. By abuse of notation, we will often identify a node N with the set of the disjuncts constituting its label.

2) N contains the formulae $x_1 I x_h, x_1 D x_2, \ldots, x_{h-1} D x_h$ (for some $h \geq 1$).

Moreover, in order to take care of symmetry of equality, the distinction between $x_i D x_j$ and $x_j D x_i$ in these conditions is considered immaterial. A node is atomically closed *if R is an atomic relation.*

Notice that the basic closure conditions of usual relational Rasiowa-Sikorski systems, cf. [ORŁ 95], are instances of the above ones when putting $h = \ell = 1$.

A derivation $\mathcal{D}$ for a disjunction of formulae S of $Rel\mathcal{L}$ is *satisfied* by a model $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ of $Rel\mathcal{L}$ if, for every variable assignment A and branch θ of $\mathcal{D}$, $\boldsymbol{\mathcal{M}}$ and A satisfy θ (and we write $\boldsymbol{\mathcal{M}}, A \models \theta$). A model $\boldsymbol{\mathcal{M}}$ and an assignment A satisfy a branch θ if they satisfy each node of θ. A node N is satisfied by $\boldsymbol{\mathcal{M}}$ and A if at least one formula φ in N is satisfied by $\boldsymbol{\mathcal{M}}$ and A. A branch θ in a derivation is said to be (*atomically*) *closed* if its leaf-node is (atomically) closed. A derivation $\mathcal{D}$ is (atomically) closed if all its branches are (atomically) closed. A closed derivation for S is a *proof* of S.

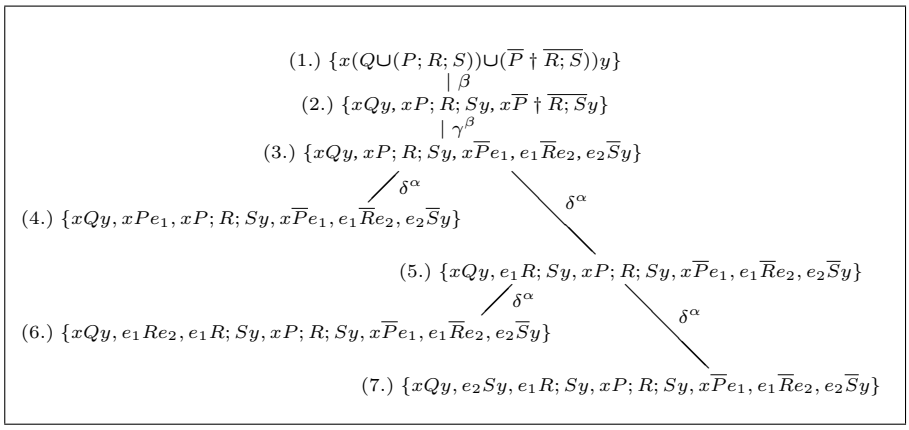

Figure 1. *Rasiowa-Sikorski proof for Example 3*

EXAMPLE 3. — Consider the valid formula

$$x(Q \cup (P; R; S)) \cup (\overline{P} \dagger \overline{R; S})y.$$

Figure 1 illustrates a Rasiowa-Sikorski proof for it. The labels of the edges indicate the decomposition rules applied in each derivation step. In particular, node (2.) has been obtained from node (1.) by applying the β-rule twice. Similarly, to obtain (3.), the γ^β-rule has been applied twice. Nodes (4.) and (6.) are closed because of the pairs of literals xPe_1 and $x\overline{P}e_1$, and e_1Re_2 and $e_1\overline{R}e_2$, respectively. Finally, the pair e_2Sy and $e_2\overline{S}y$ closes node (7.). □

4. An efficient representation of formulae and proofs

Trees are the most natural structure to represent derivations in analytic systems such as tableaux and Rasiowa-Sikorski (cf., Figure 1). Indeed, they reflect the idea of recursively decompose the formula to be proved till elementary contradictions (in case of tableaux systems) or axiomatic sequences (in case of Rasiowa-Sikorski systems) are found. Nevertheless the frequent presence of redundant parts in the trees, makes the proof-search procedure highly inefficient in practice [DAG 92, DAG 94]. Therefore, the application of more suitable data structures is required to construct efficient concrete provers.

In what follows we show how to represent relational formulae and their relative Rasiowa-Sikorski proofs by means of labeled acyclic graphs, and describe a series of refinements on such basic framework. Such a representation is akin to Binary Decision Diagrams and has several desirable properties when Boolean formulae are processed: it maximizes structure sharing since common sub-formulae are represented (and processed) once, it originates unique (up to the ordering imposed on labels of nodes) reduced canonical forms, and satisfiability and tautology checking are easily performed on reduced canonical forms. As we will see, in order to treat Peircean constructs, which involve implicit use of quantification, we need to circumvent the limited expressive power of basic BDD-style representations.

4.1. *From trees to graphs: structure sharing*

Let us consider a simplified scenario where only Boolean constructs (union of relations, intersection of relations, complement, etc) are involved. Thus, according to the calculus defined in Section 3, only α- and β-rules can be applied to construct a derivation for a given relational formula xRy. Consequently, proving that R=U amounts to fully decompose xRy by α- and β-steps to obtain a fully expanded derivation $\mathcal{D}$ (for xRy) and then to close the leaf N of every branch θ of $\mathcal{D}$ by applying the closure rule given in Section 3.3.

A similar situation arises when using BDDs to check the satisfiability of Boolean functions [BRY 86, BRY 92]. The use of graph-based representations, such as BDDs or Shannon graphs, have been proposed in the context of tableaux systems (see, among others, [POS 99, SCH 94, GOU 94a, GOU 94b]). We adopt a similar representation for relational expressions: a relational formula xRy is represented as a rooted directed acyclic graph. However our proposal differs from the previous ones since our aim is proving the validity of given relational expressions (i.e., that R=U) instead of checking the (un-)satisfiability of a propositional formula. Further, we have to develop a suitable method to deal with Peircean constructs.

In virtue of the strict analogies between propositional logic and the Boolean fragment of the relational calculus, we can simply formulate graph-based representations of α- and β-rules through the α- and β-decomposition schemas illustrated in Figure 2.

There, α_i and β_i, for $i \in \{1, 2\}$, stand for the components of α- and β-formulae, whereas **0** and **1** are new symbols labeling the only two leaf-nodes.

REMARK 4. — We recall that any assertion of the form $R{=}U$ has the sentence $\forall x \forall y(xRy)$ as first-order correspondent. Moreover, for β-decompositions we have $\forall x \forall y(x\beta y) \leftrightarrow \forall x \forall y(x\beta_1 y \vee x\beta_2 y)$. The same holds for α-decompositions. □

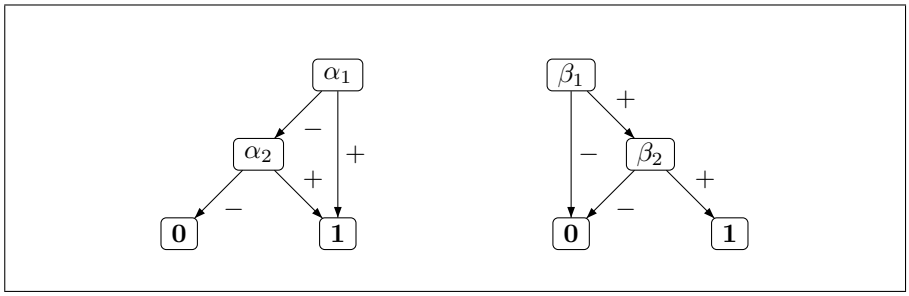

Figure 2. *α- and β-decomposition schemas*

Given a relational formula involving only Boolean constructs, by α- and β-decompositions we can obtain its representation as a (labeled) directed acyclic graph. Let us call such graph RDG (standing for relational decision graph). Each non-leaf node of an RDG has two outgoing edges labeled $-$ and $+$, respectively. Let us denote the corresponding sub-graphs by n^- and n^+, respectively. Moreover, let $r(n)$ denote the formula labeling the (non-leaf) node n. An example of RDG of an atomic formula (xSy, in this case) is depicted in Figure 3. In general, an RDG of a given relational formula is built up in syntax directed bottom-up process, from the RDGs of its subformulae, by "replacement of leaves". For instance, given the RDGs for xRy and xQy, an RDG for $x(Q \cap R)y$ can be obtained by replacing the **0**-leaf of the former with the root of the latter and merging the two **1**-leaves. This corresponds to apply α-decomposition. Similarly, an RDG for $x(Q \cap R) \cup Sy$ is obtained replacing the **1**-leaf of the RDG for $x(Q \cap R)y$, with the root node of the RDG for xSy (and merging the remaining identical leaves). This corresponds to apply β-decomposition.

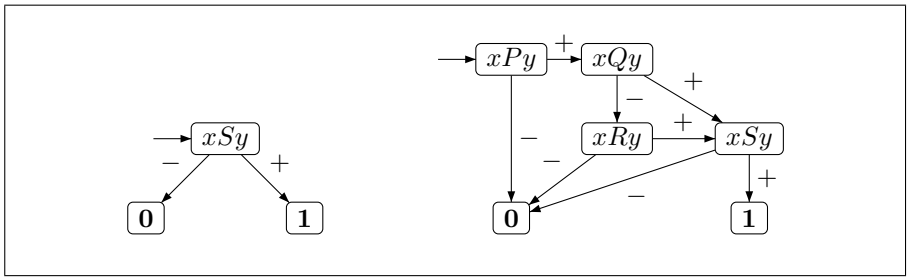

Figure 3. *The RDGs for xSy and $xP \cup ((Q \cap R) \cup S)y$*

The major advantage of adopting such a representation is the maximal structure sharing it intrinsically provides. Indeed, in building up the RDG of a formula, whenever a sub-formula occurs multiply, its sub-RDG is built only once.

An RDG constructed in this manner fulfills the following conditions: *a)* there are two leaves only (i.e., **0** and **1**); *b)* there are not two distinct non-leaf nodes n_1, n_2 such that the sub-graphs rooted at n_1 and n_2 are isomorphic. On the other hand, at this stage, we do not preclude nodes n such that $n^- = n^+$. (Hence, such RDG are not guaranteed to be *reduced* in the sense of [BRY 86].) Moreover, we do not impose any order on the (formulae labeling the) nodes of the RDG. We will deal with these aspects in the following sections, when we introduce a normalization procedure for RDGs.

We are left to deal with complementation of relations. This is easily done. For instance, if P is a relational letter, then an RDG for $x\overline{P}y$ is obtained from an RDG for P by exchanging its two leaves:

RDG for xPy RDG for $x\overline{P}y$

This simple transformation suffices to build RDGs for relational formulae in negative normal form. Such a normal form can be obtained by *pushing inside* the complement construct through applications of usual equivalences between dual operators (e.g.,$\overline{P \cup Q}$=($\overline{P} \cap \overline{Q}$)). More in general, given an RDG for xRy, an RDG for $x\overline{R}y$ can be obtained by exchanging the two leaf-nodes.

Given an RDG, a **1**-path is a path from the root node to the **1**-leaf. Any **1**-path p can be denoted as a sequence $n_1, s_1, n_2, s_2, \ldots, n_k, s_k, \mathbf{1}$, for $k \geq 0$, where each s_i is the label of the ith edge of p. A set of relational formulae $r(p)$ can be associated to any **1**-path p as follows. Let p be the **1**-path $n_1, s_1, n_2, s_2, \ldots, n_k, s_k, \mathbf{1}$, then $r(p)$ is defined as:

$$r(p) = \{r(n_i) : s_i = +\} \cup \{\overline{r(n_j)} : s_j = -\}.$$

A **1**-path p is closed if $r(p)$ is closed in the sense of Definition 2. In fact, analogously to the nodes of a derivation tree, $r(p)$ is a disjunction of formulae. Checking whether a relational formula xRy, involving only Boolean constructs, is valid (i.e., R=U), amounts to verify that every **1**-path p of an RDG for xRy is closed.

4.2. *Dealing with Peircean constructs*

The procedure described so far does not handle formulae involving Peircean constructs, i.e., relational product and sum. As mentioned, γ- and δ-decomposition rules

correspond to universal and existential quantifications. Hence dealing with them imposes going beyond the expressive power of basic BDD-like representations.

As regards first-order formulae, we can delineate a simple two-phase strategy to check satisfiability. Namely, an "inner" BDD-based procedure, performing propositional reasoning, guided by an "outer" procedure which (heuristically) generates amplifications of the given formula (see, [GOU 94a, GOU 94b], for instance). A similar approach is proposed in [GRO 03], where the amplification phase is rendered in terms of operations which directly manipulate BDDs. In [GOU 95] BDDs have been extended to treat (fragments of) quantified logic by using specific simplification rules to eliminate quantifiers. Alternatively, [BAD 02, BAD 04, BAD 05, POL 05] deal with the quantifier-free logic involving constants and function symbols as well as equality relation. In this case the authors, much in the spirit of [GRO 00, ZAN 01], exploit term rewriting techniques to handle function symbols and equality.

As we will see, we adopt a similar approach to deal with relational constants with fixed interpretation (cf. Section 6.2). On the other hand, to deal with δ-rules we operate a different choice, by refining and adapting to our context an interesting extension of un-ordered BDD proposed, for instance, in [POS 92, POS 95, POS 99, SCH 94], among others. In particular, in [POS 99] the authors propose a variant of a free-variable tableaux system where universal quantification is treated by means of a graph-nesting mechanism, and existential quantification is handled through a preliminary transformation into Skolemized negative normal form.

Since Rasiowa-Sikorski systems are dual of tableaux, such nesting mechanism will be used to process δ-formulae instead of γ-formulae. Notice also that we do not manipulate first-order formulae in their full generality, this because relational equalities characterize a proper sub-language of first-order logic obeying strict significant restrictions [TAR 87]. This simplifies the treatment and gives rise to a more profitable and efficient usage of the graph-nesting technique.

The basic idea consists in allowing an RDG to label a node in another RDG. Whenever a δ-decomposition has to be applied during the construction of an RDG, another RDG is constructed for the δ-formula at hand. Such an ancillary RDG will be "nested" as a single node (let us call δ-node a node of this kind) in the main RDG.

EXAMPLE 5. — Consider the valid formula of Example 3: $x(Q \cup (P; R; S) \cup (\overline{P} \dagger \overline{R; S}))y$. A (nested) RDG for such formula is depicted in Figure 4 (where for the sake of readability, we duplicated the **0**-leaf and the **1**-leaf. Thicker lines indicate δ-nodes.) Notice how the nesting mechanism is exploited to translate the sub-expression $P; R; S$. In particular, the formula $x(P; (R; S))y$ corresponds to RDG_1, while RDG_2 is the auxiliary graph for $p_0(R; S)y$. In the main graph, two γ-decompositions of the formula $x(\overline{P} \dagger \overline{R; S})y$ introduce the fresh individual variables e_3 and e_4. □

Notice that, by Definition 1, any application of a δ-rule must leave open the possibility of further δ-decompositions of the same δ-formula. At each decomposition, any of the individual variables occurring in the branch being extended may be employed

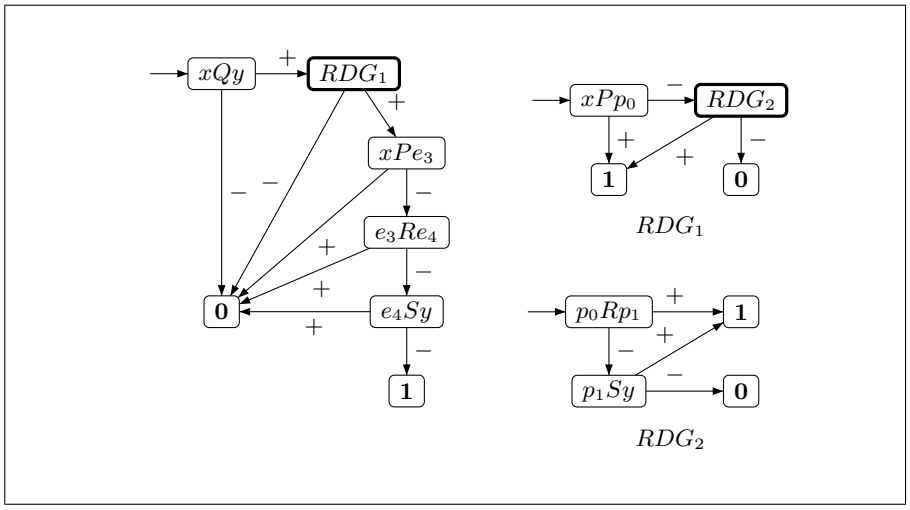

Figure 4. *RDGs for Example 5*

(z in Definition 1). As we will see (cf., Section 6.3), our graph-based proof procedure does not preliminarily perform any δ-decomposition. Indeed, δ-decomposition is executed only if the closure phase fails for the current branch: the δ-rule is applied only as the last chance. Applications of the δ-rule reflect the extension mechanism exploited in tree-based Rasiowa-Sikorski systems and in tableaux systems (for the γ-rule). Intuitively speaking, in trying to close a **1**-path p, we proceed by extending p by using an instantiation of one of its nested RDGs. Such an instantiation involves one of the individual variables occurring in the path. To prepare for this twofold step of instantiation and extension, whenever a δ-node is created, a template for the corresponding (nested) RDG is built and a "placeholder" is employed as parameter (cf., p_0 and p_1 in Example 5 and Figure 4). Such a placeholder will be replaced at each extension step (application of the δ-rule), with an individual variable selected from those already occurring in the path.

As regards γ-formulae, we proceed similarly to β-decomposition, with the only difference that a freshly generated individual variable is introduced (cf, Definition 1). This clearly corresponds to Skolemization in tableaux systems[4]. Plainly, if a γ-decomposition occurs within a nested RDG, i.e. within the scope of one or more δ-expressions, the Skolem term will be parametric in all the corresponding placeholders.

The remaining primitive Peircean construct, the conversion, is eliminated during the generation of the RDG: any formula of the form $xR^{\smile}y$ is simply rewritten as yRx.

4. Here, variables are treated as fixed, unknown individuals. In general different variables are intended to represent different elements of the domain and no unification step is allowed. For instance, in a derivation, $xRy \vee x\overline{R}y$ is an axiomatic sequence, whereas $xRy \vee w\overline{R}z$ is not.

4.3. *Non-standard relational constructs*

We have already considered in Section 3.1.1 the case of *non-classical* decomposition rules. In principle a relational rendering of any non-classical logic might involve constructs which are not expressible in terms of basic constructs.

In this section we describe a method to extend RDGs in order to treat any kind of construct, provided that its semantics can be given in terms of a formula of binary predicate logic. To this aim we generalize the use of nestings. More specifically, the formula characterizing the construct is converted into an RDG such that existentially and universally quantified sub-formulae are decomposed in such a way as to generalize the δ- and γ-decompositions described in Section 4.2. More specifically, the instantiation step of existential and universal quantifiers can be followed by any other kind of expansion step (for instance, α-, β-, κ-), and the strict coupling of the existential quantifier with the conjunction, and of the universal quantifier with the disjunction does not have to be observed.

As an example, let us consider the Until operator of temporal logic and the corresponding relational construct U. It can be characterized by the following first-order formula (where r denotes a constant relation modeling world accessibility):

$$x\, A\mathtt{U}B\, y \equiv \exists t(x\mathsf{r}t \wedge tBy \wedge (\forall u)(uAy \vee x\bar{\mathsf{r}}u \vee u\bar{\mathsf{r}}t)).$$

Notice that the structure of the formula suggests the structure of the RDG sought for. Operationally, we treat conjunctions (disjunctions) by applying α-decompositions (β-decompositions). Figure 5 shows the RDG induced by the above formula.

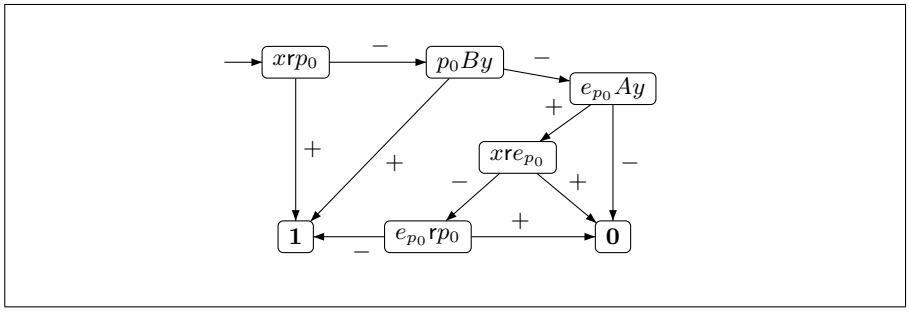

Figure 5. *RDG for the relational construct* U

Whenever the RDG for a given formula which contains the construct U has to be constructed, a δ-node with a nested RDG similar to the one in Figure 5 has to be created. Implementing such a mechanism corresponds to use a sort of "macro expansion" of non-standard constructs where two generalized δ- and γ-decompositions are applied. The former introducing a placeholder (namely, p_0 in the above figure, cf. Section 4.2) and the latter a new individual variable (i.e., denoted by the Skolem term e_{p_0}). Therefore the very same machinery developed for basic Peircean constructs can be reused to handle nested RDGs originated by non-standard constructs.

5. Soundness and completeness

In this section we prove soundness and completeness both for the relational Rasiowa-Sikorski system presented in Section 3 where derivations are represented as trees, and for the variant of the system introduced in Section 4 which adopts a graph-based representation of derivations (RDGs). In both cases proofs are carried out in a model-theoretic way focusing only on the core of the deductive system (namely, the set of decomposition rules in Table 3 and the closure rule in Definition 2).

As a result, when soundness and completeness of a logic with some specific rules have to be proved, we do not need to build the whole proofs from scratch. In fact, since soundness and completeness of the basic part of the deductive machinery have been established (once for all), the task simply consists in showing that the introduction of the specific rules in the system preserves its soundness and completeness.

Soundness and completeness of the system with graph-based proof representation could also be proved in a proof-theoretic way by building from any RDG, an equivalent derivation tree. In fact, it is possible to check that α-, β-, γ^{β}-, κ and δ^{α}-decomposition steps are equivalent to the corresponding decomposition rules of the relational Rasiowa-Sikorski system with lemmas (dual of the tableaux system with lemmas [DAG 99a]), and that every RDG can be transformed in a derivation tree with lemmas. It can also easily be shown that similar transformations hold between derivation trees with lemmas and without lemmas.

5.1. *Soundness of the Rasiowa-Sikorski system with derivation trees*

We have to show that applications of basic decomposition rules to a derivation tree preserve validity, and that the closure rule "closes" a leaf only if it is tautological.

LEMMA 6. — *Let S be a disjunction of relational formulae of $Rel\mathcal{L}$ and let $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ be a model of $Rel\mathcal{L}$ not satisfying S. Then $\boldsymbol{\mathcal{M}}$ does not satisfy any Rasiowa-Sikorski tree derivation for S constructed by the calculus in Section 3.1.*

PROOF. — A model $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ does not satisfy a derivation $\mathcal{D}$ if there exists a branch of $\mathcal{D}$, θ, and an assignment A over $\boldsymbol{\mathcal{M}}$ such that $\boldsymbol{\mathcal{M}}, A \not\models \varphi$ for each formula φ in every node of θ. This is true in particular for all the formulae in the leaf-node N of θ. By Definition 1, there exists an assignment A such that $\boldsymbol{\mathcal{M}}$ and A do not satisfy the initial derivation for S. Next, let $\mathcal{D}$ be a derivation for S and assume inductively that there is an assignment A such that $\boldsymbol{\mathcal{M}}$ and A do not satisfy it. Let $\mathcal{D}'$ be the derivation resulting from the application of one of the basic decomposition rules in Table 3 to a branch θ of $\mathcal{D}$. If $\boldsymbol{\mathcal{M}}, A \models \theta$ (before the expansion), then $\boldsymbol{\mathcal{M}}$ and A plainly do not satisfy $\mathcal{D}'$. Hence we can assume that $\boldsymbol{\mathcal{M}}$ and A do not satisfy θ and thus its leaf N, prior to the expansion. We show that there is an assignment B such that $\boldsymbol{\mathcal{M}}$ and B do not satisfy θ' (θ' is the branch θ after the expansion) and in particular its leaf N'. Therefore, even in this case, $\boldsymbol{\mathcal{M}}$ does not satisfy $\mathcal{D}'$.

Let us consider the following cases:

a) θ' is obtained by θ by decomposing a β-formula in N: the node $N' = (N \setminus \{x\beta y\}) \cup \{x\beta_1 y, x\beta_2 y\}$ is produced and attached to N. Since $\mathcal{M}, A \not\models x\beta y$, then $\mathcal{M}, A \not\models x\beta_1 y, x\beta_2 y$ and thus $\mathcal{M}, A \not\models N'$. Putting B equal to A the thesis holds;

b) If θ is expanded by an α-formula in N, the nodes $N_1 = (N \setminus \{x\alpha y\}) \cup \{x\alpha_1 y\}$ and $N_2 = (N \setminus \{x\alpha y\}) \cup \{x\alpha_2 y\}$ are produced and attached to N. Consequently, the two branches $\theta_1 = \theta, N_1$ and $\theta_2 = \theta, N_2$ are constructed. Since $\mathcal{M}, A \not\models x\alpha y$, one of the following two cases may be true: either $\mathcal{M}, A \not\models \alpha_1$, thus $\mathcal{M}, A \not\models N_1$ and putting $N' = N_1$, $\theta' = \theta_1$ and B equal to A the thesis follows, or $\mathcal{M}, A \not\models \alpha_2$, $\mathcal{M}, A \not\models N_2$ and setting $N' = N_2$, $\theta' = \theta_2$ and B equal to A we are through;

c) θ' is obtained by θ by decomposing a κ-formula $x\kappa y$ in N: the node $N' = (N \setminus \{x\kappa y\}) \cup \{y\kappa_1 x\}$ is produced and attached to N. Since $\mathcal{M}, A \not\models x\kappa y$, then $\mathcal{M}, A \not\models y\kappa_1 x$ and thus $\mathcal{M}, A \not\models N'$. Setting B equal to A the thesis follows;

d) θ' is obtained by θ by decomposing a γ^β-formula in N. The node $N' = (N \setminus \{x\gamma^\beta y\}) \cup \{x\gamma_0^{\beta_1} w, w\gamma_0^{\beta_2} y\}$, where w is a new variable to the branch, is produced and attached to N. $\mathcal{M}, A \not\models x\gamma^\beta y$, thus there exists an assignment B w-variant of A such that $\mathcal{M}, B \not\models x\gamma_0^{\beta_1} w, w\gamma_0^{\beta_2} y$. Since w is new to θ, $\mathcal{M}, B \not\models \theta$ and hence $\mathcal{M}, B \not\models \theta'$;

e) If θ is expanded by a δ^α-formula in N, the nodes $N_1 = N \cup \{x\delta_0^{\alpha_1} z\}$ and $N_2 = N \cup \{z\delta_0^{\alpha_2} y\}$, where z is chosen among the individual variables occurring in the derivation, are produced and attached to N originating the two branches $\theta_1 = \theta, N_1$ and $\theta_2 = \theta, N_2$. Since $\mathcal{M}, A \not\models x\delta^\alpha y$, one of the following two cases may be true: either $\mathcal{M}, A \not\models x\delta_0^{\alpha_1} z$, thus $\mathcal{M}, A \not\models N_1$, and putting $\theta' = \theta_1$, $N' = N_1$ and B equal to A the thesis follows, or $\mathcal{M}, A \not\models z\delta_0^{\alpha_2} y$, hence $\mathcal{M}, A \not\models N_2$, and identifying $\theta' = \theta_2$, $N' = N_2$ and setting B equal to A we are done. ■

LEMMA 7. — *If a leaf N of a Rasiowa-Sikorski tree derivation is closed by applying the closure rule from Definition 2 then it is tautological.*

PROOF. — We only need to show that if N satisfies either condition 1) or 2) in Definition 2, it is tautological:

1) If $h = l = 1$, then $x_1 = x_h$ and, since $z = y_l$, $x_1 R z \vee x_h \overline{R} y_l$ is a tautology contained in N. Now, let h, l be greater than 1. In every model of $Rel\mathcal{L}$, either x_1 is evaluated as x_h, y_1 as y_l, and hence the disjunction $x_1 R z \vee x_h \overline{R} y_l$ makes N true, or at least one of the formulae $x_1 D x_2, \ldots, x_{h-1} D x_h$, $y_1 D y_2, \ldots, y_{h-1} D y_h$ (possibly considering symmetry of D) makes N true. Thus N is a valid disjunction. A similar proof holds in case R is a right-ideal atomic relation.

2) If $h = 1$, then $x_1 I x_h$ is a valid formula of $Rel\mathcal{L}$ contained in N. Now suppose that $h > 1$. As above, either x_1 is evaluated as x_h, and thus $x_1 I x_h$ makes N true, or one of the formulae $x_1 D x_2, \ldots, x_{h-1} D x_h$ (or one of their symmetric counterparts) makes N true. N is therefore a valid disjunction. ■

THEOREM 8 (SOUNDNESS). — *If there is a closed Rasiowa-Sikorski tree derivation $\mathcal{D}$ (proof) for a disjunction of relational formulae S, then S is valid.*

PROOF. — Let us suppose that S is not valid. Then, there exists a model $\mathcal{M}$ not satisfying it. Thus, by Lemma 6, each derivation for S is not satisfied by $\mathcal{M}$. Then, $\mathcal{D}$ must have a branch θ, and in particular its leaf N, not satisfied by $\mathcal{M}$. But, by hypothesis, $\mathcal{D}$ is a closed derivation and hence, by Lemma 7, all its leaves are satisfied in every model. Absurd. Hence S is valid. ■

5.2. *Soundness of the Rasiowa-Sikorski system with graph-based representation*

In Section 4.1 a syntax directed bottom-up process is outlined to construct graph-based representations of Rasiowa-Sikorski derivations. This choice is basically due to efficiency reasons, since it allows to maximize structure sharing. Here we define a top-down recursive construction of the very same graph-based derivation. Such alternative definition is introduced uniquely for the purpose of simplifying soundness and completeness proofs.

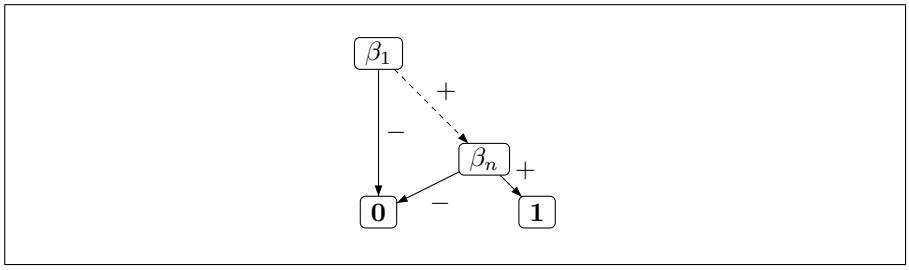

Figure 6. *Initial graph derivation for* $S = \beta_1 \cup \ldots \cup \beta_n$

DEFINITION 9. — *Let S be a disjunction of formulae of $Rel\mathcal{L}$. A relational Rasiowa Sikorski graph-based derivation $\mathcal{G}$ for S is recursively defined as follows:*

– the graph in Figure 6 (where $\beta_1, \ldots, \beta_n$ are the disjuncts in S), is a graph-based derivation for S (initial derivation);

– let $\mathcal{G}$ be a graph-based derivation for S, then the graph $\mathcal{G}'$, obtained by $\mathcal{G}$ by applying one of the decomposition steps described in Section 4.1 and 4.2, as shown by points $1-5$ below is a graph-based derivation for S:

a) if any node n in $\mathcal{G}$ is labeled with a β-formula, we replace n with its β-decomposition (see Figure 2);

b) if any node n in $\mathcal{G}$ is labeled with an α-formula, we replace n with its α-decomposition (see Figure 2);

c) if any node n in $\mathcal{G}$ is labeled with a κ-formula, we simply substitute in n the κ-formula with its component $y\kappa_1 x$;

d) if any node n in $\mathcal{G}$ is labeled with a γ^β-formula, we replace n with its γ^β-decomposition (see Figure 7), e_j is a variable new to every **1***-path containing n;*

e) if any node n in $\mathcal{G}$ is labeled with a δ^{α}-formula, we add as right successor of n its δ^{α}-decomposition (see Figure 7), where the placeholder p_i is substituted with a variable e_i, chosen among the variables already in $\mathcal{G}$.

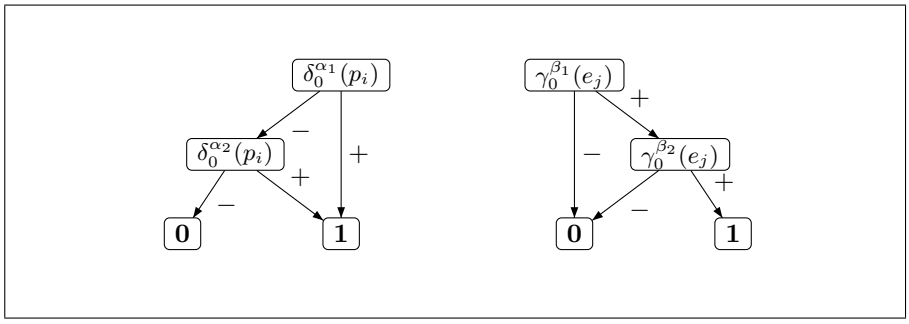

Figure 7. *δ^{α}- and γ^{β}-decompositions*

The closure of a graph-based derivation $\mathcal{G}$ is checked as outlined in Section 4.1.

We prove that our Rasiowa-Sikorski system with graph-based proofs is sound by showing that validity is preserved throughout the recursive graph construction, and that the closure rule "closes" only **1**-paths p with tautological $r(p)$.

We integrate the definitions given in Section 4.1 with some further notions. A graph-based derivation $\mathcal{G}$ for a disjunction of formulae S of $Rel\mathcal{L}$ is satisfied by a model $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ of $Rel\mathcal{L}$ if, for every assignment A, $\boldsymbol{\mathcal{M}}$ and A satisfy each **1**-path p of $\mathcal{G}$ (and we write $\boldsymbol{\mathcal{M}}, A \models p$). $\boldsymbol{\mathcal{M}}$ and A satisfy a **1**-path p of $\mathcal{G}$ if there exists at least one formula $\varphi \in r(p)$ such that $\boldsymbol{\mathcal{M}}, A \models \varphi$.

LEMMA 10. — *Let S be a disjunction of relational formulae of $Rel\mathcal{L}$ and let $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ be a model of $Rel\mathcal{L}$ not satisfying S. Then, $\boldsymbol{\mathcal{M}}$ does not satisfy any Rasiowa-Sikorski graph-based derivation constructed as shown in Definition 9.*

PROOF. — We follow the lines of the proof of Lemma 6 where a similar statement is proved for trees. A model $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ does not satisfy a graph $\mathcal{G}$ if we can find a **1**-path of $\mathcal{G}$, p, and an assignment A over $\boldsymbol{\mathcal{M}}$ such that $\boldsymbol{\mathcal{M}}, A \not\models \varphi$ for each formula $\varphi \in r(p)$. Trivially there exists an assignment A such that $\boldsymbol{\mathcal{M}}, A$ do not satisfy the initial derivation for S. Next, let $\mathcal{G}$ be a graph-based derivation for S and suppose inductively that there is an assignment A such that $\boldsymbol{\mathcal{M}}$ and A do not satisfy it. Let $\mathcal{G}'$ be the graph resulting from $\mathcal{G}$ by applying one of the decomposition steps as indicated in points $1-5$ of Definition 9, over a node n of $\mathcal{G}$. If $\boldsymbol{\mathcal{M}}, A \models p$ for every **1**-path p in $\mathcal{G}$ including n (before the decomposition step has been applied), then $\boldsymbol{\mathcal{M}}$ and A plainly do not satisfy $\mathcal{G}'$. Thus we can assume that, before the decomposition, $\boldsymbol{\mathcal{M}}$ and A do not satisfy a **1**-path p containing n. We show that there exists an assignment B such that $\boldsymbol{\mathcal{M}}$ and B do not satisfy one of the **1**-paths p' generated from p after the decomposition. Thus, even in this case, $\mathcal{G}'$ is not satisfied by $\boldsymbol{\mathcal{M}}$.

We distinguish the following cases:

a) If p is expanded by a β-decomposition, then the β-formula is substituted by its decomposition graph (see Figure 2). Since $\boldsymbol{\mathcal{M}}, A \not\models x\beta y$, then $\boldsymbol{\mathcal{M}}, A \not\models x\beta_1 y, x\beta_2 y$. Thus, setting B equal to A, the $\mathbf{1}$-path p' in $\mathcal{G}'$ originated from p by substituting $x\beta y, +$ with the sequence $x\beta_1 y, +, x\beta_2 y, +$ is not satisfied by $\boldsymbol{\mathcal{M}}$ and B;

b) If p is expanded by an α-decomposition, then the α-formula is substituted by its decomposition graph (described in Figure 2). Since $\boldsymbol{\mathcal{M}}, A \not\models x\alpha y$, then either $\boldsymbol{\mathcal{M}}, A \not\models x\alpha_1 y$ or $\boldsymbol{\mathcal{M}}, A \not\models x\alpha_2 y$. In the first case, p' is the $\mathbf{1}$-path coming out from p by replacing $x\alpha y, +$ with $x\alpha_1 y, +$. In the second, p' is obtained from p by substituting $x\alpha y, +$ with $x\alpha_1 y, -, x\alpha_2 y, +$. In any case, setting B equal to A, p' is not satisfied by $\boldsymbol{\mathcal{M}}$ and B, as wished;

c) Let p' be obtained by p by decomposing a κ-formula $x\kappa y$. Since it simply consists in replacing $x\kappa y$ with its component $y\kappa_1 x$ the thesis trivially follows;

d) If p is expanded by a γ^β-decomposition, the γ^β-formula is replaced by its decomposition graph (see Figure 7), where e_j is a variable new to every $\mathbf{1}$-path in $\mathcal{G}$ containing n. Since $\boldsymbol{\mathcal{M}}, A \not\models x\gamma^\beta y$, then there exists a B e_j-variant of A such that $\boldsymbol{\mathcal{M}}, B \not\models x\gamma_0^{\beta_1} e_j, e_j\gamma_0^{\beta_2} y$. Since e_j is new to p, $\boldsymbol{\mathcal{M}}, B \not\models p$. Thus, the $\mathbf{1}$-path p' in $\mathcal{G}'$ originated from p by superseding $x\gamma^\beta y, +$ with the sequence $x\gamma_0^{\beta_1} e_j, +, e_j\gamma_0^{\beta_2} y, +$ is not satisfied by $\boldsymbol{\mathcal{M}}$ and B as well;

e) If p is expanded by a δ^α-decomposition of a formula $x\delta^\alpha y$, we add as right successor of $x\delta^\alpha y$ an instance of the graph represented in Figure 7, by replacing the placeholder p_i with the variable e_i, chosen among the ones already in $\mathcal{G}$. Since $\boldsymbol{\mathcal{M}}, A \not\models x\delta^\alpha y$, then either $\boldsymbol{\mathcal{M}}, A \not\models x\delta_0^{\alpha_1} e_i$ or $\boldsymbol{\mathcal{M}}, A \not\models e_i\delta_0^{\alpha_2} y$. In the first case, p' is the $\mathbf{1}$-path generated by p by inserting after $x\delta^\alpha y, +$ the sequence $x\delta^{\alpha_1} e_i, +$. In the second, p' is obtained by p by inserting after $x\delta^\alpha y, +$ the sequence $x\delta^{\alpha_1} e_i, -, e_i\delta^{\alpha_2} y, +$. In any of the two cases, putting B equal to A, p' is not satisfied by $\boldsymbol{\mathcal{M}}$ and B; ■

LEMMA 11. — *Let p be a $\mathbf{1}$-path in a Rasiowa-Sikorski graph-based derivation $\mathcal{G}$. If p is closed, then $r(p)$ is tautological.*

PROOF. — Similarly to the leaves in Rasiowa-Sikorski tree derivations, $r(p)$ is a disjunction of relational formulae. Hence the proof is carried out as for Lemma 7. ■

THEOREM 12 (SOUNDNESS). — *If there is a closed graph for a disjunction of relational formulae S, then S is valid.*

PROOF. — Let us suppose that S is not valid. Then there exists a model $\boldsymbol{\mathcal{M}}$ not satisfying it. Thus, by Lemma 10 each graph for S is not satisfied by $\boldsymbol{\mathcal{M}}$. But, by Lemma 11 a closed graph (having only closed $\mathbf{1}$-paths) is satisfied in every model. Thus S has to be valid. ■

5.3. *Completeness of the Rasiowa-Sikorski system with tree-based representation*

The system is proved complete by endowing the calculus defined in Section 3 with a fair strategy to construct derivations, which transforms it in a fair deterministic

proof procedure (cf. Section 5.3.1 for details) able to determine, for any disjunction of formulae of the given language, a *saturated* Rasiowa-Sikorski tree derivation. A Rasiowa-Sikorski tree derivation $\mathcal{D}$ is saturated if it ensures a systematic and exhaustive exploration of the proof-search space, that is if every branch θ in $\mathcal{D}$ is *downward saturated*. A branch θ of a tree derivation is downward saturated if it fulfills these conditions:

1) if a formula $x\alpha y \in \theta$, then $x\alpha_1 y \in \theta$ or $x\alpha_2 y \in \theta$;

2) if a formula $x\beta y \in \theta$, then $x\beta_1 y, x\beta_2 y \in \theta$;

3) if a κ-formula $x\kappa y$ occurs in θ, then $y\kappa_1 x$ occurs in θ;

4) if a formula $x\gamma^{\beta} y \in \theta$, then $x\gamma_0^{\beta_1} w, w\gamma_0^{\beta_2} y \in \theta$ for some variables w in θ;

5) if a formula $x\delta^{\alpha} y \in \theta$, then $x\delta_0^{\alpha_1} z \in \theta$ or $z\delta_0^{\alpha_2} y \in \theta$ for every variable in θ;

Moreover, a branch θ of a tree derivation is *open* if it is not closed (or atomically closed) by the closure rule in Definition 2.

LEMMA 13. — *If θ is a downward saturated open branch, then there exists a model not satisfying it.*

PROOF. — We construct the model $\boldsymbol{\mathcal{M}} = \langle \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}} \rangle$ not satisfying θ. Let $\boldsymbol{\mathcal{D}}$ be the set of variables in θ and let A be an assignment for $\boldsymbol{\mathcal{M}}$ such that $x^A = x$ for every x in θ. Further, let the function $\boldsymbol{\mathcal{I}}$ be defined as follows. For every relational symbol R, let the atomic formula $[xRy]^{\boldsymbol{\mathcal{I}},A}$ be false if $xRy \in \theta$, true otherwise. Formulae that are not literals are interpreted in the conventional way, according to the semantics of the constructs. It is easy to check that $\boldsymbol{\mathcal{M}}$ is a well defined model, where the expected semantics is given to the equality relation. Hence, the quotient model $\boldsymbol{\mathcal{M}}_q$, whose domain has as elements the equivalence classes induced by the equality relation over the domain $\boldsymbol{\mathcal{D}}$, is defined as described in [GOL 06].

In order to prove that $\boldsymbol{\mathcal{M}}_q, A \not\models \theta$ we have to show that $\boldsymbol{\mathcal{M}}_q, A \not\models \psi$ for every $\psi \in \theta$. The proof easily follows by induction on the size of the formulae.

– All the literals in θ are not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A. Further, since θ is open, no axiomatic sequence is on θ (that can be satisfied by $\boldsymbol{\mathcal{M}}_q$);

– Let $x\alpha y$ be a formula in θ. Since θ is saturated, $x\alpha_1 y$ or $x\alpha_2 y$ is in θ. Let us suppose, without loss of generality, that $x\alpha_1 y$ is in θ. By inductive hypothesis $x\alpha_1 y$ is not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A, thus also $x\alpha y$ is not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A;

– Let $x\beta y$ be a formula in θ. Since θ is saturated, $x\beta_1 y, x\beta_2 y \in \theta$. By inductive hypothesis $x\beta_1 y$ and $x\beta_2 y$ are not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A, hence $x\beta y$ is not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A as well;

– Let $x\kappa y$ be a κ-formula in θ. Since θ is saturated, $y\kappa_1 x \in \theta$. By inductive hypothesis $y\kappa_1 x$ is not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A, it follows that $x\kappa y$ is not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A too;

– Let $x\gamma^{\beta} y$ be a formula in θ. Since θ is saturated, $x\gamma_0^{\beta_1} w, w\gamma_0^{\beta_2} y \in \theta$ for some variables w. By inductive hypothesis $x\gamma_0^{\beta_1} w$ and $w\gamma_0^{\beta_2} y$ are not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A and then, since $x\gamma^{\beta} y$ is a universal formula, it is not true in $\boldsymbol{\mathcal{M}}_q$.

– Let $x\delta^{\alpha}y$ be a formula in θ. Since θ is saturated, for every variable z in $\boldsymbol{\mathcal{D}}$, $x\delta_0^{\alpha_1}z$ or $z\delta_0^{\alpha_2}y$ is in θ. As a result, for every z in $\boldsymbol{\mathcal{D}}$ $x\delta_0^{\alpha_1}z \wedge z\delta_0^{\alpha_2}y$ is not satisfied by $\boldsymbol{\mathcal{M}}_q$ and A. Then $x\delta^{\alpha}y$ is not true in $\boldsymbol{\mathcal{M}}_q$. ■

5.3.1. *The fair deterministic proof procedure*

Completeness of the relational Rasiowa-Sikorski system can be established if we define a fair deterministic proof procedure allowing the construction of a saturated tree derivation for any disjunction of relational formulae S.

The features that have to be specified in order to eliminate the non-determinism are: frequency of the closure test, selection of the branch and of the formula to be expanded and selection of the variables to instantiate γ^{β}- and δ^{α}-formulae.

We assume that the closure test is performed before every expansion. Further, each leaf has the possibility of being expanded, but the ones without formulae containing δ^{α}-subformulae have priority. This choice is motivated by the fact that, due to the sub-formula property of the rules in the calculus, their decomposition process is finite. That is, we can check whether they are tautological or not.

Leaves proved tautological during a proof construction may be applied as lemmas (theorems deducible from the root disjunction S) in other parts of the proof, making faster the deduction process.

A similar strategy is chosen for the selection of the formula to be decomposed. Every compound formula in the proof deserves to be processed, but β-, α-, κ- and γ^{β}-formulae have a higher priority than δ^{α}-formulae. In fact, for occurrences of the former, one decomposition step sufficies to preserve the system completeness. The same does not hold for the latter that, in order to preserve completeness of the system, may need to be indefinitely decomposed, introducing a potentially infinite set of variables.

Non-determinism resulting from the choice of the variable to be used to instantiate δ^{α}- and γ^{β}-formulae is removed by establishing a suitable total ordering on the set of individual variables.

Now let S be a disjunction of formulae to be proved. We define the sequence of Rasiowa-Sikorski tree derivations for S, $\mathcal{D}_0, \mathcal{D}_1, \ldots$ as follows:

– $\mathcal{D}_0$ has a single node containing the formulae in S. Each of them is labeled with number 0;

– Now consider $\mathcal{D}_n$, $n \geq 0$. Let N be the chosen leaf in $\mathcal{D}_n$ and ψ the selected formula over N with number k:

1) If $\psi = x\beta y$, we can add as successor of N, $N' = (N \setminus \{x\beta y\}) \cup \{x\beta_i y\}$ (where with "," we indicate the disjunction) for every component $x\beta_i y$ of $x\beta y$ not contained in N;

2) If $\psi = x\kappa y$, we can add as successor of N, $N' = (N \setminus \{x\kappa y\}) \cup \{y\kappa_1 x\}$;

3) If $\psi = x\alpha y$, and none of the components of ψ is contained in N, then we can simultaneously add $N' = (N \setminus \{x\alpha y\}) \cup \{x\alpha_1 y\}$ as left successor and $N'' = (N \setminus \{x\alpha y\}) \cup \{x\alpha_2 y\}$ as right successor;

4) If $\psi = x\gamma^{\beta} y$, we can add as successor of N, $N' = (N \setminus \{x\gamma^{\beta} y\}) \cup \{x\gamma_0^{\beta_1} w, w\gamma_0^{\beta_2} y\}$, where w is the smallest variable in $\mathcal{D}$ (according to the chosen ordering) not already used on the current branch;

5) If $\psi = x\delta^{\alpha} y$ we can simultaneously add $N' = N \cup \{x\delta_0^{\alpha_1} z\}$ as left successor and $N'' = N \cup \{z\delta_0^{\alpha_2} y\}$ as right successor, where z is the kth variable in the chosen ordering on individual variables.

Formulae coming out from the decomposition of β-, α-, κ- and γ^{β}-formulae have number 0 while the decomposed formula (ψ) is deprived of its number k. δ^{α}-formulae are treated in a different way. In fact, after being decomposed, they are labeled with the number $k + 1$.

THEOREM 14 (COMPLETENESS). — *If S is a valid disjunction of relational formulae, then there exists a finite closed relational Rasiowa-Sikorski tree derivation for S.*

PROOF. — Let $\mathcal{D}$ be a saturated relational Rasiowa-Sikorski derivation for S. Let us suppose that $\mathcal{D}$ is not closed and it has a saturated open branch. By Lemma 13, S is not valid. Absurd. Finiteness of the tree comes from König's lemma [SMU 95]. ■

5.3.2. *Confluency*

Relational Rasiowa-Sikorski calculus is also *confluent*. Let us first introduce the concept of confluency for a calculus and then show why relational Rasiowa-Sikorski calculus is confluent.

DEFINITION 15. — *A calculus C is* proof confluent *or just* confluent *(for a class of formulae) if, for any valid formula φ (from the class), from any derivation $\mathcal{D}$ for φ constructed with the rules of C, a (closed) derivation for φ can be constructed with the rules of C.*

LEMMA 16. — *Relational Rasiowa-Sikorski calculus is* confluent *for the class of valid formulae.*

PROOF. — Let $\mathcal{D}$ be any partial relational Rasiowa-Sikorski derivation generated for a valid disjunction of relational formulae S by applying the classical rules from Table 3 or the specific non classical rules for the considered logic. By the completeness theorem, for any leaf N in $\mathcal{D}$, there exists a closed derivation $\mathcal{D}_N$ for the disjunction N, obtained by applying the deterministic proof procedure defined in Section 5.3.1. Construct such closed derivations for every leaf of $\mathcal{D}$ and append them to the corresponding leaves. ■

5.4. *Completeness of the Rasiowa-Sikorski system with graph-based representation*

The proof is along the lines of the one presented in Section 5.3. In particular, notions of saturated tree and branch are adapted to graphs and paths as described in the following.

A Rasiowa-Sikorski graph derivation for a disjunction of relational formulae S is *propositionally saturated* if it is constructed from the initial derivation for S by applying only decomposition steps $1-4$ from Definition 9, till all its nodes are labeled with atomic formulae or δ^α-formulae only. A δ^α-formula occurrence $x\delta^\alpha y$ in an RDG derivation $\mathcal{G}$ is *fully expanded* if, for every variable e_i in the $\mathbf{1}$-paths of $\mathcal{G}$ containing $x\delta^\alpha y$ with a positive sign, the corresponding δ^α-decomposition graph (see Figure 7) instantiated to e_i, is propositionally saturated and attached as a right successor to $x\delta^\alpha y$.

An RDG derivation $\mathcal{G}$ for a disjunction of relational formulae S is *saturated* if it is propositionally saturated and each δ^α-formula occurring in it is fully expanded. Every $\mathbf{1}$-path on a saturated RDG derivation is said saturated. Every sequence $r(p)$ of any saturated $\mathbf{1}$-path p of an RDG derivation $\mathcal{G}$ consists of literals together with δ^α-formulae. Let $r(p)-\Delta$ be the sequence $r(p)$ deprived of the δ^α-formulae, we call the *upward-saturated sequence* from $r(p)-\Delta$, the sequence $s(p)$ obtained as follows:

– every literal in $r(p)$ is in $s(p)$;
– if $x\beta_1 y$ and $x\beta_2 y$ are in $s(p)$ then $x\beta y$ is in $s(p)$;
– if $y\kappa_1 x$ is in $s(p)$ then $x\kappa y$ is in $s(p)$;
– if either $x\alpha_1 y$ is in $s(p)$ or $x\alpha_2 y$ is in $s(p)$ then $x\alpha y$ is in $s(p)$;
– if $x\gamma_0^{\beta_1} e_j$ and $e_j\gamma_0^{\beta_2} y$ are in $s(p)$ for some variable e_j, then $x\gamma^\beta y$ is in $s(p)$;
– if $x\delta_0^{\alpha_1} e_i$ or $e_i\delta_0^{\alpha_2} y$ are in $s(p)$ for every variable e_i in $\mathcal{G}$, then $x\delta^\alpha y$ is in $s(p)$.

LEMMA 17. — *If p is a saturated open $\mathbf{1}$-path, then there exists a model not satisfying $r(p)$.*

PROOF. — Let $s(p)$ be the upward saturated sequence from $r(p)-\Delta$. We can construct a model $\mathcal{M}$ not satisfying $s(p)$ analogously to what done in Lemma 13 for θ. $\mathcal{M}$ does not satisfy $r(p)$ as well, since it is a sub sequence of $s(p)$. ■

In a similar way to what shown for trees, we briefly sketch here a fair proof procedure that, in a deterministic way constructs a saturated RDG for a disjunction of relational formulae S (an effective and more refined procedure is discussed in Section 6.3).

Also in this case the closure test is executed before each decomposition step. The α-, β-, γ^β- and κ-decompositions have higher priority than the δ^α-. The δ^α-decomposition steps may be applied to the same δ^α-formula an unbounded number of times, potentially introducing an infinite number of distinct individual variables. Lemmas are handled for free since RDGs support maximal structure sharing.

THEOREM 18 (COMPLETENESS). — *If S is a valid disjunction of relational formulae, then there exists a finite closed RDG derivation for S.*

PROOF. — Let $\mathcal{D}$ be a saturated RDG derivation for S. Let us suppose that it is not closed, then it has a saturated open **1**-path p. Then, by Lemma 17 and by the definition of $s(p)$, S is not valid. Absurd. ■

6. Towards an efficient implementation

In this section we describe a number of techniques we adopted in implementing an automated Rasiowa-Sikorski deduction system. In particular, we refine the graph-based approach outlined in Section 4 in order to gain greater efficiency. The system we are going to delineate has been implemented in SICStus Prolog [WWWb].

6.1. *Ordering*

Given a relational formula, the procedure outlined in Section 4 produces an RDG without imposing any order on the (atomic formulae labeling the) nodes. Furthermore, it may be the case that the same formula labels two distinct nodes in a path. We introduce now a normalizing procedure that imposes a given order $\succ$ on the nodes of the RDG. Such a procedure is described as a term rewriting system (in the spirit of [GRO 00, ZAN 01]). The rewriting process continues until no further rewriting is applicable. An useful piece of notation: let the term $RDG(r(n), n^-, n^+)$ denote the RDG rooted in the node n. The rewriting system is constituted of the rules $1) - 5)$ in Table 4,[5] where $\succ$ is any total ordering relation on the collection of relational atomic formulae. Notice that all these rules preserve the closure property of the RDGs. Moreover, if all of the **1**-paths of the initial RDG can be closed because of pairs of complementary formulae, then the normalization process yields an RDG made of a single **0**-leaf. This immediately certifies the initial formula to be tautological. The above rewriting system can be enriched by considering additional rules (i.e. ,$6) - 9)$ in Table 4) to handle basic constants U, Z, I, D. In this manner, the rewriting process simplifies the RDG by removing those **1**-paths which are tautological because of atomic formulae of the forms xUy, xIx, etc. To impose node ordering and maximal structure-sharing (even between isomorphic sub-graphs of different nested graphs), the normalization process is recursively applied to each nested RDG.

As we will see, the adoption of a two-phases approach (i.e., a procedure to build-up the RDG coupled with a normalizing phase), instead of developing a procedure to obtain an ordered RDG directly from the formula, permits a simpler treatment of those relations subject to specific properties or axioms, such as reflexivity or symmetry.

5. We display the rewriting rules using the symbol $\rightsquigarrow$ to denote the rewriting relation, i.e. a rule of the form $t \rightsquigarrow r$, asserts that the term t is rewritten to r.

Table 4. *Rewriting system for the normalization process*

1)	$RDG(r(n), n^-, n^+) \rightsquigarrow n^-$	if $n^- = n^+$
2)	$RDG(r(n_1), RDG(r(n_2), n_2^-, n_2^+), n_1^+) \rightsquigarrow RDG(r(n_1), n_2^-, n_1^+)$	if $r(n_1) = r(n_2)$
3)	$RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+)) \rightsquigarrow RDG(r(n_1), n_1^-, n_2^+)$	if $r(n_1) = r(n_2)$
4)	$RDG(r(n_1), RDG(r(n_2), n_2^-, n_2^+), n_1^+) \rightsquigarrow$ $RDG(r(n_2), RDG(r(n_1), n_2^-, n_1^+), RDG(r(n_1), n_2^+, n_1^+))$	if $r(n_1) \succ r(n_2)$
5)	$RDG(r(n_1), n_1^-, RDG(r(n_2), n_2^-, n_2^+)) \rightsquigarrow$ $RDG(r(n_2), RDG(r(n_1), n_1^-, n_2^-), RDG(r(n_1), n_1^-, n_2^+))$	if $r(n_1) \succ r(n_2)$
6)	$RDG(r(n), n^-, n^+) \rightsquigarrow n^-$	if $r(n)$=xUy
7)	$RDG(r(n), n^-, n^+) \rightsquigarrow n^+$	if $r(n)$=xZy
8)	$RDG(r(n), n^-, n^+) \rightsquigarrow n^-$	if $r(n)$=xIx
9)	$RDG(r(n), n^-, n^+) \rightsquigarrow n^+$	if $r(n)$=xDx

REMARK 19. — Two refinements often adopted in tableaux systems are the use of lemmas and the imposition of some sort of regularity constraint on the application strategy for decomposition rules [LET 99]. These techniques have as counterpart, in our system, the adoption of a graph-based representation where ordering and maximal structure sharing are imposed. □

6.2. *Equality and interpreted relational constants*

In this section we describe how to refine the term rewriting system introduced in Section 6.1 in order to treat the identity relation as well as those constants that are interpreted as symmetric and/or reflexive relations.

Let us start by considering any relational expression R which is known to denote a symmetric binary relation, i.e., such that $\forall x \forall y(xRy \rightarrow yRx)$. In this case any label xRy in an RDG can be rewritten as yRx preserving the validity of the whole relational expression. This fact justifies the introduction of the following rewriting rule in the normalization procedure:

$$RDG(xRy, n^-, n^+) \rightsquigarrow RDG(yRx, n^-, n^+) \qquad \text{if } x \succ y$$

where we consider the total order $\succ$ as extended to the collection of all individual variables. On the other hand, whenever a relation R is such that $\forall x(xRx)$ holds, we can apply the following rewriting rule:

$$RDG(xRy, n^-, n^+) \rightsquigarrow n^- \qquad \text{if } x = y$$

Clearly, these rules apply in the case of the constants I and D, as well. Nevertheless, in the case of D a more fruitful rewriting rule can be introduced. Let us consider the following unsatisfiable sentence

$$\exists x_1 \cdots \exists x_h \exists y_1 \cdots \exists y_\ell \, (x_1 I x_2 \wedge \cdots \wedge x_{h-1} I x_h \wedge y_1 I y_2 \wedge \cdots \wedge y_{\ell-1} I y_\ell \wedge x_1 R y_1 \wedge x_h \overline{R} y_\ell). \tag{1}$$

where R is any relational expression. Since (1) is unsatisfiable, whenever all the formulae $x_1 D x_2, \ldots, x_{h-1} D x_h, y_1 D y_2, \ldots, y_{\ell-1} D y_\ell, x_1 \overline{R} y_1, x_h R y_\ell$ occur in a $\mathbf{1}$-path, such path can be closed (cf. the closure rule in Section 3.3). This observation justifies the following rewriting rules to be added to the normalizing procedure:

$$\begin{aligned} RDG(xDy, n^-, n^+) &\rightsquigarrow RDG(yDx, n^-, n^+) \qquad \text{if } x \succ y \\ RDG(xDy, n^-, n^+) &\rightsquigarrow RDG(xDy, n^-, n') \qquad \text{if } y \succ x \end{aligned}$$

where n' is obtained from n^+ by substituting each occurrence of the individual variable y with x. Consequently, further simplifications of the RDG become possible by means of application of other rewriting rules. Intuitively, assuming that $x_1 \succ \cdots \succ x_h \succ y_1 \succ \cdots \succ y_\ell$ holds, the effect of the these rules (in combination with the other ones) consists in rewriting the set of terms

$$\{x_1 D x_2, \ldots, x_{h-1} D x_h, y_1 D y_2, \ldots, y_{\ell-1} D y_\ell, x_1 \overline{R} y_1, x_h R y_\ell\}$$

into $\{x_h \overline{R} y_\ell, x_h R y_\ell\}$.

REMARK 20. — It is important to notice that, in virtue of this refinement of the rewriting system, we can sensibly simplify the closure rule. Actually, we can modify the closing conditions in Definition 2 by imposing $h = \ell = 1$. □

REMARK 21. — A treatment of symmetrical and reflexive expressions, analogous to the one outlined above, can be easily embedded into the procedure that constructs an RDG from a given formula. The same consideration can be done for the rules regarding the constants U, Z, I, D listed in Section 6.1. (Actually, both these options are actuated in the current implementation.) Much complex would be the treatment of the transitivity of equality during RDG generation. □

6.3. *Proof-search procedure*

Once an RDG is produced by the normalization phase described in the previous section, two situations may arise. If the RDG is made of a single node, then establishing validity of the initial formula xRy is immediate. Conversely, it might be the case that such RDG contains a number of non-trivial $\mathbf{1}$-paths. In order to declare xRy valid, each of them has to be closed. Checking all the $\mathbf{1}$-paths requires visiting the whole RDG. Since the graph is finite, each $\mathbf{1}$-path can be checked within a finite amount of time. For any $\mathbf{1}$-path $p: \; n_1, s_1, n_2, s_2, \ldots, n_k, s_k, \mathbf{1}$ let $l(p)$ be the set of formulae of $r(p)$ restricted to the nodes of p that are not δ-nodes. (Notice that, by the bottom-up

procedure used to obtain the RDGs, $l(p)$ is a set of literals.) Moreover, let $\delta(p)$ be the set of δ-nodes n_i, in p, such that $s_i = +$. For any **1**-path two situations may arise: *i)* $l(p)$ satisfies a closure condition. In this case p is declared closed. *ii)* $l(p)$ does not satisfy any closure condition. In this case closure may still be achieved in virtue of some extensions taking place by employing one or more nested RDG. Hence, the search procedure proceeds by selecting a δ-node m in $\delta(p)$ and an individual variable e among those occurring in the path. Let p_m be the placeholder of m. The **1**-path p is extended with a copy of the ancillary RDG corresponding to m, with e replaced for the placeholder p_m (this is done, as usual, by substituting the ending **1**-leaf of p with the root of the selected RDG). Such an extension step introduces in the initial RDG a number of new **1**-paths (all of them having p as prefix). At this point the process (recursively) proceeds trying to close these new **1**-paths. The process continues, possibly by applying further extension steps, until, either all **1**-paths are declared closed; or no more extensions are possible; or some termination conditions are verified.

Clearly, suitable strategies and heuristics should be exploited to guide the proof procedure. In particular, two questions must be answered each time an extension step has to be performed:

– which δ-node is selected? And

– which individual variable is selected?

Since more than one δ-node may be necessary in order to close a **1**-path, a *fair* selection rule must be adopted. The simplest choice consists in treating $\delta(p)$ as an ordered list. Consequently, each time a selection has to be done, the first element of $\delta(p)$ is selected and moved to the end of the list.

As regards the selection of individual variables, we rely once more on the order $\succ$ introduced in Section 6.1. In fact, for any fixed **1**-path p, the first time a specific δ-node d is selected, we consider the list V_d of variables in $l(p)$, ordered with respect to $\succ$. Each time an extension step employs d, the first unused variable, according to the ordering of V_d, is chosen.

As an (heuristic) optimization in the choice of the individual variable, one could consider the (atomic) formulae in $l(p)$ and the (atomic) formulae involving p_m in the selected RDG. Priority should be given to those variables that might introduce complementary pairs of (atomic) formulae in the extended **1**-paths.

It must be noted that Skolem terms may occur in nested RDGs. Such terms are parameterized by placeholders names. Hence each time an extension step is performed, new individual variables (may) be introduced in the (extended) **1**-path. These new variables have to be added to the set V_d, in order to be considered for future extension steps. Consequently, the search procedure is not guaranteed to terminate. This phenomenon cannot be avoided, since establishing the validity of a relational formula is, in general, undecidable. To avoid infinite computation we adopt a bounded depth-first iterative deepening [KOR 85]. At each iteration a bound is imposed on the number of extension steps for any single **1**-path. Different possibilities are explored through

backtracking. If the search does not terminate by declaring all **1**-paths closed, then the bound is increased and the procedure restarted.

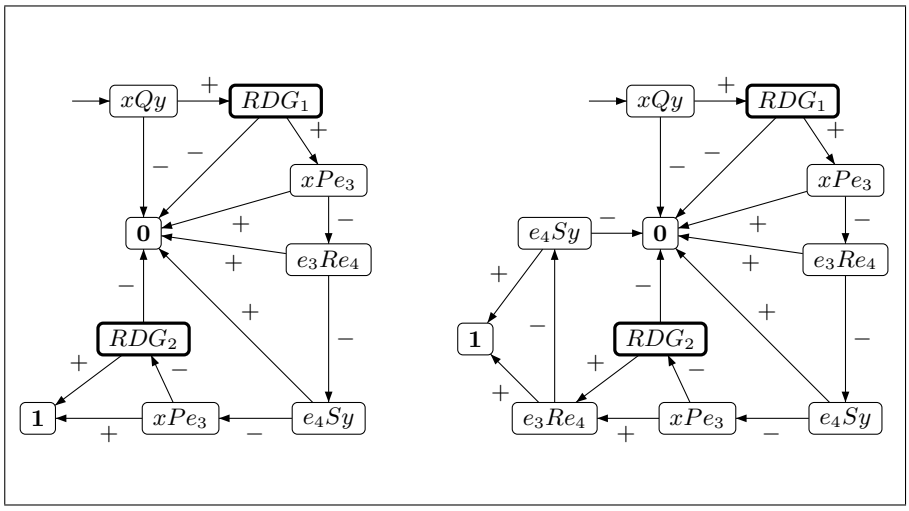

Figure 8. *The extension mechanism (see Example 22)*

EXAMPLE 22. — Let us consider the main RDG of Example 5. There is a single **1**-path, namely $p : xQy, +, xPe_3, -, e_3Re_4, -, e_4Sy, -, \mathbf{1}$. We have $l(p) = \{xQy, x\overline{P}e_3, e_3\overline{R}e_4, e_4\overline{S}y\}$, which cannot be closed. A first extension using RDG_1, and choosing e_3 as individual variable (in place of p_0), yields the situation depicted on the left-hand side of Figure 8. At this point the extension step originates two **1**-paths. One of them can be closed immediately since it contains the pair $x\overline{P}e_3, xPe_3$. To close the remaining **1**-path $p' : xQy, +, xPe_3, -, e_3Re_4, -, e_4Sy, -, xPe_3, -, \mathbf{1}$ another extension is needed. This time we extend using RDG_2 (which belongs to $\delta(p')$) and e_4 as individual variable. The resulting RDG is displayed on the right-hand side of Figure 8. Since such RDG is closed the proof terminates. □

Given a specific RDG, an effective approach to efficiently implement the proof-search procedure described so far consists in generating the Prolog code which encodes the proof-search itself. The program will then carry out the proof-search at run time. (Similar approaches have been successfully proposed in various contexts, see for instance [POS 99, SCH 00, WUN 96].) The basic technique consists in compiling the given RDG into a number of Prolog clauses and then executing the resulting program. In particular, each node n of the RDG originates a Prolog clause which, during execution, calls the clauses corresponding to n^+ and n^-. Clearly, the clause related to the **1**-leaf of the RDG also encodes the extension mechanism. Hence the proof of a modal theorem follows these phases: *a)* the given formula is translated into the relational calculus (Section 2.2); *b)* the obtained relational formula is processed to generate an RDG (Section 4); *c)* the RDG is normalized (Sections 6.1 and 6.2); *d)* the normalized RDG is compiled into a Prolog program; *e)* the execution of such Pro-

log program realizes a search for the proof (adopting a bounded depth-first iterative deepening strategy).

Some concluding remarks about the graph representation: we conclude this section by observing that our approach presents some differences if compared to the one in [POS 99] in at least two further aspects that have significant impact in the realization of the proof-search procedure. First, we do not make use of free-variables, even if, in principle, this is not precluded. Hence our system turns out to be ground. As a consequence, we do not need any unification procedure to implement closure rules. Second, we profitably impose ordering of nodes. This sensibly reduces the size of the generated RDGs to be processed by a subsequent proof-search procedure. It should be noted that the choice of the ordering can sensibly affect the size of the resulting RDG. The same phenomenon is observed in BDD-based representations [BRY 92] and it can be proved that the problem of choosing the best ordering is NP-complete [BOL 96, WEG 04]. Consequently, suitable heuristics should be used. One of the simplest heuristics that seems to perform well in the average case, consists in choosing the textual order of appearance of symbols in the given formula [GOU 96]. We adopted a similar choice. Namely, by taking advantage on the associativity and commutativity properties of relational constructs, the translator (from the modal logic into relational calculus) tries to re-order the relational (sub-)expressions so to reflect, as much as possible, the lexicografical ordering. Then, during the reasoning phase we can rely on the efficient handling lexicografical ordering of the Prolog engine.

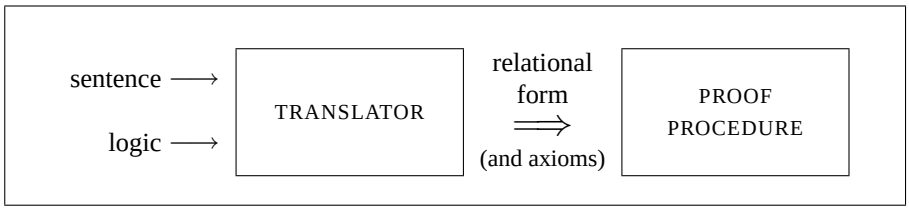

Figure 9. *Basic architecture of the inferential framework*

6.4. *The Prolog implementation*

As mentioned our Rasiowa-Sikorski proof system has been implemented in Prolog. Actually it is part of a prototypical tool supporting assisted as well as automated relational reasoning.

The architecture of the whole system (described in details in [FOR 05, FOR 06]) is depicted in Figure 9. In particular, it is composed of: an user-friendly mouse-oriented interface; a translator implementing all the translations described in Section 2.2, and producing an optimized relational rendering of the given formula; and the proof system described in this paper. At each stage of the development of a proof, the user can choose between exploiting the assistance of the tool and develop a proof by her/his

own, or leaving the system to proceed in an autonomous way. Figure 10 shows a simple example of development of a proof.

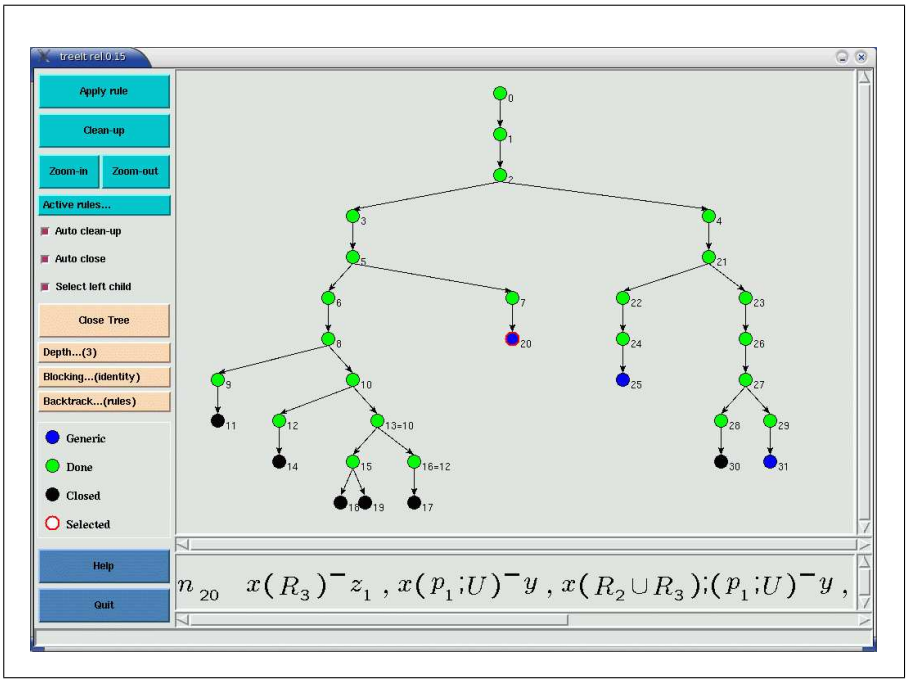

Figure 10. *Development of a proof*

The adoption of an approach based on declarative programming allows us to develop the system in an incremental way and ensures high modularity and extensibility of the application. Hence, the tool can be extended to encompass new translations for further non-classical logics. Each logic constitutes a "module" of the whole system. For any particular logic we can, in fact, identify a number of features, the most relevant being: *a)* the source language; *b)* the translation rules; *c)* the specific (interpreted) relational constants; *d)* a number of proper relational axioms characterizing such constants (usually, as accessibility relations satisfying specific properties); *e)* (possibly) a number of non-standard constructs together with the corresponding decomposition rules; *f)* (possibly) an enrichment of the closure rule (usually, to deal with the new specific constants and non-standard constructs). All these "ingredients" can be integrated in the system to make it able to prove theorems of the new logic.

7. Conclusions and future works

The research we reported upon in this paper shows that the duality results holding between (ground) tableaux systems and Rasiowa-Sikorski systems, naturally provide

strong support to cross-fertilization between the two streams of research. Actually, we explored just a small portion of the immense work and well established research done in the context of tableaux systems, considering a rather smooth application of such technology to the development of Rasiowa-Sikorski systems. Clearly, besides tableaux technology, techniques and heuristics can be borrowed from related fields such as automation of propositional logic, equational reasoning, term rewriting systems, semantic unification, theory reasoning, to mention some. In particular, in this initial work, we adopted a representation akin to Decision Diagrams, in origin designed to implement propositional logics. Moreover, we reported on the applicability of techniques developed for equality handling and based on term-rewriting rules.

Several steps in this direction of research have to be yet completed. For instance, the theoretical basis of our system-design are inspired to ground versions of tableaux, but an interesting theme for further study would consist in investigating the use of *free variables* and unification in relational Rasiowa-Sikorski systems. In general, free-variable tableaux systems can be implemented so to ensure greater efficiency, with respect to ground tableaux (see [GIE 02] for an efficient treatment of the unification procedure). It is reasonable that the use of free variables in Rasiowa-Sikorski systems could be advantageous as well.

Most of the modal logics of interest are decidable. Decidability results constitute a solid theoretical background for the development of attractive specific inference tools for such logics. A challenging theme of research consists in detecting under which conditions the decidability results for a non-classical logic can be reflected in some decidability property of its relational counterpart. A possibility could consist in identifying particular classes of deduction rules, that, together with a specific strategy in rule application, ensure decidability within the relational framework.

Finally, an extensive comparison, from the theoretical and experimental points of view, has to be completed with respect to alternative proposals, approaches, and tools present in the literature and designed to automate relational and modal inference. Among the several systems supporting various forms of relational manipulation and reasoning 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 formulae mainly by using substitution of equals for equals, weakening and strengthening (cf. [HAT 93]). The RELVIEW system (cf. [BEH 97]) offers a support for relational computation: assuming finiteness of domains and relations, it offers an explicit and extensional representation of concrete relations and provides an efficient implementation of the basic relational operations (actually, such an implementation relies on the use of BDDs [BER 02]). The Libra language (Lazy Interpreter of Binary Relational Algebra [DWY 95]) is a general-purpose programming language based on the algebra of binary relations that offers immediate support to program specification. Other systems, with different flavor but worth mentioning, are RALL [OHE 97] and δRA [DAW 98], based on HOL and Isabelle, and ARA [SIN 00], an Haskell-based theorem prover for relation algebras.

A few graphical approaches to relational calculus have been also proposed, some of them having an applicative character. For instance, relational methods are exploited in [BRO 94a, BRO 94b] to tame the problem of circuit design. This goal is achieved by developing a pictorial representation of relational terms and by providing high-level operations on pictures/circuits rendered by transformation rules. Other graphical calculi for representation of and visual reasoning on relational formulae are proposed in [CUR 96, FOR 01a]. The possibilities offered by the integration between these tools and our system should be profitably explored. Analogous considerations can be made with respect to the approach proposed in [FOR 01c] and the interaction/integration with relational reasoning tools ultimately based on first-order theorem provers [FOR 01b].

Among the tools that are expressly based on Rasiowa-Sikorski systems, we mention RelDT [MAC 03] (together with its ancestor ReVAT [LEE 02]). RelDT represents an approach to the automation of the basic rules of the relational calculus. (Some extensions, analogous to ours, are developed in order to deal with interpreted constant relations.) The implementation of RelDT is Prolog-based, but relies on a inference engine which uses *tabling*. The idea behind tabling (sometimes called memoization or lemmatization) can be intuitively summarized as follows: "never execute the same procedure-call twice". Then, the first time a call to a Prolog goal is made (in our context, an attempt to prove a lemma or to translate a sub-formula), the result is stored and if the same goal occurs in the future, previously computed answers are used. The use of tabling in RelDT can be seen as the counterpart of the structure sharing we exploited in representing formulae.

Also important should be considered the comparison with systems and tools expressly dedicated to modal reasoning. In many cases, such systems are specific to a restricted class of modal logics, especially when relying on decidability results. References for most of such systems can be found in [WWWa]. Here we mention a few of the implementations based on the tableaux technology, such as ModLeanTAP [BEC 97], Lotrec [CER 01], and cardTAP [GOR 98], closer in spirit to our system.

Acknowledgements

We thank Martin Giese, Joanna Golińska-Pilarek, Rajeev Goré, Ewa Orłowska, and Hui Wang for useful discussions on the topics of this paper.

This work is partially supported by INTAS Project — *Algebraic and deduction methods in non-classical logic and their applications to computer science* and by TARSKI Project — *Theory and Applications of Relational Structures as Knowledge Instruments COST Action 274*.

8. References

[BAD 02] BADBAN B., VAN DE POL J., "Two Solutions to Incorporate Zero, Successor and Equality in Binary Decision Diagrams", report , 2002, CWI Amsterdam , The Netherlands.

[BAD 04] BADBAN B., VAN DE POL J., "An Algorithm to Verify Formulas by Means of $(0, s, =)$-BDDs", *Proceedings of CSICC04*, 2004.

[BAD 05] BADBAN B., VAN DE POL J., "Zero, Successor and Equality in BDDs", *Annals of Pure and Applied Logic*, vol. 133, num. 1-3, 2005, p. 101–123.

[BEC 97] BECKERT B., GORÉ R., "Free Variable Tableaux for Propositional Modal Logics", *Proceedings, International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, vol. 1227 of *LNCS*, Springer, 1997, p. 91–106.

[BEH 97] BEHNKE R., BERGHAMMER R., SCHNEIDER P., "Machine Support of Relational Computations: The Kiel RELVIEW System", report num. Bericht Nr. 9711, 1997, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität, Kiel, Germany.

[BER 02] BERGHAMMER R., LEONIUK B., MILANESE U., "Implementation of Relational Algebra Using Binary Decision Diagrams", DE SWART H., Ed., *6$^{\text{th}}$ International Conference on Relational Methods in Computer Science, RelMiCS 2001*, vol. 2561 of *LNCS*, 2002, p. 241–257.

[BOL 96] BOLLIG B., WEGENER I., "Improving the Variable Ordering of OBDDs is NP-complete", *IEEE Transactions on Computers*, vol. 45, num. 9, 1996, p. 993–1002, IEEE Computer Society Press.

[BRI 97] BRINK C., KAHL W., SCHMIDT G., *Relational Methods in Computer Science*, Advances in Computing Science, Springer, 1997.

[BRO 94a] BROWN C., HUTTON G., "Categories, Allegories and Circuit Design", *Proceedings, 9$^{\text{th}}$ Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 1994, p. 372–381.

[BRO 94b] BROWN C., JEFFREY A., "Allegories of Circuits", *The 3$^{\text{rd}}$ International Symposium on Logical Foundations of Computer Science*, 1994, p. 56–68.

[BRY 86] BRYANT R. E., "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, vol. C-35, num. 8, 1986, p. 677–691.

[BRY 92] BRYANT R. E., "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams", *Computing Surveys*, vol. 24, num. 3, 1992, p. 293–318.

[CAN 03] CANTONE D., FORMISANO A., OMODEO E., ZARBA C., "Compiling Dyadic First-Order Specifications into Map Algebra", *Theoretical Computer Science*, vol. 293, num. 2, 2003, p. 447–475.

[CER 01] DEL CERRO L. F., FAUTHOUX D., GASQUET O., HERZIG A., LONGIN D., MASSACCI F., "Lotrec: the Generic Tableau Prover for Modal and Description Logics", Goré et al. [GOR 01], p. 453–458.

[CUR 96] CURTIS S., LOWE G., "Proofs with Graphs", *Science of Computer Programming*, vol. 26, num. 1-3, 1996, p. 197–216.

[DAG 92] D'AGOSTINO M., "Are Tableaux an Improvement on Truth Tables?", *Journal of Logic, Language and Information*, vol. 1, 1992, p. 235–252.

[DAG 94] D'AGOSTINO M., MONDADORI M., "The Taming of the Cut", *Journal of Logic and Computation*, vol. 4, 1994, p. 285–319.

[DAG 99a] D'AGOSTINO M., "Tableau Methods for Classical Propositional Logic", 1999, In [DAG 99b].

[DAG 99b] D'AGOSTINO M., GABBAY D. M., HÄHNLE R., POSEGGA J., Eds., *Handbook of Tableau Methods*, Kluwer Academic Publishers, Dordrecht, 1999.

[DAW 98] DAWSON J. E., GORÉ R., "A Mechanised Proof System for Relation Algebra using Display Logic", DIX J., DEL CERRO L. F., FURBACH U., Eds., *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA-98)*, vol. 1489 of *LNCS*, Springer, 1998, p. 264–278.

[DEM 02] DEMRI S., ORŁOWSKA E., *Incomplete Information: Structure, Inference, Complexity*, Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2002.

[DÜN 00] DÜNTSCH I., ORŁAWSKA E., "A Proof System for Contact Relation Algebras", *Journal of Philosophical Logic*, vol. 29, num. 3, 2000, p. 241–262.

[DÜN 01] DÜNTSCH I., ORŁOWSKA E., "Beyond Modalities: Sufficiency and Mixed Algebras", ORŁOWSKA E., SZALAS A., Eds., *Relational Methods for Computer Science Applications*, Heidelberg, 2001, Physica-Verlag, p. 277–299.

[DÜN 04] DÜNTSCH I., ORŁOWSKA E., RADZIKOWSKA A. M., VAKARELOV D., "Relational Representation Theorems for Some Lattice-Based Structures", *Journal on Relational Methods in Computer Science*, vol. 1, 2004, p. 132–160.

[DWY 95] DWYER B., "LIBRA: a Lazy Interpreter of Binary Relational Algebra.", report num. 95-10, 1995, Department of Computer Science University of Adelaide, Australia.

[FOR 01a] FORMISANO A., OMODEO E. G., SIMEONI M., "A Graphical Approach to Relational Reasoning", *Electronic Notes in Theoretical Computer Science*, vol. 44, num. 3, 2001, Elsevier Science.

[FOR 01b] FORMISANO A., OMODEO E. G., TEMPERINI M., "Instructing Equational Set-Reasoning with Otter", Goré et al. [GOR 01], p. 152–167.

[FOR 01c] FORMISANO A., OMODEO E. G., TEMPERINI. M., "Layered Map Reasoning: An Experimental Approach Put to Trial on Sets", DOVIER A., MEO M. C., OMICINI A., Eds., *Declarative Programming*, vol. 48, Elsevier Science, 2001.

[FOR 05] FORMISANO A., OMODEO E. G., ORŁOWSKA E., "A Prolog Tool for Relational Translation of Modal logics: A Front-end for Relational Proof Systems", BECKERT B., Ed., *TABLEAUX 2005 Position Papers and Tutorial Descriptions*, 2005, p. 1–10.

[FOR 06] FORMISANO A., OMODEO E. G., ORŁOWSKA E., "An Environment for Specifying Properties of Dyadic Relations and Reasoning about Them. II: Relational Presentation of Non-Classical Logics", DE SWART H., ORŁOWSKA E., SCHMIDT G., ROUBENS M., Eds., *Theory and Applications of Relational Structures as Knowledge Instruments II*, 2006, To appear.

[GIE 02] GIESE M., "Proof Search without Backtracking for Free Variable Tableaux", PhD thesis, Fakultät für Informatik, Universität Karlsruhe, Jul. 2002.

[GOL 06] GOLIŃSKA-PILAREK J., ORŁOWSKA E., "Tableaux and Dual Tableaux: Transformation of Proofs", *Studia Logica*, vol. To appear, 2006.

[GOR 90] GORANKO V., "Modal Definability in Enriched Languages", *Notre Dame Journal of Formal Logic*, vol. 31, 1990, p. 81–105.

[GOR 96] GORÉ R., "Cut-Free Display Calculi for Relation Algebras", *CSL*, vol. 1258 of *LNCS*, 1996, p. 198–210, Selected Papers of the Annual Conference of the Association for Computer Science Logic.

[GOR 98] GORÉ R., POSEGGA J., SLATER A., VOGT H., "System Description: cardTAP: The First Theorem Prover on a Smart Card", *Proceedings of the 15th International Conference on Automated Deduction*, vol. 1502 of *LNCS*, 1998, p. 239–248.

[GOR 99] GORÉ R., "Tableau Methods for Modal and Temporal Logics", 1999, In [DAG 99b].

[GOR 01] GORÉ R., LEITSCH A., NIPKOW T., Eds., *Automated Reasoning: First International Joint Conference, IJCAR 2001. Proceedings*, vol. 2083 of *LNCS*, Springer, 2001.

[GOU 94a] GOUBAULT J., "Proving with BDDs and Control of Information", BUNDY A., Ed., *Proceedings of the 12th International Conference on Automated Deduction*, vol. 814 of *LNCS*, Nancy, France, 1994, Springer, p. 499–513.

[GOU 94b] GOUBAULT J., POSEGGA J., "BDDs and Automated Deduction", RAŚ Z. W., ZEMANKOVA M., Eds., *Proceedings 8th International Symposium on Methodologies for Intelligent Systems ISMIS*, vol. 869 of *LNCS*, Springer, 1994, p. 541–550.

[GOU 95] GOUBAULT J., "A BDD-Based Simplification and Skolemization Procedure", *Journal of the IGPL*, vol. 3, num. 6, 1995, p. 827–855.

[GOU 96] GOUBAULT-LARRECQ J., "Implementing Tableaux by Decision Diagrams", report, 1996, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme.

[GRO 00] GROOTE J. F., VAN DE POL J., "Equational Binary Decision Diagrams", PARIGOT M., VORONKOV A., Eds., *Logic for Programming and Automated Reasoning: 7th International Conference, LPAR 2000*, vol. 1955 of *LNCS*, Springer, 2000, p. 161–178.

[GRO 03] GROOTE J. F., TVERETINA O., "Binary Decision Diagrams for First-Order Predicate Logic", *Journal of Logic and Algebraic Programing*, vol. 57, num. 1-2, 2003, p. 1–22.

[HAT 93] HATTENSPERGER C., BERGHAMMER R., SCHMIDT G., "RALF - A Relation-Algebraic Formula Manipulation System and Proof", NIVAT M., RATTRAY C., RUS T., SCOLLO G., Eds., *Proc. of AMAST93*, Springer-Verlag, 1993.

[HEN 80] HENNESSY M. C. B., "A Proof System for the First-Order Relational Calculus", *Journal of Computer and System Sciences*, vol. 20, num. 1, 1980, p. 96–110.

[HUM 83] HUMBERSTONE L., "Inaccessible Worlds", *Notre Dame Journal of Formal Logic*, vol. 24, 1983, p. 346–352.

[JÄR 05] JÄRVINEN J., ORŁOWSKA E., "Relational Correspondences for Lattices with Operators", DÜNTSCH I., WINTER M., Eds., *Proceedings of the 8th International Conference on Relational Methods in Computer Science (RelMiCS 8)*, 2005.

[JIF 86] JIFENG H., HOARE C. A. R., "Weakest Prespecifications, part 1", *Fundamenta Informaticae*, vol. IX, 1986, p. 51–84, Part II ibidem IX:217–252.

[KOR 85] KORF R. E., "Depth-First Iterative Deepening: an Optimal Admissible Tree Search", *Artificial Intelligence*, vol. 27, num. 1, 1985, p. 97–109.

[KWA 81] KWATINETZ M., "Problems of Expressibility in Finite Languages", Doctoral Diss., Univ. of California, Berkeley, 1981.

[LEE 02] LEE G., LITTLE R., MACCAULL W., SPENCER B., "ReVAT - Relational Validator by Analytic Tableaux", report , 2002, Department of Mathematics, Statistics and Computer Science, St. Francis Xavier University, Antigonish, Canada.

[LET 99] LETZ R., "First-Order Tableau Methods", 1999, In [DAG 99b].

[MAC 03] MACCAULL W., ORLOWSKA E., "A Logic of Typed Relations and its Applications to Relational Databases", report , 2003, Department of Mathematics, Statistics and Computer Science, St. Francis Xavier University, Antigonish, Canada.

[MAD 83] MADDUX R. D., "A Sequent Calculus for Relation Algebras", *Annals of Pure and Applied Logic*, vol. 25, 1983, p. 73–101.

[MAD 91] MADDUX R. D., "The Origin of Relation Algebras in the Development and Axiomatization of the Calculus of Relations", *Studia Logica*, vol. 50, num. 3/4, 1991, p. 421–455.

[OHE 97] VON OHEIMB D., GRITZNER T. F., "RALL: Machine-Supported Proofs for Relational Algebra", MCCUNE W., Ed., *Proceedings of the 14th International Conference on Automated Deduction*, vol. 1249 of *LNCS*, Springer, 1997, p. 380–394.

[OHL 01] OHLBACH H. J., NONNENGART A., DE RIJKE M., GABBAY D. M., "Encoding Two-Valued Non-Classical Logics in Classical Logic", ROBINSON A., VORONKOV A., Eds., *Handbook of Automated Reasoning*, vol. II, chapter 21, p. 1403–1486, Elsevier Science B.V., 2001.

[ORŁ 88] ORŁOWSKA E., "Proof System for Weakest Prespecification", *Information Processing Letters*, vol. 27, num. 6, 1988, p. 309–313.

[ORŁ 91] ORŁOWSKA E., "Relational Interpretation of Modal Logics", ANDRÉKA H., NEMETI I., MONK D., Eds., *Algebraic Logic*, vol. 54 of *Colloquia mathematica Societatis Janos Bolyai*, p. 443–471, North-Holland, Amsterdam, 1991.

[ORŁ 92] ORŁOWSKA E., "Relational Proof System for Relevant Logics", *Journal of Symbolic Logic*, vol. 57, num. 4, 1992, p. 1425–1440.

[ORŁ 94] ORŁOWSKA E., "Relational Semantics for Non-Classical Logics: Formulas are Relations.", WOLENSKI J., Ed., *Philosophical Logic in Poland.*, p. 167–186, Kluwer, 1994.

[ORŁ 95] ORŁOWSKA E., "Temporal Logics — in a Relational Framework", BOLC L., SZALAS A., Eds., *Time and Logic — A Computational Approach.*, p. 249–277, Univ. College London Press, 1995.

[ORŁ 96] ORŁOWSKA E., "Relational Proof Systems for Modal Logics", WANSING H., Ed., *Proof Theory of Modal Logic*, vol. 2, Kluwer, 1996, p. 55–77.

[ORŁ 97] ORŁOWSKA E., "Relational Formalisation of Nonclassical Logics", BRINK C., KAHL W., SCHMIDT G., Eds., *Relational Methods in Computer Science*, Advances in Computing Science, chapter 6, p. 90–105, Springer, Wien, New York, 1997.

[ORŁ 05] ORŁOWSKA E., VAKARELOV D., "Lattice-Based Modal Algebras and Modal Logics", VALDES-VILLANUEVA P. H. L., WESTERSTAHL D., Eds., *Logic, Methodology and Philosophy of Science. Proceedings of the 12th International Congress*, KCL Puplications, 2005, p. 147–170.

[POL 05] VAN DE POL J., TVERETINA O., "A BDD-Representation for the Logic of Equality and Uninterpreted Functions", *Proceedings of MFCS*, vol. 3618 of *LNCS*, 2005, p. 769–780.

[POS 92] POSEGGA J., LUDÄSCHER B., "Towards First-Order Deduction Based on Shannon Graphs", OHLBACH H. J., Ed., *GWAI*, vol. 671 of *LNCS*, Springer, 1992, p. 67–75.

[POS 95] POSEGGA J., SCHMITT P. H., "Automated Deduction with Shannon Graphs", *Journal of Logic and Computation*, vol. 5, num. 6, 1995, p. 697–729.

[POS 99] POSEGGA J., SCHMITT P. H., "Implementing Semantic Tableaux", 1999, In [DAG 99b].

[RAS 63] RASIOWA H., SIKORSKI R., *The Mathematics of Metamathematics*, Polish Scientific Publishers, 1963.

[SCH 82] SCHÖNFELD W., "Upper Bounds for a Proof-Search in a Sequent Calculus for Relational Equations", *Zeitschrift fuer Mathematische Logic und Grundlagen der Mathematik*, vol. 28, 1982, p. 239–246.

[SCH 94] SCHNEIDER K., KUMAR R., KROPF T., "Accelerating Tableaux Proofs Using Compact Representations", *Formal Methods in System Design*, vol. 5, num. 1-2, 1994, p. 145–176.

[SCH 00] SCHÜTZ H., GEISLER T., "Efficient Model Generation through Compilation", *Information and Computation*, vol. 162, num. 1-2, 2000, p. 138–157, Academic Press.

[SIN 00] SINZ C., "System Description: ARA - An Automatic Theorem Prover for Relation Algebras", MCALLESTER D. A., Ed., *Proceedings of the 17$^{\text{th}}$ International Conference on Automated Deduction*, vol. 1831 of *LNCS*, Springer, 2000, p. 177–182.

[SMU 95] SMULLYAN R. M., *First-Order Logic*, Dover Publications, New York, second corrected edition, 1995, First published 1968 by Springer.

[TAR 87] TARSKI A., GIVANT S., *A Formalization of Set Theory without Variables*, vol. 41 of *Colloquium Publications*, American Mathematical Society, 1987.

[WEG 04] WEGENER I., "BDDs-Design, Analysis, Complexity, and Applications", *Discrete Applied Mathematics*, vol. 138, num. 1-2, 2004, p. 229–251.

[WUN 96] WUNDERWALD J. E., "Memoing Evaluation by Source-to-Source Transformation", *Proceedings of 5$^{\text{th}}$ LOPSTR*, vol. 1048 of *LNCS*, 1996, p. 17–32.

[WWWa] "Web repository of computational tools for non-Classical logics, AiML", `http://www.cs.man.ac.uk/~schmidt/tools`.

[WWWb] "Web reference for SICStus Prolog", `http://www.sics.se/sicstus`.

[ZAN 01] ZANTEMA H., VAN DE POL J., "A Rewriting Approach to Binary Decision Diagrams", *Journal of Logic and Algebraic Programing*, vol. 49, num. 1-2, 2001, p. 61–86.