An Environment for Specifying Properties of Dyadic Relations and Reasoning About Them II: Relational Presentation of Non-classical Logics*

Andrea Formisano[1], Eugenio G. Omodeo[2], and Ewa Orłowska[3]

[1] Dipartimento di Informatica, Università di L'Aquila, Italy
formisano@di.univaq.it
[2] Dipartimento di Matematica e Informatica, Università di Trieste, Italy
eomodeo@units.it
[3] National Institute of Telecommunications, Warsaw, Poland
orlowska@itl.waw.pl

Abstract. This paper contributes to the vast literature on relational renderings of non-classical logics providing a general schema for automatic translation. The translation process is supported by a flexible Prolog tool. Many specific translations are already implemented, typically leading from an unquantified logic into the calculus of binary relations. Thanks to the uniformity of the translation pattern, additional source languages (and, though less commonly, new target languages) can be installed very easily into this Prolog-based translator. The system also integrates an elementary graphical proof assistant based on Rasiowa-Sikorski dual-tableau rules.

Keywords: Relational systems, translation methods, modal logic.

Introduction

Common approaches to the automation of modal inferences often exploit *ad hoc*, direct inference methods (cf., e.g., [23, 33]). An alternative approach, discussed in the ongoing and aimed at developing a uniform relational platform for modal reasoning, is intended to benefit from relational renderings of non-classical logics (cf. [27] among others).

The envisaged framework covers a full-fledged inferential apparatus, where the inferential activity is viewed as consisting of two phases. First, a translation phase carries a (propositional) modal formalization φ of a problem into its relational counterpart. Then, within the relational context, a deductive method is exploited to seek a proof of the translated formula φ (cf. Fig. 1).

There are several kinds of proof systems for relational reasoning, such as tableaux [17], Gentzen-style systems [34, 22], systems *à la* Rasiowa-Sikorski

* Research partially funded by INTAS project *Algebraic and deduction methods in non-classical logic and their applications to Computer Science*, and by the European Concerted Research Action COST 274, *TARSKI: Theory and Applications of Relational Structures as Knowledge Instruments.*

H. de Swart et al. (Eds.): TARSKI II, LNAI 4342, pp. 89–104, 2006.

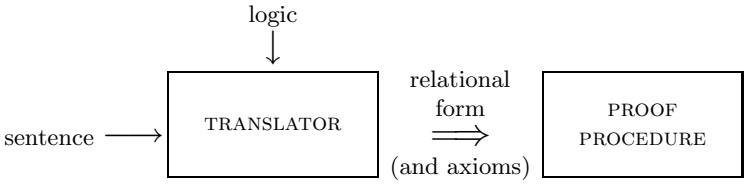

Fig. 1. General scheme of the inferential framework

[25, 30, 14], display calculus [16], and of course equational proof systems based on relation algebras [11, 12]. The system we have in mind should be seen as providing a convenient input for any of those proof systems. Specifically, the input for a tableaux-based system, a Gentzen system, or a Rasiowa-Sikorski system will be an expression of the form $x\, t(\varphi)\, y$, where x and y stand for individual variables and $t(\varphi)$ for a relational term translating the given formula φ, obtainable e.g. by means of a system which we have implemented in Prolog along the lines that will be expounded below. On the other hand, our input for an equational proof system will be an equation $t(\varphi) = \mathbf{1}$, where $\mathbf{1}$ denotes the top element of a relation algebra.

This paper focuses on the translation phase: we describe a prototypical, Prolog-based implementation of a tool, named *transIt*, which uniformly carries out translations from various modal logics to the relational formalism [35]. As an aside, we give some details about possible approaches towards the interaction/integration between the translator and a deductive engine. The development of an efficient relational deductive system (actually, in the Rasiowa-Sikorski style) is the theme of [8].

We verified that this approach offers indeed a high degree of uniformity: *transIt* is able to treat varied modal logics, all by the very same machinery. Moreover, extensions to further families of logics can easily be obtained by routine application of their declarative Prolog specifications.

Moreover, 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. As a matter of fact, in the same easy routine fashion in which source languages can be added, the system can also be extended to encompass other target languages, so as to "drive" different (relational) proof systems. We exemplify this adaptability by extending *transIt* in order to use it as a front-end for two deductive frameworks for relation algebras which are rather different in nature (Section 4). One of the two consists in a minimal implementation of a proof-assistant (with some form of automated capabilities) based on Rasiowa-Sikorski rewriting rules [29]. Actually, this proof-assistant has been easily integrated in *transIt* by means of a common graphical user interface. As a second approach to relational reasoning, we show how *transIt* can be used as a front-end for a first-order theorem-prover which is exploited as relational inference engine very much in the spirit of [11, 12].

The paper is organized as follows. In Section 1 we describe source and target languages. For most of the modal logics, we provide the corresponding translation rules. Section 2 illustrates the architecture of *transIt* and the successive phases of the translation process, while an outline of the input/output formats is given in Section 3. Finally, Sections 4 and 5 describe the interface to the built-in proof-assistant and speculate on improvements to the overall inferential framework one can envisage.

1 Source and Target Languages

The main target language which our translation supports is the algebra of binary relations. For this target, given a formula φ the system produces a relational term $t(\varphi)$ belonging to an algebraic language encompassing the usual constructs of Boolean algebra plus further operators specific to the realm of relations. To be more specific, following the work of Alfred Tarski [35], let us recall the basic notions on such formalism. The intended *universe of discourse* is a collection $\Re$ of binary relations over a non-null domain $\mathcal{U}$. We assume that the *top* relation $\bigcup\Re$, and the *diagonal* relation consisting of all pairs $\langle u, u\rangle$ with u in $\mathcal{U}$, belong to this universe, which is also closed under the *intersection* ($\cap$), *union* ($\cup$), *complement* ($\overline{\ }$) relative to $\bigcup\Re$, *composition* (;), and *converse* ($^\smile$) operations. Within such a system, two primitive constants $\mathbf{1}$ and $\boldsymbol{I}$ designate the top and the diagonal relation, while the operations are interpreted as one expects (here, for any relational expression R we are indicating by $R^{\Im}$ the relation over $\mathcal{U}$ designated by R), for instance:

- $P^\smile$ designates the relation consisting of all pairs $\langle v, u\rangle$ with $\langle u, v\rangle$ in $P^{\Im}$;
- $P;Q$ designates the relation consisting of all pairs $\langle u, w\rangle$ such that there is at least one v for which $\langle u, v\rangle$ and $\langle v, w\rangle$ belong to $P^{\Im}$ and to $Q^{\Im}$, respectively;
- $P{\cap}Q$ designates the relation consisting of all pairs $\langle u, v\rangle$ which simultaneously belong to $P^{\Im}$ and to $Q^{\Im}$;

and similarly for the other constructs.

Designations for further constants, operations over relations, or equations of a special kind, can be introduced through definitions, e.g.:

$$\begin{array}{rcl@{\qquad\qquad}rcl} \mathbf{0} & =_{\text{Def}} & \overline{\mathbf{1}}, & \boldsymbol{D} & =_{\text{Def}} & \overline{\boldsymbol{I}}, \\ P{-}Q & =_{\text{Def}} & P{\cap}\overline{Q}, & P{+}Q & =_{\text{Def}} & (Q{\cup}P){-}(Q{\cap}P), \\ P{\sqsubseteq}Q & \leftrightarrow_{\text{Def}} & P{-}Q{=}\mathbf{0}. & & & \end{array}$$

Another target language currently supported is the binary first-order predicate calculus with three variables, namely $\mathcal{L}_3$ [35]. For this target, the translation is obtained by first performing the translation into the algebra of relations, and then exploiting first-order characterizations of the relational operators. Clearly, in order to limit the overall number of first-order variables to three, in doing the latter transformation we must rely on a suitable variable-recycling mechanism.

It should be noted that a first-order sentence is logically equivalent to a sentence of $\mathcal{L}_3$ if and only if it is expressible in the algebra of relations. This is because $\mathcal{L}_3$ is equipollent to the arithmetic of binary relations [35, Chap. 3]. On the other hand, this is no more the case if we consider sentences of the full first-order predicate calculus. Actually, it is known (cf. [35, 21]) that the collection of all first-order sentences expressible with three variables (and hence having a relational rendering) is undecidable. As a consequence, the translation from first-order predicate calculus into the algebra of relations is not always doable and it can only be achieved (in favorable cases) by means of conservative techniques. Therefore, our Prolog-based translator may fail in translating a sentence. Anyway, the translation process terminates in every case, and a diagnostic message is issued when the translation is not carried through. Notice that the translation process could be improved by resorting to conservative refinements such as those proposed in [2].

Similar enhancements can be applied in order to build more target languages into the tool. One could easily achieve this goal by describing such languages in terms of suitable rewriting rules. As an example we mention another currently available translation for modal formulas (see below, for a description of the source languages), having a set-theoretical language as target. This approach is described in [4, 1, 31], where it is shown that even a very weak set theory can offer adequate means for expressing the semantics of modal systems of propositional logic. In this context, a modal formula is translated into a formula of a very weak set theory. Then, in order to perform (semi-)automated modal inference, the result of the translation could be fed into a deduction system for theory-based reasoning [13] or, alternatively, into a Rasiowa-Sikorski proof system for set-theory, as described in [31]. Another possibility could consist in performing one further translation step, from the set theoretical framework into the relational calculus, as suggested in [9], to then exploit any deductive system for relational reasoning.

Let us now briefly highlight most of the source languages currently accepted by the translator. We characterize the languages of the logics which employ binary accessibility relations in terms of their Kripke-style models. Our translator does not, as yet (although we plan extensions of this kind), deal with the languages of relevant logics or the logics with binary modalities—requiring ternary relations in their models. The translation functions for many of these languages are known, see [26] for a translation of languages of relevant logics.

The main idea of the translation is to assign relational terms to formulas of non-classical logics so that validity is preserved. These terms must represent *right ideal* relations, a binary relation R on the domain $\mathcal{U}$ being called right ideal when it meets the condition $R;\mathbf{1}{=}R$. In other words, a right ideal relation is of the form $X \times \mathcal{U}$ for some $X \subseteq \mathcal{U}$. Intuitively speaking, if a formula is replaced by a right ideal relation, then its domain represents the set of states where the formula is true, and its range represents the universe of all states. For atomic formulas the property of being right ideal can be enforced by postulating that a propositional variable, say p, is translated into a relational term $P;\mathbf{1}$, where

P is a relation variable uniquely associated with p. It follows that, given a language, a relational translation of its formulas can be defined provided that, first, the propositional operations of the language can be mapped into the relational operations which preserve the property of being right ideal and, second, the translation will preserve validity. It is known that Boolean operations preserve the property of being right ideal and the composition of any relation with a right ideal relation results in a right ideal relation. So if a logic is based on a classical logic whose propositional connectives are Boolean, or if a logic has a lattice as a basis, then the only problem is to appropriately translate the remaining intensional propositional operations of the logic. Since their semantics depends on the accessibility relation(s) which usually are not right ideal, the translation should use these relations only as first arguments of the composition operator, making use of the property stated above. If this can be done with preservation of validity, then the translation process is successful.

In the following we present definitions of the translation functions of languages for several families of logics whose accessibility relations are binary. In all the listed cases the validity-preserving theorems are known and can be found in the cited references.

Mono-Modal Logics. This is the basic translation of (propositional) modal formulas into relational terms originated in [25]. The source language involves usual propositional connectives together with necessity and possibility operators (here ψ and χ stand for propositional sentences):

- $t(p_i) \;=_{\text{Def}}\; P_i\,;\,\mathbf{1}$, where P_i is a relational variable uniquely corresponding to the propositional variable p_i;
- $t(\neg\,\psi) \;=_{\text{Def}}\; \overline{t(\psi)}$;
- $t(\psi\,\&\,\chi) \;=_{\text{Def}}\; t(\psi) \cap t(\chi)$;
- $t(\Diamond\,\psi) \;=_{\text{Def}}\; \mathsf{R}\,;\,t(\psi)$, where R is a constant relation designating the *accessibility relation* between possible worlds;

and similarly for the other customary propositional connectives (see also [25], for a very detailed treatment).

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 or necessity or sufficiency or dual sufficiency operator. 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. The target relational language for all of these lattice-based logics includes the following specific accessibility relations: binary relations $\leqslant_1$ and $\leqslant_2$, which are assumed to be reflexive and transitive and to satisfy the condition $\leqslant_1 \cap \leqslant_2 = \boldsymbol{I}$. Such relations are needed in order to provide semantics for the operation of disjunction which, in the case of lattice-based logics, does not necessarily distribute over conjunction. All of these logics have been deeply investigated in [32, 7, 20]. The translation of disjunction and conjunction is:

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

Considering a source language with a possibility operator $\Diamond$, the target language includes two relations $R_\Diamond$ and $S_\Diamond$ subject to the following conditions:

$$\breve{\leqslant_1}; R_\Diamond; \breve{\leqslant_1} \sqsubseteq R_\Diamond\,, \qquad R_\Diamond \sqsubseteq S_\Diamond; \breve{\leqslant_1}\,,$$
$$\leqslant_2; S_\Diamond; \leqslant_2 \sqsubseteq S_\Diamond\,, \qquad S_\Diamond \sqsubseteq \leqslant_2; R_\Diamond\,.$$

The translation of a formula involving the modal operator is

$$t(\Diamond\chi) \;=_{\text{Def}}\; \overline{\leqslant_1; \overline{S_\Diamond; \leqslant_2; t(\chi)}}\,.$$

Also in the case of a language involving the necessity operator $\Box$, the target language includes two relations $R_\Box$ and $S_\Box$ subject to:

$$\leqslant_1; R_\Box; \leqslant_1 \sqsubseteq R_\Box\,, \qquad R_\Box \sqsubseteq \leqslant_1; S_\Box\,,$$
$$\breve{\leqslant_2}; S_\Box; \breve{\leqslant_2} \sqsubseteq S_\Box\,, \qquad S_\Box \sqsubseteq R_\Box; \breve{\leqslant_2}\,.$$

The translation of a formula involving the modal operator is

$$t(\Box\chi) \;=_{\text{Def}}\; \overline{R_\Box; \overline{t(\chi)}}\,.$$

Formulas involving the sufficiency operator $\boxplus$ are translated into relational expressions by introducing two relations $R_\boxplus$ and $S_\boxplus$ subject to the following conditions:

$$\leqslant_1; R_\boxplus; \leqslant_2 \sqsubseteq R_\boxplus\,, \qquad R_\boxplus \sqsubseteq \leqslant_1; S_\boxplus\,,$$
$$\breve{\leqslant_2}; S_\boxplus; \breve{\leqslant_1} \sqsubseteq S_\boxplus\,, \qquad S_\boxplus \sqsubseteq R_\boxplus; \breve{\leqslant_1}\,.$$

Within such a framework, the translation of a formula involving the sufficiency operator is

$$t(\boxplus\chi) \;=_{\text{Def}}\; \overline{R_\boxplus; \leqslant_2; t(\chi)}\,.$$

Finally, the translation of formulas involving the dual sufficiency operator $\diamondsuit$, has as its target a relational language with two relations, $R_\diamondsuit$ and $S_\diamondsuit$, subject to the following conditions:

$$\breve{\leqslant_1}; R_\diamondsuit; \breve{\leqslant_2} \sqsubseteq R_\diamondsuit\,, \qquad R_\diamondsuit \sqsubseteq S_\diamondsuit; \breve{\leqslant_2}\,,$$
$$\leqslant_2; S_\diamondsuit; \leqslant_1 \sqsubseteq S_\diamondsuit\,, \qquad S_\diamondsuit \sqsubseteq \leqslant_2; R_\diamondsuit\,.$$

Then, the translation of a formula involving $\diamondsuit$ is

$$t(\diamondsuit\chi) \;=_{\text{Def}}\; \overline{\leqslant_1; \overline{S_\diamondsuit; \overline{t(\chi)}}}\,.$$

Logics of Knowledge and Information. These modal logics come from [5]:

⋆ Logic with knowledge operator **K**, subject to the following translation rule:

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

- ⋆ Logic of non-deterministic information (NIL) [5, Sect. 7.2]. A multi-modal logic with three modalities, determined by the relations of informational inclusions ($\leqslant$ and $\geqslant$) and similarity (σ) subject to the following conditions:
  - $\leqslant$ is reflexive and transitive and such that $\leqslant = \geqslant^{\smile}$,
  - σ is reflexive and symmetric,
  - $\geqslant; \sigma; \leqslant \sqsubseteq \sigma$.
- ⋆ Information logic (IL) [5, Sect. 7.3]. A modal logic with three modal operators corresponding to the relations of indiscernibility ($\equiv$), forward inclusion ($\leqslant$), and similarity (σ) subject to the following conditions:
  - $\equiv$ is an equivalence relation,
  - $\leqslant$ is reflexive and transitive,
  - σ is reflexive and symmetric,
  - $\leqslant^{\smile}; \sigma \sqsubseteq \sigma$ and $\leqslant \cap \leqslant^{\smile} = \equiv$.

Intuitionistic Logic. The translation of intuitionistic logic is based on the following rules:

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

where $\leqslant$ is a reflexive and transitive relation.

Multi-modal Logic. These logics correspond to multi-modal frames consisting of a relational system (W, Rel) where Rel is a family of accessibility relations (enjoying closure properties with respect to relational constructs). Modalities are then of the form $[R]$ and $\langle R \rangle$, where R is any relational term of Rel (cf. [27]).

The translation of modal operators is the same as in the case of mono-modal logic. The differences between operators are articulated in terms of the properties of the corresponding accessibility relations.

Temporal Logics. By taking the relational formalization of temporal logics given in [28], we extended the translator in order to deal with temporal formulas. The basic modal operators (referring to states in the future or in the past) are:

- $\mathtt{G}\varphi$ interpreted as "always, in the future, φ will be true";
- $\mathtt{F}\varphi$ interpreted as "sometimes, in the future, φ will be true ";
- $\mathtt{H}\varphi$ interpreted as "φ was always true in the past";
- $\mathtt{P}\varphi$ interpreted as "φ was true in some past time";
- $\varphi\, \mathtt{U}\, \chi$ interpreted as "at some moment χ will be true and from now till then φ will be true";
- $\varphi\, \mathtt{S}\, \chi$ interpreted as "there was a moment when χ was true and such that φ has ever since been true";
- $\mathtt{X}\varphi$ interpreted as "φ will be true in the next moment in time".

In this context, relational representations of temporal formulas 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 $\mathtt{G}$, $\mathtt{F}$, $\mathtt{H}$, and $\mathtt{P}$ are as follows:

$$\begin{aligned} t(\mathtt{G}\varphi) &=_{\text{Def}} \overline{\mathsf{R}; \overline{t(\varphi)}}\,, & t(\mathtt{H}\varphi) &=_{\text{Def}} \overline{\mathsf{R}^\smile; \overline{t(\varphi)}}\,, \\ t(\mathtt{F}\varphi) &=_{\text{Def}} \mathsf{R}; t(\varphi)\,, & t(\mathtt{P}\varphi) &=_{\text{Def}} \mathsf{R}^\smile; t(\varphi)\,, \\ t(\varphi\,\mathtt{U}\,\chi) &=_{\text{Def}} t(\varphi)\,\mathtt{U}\,t(\chi)\,, & t(\varphi\,\mathtt{S}\,\chi) &=_{\text{Def}} t(\varphi)\,\mathtt{S}\,t(\chi)\,, \\ t(\mathtt{X}\varphi) &=_{\text{Def}} t((\varphi \,\&\, \neg\,\varphi)\,\mathtt{U}\,\varphi)\,, \end{aligned}$$

where, in the translations of the modal operators $\mathtt{U}$ and $\mathtt{S}$ we use the same symbols to denote two newly introduced relational constructs. These new constructs cannot be defined in terms of the primitive relational constructs (page 91). The intended interpretation of $\mathtt{U}$ is as follows: $P\mathtt{U}Q$ 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 $\mathsf{R}^{\mathfrak{I}}$, $\langle t, v\rangle$ belongs to $Q^{\mathfrak{I}}$, and for all w, if $\langle u, w\rangle \in \mathsf{R}^{\mathfrak{I}}$ and $\langle w, t\rangle \in \mathsf{R}^{\mathfrak{I}}$ then $\langle w, v\rangle \in P^{\mathfrak{I}}$. (The interpretation of $\mathtt{S}$ is analogous, with respect of $\mathsf{R}^\smile$.)

Other Modal Logics. Other modal logics currently accepted by the translator involve: logics with specification operators [18, 24], logics with Humberstone operators [19], logics with sufficiency operators [15, 6].

Following the semantics developed by Hoare and Jifeng [18], the operators of the weakest prespecification ($\backslash$) and the weakest postspecification ($/$) are modeled with residuals of the relational composition which are definable with composition, converse and complement:

$$Q\backslash R =_{\text{Def}} \overline{\overline{R}; Q^\smile} \qquad \text{and} \qquad R/P =_{\text{Def}} \overline{P^\smile; \overline{R}}.$$

Consequently, $P; Q \sqsubseteq R$ if and only if $P \sqsubseteq Q\backslash R$ if and only if $Q \sqsubseteq R/P$.

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 ($\boxplus$) and dual sufficiency ($\diamondsuit$) operators receive the following relational translation:

$$\begin{aligned} t(\boxplus\,\varphi) &=_{\text{Def}} \overline{\overline{\mathsf{R}}; \overline{t(\varphi)}}, \\ t(\diamondsuit\varphi) &=_{\text{Def}} \overline{\mathsf{R}}; \overline{t(\varphi)}, \end{aligned}$$

where R is a relational constant representing an accessibility relation of the models of a logic under considerations.

It follows that our translation tool is able to translate the formulas of any of the information logics presented in [5], as they involve, together with Boolean or lattice operators, intensional operators that are either modal or sufficiency or knowledge operators.

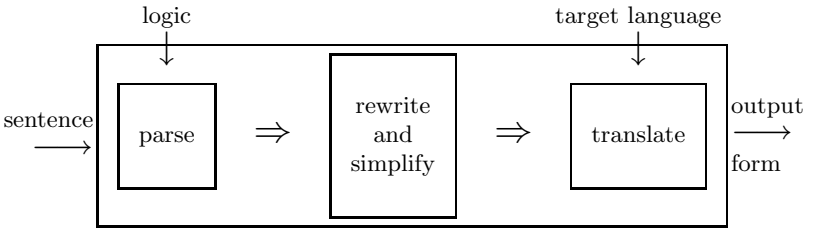

Fig. 2. The translator architecture

2 The Translation Process

The translator takes as input a formula of a specific source language (see Section 1). As shown in Fig. 2, the first of these transformations yields an internal representation of the formula, while the last step generates its final rendering. Then, a sequence of rewritings and simplifications is performed. Finally, the desired translation is produced.

More specifically, here is the sequence of the salient phases which usually form the translation (some of them being skipped in specific cases, for instance double-negation removal in intuitionistic logic):

Lexical and syntactical analyses. This phase accepts a formula only if it is syntactically correct and its constructs belong to the specific language at hand. The syntax-directed translation implemented through this stage is described by an attributed definite clause grammar. Hence, any extension to further logics can be achieved by simply adding a suitable set of grammar rules which characterize the (new) well-formed formulas. The outcome of this stage is an intermediate representation of the abstract syntax tree (AST) of the input formula.

Generation of an internal representation. By means of a rewriting process which acts in a bottom-up recursive fashion, the outcome of the preceding phase is turned into an internal representation of the AST (in form of a Prolog term), independent of the source language.

Abstract propositional evaluation. The internal representation of the given formula is analyzed in order to extract its propositional schema. The schema so obtained is then (possibly) simplified through replacements of some of its sub-formulas by tautologically equivalent ones.

Reduction to primitive constructs. In this phase the formula is rewritten in terms of a small repertoire of constructs and connectives, to be regarded as being "primitive". For instance, biimplication $\leftrightarrow$ is rewritten as a conjunction of two implications, and so on. Notice that some of these rewritings must be inhibited at times, insofar as unsound with respect to the specific logic at hand. The aim of this transformation is to make the next phase easier.

Propositional simplifications. Through this phase the internal representation of the formula is simplified by applying a number of propositional

```
rewrite(Rules,From,To) :- transl(Rules,From,M),    % rewrite until
        (From==M, To=M ; rewrite(Rules,M,To)).     % fix-point

transl(_,T,T) :- var(T).
transl(R,T,S) :- T =.. [F|Argg], translArgg(R,Argg,Brgg),
                 M =.. [F|Brgg], transl0(R,M,S).
translArgg(_,[],[]).
translArgg(R,[H|B],[SH|SB]) :- transl(R,H,SH),
                               translArgg(R,B,SB).
transl0(R,T,S) :- Goal =.. [R,T,S],  (Goal ; S=T).

rewrite1(R,T,S) :-  once(transl(R,T,S)).        % rewrite once
```

Fig. 3. A simple and powerful post-order rewriting procedure

simplifications to it, mainly aimed at reducing the size of the formula (for instance, elimination of tautological sub-formulas and of double negations).

Relational translation. This is the main step of the translation process: the internal representation of the given formula is translated into the calculus of binary relations. The kind of rewriting rules employed may depend on the source language of the input formula (see Section 1). The outcome of this phase is a relational term.

Relational simplifications. The overall translation process ends with a series of relational simplifications applied to the relational term produced by the preceding step. The simplest among these rewritings take care of the idempotency, absorption or involution properties of (some of) the relational constructs. The process can easily be extended to perform more complex simplifications.

It should be noticed that most of the above steps are all uniformly performed by exploiting the same simple meta-rewriter. Fig. 3 displays the basic Prolog specification of this post-order rewriting procedure. The main predicate is `rewrite/3`. Intuitively speaking, it accepts as its first parameter (`Rules`) a Prolog predicate describing one of the possible translation steps. Then it recursively processes the term `From` in order to produce its translation `To`.

Further phases could be added, for instance in order to apply semantical transformations to the relational term, possibly with respect to a set of axiomatic assumptions characterizing a particular class of relational structures as constituting the target framework.

Example 1. As an example we provide here the textual output produced by the various steps of the translation into the calculus of relations of the multi-modal formula:

$$[R \cup Q] < Q > p \rightarrow q.$$

Here is a tracing of the translation process (where `p1`, `p2`, and `R3` are internal names corresponding to the external names p, q, and Q, respectively):

```
?- enu2tg(A,polyModal).
|: [R+Q]<Q>p -> q.
...i(nn(pp(1,-3),u(0,-3)),2)...
from intermediate to internal representation:
    NEC(POSS(p1,R3),u(R,R3))imp p2...
in primitive connectives:
    NEC(NEC(p1 imp f,R3)imp f,u(R,R3))imp p2...
after propositional simplification:
    NEC(NEC(p1 imp f,R3)imp f,u(R,R3))imp p2...
after translation to calculus of relations:
    u(c(c(k(u(R,R3),c(u(c(c(k(R3,c(u(c(k(p1,U)),Z))))),Z))))),k(p2,U))...
after relational simplifications:
    u(k(p2,U),k(u(R,R3),c(k(k(R3,p1),U))))...

A = (u(k(p2,'U'),k(u('R','R3'),c(k(k('R3',p1),'U'))))='U')
```

The Prolog term produced is the representation of the relational equality

$$q\,;\,\mathbf{1} \cup (R \cup Q)\,;\,\overline{Q\,;\,p\,;\,\mathbf{1}} = \mathbf{1}.$$

Proving that the initial modal formula is a theorem amounts to deriving this equation within the calculus of relations.

3 Input and Output Formats

When rawly used, our Prolog-based translation tool system reads a pure-text input typed in by the user (cf. Example 1). The output is then written, again in pure-text format, to the standard output stream (usually, the screen). This kind of interaction is, however, quite unsatisfactory, because the ASCII character set is rather poor. In order to overcome this disadvantage and ease the input/output of complex formulas and expressions, a user-friendly interface has been developed. Such a graphical interface allows the user to type in formulas using graphical LaTeX-generated symbols. In doing this, we exploited the useful integration facilities offered by SICStus Prolog [37] with respect to other programming languages, in particular to the `Tcl/Tk` toolkit [36]. Hence, the input of formulas is achieved through dialogues that are generated at run-time depending on the specific language chosen by the user. For instance, Fig. 4 displays the input dialogue generated for multi-modal formulas.

The system also provides the possibility of processing a text file, as well as to generate a text file as output. Through this feature it is possible to produce input files for different deduction tools (see Section 4).

Let us briefly illustrate the use of the graphical interface with a simple example. Consider the multi-modal formula $[R \cup Q] < Q > p \rightarrow q$. This formula can be input to the translator easily, as shown in Fig. 4. The relational equation obtained can also be displayed graphically, as in Fig. 5.

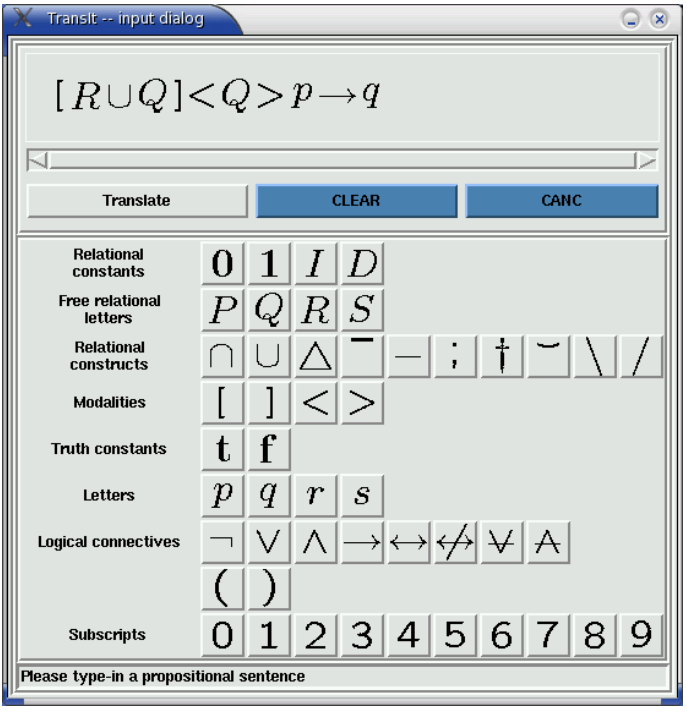

Fig. 4. Input dialogue for multi-modal formulas

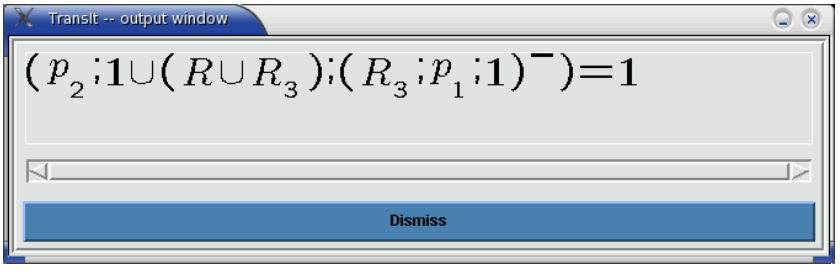

Fig. 5. Output of a translation process

4 Driving a Deductive Tool

As mentioned at the outset, the main purpose of *transIt* is to provide an extensible front-end for (relational) deductive systems.

To exemplify how well this goal is approached, in what follows we report on two extensions of *transIt*, designed in order to use it as a front-end for two deductive frameworks for relation algebras which are rather different in nature. One

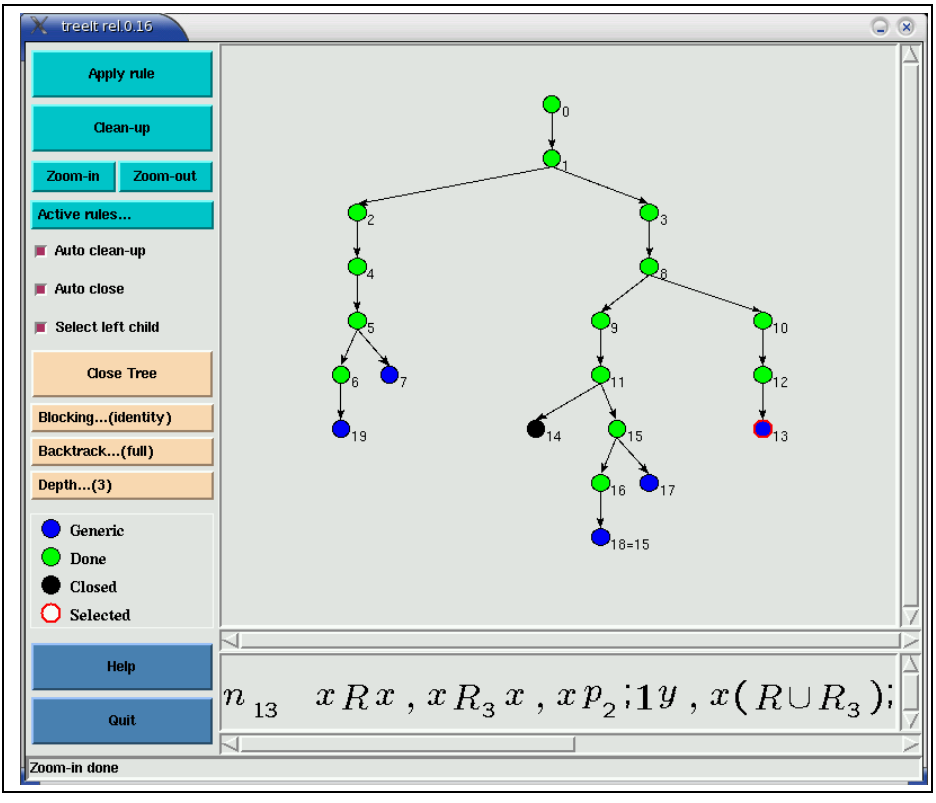

Fig. 6. Assisted development of a proof tree

of the two consists in a minimal implementation of a proof-assistant (showing some degree of autonomy) based on Rasiowa-Sikorski rewriting rules [29]. Such proof-assistant is accessible through *transIt*'s graphical interface. Once the user has obtained a relational rendering of a theorem, (s)he can proceed to try building a proof-tree of the relational translation. Fig. 6 shows a simple example of a derivation tree. The user interacts with the system by simply choosing a node of the tree in order to apply one of the rewriting rules. The system takes care of verifying applicability of rules, performing the extension of the tree, and checking whether, as a consequence of rule applications, any branch becomes closed. Some form of (semi-)automated reasoning capabilities are also implemented: it is possible to ask the system to try, autonomously, to close all branches of a (sub-)tree.

Another viable approach to relational reasoning consists in using *transIt* as a front-end for a first-order theorem-prover: Otter, in our choice. This is achieved by extending the translation process: a new set of rewriting rules is used to implement automated generation of an input file to be fed into Otter. Once the input file is available, Otter can be used as described in [11, 12] to search for a

proof of the theorem within the relational framework. Obviously, the very same approach can be used with other theorem provers.

Currently, *transIt* can be downloaded from the site `http://www.di.univaq.it/TARSKI/transIt/` and easily installed. It is developed under Linux, but we also provide a porting for Windows XP.

5 Improving the System

The modular approach we adopted both in developing the translator and in extending the collection of source and target languages, plainly permits steady improvements to and extensions of the system. At the moment, most of the phases of the translation process are carried out by means of syntactical rewritings. Nevertheless, the translation process could benefit from improvements to its ability to exploit semantic properties of connectives and constructs. As a matter of fact, in the current implementation this ability lies exclusively in the abstract propositional evaluation phase (see page 97).

Another amelioration, in the same frame of mind, is the exploitation, in the derivation process (both for the assisted and for the autonomous functioning mode), of specific rewriting rules depending on the particular logic of the theorem being proved.

As mentioned, the system can deal with different target languages (see page 92). As a further example, we mention here a particularly interesting future development: extend the collection of target languages, so as to permit the translation into languages of ternary relations needed for handling relevant logics and other substructural logics whose translations are presented in [26].

Further challenging themes for long-term activities regard exploring the possibilities offered by

- the integration with/within existing tools for translation and deduction. In particular, a fruitful synergy could develop from the integration/interaction with the "Anamorpho system", an environment for describing relational specifications which is based on definitional extension mechanisms (see [3]).
- the integration with visual-oriented tools for manipulation of relational formulas (based, for instance, on graphical representation of relational expressions and on graph-rewriting techniques [10]).

Acknowledgments

Thanks are due to Marianna Nicolosi Asmundo and to Hui Wang for fruitful discussions on the topics of this paper.

References

[1] J. F. A. K. van Benthem, G. D'Agostino, A. Montanari, and A. Policriti. Modal deduction in second-order logic and set theory-I, *Journal of Logic and Computation*, **7**(2):251–265, 1997.

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

[3] P. Caianiello, S. Costantini, E. G. Omodeo. An Environment for Specifying Properties of Dyadic Relations and Reasoning about Them. I: Language Extension Mechanisms. In H. de Swart, E. Orłowska, G. Schmidt, and M. Roubens eds., *Theory and Applications of Relational Structures as Knowledge Instruments*, LNCS **2929**, pp.87–106, Springer, 2004.

[4] G. D'Agostino, A. Montanari, and A. Policriti. A set-theoretic translation method for polymodal logics, *Journal of Automated Reasoning*, **3**(15):317–337, Kluwer, 1995.

[5] S. Demri, E. Orłowska. Incomplete Information: Structure, Inference, Complexity. EATCS Monographs in Theoretical Computer Science, Springer, 2002.

[6] I. Düntsch and E. Orłowska. Beyond modalities: sufficiency and mixed algebras. In E. Orłowska and A. Szalas eds., *Relational Methods in Computer Science Applications*. pp.277–299. Physica-Verlag, Heidelberg, 2000.

[7] I. Düntsch, E. Orłowska, A. M. Radzikowska, and D. Vakarelov. Relational Representation Theorems for Some Lattice-Based Structures. *Journal on Relational Methods in Computer Science*, **1**:132–160, 2004.

[8] A. Formisano and M. Nicolosi Asmundo. An efficient relational deductive system for propositional non-classical logics. *Journal of Applied Non-Classical Logics*, to appear. (A draft version is available as report 8/06 of the Dipartimento di Informatica, Università di L'Aquila, 2006.)

[9] A. Formisano, E. Omodeo, E. Orłowska, and A. Policriti. Uniform relational frameworks for modal inferences. In I. Düntsch and M. Winter eds., Proceedings of the 8$^{\text{th}}$ International Conference on Relational Methods in Computer Science (RelMiCS 8), 2005.

[10] A. Formisano, E. Omodeo, M. Simeoni, A graphical approach to relational reasoning. *Electronic Notes in Theoretical Computer Science*, **44**(3), Elsevier, 2001.

[11] A. Formisano, E. G. Omodeo, M. Temperini. Instructing Equational Set-Reasoning with Otter. In R. Goré, A. Leitsch, and T. Nipkow eds., *Automated Reasoning*, Proceedings of IJCAR 2001. LNCS **2083**, pp.152–167, Springer, 2001.

[12] A. Formisano, E. G. Omodeo, M. Temperini. Layered map reasoning: An experimental approach put to trial on sets. *Electronic Notes in Theoretical Computer Science*, **48**, Elsevier, 2001.

[13] A. Formisano and A. Policriti. T-Resolution: Refinements and Model Elimination. *Journal of Automated Reasoning*, **22**(4):433–483, Kluwer, 1999.

[14] M. Frias and E. Orłowska. Equational reasoning in nonclassical logics. *Journal of Applied Non-Classical Logics*, **8**(1-2):27–66, 1998.

[15] V. Goranko. Modal definability in enriched languages. *Notre Dame Journal of Formal Logic*, **31**:81–105, 1990.

[16] R. Goré. Cut-free display calculi for relation algebras. In D. van Dalen and M. Bezem eds., *CSL96: Selected Papers of the Annual Conference of the Association for Computer Science Logic*, LNCS **1258**, pp.198–210. Springer, 1996.

[17] M. C. B. Hennessy. A proof system for the first order relational calculus. *Journal of Computer and System Sciences*, **20**:96–110, 1980.

[18] C. A. R. Hoare and H. Jifeng. The weakest prespecification. Part I *Fundamenta Informaticae*, **IX**:51–84; Part II ibidem **IX**:217-252, 1986.

[19] I. L. Humberstone. Inaccessible worlds. *Notre Dame Journal of Formal Logic*, **24**:346–352, 1983.

[20] J. Järvinen and E. Orłowska. Relational correspondences for lattices with operators. In I. Düntsch and M. Winter eds., Proceedings of the 8th International Conference on Relational Methods in Computer Science (RelMiCS 8), pp.111–118, 2005.

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

[22] R. Maddux. A sequent calculus for relation algebras. *Annals of Pure and Applied Logic*, **25**:73–101, 1983.

[23] H. J. Ohlbach, A. Nonnengart, M. de Rijke, and D. Gabbay. Encoding Two-Valued Nonclassical Logics in Classical Logic, In *Handbook of Automated Reasoning*, vol. II, pp.1403–1486, Elsevier, 2001.

[24] E. Orłowska. Proof system for weakest prespecification. *Information Processing Letters*, **27**:309–313, 1988.

[25] E. Orłowska. Relational interpretation of modal logics, In H. Andreka, D. Monk, and I. Nemeti eds., *Algebraic Logic*. Colloquia Mathematica Societatis Janos Bolyai, vol. 54, pp.443–471, North Holland, 1988.

[26] E. Orłowska. Relational proof systems for relevant logics, *Journal of Symbolic Logic*, **57**, pp.167–186. 1992.

[27] E. Orłowska. Relational semantics for nonclassical logics: formulas are relations, Philosophical Logic in Poland (J. Wolenski, ed.), pp.167–186. Kluwer, 1994.

[28] E. Orłowska. Temporal logics in a relational framework. Time and Logic—A computational Approach. pp.249–227. University College London Press, 1995.

[29] E. Orłowska. Relational proof systems for modal logics, In H. Wansing ed., *Proof theory of modal logic*, Applied logic series, vol.2, pp.55–78. Kluwer, 1996.

[30] E. Orłowska. Relational formalization of nonclassical logics. In C. Brink, W. Kahl, and G. Schmidt eds., *Relational Methods in Computer Science*, pp.90-105. Springer, 1997.

[31] E. G. Omodeo, E. Orłowska, and A. Policriti. Rasiowa-Sikorski style relational elementary set theory. In R. Berghammer and B. Moeller eds., Proceedings 7th International Conference on Relational Methods in Computer Science (RelMiCS 7), LNCS **3051**, pp. 215-226, Springer, 2003.

[32] E. Orłowska and D. Vakarelov. Lattice-based modal algebras and modal logics. In P. Hajek, L. M. Valdés-Villanueva, D. Westerstahl eds., Logic, Methodology and Philosophy of Science. Proceedings of the 12th International Congress, pp.147–170. KCL Puplications, 2005.

[33] R. Schmidt and U. Hustadt. Mechanized reasoning and model generation for extended modal logics. In H. de Swart, E. Orłowska, G. Schmidt, and M. Roubens, eds., *Theory and Applications of Relational Structures as Knowledge Instruments*, LNCS **2929**, pp.38–67, Springer, 2004.

[34] W. Schoenfeld. Upper bounds for a proof search in a sequent calculus for relational equations. *Zeitschrift fuer Mathematische Logic und Grundlagen der Mathematik* **28**:239–246, 1982.

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

[36] Web resources for the `Tcl/Tk` toolkit: `http://tcl.sourceforge.net`.

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