Plan of activities on the map calculus

Andrea Formisano, Eugenio Omodeo, Marco Temperini

Abstract

Tarski-Givant's map calculus is briefly reviewed and a plan of research is outlined, aimed at investigating applications of this formalism in the theorem-proving field. The connections between first-order logic and the map calculus are investigated, focusing on techniques for translating single sentences from one context to the other as well as on the translation of entire set theories. Issues regarding 'safe' forms of map reasoning are singled out, in sight of possible generalizations to the database area.

Keywords: Algebraic logic, Relation algebras, First-order theorem-proving.

Introduction

Everybody remembers that Boole's *Laws of thought* (1854), Frege's *Begriffsschrift* (1879), and the Whitehead-Russell's *Principia Mathematica* (1910) have been three major milestones in the development of contemporary logic (cf. [3, 10, 21, 4]). Only a few people are aware that very important pre-*Principia* milestones were laid down by C.S. Peirce and E. Schröder and culminated in the monumental work [16] on the *Algebra der Logik*.

The "rather capricious line of historical development" of the algebraic form of logic —the MAP CALCULUS, as we will call it— that Peirce and Schröder had contributed to create and which short after the appearance of the *Principia* fell into general oblivion, was already signaled as an anomaly by Tarski in 1941 (cf. [18]). Even more anomalous it should be considered today, since the footprint of the map language can be recognized in relational database languages, and since, moreover, the subsequent work of Tarski and others (cf. [19]) made it clear that map calculus had no inner weaknesses preventing it from becoming the frame for an omni-comprehensive deductive system such as set theory. The rehabilitation of map calculus from its disrepute (whose historical causes are skillfully investigated in [1]) has reopened the opportunity, dismissed for decades, to put together complementary virtues of the map calculus and of first-order predicate logic.

This motivates us in proposing in this paper a few directions for research on the map calculus, aimed at bringing to light its practical value for formal computer-based verification.

A few scenarios of use of Tarski-Givant's map calculus are developed. Properties of familiar structures (natural numbers, lists, sets) endowed with operations and relations

A. Formisano is with Univ. "La Sapienza" of Rome, Dept. of Computer Science. E-mail: formisan@dm.univaq.it

E. Omodeo is with Univ. of L'Aquila, Dept. of Pure and Applied Mathematics. E-mail: omodeo@univaq.it

M. Temperini is with Univ. "La Sapienza" of Rome, DIS. E-mail: marte@dis.uniroma1.it

Work partially supported by CNR of Italy, coordinated project SETA, and by MURST 40%, "Tecniche speciali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di programmi".

are formally specified. Exercise of this nature, largely based on paper and pencil for the time being, is aimed at bringing to light translation techniques that may effectively bridge the gap between first-order predicate calculus and the map calculus. Two such translation techniques, suffering from the limitation of relying merely on syntax, are examined in detail.

It is also discussed in what way a state-of-the-art theorem-prover for first-order logic can be exploited to emulate, and reason about, map calculus—at least on a temporary basis, until tools specifically designed for the latter come into existence.

To end, we envisage the design of a 'safe' version of map calculus, paradigmatic of the way one views relations in the database field.

1 Syntactic and semantic background: $\mathcal{L}^{\times}$ and $\mathcal{L}^{+}$

$\mathcal{L}^{\times}$ is a ground equational language where one can state properties of binary relations over an unspecified, yet fixed, domain $\mathcal{U}$ of discourse. The basic ingredients of this language are:

- three CONSTANTS: $\emptyset$, $\mathbb{1}$, ι;
- infinitely many MAP LETTERS: $\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3, \ldots$ (whose typographic form can widely vary, e.g., suc, $*$, $+$, $\in$);
- binary constructs $\cap$, $\triangle$, $\circ$ of map INTERSECTION, map SYMMETRIC DIFFERENCE, and map COMPOSITION;
- the unary construct $^{-1}$ of map INVERSION.

MAP EXPRESSIONS are obtained through repeated use of $\cap$, $\triangle$, $\circ$, and $^{-1}$, starting from the map letters p_i, which can be freely interpreted as binary relations over $\mathcal{U}$, and from the mentioned constants. MAP EQUALITIES have the form $Q{=}R$, where Q and R are map expressions.

Once a nonempty $\mathcal{U}$ has been fixed and subsets $\mathsf{p}_1^{\Im}, \mathsf{p}_2^{\Im}, \mathsf{p}_3^{\Im}, \ldots$ of $\mathcal{U}^2 =_{\text{Def}} \mathcal{U} \times \mathcal{U}$ have been put in correspondence with the p_is , each map expression P comes to designate a specific map $P^{\Im}$, on the basis of the following evaluation rules:

$$\emptyset^{\Im} =_{\text{Def}} \emptyset, \qquad \mathbb{1}^{\Im} =_{\text{Def}} \mathcal{U}^2, \qquad \iota^{\Im} =_{\text{Def}} \{[a,a] : a \in \mathcal{U}\};$$
$$(Q \cap R)^{\Im} =_{\text{Def}} \{\, [a,b] \in Q^{\Im} : [a,b] \in R^{\Im} \,\},$$
$$(Q \triangle R)^{\Im} =_{\text{Def}} \{\, [a,b] \in \mathcal{U}^2 : [a,b] \in Q^{\Im} \text{ iff } [a,b] \notin R^{\Im} \,\};$$
$$(Q \circ R)^{\Im} =_{\text{Def}} \{\, [a,b] \in \mathcal{U}^2 : \text{there is a } c \in \mathcal{U} \text{ for which } [a,c] \in Q^{\Im} \text{ and } [c,b] \in R^{\Im} \,\};$$
$$(Q^{-1})^{\Im} =_{\text{Def}} \{\, [b,a] : [a,b] \in Q^{\Im} \,\}.$$

Accordingly, an equality $Q{=}R$ turns out to be either true or false in each interpretation $\Im$. One often strives to specify the collection $\mathcal{C}$ of interpretations that are of interest in some application through a set of equalities that must be true in every $\Im$ of $\mathcal{C}$. Requiring, for instance, that $\iota{=}\mathbb{1}$ leads in essence to propositional logic, because it forces $\mathcal{U}$ to be singleton, and hence makes $\emptyset$ and $\mathcal{U}^2$ the only possible values for each map expression P. Figures 3 and 5, to be commented later on, show much more sophisticated examples.

$\mathcal{L}^{+}$ is a variant version of a first-order dyadic predicate language: an atomic formula of $\mathcal{L}^{+}$ has either the form xQy or the form $Q{=}R$, where x, y stand for individual variables (ranging over $\mathcal{U}$) and Q, R stand for map expressions of $\mathcal{L}^{\times}$. Here propositional connectives and existential/universal quantifiers are employed as usual. An ordering $\mathsf{v}_1, \mathsf{v}_2, \ldots$ of individual variables is assumed.

To enrich $\mathcal{L}^{\times}$ and $\mathcal{L}^{+}$ and improve the readability of their map expressions, we can use several pieces of shorthand notation, such as:

$$\begin{array}{lcl@{\qquad}lcl} \overline{P} & \equiv_{\mathrm{Def}} & \not\mathbb{V} \quad \equiv_{\mathrm{Def}} \quad P \triangle \mathbb{1}, & \mathsf{funPart}(P) & \equiv_{\mathrm{Def}} & P \cap \overline{P \circ \overline{\iota}}, \\ P \cup Q & \equiv_{\mathrm{Def}} & (P \triangle Q) \triangle (P \cap Q), & P \setminus Q & \equiv_{\mathrm{Def}} & P \cap (Q \triangle P), \\ P \dagger Q & \equiv_{\mathrm{Def}} & \overline{\overline{P} \circ \overline{Q}}, & \Diamond P & \equiv_{\mathrm{Def}} & \mathbb{1} \circ P \circ \mathbb{1}. \end{array}$$

The interpretation of $\mathcal{L}^{\times}$ and $\mathcal{L}^{+}$ obviously extends to the new constructs; e.g.[1],

$$\mathsf{funPart}(P)^{\Im} =_{\mathrm{Def}} \{[a,b] \in P^{\Im} : [a,c] \notin P^{\Im} \text{ for any } c \neq b\}.$$

Through similar rewriting rules we can extend $\mathcal{L}^{\times}$ in order to emulate familiar constructs such as inclusion, negation, and implication (and then, plainly, all other propositional connectives):

$$P \subseteq Q \equiv_{\mathrm{Def}} P \setminus Q = \emptyset, \qquad \neg P{=}Q \equiv_{\mathrm{Def}} \overline{P \neq Q} \equiv_{\mathrm{Def}} \Diamond(P \triangle Q){=}\mathbb{1},$$
$$P{=}Q \to R{=}S \equiv_{\mathrm{Def}} \overline{\Diamond(P \triangle Q) \circ (R \triangle S)}{=}\emptyset.$$

It is also possible to overcome the limitation of not having constants or function symbols available in $\mathcal{L}^{\times}$, because these can be represented by map symbols P subject to the respective conditions that

$$P^{\Im} = \{[e,e]\} \text{ for some } e \in \mathcal{U},$$

and that *for all $a \in \mathcal{U}$, there is exactly one $b \in \mathcal{U}$ for which $[a,b] \in P^{\Im}$.*
In the map language, these two conditions can be rendered as follows:

$$\begin{array}{lcl} \mathsf{Const}(P) & \equiv_{\mathrm{Def}} & P \circ \mathbb{1} \circ P \subseteq \iota \ \wedge \ P \neq \emptyset, \\ \mathsf{TotFun}(P) & \equiv_{\mathrm{Def}} & P^{-1} \circ P \subseteq \iota \ \wedge \ P \circ \mathbb{1} = \mathbb{1}. \end{array}$$

In spite of all these extensions, $\mathcal{L}^{\times}$ remains limited in means of expression with respect to $\mathcal{L}^{+}$, due to its lack of individual variables and quantifiers. We will discuss how to circumvent the limitations of $\mathcal{L}^{\times}$ in specific but very significant cases in Sections 4 and 5.

2 Raw deductive machinery for a map calculus

We will now slightly adjust the derivability notion for $\mathcal{L}^{\times}$ formalized in [19] to our context. Admittedly, there will be map equalities $P = \mathbb{1}$ not derivable from an empty set of premisses but nonetheless VALID, in the sense that $P^{\Im}{=}\mathcal{U}^2$ is true in every interpretation $\Im$ of $\mathcal{L}^{\times}$. This is due to an intrinsic limitation: there is no way out of this lack of semantic completeness of the derivability notion for $\mathcal{L}^{\times}$.

We start with recording onto the following list of schemes an infinite collection $\Lambda^{\times}$ of valid map equalities, to be regarded as the LOGICAL AXIOMS of $\mathcal{L}^{\times}$:

$$\begin{array}{rcl@{\qquad}l} P \cap Q & = & Q \cap P & \\ \big(P \cap (Q \triangle R)\big) \triangle (P \cap Q) & = & P \cap R & \\ \mathbb{1} \cap P & = & P & \\ (P \star Q) \star R & = & P \star (Q \star R) & \star \textit{ chosen once in } \{\triangle, \cap, \circ\} \\ \iota \circ P & = & P & \\ (P \cup Q) \circ R & = & (Q \circ R) \cup (P \circ R) & \\ P^{-1-1} & = & P & \\ (P \star Q)^{-1} & = & Q^{-1} \star P^{-1} & \star \textit{ chosen once in } \{\cap, \circ\} \\ \Big(P^{-1} \circ \big(R \setminus (P \circ Q)\big)\Big) \cap Q & = & \emptyset & \end{array}$$

Given a collection $\mathtt{E}$ of map equalities, we will denote as $\Theta^{\times}(\mathtt{E})$ the smallest collection of map equalities which both fulfills the inclusion

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

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

[1] $\mathsf{funPart}$ renders the "functional aspect" of a predicate; so $\mathsf{funPart}(\mathrm{P}){=}\mathrm{P}$ means "P is a partial function".

The notation $\mathtt{E} \vdash^{\times} Q{=}R$ is employed to indicate that $Q{=}R$ belongs to $\Theta^{\times}(\mathtt{E})$. We take an analogous (but semantically complete!) definition of $\vdash^{+}$ for granted.[2]

3 First-order theorem-proving used for map logic. Can the service be reciprocated?

Notice that we have been using P, Q, R, and S, as metavariables ranging over map expressions. What would be implied by us changing perspective and regarding P, Q, R, S as individual variables (ruled by understood $\forall$-quantifiers in all logical axiom schemes)? Then each scheme in $\Lambda^{\times}$ would be regarded as a single first-order equality, and we would be dealing with an equational axiomatic first-order theory Θ_{RA} instead of with an alternative formalism. The models of Θ_{RA} are the structures traditionally known as relation algebras, on which [19], p.48, states: *every equation which is shown to be identically satisfied in every relation algebra yields a schema of which all the particular instances (obtained by substituting predicates for variables) are sentences logically provable in* $\mathcal{L}^{\times}$.

This indicates that we can use an automated deduction tool conceived for first-order logic, Otter to be specific (cf. [12, 2]), to experiment with $\mathcal{L}^{\times}$. Although Otter cannot directly produce derivations of $\mathcal{L}^{\times}$, once the equalities that form $\Lambda^{\times}$ are loaded into Otter, whatever chain of inference steps can be drawn from them witnesses the existence of corresponding chains in $\mathcal{L}^{\times}$. One can moreover load into Otter, along with $\Lambda^{\times}$, a set $\mathtt{E}$ of map equalities, and derive theorems of $\Theta^{\times}(\mathtt{E})$.

The very shape of the equalities in $\Lambda^{\times}$ is the result of us having carried out a number of experiments of this nature. We are still trying other formulations of the logical axioms of $\mathcal{L}^{\times}$, that will perhaps drive Otter better; anyway, the one we have adopted above is the outcome of a series of ameliorations carried out on an initial version, until we succeeded in getting an automatic proof of various propositions of [19], pp.49-50, that we had chosen as our benchmarks (cf. [2]).

What we have just said entails that a good first-order theorem prover such as Otter, or simply a theorem prover for pure equational logic, or perhaps a theorem prover based on T-resolution (cf. [15]) and exploiting a decider for map constructs embedded in set theory (cf. chapter 9 of [7]), provides adequate support to symbolic manipulations in the map calculus. Even more importantly, the first-order predicate formalism offers a basis for schematizing meta-theorems of the map calculus, as well as for proving them. In some cases, it enables one to compress into a single quantified sentence an infinite axiom scheme of a theory based on map calculus (examples of this, stressed in boldface in Figures 3, 6, and 5, will be the two induction principles and the restricted set comprehension scheme). Even though we are eagerly following this approach in order to play, and experiment with, specifications written in the map language, we have in mind to invert the approach in the long run. We believe that the map calculus deserves —and sorely lacks, to date— an autonomous and effective instrumentation, to be put to the service of first-order reasoning, and of automated reasoning in general. In sight of this we are developing in SETL [17] a basic layer of Boolean-Peircean simplifications applicable to $\mathcal{L}^{\times}$-expressions and equalities.

[2] Any derivability notion for first-order logic can be exploited for $\mathcal{L}^{+}$, cf., e.g., [19].

4 Translating first-order sentences into map equalities

It is shown in [19] that in $\mathcal{L}^+$ the map constructs $\emptyset, \mathbb{1}, \cap, \triangle, \circ, ^{-1}, =$ can be made to dissolve into connectives and quantifiers: this elimination (whose feasibility was already clear in [21]) leads to a far more conventional first-order language, $\mathcal{L}$, where ι generally takes the typographic form $=$. A remarkable fact about the elimination technique is the following: when one applies it to a sentence α of $\mathcal{L}_3^+$, i.e., an α of $\mathcal{L}^+$ that *involves no more than three distinct individual variables*, the resulting sentence β will also involve three or fewer variables. This is what happens, e.g., when α is an equality $Q{=}R$ of $\mathcal{L}^\times$.

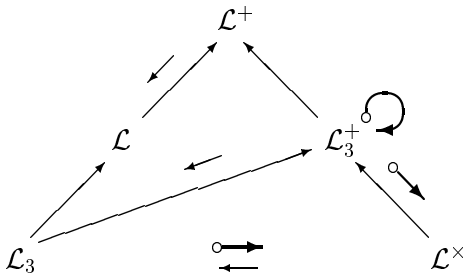

Figure 1: Embeddings and translatability relations between formalisms

To what extent is the reverse translation of $\mathcal{L}^+$ into $\mathcal{L}^\times$ possible? In the sequel, we outline a method to translate sentences from $\mathcal{L}_3^+$ to $\mathcal{L}^\times$ through a quantifier-elimination process that applies to any $\mathcal{L}_3^+$-formula. It is hard to go beyond this, because the said translations reveal that a sentence γ of $\mathcal{L}^+$ can be expressed in $\mathcal{L}^\times$ if and only if it is logically equivalent to a sentence β of $\mathcal{L}_3^+$, while [11] shows that the collection of all such γs is undecidable.

Our purpose can be achieved by means of rewriting rules (see Figure 2) defining a computable total function $H : \{\text{formulas of } \mathcal{L}_3^+\} \longrightarrow \{\text{formulas of } \mathcal{L}_3^+\}$ with the following properties:[3]

- for each $\mathcal{L}_3^+$-formula φ, $H\varphi$ is quantifier-free; moreover, φ and $H\varphi$ have the same free variables;
- when restricted to the sentences of $\mathcal{L}_3^+$, H becomes surjective on the sentences of $\mathcal{L}^\times$;
- for each set Ψ of sentences of $\mathcal{L}_3^+$, $\{H\beta : \beta \in \Psi\}$ and Ψ are logically equivalent in $\mathcal{L}_3^+$.

Translation proceeds in this manner (cf. [14]): first the occurrences of negation are moved inwards, close to atomic sub-formulas, then they are removed using the rules displayed in Figure 2. Then the *swapping* rules are exploited to reduce the number of cases to be taken into account by the *assimilation* and *merging* rules. The latter rules combine together distinct atoms of conjunctions or disjunction. Quantifiers are treated by the remaining rules: they are moved inwards to restrict their scope, and then translated into map constructs. It should be noticed that unrestrained usage of the distributive laws could critically affect the computational complexity of the translation procedure. As a matter of fact, a naïve use of these rules tends to cause an exponential growth of the size of the formula.

A translator of $\mathcal{L}_3^+$ into $\mathcal{L}^\times$ (along with a reverse translator) has been implemented in Prolog and performs well in practice, but a precise assessment of the complexity of the underlying technique is a main issue left open by our work. We are now redesigning the translator in the imperative programming language SETL, to achieve better control of the efficiency through the choice of the appropriate data structures.

[3] This H stands for the same function represented by $\circ\!\!\rightarrow$ in Figure 1.

$\neg xPy \rightsquigarrow x\overline{P}y$	$\neg P{=}Q \rightsquigarrow \Diamond(\, P \mathbin{\triangle} Q\,){=}\mathbb{1}$
ELIMINATION OF THE NEGATION CONNECTIVE.	
$yRx \rightsquigarrow xR^{-1}y$ $yRy \star xSx \rightsquigarrow xSx \star yRy$ $uRv \star P{=}Q \rightsquigarrow P{=}Q \star uRv$ $uRv \star wSw \rightsquigarrow wSw \star uRv$ if $u \not\equiv v$ SWAPPING RULES: $\star \in \{\vee, \wedge\}$. (The first two rules are applied only when x comes before y in the ordering of variables.)	$P{=}Q \star uRv \rightsquigarrow u\overline{\Diamond(\, P \mathbin{\triangle} Q\,)}v \star uRv$ $uSu \star uRv \rightsquigarrow u(\, S \cap \iota\,) \circ \mathbb{1}v \star uRv$ $uSu \star vRu \rightsquigarrow v\mathbb{1} \circ (\, S \cap \iota\,)u \star vRu$ $uSu \vee vRv \rightsquigarrow u(\, S \cap \iota\,) \circ \mathbb{1}v \vee u\mathbb{1} \circ (\, R \cap \iota\,)v$ ASSIMILATION RULES: $\star \in \{\vee, \wedge\}$.
$P{=}Q \wedge R{=}S \rightsquigarrow (\, P \mathbin{\triangle} Q\,) \cup (\, R \mathbin{\triangle} S\,){=}\emptyset$ $P{=}Q \vee R{=}S \rightsquigarrow (\, P \mathbin{\triangle} Q\,) \circ \mathbb{1} \circ (\, R \mathbin{\triangle} S\,){=}\emptyset$ $uRv \left\{ \begin{smallmatrix} \wedge \\ \vee \end{smallmatrix} \right\} uSv \rightsquigarrow uR \left\{ \begin{smallmatrix} \cap \\ \cup \end{smallmatrix} \right\} Sv$ $uSu \wedge vRv \rightsquigarrow u(\, S \cap \iota\,) \circ \mathbb{1} \circ (\, R \cap \iota\,)v$ MERGING RULES.	$\forall x(\, \varphi \wedge \psi\,) \rightsquigarrow \forall x\, \varphi \wedge \forall x\, \psi$ $\exists x(\, \varphi \vee \psi\,) \rightsquigarrow \exists x\, \varphi \vee \exists x\, \psi$ $\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u\, \psi \rightsquigarrow \psi$ $\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u \left(\varphi \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \psi \right) \rightsquigarrow \left(\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u\, \varphi \right) \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \psi$ $\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u \left(\psi \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \varphi \right) \rightsquigarrow \psi \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u\, \varphi$ RULES ON QUANTIFIERS: $u \notin vars(\psi)$.
$(\, \varphi \wedge \psi\,) \vee \chi \rightsquigarrow (\, \varphi \vee \chi\,) \wedge (\, \psi \vee \chi\,)$	$(\, \varphi \vee \psi\,) \wedge \chi \rightsquigarrow (\, \varphi \wedge \chi\,) \vee (\, \psi \wedge \chi\,)$
DISTRIBUTIVE LAWS.	
$\forall u\, uPu \rightsquigarrow \iota \subseteq P$ $\forall u\, uQv \rightsquigarrow v\overline{\emptyset} \dagger Qv$ $\forall u\, vQu \rightsquigarrow vQ \dagger \emptyset v$ $\forall u(\, uQv \vee wRu\,) \rightsquigarrow wR \dagger Qv$ $\forall u(\, uQv \vee uRw\,) \rightsquigarrow wR^{-1} \dagger Qv$ $\forall u(\, vQu \vee wRu\,) \rightsquigarrow vQ \dagger R^{-1}w$ $\forall u(\, vQu \vee uRw\,) \rightsquigarrow vQ \dagger Rw$ ELIMINATION OF $\forall$.	$\exists u\, uPu \rightsquigarrow \Diamond(\, \iota \cap P\,){=}\mathbb{1}$, $\exists u\, uQv \rightsquigarrow v\mathbb{1} \circ Qv$ $\exists u\, vQu \rightsquigarrow vQ \circ \mathbb{1}v$ $\exists u(\, uQv \wedge wRu\,) \rightsquigarrow wR \circ Qv$ $\exists u(\, uQv \wedge uRw\,) \rightsquigarrow wR^{-1} \circ Qv$ $\exists u(\, vQu \wedge wRu\,) \rightsquigarrow vQ \circ R^{-1}w$ $\exists u(\, vQu \wedge uRw\,) \rightsquigarrow vQ \circ Rw$ ELIMINATION OF $\exists$.

Figure 2: Rewriting rules employed to translate $\mathcal{L}_3^+$ into $\mathcal{L}^\times$

In parallel, we are investigating conservative techniques for translating $\mathcal{L}^+$-sentences into $\mathcal{L}^\times$ directly, without the burden of first having to reformulate them (manually or by other means) in $\mathcal{L}_3^+$. Why should one, e.g., recast the monotonicity condition

$$(\, \forall x, y, u, v\,)(\, x < y \wedge xfu \wedge yfv \rightarrow u < v\,)$$

into the unnatural form $(\, \neg\exists x, v\,)\big(\, \exists y(\, x < y \wedge yfv\,) \wedge \exists y(\, xfy \wedge y \not< v\,)\,\big)$ before being able to obtain its translation $(< \circ f) \cap (f \circ \not<) = \emptyset$? A technique to avoid this, described in [6], led us to improved and generalized techniques implemented first in SETL2 and then in Java.

Let us briefly review this algorithm. An existentially quantified conjunction φ of literals of the form xPy, where x and y are variables and P is a map expression (negative literals have been rewritten in the form $x\overline{Q}y$), is given. The algorithm tries to produce an equivalent quantifier-free conjunction as output. The process starts with building an undirected graph $G(\varphi)$ such that: a) $G(\varphi)$ has a node n_{v_i} for each distinct variable v_i occurring in φ; b) for each literal $\mathsf{v}_i P \mathsf{v}_j$ in the conjunction φ, there is an edge $\{n_{\mathsf{v}_i}, n_{\mathsf{v}_j}\}$ labeled by the map expression P or P^{-1} depending on whether $i \leq j$ or $j < i$. The nodes corresponding to variables bound in φ are called *bound* nodes.

An initial phase eliminates every edge of the form $\{n_x, n_x\}$ by creating and suitably introducing a node n_y (where y is a new variable) and an edge $\{n_x, n_y\}$ labeled $P \cap \iota$; moreover, multiple edges between the same nodes are combined (cf. rules in Figure 2).

The elimination of bound nodes (corresponding to eliminations of existential quantifiers in φ) is performed by applying two graph-transformation rules:

- BYPASS rule. Let n_x be a bound node with degree 2 and let P_1, P_2 be the labels of the incident edges $\{n_z, n_x\}$, $\{n_x, n_y\}$. Then the node n_x is removed and a new edge

labeled with the map expression R suitably drawn from $P_1 \circ P_2$, $P_1^{-1} \circ P_2$, $P_2^{-1} \circ P_1$, etc., is created between n_z and n_y. If the edge $\{n_z, n_y\}$ existed already with label Q, then its label becomes $Q \cap R$.

- BIGAMY rule. The rule applies to any bound node n_x having just one incident edge $\{n_z, n_x\}$, such that there exists an edge $\{n_z, n_y\}$ with $y \not\equiv x$. Then the bigamy rule behaves as if there were an edge $\{n_x, n_y\}$ labeled $\mathbb{1}$, in order to bypass the node n_x.

The process ends when no more applications of the previous rules are possible. If the resulting graph has no bound nodes of degree greater that 1, the formula searched for can be directly read off the graph, else we have a failure.

Example. Let us consider a base $\mathcal{B}$ of Prolog clauses subject to the following restrictions: *a)* all predicate letters in $\mathcal{B}$ are dyadic; *b)* $\mathcal{B}$ involves no function letters, but may involve constants. W.l.o.g., we can assume that $\mathcal{B} = \mathcal{B}_E \cup \mathcal{B}_I$, where

- the *extensional* part $\mathcal{B}_E$ of $\mathcal{B}$ is made of facts $e_1 q e_2 \leftarrow$, with e_1 and e_2 constants;
- the *intensional* part $\mathcal{B}_I$ of $\mathcal{B}$ is made of clauses $\quad urv \leftarrow \bigwedge_{i=1}^{n} x_i p_i y_i$, where u, v, x_i, y_i are individual variables, u is distinct from v, $n \geq 0$, each p_i is either a map letter or ι, and r is a map letter not appearing in $\mathcal{B}_E$.

Going to an extreme, we could require that e_1 coincides with e_2 and q occurs in only one fact, for any fact $e_1 q e_2 \leftarrow$ in $\mathcal{B}_E$. On the other hand, it is easy to conceive a generalization where the letters p_i are superseded by arbitrary map expressions P_i in the body of intensional rules.

The body $\bigwedge_{i=1}^{n} x_i p_i y_i$ of each intensional clause $urv \leftarrow \bigwedge_{i=1}^{n} x_i p_i y_i$ can, hence, be submitted to the algorithm described above, treating all variables as existentially bound, save u and v. When the algorithm terminates with success, it supplies an atom of form uQv, uQu, vQv, or $Q{=}R$; in the respective cases one can rewrite the clause as $Q{\subseteq}r$, $(Q \cap \iota) \circ \mathbb{1}{\subseteq}r$, $\mathbb{1} \circ (Q \cap \iota){\subseteq}r$, or $\overline{\diamond(Q \triangle R)}{\subseteq}r$. Moreover, after successfully rewriting every clause defining r in the form $S_j{\subseteq}r$, one can condense all such clauses into a single atom $\bigcup_{j=1}^{m} S_j{=}r$. □

5 From first-order theories to map calculus

It often turns out that a sentence which, taken alone, would not be translatable from first-order logic into map calculus, becomes such when treated in the context of an axiomatic theory. [8] makes the following example: resorting to four variables may seem essential to express the existence of four distinct entities in the domain $\mathcal{U}$ of discourse, but in the theory of strict total orderings $<$, the circumstance can be stated as follows:

$$(\exists x, y)\big(\, x < y \wedge \exists x\big(\, y < x \wedge \exists y\, x < y \,\big)\,\big).$$

Systematic ways of performing the translation irrespective of the number of individual variables can be based on [19], which suggests a general technique based on the notion of CONJUGATED QUASI-PROJECTIONS. These are a pair P, Q of maps such that

$$P{=}\mathsf{funPart}(\,P\,), \quad Q{=}\mathsf{funPart}(\,Q\,), \quad P^{-1} \circ Q{=}\mathbb{1};$$

i.e., partial functions that permit combination of any given pair x, y into a z (intuitively speaking, a pair) for which zPx and zQy.

To illustrate the point, let us make the exercise of specifying the properties of increment (suc), sum, and product over natural numbers. Otherwise stated, we are seeking an $\mathcal{L}^{\times}$-equivalent of Peano arithmetic. A difficulty arises from the presence of binary operations. Normally these are regarded as ternary predicates; as such, however, they do not fit well in $\mathcal{L}^{\times}$.

A key remark is that $\mathcal{U}$ is forced by the axioms to be infinite. Since a one-to-one correspondence β exists between $\mathcal{U}$ and $\mathcal{U}^2$, we can conservatively extend the theory with two unary function symbols, ℓ and r, whose rôle is to represent the 'left' and 'right' projections of numbers. That is, $b = \ell^{\Im}(a)$ and $c = r^{\Im}(a)$ are to fulfill $\beta(a) = [b, c]$ for all $a \in \mathcal{U}$. The fact that ℓ, r are total functions inverting a pairing function is easily stated in $\mathcal{L}^{\times}$, as shown in the first two lines of Figure 3. Most of the remaining axioms of Figure 3 were obtained in [6] from those of a standard first-order axiomatization of the Peano arithmetic (cf., e.g., [13]). This task was achieved with the help of the graph-thinning algorithm outlined above.

The last item in Figure 3 expresses the arithmetic induction principle. This is an example of how one can compress an infinite axiom scheme into a single formula, taking advantage of the first-order metalanguage.

$$\mathsf{TotFun}(\ell) \qquad\qquad \mathsf{TotFun}(r)$$
$$\ell^{-1} \circ r = \mathbb{1} \qquad\qquad (\ell \circ \ell^{-1}) \cap (r \circ r^{-1}) \subseteq \iota$$
$$\mathsf{Const}(0) \qquad\qquad \mathsf{suc} \circ 0 = 0$$
$$\overline{\mathsf{TotFun}(\mathsf{suc})} \qquad\qquad \mathsf{funPart}(\mathsf{suc}^{-1}) = \mathsf{suc}^{-1}$$
$$\overline{(\ell \circ +^{-1}) \cap \iota} = (r \circ \mathsf{suc}^{-1} \circ \mathbb{1}) \cap \iota \qquad\qquad * \triangle (\ell \cup r) \subseteq \mathbb{1} \circ \mathsf{suc}$$
$$(r \circ \mathsf{suc}^{-1} \circ \mathbb{1}) \cap + = ((r \circ \mathsf{suc}^{-1} \circ r^{-1}) \cap (\ell \circ \ell^{-1})) \circ + \circ \mathsf{suc}$$
$$(r \circ \mathsf{suc}^{-1} \circ \mathbb{1}) \cap * = ((((r \circ \mathsf{suc}^{-1} \circ r^{-1}) \cap (\ell \circ \ell^{-1})) \circ * \circ \ell^{-1}) \cap (\ell \circ r^{-1})) \circ +$$
$$\Big(\emptyset \dagger \big(\big(\boldsymbol{P} \cup (\mathsf{suc}^{-1} \circ \mathbb{1})\big) \cap \big(\overline{\boldsymbol{P}} \cup (\mathsf{suc} \circ \boldsymbol{P})\big)\big) \dagger \emptyset\Big) \circ \overline{\boldsymbol{P}} = \emptyset$$

Figure 3: Map-formulation of Peano arithmetic (in $\mathcal{L}^{\times}$)

As a further example, we show how to formulate a set theory in the map formalism. Figure 4 describes an axiomatization of a weak set theory including: the axioms of extension **(E)**, regularity **(R)**, and separation **(S)**. The last item —**(W)**, the *with* axiom— expresses the property that it is always possible to compose the set $x \cup \{y\}$ out of given x, y (i.e., we can adjoin any element y to any set x and obtain a set).

(E)	$\forall x \forall y \big(\forall v (v \in x \leftrightarrow v \in y) \rightarrow x = y \big)$
(R)	$\forall x \exists y \forall v \big(v \in x \rightarrow (y \in x \wedge v \notin y) \big)$
(S)	$\forall \boldsymbol{x} \exists \boldsymbol{y} \forall \boldsymbol{v} \big(\boldsymbol{v} \in \boldsymbol{y} \leftrightarrow (\boldsymbol{v} \in \boldsymbol{x} \wedge \boldsymbol{v} \boldsymbol{P} \boldsymbol{x}) \big)$
(W)	$\forall x \forall y \exists w \forall v \big(v \in w \leftrightarrow (v \in x \vee v = y) \big)$

Figure 4: First-order formulation of a weak set theory (in $\mathcal{L}^{+}$)

The axioms **(E)**, **(R)**, and **(S)** are expressed by $\mathcal{L}_3$-formulas, hence translating them into the map calculus does not present difficulties. The map-formulations of these axioms are displayed in Figure 5, where we have adopted the following definition: $\ni \equiv_{\mathrm{Def}} \in^{-1}$.

Notice that in the translation of **(S)** (analogously to what happened with the arithmetic induction principle in the previous example) we take advantage of the first-order metalanguage. In Figures 4 and 5, P ranges over the set of all map expressions.

(E)	$\overline{\iota} \subseteq (\notin \circ \in) \cup (\ni \circ \notin)$
(R)	$\in \subseteq \mathbb{1} \circ (\in \cap \overline{\ni \circ \in})$
(S)	$\mathbb{1} \circ \big((\notin \dagger \in) \cap (\notin \dagger \boldsymbol{P}) \cap (\ni \dagger \overline{\in \cap \boldsymbol{P}})\big) = \mathbb{1}$
($\mathbf{W}_1$)	$\mathbb{1} = \pi_0^{-1} \circ \pi_1$
($\mathbf{W}_2$)	$\iota \subseteq \Big(\big(((\pi_0 \circ \ni) \cup \pi_1) \dagger \overline{\in}\big) \cap \big(\overline{((\pi_0 \circ \ni) \cup \pi_1)} \dagger \in\big)\Big) \circ \mathbb{1}$

Figure 5: Map-formulation of a weak set theory (in $\mathcal{L}^{\times}$)

The first-order formulation of **(W)** uses four distinct variables. This it is not an $\mathcal{L}_3$-formula and its translation cannot be immediate;[4] we have to introduce a pair of conjugated quasi-projections (π_0 and π_1 below) in order to properly face the problem:

$$\ni\ni \equiv_{\text{Def}} \ni \circ \ni, \qquad \ni_s\ni \equiv_{\text{Def}} \ni \circ \mathsf{funPart}(\ \ni\ ),$$
$$\pi_0 \equiv_{\text{Def}} \mathsf{funPart}(\ \ni_s\ni\ ), \qquad \pi_1 \equiv_{\text{Def}} \ni\ni \cap \overline{(\ \ni\ni \cap \overline{\pi_0}\ ) \circ \overline{\iota}}.$$

The axioms $(\mathbf{W}_1)$ and $(\mathbf{W}_2)$ in Figure 5 express the *with* axiom in the map-calculus. Notice that it is $(\mathbf{W}_1)$ that forces the two maps π_0 and π_1 to be conjugated quasi-projections.

Our third and last example is borrowed from [2], with some simplifications. In Figure 6 we are characterizing the domain of flat lists composed of individuals drawn from an infinite collection of atoms (in [2] lists are not flat, in the sense that a list can be a member of another list). Needless to say, we are assuming that the sorts of 'atoms' and 'lists' are disjoint.

a.	$\mathsf{Const}(\mathsf{nl})$	$\mathsf{at} \cap \mathsf{nl} = \emptyset$
b.	$\mathsf{fun}(\mathsf{hd})$	$\mathsf{fun}(\mathsf{tl})$
c.	$\mathsf{hd} \circ \mathbb{1} = \mathsf{tl} \circ \mathbb{1}$	$\mathsf{at} \triangle (\mathsf{nl} \circ \mathbb{1}) = \overline{\mathsf{tl} \circ \mathbb{1}}$
d.	$\overline{\mathsf{at}} \cap \mathsf{at}^{-1} \subseteq \mathsf{tl}^{-1} \circ \mathsf{hd}$	$(\mathsf{hd} \circ \mathsf{hd}^{-1}) \cap (\mathsf{tl} \circ \mathsf{tl}^{-1}) \subseteq \iota$
e.	$\mathbb{1} \circ (\mathsf{at} \setminus \mathsf{ocl}) = \mathbb{1}$	$\mathsf{ocl} = (\iota \circ \mathsf{hd}^{-1}) \cup (\mathsf{ocl} \circ \mathsf{tl}^{-1})$
f.	$(\mathsf{tl} \cup \mathsf{ocl}) \cap \iota = \emptyset$	$\diamond\Big(\big(\mathsf{at} \cup (\mathsf{nl} \circ \mathbb{1}) \cup \big((\mathsf{hd} \circ \boldsymbol{P}) \cap (\mathsf{tl} \circ \boldsymbol{P})\big)\big) \setminus \boldsymbol{P}\Big) \dagger \boldsymbol{P} = \mathbb{1}$

Figure 6: Map-formulation of a theory of flat lists (in $\mathcal{L}^{\times}$)

By a. in Figure 6, a distinguished individual (to be intended as the empty list 'nil') differs from any atom. Axioms b. and c. impose hd and tl to be partial functions defined on the same domain (to be regarded as the lists), which includes everything but atoms and nil. Notice that, by c.2, the 'is atom' predicate does not truly depend on its second argument. Axiom d.1 states that given any pair a, b such that a is an atom and b is not, there exists a list whose head and tail are a and b, respectively; moreover, by d.2, lists that differ from one another cannot have the same head and the same tail. Axioms e.1 and e.2 express the concept of occurrence with the usual meaning. The acyclicity requirement is imposed by axiom f.1: nothing can either occur within itself or be its own tail. Finally, axiom f.2 expresses an induction principle for lists: P, as for the previous examples, can be any map expression.

6 Quest for syllogistics dealing with set combinators

A line of research initiated in the late seventies led to the discovery of a number of decidable fragments of set theories. Among the decision algorithms, known as SYLLOGISTICS, some deal with map constructs (cf. Chapter 9 of [7]); hence we expect that they can offer useful support to map reasoning in the framework of $\mathcal{L}^{\times}$. However, since they were originally conceived in the framework of Set Theory, they will need some adaptation to be exploitable in the new context.

Conversely, as outlined in Sec.5, one can express set theories in the map calculus. Accordingly, syllogistics that are ordinarily referred to first-order set theories can also be viewed as solvers for somewhat specific map reasoning problems. From this standpoint one may get a new insight on the decision problem for fragments of set theories, ultimately leading to enhancements of the known syllogistics in unprecedented directions.

[4] As a matter of fact, [11] shows that **(W)**, taken alone, cannot be expressed in $\mathcal{L}_3$.

To hint at the point with a simple case-study, let us briefly consider here *multi-level syllogistic with singleton*, or *MLSS*. The problem at hand is the one of testing for satisfiability an unquantified formula that can only involve set variables, the null-set constant $\emptyset$, and the remaining constructs of Figure 7, which are

- the monadic singleton operator $\{\cdot\}$;
- the dyadic operators $\cap, \setminus, \cup$—provisionally designating, here, operations on sets;
- membership and set-equality relators; and
- propositional connectives.

Primitive			*Derived*	
$\cdot \cup \cdot$	$\cdot \setminus \cdot$	$\{\cdot\}$	$\cdot \cap \cdot$	$\emptyset$
	$\cdot = \cdot$		$\cdot \in \cdot$	$\cdot \subseteq \cdot$
$\neg \cdot$	$\cdot \wedge \cdot$	$\cdot \vee \cdot$	$\cdot \rightarrow \cdot$	$\cdot \leftrightarrow \cdot$

Figure 7: Constructs of the multilevel syllogistic language

This decision problem, which was first shown to be solvable in [9], is easily reduced to the satisfiability problem for conjunctions of literals of the following forms (where x, y, u stand for individual variables):

$u \in y, \qquad u = y, \qquad u \neq y, \qquad \emptyset = x \cap y, \qquad u \subseteq y, \qquad u = \{y\}, \qquad u = x \cup y.$

To reformulate any such conjunction in map-theoretic terms, we can translate its *MLSS*-literals into corresponding map formulas, one by one, as indicated in Figure 8.

$$u = y \rightsquigarrow u \iota y \qquad u \neq y \rightsquigarrow u \overline{\iota} y \qquad u \in y \rightsquigarrow u \in y$$
$$\emptyset = x \cap y \rightsquigarrow x \overline{\ni \circ \in} y \qquad u \subseteq y \rightsquigarrow u \not\ni \dagger \in y \qquad u = \{y\} \rightsquigarrow y \in u \wedge y \, \iota \dagger \notin u$$
$$u = x \cup y \rightsquigarrow p\pi_0 x \wedge p\pi_1 y \wedge x \subseteq u \wedge y \subseteq u \wedge u \not\ni \dagger(\in \circ \in\in)p$$

Figure 8: Rules for translating *MLSS* into the map calculus

The key idea of this translation is the introduction of two conjugated quasi-projections π_0 and π_1. Having in mind the classical pair notion due to Kuratowski, namely

$$[x, y] \equiv_{\text{Def}} \big\{\{x\}, \{y, x\}\big\},$$

these can be defined as follows, to the effect that

$$p\pi_0 x \quad \leftrightarrow \quad \{x\} \in p \wedge \forall z \in p\big(z \neq \{x\} \wedge z \neq \emptyset \rightarrow (\exists u, v \in z \,) u \neq v \big),$$
$$p\pi_1 y \quad \leftrightarrow \quad y \in\in p \wedge (\forall z \in\in p \,)\big(z \neq y \rightarrow p\pi_0 z \big).$$

We have now seen how to rewrite an *MLSS*-formula as a conjunction of atoms of the form xSy, where S is drawn from a finite collection of combinators: ι, $\overline{\iota}$, $\in$, $\overline{\ni \circ \in}$, $\not\ni$ $\dagger \in$, etc. Now the question arises naturally: To what extent, and by what criteria, could we broaden the collection of admitted set combinators, without either losing decidability or leaving the complexity class of *MLSS* (whose decision problem is *NP*-complete)?

7 Safe map reasoning

There is an evident kinship between $\mathcal{L}^\times$ and the relational algebra language used in the database field. There are obvious differences, too; to mention one, the map letters of $\mathcal{L}^\times$ represent binary relations, whereas database languages have to manage relations in any number of arguments. Moreover, in $\mathcal{L}^\times$ complementation is made w.r.t. a fixed universe of discourse, which may be infinite. In the latter regard $\mathcal{L}^\times$ exceeds the needs of database management, as it may bring infinite relations into play. Thus, in order that the kinship between $\mathcal{L}^\times$ and relational algebra can really make $\mathcal{L}^\times$ paradigmatic —on the small scale— of symbolic languages of great practical value, one must occasionally

restrain the forms of notation and reasoning allowed in $\mathcal{L}^\times$, so as to ban infinite maps from consideration. Establishing tight correspondences between formalisms is generally enlightening, and we see, e.g., an analogy between translating first-order theories into $\mathcal{L}^\times$ (cf. Sec.5) and translating Datalog into relational algebra (cf. [20]), to the extent to which these translations are possible.

SAFE map reasoning ought to imply no engagement about the availability of infinite maps (even assuming an infinite domain $\mathcal{U}$ of discourse). A drastic choice, to achieve this, would be to do entirely without $\mathbb{1}$ and ι. However, this would expunge —together with undesired operations such as complementation— useful secondary operations such as those of forming $P \cap \iota$ and $P \circ \mathbb{1} \circ Q$ out of safe map expressions P, Q. These or similar operations could, as a remedy, be taken as primitives; likewise, inequalities $P \neq Q$ should somehow be re-admitted into play.

All of this would, of course, impose a redesign of the deductive apparatus proposed for $\mathcal{L}^\times$ (cf. Sec.2), that reflected the intended meaning of the additional primitive constructs while ensuring the safeness of each step in a derivation. We expect that this can be done in a way that guarantees that when P, Q are safe and $P{=}Q$ (respectively, $P{\neq}Q$) belongs to $\Theta^\times(\mathtt{C})$, where all constraints in $\mathtt{C}$ are safe, then $P{=}Q$ (resp. $P{\neq}Q$) can be safely derived from $\mathtt{C}$. As initial moves in the direction of safe map reasoning, we have adopted $\triangle$ as a primitive construct of $\mathcal{L}^\times$ (whereas [19] adopts complementation); we have accordingly chosen a set $\Lambda^\times$ of logical axioms for $\mathcal{L}^\times$ where the rôle of $\mathbb{1}$ is very marginal; moreover, we have preferred safe characterizations of $\cup$ and $\setminus$ in Sec.1 to simpler ones such as would have been: $P \cup Q \equiv_{\text{Def}} \overline{\overline{P} \cap \overline{Q}}$ and $P \setminus Q \equiv_{\text{Def}} P \cap \overline{Q}$.

Example. Let us restrain our consideration to the collection $\mathcal{F}$ consisting of all map expressions that do not involve $\triangle$, and to those interpretations that assign finite values to the map letters $\mathtt{p}_i$. We subdivide $\mathcal{F}$ into six equivalence classes, with representative elements $\emptyset, s, \iota, s \circ \mathbb{1}, \mathbb{1} \circ s$, and $\mathbb{1}$ (here s is a short for $\mathtt{p}_1$):

- $\emptyset$ represents the class consisting of all expressions whose value is *necessarily* $\emptyset$ (i.e., in each interpretation their value must be the empty set);
- s represents the class of those expressions whose value is necessarily finite but not necessarily $\emptyset$ (among them, all map letters $\mathtt{p}_i$);
- ι represents the class of those expressions whose value is necessarily a subset of $\iota^\Im$ whose complement in $\iota^\Im$ is possibly finite;
- $s \circ \mathbb{1}$ represents the class of those expressions whose value's domain and image, unless empty, are: a necessarily finite subset of $\mathcal{U}$, and a possibly *cofinite* subset of $\mathcal{U}$ (i.e., one having a finite complement), respectively;
- $\mathbb{1} \circ s$ represents the class of those expressions whose value's domain and image, unless empty, are: a possibly cofinite subset of $\mathcal{U}$, and a necessarily finite subset of $\mathcal{U}$, respectively;
- $\mathbb{1}$ represents all expressions whose value has a possibly finite complement in $\mathcal{U}^2$.

It will turn out that, in $\mathcal{F}$, the equivalence class of $\emptyset$ consists of all expressions where $\emptyset$ occurs at least once. Moreover, the equivalence class of $\mathbb{1}$ (respectively, of ι) is composed by expressions whose value is necessarily $\mathcal{U}^2$ (resp., $\{[a,a] : a \; in \; \mathcal{U}\}$).

To assess the TYPE of each expression P, i.e. the equivalence class to which P belongs, we can exploit a small algebra of types. One begins with assigning the type s to all map letters in the given P, and then propagates type information through the whole of P by the rules of the type algebra. Figure 9 shows a tabular form of the rules for $\cap$ and $\circ$. It is easily seen that P is safe if and only if its type turns out to be either $\emptyset$ or s. $\square$

$\cap$	$\mathbb{1} \circ s$	s	$s \circ \mathbb{1}$	$\mathbb{1}$	ι	$\emptyset$
$\mathbb{1} \circ s$	$\mathbb{1} \circ s$	s	s	$\mathbb{1} \circ s$	s	$\emptyset$
s	s	s	s	s	s	$\emptyset$
$s \circ \mathbb{1}$	s	s	$s \circ \mathbb{1}$	$s \circ \mathbb{1}$	s	$\emptyset$
$\mathbb{1}$	$\mathbb{1} \circ s$	s	$s \circ \mathbb{1}$	$\mathbb{1}$	ι	$\emptyset$
ι	s	s	s	ι	ι	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

$\circ$	$\mathbb{1} \circ s$	s	$s \circ \mathbb{1}$	$\mathbb{1}$	ι	$\emptyset$
$\mathbb{1} \circ s$	$\mathbb{1} \circ s$	$\mathbb{1} \circ s$	$\mathbb{1}$	$\mathbb{1}$	$\mathbb{1} \circ s$	$\emptyset$
s	s	s	$s \circ \mathbb{1}$	$s \circ \mathbb{1}$	s	$\emptyset$
$s \circ \mathbb{1}$	s	s	$s \circ \mathbb{1}$	$s \circ \mathbb{1}$	$s \circ \mathbb{1}$	$\emptyset$
$\mathbb{1}$	$\mathbb{1} \circ s$	$\mathbb{1} \circ s$	$\mathbb{1}$	$\mathbb{1}$	$\mathbb{1}$	$\emptyset$
ι	$\mathbb{1} \circ s$	s	$s \circ \mathbb{1}$	$\mathbb{1}$	ι	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Figure 9: Tables for safeness detection

Conclusions

First-order predicate logic undoubtedly deserves the primacy, with respect to the map calculus, of user-friendliness and expressive manageability. One cannot, however, discard beforehand the idea that the map calculus may perform better in the rôle of basic machine-reasoning layer. This expectation deserves, in our opinion, a serious and twofold experimentation effort. On the one hand, it calls for

- development of effective theorem-proving techniques directly rooted on the map calculus (cf. Sec.3); on the other hand, it requires
- sophisticated techniques for translating sentences —or even entire sets of axioms— from first-order logic into the map language.

In essence the latter techniques (which this paper has striven to bring to surface—see, in particular, Sections 4 and 5) are to translate formal specifications, phrased in first-order logic as is nowadays more common, into a specialized area of algebra.

It is well-known that the language of map calculus has the same strength (and weakness) as a first-order language involving three individual variables altogether. Apart from this, it is a stimulating fact of mathematics that one cannot decide the precise extent to which the envisaged translation of logic into algebra is possible (cf. [11]); as a consequence, this is an issue to be tackled pragmatically and conservatively. Powerful ideas have emerged from a protracted stream of research initiated in the forties and finally blossomed in the Tarski-Givant monograph [19], which indicates (as we have demonstrated in Sec.5) how any theory regarding either sets or arithmetics can be phrased in map-theoretic terms.

Acknowledgments

We are indebted to IASI (*Istituto di Analisi dei Sistemi ed Informatica*, CNR) for promoting exchanges of ideas between authors (cf. [2]), to DIS (Dip. di Informatica e Sistemistica, Univ. "La Sapienza" of Rome) for funding [14], to Vuokko-Helena Caseiro for careful revision of the manuscript, to Paul Broome (cf. [5]) for first bringing [19] to our attention during the ICLP conference held in Budapest in 1993, to Emidio Silvestri for paving the way to this research with a Prolog implementation of a logic-to-mapalgebra translator, to Domenico Cantone and Alessandra Cavarra for very pleasant summer conversations which led to discoveries on how to improve the translator, and to Fabiola Aureli for contributing to the design of a basic map reasoning layer in SETL.

References

[1] I. H. Anellis and N. R. Houser. Nineteenth century roots of algebraic logic and universal

algebra. In H. Andréka, J. D. Monk, and I. Németi, eds. *Algebraic Logic*, vol. 54 of *Colloquia Mathematica societatis János Bolyai*, pages 1–36. North-Holland Publishing Co., 1991.

[2] E. Aureli, E. G. Omodeo, and M. Temperini. Map calculus: Initial application scenarios and experiments based on Otter. Technical Report R499, IASI-CNR, 1998.

[3] I. M. Bocheński. *A history of formal logic.* Chelsea, Thomas, I. editor and translator, 1970.

[4] G. Boole. *An investigation of the laws of thought* on which are founded the mathematical theories of logic and probabilities. Dover books in Advanced Mathematics, 1854.

[5] P. Broome and J. Lipton. Constructive relational programming: A declarative approach to program correctness and high level optimization. Trans. of the 9th Army Conference on Applied Mathematics and Computing. Tech. Rep. ARO Report 92-1, DARPA, USA, 1992.

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

[7] D. Cantone, A. Ferro, and E. G. Omodeo. *Computable Set Theory. Vol. 1.* Oxford University Press, Int. Series of Monographs on Computer Science, 1989.

[8] E. Grädel, Ph. G. Kolaitis, and M. Y. Vardi. On the decision problem for two-variable first-order logic. Bulletin of Symbolic Logic, 3(1), 1997.

[9] A. Ferro, E. G. Omodeo, and J. T. Schwartz, Decision Procedures for Elementary Sublanguages of Set Theory I. Multilevel Syllogistic and Some Extensions. Comm. on Pure and Appl. Mathematics, 33, pages 599–608, 1980.

[10] J. van Heijenoort. *From Frege to Gödel - A source book in mathematical logic, 1879–1931.* Source books in the history of the sciences. Harvard University Press, 3^{rd} printing edition, 1977.

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

[12] W. W. McCune. Otter 3.0 Reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, 1994. (Revision A, august 1995).

[13] E. Mendelson. *Introduction to Mathematical Logic.* Van Nostrand, New York, 1979.

[14] E. G. Omodeo. Specifiche formali di proprietà di relazioni: esempi. Technical Report 36-97, Dip. Informatica e Sistemistica, Università *La Sapienza* di Roma, 1997.

[15] A. Policriti and J. T. Schwartz. *T* Theorem Proving I. *Journal of Symbolic Computation*, 20:315–342, 1995.

[16] E. Schröder. *Vorlesungen über die Algebra der Logik (exakte Logik).* B. Teubner, Leipzig, 1891-1895. [Reprinted by Chelsea Publishing Co., New York, 1966.]

[17] J. T. Schwartz, R. K. B. Dewar, E. Dubinsky, and E. Schonberg. *Programming with Sets: An introduction to SETL.* Texts and Monographs in Computer Science. Springer-Verlag, 1986.

[18] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.

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

[20] J. D. Ullman. *Database and Knowledge-base Systems, vol.2*, volume 50 of *Principles of Computer Science.* Computer Science Press, Stanford University, 1989.

[21] A. N. Whitenead and B. Russell. *Principia Mathematica.* Cambridge University Press, 1910. Reprinted 1980.