Goals and benchmarks for automated map reasoning *

Andrea Formisano†
Eugenio G. Omodeo‡
Marco Temperini§

June 1998

Abstract

Tarski-Givant's map calculus is briefly reviewed, and a plan of research is outlined aimed at investigating applications of this ground equational formalism in the theorem-proving field. The main goal is to create synergy between first-order predicate calculus and the map calculus. Techniques for translating isolated sentences, as well as entire theories, from first-order logic into map calculus are designed, or in some cases simply brought nearer through the exercise of specifying properties of a few familiar structures (natural numbers, nested lists, finite sets, lattices). It is also highlighted to what extent a state-of-the-art theorem-prover for first-order logic, namely Otter, can be exploited not only to emulate, but also to reason about, map calculus. Issues regarding 'safe' forms of map reasoning are singled out, in sight of possible generalizations to the database area.

1 Introduction

Two systems of symbolic logic were proposed around 1880. One of the two, due to Frege, developed later into the Whitehead-Russell's *Principia Mathematica* [WR10]; in somewhat revised form, it is the system that prevailed in common usage. The other formalism, due to C. S. Peirce, was much closer to the original spirit of Boole's *Laws of thought*; it temporarily reached stability in Schröder's monumental work [Sch95] on the *Algebra der Logik*, but then got engulfed in a "rather capricious line of historical development" [Tar41] which almost led it into oblivion, at least among the practitioners of logic.

The influence of Peirce-Schröder's algebraic form of logic —the MAP CALCULUS, as we will call it— lasted long only in the field of universal algebra. Its footprint can nevertheless be recognized in today's relational database languages, as well as in a variety of representation languages often called taxonomic languages or description logics (cf., e.g., [Bor96, Cal96]).

Inside and outside the specialists' community (see, e.g., [BL94, GKV97, GOS97]), there is nowadays a new burst of interest in the map calculus, stirred up by the publication of [TG87]. 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.

First-order predicate logic undoubtedly deserves the primacy, with respect to the map calculus, of user-friendliness and expressive manageability. However, this is no evidence that the map calculus may not perform better in the rôle of basic machine-reasoning layer or that one cannot profitably intermix the two ways of reasoning. These hypotheses deserve, in our opinion, a serious and twofold experimentation effort. On the one hand, they call for

*Work partially supported by the 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".

†Dept. of Pure and Applied Mathematics, University of L'Aquila, I. formisan@dimi.uniud.it
‡Dept. of Pure and Applied Mathematics, University of L'Aquila, I. omodeo@univaq.it
§Dept. of Computer and Systems Science, University "La Sapienza" of Rome, I. marte@dis.uniroma1.it

- development of effective theorem-proving techniques directly rooted on the map calculus; on the other hand, they require
- sophisticated techniques for translating sentences —or even entire sets of axioms— from first-order logic into the map language.

In essence the latter techniques (of which an example can be found in [CCO97]) are to translate formal specifications, phrased in first-order logic as is nowadays more common, into a specialized area of algebra.

It is a stimulating fact of mathematics that one cannot decide the precise extent to which this translation of logic into algebra is possible (cf. [Kwa81]); 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 above-mentioned Tarski-Givant monograph [TG87], which indicates how any theory regarding either sets or arithmetics can be phrased in map-theoretic terms.

The demonstration that map calculus had no inner weaknesses preventing it from becoming the frame for an omni-comprehensive deductive system such as set theory, rehabilitates it from its disrepute (whose historical causes are skillfully investigated in [AH91]) and reopens the opportunity, dismissed for decades, to put together complementary virtues of the map calculus and of first-order predicate logic.

This paper gives a contribution to this. A few scenarios of use of Tarski-Givant's map calculus are developed. Properties of familiar structures (natural numbers, nested lists, hereditarily finite sets, lattices) endowed with operations and relations are formally specified. Exercise of this nature, based mainly 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—a fully equational formalism.

The specification of lattice theory that will be proposed, far from being a straightforward adaptation to the map formalism of any of its preexisting first-order formulations, will reveal particularly challenging. As is well-known, in fact, the map language has the same strength —and limited expressive power— as a first-order language involving three individual variables altogether.

The paper stresses 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. The authors have begun a series of experiments based on the Otter theorem prover (cf. [McC95]). To date, as a matter of fact, our 'deductive apparatus' for the map calculus has simply consisted in a first-order axiomatization of relation algebras fed into Otter. The initial benchmarks were aimed at finding a convenient formulation of these axioms, and only now our experimentation activity can shift to theory level.

We have set to ourselves a series of goals that will contribute, hopefully, to the development of a platform for the automation of map reasoning. Such a platform should encompass techniques for translating first-order logic into map logic, tools for simplifying map expressions, decision algorithms for fragments of the map calculus or of important theories based on it, etc. We envisage, among others, the design of a 'safe' variant of the map calculus, paradigmatic of the way one views relations in the database field.

All of these themes will be touched upon in what follows.

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

$\mathcal{L}^{\times}$ is a ground equational language where one can state properties of dyadic 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., $\in$, η, void, $\mathcal{P}ow$);
- dyadic constructs $\cap$, $\triangle$, $\circ$ of map INTERSECTION, map SYMMETRIC DIFFERENCE, and map COMPOSITION;

- the monadic 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 dyadic 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:

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

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. Figure 1 and several other figures to be commented later on, show more sophisticated examples.

$\mathcal{L}^+$ is a variant version of a first-order dyadic predicate language: an ATOMIC FORMULA (briefly, an *atom*) 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 all 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:

$\overline{P} \equiv_{\text{Def}} \not{P} \equiv_{\text{Def}} P\triangle\mathbb{1}$	$P\dagger Q \equiv_{\text{Def}} \overline{\overline{P}\circ\overline{Q}}$
$P\cup Q \equiv_{\text{Def}} (P\triangle Q)\triangle(P\cap Q)$	$\diamond P \equiv_{\text{Def}} \mathbb{1}\circ P\circ\mathbb{1}$
$P\setminus Q \equiv_{\text{Def}} P\cap(Q\triangle P)$	$\mathsf{funPart}(P) \equiv_{\text{Def}} P\cap\overline{P\circ\overline{\iota}}$

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

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

Through similar rewriting rules we can extend $\mathcal{L}^\times$ in order to emulate constructs such as inclusion, negation, and implication:

	$P{\subseteq}Q \equiv_{\text{Def}} P\setminus Q = \emptyset$	
$\neg P{=}Q \equiv_{\text{Def}}$	$P{\neq}Q \equiv_{\text{Def}} \diamond(P\triangle Q){=}\mathbb{1}$	
	$P{=}Q\to R{=}S \equiv_{\text{Def}} \overline{\diamond(P\triangle Q)}\circ(R\triangle S){=}\emptyset$	

(We will see in Figure 4 similar ways to emulate the conjunction and disjunction connectives.)

Example. By saying that λ is a BISIMULATION over ϱ, one usually means that λ is *symmetric*, and *stable* over ϱ. These two properties can be stated in $\mathcal{L}^+$ as

$$\forall y\,\forall x\,(\,y\lambda x\to x\lambda y\,),$$
$$\forall u\,\forall v\,\forall w\big(\,u\lambda v\wedge v\varrho w\to\exists z\,(\,z\lambda w\wedge u\varrho z\,)\,\big),$$

and can be stated in $\mathcal{L}^\times$as

$$\lambda^{-1}\subseteq\lambda, \qquad \lambda\circ\varrho\subseteq\varrho\circ\lambda.$$

Sometimes (cf. [Acz88]), without requiring λ to be symmetric, one strengthens the stability condition as follows:

$$\forall u_0\,\forall u_1\,\Big(\,u_0\lambda u_1\to\textstyle\bigwedge_{b=0}^{1}\forall w_b\big(\,u_b\varrho w_b\to\exists w_{1-b}\big(\,u_{1-b}\varrho w_{1-b}\wedge w_0\lambda w_1\,\big)\big)\Big).$$

The latter can be formulated in $\mathcal{L}^\times$ simply as

$$\lambda\circ\varrho\subseteq\varrho\circ\lambda, \qquad \lambda^{-1}\circ\varrho\subseteq\varrho\circ\lambda^{-1}.$$

□

[1] $\mathsf{funPart}(\,\cdot\,)$ renders the "functional aspect" of a predicate; so $\mathsf{funPart}(P){=}P$ will mean "P is a partial function", very much like $\mathsf{Fun}(P)$ to be seen below.

2.1 Making constants and monadic symbols available in $\mathcal{L}^{\times}$

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

for all a in $\mathcal{U}$, there is exactly one b in $\mathcal{U}$ for which $[a,b] \in P^{\mathfrak{S}}$.

In the map language, this condition can be rendered as follows:

$$\mathsf{TotFun}(P) \equiv_{\mathrm{Def}} \mathsf{Fun}(P) \wedge \mathsf{Total}(P),$$

where

$$\mathsf{Fun}(P) \equiv_{\mathrm{Def}} P^{-1} \circ P \subseteq \iota, \qquad \mathsf{Total}(P) \equiv_{\mathrm{Def}} P \circ \mathbb{1} = \mathbb{1}.$$

One can, moreover, represent a monadic predicate (i.e., a subset of $\mathcal{U}$) by a dyadic P subject to any of the following three conditions:

Right-absoluteness: $\mathsf{RAbs}(P) \equiv_{\mathrm{Def}} P = P \circ \mathbb{1}$;
Left-absoluteness: $\mathsf{LAbs}(P) \equiv_{\mathrm{Def}} P = \mathbb{1} \circ P$;
Diagonality: $\mathsf{Diag}(P) \equiv_{\mathrm{Def}} P \subseteq \iota$.

The first two of these mean, respectively, that

for all $[a,b] \in P^{\mathfrak{S}}$ and any c in $\mathcal{U}$, $[a,c] \in P^{\mathfrak{S}}$;

and that

for all $[a,b] \in P^{\mathfrak{S}}$ and any c in $\mathcal{U}$, $[c,b] \in P^{\mathfrak{S}}$.

Likewise, we can represent an individual constant by a predicate symbol or expression P meeting one of the following:

$$\begin{aligned} \mathsf{RConst}(P) &\equiv_{\mathrm{Def}} P \neq \emptyset \wedge P^{-1} \circ \mathbb{1} \circ P \subseteq \iota \wedge \mathsf{LAbs}(P) \\ \mathsf{LConst}(P) &\equiv_{\mathrm{Def}} P \neq \emptyset \wedge P \circ \mathbb{1} \circ P^{-1} \subseteq \iota \wedge \mathsf{RAbs}(P) \\ \mathsf{Const}(P) &\equiv_{\mathrm{Def}} P \neq \emptyset \wedge P \circ \mathbb{1} \circ P \subseteq \iota \end{aligned}$$

These mean that for a fixed c in $\mathcal{U}$, either $P^{\mathfrak{S}} = \{[c,b] : b \text{ in } \mathcal{U}\}$, or $P^{\mathfrak{S}} = \{[a,c] : a \text{ in } \mathcal{U}\}$, or $P^{\mathfrak{S}} = \{[c,c]\}$ holds, respectively.

Example. The proposition

"*if mice are rodents, then every paw of a mouse is paw of a rodent*"

(which is valid by its form alone) can be rendered in $\mathcal{L}^{+}$ as

$$\forall x \big(\mathsf{mouse}(x) \rightarrow \mathsf{rodent}(x) \big) \longrightarrow \forall y \Big(\exists u \big(\mathsf{mouse}(u) \wedge \mathsf{hasPaw}(u,y) \big) \rightarrow \exists v \big(\mathsf{rodent}(v) \wedge \mathsf{hasPaw}(v,y) \big) \Big)$$

and can be rendered in $\mathcal{L}^{\times}$ as

$$\mathsf{Diag}(\mathsf{mouse}) \rightarrow \Big(\mathsf{Diag}(\mathsf{rodent}) \rightarrow \big(\mathsf{mouse} \subseteq \mathsf{rodent} \rightarrow \mathsf{mouse} \circ \mathsf{hasPaw} \subseteq \mathbb{1} \circ \mathsf{rodent} \circ \mathsf{hasPaw} \big) \Big)$$

□

Example. Figure 1 contains a map formulation of a theory whose intended domain of discourse is $\mathbb{N} = \{0, 1, 2, \ldots\}$. This theory, which is an adaptation of [End72], pp.184–187, characterizes the usual ordering relation on numbers, denoted $<$, and the increment operation $u \mapsto u + 1$, denoted suc.

As a clarification, let us translate its axioms back into first-order logic, using the map letters zero and suc as if they were a constant and a monadic function symbol. We indicate ι, $\bar{\iota}$, and exclusive disjunction by $=$, $\neq$, and $\not\leftrightarrow$, and leave universal quantifiers implicit:

$$\begin{aligned} & x < y \rightarrow \mathsf{zero} \neq y\,, && \mathsf{zero} \neq y \rightarrow \exists v\, \mathsf{suc}(v) = y\,, \\ & \exists v \big(x < v \wedge v = \mathsf{suc}(y) \big) \leftrightarrow \big(x < y \not\leftrightarrow x = y \big)\,, && x < y \not\leftrightarrow x = y \not\leftrightarrow y < x\,, \\ & \neg\big(x < y \wedge x = y \big)\,, && \exists v \big(x < v \wedge v < y \big) \rightarrow x < y\,. \end{aligned}$$

As a further explanation, let us rephrase as follows the recursive characterization in Figure 1 of the operations $u \mapsto 2 \cdot u$ and $u \mapsto 2^u$:

$$\begin{aligned} & \mathsf{twice}(\mathsf{zero}) = \mathsf{zero}\,, && \mathsf{twice}\big(\mathsf{suc}(x)\big) = \mathsf{suc}\big(\mathsf{suc}\big(\mathsf{twice}(x)\big)\big)\,, \\ & \mathsf{twoTo}(\mathsf{zero}) = \mathsf{suc}(\mathsf{zero})\,, && \mathsf{twoTo}\big(\mathsf{suc}(x)\big) = \mathsf{twice}\big(\mathsf{twoTo}(x)\big)\,. \end{aligned}$$

□

In spite of all the extension described above, $\mathcal{L}^{\times}$ 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 7 and 8.

Axioms

$> \equiv_{\mathrm{Def}} <^{-1}$	$\leq \equiv_{\mathrm{Def}} < \triangle \iota$
$\mathsf{Const}(\,\mathsf{zero}\,)$	$\mathsf{TotFun}(\,\mathsf{suc}\,)$
$< \subseteq \mathbb{1} \circ \mathsf{zero}$	$\mathbb{1} \circ \mathsf{zero} \subseteq \mathbb{1} \circ \mathsf{suc}$
$< \circ \mathsf{suc}^{-1} = \leq$	$\leq \triangle > \, = \mathbb{1}$
$< \cap \iota = \emptyset$	$< \circ < \subseteq <$

Primitive map letters: zero, suc, $<$

Lemmas

$< \subseteq \not>$	$\mathsf{suc} \subseteq <$
$\iota \cap \not< = \iota$	$\not< = (\, < \triangle \iota\,)^{-1}$
$< = \mathsf{suc} \circ < \circ \mathsf{suc}^{-1}$	$\mathsf{suc} \circ \mathsf{zero} = \emptyset$
$\mathsf{Fun}(\,\mathsf{suc}^{-1}\,)$	$\mathsf{suc} \underbrace{\circ \mathsf{suc} \cdots \circ \mathsf{suc}}_{n \text{ times}} \subseteq \bar{\iota}$

A conservative extension

$\mathsf{TotFun}(\,\mathsf{twice}\,)$	$\mathsf{TotFun}(\,\mathsf{twoTo}\,)$
$\mathsf{zero} \subseteq \mathsf{twice}$	$\mathsf{suc} \circ \mathsf{twice} \subseteq \mathsf{twice} \circ \mathsf{suc} \circ \mathsf{suc}$
$\mathsf{zero} \circ \mathsf{suc} \subseteq \mathsf{twoTo}$	$\mathsf{suc} \circ \mathsf{twoTo} \subseteq \mathsf{twoTo} \circ \mathsf{twice}$

Additional primitive map letters: twice, twoTo

Additional lemmas

$\mathsf{twice} \subseteq \leq$
$\mathsf{twice}^{-1} \circ \mathsf{twoTo} \subseteq \leq$
$(\, \leq \circ \mathsf{twice}\,) \cap (\,\mathsf{twice} \circ >\,) = \emptyset$
$(\, \leq \circ \mathsf{twoTo}\,) \cap (\,\mathsf{twoTo} \circ >\,) = \emptyset$

Figure 1: Theory of natural numbers with successor function and ordering relations

3 Raw deductive machinery for a map calculus

We will now slightly adjust the derivability notion for $\mathcal{L}^{\times}$ formalized in [TG87] 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 $\mathcal{L}^{\times}$:

$P \cap Q = Q \cap P$	
$\big(P \cap (Q \triangle R)\big) \triangle (P \cap Q) = P \cap R$	
$\mathbb{1} \cap P = P$	
$(P \star Q) \star R = P \star (Q \star R)$	$\star \in \{\triangle, \cap, \circ\}$ (chosen once)
$\iota \circ P = P$	
$(P \cup Q) \circ R = (Q \circ R) \cup (P \circ R)$	
$P^{-1-1} = P$	
$(P \star Q)^{-1} = Q^{-1} \star P^{-1}$	$\star \in \{\cap, \circ\}$ (chosen once)
$\big(P^{-1} \circ \big(R \setminus (P \circ Q)\big)\big) \cap Q = \emptyset$	

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 $\Theta^{\times}(\mathtt{E})$.*

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]

Remarks.

(1) We conjecture that the associativity of $\cap$, $\triangle$, along with the commutativity of $\cap$ and with the law $\big(P \cap (Q \triangle R)\big) \triangle (P \cap Q) {=} P \cap R$ suffice, quite independently of the rest of $\Lambda^{\times}$, to yield the Boolean ring structure (cf., e.g., [Jac51]). This conjecture rests on the affinity between this formula and the Robbins' law (cf. [Win90] and Figure 2). Should it prove false, then in a context without $\mathbb{1}$ (cf. Sec.10) we would replace the latter two laws by the following logical axioms schemes:

$P \triangle P {=} Q \triangle Q$	$P \triangle (Q \triangle P) {=} Q$	$P \cap P {=} P$
$(R \cap Q) \triangle (R \cap P) {=} (P \triangle Q) \cap R$		

[2] Any derivability notion for first-order logic can be exploited for $\mathcal{L}^{+}$, cf., e.g., [TG87].

(Anyway, this lengthier axiomatization may be preferred in order to drive derivations more efficiently.)

(2) The language $\mathcal{L}^{\times}$ was originally based on the constructs of union $\cup$ and complementation $^{-}$. Accordingly it employed, as Boolean part of $\Lambda^{\times}$, the three classical axioms due to Huntington (cf. [Hun33]). Quite recently it was proved that, in the context of a purely Boolean axiomatization based on $\cup$ and $^{-}$, the Huntington's law can be substituted by the Robbins' law (cf. [McC97]).

In the version of map calculus proposed by [TG87], the problem of deriving the Huntington's law from the Robbins' law surprisingly turns out to be as easy as the reverse problem, thanks to the contribution of the axioms on $\circ$ and $^{-1}$.

□

4 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 [TG87], 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. [McC95]), to experiment with $\mathcal{L}^{\times}$ (cf. [AOT98]). 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 [TG87], pp.49-50, that we had chosen as our benchmarks (cf. Figure 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. [PS95]) and exploiting a decider for map constructs embedded in set theory (cf. Chapter 9 of [CFO89]), 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 or theorem scheme of a theory based on map calculus (examples of this, stressed in boldface in Figures 5, 7, 10, and 11, will be two induction principles, various formulations of the subset axiom scheme, and the replacement 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 [SDDS86] a basic layer of Boolean-Peircean simplifications applicable to $\mathcal{L}^{\times}$-expressions and equalities (cf. Sec.9.1).

a.	$P \circ \iota = P$	right unit for $\circ$
b.	$(P \triangle \mathbb{1}) \triangle \mathbb{1} = P$	double complementation law
c.	$P \triangle P = \emptyset$	periodicity of $\triangle$
d.	$P \triangle Q = Q \triangle P$	commutativity of $\triangle$
e.	$(R \cap Q) \triangle (R \cap P) = (P \triangle Q) \cap R$	distributive-commutative law
f.	$P \subseteq Q \rightarrow (Q \subseteq R \rightarrow P \subseteq R)$	
g.	$P \triangle (Q \triangle P) = Q \qquad (P \triangle Q) \triangle Q = P \qquad P \cap P = P$	
h.	$P \cup Q = Q \cup P \qquad (P \cup Q) \cup R = P \cup (Q \cup R) \qquad P \cup Q = \overline{\overline{P} \cap \overline{Q}}$	
i.	$\overline{\overline{P \cup Q} \cup \overline{P \cup \overline{Q}}} = P \qquad \overline{\overline{P \cap Q} \cap \overline{P \cap \overline{Q}}} = P$	Robbins' laws
j.	$(P \cap Q) \triangle (P \cap \overline{Q}) = P$	variant of Robbins' law
k.	$(P \cup Q) \cap (Q \cup \overline{P}) = Q \qquad (\overline{P} \cap Q) \cup (Q \cap P) = Q$	Huntington's laws
l.	$\emptyset \circ P = \emptyset \ \wedge \ P \circ \emptyset = \emptyset \ \wedge \ P \triangle \emptyset = P \ \wedge \ P \cap \emptyset = \emptyset$	
m.	$\big((P \circ Q) \cap R\big) = \emptyset \ \rightarrow \ \big((P^{-1} \circ R) \cap Q\big) = \emptyset$	cycle law
n.	$\big((P^{-1} \circ R) \cap Q\big) = \emptyset \ \rightarrow \ \big((P \circ Q) \cap R\big) = \emptyset$	cycle law
o.	$(P \cap \iota) = P \ \rightarrow \ (P^{-1} \cap \iota) = P \ \wedge \ P^{-1} = P$	
p.	$P \subseteq P \circ \mathbb{1} \circ P \qquad \mathsf{Const}(P) \rightarrow P \subseteq \iota$	
q.	$(\forall x, u, v)\big((x \cap u) \triangle v = \emptyset \ \longleftrightarrow$	
	$v \cap (u \triangle \mathbb{1}) = \emptyset \ \wedge \ \exists y \ x = v \triangle \big(y \cap (y \triangle \mathbb{1})\big)\big)$	key for Boolean unification

Figure 2: Some benchmark theses, with Otter serving $\mathcal{L}^\times$ and $\mathcal{L}^+$

5 Translating first-order sentences into map equalities

It is shown in [TG87] 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 [WR10]) leads to a far more conventional first-order language, $\mathcal{L}$, where ι generally takes the typographic form $=$ (not to be confused with $=$). A remarkable fact about the elimination technique[3] is the following: when one applies it to a sentence β of $\mathcal{L}_3^+$, i.e., a β of $\mathcal{L}^+$ that *involves no more than three distinct individual variables*, the resulting sentence α will also involve three or fewer variables. This is what happens, e.g., when β is an equality $Q{=}R$ of $\mathcal{L}^\times$.

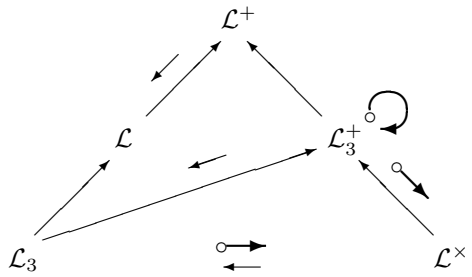


Figure 3: Embeddings and translatability relations between formalisms

To what extent is the reverse translation of $\mathcal{L}^+$ into $\mathcal{L}^\times$ possible? Let us recall a definition from [TG87], p.62:

Definition. *A sentence α of $\mathcal{L}^+$ is said to be* EXPRESSIBLE *in $\mathcal{L}^\times$ if there is a map equality β of $\mathcal{L}^\times$ for which $\alpha \dashv\models^+ \beta$, i.e., $\alpha^{\Im} = \beta^{\Im}$ in every interpretation $\Im$.* □

Among sentences expressible in this sense, one finds all sentences in three variables, where the following definition applies:

[3]This translation is represented by downward and leftward arrows in Figure 3.

Definition. *A formula ψ of $\mathcal{L}^+$ is said to be* IN k VARIABLES *(k a natural number) if no subformula φ of ψ involves more than k distinct free variables.* □

In the sequel, we outline a quantifier-elimination process that applies to any formula in three variables. When it receives in input a sentence α, this algorithm re-expresses it as a sentence β of $\mathcal{L}^\times$. One cannot do entirely without the three-variable restriction; indeed, the possibility of performing the reverse translation (from $\mathcal{L}^\times$ to $\mathcal{L}_3^+$), reveals 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 of $\mathcal{L}_3^+$. As was shown in [Kwa81], the collection of all such γs is undecidable.

Our current translation purpose can be achieved by means of rewriting rules (see Figure 4) defining a computable total function[4] H : {formulas of $\mathcal{L}_3^+$} $\longrightarrow$ {formulas of $\mathcal{L}_3^+$} with the following properties:

- for each formula φ in three variables, $H\varphi$ is quantifier-free; moreover, φ and $H\varphi$ have the same free variables;
- when restricted to the sentences in three variables, 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. [Omo97]): first the occurrences of negation are moved inwards, near atomic sub-formulas, then they are removed using the rules displayed in Figure 4. 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 disjunctions. 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 (cf. Sec.9.1).

5.1 A conservative translation technique

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 [CCO97], led us to improved and generalized techniques implemented first in SETL2 and then —partly— in Java. We will briefly review now one such technique.

Algorithm. (Graph thinning) An existentially quantified conjunction φ of literals of the form xPy is given, where x and y are variables and P is a map expression (negative literals have been rewritten in the form $x\overline{Q}y$). Free variables may occur in φ.

The goal is to find a quantifier-free conjunction —or simply an atom, if there are at most two free variables in φ— equivalent to φ. Initially, an undirected graph G_φ is built so that:

1) G_φ has a node n_{v_i} for each distinct variable v_i occurring in φ; and

2) 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 $j < i$.

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

$\neg xPy \rightsquigarrow x\overline{P}y$	$\neg P{=}Q \rightsquigarrow \diamond(\, P \mathbin{\triangle} Q\,){=}\mathbb{1}$
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 $u \not\equiv v$ (The first two of these rules are applied only when x precedes y w.r.t. the ordering $\mathsf{v}_1, \mathsf{v}_2, \ldots$ of variables.) Swapping rules: $\star \in \{\vee, \wedge\}$.	$P{=}Q \star uRv \rightsquigarrow u\diamond\overline{(\, P \mathbin{\triangle} Q\,)}v \star uRv$ $uSu \star uRv \rightsquigarrow u(\, S \cap \iota\,) \circ \mathbb{1}v \star uRv$ $uSu \star vRu \rightsquigarrow v\mathbb{1} \circ (\, S \cap \iota\,)u \star vRu$ $uSu \vee vRv \rightsquigarrow u(\, S \cap \iota\,) \circ \mathbb{1}v \vee u\mathbb{1} \circ (\, R \cap \iota\,)v$ 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.	$\forall x(\, \varphi \wedge \psi\,) \rightsquigarrow \forall x\, \varphi \wedge \forall x\, \psi$ $\exists x(\, \varphi \vee \psi\,) \rightsquigarrow \exists x\, \varphi \vee \exists x\, \psi$ $\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u\, \psi \rightsquigarrow \psi$ $\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u \Big(\, \varphi \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \psi\,\Big) \rightsquigarrow \Big(\, \left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u\, \varphi\,\Big) \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \psi$ $\left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u \Big(\, \psi \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \varphi\,\Big) \rightsquigarrow \psi \left\{ \begin{smallmatrix} \vee \\ \wedge \end{smallmatrix} \right\} \left\{ \begin{smallmatrix} \forall \\ \exists \end{smallmatrix} \right\} u\, \varphi$ 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.	
$\forall u\, uPu \rightsquigarrow \iota \subseteq P$ $\forall u\, uQv \rightsquigarrow v\emptyset \dagger Qv$ $\forall u\, vQu \rightsquigarrow vQ \dagger \emptyset v$ $\forall u(\, uQv \vee wRu\,) \rightsquigarrow wR \dagger Qv$ $\forall u(\, uQv \vee uRw\,) \rightsquigarrow wR^{-1} \dagger Qv$ $\forall u(\, vQu \vee wRu\,) \rightsquigarrow vQ \dagger R^{-1}w$ $\forall u(\, vQu \vee uRw\,) \rightsquigarrow vQ \dagger Rw$ 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 4: Rewriting rules employed to translate $\mathcal{L}_3^+$ into $\mathcal{L}^\times$

The nodes corresponding to variables bound in φ are called *bound* nodes.

This G_φ and its labels will be manipulated as stated below, and the map labels of edges $\{n_{\mathsf{v}_i}, n_{\mathsf{v}_j}\}$ will always refer to the orientation leading from smaller variable subscripts to greater ones.

Initially, every loop-edge $\{n_x, n_x\}$ is eliminated 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 together (by one of the merging rules).

The elimination of bound nodes (which represents the elimination of existential quantifiers from φ) is performed by repeatedly 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 placed 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 $\mathbb{1}$, performing bypass of 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 than 1, the sought conjunction can be directly read off the graph, else we have a failure. □

Remark. It can be shown that the computational complexity of the above graph thinning algorithm is $\mathcal{O}(n^2 + m)$ where n is the number of distinct variables in the input formula φ, and m is the number of conjuncts of φ. □

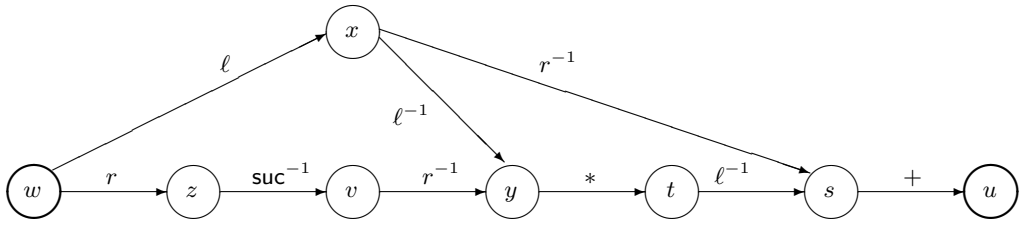
The following example shows how the above-outlined algorithm works; it also illustrates how, sometimes, one can recover from failure, given that certain map letters must designate functions.

Example. Let us consider the following first-order formula

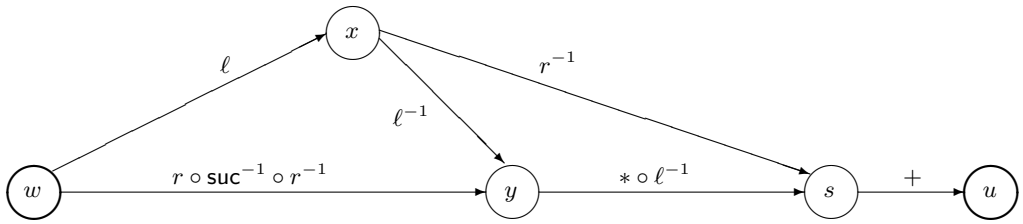
$$\forall w\,\forall u\,\big(\,w * u \;\longrightarrow\; \exists x\,\exists z\,\exists v\,\exists y\,\exists t\,\exists s\;\;(\,w\,\ell\,x \,\wedge\, w\,r\,z \,\wedge\, v\,\mathsf{suc}\,z \,\wedge\, y\,\ell\,x \,\wedge\, y\,r\,v \,\wedge y * t \,\wedge\, s\,\ell\,t \,\wedge\, s\,r\,x \,\wedge\, s + u\,)\,\big),$$

where the assumption $\mathsf{Fun}(\,\ell\,)$ is made.

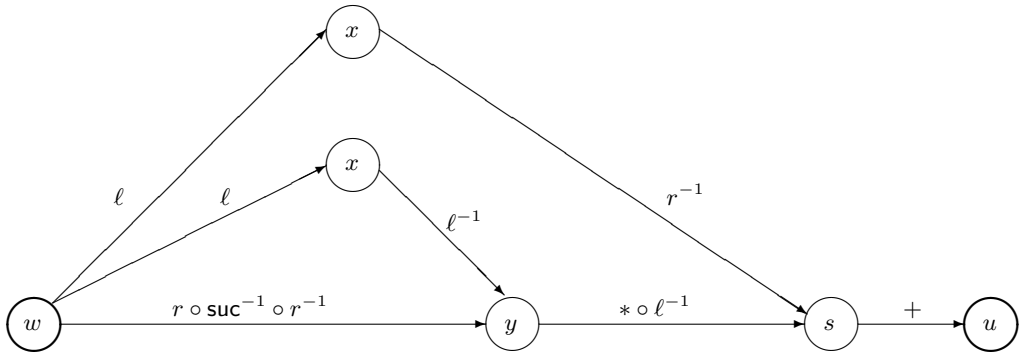
Consider the consequent of the implication. The algorithm starts with building the following graph G. The nodes labeled w and u can be regarded as *source* and *sink*, respectively; the remaining nodes are the bound nodes of G. The aim is to reduce the whole graph G to a single edge between source and sink.



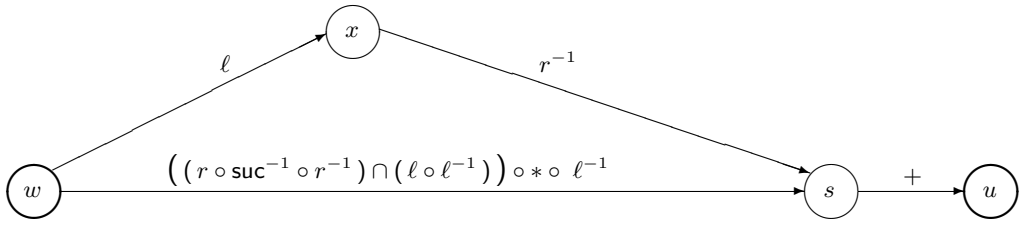
Three applications of the bypass rule, eliminate the nodes labeled z, v, and t:



At this point the graph-thinning process is stuck, because no further application of bypass or bigamy is possible. Then the fact that ℓ is a function comes into play. We can duplicate the node labeled x together with the edge labeled ℓ. The exiting edges (labeled ℓ^{-1} and r^{-1}, respectively) are divided between the two x-nodes, as shown below.



Now we can resume, and bypass an occurrence of x (the lower one in the picture, to be specific), and then the node labeled y. The resulting graph is:



Further applications of the bypass rule yield the map expression

$$\Big(\big(\big((r \circ \mathsf{suc}^{-1} \circ r^{-1}) \cap (\ell \circ \ell^{-1})\big) \circ * \circ\ \ell^{-1}\big) \cap (\ell \circ r^{-1})\Big) \circ +,$$

which translates the consequent of the original implication (the translation of the entire formula will be seen in Figure 7). □

Example. Let us consider a base $\mathcal{B}$ of Horn clauses subject to the following restrictions:

- all predicate letters in $\mathcal{B}$ are dyadic;
- $\mathcal{B}$ 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

$$urv \leftarrow \bigwedge_{i=1}^{n} x_i p_i y_i,$$

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 (no longer subject to the Horn restriction) 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 $\diamond(Q \mathbin{\triangle} R){\subseteq}r$. 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 $\bigcup_{j=1}^{m} S_j{=}r$. □

6 A theory of lists, within map calculus

As will be explained below, the axioms in Figure 5 describe a somewhat more challenging scenario than the ones in Figure 1. Intuitively speaking, here the domain $\mathcal{U}$ of discourse consists of entities of three separate kinds: an infinity of atoms, nested lists constructed out of them, and an individual NL. We are now beginning to experiment with axioms of this kind in Otter.

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

Figure 5: Axioms on nested lists

The 'nil' predicate, according to a., represents the distinguished individual NL, not to be counted among atoms. By b. and c., the 'head' and 'tail' predicates are partial functions defined

for the same entities (to be regarded as the 'lists') of $\mathcal{U}$: the complement of their common domain consists of all atoms together with NL. Notice that, by c.2, the 'is atom' predicate does not truly depend on its second argument.

The axiom d.1 states that for any given pair a, b in $\mathcal{U}$ with b a list or $b = \mathsf{NL}$, one can find a list whose head is a and whose tail is b; moreover, by d.2, lists that differ from one another cannot have the same head and the same tail. By axiom e., for any given c in $\mathcal{U}$ there is an atom that does not occur in c (as a consequence, the number of atoms will turn out to be infinite as desired). In this connection a recursive characterization of the 'occurs in' predicate applies: it can be read as "an entity occurs in a list b if and only if it either coïncides with, or occurs in, the head of b, or it occurs in the tail of b". By the acyclicity requirement f.1, a c in $\mathcal{U}$ can neither occur within itself nor be its own tail.

Last comes the somewhat intriguing axiom f.2, which is an induction principle for lists. When its outer map-constructs get eliminated in $\mathcal{L}^+$, it becomes

$$(\forall x,\, y)\Big(\, x \,\mathsf{at}\, y \vee x = \mathsf{NL} \vee \big(\, \underbrace{\exists v(\, x \,\mathsf{hd}\, v \wedge v\, P\, y\,)}_{\mathsf{hd}(x)Py} \wedge \underbrace{\exists w(\, x \,\mathsf{tl}\, w \wedge w\, P\, y\,)}_{\mathsf{tl}(x)Py} \,\big) \to x\, P\, y \,\Big) \longrightarrow (\forall x, y)\, x\, P\, y\,,$$

where P can be any map expression. Later on, we will likewise express in map-theoretic terms the induction principle of Peano arithmetic:

$$\Big(\, \emptyset \dagger \big(\, P \cup (\, \mathsf{suc}^{-1} \circ \mathbb{1}\,) \,\big) \cap \big(\, \overline{P} \cup (\, \mathsf{suc} \circ P\,) \,\big) \dagger \emptyset \,\Big) \circ \overline{P} = \emptyset\,,$$

with suc retaining the same intended meaning as in Figure 1.

As a side remark we show how small modifications are needed in order to treat flat lists. Simply, axioms d.1 and e.2 have to be replaced by those listed in Figure 6.

d.1	$\overline{\mathsf{at}} \cap \mathsf{at}^{-1} \subseteq \mathsf{tl}^{-1} \circ \mathsf{hd}$
e.2	$\mathsf{ocl} = (\, \iota \circ \mathsf{hd}^{-1}\,) \cup (\, \mathsf{ocl} \circ \mathsf{tl}^{-1}\,)$

Figure 6: Variant axioms for flat lists

7 From first-order theories to the map calculus

It often turns out that a first-order sentence α, even though inexpressible in $\mathcal{L}^\times$ when taken in isolation, *becomes expressible* within the context of a theory. This is to say, when one adopts a decidable collection T of sentences as axioms, it may well be the case that $T \vdash^+ \alpha \leftrightarrow \beta$, where β is in three variables (or, which amounts to the same thing, β belongs to $\mathcal{L}^\times$). [GKV97] 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 T total orderings $<$, the circumstance can be stated as follows:

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

This new, more generous, meaning of the term "expressibility", trivializes the entire question of expressibility in STRONG THEORIES, such as are number theories (e.g. the Peano arithmetic or the additive-multiplicative theory of real numbers) and set theories (Zermelo-Fraenkel, Gödel-Bernays, etc.). We are referring to theories where by a sentence in three variables one can state that two specific relations —let us denote them here as ℓ (for "left") and r (for "right")— are CONJUGATED QUASI-PROJECTIONS, in the sense that

- ℓ and r are functions (at least partial) on the domain $\mathcal{U}$ of discourse,
- for any pair a, b of entities in $\mathcal{U}$ there is a c in $\mathcal{U}$ such that $\ell(c) = a$ and $r(c) = b$.

Indeed, there is a general technique that enables one to reduce to three the number of variables in any sentence α of the language of a strong theory, by suitably exploiting the map expressions L, R that describe conjugated quasi-projections.

Notice that L, R are to meet the conditions

$$\mathsf{Fun}(\, L\,), \quad \mathsf{Fun}(\, R\,), \quad L^{-1} \circ R = \mathbb{1}.$$

Systematic ways of performing the translation of any sentence of $\mathcal{L}^+$ into $\mathcal{L}^\times$, irrespective of the number of individual variables, relative to a pair of conjugated quasi-projections, are discussed in [TG87], pp.95-124.

7.1 Peano arithmetic, within map calculus

To illustrate the use of conjugated projections, 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 dyadic operations. Normally these are regarded as triadic 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 B exists between $\mathcal{U}$ and $\mathcal{U}^2$, we can conservatively extend the theory with two monadic function symbols, ℓ and r, which represent the 'left' and 'right' projections of numbers. That is, $b = \ell^{\Im}(a)$ and $c = r^{\Im}(a)$ are to fulfill $B(a) = [b, c]$ 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 7.

Most of the remaining axioms of Figure 7 were obtained in [CCO97] from those of a standard first-order axiomatization of the Peano arithmetic (cf., e.g., [Men79]). The graph-thinning algorithm outlined above played a crucial rôle in this task (see the first example in Sec.5.1). The formulation provided here differs only marginally from the one in [CCO97].

$\mathsf{TotFun}(\,\ell\,)$	$\mathsf{TotFun}(\,r\,)$
$\ell^{-1} \circ r{=}\mathbb{1}$	$(\,\ell \circ \ell^{-1}\,) \cap (\,r \circ r^{-1}\,) \subseteq \iota$
$\mathsf{Const}(\,\mathsf{zero}\,)$	$\mathsf{suc} \circ \mathsf{zero}{=}\mathsf{zero}$
$\mathsf{TotFun}(\,\mathsf{suc}\,)$	$\mathsf{Fun}(\,\mathsf{suc}^{-1}\,)$
$\mathsf{TotFun}(\,+\,)$	$\mathsf{TotFun}(\,*\,)$
$\ell \cap (\,r \circ \mathsf{zero} \circ \mathbb{1}\,) \subseteq +$	$r \circ \mathsf{zero} \subseteq *$
$\big((\,\ell \circ \ell^{-1}\,) \cap (\,r \circ \mathsf{suc}^{-1} \circ r^{-1}\,)\big) \circ + \circ \mathsf{suc} \subseteq +$	
$\big(\big(((\,\ell \circ \ell^{-1}\,) \cap (\,r \circ \mathsf{suc}^{-1} \circ r^{-1}\,)) \circ * \circ \ell^{-1}\big) \cap (\,\ell \circ r^{-1}\,)\big) \circ + \subseteq *$	
$\big(\boldsymbol{P} \cup (\,\mathsf{suc}^{-1} \circ \mathbb{1}\,)\big) \cap \big(\overline{\boldsymbol{P}} \cup (\,\mathsf{suc} \circ \boldsymbol{P}\,)\big) = \mathbb{1} \rightarrow \boldsymbol{P}{=}\mathbb{1}$	
Primitive map letters: zero, suc, $+$, $*$, ℓ, r	

Figure 7: Map-formulation of Peano arithmetic (in $\mathcal{L}^\times$)

The last item in Figure 7 states 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.

7.2 Finite set theory, within map calculus

As a further example of use of conjugated quasi-projections, we now show how to formulate a set theory in the map formalism. Figure 9 contains an axiomatization of a theory of hereditarily finite sets including the axioms of extensionality **(E)**, pairing **(Pair)**, subset **(S)**, unionset **(U)**, powerset **(Pow)**, transitive embedding **(T)**, finiteness **(F)**, and regularity **(R)**. Notice that **(S)** is an axiom scheme parametrized by the map expression P.

To set the ground for the map formulation of these axioms, some helpful notation is provided, see Figure 8, (the set inclusion relation $\subseteq$ should not to be confused with the map inclusion relation $\subseteq$). The rather straightforward outcome of the translation is then displayed in Figure 10. The main map constructor is $\nabla(\,\cdot\,)$. It is defined in such a manner that, for each map expression P, it holds that $x\nabla(\,P\,)y \leftrightarrow \forall u(\,xPu \leftrightarrow u \in y\,)$. Through this constructor, the formal characterization of familiar set operations such as monadic union $\mathcal{U}n$, power set $\mathcal{P}ow$, dyadic union $\cup$, and set difference $\setminus$, are immediate. The null set is characterized by means of the predicate void (cf. **(N)** in Figure 11), absolute on its right. π_0 and π_1 are a pair of partial functions that, thanks to the axiom **(Pair)** of Figure 10, will turn out to be conjugated quasi-projections (cf. [TG87] p. 129, and

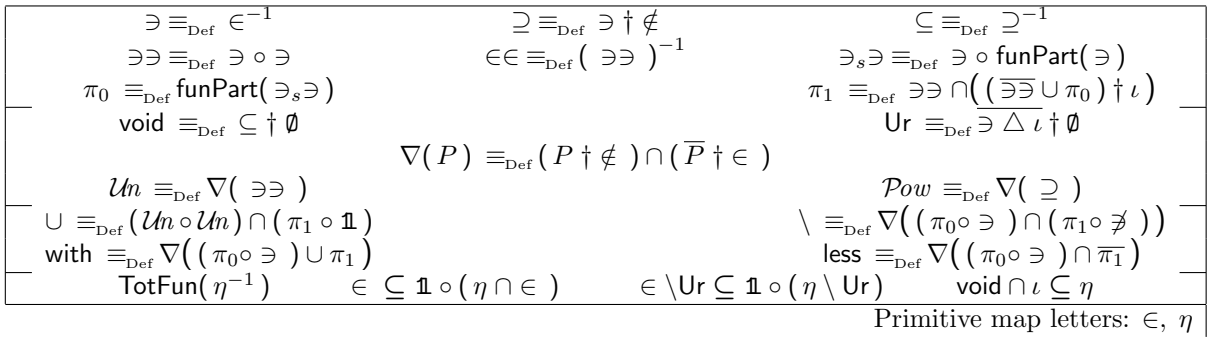


Figure 8: Definitions of useful derived set constructs (in $\mathcal{L}^{\times}$ and $\mathcal{L}^{+}$)

(E)	$\forall x\, \forall y \big(\forall u \,(u \in x \leftrightarrow x \in y) \rightarrow x = y \big)$
(Pair)	$\forall x\, \forall y\, \exists z\, \forall u \,(u \in z \leftrightarrow u = x \vee u = y)$
(S)	$\boldsymbol{\forall w\, \exists y\, \forall u \,\big(\exists x \,(w \pi_1 x \wedge u \in x) \wedge w P u \leftrightarrow u \in y \big)}$
(U)	$\forall x\, \exists z\, \forall y \big(y \in z \leftrightarrow \exists u \,(u \in x \wedge y \in u) \big)$
(Pow)	$\forall x\, \exists z\, \forall y \big(y \in z \leftrightarrow \forall u \,(u \in y \rightarrow u \in x) \big)$
(T)	$\forall x\, \exists z \big(x \in z \wedge \forall y\, \forall u \,(y \in u \wedge u \in z \rightarrow y \in z) \big)$
(F)	$\forall x \Big(\exists y \,(y \in x) \ \rightarrow\ \exists z \,\big(z \in x \wedge \forall u \,(u \in x \wedge u \neq z \rightarrow \exists v \,(v \in u \wedge v \notin z)) \big) \Big)$
(R)	$\forall x\, \forall v \big(v \in x \wedge \neg \mathsf{Ur}(v) \rightarrow \eta x \in x \wedge \neg \mathsf{Ur}(\eta x) \wedge v \notin \eta x \big)$

Figure 9: First-order formulation of the theory of hereditarily finite sets (in $\mathcal{L}^{+}$)

Sec.9.2 below). The predicate η, defined by the last four map equalities in Figure 8, corresponds to the Skolem function occurring in the first-order formulation of **(R)**.

(E)	$\nabla(\ \ni\ ) \subseteq \iota$	**(Pow)**	$\mathsf{Total}(\mathcal{P}ow)$
(Pair)	$\mathbb{1} = \pi_0^{-1} \circ \pi_1$	**(T)**	$\mathsf{Total}\big((\emptyset \dagger (\overline{\in\in} \cup \in)) \cap \in \big)$
(S)	$\mathsf{Total}\big(\boldsymbol{\nabla}\big((\boldsymbol{\pi_1} \circ \ni) \cap \boldsymbol{P} \big) \big)$	**(F)**	$\in \subseteq \mathbb{1} \circ \big(\in \cap \big((\iota \cup \not\supseteq) \dagger \notin \big) \big)$
(U)	$\mathsf{Total}(\mathcal{U}n)$	**(R)**	$\in \setminus \mathsf{Ur} \subseteq \ \notin \circ \eta$

Figure 10: Map-formulation of finite set theory (in $\mathcal{L}^{\times}$)

Let us explain the few theorems of this set theory that are listed in Figure 11: **(N)** states the existence of a null set; the addition and removal lemmas **(W)**, **(L)** enable one to insert/withdraw a given v into/from any set x so as to obtain the sets $x \cup \{v\}$ and $x \setminus \{v\}$ respectively.

The consequences of the proposed axioms also comprise (mainly by virtue of **(T)** and **(F)**) the usual replacement scheme **(Repl)**, the principle **(Ch)** of choice, a weaker formulation, **($\mathbf{S}_0$)**, of the subset scheme and a broader formulation, **($\mathbf{S}_1$)**, of it.

In order to allow the theory to deal with urelements, the following extension is needed to bring plenty of urelements into set theory:

(Ur)	$\overline{\ni\ni} \circ \mathsf{Ur} = \mathbb{1}$
i.e.,	$\forall y\, \exists z \big(z \in z \wedge \forall u \,(u \in y \rightarrow z \notin u) \big)$

As by Figure 8, Ur is a predicate (absolute on its right) that characterizes urelements as being those individuals a such that $a \in \{a\}$, very much in the spirit of [Qui63]. The 'plenitude' axiom **(Ur)** implies the existence of infinitely many urelements.

To avoid urelements altogether, one might choose to replace plenitude by the antithetic axiom $\mathsf{Ur} = \emptyset$, which would transform the above **(R)** into the unrestrained regularity axiom:

$$\in \subseteq \notin \circ \eta, \quad \text{i.e., } \forall x \forall y (\, y \in x \rightarrow \eta x \in x \wedge y \notin \eta x \,).$$

In the same frame of mind, one could supersede the finiteness axiom **(F)** (based on [Tar24]) by an infinity axiom **(I)** such as, for instance, the one of [PP91], presented in Figure 12.

(N)	$\mathsf{LConst}(\,\mathsf{void}\,)$	**(Repl)**	$\mathsf{Total}\big(\,\boldsymbol{\nabla}\big(\,(\ \ni \circ \mathsf{funPart}(\,\boldsymbol{Q}\,)\,)\cap \boldsymbol{P}\,\big)\,\big)$
(W)	$\mathsf{Total}(\,\mathsf{with}\,)$	$(\mathbf{S}_0)$	$\mathsf{Total}\big(\,\boldsymbol{\nabla}(\ \ni \cap \boldsymbol{P}\,)\,\big)$
(L)	$\mathsf{Total}(\,\mathsf{less}\,)$	$(\mathbf{S}_1)$	$\mathsf{Fun}(\,\boldsymbol{Q}\,)\rightarrow \mathsf{Total}\big(\,\boldsymbol{\nabla}\big(\,(\,\boldsymbol{Q}\circ\ni\ )\cap \boldsymbol{P}\,\big)\,\big)$
		(Ch)	$\mathsf{Total}\big(\ \not\ni \dagger \big(\,\mathsf{void}^{-1}\cup(\ \ni\circ\in\ )\big)\big)$

Figure 11: Theorems in the hereditarily finite set theory (in $\mathcal{L}^{\times}$)

(I) $\quad(\ \ni \dagger \overline{\in\in}\,)\cap\notin\cap\big(\ \not\ni\dagger(\ \in\cup\ni\ )\dagger\notin\ \big)\cap\not\ni\cap(\,\overline{\ni\ni}\dagger\in\ )\not\subseteq \iota$

i.e., $\quad \exists w_0\,\exists w_1\,\big(\,\bigcup w_0\subseteq w_1\ \wedge\ \bigcup w_1\subseteq w_0\ \wedge\ w_0\notin w_1\ \wedge\ w_1\notin w_0\ \wedge\ w_0\neq w_1\ \wedge\ \forall x\in w_0\,\forall y\in w_1\,(\,x\in y\vee y\in x\,)\,\big)$

Figure 12: Infinity axiom for set theory

However, when it comes to exploiting Set Theory in algorithm specification (cf. [SDDS86]) or program certified derivation (cf. [KP95]), an axiomatic system hosting urelements at will and imposing that all sets be finite, like the one proposed above, seems preferable to such well-established systems as Zermelo-Fraenkel (cf. [Sho67]).

8 A tough exercise in map specification: lattice theory

As for expressibility in $\mathcal{L}^{\times}$, [Kwa81] and [TG87] classify many sentences as shown in Figure 13. Of these sentences, the ones in the first group are provable in the most varied theories of sets; those in the second group can be interpreted in LATTICE THEORY (which deals with the ordering relations where every finite nonempty subset of the domain has inf[imum] and sup[remum]). In contrast with [Kwa81], which claims that **(3)** is inexpressible, [TG87] says that it is not known whether or not **(3)** is expressible in $\mathcal{L}^{\times}$. The collection of all sentences expressible in $\mathcal{L}^{\times}$ was

(1)	$(\forall x,\,y)\exists w\,\forall v(\,v\in w\leftrightarrow v=x\vee v=y\,)$	YES
(1a)	$(\forall x,\,y)\big(\,\exists u\,(\,x\in u\wedge y\in u\,)\rightarrow\exists w\,\forall v(\,v\in w\leftrightarrow v=x\vee v=y\,)\,\big)$	NO
(1b)	$(\forall x,\,y)\big(\,\exists u\,x\in u\wedge\exists v\,y\in v\rightarrow\exists w\,\forall v(\,v\in w\leftrightarrow v=x\vee v=y\,)\,\big)$	NO
(1c)	$(\forall x,\,y)\big(\,\exists u\,x\in u\wedge\exists v\,y\in v\rightarrow\exists w\,\big(\,\forall v(\,v\in w\leftrightarrow v=x\vee v=y\,)\wedge\exists u\,w\in u\,\big)\,\big)$	YES
(2)	$(\forall x,\,y)\exists u\,\forall v(\,v\in u\leftrightarrow v\in x\vee v\in y\,)$	NO
(2a)	$(\forall x,\,y)\exists u\,\forall v(\,v\in u\leftrightarrow v\in x\vee v\in y\,)\wedge\forall x\,\exists y\,\forall v(\,v\in y\leftrightarrow v=x\,)$	YES
(3)	$(\forall x,\,y)\exists w\,\forall u(\,u\in w\leftrightarrow u=x\vee u\in y\,)$	NO (?)
(3a)	$(\forall x,\,y)\exists w\,\forall u(\,u\in w\leftrightarrow u=x\vee u\in y\,)\wedge\exists z\forall x\,\neg x\in z$	YES
(4)	$(\forall x,y)\exists v\,\forall u(\,x\leq u\wedge y\leq u\leftrightarrow v\leq u\,)$	NO
(5)	$(\forall x,y)\exists v\,\forall u(\,u\leq x\wedge u\leq y\leftrightarrow u\leq v\,)$	NO
(4a)	$(\forall x,\,y)\exists v\big(\,x\leq v\wedge y\leq v\wedge\forall u(\,x\leq u\wedge y\leq u\rightarrow v\leq u\,)\,\big)$	NO
(5a)	$(\forall x,\,y)\exists v\big(\,v\leq x\wedge v\leq y\wedge\forall u(\,u\leq x\wedge u\leq y\rightarrow u\leq v\,)\,\big)$	NO
(6)	$\mathbf{(4)}\wedge\mathbf{(5)}\wedge\forall x\,x\leq x\wedge(\forall x,y,z)(\,x\leq y\wedge y\leq z\rightarrow x\leq z\,)$ $\wedge(\forall x,y)(\,x\leq y\wedge y\leq x\rightarrow x=y\,)\wedge\exists x\,\forall y\,x\leq y\wedge\exists y\,\forall x\,x\leq y$	NO

Figure 13: Expressibility in set theory and lattice theory

shown to be undecidable in [Kwa81].

What shall we do if a theory is not strong enough to make two conjugated quasi-projections available? Succeeding in expressing in $\mathcal{L}^{\times}$ its axioms, if nothing else, would itself be a gratification; however, the axioms of lattice theory (cf. sentences **(4a)**, **(5a)**, **(6)** of Figure 13) appear to be already beyond the expressive boundaries of $\mathcal{L}^{\times}$.

Our goal, in the next four sections, is to encode in $\mathcal{L}^{\times}$ a satisfactory surrogate of lattice theory by resorting to a device of a semantic, rather than of a syntactic, nature. This device consists in enlarging the domain of discourse. The idea is simply to avoid referring to the domain D,

partially ordered and equipped with infimum and supremum operations; this will be superseded by the domain $\mathbb{1} = \{\emptyset\} \cup \{\{a, b\} : a, b \text{ in } D\}$, whose singletons $\{a\}$ can be identified with the entities a in the original domain D.[5] Unordered pairs are in a sense extraneous to the structure $D, \leq, \mathsf{inf}, \mathsf{sup}$ in which we are truly interested, very much like singletons are replacements for the true entities of discourse; nevertheless, the formal properties to be stated in the following will enable one to reason, by 'metaphor', on the usual lattices. The operations inf e sup shall be treated as monadic operations, hence dyadic relations, over the (enlarged) domain of discourse: an essential condition, this, for their properties to be formalizable in the map calculus.

The characterization of lattices will be carried out so as to get along the way weak but useful deductive contexts. The technique to be exploited can be adapted to any context where each $(n+1)$-ary operation $\star$ (like inf and sup) meets the condition

$$\star(x_0, \ldots, x_n) = \star(x_{\pi_0}, \ldots, x_{\pi_n}) \text{ for any permutation } \pi.$$

8.1 Inclusion theory, within map calculus

Can one instruct a theory of inclusion which does not presuppose a more fundamental theory of membership? By performing this task, one should rediscover something strictly akin to Aristotle's assertory syllogistic.

To make this and the subsequent three sections easier to read, we will develop in parallel a map-based formalization and a first-order formulation of the notions entering into play.

Let us recall, to start with, that the set inclusion relation $\subseteq$ is a partial ordering relation:

$$\subseteq \circ \subseteq \subseteq \subseteq, \qquad \subseteq \cap \supseteq = \iota.$$

Clearly, the former of these condition expresses the transitivity law while the latter combines the reflexivity law $\iota \subseteq \subseteq$ and the antisymmetry law $\subseteq \cap \supseteq \subseteq \iota$.

Preliminary to introducing a counterfeit membership relation (which will, in fact, be a subrelation of inclusion), we now introduce new forms of abbreviations.[6] By putting

$$\mathsf{void} \equiv_{\text{Def}} \subseteq \dagger \emptyset, \qquad \text{i.e.} \qquad V\ \mathsf{void}\ \_ \equiv_{\text{Def}} \forall x\, V \subseteq x,$$

and

$$\mathsf{sngl}\mathsf{OR}\mathsf{void} \equiv_{\text{Def}} (\iota \cup \not\supseteq) \dagger \mathsf{void}, \qquad \mathsf{sngl} \equiv_{\text{Def}} \overline{\mathsf{void}} \cap \mathsf{sngl}\mathsf{OR}\mathsf{void},$$

i.e.

$$S\ \mathsf{sngl}\ \_ \equiv_{\text{Def}} \neg S\ \mathsf{void}\ \_ \wedge \forall x\, (x \subseteq S \rightarrow x = S \vee x\ \mathsf{void}\ \_),$$

one easily verifies in $\mathcal{L}^{\times}$ the right-absoluteness of the predicates 'is void' and 'is singleton':

$$\mathsf{void} \circ \mathbb{1} = \mathsf{void} \dagger \emptyset, \qquad \mathsf{sngl} \circ \mathbb{1} = \mathsf{sngl} \dagger \emptyset.$$

The existence of a sole void is easily proved along the following line: if V and W are void, then either of them is included in the other, hence the two are equal, thanks to antisymmetry. (Of course this argument cannot be mimicked directly in $\mathcal{L}^{\times}$ where variables for V, W are missing.)

Here follows the definition of $\in$ that we set up:

$$\in \equiv_{\text{Def}} \mathsf{sngl} \cap \subseteq, \qquad \text{i.e.} \qquad S \in X \equiv_{\text{Def}} S\ \mathsf{sngl}\ \_ \wedge S \subseteq X.$$

The addition of the following EXTENSIONALITY axiom appears to be mandatory:

$$\supseteq \dagger \notin \subseteq \supseteq, \qquad \text{i.e.} \qquad \forall w (w \in X \rightarrow w \subseteq Y) \rightarrow X \subseteq Y.$$

One easily sees (at least in first-order logic) that if the entities belonging to X and to Y are the same (i.e. $\forall w (w \in X \leftrightarrow w \in Y)$), then X and Y are, by antisymmetry, equal; analogously, if X and Y include the same entities (i.e. $\forall w (w \subseteq X \leftrightarrow w \subseteq Y)$), then extensionality and antisymmetry, taken together, force them to be equal.

It seems convenient to end by postulating the EXISTENCE OF VOID:

$$\mathsf{void} \neq \emptyset, \qquad \text{i.e.} \qquad \exists v \forall y\, v \subseteq y.$$

Analogously we could —but we refrain from this— add postulates like

$$\supseteq \dagger \emptyset \neq \emptyset \qquad (\text{i.e.} \qquad \exists t \forall y\, y \subseteq t);$$

however this assumption that an omni-comprehensive set exists, or similar ones (e.g., assuming

[5] A non-hierarchical domain of sets richer than this, for example the family $\mathcal{P}ow(D)$ of all subsets of D, would also do to the job. This is an idea to which we might resort in the future, in order to cope with the theories of *complete* lattices, and of Boolean algebras.

[6] Certain customary forms of abbreviation are left understood; e.g., $(\forall xPy)\varphi \equiv_{\text{Def}} \forall x (xPy \rightarrow \varphi)$, $(\exists xQy)\psi \equiv_{\text{Def}} \exists x (xQy \wedge \psi)$, and $xPyQz \equiv_{\text{Def}} xPy \wedge yQz$. Moreover, following Prolog, we represent by $\_$ a variable that occurs only once.

that the family of sets to be dealt with enjoys particular closure properties), would be out of scale w.r.t. the limited goals of the present work.[7]

8.2 Selectors and choice functions, within map calculus

We say that σ is a SELECTOR if it enjoys the following properties:

$$\sigma \subseteq \subseteq \cap \mathsf{sngl}\mathsf{OR}\mathsf{void}, \quad \text{i.e.} \quad X\sigma Y \rightarrow X \subseteq Y \wedge (\forall v \subseteq X)(v = X \vee v \ \mathsf{void}\ \_).$$

We then say that a selector is GLOBAL when

$$\overline{\mathsf{void}} \subseteq \sigma^{-1} \circ \mathbb{1} \quad \text{and moreover} \quad \mathsf{void} \cap \sigma \subseteq \mathsf{void}^{-1},$$

i.e.

$$\neg Y \ \mathsf{void}\ \_ \rightarrow \exists x \ x\sigma Y \wedge (\forall x \sigma Y) \neg x \ \mathsf{void}\ \_ .$$

An example of global selector is membership. In a sense this is the greatest of all global selectors σ, each one of which is in fact bound to fulfill the condition

$$\sigma \subseteq \ \in \cup (\ \mathsf{void} \cap \ \mathsf{void}^{-1}\ ).$$

To switch to the opposite extreme, let us consider the global selectors that are in a sense minimal: these are the so-called GLOBAL CHOICE FUNCTIONS. Any such η is characterized by

- being a global selector;
- being a function in its second argument, and, as such
- being total (for this, it suffices to require that $\mathsf{void} \cap \mathsf{void}^{-1} \subseteq \eta$).

One easily sees what is the task of η (cf. Figure 14), namely to extract from any nonnull set one and only one singleton contained in it—furthermore, for the sake of definiteness, we want that η associates to the null set the null set itself.

In order to base a lattice theory on $\mathcal{L}^{\times}$, we need *two* global choice functions, η_1 and η_2, (for our purposes what really counts is that they be defined for all sets with one or two members; however, while singletons have been characterized already, pairs will be defined just through η_1 and η_2). In addition to being global choice functions, η_1 and η_2 shall disagree whenever possible. DISAGREEMENT postulate:

$$\eta_1 \cap \eta_2 \subseteq \iota, \quad \text{i.e.} \quad Y\ \eta_1\ X \wedge Y\ \eta_2\ X \rightarrow X = Y.$$

If we were to treat more than two global choice functions —which would be useless here, as we need to consider sets with two members at most—, the disagreement postulate would take the form shown at the bottom of Figure 14.

η_i	$\subseteq$	$\in \cup (\ \mathsf{void} \cap \mathsf{void}^{-1}\ )$,		i.e.,
$Y \eta_i X$	$\rightarrow$	$Y \in X \vee (\ Y\ \mathsf{void}\ \_ \ \wedge \ X\ \mathsf{void}\ \_\ )$;		
$\eta_i \circ \eta_i^{-1}$	$\subseteq$	ι ,		i.e.,
$Y \eta_i X \wedge Z \eta_i X$	$\rightarrow$	$Z = Y$;		
ι	$\subseteq$	$\eta_i^{-1} \circ \eta_i$,		i.e.,
	$\exists x$	$x \eta_i Y$;		
$\mathbb{1} \circ (\ \eta_i \cap \eta_{i+1}\ )$	$=$	$\emptyset \dagger (\ \notin \cup \eta_1 \cup \cdots \cup \eta_i\ )$,		i.e.,
$\exists y (\ y \eta_i X \wedge y \eta_{i+1} X\ )$	$\leftrightarrow$	$\forall y (\ y \notin X \vee y \eta_1 X \vee \cdots \vee y \eta_i X\ )$.		

Figure 14: Properties of global choice functions $\eta_1, \eta_2, \eta_3, \ldots$

8.3 Unordered pair theory, within map calculus

With the conceptual devices made ready so far, we can instruct a theory of unordered pairs. To the postulates on inclusion we add the following PAIRING postulate:

$$\mathsf{sngl} \cap \mathsf{sngl}^{-1} \subseteq \eta_1 \circ \eta_2^{-1} \cup \eta_2 \circ \eta_1^{-1},$$

[7] On the other hand, something like the following UNION postulate might turn out useful in order to move from lattice theory to the theory of *complete* lattices:

$$\subseteq \circ \supseteq = \mathbb{1}, \quad \text{i.e.} \quad \exists u (\ X \subseteq u \wedge Y \subseteq u).$$

i.e.

$$X\ \mathsf{sngl}\ \_\wedge Y\ \mathsf{sngl}\ \_\to\exists\, w\big(\,(\,X\,\eta_1\,w\wedge Y\,\eta_2\,w\,)\vee(\,X\,\eta_2\,w\wedge Y\,\eta_1\,w\,)\,\big).$$

(With this, η_1^{-1} and η_2^{-1} come close to being conjugated quasi-projections, relative to the subdomain of $\mathcal{U}$ made of all singletons.)

A set devoid of members, or with only one or two, can be characterized as follows in the theory of inclusion enriched with the axioms already seen about η_1 and η_2:

$$\mathsf{dbl}\ \equiv_{\text{Def}}(\ \not\supseteq\cup\,\iota\,)\dagger(\,\mathsf{void}\cup\eta_1\cup\eta_2\,)$$

i.e.

$$W\ \mathsf{dbl}\ X\ \equiv_{\text{Def}}\forall\, z(\,z\subseteq W\neq z\to z\ \mathsf{void}\ \_\vee z\,\eta_1\,X\vee z\,\eta_2\,X\,).$$

At this point we add a weak postulate of UNION, which regards singletons:

$$\eta_1\cap\overline{\mathsf{void}}\subseteq\ \subseteq\circ\big(\,(\ \supseteq\circ\,\eta_2\,)\cap\mathsf{dbl}\,\big),$$

i.e.

$$L\,\eta_1\,X\wedge\neg L\ \mathsf{void}\ \_\to(\,\exists\, w,u\,)(\,L\subseteq w\ \mathsf{dbl}\ X\wedge w\supseteq u\,\eta_2\,X\,).$$

8.4 Lattice theory: an inexpressible outflanked

In sight of a theory of lattices, we introduce now the predicate letters inf and sup, for which we demand in the first place that

- they be (partial) functions in their second argument:
$$\mathsf{inf}\circ\mathsf{inf}^{-1}\subseteq\iota,\qquad\mathsf{sup}\circ\mathsf{sup}^{-1}\subseteq\iota;$$
- they have in their domain all singletons and pairs:
$$\mathsf{dbl}\setminus\mathsf{void}\subseteq\left\{\begin{array}{c}\mathsf{inf}^{-1}\\ \mathsf{sup}^{-1}\end{array}\right\}\circ\mathbb{1},\qquad\text{i.e.}\qquad\neg X\ \mathsf{void}\ \_\wedge X\ \mathsf{dbl}\ \_\to\exists\, y\, y\left\{\begin{array}{c}\mathsf{inf}\\ \mathsf{sup}\end{array}\right\}X;$$
- every image of theirs be a singleton:
$$\mathsf{inf}\cup\mathsf{sup}\subseteq\mathsf{sngl},\qquad\text{i.e.}\qquad Y\ \mathsf{inf}\ X\vee Y\ \mathsf{sup}\ X\to Y\ \mathsf{sngl}\ \_;$$
- they disagree with each other:
$$\mathsf{inf}\cap\mathsf{sup}\subseteq\iota.$$

Nothing very engaging so far: we could take inf and sup to be $\eta_1\setminus\mathsf{void}$ and $\eta_2\setminus\mathsf{void}$. However, here we arrive at the partial ordering relation of a lattice:

$$\leq\ \equiv_{\text{Def}}(\ \in\cap\,\mathsf{inf}\,)\circ(\ \in\cap\mathsf{sup}\,)^{-1},$$

i.e.

$$X\leq Y\ \equiv_{\text{Def}}\exists\, u\,(\,X\in u\ni Y\wedge X\ \mathsf{inf}\ u\ \mathsf{sup}^{-1}\,Y\,).$$

We impose on this relation the partial ordering laws, which we state in the following form:

$$\leq\circ\leq\subseteq\leq,\qquad\leq\cap\geq\ =\mathsf{sngl}\cap\iota.$$

(Therefore $X\ \mathsf{sngl}\ \_\to X\ \leq\ X$, whence, by exploiting the definition of $\leq$ and the disagreement between inf and sup, we will obtain that

$$X\ \mathsf{sngl}\ \_\to X\ \mathsf{inf}\ X\ \mathsf{sup}\ X,$$

to wit that

$$\mathsf{sngl}\cap\iota\subseteq\mathsf{inf}\cap\mathsf{sup},$$

i.e.: *Every singleton is a fixpoint of both* inf *and* sup.)

We conclude with more intricate conditions on inf and sup, which are the ones of MONOTONICITY and of MIN-/MAX-IMALITY. Let us begin with the former two:

$$X\subseteq Y\wedge L\ \mathsf{inf}\ X\wedge M\ \mathsf{inf}\ Y\wedge L\neq M\to(\,\forall\, w,u\,)(\,w\ \mathsf{dbl}\ u\wedge L\in w\ni M\to M\ \mathsf{inf}\ w\,),$$

which is to say

$$\bar{\iota}\cap(\,\mathsf{inf}\circ\supseteq\circ\mathsf{inf}^{-1}\,)\subseteq(\ \notin\cup\,\mathsf{inf}\,)\dagger\big((\,\overline{\mathsf{dbl}}\dagger\emptyset\,)\cup\ \not\ni\ \big),$$

and an analogous condition on sup. From these one gets that the inf of a set is a lower bound for its members ($\mathsf{inf}\circ\ni\subseteq\leq$), while its sup is an upper bound.

As for the latter two conditions, one is:

$$(\ \leq\dagger\notin\ )\circ\mathsf{inf}^{-1}\subseteq\leq,\qquad\text{i.e.}\qquad Y\ \mathsf{inf}\ W\wedge(\,\forall\, z\in W\,)X\leq z\to X\leq Y.$$

This states that when the operation inf is applied to a pair or singleton W, or when it anyway associates an image to W, it produces the greatest lower bound of the members of W as it should. Analoguously one characterizes the minimality of the upper bound produced by sup.

Let us now put together the various pieces of lattice theory developed so far. The map letters that we are regarding as PRIMITIVE are $\subseteq$, η_1, η_2, inf, sup; the DERIVED ones are introduced via the following abbreviations:

$\geq$	$\equiv_{\mathrm{Def}}$	$\leq^{-1}$	$\supseteq \equiv_{\mathrm{Def}} \subseteq^{-1}$
$\not\subseteq$	$\equiv_{\mathrm{Def}}$	$\overline{\subseteq}$	$\not\supseteq \equiv_{\mathrm{Def}} \overline{\supseteq}$
void	$\equiv_{\mathrm{Def}}$	$\subseteq \dagger \emptyset$	$\mathsf{sngIORvoid} \equiv_{\mathrm{Def}} (\iota \cup \not\supseteq) \dagger \mathsf{void}$
sngl	$\equiv_{\mathrm{Def}}$	$\mathsf{sngIORvoid} \setminus \mathsf{void}$	
$\in$	$\equiv_{\mathrm{Def}}$	$\mathsf{sngl} \cap \subseteq$	$\ni \equiv_{\mathrm{Def}} \in^{-1}$
$\notin$	$\equiv_{\mathrm{Def}}$	$\overline{\in}$	$\not\ni \equiv_{\mathrm{Def}} \overline{\ni}$
dbl	$\equiv_{\mathrm{Def}}$	$(\not\supseteq \cup \iota) \dagger (\mathsf{void} \cup \eta_1 \cup \eta_2)$	$\leq \equiv_{\mathrm{Def}} (\in \cap \mathsf{inf}) \circ (\in \cap \mathsf{sup})^{-1}$

In the AXIOMS listed in Figure 15, through which we have meant to characterize the LATTICE structure, η stands in turn for η_1 and for η_2.

$\subseteq \circ \subseteq \subseteq \subseteq$	$\subseteq \cap \supseteq = \iota$
$\not\ni \dagger \subseteq \subseteq \subseteq$	$\mathsf{void} \neq \emptyset$
$\eta \subseteq \subseteq \cap \mathsf{sngIORvoid}$	
$\overline{\mathsf{void}} \subseteq \eta^{-1} \circ \mathbb{1}$	$\mathsf{void} \cap \eta \subseteq \mathsf{void}^{-1}$
$\eta \circ \eta^{-1} \subseteq \iota$	$\mathsf{void} \cap \mathsf{void}^{-1} \subseteq \eta$
$\eta_1 \cap \eta_2 \subseteq \iota$	
$\mathsf{sngl} \cap \mathsf{sngl}^{-1} \subseteq \eta_1 \circ \eta_2^{-1} \cup \eta_2 \circ \eta_1^{-1}$	
$\eta_1 \setminus \mathsf{void} \subseteq \subseteq \circ\big((\supseteq \circ \eta_2) \cap \mathsf{dbl}\big)$	
$\mathsf{inf} \circ \mathsf{inf}^{-1} \subseteq \iota$	$\mathsf{sup} \circ \mathsf{sup}^{-1} \subseteq \iota$
$\mathsf{dbl} \setminus \mathsf{void} \subseteq \mathsf{inf}^{-1} \circ \mathbb{1}$	$\mathsf{dbl} \setminus \mathsf{void} \subseteq \mathsf{sup}^{-1} \circ \mathbb{1}$
$\mathsf{inf} \cup \mathsf{sup} \subseteq \mathsf{sngl}$	$\mathsf{inf} \cap \mathsf{sup} \subseteq \iota$
$\leq \circ \leq \subseteq \leq$	$\leq \cap \geq = \mathsf{sngl} \cap \iota$
$\bar{\iota} \cap (\mathsf{inf} \circ \supseteq \circ \mathsf{inf}^{-1}) \subseteq (\notin \cup \mathsf{inf}) \dagger \big((\overline{\mathsf{dbl}} \dagger \emptyset) \cup \not\ni\big)$	
$\bar{\iota} \cap (\mathsf{sup} \circ \supseteq \circ \mathsf{sup}^{-1}) \subseteq (\notin \cup \mathsf{sup}) \dagger \big((\overline{\mathsf{dbl}} \dagger \emptyset) \cup \not\ni\big)$	
$(\leq \dagger \notin) \circ \mathsf{inf}^{-1} \subseteq \leq$	$\mathsf{sup} \circ (\not\ni \dagger \leq) \subseteq \leq$

Figure 15: Map formulation of lattice theory

9 Towards the implementation of a basic map reasoning layer

As stated in the introduction and stressed again at the end of Sec.4, the authors have in mind to bring various techniques and tools useful to map reasoning inside a platform for computational logic. In sight of this, the following section addresses an issue of a most fundamental character presupposed by any effective map-expression simplifier: What would be a suitable internal representation of expressions and formulas?

Then, in the subsequent section, we speculate on plausible interactions between research on the automation of map reasoning and research on decidable fragments of set theory.

9.1 Clues for a representation of map expressions and formulas

It has become customary since the early sixties to represent formulas in conjunctive normal form as sets of sets of literals, and the same treatment can, of course, be given to disjunctive normal forms. The rationale for employing sets in the representation of expressions of some kind holds whenever one is to deal with an associative-commutative binary operator, $\star$, which enjoys the idempotency law $x \star x = x$, or something similar, e.g. a rule of the form $x \star x \star y = \bullet y$ (with $\bullet$ designating a monadic operation). Hence this idea applies, in the contexts of map calculus and of $\mathcal{L}^+$, to all of the constructs $\cap, \triangle, \cup, \wedge, \vee, \leftrightarrow$, and $\not\leftrightarrow$.

Another useful idea is that every map expression P has a certain POLARITY; although conventions on this may widely vary, it looks reasonable to regard, e.g., as

positive: all expressions of the forms

$$\mathsf{p}_i, \quad \mathsf{p}_i^{-1}, \quad Q \cap R, \quad Q \circ R, \quad \diamond P, \quad \mathbb{1}, \text{ etc.;}$$

negative: all expressions of the forms

$$\overline{\mathsf{p}}_i, \quad \overline{\mathsf{p}_i^{-1}}, \quad \overline{\mathsf{p}}_i^{-1}, \quad Q \cup R, \quad Q \dagger R, \text{ etc.}$$

The same idea can be extended to formulas: if one —say— chooses to regard conjunctions, exclusive disjunctions, and existentially quantified formulas as positive, then one should regard inclusive disjunctions, bi-implications, and universally quantified formulas as negative; and so on.

There is a single point where symmetry breaks off, namely at the formation of atomic formulas. We see no good reasons to associate opposite polarities to $Q{\neq}R$ and to $Q{=}R$; rather, it seems convenient to standardize these as $\diamond(\, Q \mathbin{\triangle} R\,){=}\mathbb{1}$ and $\overline{\diamond(\, Q \mathbin{\triangle} R\,){=}\mathbb{1}}$ respectively. The expressions $\diamond(\, Q \mathbin{\triangle} R\,)$ and $\overline{\diamond(\, Q \mathbin{\triangle} R\,)}$ will, of course, have opposite polarities, but their polarities do not propagate to the respective map equalities. Likewise, we refrain from considering the polarity of an atom of the form $x\overline{Q}y$ as being opposite to the polarity of an atom of the form uRv, whatever the polarities of Q and R may be.

Broadly speaking, the effectiveness of a symbolic method of simplification rests on the choice of a convenient *internal representation* for the expressions to be treated. In the following, we indicate with $\langle P\rangle$ the internal representation of an expression of which P denotes the external (or 'concrete') representation.

The ideas in the preceding paragraphs lead to the internal representation of a multiple intersection $Q_1 \cap \cdots \cap Q_m$ (where no Q_j is of the form $R_j \cap S_j$) as a single operator $\bigcap$ applied to two arguments:

$$\langle Q_1 \cap \cdots \cap Q_m\rangle \;\equiv_{\text{Def}}\; \bigcap(\, \{\langle P_1\rangle, \ldots, \langle P_h\rangle\}, \{\langle N_1\rangle, \ldots, \langle N_k\rangle\}\,),$$

where $\{\langle P_1\rangle, \ldots, \langle P_h\rangle, \langle \overline{N}_1\rangle, \ldots, \langle \overline{N}_k\rangle\} = \{\langle Q_1\rangle, \ldots, \langle Q_m\rangle\}$, and the P_is and the N_ℓs all have positive polarity. Symmetrically,

$$\langle Q_1 \cup \cdots \cup Q_m\rangle \;\equiv_{\text{Def}}\; \bigcup(\, \{\langle N_1\rangle, \ldots, \langle N_k\rangle\}, \{\langle P_1\rangle, \ldots, \langle P_h\rangle\}\,),$$

where the Q_js are related to P_is and N_ℓs in the same manner as before.

The representation of a map expression $Q_1 \mathbin{\triangle} \cdots \mathbin{\triangle} Q_m$ is similar but simpler. For $j = 1, \ldots, m$, let P_j be of positive polarity and suitably chosen in order that either $P_j \equiv Q_j$ or $\vdash^{\times} P_j{=}\overline{Q}_j$. Moreover, let

$$\langle Q_1 \mathbin{\triangle} \cdots \mathbin{\triangle} Q_m\rangle \;\equiv_{\text{Def}}\; \begin{cases} \Sigma(\, \{\langle P_1\rangle, \ldots, \langle P_m\rangle\}\,) & \text{if the number of } j\text{s for which } P_j \equiv Q_j \text{ is even,} \\ \overline{\Sigma}(\, \{\langle P_1\rangle, \ldots, \langle P_m\rangle\}\,) & \text{otherwise.} \end{cases}$$

Let us make it clear that, in the above, $\bigcap$, $\bigcup$, Σ, and $\overline{\Sigma}$ are operators used only in the internal representation of expressions, and that Σ and $\overline{\Sigma}$ (in analogy to $\bigcap$ and $\bigcup$) carry opposite polarities. Insisting on an exact correspondence between external and internal operators would only make an obstacle to the ease of internal standardization of expressions, leading to a more cumbersome map simplification process.

Once the design choice is made of having a harmless discrepancy between the signature that supports the internal representation and the one used in the external representation, it also becomes acceptable to introduce a number of combined operators, e.g. $\overset{\iota}{\bigcap}$ and $\overset{\overline{\iota}}{\bigcup}$. Internally $\overset{\iota}{\bigcap}(\, \{\langle P_1\rangle, \ldots, \langle P_h\rangle\}, \{\langle N_1\rangle, \ldots, \langle N_k\rangle\}\,)$ will represent $(\, \iota \cap P_1 \cap \cdots \cap P_h\,) \setminus (\, N_1 \cup \cdots \cup N_k\,)$; therefore $\overset{\iota}{\bigcap}$ will —together with $\emptyset$, $\mathbb{1}$, ι, with its dual operator $\overset{\overline{\iota}}{\bigcup}$, and with a few other internal constructs— represent a self-inverse operator, i.e., it will produce map expressions R for which $\vdash^{\times} R^{-1}{=}R$.

In connection with the Peircean operators $\circ$ and $\dagger$, the internal representation can again deviate from the external one; however, since these operators are not commutative, sets cannot be of great help here.

A possibility is to exploit an operator

$$\diamond\big(\, [\,[\langle P_{00}\rangle, \ldots, \langle P_{0n_0}\rangle], \ldots, [\langle P_{h0}\rangle, \ldots, \langle P_{hn_h}\rangle]\,]\,\big)$$

to represent

$$\diamond(\, P_{00} \circ \cdots \circ P_{0n_0} \circ \mathbb{1} \circ P_{10} \circ \cdots \circ P_{1n_1} \circ \mathbb{1} \circ \cdots \circ \mathbb{1} \circ P_{h0} \circ \cdots \circ P_{hn_h}\,),$$

an operator dual to this to represent

$$\emptyset \dagger (\, \overline{P}_{00} \dagger \cdots \dagger \overline{P}_{0n_0} \dagger \emptyset \dagger \overline{P}_{10} \dagger, \cdots \dagger \overline{P}_{1n_1} \dagger \emptyset \dagger \cdots \dagger \emptyset \dagger \overline{P}_{h0} \dagger \cdots \dagger \overline{P}_{hn_h} \,) \dagger \emptyset,$$

and four additional internal operators to represent left-absolute expressions of the forms $\mathbb{1} \circ P_{00} \circ \cdots \circ P_{hn_h}$, $\_\ \emptyset \dagger \overline{P}_{00} \dagger \cdots \dagger \overline{P}_{hn_h}$ and right-absolute expressions of the forms $P_{00} \circ \cdots \circ P_{hn_h} \circ \mathbb{1}$, $\overline{P}_{00} \dagger \cdots \dagger \overline{P}_{hn_h} \dagger \emptyset$.

At the connective level, a convenient representation of conjunctions and disjunctions is as

$$\bigwedge \big(\{[n_1, \sigma_1], \ldots, [n_h, \sigma_h], [-m_1, \tau_1], \ldots, [-m_k, \tau_k]\}, \quad \{\bigvee (\, D_1, C_1 \,), \ldots, \bigvee (\, D_r, C_r \,)\} \big)$$

and, dually, as

$$\bigvee \big(\{[m_1, \tau_1], \ldots, [m_k, \tau_k], [-n_1, \sigma_1], \ldots, [-n_h, \sigma_h]\}, \quad \{\bigwedge (\, C_1, D_1 \,), \ldots, \bigwedge (\, C_r, D_r \,)\} \big).$$

Here n_is, m_js and r are natural numbers and the n_is (respectively, the m_js) are pairwise distinct; moreover $h + k + r > 1$ and every σ_i or τ_j is a function defined on a finite set of natural numbers which sends them into predicates of positive polarity. The meaning of $\sigma_i(s) = P$ is $P{=}\mathbb{1}$ when $n_i = s = 0$, it is $\mathsf{v}_s P \mathsf{v}_s$ when $n_i = 0$ and $s \neq 0$, and is $\mathsf{v}_{n_i} P \mathsf{v}_s$ when $n_i \neq 0$ (in which case $n_i < s$ must hold); likewise, $\tau_j(t) = N$ means either $N{=}\emptyset$ or $\mathsf{v}_t \overline{N} \mathsf{v}_t$ or $\mathsf{v}_{m_j} \overline{N} \mathsf{v}_t$.

If we now reconsider the graph-thinning algorithm in Sec.5.1, it should be clear that its aim is to apply a bunch of existential quantifiers to a multiple conjunction represented as above (with $r = 0$), in an effort to eliminate those quantifiers by suitably restructuring the conjunction. The very same algorithm can, of course, succeed in eliminating universal quantifiers placed in front of a disjunction of atoms. The import of that algorithm should be broadened in the future, in particular by removing the restriction that $r = 0$.

9.2 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 [CFO89]); 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 we have seen in Sec.7.2, 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.

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 16, 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 16: Constructs of the multilevel syllogistic language

This decision problem, which was first shown to be solvable in [FOS80], is easily reduced to the satisfiability problem for conjunctions of literals of the following forms (where x, y, u stand for individual variables):

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

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 17.

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

Figure 17: 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 already seen in Figure 8, to the effect that

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

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, $\bar{\iota}$, $\in$, $\overline{\ni \circ \in}$, $\not\ni$ $\dagger \in$, etc. Now the question arises naturally: To what extent, and by what criteria, could we broaden the collection of admitted set combinators, without losing decidability or leaving the complexity class of *MLSS* (whose decision problem is *NP*-complete)?

10 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 dyadic 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.7) and translating Datalog into relational algebra (cf. [Ull89]), 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 readmitted into play.

All of this would, of course, impose a redesign of the deductive apparatus proposed for $\mathcal{L}^\times$ (cf. Sec.3), 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 [TG87] 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, in Sec.2 we have preferred safe characterizations of $\cup$ and $\setminus$ to simpler ones such as would have been: $P \cup Q \equiv_{\text{Def}} \overline{\overline{P} \cap \overline{Q}}$ and $P \setminus Q \equiv_{\text{Def}} P \cap \overline{Q}$.

10.1 Safe map expression recognition

Definition. *An interpretation $\mathfrak{S}$ is said to be* SAFE *if it assigns maps of finite cardinality to the map letters.*

A map expression S is said to be SAFE *if its value $S^{\mathfrak{S}}$ is* necessarily *finite, i.e., finite in any safe interpretation $\mathfrak{S}$.* □

(Accordingly, a map expression P is said to be *unsafe* if it is *possibly* infinite, i.e., $P^{\mathfrak{S}}$ is infinite in some safe interpretation $\mathfrak{S}$.)

In connection with the goals just explained, one would like to be able to determine which map expressions are safe, and which ones are not. Success can be achieved, but only in special subcases, i.e., by placing syntactic restrictions on the form of the map expressions to be tested for safeness. In general (if only for efficiency purposes), one may have to resort to criteria which occasionally classify as unsafe an expression which is actually safe. This makes it sensible to define effective surrogates of the safeness notion:

Definition. *Let $\mathcal{S}$ be an algorithm which gives a yes/no answer to any P drawn from a collection $\mathcal{F}$ of map expressions, so as to ensure that P is safe whenever $\mathcal{S}(P) = \mathtt{yes}$.*

When $\mathcal{S}(S) = \mathtt{yes}$, one says that S is DERIVABLY SAFE *w.r.t. $\mathcal{S}$.* □

A simple-minded example is the following.

Example. Take

- $\mathcal{F}_0$ to be the collection of all map expressions that do not involve any of the Peircean constructs (which are $\circ, ^{-1}, \iota$; in essence we are considering ordinary Boolean expressions, interpreting their letters as sets of pairs);
- $\mathcal{S}_0(P)$ to be a sound and complete unsatisfiability test for the inequality $P \neq \emptyset$, where P belongs to $\mathcal{F}_0$.

It should be apparent that, w.r.t. $\mathcal{S}_0$, the derivably safe map expressions correspond to unsatisfiable propositional sentences, with $\cap, \triangle, \emptyset, \mathbb{1}$ playing the rôles of conjunction, exclusive disjunction, false, and true. □

We have thus reduced a subproblem of safeness to 3-SAT (cf. [GJ79]), possibly at the price of missing many safe expressions; there is no evidence, indeed, that $\mathcal{S}_0(S) = \mathtt{yes}$ when S is safe, though the opposite implication obviously holds.

An *absolute* safeness test for the same 'Boolean' collection of expressions we have just considered can be based on the version of two-level syllogistic described in [CFO89], pp.192–194. More directly:

Example. Let

- $\mathcal{F}_1 = \mathcal{F}_0$, where $\mathcal{F}_0$ is as in the preceding example;
- $\mathcal{S}_1(P)$ work on any P belonging to $\mathcal{F}_1$ as follows:
  A letter $\mathtt{p}_f$ not occurring in P is chosen; for every letter $\mathtt{p}_i$ in P, one imposes that $\mathtt{p}_i \subseteq \mathtt{p}_f$; moreover, one imposes that $P \subseteq \mathtt{p}_f$.
  By merely Boolean techniques, $\mathcal{S}_1(P)$ establishes whether or not the conjunction of all these inclusions entails $\mathtt{p}_f = \mathbb{1}$, and, accordingly, answers negatively or affirmatively.

□

A detailed analysis of the approach in the latter example shows that not only we have a sound and complete safeness test (though limited to the collection $\mathcal{F}_1$ of expressions), but that we can make $P^{\mathfrak{S}} = \emptyset$ when $\mathcal{S}_1(P)$ gives a `yes` answer, simply by interpreting all map letters as $\emptyset$.

Preliminary to seeing another line of attack to the safeness problem, we make an easy observation:

Lemma. Let P be any map expression; moreover, let

$$\mathcal{I}_P^{\mathfrak{S}}(a) =_{\text{Def}} \{b \mid [a,b] \in P^{\mathfrak{S}}\}, \qquad \mathcal{D}_P^{\mathfrak{S}}(b) =_{\text{Def}} \{a \mid [a,b] \in P^{\mathfrak{S}}\},$$

and

$$\mathcal{I}_P^{\Im} =_{\text{Def}} \bigcup_{c \text{ in } \mathcal{U}} \mathcal{I}_P^{\Im}(c), \qquad \mathcal{D}_P^{\Im} =_{\text{Def}} \bigcup_{c \text{ in } \mathcal{U}} \mathcal{D}_P^{\Im}(c),$$

where $\Im$, $\mathcal{U}$ are an interpretation and its domain, and a,b are elements of $\mathcal{U}$.

In a safe interpretation $\Im$ it holds that:

1) the set $\mathcal{I}_P^{\Im}(a)$, as well as the set $\mathcal{D}_P^{\Im}(b)$, is either finite or has a finite complement w.r.t. $\mathcal{U}$ (in short, it is *cofinite*); consequently,

2) the set $\mathcal{I}_P^{\Im}$, as well as the set $\mathcal{D}_P^{\Im}$, is either finite or cofinite. □

Let us now consider the collection $\mathcal{F}_2$ consisting of all map expressions that do not involve $\triangle$.

We subdivide $\mathcal{F}_2$ into six equivalence classes, with representative elements $\emptyset, s, \iota, s \circ \mathbb{1}, \mathbb{1} \circ s, \mathbb{1}$ (here s is a short for p_1):

$\emptyset$ represents the class consisting of all expressions whose value is necessarily $\emptyset$;

s represents the class of those expressions whose value is necessarily finite but not necessarily $\emptyset$ (among them, all map letters 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 $\mathcal{U}$, 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 the class of all expressions whose value has a possibly finite complement in $\mathcal{U}^2$.

It will turn out from the ongoing that, in $\mathcal{F}_2$, 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, with operations analogous to those in $\mathcal{F}_2$. The laws of this algebra are shown in Figure 18. 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. It is easily seen that P is safe if and only if its type, so determined, turns out to be either $\emptyset$ or s.

$(A \star B) \star C = A \star (B \star C)$	
$A \star A = A$	
$\emptyset \star A = A \star \emptyset = \emptyset$	$\star \in \{\circ, \cap\}$
$A \cap B = B \cap A$	
$s \cap D = s$ $\qquad$ $\mathbb{1} \cap A = A$	$D \not\equiv \emptyset$
$\iota \cap (\mathbb{1} \circ s) = (\mathbb{1} \circ s) \cap \iota = s$	
$(\mathbb{1} \circ s) \cap (s \circ \mathbb{1}) = s$	
$\iota \circ A = A \circ \iota = A$	
$s \circ \mathbb{1} \circ s = s$	
$\mathbb{1} \circ s \circ \mathbb{1} = \mathbb{1}$	
$(A \circ B)^{-1} = B^{-1} \circ A^{-1}$	
$s^{-1} = s$ $\qquad$ $\iota^{-1} = \iota$	
$\emptyset^{-1} = \emptyset$ $\qquad$ $\mathbb{1}^{-1} = \mathbb{1}$	

Figure 18: Abstract algebraic interpretation of $\mathcal{F}_2$ for safeness determination

In order to treat $\triangle$ together with the constructs of $\mathcal{F}_2$, as regards the safeness problem, one may need a combination of the two lines of attack discussed above.

To see why, consider the expression $\big(\,\mathbb{1}\circ\big(\,(\,\mathsf{p}_1\cap\iota\,)\setminus(\,\mathsf{p}_2\cap\iota\,)\,\big)\big)\cap\big(\,\mathbb{1}\circ(\,\mathsf{p}_2\cap\iota\,)\,\big)$, where $P\setminus Q$ is a short for $P\cap(\,Q\triangle P\,)$ as before. Of course this designates $\emptyset$, and hence should be regarded as safe, but it would be typed as unsafe (specifically, it would be assigned to the class $\mathbb{1}\circ s$) if one were to enhance the rules of Figure 18 with the new rule $s\triangle s=s$ —as it seems, at first glance, reasonable to do. The point is that disjoint *union types* such as $s\cup\emptyset$ would be needed for a fine safeness analysis as soon as $\triangle$ enters into play; in particular rules such as $A\star A=A\cup\emptyset$ would be more accurate than the (formerly adequate) $A\star A=A$, when $\star\in\{\cap,\triangle\}$. Difficulties arising from the presence of $\triangle$ can be circumvented in favorable cases by ad hoc algorithms that can decide whether a specific expression necessarily bears the value $\emptyset$ (cf. the first example in this section).

11 Conclusions

The language of the map calculus, when properly used, may lead to terse specifications and even to a certain 'knowledge compression' with respect to quantified logic (suffice it to consider the characterization of bisimulations in Sec.2). Nevertheless it appears to be overly machine-oriented and a bit too awkward to attract human beings. This is why one should invest a certain amount of effort in developing programs and techniques for an effective translation of first-order theories into the map language. Then, and only then, a fair comparison between formalisms can be carried out.

At any rate, this paper does not claim that the map calculus should substitute predicate logic in the practice of automated theorem-proving. Even at the machine level, predicate logic has a rich battery of refinements of resolution to offer, and it would make no sense to discard the results of over thirty-five years of experience. A primary reason to be interested in the map calculus is that it bridges first-order predicate reasoning without equality with purely equational reasoning, which also has a long history, a rich field of research, and a number of successes.

An equational formulation of the theory of successor (as the one seen in Figure 1) may not be rewarding in any sense, due to the very elementary character of the theory itself. However, the situation might change with the Peano arithmetic or with theories of sets (cf. Figures 7 and 10); here the challenging nature of principles such as the arithmetic induction scheme or the subset axiom scheme may justify the overhead of the axioms on relation algebras for an approach alternative to resolution.

As far as we know, this paper provides the first detailed formulation, in the map language, of several fundamental theories. The strength of these theories varies greatly—we have examined, in fact, very simple theories such as the ones on flat set inclusion and on the successor, and comparatively intricate ones, including a full-blown theory of nested lists and a theory of finite hierarchical sets with (or without) urelements.

For the time being we only have a few experiments to report about —it took quite a while to find a collection of logical axioms for $\mathcal{L}^\times$ that could drive satisfactorily a standard general-purpose theorem prover; moreover, the implementation of a platform of techniques dealing directly with map expressions and formulas is still under development—. The initial effort having now been done of working out the map formulation of a number of interesting theories, we can today start a rich variety of novel experiments that may lead to a precise assessment of the usability of the map calculus and, we hope, to a recognition of its paradigmatic value.

Acknowledgments

We are indebted to IASI (*Istituto di Analisi dei Sistemi ed Informatica*, CNR) and to DIS (*Dept. of Computer and Systems Science*, Univ. "La Sapienza" of Rome), for promoting exchanges of ideas between authors, to Vuokko-Helena Caseiro for careful revision of the manuscript, to Paul Broome for first bringing [TG87] to our attention during the ICLP conference held in Budapest

in 1993. Fabiola Aureli contributed to the design of a basic map reasoning layer in SETL, and Candida Attanasio implemented part of it in Java. Emidio Silvestri paved the way to this research with a Prolog implementation of a logic-to-mapalgebra translator. With Domenico Cantone and Alessandra Cavarra we enjoyed very pleasant summer conversations which led to discoveries on how to improve the translator; they contributed to the design of the graph-thinning algorithm and determined its complexity.

References

[Acz88] P. Aczel. *Non-Well-Founded Sets*. CSLI Lecture Notes, vol. 14, Stanford, 1988.

[AH91] 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.

[AOT98] E. Aureli, E. G. Omodeo, and M. Temperini. Map calculus: Initial application scenarios and experiments based on Otter. Technical Report R.499, IASI-CNR, May 1998.

[BL94] P. Broome and J. Lipton. Combinatory logic programming: computing in relation calculi. In M. Bruynooghe editor, *Proc. of the 1994 International Symposium on Logic Programming*, The MIT Press, pages 269–285, 1994.

[Bor96] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence Journal*, to appear.

[Cal96] D. Calvanese. *Unrestricted and finite model reasoning in class-based representation formalisms*. PhD thesis, Università degli Studi di Roma "La Sapienza", 1996.

[CCO97] 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, *Proc. of the FTP97 Int. workshop on first-order theorem proving*, RISC-Linz Report Series No.97-50, pages 45–52, 1997.

[CFO89] D. Cantone, A. Ferro, and E. G. Omodeo. *Computable Set Theory. Vol. 1.* Oxford University Press, Int. Series of Monographs on Computer Science, 1989.

[End72] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York and London, 1972.

[FOS80] 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.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability. A guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, Books in the Mathematical Sciences, 1979.

[GKV97] 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.

[GOS97] E. Grädel, M. Otto, and E. Rosen. Undecidability results on two-variable logics. *14th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 1200, Springer-Verlag, pages 249–260, 1997.

[Hun33] E. V. Huntington. New sets of independent postulates for the algebra of logic, with special reference to Whitehead and Russell's Principia Mathematica. A correction. *Transactions of the American Mathematical Society*, 35:274–304,557–558, 1933.

[Jac51] N. Jacobson. *Lectures in abstract algebra, vol.1 – Basic concepts.* D. Van Nostrand Company, Inc. The university series in higher mathematics, 1951.

[KP95] J. P. Keller and R. Paige. Program derivation with verified transformations - A case study. *Comm. Pure App. Math.*, 48(9-10):1053–1113, 1995.

[Kwa81] M. K. Kwatinetz. *Problems of expressibility in finite languages.* PhD thesis, University of California, Berkeley, 1981.

[McC95] W. W. McCune. Otter 3.0 Reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, 1994. (Revision A, august 1995).

[McC97] W. W. McCune. Solution of the Robbins Problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[Men79] E. Mendelson. *Introduction to Mathematical Logic.* Van Nostrand, New York, second edition, 1979.

[Omo97] E. G. Omodeo. Specifiche formali di proprietà di relazioni: esempi. Technical Report 36-97, Dip. Informatica e Sistemistica, Università *La Sapienza* di Roma, 1997.

[PP91] F. Parlamento and A. Policriti. Expressing Infinity without Foundation. *Journal of Symbolic Logic*, 56(4):1230–1235, 1991.

[PS95] A. Policriti and J. T. Schwartz. T-Theorem Proving I. *Journal of Symbolic Computation*, 20:315–342, 1995.

[Qui63] W. V. Quine. *Set theory and its logic.* The Belknap Press of Harvard University Press, Cambridge, Massachussetts, revised edition, 3^{rd} printing, 1971.

[Sch95] E. Schröder. *Vorlesungen über die Algebra der Logik (exakte Logik).* B. Teubner, Leipzig, 1891-1895. [Reprinted by Chelsea Publishing Co., New York, 1966.]

[SDDS86] 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.

[Sho67] J. R. Shoenfield. *Mathematical Logic.* Addison Wesley, 1967.

[Tar24] A. Tarski. Sur les ensembles fini. *Fundamenta Mathematicae*, 6:45–95, 1924.

[Tar41] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.

[TG87] A. Tarski and S. Givant. *A formalization of Set Theory without variables*, volume 41 of *Colloquium Publications.* American Mathematical Society, 1987.

[Ull89] J. D. Ullman. *Database and Knowledge-base Systems, vol.2*, volume 50 of *Principles of Computer Science.* Computer Science Press, Stanford University, 1989.

[Win90] S. Winker. Robbins algebra: conditions that make a near-Boolean algebra Boolean. *Journal of Automated Reasoning*, 6(4):465–489, 1990.

[WR10] A. N. Whitehead and B. Russell. *Principia Mathematica.* Cambridge University Press, 1910. Reprinted 1980.