Proving Implications by Algebraic Approximation

Michael Codish Grigory Mashevitzky

Department of Mathematics and Computer Science
Ben-Gurion University of the Negev
PoB 653, Beer-Sheba
84105, Israel
{codish,gmash}@bengus.bgu.ac.il

Abstract

This paper applies techniques of algebraic approximation to provide effective algorithms to determine the validity of universally quantified implications over lattice structures. We generalize the known result which states that any semi-lattice is approximated in the two element lattice. We show that the validity of a universally quantified implication ψ over a possibly infinite domain can be determined by examining its validity over a simpler domain the size of which is related to the number of constants in ψ. Both the known as well as the new results have high potential in providing practical automated techniques in various areas of application in computer science.

1 Introduction

This paper applies techniques of algebraic approximation to provide effective algorithms to determine the validity of universally quantified implications between identities over lattice structures. Approximation is one of the main methods in the scientific process and is common practice also in various branches of mathematics and computer science. The basic idea in approximating one (mathematical) structure by another is that properties of the objects from a perhaps complex structure may be investigated by examination of a simpler structure which preserves some important features of the first structure. The methods of approximation in algebra are developed in the works of Maĺcev [19, 21] and Birkhoff [3, 4]. In computer science, approximation is applied in the context of semantic based program analysis as formalized in terms of Galois connections [26, 22] by Cousot and Cousot [12].

Given a universally quantified formula ψ of the form $(\forall)$ $(p \rightarrow q)$ and a domain D which is a structure which obeys the axioms of some theory (e.g. set theory, propositional logic, group theory, etc.), we are familiar with two general approaches which can be applied when trying to determine if ψ is true in D. The *proof theoretic*

approach, in which a suitable system of axioms is used to reduce ψ to some solved form; and the *model theoretic* approach in which interpretations are enumerated and every interpretation which satisfies p is checked to determine if it satisfies also q. Both approaches are, in general, computationally hard, even in the case of a finite domain. The proof theoretic approach generally seeks a proof in an exponential search space. The number of interpretations examined in a naive model theoretic approach may grow exponentially with the number of variables occurring in the formulae. Of course both approaches may also support efficient algorithms. For example when the search is guided by a specific strategy in the proof theoretic approach or when it is not necessary to examine all interpretations in the model theoretic approach. From a programmer's perspective, the model theoretic approach is attractive because it is straightforward in almost any programming language to enumerate interpretations. Examples 1 and 3 below illustrate an almost trivial implementation in Prolog. On the other hand, the proof theoretic approach is attractive because it enables the consideration of search strategies and heuristics.

To show that an implication ψ is valid in a given class of algebraic structures $\mathcal{K}$, we do not need to check all substitutions of variables in ψ to elements of these structures or for all structures in $\mathcal{K}$. It follows from results of algebraic approximation [4, 21], that it is sufficient to check the validity of ψ for the so called subdirectly irreducible structures which generate the given ones and which are much less complicated. For example, the only subdirectly irreducible Boolean algebra is the 2-element Boolean algebra. Often the required subdirectly irreducible structures are substructures of members of $\mathcal{K}$ and in this case the validity of a ψ in $\mathcal{K}$ is exactly defined by its validity in the corresponding subdirectly irreducible structures. In other words *ψ is valid in $\mathcal{K}$ if and only if it is valid in the corresponding subdirectly irreducible structures.*

We know the description of subdirectly irreducible structures for many subclasses of classical algebraic structures: semigroups, groups, lattices and so on (see for example [4, 14, 27]). But if we want to apply these results to determine the validity of implications for specific structures which contain constants (the names of elements of the structure), we run into problems. In the classical algebraic theories, formulae do not explicitly reference arbitrary elements of the underlying structures. For example, in lattice theory, we may mention only the top or bottom elements of a given lattice, but we do not usually refer explicitly to other elements. The signature consists of a few special symbols. E.g., in the case of lattices: $\bot, \top, \sqcup, \sqcap$.

In contrast, when proving theorems in the context of specific applications, it is common to refer explicitly to arbitrary elements of the underlying structure. This means that we want to prove properties of a given structure and not necessarily about the general class of objects of that structure. For example, consider a lattice of types in the context of a programming language with type declarations. In this context it is natural to ask if a given program is well-typed with respect to a type declaration. Namely, if particular program constructs belong to corresponding declared types in the specific underlying lattice of types. Consequently, we may question the validity of an implication which contains explicit reference to elements of the underlying lattice of types. To express or validate theorems which refer to explicit constants from the underlying structure we must enrich the signature of our language. This is done by including the names of constants as additional 0-ary functions in the signature. Hence,

the models constructed for the signature have marked elements which correspond to the constants of the signature. In particular, if we wish to discuss lattice structures with explicit constants, we are no longer within the classic lattice theory. Instead, we are dealing with a structure consisting of *marked elements*. It is worthwhile to notice, that the addition of even one constant to the signature can change crucially the properties of its models. The set of quasi-identities of any non-trivial Boolean algebra coincides with the set of quasi-identities of the class of all Boolean algebras. Of course this property is not true for the Boolean algebra with marked elements. We shall give some additional examples of this in the following.

Note that to determine the validity of a given formula we do not need to consider all of the elements of the underlying domain as marked elements. It is sufficient to consider as marked only those elements which correspond to the constant symbols in the given formula.

In this paper we investigate a practical approach to determine the truth of implications in a (possibly infinite) semilattice or distributive lattice D. For any such formula ψ with less than three constants, we identify a minimal class of objects $\mathcal{K}$ such that the validity of ψ over $\mathcal{K}$ implies the validity of ψ over D. The size of the objects in $\mathcal{K}$ is related to the number of constants from D occurring in ψ. In particular, if ψ contains no constants then $\mathcal{K}$ contains a single object: the two valued Boolean lattice. In this case, the result is well-known and it provides a decision procedure for implications even if the underlying structure of D is infinite. Even in this case, a naive approach based on enumeration is, of course, exponential. However, the number of cases is reduced from $|D|^n$ to 2^n (where $|D|$ is the size of the domain and n is the number of variables in the formula). Moreover, in many cases there exist efficient techniques to determine the validity of the specific types of propositional formula. Examples include the use of Binary Decision Diagrams in circuit design [6, 7] and the use of propositional formulae in program analysis [17, 11, 8]. These techniques have been shown to remain effective when the domain is extended to contain a small number of additional constants [11].

Lattices (semilattices) play a central role in domain theory and provide the foundation for denotational semantics and various formal techniques in computer science. The need to determine the validity of an implication is also common in various application areas of computer science and hence also the general applicability of our results. In particular these results are of importance in: (a) the context of static program analysis (e.g. [23]), where for example a central question is to determine if a program state implies the precondition of a conditional statement [15]; (b) type analysis, where set constraints and implications between set constraints often arise (e.g [1]); and (c) semantic based techniques for program analysis (e.g [12]), where successive approximations to a solution are evaluated as long as the next approximation is not implied by the previous. Another potential area of application is in the context of constraint based languages where constraint satisfaction is one of the basic operations in computations.

The main contribution of this paper is the formal justification which enables us to provide effective algorithms to determine the validity of universally quantified implications, possibly containing less than three constants, in the spirit of the examples of the next section. It is important to consider the case with constants as this is a

common case in practical applications. Our approach becomes quickly impractical as the number of constants increases. Consequently, the approach is mainly applicable in cases where the number of constants which appear in formulae is small. This is the case, for example, in the context of polymorphic type analysis recently investigated in [9] which involves implications of the form illustrated by Example 3 (below). An implementation for this application based on the principles described here has illustrated considerable improvements. The only other application we are aware of is the zero-one principle as described in Example 2 (below), which does not involve marked elements. We believe that the results described here are of general interest and have high potential for applications in computer science.

The rest of this paper is organized as follows. The next section illustrates several motivating examples. Section 3 provides the necessary preliminary background. Section 4 provides our main contribution. Section 5 describes an application and Section 6 presents a short conclusion. A preliminary version of this paper can be found in [10] where an unsuitable notion of homomorphism was used. The current paper refines the notion of homomorphism making explicit the mapping between elements as implicitly assumed in the proofs.

2 Motivating Examples

The following examples illustrate the spirit of the results presented in this paper.

Example 1 *To determine the truth of an equality of the form,*

$$E_1 \equiv \quad (A \cap B) \cup (C \cap D) = (A \cup C) \cap (B \cup C) \cap (A \cup D) \cup (B \cup D); \quad \textit{or}$$

$$E_2 \equiv \quad \overline{A \cap B} = \overline{A} \cup \overline{B}$$

in a (possibly infinite) Boolean algebra of sets, it is sufficient to test the equality for values of A, B, C, D ranging over $\{0, 1\}$ viewing set union, intersection and completion respectively as propositional disjunction, conjunction and negation. This in itself is not surprising as it is well known that a Boolean algebra of sets is approximated by the two element Boolean algebra. However, it is interesting to observe that the following concise Prolog program can be used to prove or disprove equations of this form:

```
prove(and(0,0,0)).    prove(or(0,0,0)).    prove(neg(0,1)).
prove(and(0,1,0)).    prove(or(0,1,1)).    prove(neg(1,0)).
prove(and(1,0,0)).    prove(or(1,0,1)).
prove(and(1,1,1)).    prove(or(1,1,1)).

prove((F1,F2)) :- prove(F1), prove(F2).

imply(A,B) :- not( (prove(A), not(prove(B))) ).
equiv(A,B) :- imply(A,B), imply(B,A).
```

The predicate `prove/1` *specifies what can be proven about conjunctions of formula specified using the relations* `and(A,B,C)`, `or(A,B,C)` *and* `neg(A,B)`. *The program defines these relations to correspond to the equations:* $A \wedge B = C$, $A \vee B = C$ *and* $\neg A = B$.

The clause `imply(A,B) :- not( (prove(A), not(prove(B))) )` *specifies how an implication of the form* $A \rightarrow B$ *can be verified. This definition is based on the simple intuition that* $A \rightarrow B$ *is logically equivalent to* $\neg\neg(A \rightarrow B)$ *and hence to* $\neg(A \wedge \neg B)$. *However, the Prolog program expresses this in terms of provability and substitutes logical negation by non-provability (negation by failure). Hence, the relation* `implies(A,B)` *should be interpreted as follows: it should not be the case that a truth assignment makes A true while B is not true. Observe that A is true if it is provable as defined above and that executing* `prove(A)` *constructs (by backtracking) every truth assignment that makes A true. None of these should make B false, or otherwise stated, make the goal* `prove(B)` *fail.*

The clause `equiv(A,B) :- imply(A,B), imply(B,A)` *specifies that the equivalence of A and B is determined by checking that A implies B and that B implies A.*

The knowledgeable Prolog programmer will observe that the use of negation by failure in this context is only correct for queries of the form `imply(A,B)` *for which all variables in B are also in A. We may assume without loss of generality that this is always the case: If A does not contain a variable X we can replace A by* $A \wedge (X \vee \neg X)$.

To determine the truth of the equation E_1, *we query the program with the goal:*

```
?- equiv(  prove( and(A,B,T1),and(C,D,T2),or(T1,T2,T3) ),
           prove( or(A,C,T4),or(B,C,T5),or(A,D,T6),or(B,D,T7),
                  and(T4,T5,T8),and(T6,T7,T9),and(T8,T9,T3) ) ).
```

To check E_2 *we query the program with the goal:*

```
?- equiv(  prove( and(A,B,T1), neg(T1,T2) ),
           prove( neg(A,T3), neg(B,T4), or(T3,T4,T2) ) ).
```

Example 2 *A* comparison network *is a network constructed from* wires *and* comparators *with n inputs and n outputs. A comparator is a device with two inputs x and y and two outputs* $min(x, y)$ *and* $max(x, y)$. *We typically refer to the* sequences *of inputs and outputs of a comparison network. A* sorting network *is a comparison network for which the output sequence is monotonically increasing for every input sequence. The* zero-one principle *(cf [16]) states that a comparison network is a sorting network if and only if it correctly sorts all* 2^n *sequences of 0's and 1's. Namely if the network correctly sorts 0's and 1's then it correctly sorts any numeric inputs.*

We can generalize the notion of comparison network to accept inputs from any distributive lattice. A comparator is then a device with two inputs x and y and two outputs $x \sqcup y$ *and* $x \sqcap y$. *It follows that if such a comparison network transforms all sequences of 0's and 1's to an ordered chain, then it transforms any sequence of inputs to an ordered chain. To see this consider a comparison network which transforms the inputs* $x_1, \ldots, x_n$ *to the outputs* $x'_1, \ldots, x'_n$. *Each output* x'_i *can be expressed as a formula* w_i *involving the inputs* $\{x_1, \ldots, x_n\}$ *and the two operators* $\sqcup$ *and* $\sqcap$ *(exercise 5.3.4-28 in [16]). To prove that the comparison network transforms its inputs to a*

chain we must prove that $w_i \sqcup w_{i+1} = w_{i+1}$ for $1 \leq i < n$. This paper illustrates that if this is true for inputs ranging over $\{0,1\}$ then it is true for inputs ranging over any distributive lattice. In particular this proves the zero-one principle for sorting networks.

A less known application which will be justified by our results involves implications which contain constants from the underlying domain and is illustrated by the following,

Example 3 *Consider an arbitrary (possibly infinite) semi-lattice L which contains an element 'a'. We will show that to determine the truth of the implication*

$$\psi \equiv \quad (D \sqcup E = G) \wedge (F \sqcup D = C) \wedge (E \sqcup B = F) \wedge (a \sqcup B = B) \quad \rightarrow \quad (a \sqcup C = C)$$

in L, it is sufficient to test the implication over the three-element chain $0 < a < 1$. A Prolog program which implements a theorem prover for implications involving a constant 'a' and the least upper bound operation is obtained by adding the following facts to the program from Example 1.

```
prove(lub(0,0,0)).     prove(lub(a,1,1)).     prove(lub(0,a,a)).
prove(lub(1,0,1)).     prove(lub(0,1,1)).     prove(lub(1,a,1)).
prove(lub(a,0,a)).     prove(lub(1,1,1)).     prove(lub(a,a,a)).
```

where `lub(A,B,C)` *corresponds to $A \sqcup B = C$. The Prolog query to determine if ψ is valid is*

```
?- imply( prove( lub(D,E,G),lub(F,D,C),lub(E,B,F),lub(a,B,B) ),
          prove( lub(a,C,C) ) ).
```

We also prove that in order to prove implications of this kind, it is sufficient to test the implication over two two-element chains $0 < a$ and $a < 1$ instead of over a three-element chain. For an implication with n variables, this reduces the cost of a naive enumeration from $O(3^n)$ to $O(2^n)$. Moreover, since the two two-element chains are dual we may consult a single two-element chain twice replacing the above definition of `lub(A,B,C)` *by:*

```
prove(lub1(0,0,0)).    prove(lub2(0,0,0)).
prove(lub1(0,a,a)).    prove(lub2(0,a,0)).
prove(lub1(a,0,a)).    prove(lub2(a,0,0)).
prove(lub1(a,a,a)).    prove(lub2(a,a,a)).
```

To prove ψ we check the following two (dual) queries:

```
?- imply( prove( lub1(D,E,G),lub1(F,D,C),lub1(E,B,F),lub1(a,B,B) ),
          prove( lub1(a,C,C) ) ).
?- imply( prove( lub2(D,E,G),lub2(F,D,C),lub2(E,B,F),lub2(a,B,B) ),
          prove( lub2(a,C,C) ) ).
```

In some cases the given implication can also be verified without considering marked elements. For example, ψ is valid over the two element chain $0 < 1$ (viewing the constant a as a universally quantified variable). As a consequence, the techniques of Example 1 suffice to determine its validity. On the other hand, our results show that the invalidity of an implication in a semilattice with a marked element cannot be determined by testing over the two element chain $0 < 1$. Consider the implication

$$\varphi \equiv \quad (D \sqcup E = G) \wedge (F \sqcup D = C) \wedge (E \sqcup B = F) \wedge (a \sqcup B = B) \quad \rightarrow \\ (a \sqcup E = E)$$

It is easy to verify that φ is not valid over the two element chain $0 < 1$ and hence the previous approach is not applicable. However, by testing its validity over the two two-element chains $0 < a$ and $a < 1$ we can determine that φ is valid in a semi-lattice L if and only if a is the bottom element of L.

Consider now the implication

$$\chi \equiv \quad (D \sqcup E = G) \wedge (F \sqcup D = C) \wedge (E \sqcup B = F) \wedge (a \sqcup B = B) \quad \rightarrow \\ (a \sqcup E = E \wedge a \sqcup D = a)$$

which is not valid over the two element chain $0 < 1$. Hence we should test its validity over the the two two-element chains $0 < a$ and $a < 1$. From the fact that χ is invalid over both chains we can determine that it is invalid in any semilattice with marked elements.

3 Preliminaries

We review here the basic algebraic concepts which are used to obtain our results. For a more detailed discussion see for example [13, 14].

Semilattices and lattices

A *lattice* is an algebra $(\mathcal{L}, \sqcup, \sqcap)$ with two binary operations which are associative, commutative, idempotent and satisfy the axioms: $\forall_{a,b \in \mathcal{L}}.\ a \sqcap (a \sqcup b) = a$ and $\forall_{a,b \in \mathcal{L}}.\ a \sqcup (a \sqcap b) = a$. If we add also the axiom of distributivity, $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$ (or its dual which is equivalent), then we obtain a *distributive lattice*.

A *semilattice* is an algebra $(\mathcal{L}, \sqcup)$ with a single binary operation which is associative, commutative and idempotent. In this paper when we write "semilattice" we refer to an *upper semilattice* in which the binary operation corresponds to a least upper bound operator. All results can be equally stated also for a *lower semilattice* in which the binary operation corresponds to a greatest lower bound operator. The identity between lattices (semilattices) and partially ordered sets in which every pair of elements has a least upper bound and a greatest lower bound (a least upper bound) is well known. The correspondence between the operations $\sqcup$ and $\sqcap$ and the partial order $\sqsubseteq$ is characterized by: $\forall_{a,b \in \mathcal{L}}.\ a \sqsubseteq b \Leftrightarrow a \sqcup b = b$ and $\forall_{a,b \in \mathcal{L}}.\ a \sqsubseteq b \Leftrightarrow a \sqcap b = a$.

Let x and y be different elements of a (semi) lattice. Then, either $x \not\sqsubseteq y$ or $y \not\sqsubseteq x$. That is why we may assume without loss of generality that $x \not\sqsubseteq y$.

Marked elements

When we consider a concrete structure such as $L = (\mathcal{L}, \sqcup)$ (for example in computer science applications) we usually assume that every element of $\mathcal{L}$ has a name in the language (of formulae). Typically, the formulae of interest contain explicitly referenced constants from $\mathcal{L}$. Of course, ground formula pose no problem, as each symbol has a unique interpretation in the given domain. In the case, when a formula contains some variables and some constants we are confronted with a problem when attempting to determine the validity of the formula in a non-classical object. That is, the given structure L with the additional 0-ary operations — marked elements, which correspond to the constants of the formula.

As mentioned above, adding even one 0-ary operation to a signature may change greatly the properties. For example, it is obvious, that the implication from Example 3 is not valid in any two-element lattice. Other examples are nontrivial: (1) The well known fact, that every finite group has a finite basis of identities [25] turns out to be wrong for finite groups with one marked element [5]; and (2) The well known result of Tarsky ([2], part 3), that the elementary theory of the ordered field of real numbers is decidable turns out to be wrong for the ordered field of real numbers with one constant (corresponding to any non-recursive real number) [24].

A lattice with marked elements is a lattice $(\mathcal{L}, \sqcup, \sqcap, E)$, together with the set $E \subseteq \mathcal{L}$ of additional 0-ary operations, namely the set of names of the marked elements. For the purposes of this paper we may assume that there are finitely many marked elements. This is because we are reasoning about the validity of a given formula which may contain only finitely many constants. For any (semi) lattice L with marked elements E we denote by L_0 the sub(semi) lattice of L generated by E. We denote the unit of L_0 by $e = \sqcup E$.

Homomorphisms and congruences

Let $L_1 = (\mathcal{L}_1, \sqcup, \sqcap, E_1)$ and $L_2 = (\mathcal{L}_2, \sqcup, \sqcap, E_1)$ be two lattices with marked elements $E_1 = \{a_1, a_2, a_3, \ldots\}$ and $E_2 = \{b_1, b_2, b_3, \ldots\}$. A mapping $\varphi : \mathcal{L}_1 \to \mathcal{L}_2$ is called a *homomorphism*, if

1. $\varphi : E_1 \to E_2$ satisfies $\varphi(a_i) = b_i$ for $i \geq 1$;
2. $\forall_{x,\, y \in \mathcal{L}_1}.\ \varphi(x \sqcup y) = \varphi(x) \sqcup \varphi(y)$; and
3. $\forall_{x,\, y \in \mathcal{L}_1}.\ \varphi(x \sqcap y) = \varphi(x) \sqcap \varphi(y)$.

Note that we do not add any new axioms with marked elements to the definitions of classical structures. Homomorphisms of structures with marked elements are required to preserve marked elements (condition 1). For semilattices we have the same definition but without the third condition. It follows from the definition that L_1 and L_2 have the same number of marked elements and that a homomorphism preserves not only the marked elements but also any element from the sub(semi) lattice, generated by the marked elements.

We associate with a homomorphism $\varphi : L_1 \to L_2$ the binary relation $\sim_\varphi$ on $\mathcal{L}_1$ defined by $a \sim_\varphi b \Leftrightarrow \varphi(a) = \varphi(b)$. It is well known that $\sim_\varphi$ is a congruence on L_1 (cf

[13]). Namely, $\forall_{x,y,z\in L_1}.\ x \sim_\varphi y \rightarrow z \sqcup x \sim_\varphi z \sqcup y$. This congruence is named the *kernel* of the homomorphism φ. We say that a congruence $\sim$ on L_1 is *trivial* if $\forall_{x,y\in L_1}.\ x \sim y \Leftrightarrow x = y$.

Ideals, filters and closed filters

Let $L = (\mathcal{L}, \sqcup)$ be a semilattice. If L is an upper or lower semilattice we refer to its filters or ideals respectively. A filter $\mathcal{F}$ will be named a *closed filter* if $\forall_{x,y\in L}.\ x \sqcup y \in \mathcal{F} \rightarrow x \in \mathcal{F} \vee y \in \mathcal{F}$. One can observe, that a filter $\mathcal{F}$ is closed if and only if $L \setminus \mathcal{F}$ is subsemilattice.

If $\varphi : L_1 \rightarrow L_2$ is a homomorphism of semilattices and 1 is the unit of L_2 which is join-irreducible then $\varphi^{-1}(1)$ is a closed filter. If L_2 is nontrivial, 1 is the unit of L_2 which is join-irreducible and $\mathcal{F}$ is a closed filter of L_1, then there exists a homomorphism $\varphi : L_1 \rightarrow L_2$ such that $\varphi^{-1}(1) = \mathcal{F}$.

For every $\ell \in L$, we denote the *ideal* and the *filter* of ℓ by

$$\mathcal{I}_\ell = \left\{\ x \;\middle|\; x \sqsubseteq \ell\ \right\};\quad \textit{and}\quad \mathcal{F}_\ell = \left\{\ x \;\middle|\; \ell \sqsubseteq x\ \right\}.$$

Let $x \in L$. Let us denote by $\mathcal{F}^x$ the maximal filter of L which does not contain x. One can observe, that $\mathcal{F}^x = L \setminus \mathcal{I}_x$ and that $\mathcal{F}^x$ is a closed filter. Let $y \in L$, such that $x \not\sqsupseteq y$, then $y \in \mathcal{F}^x$.

Let $x, y \in L$. Let us denote by $\mathcal{F}^{x,y}$ the maximal filter of L which does not contain x and y. Observe that $\mathcal{F}^{x,y} = \mathcal{F}^x \cap \mathcal{F}^y$.

Algebraic approximation

We say that a universal algebra $\mathcal{U}$ is *approximated* in a class of universal algebras $\mathcal{K}$ if for every $a, b \in \mathcal{U}$ such that $a \neq b$ there exists a homomorphism φ of $\mathcal{U}$ onto an algebra from $\mathcal{K}$ such that $\varphi(a) \neq \varphi(b)$. Such a homomorphism is said to *separate* the elements a and b.

Let L be a lattice with marked elements approximated in a class $\mathcal{K}$ of lattices. It follows from the definition of approximation that the intersection of all congruences corresponding to the homomorphisms separating elements of L is trivial.

Quasi-identities

Let Σ_L and $\mathcal{V}$ denote the signature of an algebra L and a set of variable symbols. Then, $\mathcal{T}_L(\mathcal{V})$ denotes the term algebra over Σ_L and $\mathcal{V}$ which is the corresponding free algebra. Any mapping $\varphi : \mathcal{V} \rightarrow L$ can be uniquely extended to a homomorphism $\mathcal{T}_L(\mathcal{V}) \rightarrow L$ which is also denoted φ.

A *quasi-identity* of L is a universally quantified formula of the form

$$\psi \equiv \quad (\forall)\ (u_1 = v_1 \wedge \cdots \wedge u_n = v_n \rightarrow u_0 = v_0)$$

where $u_i, v_i \in \mathcal{T}_L(\mathcal{V})\quad (0 \leq i \leq n)$. We say that ψ is *valid* in L, denoted $L \models \varphi$, if for every homomorphism $\varphi : \mathcal{T}_L(\mathcal{V}) \rightarrow L$, $\varphi(\psi)$ is true.

It is well known that the validity of a quasi-identity τ is preserved by direct products and subsystems. In particular, if $L_i \models \tau$ for a class of semilattices L_i with marked elements $(i \in I)$ and L is a subsemilattice of the product $\prod_{i\in I} L_i$, then $L \models \tau$.

4 Approximating (semi) lattices

Let ψ be a quasi-identity over a semilattice L. In this section we show that the validity of ψ in L can be determined by inspection in a smaller (sub) semilattice L' of L the size of which is determined by the number of constants from L occurring in ψ. It is well known that any semilattice is approximated in the class of isomorphic copies of the two-element semilattice. We show that any semilattice L with marked elements (constants) is approximated in the class of semilattices which are constructed from the semilattice generated by the marked elements. We distinguish several cases depending on the number of marked elements: *none*, *one* and *two*. We also show that the result is minimal with respect to the size of L'. Ongoing research addresses the general result. All of the results given below hold for distributive lattices as well as for semilattices. The proofs are similar.

Our approach to the problem of determining the validity of a quasi-identity is based on the following proposition which follows from Birkhoff's well known theorem on subdirect decomposition [4].

Proposition 4.1
Let L be a semilattice with marked elements approximated by a class K of semilattices with marked elements. If a quasi-identity τ is valid in every semilattice from K then it is valid in L.

PROOF.
As stated in Section 3, the validity of a quasi-identity is preserved by direct products and subsystems. As L is approximated by K, it follows from the definition of algebraic approximation that the intersection of the kernels of homomorphisms of L onto semilattices from K which separate elements of L is trivial. Thus, it follows from Birkhoff's Theorem [3, 4] that L can be represented as a subdirect product of semilattices from K. Hence, if τ is valid in every semilattice from K then it is valid in L.

□

Remark 1
Let L be a semilattice with marked elements approximated by a class K of subsemilattices of L. Then Proposition 4.1 can be reformulated as follows: A quasi-identity τ is valid in L if and only if it is valid in K.

Remark 2
The word "semilattice" in Proposition 4.1 can be replaced by the word "universal algebra". In particular, the result is true for distributive lattices with marked elements as well.

Remark 3
Malćev [21] proves that for an axiomatizable class K of universal algebras the quasivariety generated by K coincides with the prevariety generated by K. A class K_0, consisting of a finite number of finite (semi)lattices with marked elements is axiomatizable. Consequently, every quasi-identity which is valid in K_0 is valid in a (semi)lattice A if and only if A is approximated in K_0.

Remark 4
If a semilattice contains the elements 0 or 1 then these elements are preserved by any homomorphism and hence may also be viewed as marked elements. In this paper by "marked elements" we mean additional elements besides 0 and 1.

4.1 Semilattice with no marked elements

The results for this case are known and are included here for completeness. Let C_2 be a two element chain $0 \sqsubset 1$.

Proposition 4.2 *Birkhoff [4]*
Every distributive lattice is approximated by $\{C_2\}$.

Proposition 4.3 *folklore*
Every semilattice is approximated by $\{C_2\}$.

4.2 Semilattice with one marked element

Let $L = (\mathcal{L}, \sqcup, E)$ be a semilattice with one marked element and C_3 a three element chain: $0 \sqsubset a \sqsubset 1$. If possible we consider the semilattices by which we approximate L as subsemilattices of L. In particular we denote the marked element of C_3 by the same letter as in L.

Proposition 4.4
A semilattice L with one marked element is approximated in the class of subsemilattices of C_3.

Proof.
Let $x, y \in L$, $x \not\sqsubseteq y$, that is $y \notin \mathcal{F}_x$. That is why $x \in \mathcal{F}^y$. Let us consider two cases:

1. $a \notin \mathcal{F}^y$. In this case let us consider the mapping $\varphi : L \to C_3$ defined by:

$$\varphi(z) = \begin{cases} 1 & \text{if } z \in \mathcal{F}^y \\ a & \text{if } z \in L \setminus \mathcal{F}^y. \end{cases}$$

2. $a \in \mathcal{F}^y$. In this case let us consider the mapping $\varphi : L \to C_3$ defined by:

$$\varphi(z) = \begin{cases} a & \text{if } z \in \mathcal{F}^y \\ 0 & \text{if } z \in L \setminus \mathcal{F}^y. \end{cases}$$

In both cases φ is a homomorphism which separates x from y. □

Remark 5
A three-element chain $0 \sqsubset a \sqsubset 1$ is approximated by two two-element chains $0 \sqsubset a$ and $a \sqsubset 1$. Let ψ be a formula containing n variables and 1 marked letter (constant). To examine the validity of ψ in a three-element chain we are to fulfill 3^n examinations. To examine the validity of ψ in two two-element chains we are to fulfill 2^{n+1} examinations. Thus, the approximation of a semilattice L with one marked element by two two-element chains is usually more efficient.

In some cases L does not contain a subsemilattice which is isomorphic to C_3. Moreover, some semilattices with one marked element can be approximated by a two-element chain. We prove that we can always use for approximation a subsemilattice of L which is minimal in the number of elements. Let us assume that L contains more than one element. In this case L contains a subsemilattice which is isomorphic to C_2.

Proposition 4.5
Let L be a semilattice with one marked element 'a'. Then, L is approximated in $\{C_2\}$ if and only if $\mathcal{I}_a$ or $\mathcal{F}_a$ contains exactly one element. That is, $a \equiv 0$ or $a \equiv 1$.

PROOF.

- $\Leftarrow$ The proof of this direction is analogous to the proof of the Proposition 4.4;
- $\Rightarrow$ To prove this direction we observe the following:
  - to separate an element x such that $x \sqsubset a$ from a we need a two-element chain $0 \sqsubset a$;
  - to separate an element x such that $x \sqsupset a$ from a we need a two-element chain $a \sqsubset 1$;
  - these two chains are not approximated one by another.

□

Remark 6
We can change in the previous statements the word "semilattice" for the word "distributive lattice". The proofs are the same. This is because we can define $\mathcal{F}^x$ for a distributive lattice as well and also in this case $\mathcal{F}^x$ will be a closed filter.

4.3 Semilattice with two marked elements

Let L be a semilattice with two marked elements a and b. Without loss of generality we assume that $a \not\sqsupseteq b$. If $a \sqsupseteq b$ we switch their names. Let us denote the three-element chains $a \sqsubset b \sqsubset 1$ and $0 \sqsubset a \sqsubset b$ by $\overline{C}_3$ and $\underline{C}_3$ respectively and the the three-element semilattice $\{a, b, c\}$ where $b = a \sqcup c$ by V_3. The four-element lattice $\{a, b, c, 0\}$ denoted S_4 is obtained by adding a bottom element to V_3 and the lattice S_5 is obtained by adding a top element to S_4. These semilattices are illustrated in Figure 1.

Proposition 4.6
A semilattice L with two marked elements is approximated in the class of five semilattices: C_2, $\overline{C}_3$, $\underline{C}_3$, V_3 and S_4.

PROOF.
Let $x, y \in L$, $x \not\sqsubseteq y$, that is $y \notin \mathcal{F}_x$ and hence $x \in \mathcal{F}^y$. Let us consider three cases:

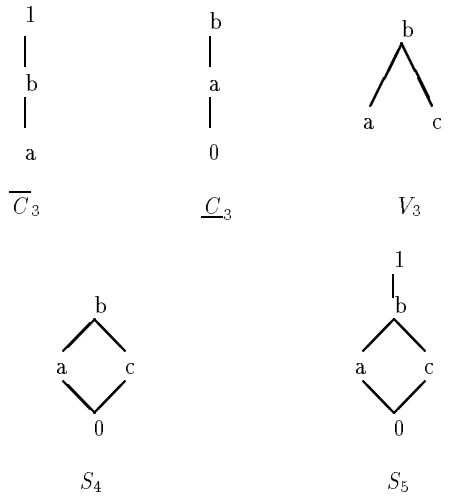

Figure 1: The semilattices $\overline{C}_3$, $\underline{C}_3$, V_3, S_4 and S_5.

1. $a \notin \mathcal{F}^y$ and $b \notin \mathcal{F}^y$. In this case consider the mapping $\varphi : L \to \overline{C}_3$ defined by:

$$\varphi(z) = \begin{cases} 1 & \text{if } z \in \mathcal{F}^y \\ b & \text{if } z \in \mathcal{F}^a \setminus \mathcal{F}^y \\ a & \text{otherwise.} \end{cases}$$

   To prove that φ is a homomorphism we have to show that (*) for any $z_1, z_2 \in L$

$$\varphi(z_1 \sqcup z_2) = \varphi(z_1) \sqcup \varphi(z_2) : \tag{1}$$

   We consider three cases:

   (a) If one of the elements z_1, z_2 is in $\mathcal{F}^y$ then so is $(z_1 \sqcup z_2)$ and consequently (1) is true. Let us assume that $z_1, z_2 \notin \mathcal{F}^y$.

   (b) If both of elements z_1, z_2 are in $\mathcal{F}^b \setminus \mathcal{F}^y$ or in $L \setminus (\mathcal{F}^y \cup \mathcal{F}^b)$ then so is $(z_1 \sqcup z_2)$ and consequently (1) is true.

   (c) The remaining case is $z_1 \in (\mathcal{F}^b \setminus \mathcal{F}^y)$, $z_2 \in L \setminus (\mathcal{F}^y \cup \mathcal{F}^b)$. In this case $(z_1 \sqcup z_2) \in (\mathcal{F}^b \setminus \mathcal{F}^y)$, consequently (1) is true.

   Hence, φ is a homomorphism and it separates x from y.

2. $a \in \mathcal{F}^y$ and $b \in \mathcal{F}^y$.

   We consider three subcases: $b \not\sqsubseteq a \sqcup y$, $a \not\sqsubseteq b \sqcup y$, and $b \sqsubseteq a \sqcup y$ & $a \sqsubseteq b \sqcup y$, that is $a \sqcup y = b \sqcup y$.

In the first case we consider the mapping $\varphi : L \to \underline{C_3}$ defined by:

$$\varphi(z) = \begin{cases} a & \text{if } z \in \mathcal{I}_{a \sqcup y} \cap \mathcal{F}^y \\ b & \text{if } z \in \mathcal{F}^y \setminus \mathcal{I}_{a \sqcup y} \\ 0 & \text{if } L \setminus \mathcal{F}^y. \end{cases}$$

Let us prove, that φ is a homomorphism: let $z_1, z_2 \in L$ and consider the following three cases.

(a) If one of the elements z_1, z_2 is in $\mathcal{F}^y \setminus \mathcal{I}_{a \sqcup y}$ then so is $(z_1 \sqcup z_2)$ and consequently (1) is true. Let us assume that $z_1, z_2 \notin \mathcal{F}^y \setminus \mathcal{I}_{a \sqcup y}$.

(b) If both of the elements z_1, z_2 is in $\mathcal{I}_{a \sqcup y} \cap \mathcal{F}^y$ or in $L \setminus \mathcal{F}^y$ then so is $(z_1 \sqcup z_2)$ and consequently (1) is true.

(c) If $z_1 \in \mathcal{I}_{a \sqcup y} \cap \mathcal{F}^y$, $z_2 \in L \setminus \mathcal{F}^y$ then $(z_1 \sqcup z_2) \in \mathcal{I}_{a \sqcup y} \cap \mathcal{F}^y$ since $z_1 \sqcup z_2 \not\sqsubseteq y$ and $z_1 \sqcup z_2 \sqsubseteq a \sqcup y$ and consequently (1) is true.

Hence, φ is a homomorphism and it separates x from y.

The second case is similar to the first.

To examine the third case, $b \sqsubseteq a \sqcup y \;\&\; a \sqsubseteq b \sqcup y$, we consider the mapping $\varphi : L \to S_4$ defined by:

$$\varphi(z) = \begin{cases} b & \text{if } z \in \mathcal{F}^{a,y} \\ 0 & \text{if } z \in \mathcal{I}_a \cap \mathcal{I}_y \\ a & \text{if } z \in \mathcal{I}_a \setminus \mathcal{I}_y \\ c & \text{if } z \in \mathcal{I}_y \setminus \mathcal{I}_a. \end{cases}$$

Let us prove that φ is a homomorphism. Consider the following five cases:

(a) If one of the elements z_1, z_2 is in $\mathcal{F}^{a,y}$ then so is $(z_1 \sqcup z_2)$ and consequently (1) is true.

(b) If both of the elements z_1, z_2 are in $\mathcal{I}_a \cap \mathcal{I}_y$ or in $\mathcal{I}_a \setminus \mathcal{I}_y$ or in $\mathcal{I}_y \setminus \mathcal{I}_a$ then so is $(z_1 \sqcup z_2)$ and consequently (1) is true.

(c) If $z_1 \in \mathcal{I}_a \cap \mathcal{I}_y$, $z_2 \in \mathcal{I}_a \setminus \mathcal{I}_y$ then $(z_1 \sqcup z_2) \in \mathcal{I}_a \setminus \mathcal{I}_y$ and consequently (1) is true.

(d) If $z_1 \in \mathcal{I}_a \cap \mathcal{I}_y$, $z_2 \in \mathcal{I}_y \setminus \mathcal{I}_a$ then $(z_1 \sqcup z_2) \in\in \mathcal{I}_y \setminus \mathcal{I}_a$ and consequently (1) is true.

(e) If $z_1 \in \mathcal{I}_a \setminus \mathcal{I}_y$, $z_2 \in \mathcal{I}_y \setminus \mathcal{I}_a$ then $(z_1 \sqcup z_2) \not\sqsubseteq y$ and $(z_1 \sqcup z_2) \not\sqsubseteq a$. That is, $(z_1 \sqcup z_2) \in \mathcal{F}^{a,y}$ and consequently (1) is true.

Hence, φ is a homomorphism and it separates x from y.

One can observe that if $\mathcal{I}_a \cap \mathcal{I}_y$ is empty then we have obtained a homomorphism $\varphi : L \to V_3$.

3. $a \notin \mathcal{F}^y$ and $b \in \mathcal{F}^y$. In this case consider the mapping $\varphi : L \to C_2$ defined by:

$$\varphi(z) = \begin{cases} 1 & \text{if } z \in \mathcal{F}^y \\ 0 & \text{if } z \in L \setminus \mathcal{F}^y. \end{cases}$$

It is obvious that φ is a homomorphism which separates x from y.

□

Corollary 4.7
A quasi-identity is valid in the class of all semilattices with two marked elements if and only if it is valid in the class of four semilattices: C_2, $\underline{C}_3$, $\overline{C}_3$ and S_4.

Remark 7
One can observe that none of the semilattices mentioned in Proposition 4.6 is approximated by the others. All of these semilattices except $\underline{C}_3$ are subsemilattices of $\underline{S}_4$. Consequently, to determine the truth of a quasi-identity ψ with less than three constants in a semilattice it is sufficient to determine its validity in $\underline{C}_3$ and S_4. Sometimes it is convenient to present these semilattices as subsemilattices of S_5 and to determine the validity of ψ in S_5 only.

Remark 8
It is worthy to notice that the statement of Proposition 4.6 does not depend on the relationship between the elements a and b. That is, in general we can not simplify the class of semilattices in which we approximate even in the case when $a \leq b$ or in the case $a \not\leq b$ and $b \not\leq a$.

Proposition 4.8
A distributive lattice with two marked elements is approximated in the class of the following distributive lattices: C_2, $\underline{C}_3$, $\overline{C}_3$ and S_4.

PROOF. The proof is similar to that of Proposition 4.6. □

Corollary 4.9
A quasi-identity is valid in the class of all distributive lattices with two marked elements if and only if it is valid in the class of four distributive lattices: C_2, $\underline{C}_3$, $\overline{C}_3$ and S_4.

5 An Application

In [9], Codish and Demoen describe a polymorphic type analysis for logic programs. In this context a domain of types is a lattice such as that depicted in Figure 2. The ordering on the lattice reflects the inclusion of the sets of terms described by each type. In [9] types are represented using so called *lub* clauses of the form $h \leftarrow body$ where *body* is a conjunction of atoms of the form $lub(t_1, t_2, t_3)$ and each such atom specifies a least upper bound of the form $t_1 \sqcup t_2 = t_3$ over the underlying domain of types. For example, the *lub* clause

$$append(list(A), list(B), list(C)) \leftarrow lub(A, B, C)$$

specifies a type for the append/3 relation in which the three arguments are respectively lists of type A, B and C such that $A \sqcup B = C$. A type of this form can be applied to determine that the result of concatenating a list of `real` numbers with a list of `int` (integer) numbers is a list of `numbers`.

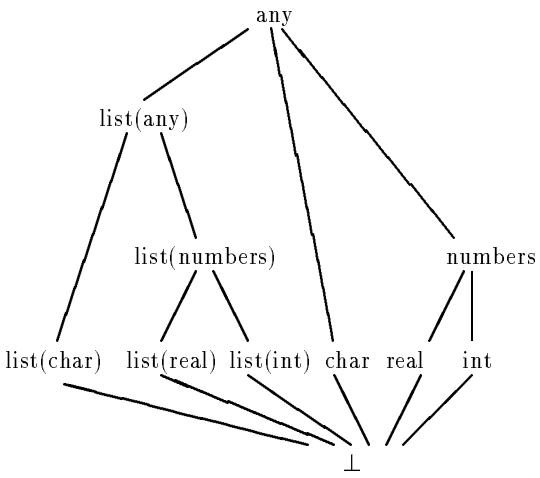

Figure 2: A simple domain of types

While performing a type analysis, the inference engine repeatedly adds new lub-clauses to the current approximation of a type until no new information is added. At each stage in the analysis the current approximation of the type is a set T of lub-clauses and a candidate lub clause $c \equiv h \leftarrow lubs$ is considered. The analysis must determine if c contains new information. An approximation is obtained by checking if for some clause $h \leftarrow lubs'$ in T, $(\forall)(lubs' \rightarrow lubs)$. In practice, there are more details. However, this is the topic of another paper. At the bottom line we need to determine the validity of an implication of the form $lubs' \rightarrow lubs$ in which $lubs'$ and $lubs$ are conjunctions of least upper bound operations which typically contain a small number of marked elements. The application of the principles described in this paper have led to substantial (2-3 orders of magnitude on the examples we have tested) gains in the efficiency of the type analysis described in [9].

6 Conclusion

We have addressed the question of determining the validity of universally quantified implications over algebraic structures such as semilattices and distributive lattices. We start out with the goal of determining the truth of a quasi-identity ψ in a given structure D. The goal is achieved by determining the validity of ψ in the general class $\mathcal{K}$ of objects to which D belongs, enriched by marked elements which correspond to the constants in ψ. Consequently, we obtain a stronger result, Namely, we determine the

truth of ψ for any structure of elements D' in $\mathcal{K}$. This observation has an important practical consequence. Assume, that we are given a simple domain of types T such as that depicted in Figure 2 and that we have determined based on the techniques of this paper that a program P is well typed. Now, assume that we change the underlying domain of types, perhaps by adding a new type to represent complex numbers. If the new domain contains at least the marked elements of P, then we do not have to check that P is well-typed in the new domain. The previous result still holds.

Our approach is based on the fact that the validity of a quasi-identity ψ in a universal algebra $\mathcal{L}$ is preserved under direct products and subalgebras. That is why it is sufficient to decompose $\mathcal{L}$ into a subdirect product and to determine the validity of ψ on the components of this decomposition. This paper applies techniques of algebraic approximation to obtain such decompositions.

Birkhoff [4] proves that every universal algebra $\mathcal{L}$ can be decomposed into the subdirect product of subdirectly irreducible universal algebras and that all subdirectly irreducible components belong to the variety generated by $\mathcal{L}$. For example, it is well known [14] that the only subdirectly irreducible Boolean algebra is a two-element Boolean algebra and that the only subdirectly irreducible Stone algebras are two-element and three-element Stone algebras. Hence to determine the validity of a quasi-identity (without constants) in any Boolean or Stone algebra it is sufficient to determine the validity for the two-element Boolean algebra or the two- and three-element Stone algebras. The corresponding results for these algebras with a signature enriched by marked elements and for quasi-identities with less than three constants follow from the results of this paper.

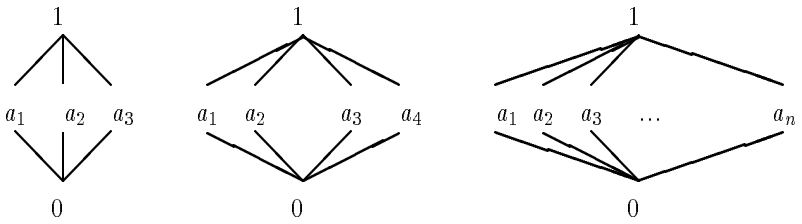

Figure 3: A family of subdirectly irreducible lattices

Let us mention, that for some other classes of lattices the approach described in this paper fails and the proof theoretic approach should be preferred. For example, it is easily verified that each of the lattices depicted in Figure 3 are subdirectly irreducible. Such domains cannot be simplified when trying to determine implications (even in the case with no constants). On the other hand, in the proof theoretic approach it is straightforward to observe that for any two distinct elements a, b (not including 0 or 1), $a \sqcup b = 1$ and $a \sqcap b = 0$. Observe that if the structures in Figure 3 are viewed as semi-lattices then they are obviously subdirectly reducible.

The results of Birkhoff are generalized by Mal'cev [18] who proves, under very general assumptions on the class of models, that models can be decomposed into subdirect products of subdirectly irreducible models. Hence we might apply the same

approach to determine the validity of quasi-identities of the general form

$$p_1(t_1^1, \ldots, t_{k_1}^1) \ \wedge \ \cdots \ \wedge \ p_n(t_1^n, \ldots, t_{k_n}^n) \ \ \rightarrow \ \ p_0(t_1^0, \ldots, t_{k_0}^0)$$

which are also known as definite Horn clauses. Of particular interest is to consider the predicates of set inclusion and set membership instead of the predicate of equality. However, in this case it is still not clear (even for the case with no marked elements) how to decompose the given algebraic structure and still obtain useful results. This is the topic of ongoing research.

Acknowledgments

The useful discussions with Bart Demoen are gratefully acknowledged. Avraham Melkman pointed out the reference to the zero-one principle for sorting networks. The comments of the anonymous referees are gratefully acknowledged.

References

[1] A. Aiken and E. L. Wimmers. Solving systems of set constraints (extended abstract). In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 329–340, Santa Cruz, California, June 22–25 1992. IEEE Computer Society Press.

[2] J. Barwise (ed.). Handbook of Mathematical Logic. North Holland Publishing Company, 1977.

[3] G. Birkhoff. Subdirect unions in univeral algebras. *Bull. Amer. Math. Soc.*, 50:764–768, 1944.

[4] G. Birkhoff. Lattice Theory. In *AMS Colloquium Publication, third ed.*, 1967.

[5] R. Bryant. The laws of finite pointed groups. *Bull. London Math. Soc.*, 14(2), 1982.

[6] R. Bryant. Graph based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[7] R. Bryant. Ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3), 1992.

[8] M. Codish and B. Demoen. Analysing logic programs using "`prop`"-ositional logic programs and a magic wand. *The Journal of Logic Programming*, 25(3):249–274, December 1995.

[9] M. Codish and B. Demoen. Deriving polymorphic type dependencies for logic programs using multiple incarnations of prop. In *Proceedings of the First International Static Analysis Symposium*, pages 281–296, Lecture Notes in Computer Science, Vol 864, Springer-Verlag, Sept. 1994.

[10] M. Codish and G. Mashevitzky. Proving implications by algebraic approximation. In *Proceedings of the Fourth International Conference on Algebraic and Logic Programming*, pages 6–22, Lecture Notes in Computer Science, Vol 850, Springer-Verlag, Sept. 1994.

[11] M. Corsini, K. Musumbu, A. Rauzy, and B. Le Charlier. Efficient bottom-up abstract interpretation of Prolog by means of constraint solving over symbolic finite domains. In *Proceedings of the Fifth International Symposium on Programming Language Implementation and Logic Programming*, Lecture Notes in Computer Science, Talin, Aug. 1993. Springer Verlag.

[12] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth ACM Symposium on Principles of Programming Languages*, pages 238–252, Jan. 1977.

[13] G. Grätzer. *Universal Algebra*. D. van Nostrand Company, Inc., 1968.

[14] G. Grätzer. *General Lattice Theory*. Akademie-Verlag, Berlin, 1978.

[15] K. Kennedy. A survey of data flow analysis techniques. pages 5–54. Chapter 1 in [23].

[16] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, 1973.

[17] B. Le Charlier and P. V. Hentenryck. Groundness analysis for Prolog: implementation and evaluation of the domain Prop. In *Proceedings Symposium on Partial Evaluation and Semantics-based Program Manipulation*, 1993.

[18] A. Malćev. Subdirect products of models. *Dokl. Akad. Nauk SSSR*, 109:264–266, 1956. In Russian and Chapter 5 in [20] (English translation).

[19] A. Malćev. On homomorphisms onto finite groups. In *Učenye Zapiski Ivanov*, volume 28, pages 49–60. Ped. Inst., 1958. In Russian. (English translation in: Amer. Math. Soc. Translations, series 2, volume 119, pages 67 - 81).

[20] A. Malćev. *The Metamathematics of Algebraic Systems*. North-Holland Publishing Company, 1971.

[21] A. Malćev (Maltsev). *Algebraic Systems*. Springer-Verlag, 1973.

[22] A. Melton, D. Schmidt, and G. Strecker. Galois connections and computer science applications. In D. P. *et al.*, editor, *Category Theory and Computer Programming*, pages 299–312. Springer-Verlag, 1986. Lecture Notes in Computer Science 240.

[23] S. S. Muchnick and N. D. Jones. *Program Flow Analysis: Theory and Applications*. Prentice Hall, 1981.

[24] Myasnikov and Remeslennikov. *Recursive p-adic numbers and elementary theories of finitely generated pro-p-groups. Izv. Akad. Nauk SSSR, Ser. Mat.*, v. 51, 613 - 634, 1987 (in Russian).

[25] H. Neuman. Varieties of groups. Springer-Verlag, Berlin-Heidelberg-New York, 1967.

[26] O. Ore. Galois connections. In *Trans. AMS*, volume 55, pages 493–513, 1944.

[27] B.M. Schein. On subdirectly irreducible semigroups. *Dokl. Akad. Nauk SSSR*, 144:999–1002, 1962 (in Russian).