November 1990

Decomposition of Domains

Achim Jung
University of Pennsylvania

Leonid Libkin
University of Pennsylvania

Hermann Puhlmann
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-84.

Decomposition of Domains

Abstract

The problem of decomposing domains into sensible factors is addressed and solved for the case of dI-domains. A decomposition theorem is proved which allows to represent an arbitrary dI-domain as a set of records with variants. The special case of direct product decompositions of Scott-domains is studied separately.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-84.

Decomposition Of Domains

MS-CIS-90-84
LOGIC & COMPUTATION 26

Achim Jung
Leonid Libkin
Hermann Puhlmann

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

November 1990

Decomposition of Domains

Extended Abstract

Achim Jung Leonid Libkin Hermann Puhlmann *

Abstract

The problem of decomposing domains into sensible factors is addressed and solved for the case of dI-domains. A decomposition theorem is proved which allows to represent an arbitrary dI-domain as a set of records with variants. The special case of direct product decompositions of Scott-domains is studied separately.

1 Introduction

Domain Theory is still a field which many computer scientists find hard to penetrate. In his last expository paper on the subject [18], Dana Scott confesses: "When I think of the number of headaches I have caused people in Computer Science who have tried to figure out the mathematical details of the Theory of Domains, I have to cringe." Indeed, the newcomer is usually confronted with a host of definitions almost none of which he can readily relate to concrete problems, although the concepts may appear 'plausible' to him. In the (in our opinion) best introduction to the field, the 'Pisa Lecture Notes' [16], Gordon Plotkin chose a rather more gentle approach. The 'domains' he considers are very primitive at the beginning, just sets, and step by step new constructs and properties are added to them: A bottom element transforms sets into flat domains, and thus the information order is introduced; next come slightly more complicated orders created by forming finite products of flat domains; function spaces call for the definition of dcpo and Scott-continuous function and via bilimits and powerdomains he finally arrives at bifinite domains. Furthermore, along the way he develops a syntax which allows to denote (most of) the elements of the domains, making them available for computation: The product appears as a set of arrays, the function space as a set of λ-terms, etc. (This aspect is also described elegantly and comprehensively in [1].) In this way, Plotkin creates the impression that all (bifinite) domains are built up from flat domains using various domain constructors. This may be reassuring for the novice but of course it is not explicitly confirmed in the text. Plotkin is just very carefully expanding his definitions and motivating each

*Addresses: Achim Jung and Hermann Puhlmann: Fachbereich Mathematik, Technische Hochschule Darmstadt, Schloßgartenstraße 7, D-6100 Darmstadt, Germany. Leonid Libkin: Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA 19104, U.S.A. L.Libkin was supported in part by NSF Grants IRI-86-10617 and CCR-90-57570.

new concept. But we may still ask to what extend this first impression could be transformed into a rigorous proof. To be more precise, we may ask: "Is it true, that every bifinite domain can be derived from flat domains using only lifting, product, coalesced sum, function space and convex powerdomain as constructors?" (A similar question was in fact asked — and found difficult — by Carl Gunter for the universal bifinite domain.)

How would one attack such a problem? We think the natural way to do it is to work backwards and to try to *decompose* domains into pieces that decompose no further. If we can show that the only irreducible domains are the flat domains then we are done.

At this point the informed reader may already have become nervous because he may know small finite counterexamples to the above question. But there are many variations of it which are equally interesting. We can restrict (or augment) the number of allowed constructions, we can change the class of domains we want to analyze, we can allow more (or fewer) primitive (i.e. irreducible) building blocks. The choice we have made for this paper is to consider Scott and dI-domains (cf. [4, 3]) and a single, albeit rather general, constructor, and instead of prescribing the irreducible factors we are curious what they will turn out to be. The advantage of a decomposition theorem of this kind is apparent: Instead of proving a property for general domains we can prove that it holds for the irreducible factors and that it is preserved under the constructions. We allow ourselves to compare this endeavor with the similar (and only recently completed) project of decomposing Finite Groups into finite simple groups, although the comparison is somewhat flattering: we cannot expect to find so much mathematically intriguing structure in domains.

What are the practical implications of our decomposition theorem? Well, in our particular setting we derive a very concrete representation of dI-domains as a set of 'tuples' which should simplify the implementation of dI-domains as abstract data types. Of course, there is a well-developed theory of effective representations (see [19, 13, 21, 14]), where one enumerates the set of compact elements and represents (a subset of) the infinite elements by recursively enumerable sets of compact approximations. However, this is more theoretical work and no one expects that we really ever use domains as dat types represented this way. Instead, our representation is much more concrete. To give an example, consider a domain which is the product of two flat domains. The traditional effective domain theory simply enumerates all elements, and, if enumerations of the elements of the two factors are already given, then these are combined with the help of pairing functions. We work rather in the opposite direction. For a given domain we seek to decompose it as far as possible and we will only enumerate the bases of the (irreducible) factors in the traditional way. The representation of the original domain is then put together as a set of 'tuples'.

This work was initiated by Peter Buneman's interest in generalizing relational databases, see [6]. He — quite radically — dismissed the idea that a database should be forced into the format of an n-ary relation and instead he allowed it to be an arbitrary anti-chain in a Scott-domain. The reason for this was that advanced concepts in database theory, such as 'null values', 'nested relations', 'complex objects' force one to augment relations and values with a notion of information order. Following Buneman's general approach, the question arises how to define basic database theoretic concepts such as 'functional dependency' for anti-chains in Scott-domains. For this one needs a way to speak about 'relational schemes' which are nothing but factors of the product from which the relation is a subset of. Buneman successfully defined a notion of 'scheme' for Scott-domains and it is this definition which at the heart of our results. So, we may view this paper as saying: Even generalized databases are relations.

The definition of scheme was discussed in [15] and an alternative definition was proposed. The

idea of both definitions is that the elements of a domain are treated as objects, and projecting an element into a scheme corresponds to loosing some information about this object. The definition of [15] is based on the assumption that the same piece of information is lost for every object. For example, if objects are records, it means that we loose information about some attributes' values. The idea of [6] is that every scheme has a sort of complement, and if we project one object to a scheme and the other to its complement, then there exists a join of two projections, i.e. every object consists of two independent "pieces of information". Intuitively it means that the domain itself could be decomposed into two corresponding domains.

The definition of [6] is stronger than the definition of [15]. It is the first definition that is used in our decomposition theory while the second definition serves as a tool to describe direct product decompositions of domains. Combining the decomposition theorems, we will prove a formal statement that clarifies the informal reasonings from the previous paragraph.

The paper is organized as follows. In the next section we shall quickly review some basic definitions from Domain Theory, mostly to fix notation and to remind the reader of a few less common concepts. In Section 3 we introduce semi-factors and prove basic properties of them. We apply these ideas and get a first decomposition theorem. This representation still contains a lot of redundancy and in Section 4 we show how to 'factor away' this redundancy. The resulting decomposition theorem yields a representation of dI-domains which is very tight.

A direct product decomposition is a particular and interesting instance of our general goal and deserves more detailed study. In the last section we do this by establishing a relationship between these decompositions and particular instances of congruence relations and neutral ideals. The idea to describe direct product decomposition via neutral ideals is borrowed from Lattice Theory where neutral ideals describe decompositions of bounded lattices. For domains we will obtain a more general kind of decomposition including direct product and coalesced sum as limit cases. These decompositions are given by families of subsets of a domain such that every element of domain has unique representation as the join of suitably chosen representatives of these sets. Pairs of permutable complemented congruences also describe direct product decomposition as well as they describe decompositions of algebras. Having proved characterizations of decompositions, we establish the result showing the relationship between the two notions of scheme.

2 Definitions

We are using the standard definitions such as they can be found in [12] and in [1]. In particular, *dcpo's* are *directed-complete partial orders* and they have suprema for all directed sets. Most of the time they have a least element which we denote by $\perp$. *Compact elements* in a domain are such that they cannot be below a supremum of a directed set without being below some element of that set already and if there are enough compact elements such that every element is the supremum of a directed collection of them, we call the dcpo *algebraic*. More suprema than just those of directed sets can exist: If every bounded set has a join then we call the dcpo *bounded-complete*, if every set has a join then we have a *complete lattice*. In case a bounded-complete dcpo is also algebraic we call it a *Scott-domain*. The expression 'algebraic complete lattice' is shortened to *algebraic lattice*. We will mostly study distributive Scott-domains for which it is sufficient to require the distributive law to hold in the principal ideals. (The standard textbook on distributive lattices is [2]). Even more restrictive is the definition of *dI-domains* (cf. [4, 3]): They are distributive Scott-domains

in which every principal ideal generated by a compact element is finite. Because of this strong finiteness property we can usally derive theorems about dI-domains very quickly from the same theorems stated for finite distributive Scott-domains.

All our functions are *Scott-continuous*, which means they carry the supremum of a directed set to the supremum of the image of the set. We do not make much use of them in this generality but mostly consider *projections* which are in addition idempotent and below the identity. Recall that projections always preserve existing infima and are completely determined by their image. Even the order between projections can be read off their image: it is simply inclusion. For more detailed information we refer to [8].

An element x in a lattice is *join– (meet–) irreducible* if from the equation $y \vee z = x$ ($y \wedge z = x$) we can deduce that x equals y or z. (In the presence of distributivity this is equivalent to the stronger property of *join– (meet–) primeness* but we will not make much use of this.)

Domain Theory also knows the concept of *ideal* which is a directed and downward closed subset. This is a generalization of 'ideal' as it is known in Lattice Theory, where these are sets which are downward closed and closed under finite suprema. We need a generalization which goes in a different direction:

Definition. A *quasi-factor* in a Scott-domain D is a downward closed subset which is closed under all existing joins.

In Section 4 we shall try to justify the word 'quasi-factor'. Factors of products of dcpo's with bottom have the property that there is always a canonical projection onto them. This is also true for quasi-factors in Scott-domains:

Lemma 1 *Let A be a quasi-factor of the Scott-domain D. Then $p_A: D \to D$, defined by*

$$p_A(x) = \bigvee(\downarrow x \cap A)$$

is a projection on D with image A.

Our first decomposition has the form of a general categorical *limit*. Their concrete description is given in terms of certain elements of the product of the dcpo's involved.

Definition. Let $\mathcal{D}$ be a set of dcpo's and let $\mathcal{F}$ be a set of Scott-continuous functions between elements of $\mathcal{D}$ (in the language of Category Theory: A diagram in **DCPO**). Furthermore, let $\bar{x} = (x_D)_{D \in \mathcal{D}}$ be an element (a *tuple*) of the cartesian product of all elements of $\mathcal{D}$. We say that $\bar{x}$ is *commuting* if the equation $x_E = f(x_D)$ holds for all functions $f: D \to E$ and all elements D, E in $\mathcal{D}$. Similarly, it is called *hyper-commuting (hypo-commuting)* if the inequation $x_E \geq f(x_D)$ ($x_E \leq f(x_D)$) holds.

The set of all commuting tuples forms the categorical limit of the diagram $(\mathcal{D}, \mathcal{F})$ and we denote it by $\lim_{\mathcal{F}} \mathcal{D}$. The set of hyper-commuting (hypo-commuting) tuples we call the *hyper-limit (hypo-limit)* and we reserve the notation $\mathsf{hyperlim}_{\mathcal{F}} \mathcal{D}$ ($\mathsf{hypolim}_{\mathcal{F}} \mathcal{D}$) for it. (We admit that so far we have not explored the categorical significance of these two constructions.) It is easy to see that **DCPO** is closed under limits and the two kinds of 'quasi limits'. Whether any of the other properties generally associated with domains is preserved depends on the structure of the diagram. For more detailed information consult [20].

3 Quasi-factors, semi-factors, and the First Decomposition Theorem

We begin by recalling from [6] and [17] some of the properties of quasi-factors.

Proposition 2 *Let D be a Scott-domain. Then the following holds:*

(i) $\{\bot_D\}$ *and D are quasi-factors of D.*

(ii) *If x is an element of a quasi-factor A of D and if $p_A(y)$ is less than x then $p_A(y) = x \wedge y$.*

(iii) *If D is distributive then p_A preserves existing suprema.*

(iv) *The set Q_D of all quasi-factors of D ordered by inclusion is an algebraic lattice.*

(v) *If D is distributive then Q_D is distributive.*

(vi) *In Q_D, the finite meet of quasi-factors is given by their intersection and $p_{A\cap B} = p_A \circ p_B = p_B \circ p_A$.*

(vii) *If D is distributive then (arbitrary) suprema in Q_D can be calculated pointwise and for $A, B \in Q_D, x \in D : p_{A\vee B}(x) = p_A(x) \vee p_B(x)$.*

The concept of 'quasi-factor' is still too general to serve as a definition of 'distinguished piece of a domain'. For example, every element x of a domain generates a quasi-factor $\downarrow x$, but in general such a principal ideal cannot be hoped to lead to a sensible decomposition. In [6] a more restrictive definition is introduced, that of a *scheme* and it is motivated by the database applications we had in mind there. Here we can give a new motivation based on the desired decomposition result. Consider the following theorem:

Theorem 3 *Let D be a finite distributive Scott-domain and let $\mathcal{A}$ be a set of quasi-factors which forms a sup-basis in Q_D. Let $\mathcal{F}$ be the set of projections $p_A|_B : B \to A$ where $A \subseteq B$ are two elements of $\mathcal{A}$. Furthermore, let $\hat{D}$ consist of those commuting tuples $\bar{x} = (x_A)_{A\in\mathcal{A}}$ for which the set $\{x_A \mid A \in \mathcal{A}\}$ is bounded in D. Then $\hat{D}$ is isomorphic to D with the isomorphisms*

$$\begin{aligned} \Psi: D \to \hat{D}, \Psi(x) &= (p_A(x))_{A\in\mathcal{A}} \\ \Phi: \hat{D} \to D, \Phi(\bar{x}) &= \bigvee\{x_A \mid A \in \mathcal{A}\}. \end{aligned}$$

The proof of this theorem is straightforward, one only has to bear in mind that the quasi-factor $D \subseteq D$ must be a supremum of elements of $\mathcal{A}$ and that suprema in Q_D are calculated pointwise. The theorem is unsatisfying, however, because in order to represent D through a set of quasi-factors, we need to include information that can only be gained by looking at D itself: the boundedness of the coordinates of $\bar{x}$. We shall now give a definition of a *semi-factor*, such that boundedness comes for free if only the tuple commutes.

Definition. A quasi-factor A of a Scott-domain D is called *semi-factor* if for all $x \in D, y \in A$ such that $p_A(x) \leq y$, it follows that x and y are bounded.

In [6] and in [17] it is shown that this definition works well in the test case of direct product decompositions: The semi-factors of a direct product $D \times E$ are in 1–1 correspondence to products of semi-factors of D and E. In particular, $D \times \{\bot_E\}$ and $\{\bot_D\} \times E$ are semi-factors in $D \times E$.

We collect the basic properties of semi-factors in a similar fashion as for quasi-factors:

Proposition 4 *Let D be a distributive Scott-domain. Then the following holds:*

(i) *$\{\bot_D\}$ and D are semi-factors of D.*

(ii) *The set S_D of all semi-factors of D ordered by inclusion is a distributive complete lattice.*

(iii) *If S and T are semi-factors of D then so are $S \cap T$ and $S \vee T$ where again the join is taken pointwise. (The latter also holds for arbitrary joins.) S_D is a sublattice of Q_D.*

The following lemma states that our definition yields the desired extension property:

Lemma 5 *Let $\mathcal{S}$ be a family of semi-factors of a finite distributive Scott-domain D and let $\mathcal{F}$ consist of all connecting projections. Let $\mathcal{S}$ be such that with $S, T \in \mathcal{S}$ we also have $S \cap T \in \mathcal{S}$. If $\bar{x} = (x_S)_{S \in \mathcal{S}}$ is a commuting tuple then the set $\{x_S \mid S \in \mathcal{S}\}$ is bounded in D.*

In our decomposition theorem we want to use as few semi-factors as possible which in turn should be as primitive as possible. As a first approximation we choose the set $J(S_D)$ of semi-factors which are join-irreducible in Q_D. This set has two properties which make it attractive: Every semi-factor is a join of irreducibles (in the finite case, but it will generalize to dI-domains) and a join-irreducible cannot be reached by a join of strictly smaller semi-factors, so it is in a sense unavoidable. But in order to apply the previous lemma we need a set closed under finite intersections, and in general $J(Q_D)$ will not do us this favor. We need another preparatory lemma:

Lemma 6 *Let D be a finite distributive Scott-domain and let $J(Q_D)$ be the set of join-irreducible semi-factors of D. Let $\bar{x} = (x_S)_{S \in J(S_D)}$ be a commuting tuple for $J(S_D)$ and the connecting projections $\mathcal{F}$. Let $\mathcal{D}$ be the set of finite non-empty intersections of elements of $J(S_D)$ and let $\mathcal{F}'$ be the appropriately extended set of connecting projections. Then $\bar{x}$ can be extended uniquely to a commuting element $\bar{x}'$ for $\mathcal{D}, \mathcal{F}'$.*

We can now state

Theorem 7 (The First Decomposition Theorem) *Let $J(S_D)$ be the set of all join-irreducible semi-factors of the finite distributive Scott-domain D and let $\mathcal{F}$ be the set of connecting projections. Then D is isomorphic to the limit of $J(S_D)$ over $\mathcal{F}$. The isomorphisms are given by*

$$\begin{aligned} \Psi\colon D &\to \varprojlim_{\mathcal{F}} J(S_D) \\ x &\mapsto (p_S(x))_{S \in J(S_D)} \end{aligned}$$

and

$$\begin{aligned} \Phi\colon \varprojlim_{\mathcal{F}} J(S_D) &\to D \\ (x_S)_{S \in J(S_D)} &\mapsto \bigvee_{S \in J(S_D)} x_S. \end{aligned}$$

For D not finite we have to deal with the following complication: The intersection of an infinite family of semi-factors is not necessarily a semi-factor again. We therefore do not know whether S_D is algebraic in general. In the case of dI-domains we are fine:

Proposition 8 *Let D be a dI-domain. Then S_D is algebraic and co-algebraic (i.e. S_D^{op} is algebraic).*

From [8] we recall that algebraic lattices have an inf-basis of meet-irreducible elements, and so for a dI-domain D the distributive lattice S_D has both a sup-basis of join-irreducibles and an inf-basis of meet-irreducibles. We can therefore state:

Corollary 9 *The First Decomposition Theorem holds for dI-domains.*

4 Factoring by quasi - factors and the Second Decomposition Theorem

In Group Theory and in Ring Theory we are familiar with the following technique. For a given strong substructure (normal subgroup, ideal, respectively) one studies the equivalence relation which identifies those elements which differ only by an amount contained in the substructure. A similar notion works for ideals in distributive lattices: If A is an ideal in L then we can set $x \sim y$ if there is an $a \in A$ such that $x \vee a = y \vee a$. (for details see [2].) Since domains lack arbitrary suprema we have to rework this definition a little bit:

Definition. Let A be a quasi-factor in a distributive Scott-domain D. On D define a binary relation θ_A by setting $x\ \theta_A\ y$ if there is $a \in A$ such that $y = x \vee a$ and $a \geq p_A(x)$. Let Θ_A be the symmetric and transitive hull of θ, that is, the smallest equivalence relation containing θ_A.

This definition proves to be extremely fruitful. Without proof we list the following properties:

Proposition 10 *Let D be a finite distributive Scott-domain and let A be a quasi-factor in D. Then the following holds:*

(i) $x\ \theta_A\ y \Longrightarrow x \leq y$.

(ii) $x\ \theta_A\ y \Longrightarrow y = x \vee p_A(y)$ *and for all* $a \in A$ *with* $y = x \vee a$, $a = p_A(y)$.

(iii) *Each equivalence class of Θ_A is connected and convex.*

(iv) *Θ_A is a congruence relation on D with respect to finite infima and existing suprema.*

(v) *Each equivalence class of Θ_A contains a least element.*

(vi) $\Theta_A = \theta_A^{-1} \circ \theta_A$.

We denote the function which maps each element onto the smallest element in its equivalence class by q_A. With this notation we can add to the previous proposition the following clauses:

(vii) *q_A is a projection on D.*

(viii) *p_A is injective on every equivalence class of Θ_A.*

(ix) *If A is a semi-factor then the image of an equivalence class under p_A is upward closed in A.*

(x) *If A is a semi-factor and if $\forall x \in D : p_A \circ q_A(x) = \perp_D$ then A is a direct factor of D.*

(The last clause justifies the wording 'semi-' and 'quasi-factor'.)

Every homomorphism $f: D \to E$ induces a canonical congruence relation on D, called the *kernel of f* ($\ker f$), which identifies exactly those elements of D which are mapped to the same element. Obviously, $\ker q_A = \Theta_A$. Let $Con(D)$ be the complete lattice of all congruences (with respect to finite infima and existing suprema) on D.

Proposition 11 *Let D be a finite distributive Scott-domain and let A be a quasi-factor in D. Then the following is true:*

(i) $\ker p_A$ *is a congruence with respect to arbitrary infima and arbitrary (existing) suprema.*

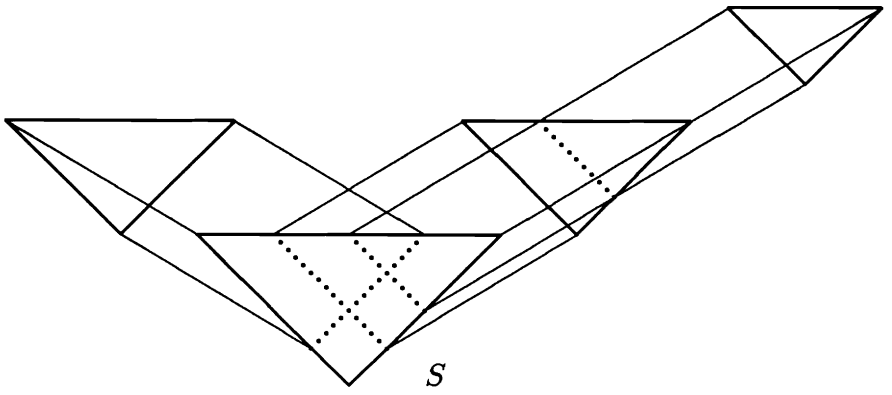
(ii) $\ker p_A \cap \Theta_A = \Delta_{D \times D} = 0_{Con(D)}$.

(iii) $\ker p_A \vee \Theta_A = D \times D = 1_{Con(D)}$.

Also without proof we assure the reader that extending these results to dI-domains is possible. (Cf. [17])

Corollary 12 *Proposition 10 and Proposition 11 hold for dI-domains, in particular, equivalence classes of Θ_A and $\ker p_A$ are closed under directed suprema and q_A is Scott-continuous.*

Since this section lacks all proofs we offer the reader a picture as compensation. If S is a semi-factor in a dI-domain D then we may visualize D as composed of its Θ_S-equivalence classes as follows:



Knowing S, $\mathrm{im}\ q_S$ and the action of p_S on the image of q_S we can reconstruct the domain:

Proposition 13 *Let D be a finite distributive domain and let S be a semi-factor in D. Then D is isomorphic to the set $\hat{D} = \{(s, a) \in S \times \mathrm{im}\ q_A \mid p_S(a) \leq s\}$ ordered pointwise. The isomorphism is given by $p_S \times q_S : D \to \hat{D}$ and by the supremum function for the other direction.*

We denote the image of q_S by $D/_S$ because it may also be seen as a set of representatives for the equivalence classes of Θ_S. Also, in an ordered set let $\overset{\vee}{\downarrow}x$ be the principal ideal generated by x without x itself. With this notation we are now ready to formulate:

Theorem 14 *Let D be a finite distributive Scott-domain (a dI-domain) and let $J(S_D)$ be the set of all join-irreducible semi-factors of D. Define*

$$RJ(S_D) = \{S/_{\bigvee \overset{\vee}{\downarrow} S} \mid S \in J(S_D)\}$$

and

$$\mathcal{F} = \{q_{\bigvee \overset{\vee}{\downarrow} S} \circ p_S \Big|_{T/_{\bigvee \overset{\vee}{\downarrow} T}} \mid S \subseteq T \in J(S_D)\}.$$

Then D is isomorphic to the hyper-limit of $RJ(S_D)$ over $\mathcal{F}$ with the isomorphisms

$$\begin{aligned} \Psi\colon D &\to \operatorname*{hyperlim}_{\mathcal{F}} RJ(S_D) \\ x &\mapsto (q_{\bigvee \overset{\vee}{\downarrow} S} \circ p_S(x))_{S \in J(S_D)} \end{aligned}$$

and

$$\begin{aligned} \Phi\colon \operatorname*{hyperlim}_{\mathcal{F}} RJ(S_D) &\to D \\ (x_S)_{S \in J(S_D)} &\mapsto \bigvee_{S \in J(S_D)} x_S. \end{aligned}$$

Due to the limitations that the organizers of MFPS91 set us we have to stop here with our theory of decompositions. Of course, what we have described so far are only the very first steps and much more could be done (quite a bit more has been done already). A full version of this paper will contain (besides proofs) a decomposition theorem which works 'from above' by factoring with very large semi-factors, the relationship between our representation and the usual inverse limit representation will be examined, the action of domain constructors in terms of this representation will be described and much more. In the last section of this extended abstract we do a little more basic theory, we study the particular role of direct product decompositions by looking at the lattice of congruences and the lattice of quasi-factors.

5 Characterization of direct product decompositions

There exist several nice characterizations of the direct product decompositions of arbitrary algebras, see [5, 10]. In this section we will find the analogies of two of them for domains. We will characterize direct product decompositions via complemented permutable congruences and neutral complemented ideals. Using our characterization, we will prove a result which explains the notion of scheme proposed in [6] in order to develop a domain-theoretic model of generalized databases. In fact, the definition of schemes of [6] (here we call them semi-factors) works for domains which are similar to domains of flat records. An alternative definition of scheme was introduced in [15]. According to this definition, a scheme is a quasi-factor such that the associated projection maps the maximal elements of the domain to the maximal elements of the quasi-factor. This definition emphasizes the fact that schemes are significant parts of domains. It is more general than the

original definition of semi-factor which, in turn, is based on properties that schemes of domains of flat records should satisfy. We will prove that in a certain class of domains the condition that every scheme is a semi-factor implies that the domain is very similar to a domain of flat records. Thus, our decomposition results are helpful in understanding the domain-theoretic generalization of databases.

This section contains six short subsections. In the first we recall two well-known results from universal algebra. The second demonstrates how domains can be treated as partial algebras with operations of infinite arity, and introduces the concepts of ideal and congruence for them, the first one being identical to the quasi-factors. In the third subsection we give our basic lemma which reduces direct product decompositions of domains to those of principal ideals generated by maximal elements. Neutral ideal are studied in the fourth subsection. They give rise to so-called *general decompositions* which have coalesced sum and direct product decompositions as two limit cases. The fifth subsection deals with characterizations of direct product and general decompositions via congruences. Finally we apply the previous results in order to characterize domains in which two notions of scheme coincide.

From now on all domains are Scott-domains, and we will write domain instead of Scott-domain. We will denote the sets of maximal elements of a domain D and of a quasi-factor A by D^{max} and A^{max}, respectively.

5.1 Algebraic preliminaries

Given an algebra $\langle A, \Omega \rangle$, a congruence Θ on it is an equivalence relation on A such that for any $\omega \in \Omega$ of arity n and for any $x_i, y_i \in A, i \in [1,n]$, $\forall i \in [1,n] : x_i \Theta y_i$ implies $\omega(x_1, \ldots, x_n) \Theta \omega(y_1, \ldots, y_n)$. Congruences considered as binary relations form an algebraic lattice $Con(\langle A, \Omega \rangle)$ in which the bottom element is the equality and the top element is the total equivalence relation. Then direct product decompositions of $\langle A, \Omega \rangle$ (i.e. decompositions $\langle A, \Omega \rangle \simeq \langle A_1, \Omega \rangle \times \langle A_2, \Omega \rangle$) are in one-to-one correspondence with pairs (Θ_1, Θ_2) such that Θ_2 is a complement of Θ_1 in $Con(\langle A, \Omega \rangle)$ and Θ_1, Θ_2 are permutable, i.e. $\Theta_1 \circ \Theta_2 = \Theta_2 \circ \Theta_1$, see [5]. In fact, $\langle A_1, \Omega \rangle \simeq \langle A, \Omega \rangle / \Theta_1$ and $\langle A_2, \Omega \rangle \simeq \langle A, \Omega \rangle / \Theta_2$ or vice versa.

Let L be a lattice. An element $a \in L$ is called *neutral* if for every $x, y \in L$ the sublattice $\langle x, y, a \rangle$ generated by x, y, a is distributive. If L is bounded, then every direct product decomposition $L \simeq L_1 \times L_2$ can be represented as $L \simeq (a] \times (\overline{a}]$, where a is a neutral complemented element and $\overline{a}$ is its complement, see [10]. Both ideals $(a]$ and $(\overline{a}]$ are neutral elements of the ideal-lattice, i.e. so-called *neutral ideals*. It is also well-known that there is a one-to-one correspondence between direct product decompositions $L \simeq L_1 \times L_2$ and pairs (I_1, I_2) where I_1, I_2 are neutral ideals and I_2 is a complement of I_1 in the ideal-lattice. In fact, $L_1 \simeq I_1$ and $L_2 \simeq I_2$ or vice versa.

5.2 Domains as algebras. Congruences and ideals

In order to transfer the previous characterizations to domains, we have to introduce algebraic structure on domains. Let D be a domain. We consider it as a partial algebra containing the operations of infinite arity, namely infima and existing suprema for all possible subsets $X \subseteq D$. Thus, D becomes a partial algebra whose operations may be of arbitrary arity. It is well-known that the previous results about decompositions are not true for algebras with partial operations of infinite arity [5].

A subalgebra of this partial algebra we could call *subdomain* but in Semantics this notion has no generally accepted meaning. In Lattice Theory an ideal is a downward closed set which is closed under finite joins. In our algebraic interpretation the *ideals* are downward closed subsets of a domain which are closed under all existing joins, i.e. they are *quasi-factors*. A *congruence* is an equivalence relation Θ such that for any $x_i, y_i \in D, i \in I, I$ an arbitrary set of indices, $x_i \Theta y_i$ for all $i \in I$ implies $\bigwedge\{x_i : i \in I\} \Theta \bigwedge\{y_i : i \in I\}$, and if both $x = \bigvee\{x_i : i \in I\}$ and $y = \bigvee\{y_i : i \in I\}$ exist, then $x \Theta y$.

If D is a lattice, our definition of congruence coincides with the definition of *complete congruence* of a lattice introduced recently in [11, 7]. These congruences form a complete lattice denoted $Con(D)$. However, in contrast to the case of operations of finite arity, this lattice may fail to be algebraic. It was proved in [7] that for every complete lattice L there exists a lattice M such that L is isomorphic to the lattice of complete congruences of M. Moreover, M can be chosen among modular algebraic lattices. Therefore, we cannot guarantee algebraicity of $Con(D)$. See [11, 7] for details.

If $x \in D$, then Θ^x is the restriction of Θ to $\downarrow x$. The lattices of congruences of $\downarrow x$ will be denoted by $Con(x)$ if D is understood.

The one-to-one correspondence between ideals and congruences which is of great importance in ring and lattice theory also holds in our setting. In fact, if $[x]\Theta$ denotes the equivalence class containing x, then $[\bot]\Theta$ is a quasi-factor. And, if A is a quasi-factor then Θ_A (as defined in the previous section) is the corresponding congruence.

We will also also need a concept of *scheme*. It was introduced in [15] in order to generalize the analogous concept of [6] known in this paper as semi-factor. A quasi-factor $A \subseteq D$ is called a *scheme* if $p_A(x)$ is a maximal element of A for every $x \in D^{max}$. Any semi-factor is a scheme. Although all the factors we will decompose domains into, will be semi-factors, it is enough to require that they be schemes.

5.3 The main lemma

The following lemma generalizes the result of [10] which describes direct product decompositions of bounded lattices.

Lemma 15 *The direct product decompositions of a domain D are given by pairs of schemes* (A_1, A_2) *such that* $A_1 \cap A_2 = \{\bot\}$, *and* $\forall x \in A_1^{max}, y \in A_2^{max} : x \vee y$ *exists,* $x \vee y \in D^{max}$, *and* x *is a neutral element in the principal ideal* $\downarrow(x \vee y)$. *In fact,* $D \simeq A_1 \times A_2$. *Moreover,* A_1 *and* A_2 *are neutral complemented elements of the lattice* Q_D *of quasi-factors.*

5.4 General decompositions of domains

It is a natural question whether all pairs of neutral complemented elements of the lattice of quasi-factors give rise to a direct product decomposition as it is the case for lattices. The answer, as we are going to show, is "no". In fact, neutral complemented ideals describe a much more general kind of decomposition which, for example, also includes coalesced sum decomposition.

Definition. A pair of quasi-factors A_1, A_2 of a domain D is called a *general decomposition* of D, which is denoted by $D = comp(A_1, A_2)$, if every element $x \in D$ has unique representation as $x = x_1 \vee x_2$ such that $x_1 \in A_1$ and $x_2 \in A_2$.

Obviously, if $D = comp(A_1, A_2)$ then $A_1 \cap A_2 = \{\perp\}$. The examples of general decomposition are direct product decompositions $D \simeq A_1 \times A_2$ (we may suppose that A_1, A_2 are embedded in D) and coalesced sum decomposition $D = D_1 + D_2$.

Theorem 16 *The general decompositions of a domain D are in one-to-one correspondence to pairs of neutral complemented elements of the lattice Q_D.*

In the other words, if $D = comp(A_1, A_2)$, then A_1 and A_2 are neutral elements of Q_D complementing each other and vice versa. Direct product and coalesced sum decompositions appear now as two limit cases of general decompositions.

Proposition 17 *A general decomposition $D = comp(A_1, A_2)$ is a direct product decomposition $D \simeq A_1 \times A_2$ iff $\forall x \in A_1, y \in A_2 : x \vee y$ exists.*

Proposition 18 *A general decomposition $D = comp(A_1, A_2)$ is a coalesced sum decomposition $D = A_1 + A_2$ iff $\forall x \in A_1, y \in A_2, x, y \neq \perp : x \vee y$ does not exist.*

5.5 Characterization via congruences

In this subsection we show that the characterization of direct product decompositions via congruences works for domains, although we introduced domains as algebras with partial operations of infinite arity. We also show how to describe general decompositions via congruences.

Theorem 19 *Let D be a domain. There is one-to-one correspondence between general decompositions $D = comp(A_1, A_2)$ and pairs of congruences (Θ_1, Θ_2) such that for every $x \in D^{max}$ the congruences Θ_1^x and Θ_2^x are permutable, and Θ_2^x is a complement of Θ_1^x in $Con(x)$. This correspondence is given by*

$$(\Theta_1, \Theta_2) \to (A_1, A_2) \quad : \quad A_1 = [\perp]\Theta_2, A_2 = [\perp]\Theta_1,$$

$$(A_1, A_2) \to (\Theta_1, \Theta_2) \quad : \quad x \Theta_i y \text{ iff } p_{A_i}(x) = p_{A_i}(y),$$

$$i = 1, 2.$$

Theorem 20 *The above described mappings $(A_1, A_2) \leftrightarrow (\Theta_1, \Theta_2)$ set up one-to-one correspondence between direct product decompositions $D \simeq A_1 \times A_2$ and pairs (Θ_1, Θ_2) of congruences such that Θ_1 and Θ_2 are permutable, and Θ_2 is a complement of Θ_1 in $Con(D)$.*

A simpler result can be stated for *qualitative* domains. Recall that a domain D is called *qualitative* iff $\downarrow x$ is a Boolean lattice for every $x \in D$ [9].

Corollary 21 *General decompositions of a qualitative domain are given by pairs of congruences (Θ_1, Θ_2) such that for every $x \in D^{max}$ Θ_1^x and Θ_2^x are complementary elements in $Con(x)$.*

5.6 Schemes and semi-factors in qualitative domains

In this subsection we show that those qualitative domains in which our two notions of scheme coincide (in fact, the above defined schemes and Buneman's schemes [6] known here as semi-factors) are very similar to the domains of flat records. Motivation of the two definitions of scheme was given in [15]. Database motivation for the definition of semi-factor makes use of the assumption that every element of a domain, which is a database object, can be represented as two subobjects, each carrying independent piece of information, while the motivation of the definition of scheme appeals only to our intuition about what a "significant part of a domain should be". We give a precise mathematical formulation of these informal reasonings.

Let D be a domain. It is called *simple* if it has no proper scheme, i.e. if it has no scheme but $\{\perp\}$ and itself. Since schemes appear as equivalence classes of congruences, this definition is motivated by the definition of simple lattices, i.e. lattices having no nontrivial congruences [5, 10].

Every semi-factor is a scheme; the converse does not hold in general. The next theorem gives a characterization of those qualitative domains in which every scheme is a semi-factor, i.e. in which the two notions of scheme ([6] and [15]) coincide.

Theorem 22 *Let D be a qualitative domain. Then every scheme in D is a semi-factor if and only if D is isomorphic to a direct product of simple domains.*

Corollary 23 *Let D be a qualitative domain in which every scheme is a semi-factor. Then S_D is an atomistic Boolean lattice and the schemes of D are in 1-1 correspondence with subsets of the set of atoms of S_D.*

This corollary gives a mathematical description of the assumption that every database object represented as an element of a domain can be decomposed into two "independent" subobjects, namely to its projection onto a scheme and its complement in S_D.

The proof of the above theorem is based on the decomposition theory developed in this section and algebraicity of S_D for dI-domains proved in Section 3.

Acknowledgement

We would like to thank Peter Buneman for inventing the beautiful tool of semi-factors for us domain-theorists. Together with the first author he found a preliminary version of the First Decomposition Theorem while he visited Darmstadt in July 1989. We also thank Klaus Keimel for directing our attention to the classical theory of ideals in distributive lattices. He realized that our definition of the congruence Θ_S generalizes the classical definition to domains. E.T. Schmidt informed us about recent results on complete congruences and interrupted our vain attempts to prove algebraicity for complete congruence lattices of domains.

References

[1] S. Abramsky. Domain Theory In Logical Form. *Annals of Pure and Applied Logic*, 1988. To appear.

[2] R. Balbes and P. Dwinger. *Distributive Lattices.* University of Missouri Press, Columbia, 1974.

[3] G. Berry. Modèles Complèment Adéquats et Stables des Lambda-calculs typés. Thèse de Doctorat d'Etat, Université Paris VII, 1979.

[4] G. Berry. Stable Models of Typed λ-calculi. In *Proceedings of the 5th International Colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89. Springer Verlag, 1978.

[5] G. Birkhoff. *Lattice Theory.* 3rd ed., AMS, Providence, RI, 1967.

[6] P. Buneman, A. Jung, and A. Ohori. Using Powerdomains to Generalize Relational Databases. *Theoretical Computer Science*, 1990. (Accepted for publication.).

[7] R. Freese, G. Grätzer, E.T. Schmidt. On complete congruence lattices of complete modular lattices. Draft, July 1990.

[8] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices.* Springer Verlag, Berlin, 1980.

[9] J.-Y. Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science* 45:159–192, 1986.

[10] G. Grätzer. *General Lattice Theory.* Academic Press, New York, 1978.

[11] G. Grätzer, H. Lakser. On complete congruence lattices of complete lattices. *Trans. Amer. Math. Soc.*, to appear.

[12] A. Jung. *Cartesian Closed Categories of Domains*, volume 66 of *CWI Tracts.* Centrum voor Wiskunde en Informatica, Amsterdam, 1989.

[13] A. Kanda. Fully Effective Solutions of Recursive Domain Equations. In J. Beçvar, editor, *Mathematical Foundations of Computer Science*, volume 74. Springer-Verlag, 1979. Lecture Notes in Computer Science.

[14] K. G. Larsen and G. Winskel. Using Information Systems to Solve Recursive Domain Equations Effectively. In G. Kahn, D. B. MacQueen, and G. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 109–130. Springer-Verlag, 1984.

[15] L. Libkin. Domain-theoretic approach to extending relational algebra to complex objects. Draft, University of Pennsylvania, 1990.

[16] G. D. Plotkin. Post-Graduate Lecture Notes in Advanced Domain Theory (incorporating the "Pisa Notes"). Dept. of Computer Science, Univ. of Edinburgh, 1981.

[17] H. Puhlmann. Verallgemeinerung relationaler Schemata in Datenbanken mit Informationsordnung, 1990. (Diplomarbeit, Technische Hochschule Darmstadt.).

[18] D. S. Scott. Domains for Denotational Semantics. In M. Nielson and E. M. Schmidt, editors, *International Colloquium on Automata, Languages and Programs*, volume 140, pages 577–613, Berlin, 1982. Springer Verlag.

[19] M. B. Smyth. Effectively Given Domains. *Theoretical Computer Science*, 5:257–274, 1977.

[20] P. Taylor. Homomorphisms, Bilimits and Saturated Domains. Some Very Basic Domain Theory. Draft, Imperial College London, 15pp., 1987.

[21] K. Weihrauch and T. Deil. Berechenbarkeit auf cpo's. Technical Report 63, Rheinisch-Westfälische Technische Hochschule Aachen, 1980.