Symbolic Domain Decomposition

Jacques Carette[1], Alan P. Sexton[2], Volker Sorge[2], and Stephen M. Watt[3]

[1]Department of Computing and Software, McMaster University
www.cas.mcmaster.ca/~carette
[2]School of Computer Science, University of Birmingham
www.cs.bham.ac.uk/~aps|~vxs
[3]Department of Computer Science, University of Western Ontario
www.csd.uwo.ca/~watt

Abstract. Decomposing the domain of a function into parts has many uses in mathematics. A domain may naturally be a union of pieces, a function may be defined by cases, or different boundary conditions may hold on different regions. For any particular problem the domain can be given explicitly, but when dealing with a family of problems given in terms of symbolic parameters, matters become more difficult. This article shows how hybrid sets, that is multisets allowing negative multiplicity, may be used to express symbolic domain decompositions in an efficient, elegant and uniform way, simplifying both computation and reasoning. We apply this theory to the arithmetic of piecewise functions and symbolic matrices and show how certain operations may be reduced from exponential to linear complexity.

1 Introduction

The goal of this paper is to develop general methods to work with domains having symbolically defined parts. The *raison d'être* of symbolic mathematical computation is to compute and reason about general expressions, rather than working with particular values valid only at specific points. Matters are simplest when variables range over a domain of interest and all expressions are valid over the entire domain. Sometimes it is useful to perform simplifications or other operations that are valid over part, but not all, of the domain. In this situation, software systems may or may not record the excluded region. But we are not always so fortunate to have this one-region situation. More generally, the domain of interest may be made up of several pieces with expressions taking different forms on different parts. Moreover, the demarcation of the parts may be defined symbolically. This paper explores how to represent such expressions concisely in a uniform way that simplifies computation and reasoning.

When we do arithmetic with piecewise functions defined on explicit partitions, we can do a mutual refinement of domain partitions to obtain regions that may each be handled uniformly. When the partitions are defined symbolically, however, we obtain a massive explosion of cases — for N binary operations on functions of k pieces there are k^N potential regions. We say "potential" regions

because, of this large number, not all of the regions are in fact feasible. Furthermore, it is usually not possible to determine which regions are feasible and which are not. For example, the sum $\sum_{i=1}^{N} f_i(x)$ of the functions

$$f_i(x) = \begin{cases} 0 & \text{for } x < k_i, \\ A_i & \text{for } x \geqslant k_i. \end{cases}$$

has $\sum_{i=0}^{N} N!/i!$ possible orderings of the k_i, with each ordering having between 2 and $N+1$ regions. There is no ordering in which all 2^N regions are realized.

We take the view that it is generally preferable to have a single compact closed-form expression rather than a collection of cases, even if it means introducing some new operations. For example, we are perfectly satisfied using the Heaviside step function and giving the sum of the f_i as $\sum_{i=1}^{N} A_i H(x - k_i)$.

In this paper we show how hybrid sets, a variation on multisets allowing negative multiplicities, enable us to write elegant closed form expressions of the form we desire. This use of hybrid sets also allows us to define a generalised notion of partition, where symbolically defined parts are combined in more useful ways than the usual set operations. Our approach unifies and generalises a number of other techniques, such as the use of oriented regions for domains of integration.

Introducing new operators or generalising existing ones to write single closed form expressions is more than just a cosmetic re-arrangement. It allows one to perform arithmetic and simplifications on whole expressions, and to reason about the expression and about the regions themselves. It is already customary to do this for certain operators. For example, by defining $\int_a^b f \mathrm{d}x = -\int_b^a f \mathrm{d}x$, it becomes possible to write identities that hold universally. Then we have that, independent of the relative order of a, b and c, and subject to f being defined on the requisite domains,

$$\int_a^b f \mathrm{d}x = \int_a^c f \mathrm{d}x + \int_c^b f \mathrm{d}x. \tag{1}$$

With a little work, we can also generalise the integral formula to integrating over oriented subsets of $\mathbb{R}^n$. Some authors similarly adjust the definitions of other operators to obtain universally true statements. Similarly, Karr [8] defines the summation operator $\sum_{m \leqslant i < n}$ so that $\sum_{m \leqslant i < n} = -\sum_{n \leqslant i < m}$ when $n < m$. This allows equations such as the following to hold for any ordering of ℓ, m, n:

$$\sum_{m \leqslant i < n} \big(g(i+1) - g(i)\big) = g(n) - g(m), \qquad \sum_{\ell \leqslant i < n} f(i) = \sum_{\ell \leqslant i < m} f(i) + \sum_{m \leqslant i < n} f(i).$$

This paper formalises and extends these ideas in several ways, giving a generalised framework for domain partitions and piecewise defined functions. We first introduce some preliminaries and hybrid sets (§2) and then their generalisations to our notions of generalised partitions and hybrid functions (§3). We then present how we can decompose domains of hybrid functions to allow for the combination of their symbolically defined pieces (§4), before presenting some applications (§5) and discussing some concrete examples (§6).

2 Preliminaries

2.1 Partitions and piecewise functions

The domain of definition of various mathematical objects (functions, vectors, matrices, sequences, etc) may naturally be decomposed into a (disjoint) union of pieces where our object is then defined uniformly. When the pieces are given as an explicit union of provably disjoint sets which form a partition of the domain, the interpretation of a given expression is reasonably straightforward. More formally,

Notation 1 *We use the notation $\mathcal{C}_{i\in I}X_i$, or, more briefly, $\mathcal{C}_I X_i$ to describe a collection of elements $X_1, X_2, \ldots$, indexed by a set I. For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$.*

Definition 1. *A* partition *of a set U is a collection $\mathcal{C}_I P_i$ of pairwise disjoint sets such that $\bigcup_I P_i = U$.*

A partition, $\mathcal{C}_I P_i$, induces a total function $\mathcal{X} : U \to I$ which gives the index of P_* where each $u \in U$ sits. Piecewise expressions are then defined on top of a partition.

Definition 2. *A* piecewise expression *over a set U is a collection $\mathcal{C}_I(P_i, e_i)$ where $\mathcal{C}_I P_i$ is a partition of U and each e_i is an expression.*

Typically, each e_i contains a distinguished variable y which is interpreted to range over U.

Definition 3. *We say that $f : U \to S$ is a* piecewise-defined function *if we have a collection $\mathcal{C}_I(P_i, f_i)$ where $\mathcal{C}_I P_i$ is a partition of U, $\forall i \in I.\ f_i : P_i \to S$, and*

$$\forall x : U.\ f(x) = f_{\mathcal{X}(x)}(x).$$

We call $\mathcal{C}_I(P_i, f_i)$ the definition *of f, and each f_i a* piece *of f.*

It is important to note that piecewise-defined functions use their argument in two different ways: once *geometrically* by choosing a set P_i "over" which to work, and once *analytically* to evaluate a function f_i. The definitions above are straightforward generalisations of those found in [5].

Lastly, we have that arithmetic operations on piecewise-defined functions operate component-wise. Note that we will silently use the convention that operations defined on the codomain of a function are lifted pointwise to apply to functions, in other words $(f + g)(x) = f(x) + g(x)$.

Proposition 1. *Let $f, g : U \to S$ be two piecewise-defined functions on the same partition $\mathcal{C}_I P_i$ of U, with $\mathcal{C}_I f_i$ (respectively $\mathcal{C}_I g_i$) the collections of pieces of f (resp. g). Further, let $\star : S \times S \to S$. Then $\mathcal{C}_I\,(f_i \star g_i)$ is the collection of pieces of $f \star g$ over the partition $\mathcal{C}_I P_i$.*

Note how the partition is entirely untouched. This "separation of concerns" is what enables us to separate the issues of domain decompositions from arithmetic issues of piecewise-defined functions (and expressions).

To simplify our presentation, we introduce a domain restriction operation and a join combinator on (partial) functions, to allow us a more syntactic method of "building up" piecewise functions. These are quite similar to Kahl's table composition combinators [7].

Definition 4. *The restriction f^A of a function f to a domain specified by a set A is*

$$f^A(x) \ ::= \begin{cases} f(x) & \text{if } x \in A \\ \bot & \text{otherwise} \end{cases}$$

Definition 5. *The join, $f \oplus g$, of two (partial) functions f and g, is defined as*

$$(f \oplus g)(x) \ ::= \begin{cases} f(x) & \text{if } f(x) \text{ is defined and } g(x) \text{ is undefined} \\ g(x) & \text{if } g(x) \text{ is defined and } f(x) \text{ is undefined} \\ \bot & \text{otherwise} \end{cases}$$

This allows us to rewrite a piecewise-defined function f defined by $\mathcal{C}_I(P_i, f_i)$, in terms of its pieces as

$$f = f_1^{P_1} \oplus f_2^{P_2} \oplus \ldots \oplus f_n^{P_n}.$$

But our goal is to work with piecewise-defined functions where we have a *symbolic* partition. We need some new tools for this, which we will develop in the next two sections.

2.2 Hybrid Sets

We consider an extension of multisets, in which elements can occur multiple times, to *hybrid sets*, where the multiplicity of an element in a hybrid set can range over all of $\mathbb{Z}$, instead of just $\mathbb{N}_0$. Thus a hybrid set, over an underlying set U, is a mapping $U \to \mathbb{Z}$, i.e., it is an element of $\mathbb{Z}^U$. We use the following definition, adapted from [9]:

Definition 6. *Given a universe U, any function $H : U \to \mathbb{Z}$ is called a* hybrid set*.*

We can immediately define some useful vocabulary for working with hybrid sets.

Definition 7. *The value of $H(x)$ is said to be the* multiplicity *of the element x. If $H(x) \neq 0$, we say that x is a member of H and write $x \in H$; otherwise, we write $x \notin H$. The* support *of a hybrid set is the (non-hybrid) subset S of U where $s \in S \iff s \in H$; we will denote the support of H by* $\operatorname{supp} H$*. We (re)use $\emptyset$ to (also) denote the empty hybrid set, i.e. the hybrid set for whom all elements have multiplicity* 0*.*

Notation 2 *We use the notation* $\{\!|x_1^{m_1}, x_2^{m_2}, \ldots|\!\}$ *to describe the hybrid set containing elements* x_1 *with multiplicity* m_1*,* x_2 *with multiplicity* m_2*, etc. While our notation allows writing hybrid sets with multiple copies of the same element with different multiplicities, these denote the same hybrid set as that denoted by the normalised form with one copy of each element with a multiplicity which is the sum of the multiplicities of the copies of that element in the non-normalised form. Thus* $\{\!|a^2, b^1, a^{-3}, b^4|\!\} = \{\!|a^{-1}, b^5|\!\}$.

Set unions are usually defined by the boolean algebra structure (and more specifically, via $\vee$) of the membership relation. For hybrid set, this is replaced by arithmetic over $\mathbb{Z}$.

Definition 8. *We define the sum,* $A \oplus B$ *of two hybrid sets* A *and* B *over a universe* U*, to be their pointwise sum. That is* $(A \oplus B)(x) = A(x) + B(x)$ *for all* $x \in U$*. We similarly define their difference,* $A \ominus B$ *to be their pointwise difference, and their product,* $A \otimes B$ *to be their pointwise product. Let* $\ominus B$ *denote* $\emptyset \ominus B$*.*

In other words, we do not use operations $A \cup B$, $A \cap B$ and $A \setminus B$ for hybrid sets, but just $\oplus$, $\ominus$ and $\otimes$. We can easily establish some identities such as $(A \oplus B) \ominus A = B$, $A \ominus A = \emptyset$ and $A \oplus (\ominus B) = A \ominus B$ as these follow directly from $\mathbb{Z}$.

Putting all of this together, we get:

Proposition 2. $\mathbb{Z}^U$ *is a* $\mathbb{Z}$*-module.*

Proof. The abelian group structure is given by $(\mathbb{Z}^U, \oplus, \ominus, \emptyset)$, and $\mathbb{Z}$ acts on hybrid sets by $nH = u \mapsto n \cdot H(u)$.

We need two more technical definitions, which will be useful later.

Definition 9. *We say that two hybrid sets* A *and* B *are* disjoint *if* $A \otimes B = \emptyset$*.*

Definition 10. *We call a hybrid set* reducible *if all its members have multiplicity* 1*. We define a reduction function,* $\mathcal{R}(\cdot)$*, on reducible hybrid sets that returns the (normal) set of members of the hybrid set.*

We should note that these hybrid sets (sometimes also called generalised sets) have been studied before. Hailperin [6] makes the case that Boole [3] actually started from hybrid sets for his algebraization of logic, but restricted himself to nilpotent solutions of the resulting equations, which then correspond closely to our modern notion of Boolean algebra. Whitney wrote two nice papers [13, 14] taking up the theme of algebraising logic via characteristic functions. He does allow arbitrary multiplicities, and derives some nice normal forms for certain kinds of partitions, in a way foreshadowing some of our own results (see §3.2 and §4). Blizard [1] focuses on multisets (disallowing negative multiplicities) but has an extensive bibliography of related works, several of which being on (mechanised) theorem proving; he then formalised sets with negative membership in [2]. Blizard concentrates on concepts of union and intersection which closely resemble those of normal set theory, although he does also define the sum union (but not other related concepts). Burgin [4] lists several more works on hybrid sets, some reaching back to the early middle ages. Syropoulos [12] gives a very readable introduction to both multisets and hybrid sets.

3 Generalisations

We now revisit a few basic mathematical constructs and show how they may be modified to work with hybrid sets. This will provide the machinery that we need for symbolic domain decomposition. First, we will examine the notion of a hybrid function on a domain. We then show how sets and hybrid sets may be decomposed using a notion of generalised partitions — an extension of partitions to the hybrid set case. We then address the practical issue of how to make two hybrid partitions compatible by constructing a common refinement. Finally, when working with functions defined over hybrid partitions, we need some way to compute values. Over any given point in the domain, we need to know which functions must be evaluated in computing the final value, which is rather complex for hybrid functions over generalised partitions. For this task, we introduce the notion of pseudo-functions. When expressions on generalised partitions are evaluated, these pseudo-functions avoid evaluating at places where the functions are undefined or where the values are not needed. This allows us to deal with the cases, as in equation (1), where component functions are not defined on some parts of the domain decomposition but any application of the function in those places would anyway have multiplicity zero (*i.e.* not be used).

3.1 Functions of hybrid domain

It turns out that a useful definition of a "function" involving hybrid sets is not entirely straightforward. Defining its graph is easiest. The underlying intuition is that we capture the restriction of a function to a domain through the multiplicities of the elements of the function graph in a hybrid set. The hybrid set of a single element of the function graph for element x in U is of the form $\{|(x, f(x))^1|\}$. Therefore the scalar multiplication of that set by the multiplicity of x in A will impose the appropriate restriction. We use our function restriction notation of Def. (4) only for this hybrid version of function restriction henceforth.

Definition 11. *Let A be a hybrid set over U, $B \subseteq U$, S a set and $f : B \to S$ a (total-on-B) function.* A hybrid function $f^A : U \times S \to \mathbb{Z}$ *is defined by*

$$f^A = \bigoplus_{x \in B} A(x)\{|(x, f(x))^1|\}$$

Note how the hybrid set $\mathbb{Z}$-module structure automatically takes care of restricting the sum over the support of A. Caution: some hybrid sets do not form a hybrid function (for example $\{|(1,1)^1, (1,2)^1|\}$ is not a hybrid function).

Our definitions work just as well with partial functions as with total functions. But if for a hybrid function f^F, f is undefined at some point in the support of F, f^F will not be defined at that point either. So, without loss of generality, we can always restrict F to where f is defined. For the remainder of this paper, we shall assume this, i.e. whenever we write a hybrid function f^F, f is total over $\operatorname{supp} F$.

Definition 12. *We call a hybrid function* f^H reducible *if the hybrid set* H *is reducible. We extend* $\mathcal{R}(\cdot)$ *in this case by*

$$\mathcal{R}(f^H)(x) = \begin{cases} f(x) & \text{if } H(x) = 1 \\ \bot & \text{if } H(x) = 0 \end{cases}$$

We can generalise the join combinator to hybrid sets. This definition is quite central to "making things work".

Definition 13. *The join,* $f^F \oplus g^G$*, of two hybrid functions* f^F *and* g^G *(with codomain* B*), gives a* hybrid relation*, a subset of* $U \times B \times \mathbb{Z}$ *given by*

$$f^F \oplus g^G \ ::= \ f^F \oplus g^G$$

This is a rather "dangerous" definition, as it moves us from the land of functions to that of relations. In other words, it is quite possible that $f^F \oplus g^G$ restricted to $U \times B$ is no longer the graph of a function, but the graph of a relation. But this extra generality will be quite useful for us, although we will have to prove that in the cases which interest us, the resulting hybrid relations are in fact (reducible) hybrid functions.

Theorem 1. *Let* A*,* B *be hybrid sets over* U*,* S *an arbitrary set, and* $f : U \to S$ *a total function. Then*

1. $\mathcal{R}(f^\emptyset)$ *is the empty function,*
2. $f^A \oplus f^A = f^{2A}$
3. $f^A \oplus f^B = f^{A \oplus B}$*, and thus a hybrid function,*
4. *For* $g : U \to S$ *another total function, then* $f^A \oplus g^B = (f \oplus g)^{A \oplus B}$ *if and only if* $A \oplus B = \emptyset$ *(where* $f \oplus g$ *is the join of regular functions).*
5. *Let* H_1, H_2 *be hybrid sets, with* $\operatorname{supp} H_1$ *and* $\operatorname{supp} H_2$ *disjoint,* $f_1 : \operatorname{supp} H_1 \to S$ *and* $f_2 : \operatorname{supp} H_2 \to S$*, then* $f_1^{H_1} \oplus f_2^{H_2} = (f_1 \oplus f_2)^{H_1 \oplus H_2}$

The proofs are omitted, and follow straightforwardly from the definitions. Note the strong dichotomy between (3) and (4), which comes from the fact that the non-hybrid $\oplus$ is designed to work with functions defined over separate regions.

3.2 Generalised Partitions

Theorem 1 tells us that some collections of hybrid sets are better than others. Being disjoint is much too strong a property. Nicely, for hybrid sets, partitions easily generalise in useful ways.

Definition 14. *We define a generalised partition of a (hybrid) set,* P*, to be a finite collection of hybrid sets,* $\mathcal{C}_{[n]} P_i$*, such that* $P_1 \oplus P_2 \oplus \ldots \oplus P_n = P$

All set partitions of a set are also generalised partitions. Conversely, a generalised partition of a reducible set is a set partition if and only if each generalised partition element is reducible.

Remark 1. We have lifted the *disjointness* condition on partitions. For something to be called a partition of P, it is necessary that the result be equal to P. Still, P belongs to a larger universe U, and a generalised partition's pieces range over U. As long as, in the end, all elements of $U \setminus P$ have multiplicity 0, we get a generalised partition. In this way, we have also lifted the *coverage* condition.

Proposition 3. *For any generalised partition $\mathcal{C}_{[n]} P_i$ of a hybrid set P over U, arbitrary set S, and any function $f : \operatorname{supp} P \to S$,*

$$f^P = f^{P_1} \oplus f^{P_2} \oplus \ldots \oplus f^{P_n} = f^{P_1 \oplus P_2 \oplus \ldots \oplus P_n}$$

is a hybrid function.

For brevity, we sometimes write f^P for either of the right-hand side expressions above. When we want to join different functions over a partition and still get a function, we have to be careful and ensure we are joining "compatible" functions.

Definition 15. *Let P_1, P_2 be a generalised partition of a hybrid set P over U, S an arbitrary set and $f^{P_1} : P_1 \to S$, $g^{P_2} : P_2 \to S$ hybrid functions. We say that P_1, P_2 is a* compatible *partition for f, g if $f(x) = g(x)$ for all $x \in \operatorname{supp} P_1 \cap \operatorname{supp} P_2$.*

Theorem 2. *Using the same notation as above, $f^{P_1} \oplus g^{P_2}$ is a hybrid function if and only if P_1, P_2 is a compatible partition for f, g.*

It is important to note that although $\oplus$ is an associative, commutative operation, the notion of compatibility, while commutative, does not lift to a simple associative condition.

Remark 2. Some of our computations will purposefully use *incompatible* partitions. Note that

$$(f^U \oplus g^U) \oplus g^{\ominus U} = f^U \oplus (g^U \oplus g^{\ominus U}) = f^U \oplus g^{\emptyset} = f^U$$

but that $f^U \oplus g^U$ is in general a hybrid relation, yet the final result is a hybrid function whenever f^U is. We will "design" our hybrid partitions with this feature in mind.

3.3 Refinement

To do arithmetic with hybrid functions, we first need the notion of a refinement and a common refinement. This is similar to the treatment of [5] for piecewise functions.

Definition 16. *A* refinement *of a generalised partition $\mathcal{C}_I P_i$ of P is another generalised partition $\mathcal{C}_J Q_j$ (of another hybrid set Q not necessarily equal to P) such that for every $i \in I$ there exists a sub-collection Q_{j_k} of Q_j such that $P_i = Q_{j_1} \oplus Q_{j_2} \oplus \ldots \oplus Q_{j_m}$. A* common refinement *of a set of generalised partitions is a generalised partition that is simultaneously a refinement to every partition in the set.*

A refinement in this sense may well seem "larger" than the original partition, as in the next example.

Example 1. Let the interval $P = [0,1]$, seen as $\{|P^1|\}$, and $I_1 = [-1,0)$, $I_2 = (1,2]$, $I_3 = [-1,2]$, then $Q = \{|I_1^{-1}, I_2^{-1}, I_3^1|\}$ is a refinement of P.

Definition 17. *A refinement is called* strict *if, for each generalised partition being refined, the support of the associated sub-collection is equal to the support of the generalised partition it refines.*

Example 2. $\{|[0,1]^1|\}, \{|(1,2]^1|\}, \{|(2,3]^1|\}$ is a common strict refinement of the two (trivial) hybrid partitions $\{|[0,2]^1|\}$ and $\{|(1,3]^1|\}$.

3.4 Pseudo-functions and pseudo-relations

As seen in the example above, a refinement may "spill over" the original domain, so that if we look at a hybrid function f^P where the underlying f is defined exactly on (the support of) P, f^Q evaluated "pointwise" will not make sense. Nevertheless, we want $f^P = f^Q$. To achieve this, we apply the lambda-lifting trick already used in [5].

Definition 18. *Using the same notation as in Def. 11, we define a* pseudo-function $\tilde{f}^A$ *as*

$$\tilde{f}^A = \bigoplus_{x \in B} A(x)\{|(x,f)^1|\}$$

i.e. as a member of $U \times (U \to S) \to \mathbb{Z}$. *A* pseudo-relation *is defined similarly. The* evaluation *of a pseudo-function (resp. relation) is defined by mapping each point* $(x,f)^k$ *to* $(x,f(x))^k$.

The usefulness of pseudo-functions comes from the following property.

Proposition 4. *For all refinements* Q *of the generalised partition* P, $\tilde{f}^P = \tilde{f}^Q$.

In other words, even though the pieces of Q might "spill over", if we first "simplify" $\tilde{f}^Q$ by performing $\bigoplus_i Q_i$ to get P, we get a result $\tilde{f}^P$ which can then be safely evaluated. We will elide the $\tilde{}$ to lighten the notation whenever this would not lead to confusion.

Another useful property of pseudo-functions is that in some cases we can simplify them, regardless of what the underlying function does.

Proposition 5. $\tilde{f}^P \oplus \tilde{g}^Q \oplus \tilde{g}^{\ominus Q} = \tilde{f}^P$

One of the chief advantages of pseudo-functions is that we can do some symbolic manipulations of expressions in terms of these functions as if they were defined on a much larger domain, as long as the eventual term we evaluate does not involve any of these "virtual" terms.

To aid in such computations, when we have pseudo-functions $\tilde{f}^P$ and $\tilde{g}^P$, with $f, g : \operatorname{supp} P \to S$, and some binary operation $\star : S \times S \to S$, we will allow

ourselves to write expressions such as $\tilde{f}^P \star \tilde{g}^P$ in the induced term algebra over pseudo-functions. As usual, we lift evaluation pointwise, $(\tilde{f} \star \tilde{g})(x) = f(x) \star g(x)$. Furthermore we say that $\tilde{f} = \tilde{g}$ over a set $\operatorname{supp} P$ whenever $\forall x \in \operatorname{supp} P. f(x) = g(x)$. In other words, we use extensional equality for the intensional terms $\tilde{f}$ and $\tilde{g}$. This means that properties like commutativity, associativity and having inverses lift to the term algebra. As an example, we have that

Proposition 6. *If $\star : S \times S \to S$ is associative and commutative then*

$$\tilde{g}^Q \star \tilde{f}^P \star \tilde{g}^{\ominus Q} = \tilde{f}^P$$

4 Hybrid domain decomposition

We now have all the ideas necessary to decompose hybrid domains. An elegant consequence of the formalism is that it allows us to use linear algebra to construct the partitions that we require.

Let $\mathcal{C}_{[n]} A_i$ and $\mathcal{C}_{[m]} B_j$, be generalised partitions of U. We want to find a generalised partition of U that is a common strict refinement of A_i and B_j and has minimal cardinality. The cardinality restriction is to minimise the number of terms required for a symbolic representation of the resulting domain decomposition. Thus we want to choose a minimal generalised partition, $\mathcal{C}_I P_i$ of U such that, in the $\mathbb{Z}$-module of hybrid sets and for some integers $a_{i,j}$, $b_{i,j}$ we have $\bigoplus_i P_i = U$ and $\forall i : 1..n.\ \bigoplus_j a_{i,j} P_j = A_i$ and $\forall j : 1..m.\ \bigoplus_i b_{i,j} P_i = B_j$.

Since this forms a system of $n + m + 1$ simultaneous equations, of which only $n + m - 1$ can be independent, because A and B are, separately, partitions of U, we need that number of independent variables to solve the system. Thus the cardinality of the minimal partition is $n + m - 1$ in the general case, and can only be smaller in specific cases if there are some extra dependencies among $U, A_1, \ldots, A_{n-1}, B_1, \ldots, B_{m-1}$.

This result generalises to a decomposition of U into a minimal generalised partition that is a common strict refinement of r generalised partitions of cardinality $n_1, \ldots n_r$ respectively: The minimal partition required, assuming full independence of the individual domain partitions, has cardinality

$$\left(\sum_{i=1}^{r} n_i \right) + 1 - r$$

If all the individual partitions have the same cardinality, n, this reduces to $r(n-1) + 1$.

We can automatically compute suitable minimal strict refinement partitions U as follows. If we remove the equations for A_n and B_m from the system of equations in order to get an independent set of simultaneous equations, we get

$n+m-1$ equations that can be written as a linear system in the $\mathbb{Z}$-module:

$$C \cdot \begin{pmatrix} P_1 \\ \vdots \\ P_{n+m-1} \end{pmatrix} = \begin{pmatrix} U \\ A_1 \\ \vdots \\ A_{n-1} \\ B_1 \\ \vdots \\ B_{m-1} \end{pmatrix} \quad \text{where } C = \begin{pmatrix} 1 & 1 & \dots & 1 \\ a_{1,1} & a_{1,2} & \dots & a_{1,n+m-1} \\ \vdots & & & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n+m-1} \\ b_{1,1} & b_{1,2} & \dots & b_{1,n+m-1} \\ \vdots & & & \vdots \\ b_{m-1,1} & b_{m-1,2} & \dots & b_{m-1,n+m-1} \end{pmatrix}$$

Note that C is an integer matrix. Further, the partition choice matrix, C, must be invertible and C^{-1} must be an integer matrix so that we obtain integral partitions of each domain piece with respect to our partition P of U. An integer matrix is invertible and has an integer matrix inverse if and only if it has a determinant of $+1$ or -1. Hence our problem of constructing an appropriate partition reduces to choosing an integer matrix of the form of C such that its determinant is ± 1. Finally, note that this directly generalises to an arbitrary number of piecewise functions, each of an arbitrary number of pieces.

If we restrict ourselves to triangular matrices, we can choose C to be any integer triangular matrix (upper because the first row of C is all 1s) for which the product of the diagonal elements is 1. Again, a simple way to do this is to choose C to be all 1s along the top row, 1s along the diagonal and 0 everywhere else. Another possibility is all 1s in the whole upper triangle.

For example, two suitable choice matrices with their inverses are

$$\begin{pmatrix} 1 & \dots & \dots & 1 \\ 0 & 1 & 0 & \dots 0 \\ \vdots & \ddots & \ddots & \ddots \vdots \\ \vdots & & \ddots & \ddots 0 \\ 0 & \dots & \dots & 0 \ 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -1 & \dots & \dots & -1 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & \dots & \dots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}$$

5 Applications

5.1 Arithmetic on Piecewise Functions

We are now ready to generalise the arithmetic properties (see prop. 1) of hybrid functions and pseudo-functions.

Proposition 7. *Let $\mathcal{C}_{[n]}P_i$ be a partition of P, $f^P = f_1^{P_1} \oplus \dots \oplus f_n^{P_n}$ and $g^P = g_1^{P_1} \oplus \dots \oplus g_n^{P_n}$ be two hybrid functions on $P \to S$. Let $\star : (S \times S) \to S$, then for all $x \in \operatorname{supp} P$,*

$$f^P(x) \star g^P(x) = (f_1(x) \star g_1(x))^{P_1} \oplus \dots \oplus (f_n(x) \star g_n(x))^{P_n}.$$

Note how we can apply the above proposition to $f^F \star g^G$, by first restricting F and G to be over a common support (as $x \star \bot = \bot \star y = \bot$), then taking a common *strict* refinement of (the restricted) F and G.

We would like to lift this strictness condition. We can almost do this with pseudo-functions – and with the help of a marked $\oplus$, we can.

Definition 19. *We can* mark *a* $\oplus$ *with a binary operation* $\star : S \times S \to S$, *denoted* $\oplus^\star$. *We define evaluation of* $\oplus^\star$ *by*

$$(\tilde{f}^F \oplus^\star \tilde{g}^G)(x) = (F(x) + G(x))\left\{\left|(x, (\tilde{f} \star \tilde{g})(x))^1\right|\right\}$$

This operation clearly inherits properties of $\star$ like commutativity, associativity and invertibility. Unlike $\oplus$, $\oplus^\star$ will always result in a (pseudo) function.

Proposition 8. *Let* $P_* = \mathcal{C}_{[n]} P_i$ *be a partition of* P, $Q_* = \mathcal{C}_{[m]} Q_j$ *another partition of* P, *and* $R_* = \mathcal{C}_K R_k$ *a common refinement of* P_* *and* Q_*. *Let* $\tilde{f}^P = f_1^{P_1} \oplus \ldots \oplus f_n^{P_n}$ *and* $\tilde{g}^P = g_1^{Q_1} \oplus \ldots \oplus g_m^{Q_m}$ *be two* pseudo *functions with codomain* S. *Let* $\star : (S \times S) \to S$, *be associative and commutative, then* $\star$ *distributes over the partition* R_* *in terms of* $\oplus^\star$. *By the results of section 4, we can always choose* R_* *to be*

$$P_1 \oplus \ldots \oplus P_{n-1} \oplus Q_1 \oplus \ldots \oplus Q_{m-1} \oplus (U \ominus (P_1 \oplus \ldots \oplus P_{n-1} \oplus Q_1 \oplus \ldots \oplus Q_{m-1}))$$

Example 3. Let $A_1 = [0, a), A_2 = [0, 1] \setminus A_1, B_1 = [0, b), B_2 = [0, 1] \setminus B_1$, all seen as hybrid sets. Let

$$f(x) = \begin{cases} 2 & 0 \le x < a \\ 0 & a \le x < 1 \end{cases} \qquad \text{and} \qquad g(x) = \begin{cases} 5 & 0 \le x < b \\ 7 & b \le x < 1 \end{cases}$$

We choose the hybrid (symbolic!) partition $A_1, B_1 \ominus A_1, B_2$, which simultaneously refines both. Then after a few computations we get

$$f * g = \{\!|2 * 5|\!\}^{A_1} \oplus^* \{\!|2 * 5|\!\}^{B_1 \ominus A_1} \oplus^* \{\!|0 * 7|\!\}^{B_2} \tag{2}$$

regardless of whether $a < b$ or $a \ge b$; in fact either (or both) could be outside of $[0, 1)$ and the result, interpreted as a hybrid function, are still correct. Moving from one choice of partition to another is done by undoing the distribution, performing the change of partition, and using commutativity and associativity to regroup like terms. Note that we should have written $2 * 5$ as $(x \mapsto 2) \tilde{*} (x \mapsto 5)$, but we chose the above for greater clarity.

It should be very clear that expressions such as equation 2 are a formal representation of a piecewise function, and need to be *interpreted* properly in each context.

5.2 Identities for invertible operators

When the binary operation we use is invertible, we can perform the operations at any time, as operands may later be removed from a cumulative result by applying their inverses. This may lead to considerable efficiency improvements; even though values and their inverses are cancelled in the calculation (leading to no net effect) this may be more efficient than retaining an "un-evaluable" expression as a pseudo-function.

Proposition 9. *Let f^P be a hybrid function over S where $(S, \star)$ has the structure of an Abelian group (where we will use e for the unit and $-$ for the inverse), then for all $x \in \operatorname{supp} P$,*

$$(f^P \star f^{-P})(x) = (f^P \star (-f)^P)(x) = P(x)\{|(x, e)^1|\}$$

where $-f$ denotes $x \mapsto -f(x)$.

Both equalities follow readily from the definitions.

5.3 Identities for linear operators

Definition 20. *For a linear operator L, and f^P a hybrid function,*

$$L(f^P) \ ::= \ L(x \mapsto P(x) \cdot f(x))$$

Note $L(f^P)$ may not be defined even when $L(f)$ is. If the multiplicity function $P(x)$ is uniformly bounded, then it will exist.

Proposition 10. *Let f^P be a hybrid function, $\mathcal{C}_{[n]}P_i$ a partition of P such that each $P_i(x)$ is uniformly bounded, then*

$$L(f^P) = \sum_{i=1}^{n} L(f^{P_i})$$

The above is the fundamental reason why, under Karr's definition, the summation identities of the introduction hold.

Corollary 1. *For all total functions $f : \mathbb{Z} \to G$ with G an Abelian group, and all $\ell, m, n \in \mathbb{Z}$,*

$$\sum_{\ell \leqslant i < n} f(i) = \sum_{\ell \leqslant i < m} f(i) + \sum_{m \leqslant i < n} f(i).$$

6 Examples

We present two examples of the application of hybrid functions to symbolic computation problems. The first example is concerned with the arithmetic of symbolic matrices, the second presents the idea of merging symbolic spline functions.

6.1 Matrix Addition

Earlier work [10, 11] introduced The idea of support functions has been introduced previously to represent symbolic matrices — matrices given in terms of symbolic regions with underspecified elements and symbolic dimensions — and defined arithmetic operations between them. The paper [10] presented a support function based on half-plane constraints that enables full arithmetic, but that suffered from a combinatorial explosion in the number of terms needed to express sum or product matrices. The paper [11] moved to a support function based on interval addition that automatically dealt with cancellation for negative intervals. While this avoided the combinatorial explosion, the approach was restricted to certain types of regions and could not be easily generalised to matrix multiplication. Hybrid functions and generalised partitions solve both of these problems simultaneously. We demonstrate this with the example of matrix addition of two 2×2 symbolic block matrices. Let

$$M_1 = \begin{pmatrix} A_1 & B_1 \\ C_1 & D_1 \end{pmatrix}, \qquad M_2 = \begin{pmatrix} A_2 & B_2 \\ C_2 & D_2 \end{pmatrix}$$

where M_1 and M_2 are $n \times m$ matrices, A_1 and A_2 are of dimensions $h_1 \times k_1$ and $h_2 \times k_2$ respectively, $n, m, h_1, h_2, k_1, k_2 \in \mathbb{N}_0$.

Let $U = \{(i, j) \mid 1 \leqslant i \leqslant n \wedge 1 \leqslant j \leqslant m\}$ be the set of all cell points in the matrices. We define the region occupied by a matrix block similarly, and refer both to a matrix block and to the region it occupies by the same name, relying on context to distinguish them. We can thus write

$$M_1 = A_1^{A_1} \oplus B_1^{B_1} \oplus C_1^{C_1} \oplus D_1^{D_1}, \qquad M_2 = A_2^{A_2} \oplus B_2^{B_2} \oplus C_2^{C_2} \oplus D_2^{D_2} \tag{3}$$

To calculate $M_1 + M_2$ we: (1) Choose a suitable generalised partition P_* of U. (2) Rewrite each block of each matrix into terms restricted to the chosen partition. (3) Substitute into the expressions for the matrices. (4) Add the two matrices region-wise.

As established in the section 4, the maximal number of partitions required in our case is $4 + 4 - 1 = 7$. We therefore choose 6 independent regions to be $A_1, B_1, C_1, A_2, B_2, C_2$ and obtain the seventh, P_1, by subtracting all other regions from the universe U,

$$P_1 = U \ominus (A_1 \oplus B_1 \oplus C_1 \oplus A_2 \oplus B_2 \oplus C_2). \tag{4}$$

We can then express regions D_1 and D_2 in terms of P_1: $D_1 = U \ominus (A_1 \oplus B_1 \oplus C_1) = P_1 \oplus A_2 \oplus B_2 \oplus C_2$, and $D_2 = P_1 \oplus A_1 \oplus B_1 \oplus C_1$.

Rewriting M_1, M_2 from Eq. (3), we get:

$$\begin{aligned} M_1 &= A_1^{A_1} \oplus B_1^{B_1} \oplus C_1^{C_1} \oplus D_1^{P_1 \oplus A_2 \oplus B_2 \oplus C_2} \\ &= D_1^{P_1} \oplus A_1^{A_1} \oplus B_1^{B_1} \oplus C_1^{C_1} \oplus D_1^{A_2} \oplus D_1^{B_2} \oplus D_1^{C_2} \\ M_2 &= D_2^{P_1} \oplus D_2^{A_1} \oplus D_2^{B_1} \oplus D_2^{C_1} \oplus A_2^{A_2} \oplus B_2^{B_2} \oplus C_2^{C_2} \end{aligned}$$

This lets us express the sum of the two matrices as the following pseudo-function:

$$M_1 + M_2 = (D_1 + D_2)^{P_1} \oplus^+ (A_1 + D_2)^{A_1} \oplus^+ (B_1 + D_2)^{B_1} \oplus^+ (C_1 + D_2)^{C_1} \\ \oplus^+ (D_1 + A_2)^{A_2} \oplus^+ (D_1 + B_2)^{B_2} \oplus^+ (D_1 + C_2)^{C_2} \quad (5)$$

Observe that the seven terms of the hybrid function fully capture the result of the matrix addition independent of the order of the symbolic dimensions h_1, h_2, k_1, k_2 of the original blocks. We demonstrate this by evaluating the function for a couple of concrete points in the sum matrix.

First let $h_1 < h_2$ and choose a concrete value of a cell (i, j) where $h_1 < i \leq h_2$ and $1 \leq j < k_1, k_2$. The point should therefore be in a region composed of elements from B_1 and A_2. Instantiating the multiplicities in equation (5) verifies this. Observe that indeed the only regions with multiplicity 1 are $B_1 = \{(i, j) \mid h_1 \leqslant i \leqslant n \wedge 1 \leqslant j \leqslant k_1\}$ and $A_2 = \{(i, j) \mid 1 \leqslant i \leqslant h_2 \wedge 1 \leqslant j \leqslant k_2\}$ whereas the multiplicities for A_1, B_2, C_1, C_2 are all 0. Furthermore, we can compute the multiplicity for P_1 using equation (4). Since the multiplicity of the universe U is always 1 — every element is in this partition — we get $1 - (0 + 1 + 0 + 1 + 0 + 0) = -1$. This then yields the computation below, which confirms that our element is indeed in the anticipated region (where we write region-wise sets $\left\{\left|(D_1 + D_2)^{-1}\right|\right\}$ as $(D_1 + D_2)^{-1}$ to alleviate notation)

$$(D_1 + D_2)^{-1} \oplus^+ (B_1 + D_2)^1 \oplus^+ (D_1 + A_2)^1 = B_1^1 \oplus^+ A_2^1 = B_1 + A_2$$

Now assume that the order of the symbolic dimensions for blocks A_1, A_2 is reversed and we have $h_2 < h_1$. If we now compute the value of a cell with (i, j) with $h_2 < i \leq h_1$ and $1 \leq j < k_1, k_2$, we get 0 multiplicity for regions B_1, A_2, but instead multiplicity 1 for A_1, B_2. Again the multiplicity for P_1 is -1 and equation (5) will yield that our cell is in the $A_1 + B_2$ region.

As a final example we compute cell (i, j) with $h_1, h_2 < i \leq n$ and $k_1, k_2 < j \leq m$, which is a point from D_1 and D_2. This time equation (5) simplifies even more quickly, as the multiplicities for $A_1, A_2, B_1, B_2, C_1, C_2$ are all 0 and we only have to consider the multiplicity for P_1 which is 1 with the same considerations as above, yielding $D_1 + D_2$ as the only term that does not vanish.

6.2 Symbolic Spline Interpolation

Numerical computation and manipulation of splines is well understood. Here we show how hybrid domain decomposition can be used to support the previously unexplored symbolic manipulation of splines.

Let $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ be a partition of the interval $[a, b] \subset \mathbb{R}$. We call the x_i knots and assume that for each knot we have a corresponding value y_i. Then we can define a spline function S over $[a, b]$ piecewise as

$$S(x) = \begin{cases} S_0(x) & x \in [x_0, x_1] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [x_{n-1}, x_n] \end{cases} \quad (6)$$

While spline interpolation is traditionally defined for numerical values of the x_i, y_i pairs, we will now define a symbolic spline function. Let $a = c_0 < c_1 < \cdots < c_{n-1} < c_n = b$ be symbolic or "abstract" knots with associated symbolic values $d_0, d_1, \ldots, d_n$, where a d_i is generally given as a function in c_i. We define spline segments $S_{c_i,c_{i+1}}(x)$ for $i = 0, \ldots, n-1$. That is, the segments are parameterised with respect to two knots and their values. Let $P = P_1 \oplus \ldots \oplus P_n$ with $P_i = [c_{i-1}, c_i]$ be a generalised partition. We then define a symbolic spline function $S(x) = S^{P_1}_{c_0,c_1}(x) \oplus \cdots \oplus S_{c_{n-1},c_n}(x)^{P_n}$. For clarity we often omit the (x) part of the term.

Define the merge of two spline segments $S^P_{a,b} \bowtie S^Q_{a',b'}$ to be $S^{P\otimes Q}_{max(a,a'),min(b,b')}$. Clearly this merge will be empty if the two segments do not overlap, otherwise it will be the smallest possible spline for the overlapping interval of the segments.

Now let $P_1 \oplus \cdots \oplus P_n$ and $Q_1 \oplus \cdots \oplus Q_m$ be two generalised partitions of the interval $[a, b]$ and let $S = S^{P_1}_{c_0,c_1} \oplus \cdots \oplus S^{P_n}_{c_{n-1},c_n}$ and $T = T^{Q_1}_{d_0,d_1} \oplus \cdots \oplus T^{Q_m}_{d_{m-1},d_m}$ be two symbolic splines. Observe that the d_i here are knots and not knot values. We can then define the merge $S \bowtie T$ as a binary operation on two hybrid functions as given above in proposition 8.

We observe the merge operation using a simple example. Let $P = P_1 \oplus P_2$ and $Q = Q_1 \oplus Q_2$ be the generalised partitions $a < c < b$ and $a < d < b$ of our universe $[a, b]$, respectively. Let $S = S^{P_1}_{a,c} \oplus S^{P_2}_{c,b}$ and $T = T^{Q_1}_{a,d} \oplus T^{Q_2}_{d,b}$ be two symbolic splines. We choose a common refinement as $P_1, Q_1, R = U \ominus (P_1 \oplus Q_1)$. We can then write $S = S^{P_1}_{a,c} \oplus S^{R\oplus Q_1}_{c,b} = S^{P_1}_{a,c} \oplus S^{R}_{c,b} \oplus S^{Q_1}_{c,b}$ and similarly $T = T^{Q_1}_{a,d} \oplus T^{P_1}_{d,b} \oplus T^{R}_{d,b}$. When we merge both symbolic splines we get

$$S \bowtie T = (S_{a,c} \bowtie T_{d,b})^{P_1} \oplus^{\bowtie} (S_{c,b} \bowtie T_{a,d})^{Q_1} \oplus^{\bowtie} (S_{c,b} \bowtie T_{d,b})^{R} \tag{7}$$

If we now fix the order of our symbolic knots to be $a < c < d < b$ we can evaluate the three components of our spline. First let $a \le x \le c$, which means $P_1 = Q_1 = 1$ and $R = -1$ and (7) evaluates to $S_{a,c} \bowtie T_{a,d}$ which yields a spline between knots a, c. Similarly the other two segments evaluate $S_{c,b} \bowtie T_{a,d}$ and $S_{c,b} \bowtie T_{d,b}$, which yields splines for the intervals $[c, d]$ and $[d, b]$, respectively.

7 Conclusion

We have presented a framework of generalised partitions and domain decomposition based on hybrid sets. This framework has a number of pleasing properties and unifies a number of *ad hoc* notions in common use. More importantly for our purposes, this representation allows easy manipulation of and reasoning about partitions whose pieces are defined symbolically. These specialise correctly for all choices of parameters by negative and positive multiplicities cancelling as needed. The representation (see for example equation (2)) needs to be carefully interpreted, as out of context simplification can result in meaningless results. But if the rules we lay out are followed, our compact representation effectively allows one to compute with very general piecewise-defined functions.

Although a number of previous authors have studied hybrid sets, our study of functions over hybrid sets, generalised partitions, the superposition $\oplus$ and

marked superposition $\oplus^\star$ operators appear to all be new. Our applications to computation and reasoning are certainly new.

There remain several intriguing directions for future work. These include normal forms for piecewise functions defined on hybrid partitions, simplification of intermediate expressions involving linear operators or inverse elements and creating partition schemes from algebraic specialisation properties, *e.g.* in Cylindrical Algebraic Decomposition. We have implemented a prototype Maple package for hybrid sets and hybrid functions, but a more general implementation would be of interest.

References

1. Wayne D. Blizard. Multiset theory. *Notre Dame Journal of Formal Logic*, 30(1):36–66, Winter 1989.
2. Wayne D. Blizard. Negative membership. *Notre Dame Journal of Formal Logic*, 31(3):346–368, 1990.
3. George Boole. *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities.* Walton and Maberly, London, 1854. Available at `http://www.gutenberg.org/etext/15114`.
4. M.S. Burgin. Concept of multisets in cybernetics. *Cybernetics and Systems Analysis*, 28(3):469–371, 1992.
5. J. Carette. A canonical form for piecewise defined functions. In *Proc. of ISSAC 2007*, pages 77–84. ACM Press, 2007.
6. Theodore Hailperin. *Boole's Logic and Probability.* North-Holland Publishing Company, $1^{st}/2^{nd}$ edition, 1976/1986.
7. W. Kahl. Compositional syntax and semantics of tables. SQRL Report 15, Department of Computing and Software, McMaster University, 2003. available from `http://www.cas.mcmaster.ca/sqrl/sqrl_reports.html`.
8. M. Karr. Summation in finite terms. *J. ACM*, 28(2):305–350, 1981.
9. D. Loeb. Sets with a negative number of elements. *Advances in Mathematics*, 91(1):64–74, 1992.
10. A.P. Sexton, V. Sorge, and S.M. Watt. Computing with abstract matrix structures. In *Proc. of ISSAC'2009*, pages 325–332. ACM Press, 2009.
11. A.P. Sexton, V. Sorge, and S.M. Watt. Reasoning with generic cases in the arithmetic of abstract matrices. In *Proc. of Calculemus 2009*, LNAI 5625. Springer Verlag, 2009.
12. A. Syropoulos. Mathematics of multisets. In Cristian Calude, Gheorghe Paun, Grzegorz Rozenbeg, and Arto Salomaa, editors, *Multiset Processing*, volume 2235 of *LNCS*, pages 347–358. Springer, 2001.
13. Hassler Whitney. A logical expansion in mathematics. *Bulletin of the American Mathematical Society*, 34(8):572–579, August 1932.
14. Hassler Whitney. Characteristic functions and the algebra of logic. *Annals of Mathematics*, 34(3):405–414, July 1933. `http://www.jstor.org/stable/1968168`.