Francesco M. Malvestuto

The sum-product algorithm: algebraic independence and computational aspects

Kybernetika, Vol. 49 (2013), No. 1, 4--22

Persistent URL: http://dml.cz/dmlcz/143237

KYBERNETIKA — VOLUME *49* (2013), NUMBER 1, PAGES 4–22

THE SUM-PRODUCT ALGORITHM: ALGEBRAIC INDEPENDENCE AND COMPUTATIONAL ASPECTS

Francesco M. Malvestuto

The sum-product algorithm is a well-known procedure for marginalizing an "acyclic" product function whose range is the ground set of a commutative semiring. The algorithm is general enough to include as special cases several classical algorithms developed in information theory and probability theory. We present four results. First, using the sum-product algorithm we show that the variable sets involved in an acyclic factorization satisfy a relation that is a natural generalization of probability-theoretic independence. Second, we show that for the Boolean semiring the sum-product algorithm reduces to a classical algorithm of database theory. Third, we present some methods to reduce the amount of computation required by the sum-product algorithm. Fourth, we show that with a slight modification the sum-product algorithm can be used to evaluate a general sum-product expression.

Keywords: sum-product algorithm, distributive law, acyclic set system, junction tree

Classification: 47A67, 62-09, 62c10, 68P15, 68W30

1. INTRODUCTION

Marginalizing a product function whose range is the ground set of a commutative semiring is a ubiquitous problem with applications in information theory, probability theory and artificial intelligence. Finding a solution algorithm with a minimum amount of computation (additions and multiplications) is an unavoidable task. In two landmark papers [1, 6], the authors proposed an algorithm, henceforth referred to as the sum-product algorithm (SPA, for short) as in [6], which efficiently solves the marginalization problem for a function defined by an acyclic product of factors and is general enough to include as special cases several classical algorithms developed in information theory (e. g., the Baum–Welch algorithm, the Fast Fourier transform on any finite Abelian group, Viterbi's algorithm, the Gallager–Tanner–Wiberg algorithm, the Bahl–Cocke–Jelinek–Raviv algorithm, the discrete-state Kalman filtering) and in probability theory (e. g., Pearl's algorithm and the Shafer–Shenoy algorithm). The SPA consists of a message-passing procedure in a suitable graphical representation (the "junction tree" in [1], and the "factor graph" in [6]) of the factorization. In this paper, we present four results. First, we prove that for an acyclic factorization the variable sets that are the arguments of factors satisfy a relation that is a natural generalization of probability-theoretic independence. Second, we show that in the Boolean semiring the SPA reduces to a classical algorithm developed in database theory. Third, we present some techniques to reduce the amount of computation required by the SPA. Fourth, we show how to modify the SPA in order to evaluate a general sum-product expression, that is, to compute any marginal of a product function.

Here is an outline of the paper. Section 2 contains the definition of an acyclic set system with some graph-theoretic properties. In Section 3 we review the SPA. Section 4 is devoted to the proof of independence of the variable sets involved in an acyclic factorization. In Section 5 we deal with the Boolean version of the SPA and show the usefulness of the notion of the "support" of a Boolean function to reduce the amount of computation required by the SPA. The support of a general function is also used in Section 6 to obtain an efficient implementation of the SPA, which in the best case makes the output size equal to the input size. In Section 7 we show that the problem of evaluating a general sum-product expression can be solved using a slightly modified version of the SPA. Section 8 contains some closing notes.

2. PRELIMINARIES

Henceforth, $(R, +, \cdot)$ stands for a commutative semiring, that is, $(R, +)$ and $(R, \cdot)$ are both commutative semigroups whose identities are denoted by 0 and 1, respectively; moreover, multiplication distributes over addition and multiplication by 0 annihilates every element of R, that is,

$$u \cdot (v + w) = u \cdot v + u \cdot w \qquad\qquad 0 \cdot v = 0.$$

Let $X = \{x_1, \ldots, x_k\}$ be a finite set of variables, where x_h takes values in a finite set A_h $(1 \le h \le k)$. Let Y be a nonempty subset of X; a *Y-tuple* is an assignment of values to the variables in Y, and by A_Y we denote the set of Y-tuples. The set of X-tuples is denoted simply by A; moreover, given an X-tuple a and a nonempty subset Y of X, by a_Y we denote the Y-tuple obtained from a by ignoring the values of variables in $X \setminus Y$.

An *R-valued function* is a function with codomain R; if it takes on the value 1 everywhere, then it is called a *unitary function.*

Let f be an R-valued function of X; the *marginal* of f on a nonempty proper subset Y of X is the R-valued function of Y defined as follows:

$$f[Y] = \sum_{x \in X \setminus Y} f$$

that is, for Y-tuple b, the corresponding value of $f[Y]$ is

$$f[Y](b) = \sum_{a \in A : a_Y = b} f(a).$$

A *set system over* X is a set of nonempty subsets of X whose union recovers X. A set system $\mathcal{S}$ over X is *acyclic* [1] if either $\mathcal{S}$ is a singleton or there exists a *running-intersection ordering* of $\mathcal{S}$, that is, an ordering $(X_1, \ldots, X_n)$ of the sets in $\mathcal{S}$ that enjoys the following property:

(*running-intersection property*) For each i, $2 \le i \le n$, there exists $j_i < i$ such that $(X_1 \cup \cdots \cup X_{i-1}) \cap X_i \subseteq X_{j_i}$.

Several equivalent definitions of acyclicity exist [2]. We now recall two graph-theoretic characterizations [2] of acyclic set systems, which will be used in the sequel.

The *adjacency graph* (or "2-section" or "primal graph" or "moral graph") of a set system $\mathcal{S}$ over X is a graph G on X where two vertices x and y are adjacent if and only if there exists a set in $\mathcal{S}$ that contains both x and y. If every cycle $(x_1, \ldots, x_l, x_1)$, $l \geq 4$, in G contains two non-consecutive adjacent vertices, then G is a *chordal graph*. A set system $\mathcal{S}$ is acyclic if and only if its adjacency graph G is a chordal graph, and every clique (that is, every nonempty set of pairwise adjacent vertices) of G is a subset of some set in $\mathcal{S}$.

A set system $\mathcal{S} = \{X_1, \ldots, X_n\}$ over X is acyclic if and only if there exists an acyclic (undirected) graph T on $V = \{1, \ldots, n\}$ with vertex i labeled with X_i, that enjoys the following property:

> (*junction property*) For every element x of X, the subgraph of T induced by the set of vertices whose labels contain x is connected.

An acyclic graph such as T is called a *junction forest* (or "join forest") for $\mathcal{S}$ if T is not connected, and a *junction tree* (or "join tree") if T is connected. Assume that T is a junction tree for $\mathcal{S}$. A *rooted junction tree* is obtained from T by rooting T at any vertex; if r is the root, we denote the corresponding rooted junction tree by T_r. For every two adjacent vertices i and j of T_r, if the unique path in T_r from r to i passes through j, then we say that j is the *parent* of i or, equivalently, i is a child of j. For each vertex i of T_r, by $Ch(i)$ we denote the set of children of i. A vertex i is a *leaf* of T_r if $Ch(i) = \emptyset$. Henceforth, we denote an edge of T_r with end vertices i and j by ij if i is a child of j.

Remark 1. Given a running-intersection ordering $(X_1, \ldots, X_n)$ of $\mathcal{S}$, we can construct a rooted junction tree by taking vertex 1 to be the root and making vertex i child of vertex j_i for each $i > 1$. On the other hand, given a rooted junction tree T_r for $\mathcal{S}$, we can construct a running-intersection ordering of $\mathcal{S}$, by visiting vertices of T_r in a top-down way, that is, we visit a vertex after visiting its parent.

From a computational point of view, a set system $\mathcal{S}$ can be tested for acyclicity in polynomial time using the following algorithm [2].

Algorithm 1

Repeat the following two operations until neither can be longer applied:

1. Delete a variable if it occurs in exactly one member of $\mathcal{S}$.
2. Delete a member of $\mathcal{S}$ if it is contained in another member of $\mathcal{S}$.

It is well-known that the output of Algorithm 1 is defined uniquely and is independent of the sequence of deletions chosen (see Lemma 1 in [6]), and that $\mathcal{S}$ is acyclic if and only if the output of Algorithm 1 is $\{\emptyset\}$ (see Theorem 3.4 in [2]). Finally, a linear-time test for acyclicity can be found in [11], where a very efficient procedure for constructing a junction tree was also provided.

3. THE SUM-PRODUCT ALGORITHM

In this section we review the way in which SPA solves the following marginalization problem which is called the *all-vertices problem* in [1]:

> Let $\mathcal{S} = \{X_1, \ldots, X_n\}$ be an acyclic set system over X, and let f_i be an R-valued function of X_i $(1 \leq i \leq n)$. Compute the marginals of the product function $f_1 \cdot \ldots \cdot f_n$ on each X_i $(1 \leq i \leq n)$.

As in [1], we represent $\mathcal{S}$ by a junction forest, say T. Without loss of generality, henceforth we assume that T is connected. The SPA operates by "message-passing" in T: each edge of T is traversed twice, one for each of the two directions on the edge. When an edge with end vertices i and j is traversed from i to j, a "message" consisting of a function of $X_i \cap X_j$ is sent from i to j. An effective schedule makes use of a rooted junction tree T_r for $\mathcal{S}$, and consists of two phases: Phase I and Phase II. During Phase I, T_r is traversed in a bottom-up way, that is, each edge ij of T_r is traversed in the child-parent direction $i \to j$ and only after traversing all the edges hi for each child h of i; the message sent from i to j is denoted by $\mu_{i\to j}$. During Phase II, T_r is traversed in a top-down way, that is, each edge ij of T_r is traversed in the parent-child direction $j \to i$ and only after traversing the edge jk where k is the parent of j; the message sent from j to i is denoted by $\mu_{i\leftarrow j}$. The output of the procedure is the set of marginals $\{m_1, \ldots, m_n\}$ of the product function $f_1 \cdot \ldots \cdot f_n$ over $\mathcal{S}$.

SPA

Phase I
During a bottom-up traversal of T_r, when edge ij is traversed (from i to j) compute

$$\mu_{i\to j} := \Big(f_i \cdot \prod_{h\in Ch(i)} \mu_{h\to i}\Big)[X_i \cap X_j] \tag{1}$$

where $\prod_{h\in Ch(i)} \mu_{h\to i}$ is taken to be the unitary function if i is a leaf of T_r. After traversing all the edges ir, compute

$$m_r := f_r \cdot \prod_{i\in Ch(r)} \mu_{i\to r}. \tag{2}$$

Phase II
During a top-down traversal of T_r, when edge ij is traversed (from j to i) compute

$$\mu_{i\leftarrow j} := \Big(\mu_{j\leftarrow k} \cdot f_j \cdot \prod_{h\in Ch(j)\setminus\{i\}} \mu_{h\to j}\Big)[X_i \cap X_j] \tag{3}$$

$$m_i := \mu_{i\leftarrow j} \cdot f_i \cdot \prod_{h\in Ch(i)} \mu_{h\to i} \tag{4}$$

where k is the parent of j, $\mu_{j\leftarrow k}$ is taken to be the unitary function if $j = r$ and $\prod_{h\in Ch(j)\setminus\{i\}} \mu_{h\to j}$ is taken to be the unitary function if i is the only child of j.

It should be noted that only Phase I of the SPA is needed to compute m_r. On the other hand, the junction tree T can be rooted at any vertex, so that Phase I of the SPA is by itself a solution algorithm for the following marginalization problem which is called the *single-vertex problem* in [1]:

> Let $\mathcal{S} = \{X_1, \dots, X_n\}$ be an acyclic set system over X, and let f_i be an R-valued function of X_i ($1 \le i \le n$). For a given i, compute the marginal of the product function $f_1 \cdot \ldots \cdot f_n$ on X_i.

4. ALGEBRAIC INDEPENDENCE

In this section we make use of the SPA to prove that a product of R-valued functions enjoys a property (to be called *algebraic independence*) that is a natural generalization of probability-theoretic independence with a significant difference. It is well-known [8, 9] that, given a set M of marginals of some probability distribution of X over an acyclic set system $\mathcal{S}$ over X, the extension of M under which the sets in $\mathcal{S}$ are independent is uniquely determined ; moreover, this extension of M is characterized by the property of being factorable in terms of functions of sets in $\mathcal{S}$. We shall see that this is not the general case for algebraic independence; that is, given a set M of marginals of some R-valued function of X over an acyclic set system $\mathcal{S}$ over X, there may exist two or more extensions of M under each of which the sets in $\mathcal{S}$ are algebraically independent.

We begin by defining algebraic independence. Let X be a set of variables, let $\mathcal{S} = \{X_1, \dots, X_n\}$ be an acyclic set system over X, let $(X_1, \dots, X_n)$ be a running-intersection ordering of $\mathcal{S}$ and let $Y_i = (X_1 \cup \dots \cup X_{i-1}) \cap X_i$ for each i ($2 \le i \le n$). The sets in $\mathcal{S}$ are (*algebraically*) *independent* under an R-valued function p of X if the equality

$$p \cdot \prod_{i=2,\dots,n} p[Y_i] = \prod_{i=i,\dots,n} p[X_i]$$

holds everywhere; that is, for every X-tuple a one has

$$p(a) \cdot \prod_{i=2,\dots,n} p[Y_i](a_{Y_i}) = \prod_{i=i,\dots,n} p[X_i](a_{X_i}).$$

Given a junction tree $T = (V, E)$ for $\mathcal{S}$, by Remark 1 the independence among the sets in $\mathcal{S}$ can be re-stated as follows:

$$p \cdot \prod_{ij \in E} p[X_i \cap X_j] = \prod_{i \in V} p[X_i]. \tag{5}$$

We shall prove (see Theorem 1 below) that, if p is factorable by $\mathcal{S}$ (that is, $p = f_1 \cdot \ldots \cdot f_n$, where f_i is an R-valued function of X_i), then $X_1, \dots, X_n$ are independent under p. The proof is based on the following property of the SPA.

Lemma 1. Let $\mathcal{S} = \{X_1, \dots, X_n\}$ be an acyclic set system over X, let T_r be a rooted junction tree for $\mathcal{S}$ and, for every edge ij of T_r, let $\mu_{i \to j}$ and $\mu_{i \leftarrow j}$ be the messages computed by SPA with input $f_1, \dots, f_n$. For every edge ij of T_r, one has

$$(f_1 \cdot \ldots \cdot f_n)[X_i \cap X_j] = \mu_{i \to j} \cdot \mu_{i \leftarrow j}.$$

P r o o f. Consider any edge ij of T_r with $i \in Ch(j)$. By (4), one has

$$(f_1 \cdot \ldots \cdot f_n)[X_i \cap X_j] = m_i[X_i \cap X_j] = \sum_{x \in X_i \setminus X_j} \Big(\mu_{i \leftarrow j} \cdot f_i \cdot \prod_{h \in Ch(i)} \mu_{h \to i} \Big).$$

Since $\mu_{i \leftarrow j}$ is a function of $X_i \cap X_j$, we can move $\mu_{i \leftarrow j}$ to the left of the summation, so that one has

$$(f_1 \cdot \ldots \cdot f_n)[X_i \cap X_j] = \mu_{i \leftarrow j} \cdot \sum_{x \in X_i \setminus X_j} \Big(f_i \cdot \prod_{h \in Ch(i)} \mu_{h \to i} \Big)$$

which by (1) equals the product $\mu_{i \to j} \cdot \mu_{i \leftarrow j}$. □

Theorem 1. Let $\mathcal{S} = \{X_1, \ldots, X_n\}$ be an acyclic set system over X, let $p = f_1 \cdot \ldots \cdot f_n$, where f_i is an R-valued function of X_i. The sets $X_1, \ldots, X_n$ are independent under p.

P r o o f. Let $T_r = (V, E)$ be a rooted junction tree for $\mathcal{S}$. By Lemma 1, the left-hand side of (5) can be written as

$$\Big(\prod_{i \in V} f_i \Big) \cdot \Big(\prod_{ij \in E} \mu_{i \to j} \cdot \mu_{i \leftarrow j} \Big) = \Big(f_r \cdot \prod_{ir \in E} \mu_{i \to r} \Big) \cdot \Big(\prod_{ij \in E, i \neq r} f_i \cdot \mu_{i \leftarrow j} \cdot \prod_{hi \in E} \mu_{h \to i} \Big)$$

which by (2) and (4) is equal to $\prod_{i \in V} m_i$, that is, to the right-hand side of (5). □

We now address the problem of the uniqueness of the product-extension of a set of marginals over an acyclic set system. Let $\mathcal{S} = \{X_1, \ldots, X_n\}$ be a set system over X, let $M = \{m_1, \ldots, m_n\}$ be the set of marginals over $\mathcal{S}$ of some R-valued function. An R-valued function p of X is an *extension* of M if $p[X_i] = m_i$ for all i; if in addition for each i there exists an R-valued function f_i of X_i such that $p = f_1 \cdot \ldots \cdot f_n$, then p is a *product-extension* of M. Note that if $\mathcal{S}$ is acyclic then, by Theorem 1, the sets in $\mathcal{S}$ are independent under any product-extension of M. The following simple example shows that, given a set M of marginals over an acyclic set system $\mathcal{S}$, there may exist two or more product-extensions of M and, hence, there may exist two or more extensions of M under each of which the sets in $\mathcal{S}$ are independent.

Example 1. Let $(R, +, \cdot)$ be the commutative semiring where $R = \{0, 1\}$, $+$ is the addition mod 2, and $\cdot$ is the ordinary multiplication. Let x_1 and x_2 be two binary variables, and let m_i be the function of $\{x_i\}$ with $m_i(x_i) = 0$ everywhere, $i = 1, 2$. Let p be the unitary function of $\{x_1, x_2\}$ and let q be the function of $\{x_1, x_2\}$ with $q(x_1, x_2) = 0$ everywhere. It is trivial to check that both p and q are extensions of $\{m_1, m_2\}$. Moreover, p is the product of the two unitary functions of $\{x_1\}$ and $\{x_2\}$, and q is the product of m_1 and m_2. Therefore, since both p and q are product-extensions of $\{m_1, m_2\}$, the two sets $\{x_1\}$ and $\{x_2\}$ are independent under both p and q.

We shall prove that, given a set M of marginals over an acyclic set system $\mathcal{S}$, there exists exactly one product-extension of M if the commutative semiring $(R, +, \cdot)$ enjoys the following three properties:

(P1) For $u, v \in R$, if $u + v = 0$ then $u = v = 0$.

(P2) For $u, v \in R$, if $u \cdot v = 0$ then $u = 0$ or $v = 0$.

(P3) $(R, \cdot)$ is a group.

Note that (P2) states that there exist no zero divisors and, since one always has $0 \cdot v = 0$ (see Section 2), by (P2) one has that $u \cdot v = 0$ if and only if $u = 0$ or $v = 0$. Also note that, by (P3), if $v \neq 0$ then there exists w such that $v \cdot w = 1$ so that $x \cdot v = y \cdot v$ and $v \neq 0$ imply $x = y$. An example of a commutative semiring that enjoys (P1), (P2) and (P3) is the Boolean semiring which will be discussed in Section 5.

Theorem 2. Let $(R, +, \cdot)$ be a commutative semiring that enjoys (P1), (P2) and (P3). Let $\mathcal{S}$ be an acyclic set system over X, and let M be a set of marginals over $\mathcal{S}$. Then, there exists exactly one product-extension of M.

Proof. Let $\mathcal{S} = \{X_1, \ldots, X_n\}$ and let $M = \{m_1, \ldots, m_n\}$, and let p and q be two product-extensions of M. By Theorem 1, the sets in $\mathcal{S}$ are independent both under p and under q. Given a junction tree $T = (V, E)$ for $\mathcal{S}$, let $m_{ij} = m_i[X_i \cap X_j]$ for each edge ij of T. By (5) one has

$$p \cdot \prod_{ij \in E} m_{ij} = \prod_{i \in V} m_i \qquad q \cdot \prod_{ij \in E} m_{ij} = \prod_{i \in V} m_i.$$

Therefore, for every X-tuple a, one has

$$p(a) \cdot \prod_{ij \in E} m_{ij}(a_{X_i \cap X_j}) = q(a) \cdot \prod_{ij \in E} m_{ij}(a_{X_i \cap X_j}). \tag{6}$$

We now prove that $p(a) = q(a)$ for every X-tuple a. Let us distinguish two cases depending on whether or not $m_{ij}(a_{X_i \cap X_j}) = 0$ for some ij.

Case 1: $m_{ij}(a_{X_i \cap X_j}) = 0$ for some ij. Let $b = a_{X_i \cap X_j}$. Since

$$0 = m_{ij}(b) = \sum_{a' \in A, a'_{X_i \cap X_j} = b} p(a'),$$

by (P1) one has $p(a') = 0$ for every X-tuple a' with $a'_{X_i \cap X_j} = b$ and, hence, $p(a) = 0$. Using the same arguments, we can prove that $q(a) = 0$.

Case 2: $m_{ij}(a_{X_i \cap X_j}) \neq 0$ for all ij. Let

$$v = \prod_{ij \in E} m_{ij}(a_{X_i \cap X_j}).$$

Note that, by (P2), no factor of v is a zero divisor so that $v \neq 0$. At this point, we can re-write (6) as $p(a) \cdot v = q(a) \cdot v$ and, by (P3), one has $p(a) = q(a)$.

To sum up, the equality $p(a) = q(a)$ holds everywhere and, hence, $p = q$. □

5. THE BOOLEAN SEMIRING

In this section we show that, for the Boolean semiring, the SPA can be given a simpler formulation, which reduces the amount of computation.

In the Boolean semiring $(R, +, \cdot)$, R is $\{0, 1\}$, $+$ is the Boolean addition (OR), and $\cdot$ is the Boolean multiplication (AND). Note that the Boolean semiring enjoys not only properties (P1), (P2) and (P3) but also the following property

(P4) $u \cdot (u + v) = u$

which implies the following.

Lemma 2. Let f be a Boolean function of X. For every subset Y of X, one has

$$f \cdot f[Y] = f.$$

Proof. Let a be any X-tuple. Then, one has

$$f(a) \cdot f[Y](a_Y) = f(a) \cdot \sum_{a' : a'_Y = a_Y} f(a') = f(a) \cdot \Big(f(a) + \sum_{a' \neq a : a'_Y = a_Y} f(a') \Big)$$

which by (P4) is equal to $f(a)$. □

Turning back to the SPA with as inputs an acyclic set system $\mathcal{S} = \{X_1, \ldots, X_n\}$, a rooted junction tree T_r and functions $f_1, \ldots, f_n$, let

$$g_i = f_i \cdot \prod_{h \in Ch(i)} \mu_{h \to i} \tag{7}$$

for each vertex i of T_r. By (1), we can express $\mu_{i \to j}$ as

$$\mu_{i \to j} = g_i[X_i \cap X_j] \tag{8}$$

and, by (4), we can express m_i as

$$m_i = \begin{cases} g_i & \text{if } i = r \\ g_i \cdot \mu_{i \leftarrow j} & \text{else.} \end{cases} \tag{9}$$

We now prove that, for each edge ij, one has

$$m_i = g_i \cdot m_j[X_i \cap X_j]. \tag{10}$$

Consider the right-hand side of (10). By Lemma 1, it can be written as

$$g_i \cdot \mu_{i \to j} \cdot \mu_{i \leftarrow j}$$

which by (8) is equal to

$$g_i \cdot g_i[X_i \cap X_j] \cdot \mu_{i \leftarrow j}.$$

By Lemma 2, $g_i \cdot g_i[X_i \cap X_j] = g_i$ so that

$$g_i \cdot g_i[X_i \cap X_j] \cdot \mu_{i \leftarrow j} = g_i \cdot \mu_{i \leftarrow j}$$

which by (9) is equal to the left-hand side (m_i) of (10).

Using (8) and (10), we can re-formulate the SPA in the Boolean case as follows.

Algorithm 2

Initialization
Set $g_i := f_i$ for all i.

Phase I
During a bottom-up traversal of T_r, when edge ij is traversed (from i to j) update g_j as follows:

$$g_j := g_j \cdot g_i[X_i \cap X_j].$$

After traversing all the edges ir, set $m_r := g_r$.

Phase II
During a top-down traversal of T_r, when edge ij is traversed (from j to i) compute

$$m_i := g_i \cdot m_j[X_i \cap X_j].$$

We can give an even simpler formulation of Algorithm 2 using the notion of a "relation" and of two operators of relational algebra [2], which are now recalled.

A *relation* on a set X of variables is a set of X-tuples. Let s be a relation on X and let Y be a subset of X; the *projection* of s onto Y, denoted by $s[Y]$, is the relation on Y defined as follows:

$$s[Y] = \{a_Y : a \in s\}.$$

Let s and t be two relations on X and Y, respectively; the (*natural*) *join* of s and t, denoted by $s \bowtie t$, is the relation on $X \cup Y$ defined as follows:

$$s \bowtie t = \{a \in A_X \cup A_Y : a_X \in s \text{ and } a_Y \in t\}.$$

Note that the join is commutative.

Remark 2. Let s and t be two relations on X and Y, respectively. Then, one has

$$(s \bowtie t)[X] \subseteq s$$

and

$$(s \bowtie t)[Y] \subseteq t.$$

Consider now any Boolean function f of X. We call the *support* of f the set of X-tuples a for which $f(a) = 1$. Note that a Boolean function is fully specified by its support. It is easy to see that: (i) if f is a Boolean function of X with support s and Y is a subset of X, then the support of the marginal $f[Y]$ is equal to the projection $s[Y]$, and (ii) if f is a Boolean function of X with support s and g is a Boolean function of Y with support t, then the support of $f \cdot g$ is equal to the join $s \bowtie t$. Therefore, if we pre-compute the support s_i of each input function f_i of the SPA and update them using the following procedure, then the final values of $s_1, \dots, s_n$ will be exactly the supports of the output functions of the SPA.

Algorithm 3

Initialization
For $i = 1, \ldots, n$, compute the support s_i of f_i.

Phase I
During a bottom-up traversal of T_r, when edge ij is traversed (from i to j) update s_j as follows:

$$s_j := s_j \bowtie s_i[X_i \cap X_j].$$

Phase II
During a top-down traversal of T_r, when edge ij is traversed (from j to i) update s_i as follows:

$$s_i := s_i \bowtie s_j[X_i \cap X_j].$$

Note that Algorithm 3 coincides exactly with a classical algorithm (called the *full reducer*) developed in relational database theory [2, 3].

6. COMPUTATIONAL ASPECTS

In this section we show how to reduce the amount of computation performed by the SPA, which can be measured by the *arithmetic complexity* [1], that is, by the total number of (semiring) additions and multiplications required by the SPA. To this end, we first present a general procedure for transforming sum-product expressions of functions $\mu_{i \to j}$ and $\mu_{i \leftarrow j}$ into equivalent formulas that are easier from a computational point of view. Next, using the notion of the support of a function, we show how to reduce the number of tuples for which such sum-product expressions are to be evaluated.

6.1. Computing μ-functions

We shall give an effective plan to evaluate the sum-product expressions of $\mu_{i \to j}$ and $\mu_{i \leftarrow j}$, which requires fewer arithmetic operations. We begin with the sum-product expression (3) of $\mu_{i \leftarrow j}$ for each child i of j:

$$\mu_{i \leftarrow j} = \sum_{x \in X_j \setminus X_i} \Big(\mu_{j \leftarrow k} \cdot f_j \cdot \prod_{h \in Ch(j) \setminus \{i\}} \mu_{h \to j} \Big).$$

We can reduce the number of additions and multiplications needed to evaluate such a sum-product expression if, after computing $\mu_{j \leftarrow k}$, we also compute the marginal of the function $\mu_{j \leftarrow k} \cdot f_j$ on the set

$$X'_j = X_j \cap \big(\cup_{h \in Ch(j)} X_h \big).$$

Let χ_j denote this function of X'_j, that is,

$$\chi_j = \sum_{x \in X_j \setminus X'_j} (\mu_{j \leftarrow k} \cdot f_j).$$

The advantage is that, for each child i of j, one always has $X_i \cap X_j \subseteq X_i \cap X'_j$ so that each $\mu_{i\leftarrow j}$ can be evaluated more simply as

$$\mu_{i\leftarrow j} = \sum_{x \in X'_j \setminus X_i} \Big(\chi_j \cdot \prod_{h \in Ch(j)\setminus\{i\}} \mu_{h\rightarrow j}\Big).$$

Example 2. Consider the acyclic set system $\mathcal{S} = \{X_1 = \{x_1, x_2\}, X_2 = \{x_1, x_5\}, X_3 = \{x_1, x_6\}, X_4 = \{x_2, x_7\}, X_5 = \{x_1, x_2, x_3, x_4\}, X_6 = \{x_4\}\}$. A junction tree T for $\mathcal{S}$ has six vertices: 1, 2, 3, 4, 5 and 6, and vertex i is labeled by X_i $(1 \le i \le 6)$. Suppose we root T at the vertex 6; thus, $Ch(6) = \{5\}$ and $Ch(5) = \{1, 2, 3, 4\}$. Let f_i be an R-valued function of X_i $(1 \le i \le 6)$. After computing $\mu_{5\leftarrow 6}(x_4) = f_6(x_4)$, we also compute the function $\chi_5(x_1, x_2)$:

$$\chi_5(x_1, x_2) = \sum_{x_3, x_4} \Big(\mu_{5\leftarrow 6}(x_4) \cdot f_5(x_1, x_2, x_3, x_4)\Big).$$

At this point, the remaining $\mu_{i\leftarrow j}$ functions can be evaluated as follows:

$$\mu_{1\leftarrow 5}(x_1, x_2) = \chi_5(x_1, x_2) \cdot \mu_{2\rightarrow 5}(x_1) \cdot \mu_{3\rightarrow 5}(x_1) \cdot \mu_{4\rightarrow 5}(x_2)$$

$$\mu_{2\leftarrow 5}(x_1) = \textstyle\sum_{x_2} \big(\chi_5(x_1, x_2) \cdot \mu_{1\rightarrow 5}(x_1, x_2) \cdot \mu_{3\rightarrow 5} \cdot \mu_{4\rightarrow 5}(x_2)\big)$$

$$\mu_{3\leftarrow 5}(x_1) = \textstyle\sum_{x_2} \big(\chi_5(x_1, x_2) \cdot \mu_{1\rightarrow 5}(x_1, x_2) \cdot \mu_{2\rightarrow 5}(x_1) \cdot \mu_{4\rightarrow 5}(x_2)\big)$$

$$\mu_{4\leftarrow 5}(x_2) = \textstyle\sum_{x_1} \big(\chi_5(x_1, x_2) \cdot \mu_{1\rightarrow 5}(x_1, x_2) \cdot \mu_{2\rightarrow 5}(x_1) \cdot \mu_{3\rightarrow 5}(x_1)\big).$$

Observe that the above expression of $\mu_{i\leftarrow j}$ contains the product function

$$\pi_{j,i} = \prod_{h \in Ch(j)\setminus\{i\}} \mu_{h\rightarrow j}.$$

Consider now the sum-product expression of $\mu_{j\rightarrow k}$ (see (1)):

$$\mu_{j\rightarrow k} = \sum_{x \in X_j \setminus X_k} \Big(f_j \cdot \prod_{h \in Ch(j)} \mu_{h\rightarrow j}\Big)$$

which features the product function

$$\pi_j = \prod_{h \in Ch(j)} \mu_{h\rightarrow j}.$$

In [1], the authors proposed an efficient method which, given the functions $\mu_{h\rightarrow j}$, $h \in Ch(j)$, computes the product functions π_j and $\pi_{j,i}$ for each child i of j. We now present another method to reduce the number of arithmetic operations needed to evaluate the functions $\mu_{i\rightarrow j}$, $\mu_{i\leftarrow j}$ and χ_j. We begin with the sum-product expression (1) of $\mu_{i\rightarrow j}$:

$$\mu_{i\rightarrow j} = \sum_{x \in X_i \setminus X_j} \Big(f_i \cdot \prod_{h \in Ch(i)} \mu_{h\rightarrow i}\Big).$$

Let $(x_1, \ldots, x_k)$ be any ordering of the variables in $X_i \setminus X_j$. First, we break the multiple sum $\sum_{x \in X_i \setminus X_j}$ into k sums, each over one variable. Thus, we may write the sum-product expression of $\mu_{i \to j}$ as

$$\sum_{x_1} \cdots \sum_{x_k} \Big(f_i \cdot \prod_{h \in Ch(i)} \mu_{h \to i} \Big).$$

At this point, each factor $\mu_{h \to i}$ is moved left "as far as possible", that is, $\mu_{h \to i}$ stops when reaching a sum $\sum_{x_l}$ for which $x_l \in X_h \cap X_i$. Next, for each $l < k$, we merge the factors $\mu_{h \to i}$ that are after $\sum_{x_l}$ and before $\sum_{x_{l+1}}$ into one function of $X_i \setminus \{x_{l+1}, \ldots, x_k\}$, denoted by φ_l, which is preliminarily computed. Finally, the k summations are evaluated from right to left. It should be noted that the choice of a variable ordering which minimizes the cost of evaluating a sum is conjectured to be NP-complete [4]. We propose the following heuristic. With each variable x in $X_i \setminus X_j$ we associate the number $c(x)$ of the children h of i for which $x \in X_h$. We then order the variables x in $X_i \setminus X_j$ by non-increasing value of $c(x)$.

Example 2 (continued). The functions $\mu_{i \to 5}$ are evaluated easily as follows:

$$\mu_{1 \to 5}(x_1, x_2) = f_1(x_1, x_2)$$

$$\mu_{2 \to 5}(x_1) = \textstyle\sum_{x_5} f_2(x_1, x_5)$$

$$\mu_{3 \to 5}(x_1) = \textstyle\sum_{x_6} f_3(x_1, x_6)$$

$$\mu_{4 \to 5}(x_2) = \textstyle\sum_{x_7} f_4(x_2, x_7).$$

As to $\mu_{5 \to 6}(x_4)$, its sum-product expression is as follows:

$$\sum_{x_1, x_2, x_3} \Big(f_5(x_1, x_2, x_3, x_4) \cdot \mu_{1 \to 5}(x_1, x_2) \cdot \mu_{2 \to 5}(x_1) \cdot \mu_{3 \to 5}(x_1) \cdot \mu_{4 \to 5}(x_2) \Big).$$

If we order the variables in $X_5 \setminus X_6$ as (x_1, x_2, x_3), then we obtain the following sum-product expression for $\mu_{5 \to 6}(x_4)$:

$$\sum_{x_1} \Big(\mu_{2 \to 5}(x_1) \cdot \mu_{3 \to 5}(x_1) \cdot \sum_{x_2} \Big(\mu_{1 \to 5}(x_1, x_2) \cdot \mu_{4 \to 5}(x_2) \cdot \sum_{x_3} f_5(x_1, x_2, x_3, x_4) \Big) \Big).$$

At this point, we compute the following two functions

$$\varphi_1(x_1) = \mu_{2 \to 5}(x_1) \cdot \mu_{3 \to 5}(x_1)$$

$$\varphi_2(x_1, x_2) = \mu_{1 \to 5}(x_1, x_2) \cdot \mu_{4 \to 5}(x_2)$$

and evaluate the three summations

$$\sum_{x_1} \Big(\varphi_1(x_1) \cdot \sum_{x_2} \Big(\varphi_2(x_1, x_2) \cdot \sum_{x_3} f_5(x_1, x_2, x_3, x_4) \Big) \Big)$$

from right to left.

Needless to say, the same technique above can be used to evaluate the product functions $\mu_{i \leftarrow j}$ and χ_j.

Example 2 (continued). After computing $\mu_{5\leftarrow 6}(x_4) = f_6(x_4)$, the function $\chi_5(x_1, x_2)$ and the remaining $\mu_{i\leftarrow j}$ functions can be computed as follows:

$$\chi_5(x_1, x_2) = \sum_{x_4} \Big(\mu_{5\leftarrow 6}(x_4) \cdot \sum_{x_3} f_5(x_1, x_2, x_3, x_4)\Big)$$

$$\mu_{1\leftarrow 5}(x_1, x_2) = \chi_5(x_1, x_2) \cdot \mu_{2\rightarrow 5}(x_1) \cdot \mu_{3\rightarrow 5}(x_1) \cdot \mu_{4\rightarrow 5}(x_2)$$

$$\mu_{2\leftarrow 5}(x_1) = \mu_{3\rightarrow 5}(x_1) \cdot \sum_{x_2} \big(\chi_5(x_1, x_2) \cdot \mu_{1\rightarrow 5}(x_1, x_2) \cdot \mu_{4\rightarrow 5}(x_2)\big)$$

$$\mu_{3\leftarrow 5}(x_1) = \mu_{2\rightarrow 5}(x_1) \cdot \sum_{x_2} \big(\chi_5(x_1, x_2) \cdot \mu_{1\rightarrow 5}(x_1, x_2) \cdot \mu_{4\rightarrow 5}(x_2)\big)$$

$$\mu_{4\leftarrow 5}(x_2) = \sum_{x_1} \big(\chi_5(x_1, x_2) \cdot \mu_{1\rightarrow 5}(x_1, x_2) \cdot \mu_{2\rightarrow 5}(x_1) \cdot \mu_{3\rightarrow 5}(x_1)\big).$$

6.2. Reduction of the input size

We now show how to reduce the number of tuples at which the above sum-product expressions are to be evaluated. To this end, in analogy with a Boolean function, we define the support of an R-valued function f of X as the relation containing the X-tuples a for which $f(a) \neq 0$. First of all, observe that, given an R-valued function f of X and a subset Y of X, one has $f[Y](b) = 0$ if b is a Y-tuple such that $f(a) = 0$ for every X-tuple a with $a_Y = b$. Therefore, if s is the support of f, then

$$f[Y](b) = \begin{cases} \sum_{a\in s: a_y = b} f(a) & \text{if } b \in s[Y] \\ 0 & \text{else} \end{cases} \tag{11}$$

so that the support of $f[Y]$ is a subset of $s[Y]$. As a consequence, we can avoid evaluating the sum expression $f[Y]$ at the Y-tuples that do not belong to $s[Y]$. Moreover, given two R-valued functions f and g respectively of X and Y, one has $(f \cdot g)(a) = 0$ if a is an $(X \cup Y)$-tuple such that $f(a_X) = 0$ or $g(a_Y) = 0$. Therefore, if s and t are the supports of f and g, respectively, then one has

$$(f \cdot g)(a) = \begin{cases} f(a_X) \cdot g(a_Y) & \text{if } a \in s \bowtie t \\ 0 & \text{else} \end{cases} \tag{12}$$

so that the support of $f \cdot g$ is a subset of $s \bowtie t$. As a consequence, we can avoid evaluating the product $f \cdot g$ at the $(X \cup Y)$-tuples that do not belong to $s \bowtie t$.

We now show that further computation savings can be gained if the commutative semiring enjoys property (P2), that is, $u \cdot v = 0$ if and only if $u = 0$ or $v = 0$ (see Section 4).

Lemma 3. Let $(R, +, \cdot)$ be a commutative semiring that enjoys property (P2). Let f and g be two R-valued functions of X and Y, respectively. Let s and t be the supports of f and g, respectively. Then, the support of $f \cdot g$ is equal to $s \bowtie t$.

P r o o f. Since the support of $f \cdot g$ is always a subset of $s \bowtie t$, it is sufficient to prove that if $a \in s \bowtie t$ then $(f \cdot g)(a) \neq 0$. Assume that $a \in s \bowtie t$. By (12), $(f \cdot g)(a) = f(a_X) \cdot g(a_Y)$. Since $a \in s \bowtie t$, both $a_X \in s$ and $a_Y \in t$ so that $f(a_X) \neq 0$ and $g(a_Y) \neq 0$. By (P2), one has $(f \cdot g)(a) \neq 0$, which proves the statement. □

Corollary 1. Let $(R,+,\cdot)$ be a commutative semiring that enjoys property (P2). Let f and g be two R-valued functions of X and Y, respectively. Let s and t be the supports of f and g, respectively. If Z is a subset of $X \cup Y$, then the support of the marginal $(f \cdot g)[Z]$ is a subset of $(s \bowtie t)[Z]$.

Proof. Let b be any Z-tuple. If $(f \cdot g)[Z](b) \neq 0$ then, by (11), b belongs to the projection onto Z of the support of $f \cdot g$, which by Lemma 3 is equal to $s \bowtie t$. So, the support of $(f \cdot g)[Z]$ is a subset of $(s \bowtie t)[Z]$. □

Let us now consider the SPA with input $f_1, \ldots, f_n$ and output $m_1, \ldots, m_n$. Let s_i and t_i be the supports of f_i and m_i respectively, $i = 1, \ldots, n$. From the foregoing it follows that a good storage representation for f_i (or m_i) is given by a table which reports the values $f_i(b)$ ($m_i(b)$, respectively) only for the X_i-tuples b in s_i (in t_i, respectively). Accordingly, we can measure the size of the input of the SPA by the total size of the tables of $f_1, \ldots, f_n$, and the size of the output of the SPA by the total size of the tables of $m_1, \ldots, m_n$. By Corollary 1, one has

$$t_i \subseteq \big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i].$$

On the other hand, by Remark 2 one always has

$$\big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i] \subseteq s_i$$

so that $t_i \subseteq s_i$. In other words, the size of the output of the SPA is always less than or equal to the size of its input. The following example is illustrative.

Example 3. Let R be the set of nonnegative integers, and let $+$ and $\cdot$ be the ordinary addition and multiplication, respectively. Note that the commutative semiring $(R,+,\cdot)$ enjoys property (P2). Let $X = \{x_1, x_2, x_3, x_4\}$ and assume that the four variables share the same value set, say $\{a_1, a_2, \ldots, a_q\}$ with $q \geq 8$. Let f_1, f_2 and f_3 be R-valued functions of $X_1 = \{x_1, x_2\}$, $X_2 = \{x_2, x_3\}$ and $X_3 = \{x_3, x_4\}$, respectively. Assume that the support of f_i is $s_i = \{(a_1,a_2), (a_2,a_4), (a_3,a_6), (a_4,a_8)\}$ and that

$$f_1(a_1,a_2) = f_2(a_1,a_2) = f_3(a_1,a_2) = 1$$

$$f_1(a_2,a_4) = f_2(a_2,a_4) = f_3(a_2,a_4) = 2$$

$$f_1(a_3,a_6) = f_2(a_3,a_6) = f_3(a_3,a_6) = 3$$

$$f_1(a_4,a_8) = f_2(a_4,a_8) = f_3(a_4,a_8) = 4.$$

It is easily seen that $s_1 \bowtie s_2 \bowtie s_3 = \{(a_1,a_2,a_4,a_8)\}$ and $\big(f_1 \cdot f_2 \cdot f_3\big)(a_1,a_2,a_4,a_8) = 8$. Therefore, the supports of the output (m_1, m_2 and m_3) of the SPA with input f_1, f_2 and f_3 are

$$t_1 = \{(a_1,a_2)\} \qquad t_2 = \{(a_2,a_4)\} \qquad t_3 = \{(a_4,a_8)\}$$

and $m_1(a_1,a_2) = m_2(a_2,a_4) = m_3(a_4,a_8) = 8$. Apparently, the size of the output of the SPA is less than the size of the input.

We now show how to reduce the input size of the SPA. To this end, we introduce the notion of the *reduction* of the input function f_i (with respect to $f_1, \ldots, f_n$), by which we mean the function f_i' of X_i defined as follows:

$$f'(b) = \begin{cases} f(b) & \text{if } b \in \big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i] \\ 0 & \text{else} \end{cases} \tag{13}$$

where s_i is the support of f_i.

Example 3 (continued). The reduction f_1' of f_1 takes on the value 1 at (a_1, a_2) and 0 elsewhere. The reduction f_2' of f_2 takes on the value 2 at (a_2, a_4) and 0 elsewhere. The reduction f_3' of f_3 takes on the value 4 at (a_4, a_8) and 0 elsewhere.

Theorem 3. Let $(R, +, \cdot)$ be a commutative semiring that enjoys property (P2), and let f_i' be the reduction of f_i, $i = 1, \ldots, n$. Then $f_1 \cdot \ldots \cdot f_n = f_1' \cdot \ldots \cdot f_n'$.

Proof. We need to prove that the equality

$$f_1(a_{X_1}) \cdot \ldots \cdot f_n(a_{X_n}) = f_1'(a_{X_1}) \cdot \ldots \cdot f_n'(a_{X_n}) \tag{14}$$

holds everywhere. Let a be any X-tuple, and let v be the left-hand side of (14). Two cases are distinguished depending on whether or not $v = 0$.

Case 1: $v = 0$. By Lemma 3, the relation $s_1 \bowtie \ldots \bowtie s_n$ does not contain a and, hence, there exists i for which $a_{X_i} \notin s_i$. By (12), one also has $a_{X_i} \notin \big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i]$. From (13) it follows that $f_i'(a_{X_i}) = 0$ and, hence, also the right-hand side of (14) is equal to 0.

Case 2: $v \neq 0$. By Lemma 3, the relation $s_1 \bowtie \ldots \bowtie s_n$ contains a and, hence, $a_{X_i} \in s_i$ for all i. By (13) one has that $f_i'(a_{X_i}) = f_i(a_{X_i})$ and, hence, the right-hand side of (14) is equal to v. □

Suppose we now apply the SPA with input $f_1', \ldots, f_n'$ instead of $f_1, \ldots, f_n$. By Theorem 3, the output is the same. Let s_i' be the support of f_i'. By (13), one has

$$s_i' \subseteq \big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i]$$

so that by (12), one has $s_i' \subseteq s_i$. In other words, the size of the input of the SPA with input $f_1', \ldots, f_n'$ is less than or equal to the size of the input of the SPA with input $f_1, \ldots, f_n$. It should be noted that, in order to apply the SPA with input $f_1', \ldots, f_n'$, we need to compute for each i: the support s_i of f_i, next the relation $\big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i]$ and finally the reduction f_i' of f_i. Suppose we have already computed $s_1, \ldots, s_n$. Then, the relations $\big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i]$ $(1 \leq i \leq n)$ can be obtained by solving the Boolean all-vertices problem of Section 5, that is, by applying Algorithm 3. Finally, we can obtain the functions $f_1', \ldots, f_n'$ via (13).

Let us now assume that each input function f_i of the SPA is in "reduced form" by which we mean that f_i equals its reduction f_i'. We now prove that the input and the output of the SPA have the same size if the commutative semiring enjoys not only property (P2) but also property (P1), that is, if $u+v = 0$ then $u = v = 0$ (see Section 4).

Theorem 4. Let $(R, +, \cdot)$ be a commutative semiring that enjoys properties (P1) and (P2), and let $f_1, \ldots, f_n$ be R-valued functions in reduced form. Then, the output of the SPA with input $f_1, \ldots, f_n$ have the same size as the input.

Proof. Let s_i be the support of f_i and let t_i be the support of the output function m_i $(1 \le i \le n)$. We need to prove that $t_i = s_i$ for all i. Since $t_i \subseteq s_i$, it is sufficient to prove that, if an X_i-tuple b does not belong to t_i, then b does not belong to s_i. Assume that b does not belong to t_i; thus, $m_i(b) = 0$. By (P1), for every X-tuple a with $a_{X_i} = b$, one has $\big(f_1 \cdot \ldots \cdot f_n\big)(a) = 0$. By Lemma 3, no X-tuple a with $a_{X_i} = b$ belongs to the relation $s_1 \bowtie \ldots \bowtie s_n$. It follows that b does not belong to the relation $\big(s_1 \bowtie \ldots \bowtie s_n\big)[X_i]$. By (13), one has $f_i'(b) = 0$. Finally, since f_i is in reduced form (that is, $f_i' = f_i$), we obtain $f_i(b) = 0$, which proves that b does not belong to s_i. □

7. EVALUATING A SUM-PRODUCT EXPRESSION

In this section we show that the SPA can be used to solve a marginalization problem (to be called the *sum-product expression problem*) which generalizes the single-vertex problem of Section 3 to the case in which the set system $\mathcal{S} = \{X_1, \ldots, X_n\}$ is arbitrary and the objective function is the marginal, say m, on an arbitrary subset Y of X. In the following two subsections we distinguish two cases depending on whether or not $\mathcal{S}$ is acyclic.

7.1. The acyclic case

If $\mathcal{S}$ is acyclic, then the following variant of Phase I of the SPA solves the sum-product expression problem.

Algorithm 4

During a bottom-up traversal of T_r, when edge ij is traversed (from i to j) update the label X_j of j as follows

$$X_j := X_j \cup (X_i \cap Y)$$

and compute

$$\mu_{i\to j} := \Big(f_i \cdot \prod_{h \in Ch(i)} \mu_{h\to i}\Big)[X_i \cap X_j].$$

After traversing all the edges ir for each child i of the root r, compute the objective function m as follows

$$m := \Big(f_r \cdot \prod_{i \in Ch(r)} \mu_{i\to r}\Big)[Y]. \tag{15}$$

The key-point of Algorithm 4 is that the label of each non-leaf vertex j is updated in such a way that, after traversing all the edges ij, the label X_j of j also contains the elements of Y that occur in the labels of vertices of the subtree T_j of T_r rooted at j. Then, it is easy to show by induction that

$$f_j \cdot \prod_{i \in Ch(j)} \mu_{i\to j} = \Big(\prod_{i \in V_j} f_i\Big)[X_j]$$

where V_j is the vertex set of T_j. So, after traversing all the edges ir, one has $Y \subseteq X_r$ and

$$f_r \cdot \prod_{i \in Ch(r)} \mu_{i \to r} = \Big(\prod_{i \in V_r} f_i \Big)[X_r]$$

from which (15) easily follows.

Example 4. Consider again the acyclic set system $\mathcal{S}$ and the rooted junction tree T_6 of Example 2. Suppose we want the marginal m on $\{x_4, x_7\}$. The functions $\mu_{1\to5}$, $\mu_{2\to5}$, $\mu_{3\to5}$ are the same as in Example 2. For $\mu_{4\to5}$ and $\mu_{5\to6}$, we have

$$\mu_{4\to5}(x_2, x_7) = f_4(x_2, x_7)$$

$$\mu_{5\to6}(x_4, x_7) = \sum_{x_1, x_2, x_3} f_5(x_1, x_2, x_3, x_4) \cdot \mu_{1\to5}(x_1, x_2) \cdot \mu_{2\to5}(x_1) \cdot \mu_{3\to5}(x_1) \cdot \mu_{4\to5}(x_2, x_7).$$

After computing $\mu_{5\to6}(x_4, x_7)$, we obtain the marginal m as follows:

$$m(x_4, x_7) := f_6(x_4) \cdot \mu_{5\to6}(x_4, x_7).$$

7.2. The cyclic case

If $\mathcal{S}$ is cyclic, then we can reduce the sum-product expression problem to the acyclic case by adding edges to the adjacency graph G of $\mathcal{S}$ to obtain a chordal graph using a triangulation algorithm (e. g., see [11]) or, better, a minimal triangulation algorithm (e. g., see [10]). Let $\widehat{G}$ be the chordal graph resulting from a triangulation of G, and let $\widehat{\mathcal{S}}$ be the set of maximal cliques of $\widehat{G}$; thus, $\widehat{\mathcal{S}}$ is an acyclic set system over X. Let $\widehat{T}$ be a junction tree for $\widehat{\mathcal{S}}$. Of course, the set system $\mathcal{S}' = \widehat{\mathcal{S}} \cup \mathcal{S}$ is still acyclic since every set in $\mathcal{S}$ is contained in some set in $\widehat{\mathcal{S}}$, and a junction tree T' for $\mathcal{S}'$ can be obtained from $\widehat{T}$ by adding, for each set X_i in $\mathcal{S} \setminus \widehat{\mathcal{S}}$, a vertex which is labelled by X_i and is made adjacent to a vertex of $\widehat{T}$ labelled by a set in $\widehat{\mathcal{S}}$ that contains X_i. At this point, with each vertex of T' labelled by a set $X' \notin \mathcal{S}$ we associate a unitary function of X'. Finally, given a subset Y of X, we can compute the objective function as in the acyclic case. Sometimes, we can reduce the computational effort to construct the acyclic set system $\mathcal{S}'$ by triangulating a suitable subgraph of the adjacency graph G of $\mathcal{S}$, based on a well-known result (see Corollary 3.2 in [5]) which states that, if $\mathcal{S}$ is cyclic, then the union of sets in the output of Algorithm 1 is the set of least cardinality whose addition to $\mathcal{S}$ makes it acyclic. Therefore, we only need to triangulate the subgraph of G being the adjacency graph of the output of Algorithm 1. If $\widetilde{\mathcal{S}}$ is the set of maximal cliques of the resulting chordal graph, then the set system $\mathcal{S}' = \widetilde{\mathcal{S}} \cup \mathcal{S}$ is acyclic as the application of Algorithm 1 to $\mathcal{S}'$ yields $\{\emptyset\}$. It should be noted that the acyclic set system $\mathcal{S}'$ constructed by either method is independent of Y and, hence, of the specific objective function of the sum-expression problem at hand. A different approach was followed in [1], whose authors construct an acyclic set system which is "tailored" to Y and is obtained by triangulating the adjacency graph of the set system $\mathcal{S} \cup \{Y\}$, regardless of whether or not $\mathcal{S}$ is cyclic.

Example 5. Consider the cyclic set system $\mathcal{S} = \{X_1 = \{x_1, x_2\}, X_2 = \{x_1, x_3\}, X_3 = \{x_2, x_4, x_5\}, X_4 = \{x_3, x_4\}\}$. The adjacency graph G of $\mathcal{S}$ is a "house". The application of Algorithm 1 to $\mathcal{S}$ yields the cyclic set system obtained from $\mathcal{S}$ by deleting x_5, whose adjacency graph is the 4-cycle of G. Let $\widetilde{\mathcal{S}}$ be the set of maximal cliques of the chordal graph obtained from the 4-cycle of G by adding an edge joining the vertices 2 and 3; that is, $\widetilde{\mathcal{S}} = \{X_5 = \{x_1, x_2, x_3\}, X_6 = \{x_2, x_3, x_4\}\}$. The set system $\mathcal{S}' = \widetilde{\mathcal{S}} \cup \mathcal{S}$ is acyclic and the junction tree T' for $\mathcal{S}'$ has six vertices which are labelled by the sets $X_1, \ldots, X_6$; moreover, the vertex of T' labelled by X_5 is adjacent to the vertices labelled by X_1, X_2 and X_6, and the vertex of T' labelled by X_6 is also adjacent to the vertices labelled by X_3 and X_4.

8. CONCLUSIONS

We analyzed the SPA to prove algebraic independence among factors, and an interesting question that naturally arises is whether or not algebraic independence has the same axiomatization as its probability-theory counterpart. We also showed that the SPA in the Boolean semiring reduces to a classical algorithm developed in database theory. From a computational point of view, we presented some methods to reduce the arithmetic complexity of the SPA as well as its input size, and it is easy to see that they are also applicable to the case in which the variables have infinite value sets provided that the supports of functions are finite relations. Finally, we showed how to modify the SPA to compute an arbitrary marginal of a product function, both in acyclic and cyclic cases.

9. ACKNOWLEDGEMENT

The author sincerely thanks the referees for their careful reading of the paper and their constructive suggestions which led to a substantial improvement of the manuscript.

(Received November 7, 2011)

REFERENCES

[1] S. M. Aji and R. J. McEliece: The generalized distributive law. IEEE Trans. Inform. Theory *46* (2000), 325–343.

[2] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis: On the desirability of acyclic database schemes. J. Assoc. Comput. Mach. *30* (1983), 479–513.

[3] P. A. Bernstein and N. Goodman: The power of natural semijoins. SIAM J. Comput. *10* (1981), 751–771.

[4] S. A. Goldman and R. L. Rivest: Making maximum-entropy computations easier by adding extra constraints. In: Maximum-Entropy and Bayesian Methods in Science and Engineering 2 (G. J. Erikson and C. R. Smith, eds.), Kluwer Academic Pub. 1988, pp. 323–340.

[5] N. Goodman, O. Shmueli, and T. Tay: GYO reductions, canonical connections, tree and cyclic schema, and tree projections. J. Comput. and System Sci. *29* (1984), 338–358.

[6] F. R. Kschinschang, B. J. Frey, and H.-A. Loeliger: Factor graphs and the sum-product algorithm. IEEE Trans. Inform. Theory *47* (2001), 498–519.

[7] D. Maier and J. D. Ullman: Connections in acyclic hypergraphs. Theoret. Comput. Sci. *32* (1984), 185–199.

[8] F. M. Malvestuto: Existence of extensions and product extensions for discrete probability distributions. Discrete Math. *69* (1988), 61–77.

[9] F. M. Malvestuto: From conditional independences to factorization constraints with discrete random variables. Ann. Math. Artif. Intel. *35* (2002), 253–285.

[10] M. Mezzini: Fast minimal triangulation algorithm using minimum degree criterion. Theoret. Comput. Sci. *412* (2011), 3775–3787.

[11] R. E. Tarjan and M. Yannakakis: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput. *13* (1984), 566–579.

Francesco M. Malvestuto, Department of Informatics, Faculty of Information Engineering, Informatics and Statistics, Sapienza University of Rome, Via Salaria 113, 00198 Rome. Italy.
e-mail: malvestuto@di.uniroma1.it