Decomposition of multi-operator queries on semiring-based graphical models

Cédric Pralet[1][2], Thomas Schiex[2], and Gérard Verfaillie[3]

[1] LAAS-CNRS, Toulouse, France cpralet@laas.fr
[2] INRA, Castanet Tolosan, France tschiex@toulouse.inra.fr
[3] ONERA, Centre de Toulouse, France gerard.verfaillie@onera.fr

Abstract. In the last decades, the Satisfiability and Constraint Satisfaction Problem frameworks were extended to integrate aspects such as uncertainties, partial observabilities, or uncontrollabilities. The resulting formalisms, including Quantified Boolean Formulas (QBF), Quantified CSP (QCSP), Stochastic SAT (SSAT), or Stochastic CSP (SCSP), still rely on networks of local functions defining specific *graphical models*, but they involve queries defined by sequences of distinct *elimination operators* ($\exists$ and $\forall$ for QBF and QCSP, max and $+$ for SSAT and SCSP) preventing variables from being considered in an arbitrary order when the problem is solved (be it by tree search or by variable elimination).
In this paper, we show that it is possible to take advantage of the actual structure of such *multi-operator queries* to bring to light new ordering freedoms. This leads to an improved *constrained induced-width* and doing so to possible exponential gains in complexity. This analysis is performed in a generic semiring-based algebraic framework that makes it applicable to various formalisms. It is related with the *quantifier tree* approach recently proposed for QBF but it is much more general and gives theoretical bases to observed experimental gains.

1 Introduction

Searching for a solution to a Constraint Satisfaction Problem (CSP [1]) is equivalent to searching for an assignment of the problem variables maximizing the quantity given by the constraints conjunction, i.e. to *eliminating* variables using max.[4] As max is the only elimination operator involved in such a *mono-operator* query, variables can be considered in any order. The situation is similar with the Satisfiability problem (SAT) but not with Quantified CSP (QCSP [2]) or Quantified Boolean Formulas (QBF), where min (equivalent to $\forall$) and max (equivalent to $\exists$) operators can alternate, or with Stochastic SAT (SSAT [3]) or Stochastic CSP (SCSP [4]), involving max and $+$ operators: these frameworks define *multi-operator* queries for which the order in which variables can be considered is not free.

[4] Eliminating variables in a set S' with an operator $\oplus$ from a function φ defined on the set $dom(S)$ of assignments of a set of variables S means computing the function $\oplus_{S'}\varphi$ defined by $(\oplus_{S'}\varphi)(A) = \oplus_{A' \in dom(S')}\varphi(A.A')$ for all assignments A of $S - S'$. $\oplus_{S'}\varphi$ synthesizes the information given by φ if we disregard variables in S'.

To overcome this difficulty, variables are usually considered in an order compatible with the sequence of eliminations (if this sequence is "$\forall x_1, x_2 \, \exists x_3$" for a QCSP, then x_1 and x_2 are considered after x_3 in a variable elimination algorithm). This suffices to obtain the correct result but does not take advantage of all the actual structural features of multi-operator queries. For example, as shown by the *quantifier trees* approach [5] recently introduced for QBF, analyzing hidden structures of "flat" prenex normal form QBF can lead to important gains in terms of solving time.

After the introduction of some notations, we define a generic systematic approach for analyzing the actual *macrostructure* of multi-operator queries by transforming them into a tree of mono-operator ones (Section 3). Being defined in a generic algebraic framework, this approach extends and generalizes the all quantifier tree proposal [5]. It is applicable to multiple formalisms, including QCSP, SSAT, or SCSP. Its efficiency, experienced on QBF with quantifier trees, is interpreted theoretically in terms of a parameter called the *constrained induced-width*. Last, we define on the built macrostructure a generic variable elimination (VE) algorithm exploiting *cluster tree decompositions* [6] (Section 4).

2 Background notations and definitions

The domain of values of a variable x is denoted $dom(x)$. By extension, the domain of a set of variables S is $dom(S) = \prod_{x \in S} dom(x)$. A *scoped function* φ on S is a function $dom(S) \rightarrow E$. S is called the *scope* of φ and is denoted $sc(\varphi)$.

In order to reason about scoped functions, we need to combine and synthesize the information they express: e.g., to answer a QCSP $\forall x_1, x_2 \, \exists x_3 (\varphi_{x_1,x_3} \wedge \varphi_{x_2,x_3})$, we need to aggregate local constraints using $\wedge$ and to synthesize the result using $\exists$ on x_3 and $\forall$ on x_1, x_2. The operator used to aggregate scoped functions is called a *combination operator* and is denoted $\otimes$. The multiple operators used to synthesize information are called *elimination operators* and are denoted $\oplus$. More precisely, the algebraic structure we consider, defining elimination and combination operators, is a *Multi Commutative Semiring* (MCS).

Definition 1. *$(E, \oplus, \otimes)$ is a* commutative semiring *iff E is a set such that $\oplus$ and $\otimes$ are binary associative, commutative operators on E, $\oplus$ has an identity $0_\oplus \in E$ ($x \oplus 0_\oplus = x$), $\otimes$ has an identity $1_\otimes \in E$ ($x \otimes 1_\otimes = x$), and $\otimes$ distributes over $\oplus$ ($x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$).*[5]

$(E, \{\oplus^i, i \in I\}, \otimes)$ is a Multi Commutative Semiring *(MCS) iff for all $i \in I$, $(E, \oplus^i, \otimes)$ is a commutative semiring.*

Table 1 shows MCS examples and frameworks in which they are used. There exist many other examples, such as $(E, \{\cap, \cup\}, \cap)$.

Definition 2. *A* graphical model *on a MCS $(E, \{\oplus^i, i \in I\}, \otimes)$ is a pair (V, Φ) where V is a finite set of finite domain variables and Φ is a finite multiset of scoped functions taking values in E and whose scopes are included in V.*

[5] Compared to other definitions of commutative semirings, $0_\oplus$ is not assumed to be an annihilator for $\otimes$, so that e.g. $(\mathbb{N} \cup \{\infty\}, \max, +)$ is seen as a commutative semiring.

E	$\{\oplus^i, i \in I\}$	$\otimes$	Frameworks
$\mathbb{R}^+ \cup \{\infty\}$	$\{\max, +\}$	$\times$	SSAT [3], SCSP [4], Bayesian networks [7]
$\mathbb{R}^+ \cup \{\infty\}$	$\{\min, \max, +\}$	$\times$	Extended-SSAT [3]
$\mathbb{N} \cup \{\infty\}$	$\{\min, \max\}$	$+$	MDPs based on kappa-rankings [8]
$[0, 1]$	$\{\min, \max\}$	min	possibilistic optimistic MDPs [9]
$\{t, f\}$	$\{\wedge, \vee\}$ (i.e. $\{\forall, \exists\}$)	$\wedge$	QBF, QCSP [2]

Table 1. Examples of MCS (t stands for *true* and f for *false*).

A CSP is a graphical model (V, Φ) where Φ contains constraints on V. We introduce *operator-variables sequences* and *queries* to reason about graphical models.

Definition 3. *Let $\preceq$ be a partial order on V. The set of* linearizations *of $\preceq$, denoted $lin(\preceq)$, is the set of total orders $\preceq'$ on V satisfying $(x \preceq y) \rightarrow (x \preceq' y)$.*

Definition 4. *Let $(E, \{\oplus^i, i \in I\}, \otimes)$ be a MCS. A* sequence of operator-variables *on a set of variables V is defined by $SOV = op_{1_{S_1}} \cdot op_{2_{S_2}} \cdot \ldots \cdot op_{p_{S_p}}$, where $\{S_1, S_2, \ldots, S_p\}$ is a partition of V and $op_j \in \{\oplus^i, i \in I\}$ for all $j \in \{1, \ldots, p\}$. The* partial order $\preceq_{SOV}$ *induced by SOV is given by $S_1 \prec_{SOV} S_2 \prec_{SOV} \ldots \prec_{SOV} S_p$ (it forces variables in S_j to be eliminated before variables in S_i whenever $i < j$). An elimination order $o : x_{o_1} \prec x_{o_2} \prec \ldots \prec x_{o_q}$ on V is a total order on V. It is* compatible with *SOV iff $o \in lin(\preceq_{SOV})$. If $op(x)$ corresponds to the elimination operator of x in SOV, $SOV(o)$ denotes the sequence of operator-variables $op(x_{o_1})_{x_{o_1}} \cdot op(x_{o_2})_{x_{o_2}} \cdot \ldots \cdot op(x_{o_q})_{x_{o_q}}$.*

For the MCS $(\mathbb{R}^+ \cup \{\infty\}, \{\min, \max, +\}, \times)$, a sequence of operator-variables on $V = \{x_1, x_2, x_3, x_4, x_5\}$ is e.g. $SOV = \min_{x_1, x_2} \sum_{x_3, x_4} \max_{x_5}$. The partial order it induces satisfies $\{x_1, x_2\} \prec_{SOV} \{x_3, x_4\} \prec_{SOV} x_5$. The elimination order $o : x_1 \prec x_2 \prec x_4 \prec x_3 \prec x_5$ is compatible with SOV (and $SOV(o) = \min_{x_1} \min_{x_2} \sum_{x_4} \sum_{x_3} \min_{x_5}$), whereas $o' : x_4 \prec x_2 \prec x_1 \prec x_3 \prec x_5$ is not.

Definition 5. *Given a MCS $(E, \{\oplus^i, i \in I\}, \otimes)$, a* query *is a pair $Q = (SOV, \mathcal{N})$ where $\mathcal{N} = (V, \Phi)$ is a graphical model and SOV is a sequence of operator-variables on V. The* answer to a query *is $Ans(Q) = SOV\,(\otimes_{\varphi \in \Phi}\, \varphi)$.*

All the elimination operators considered here being commutative and associative, every elimination order compatible with $\preceq_{SOV}$ can be used to answer a query, i.e. for every $o \in lin(\preceq_{SOV})$, $Ans(Q) = SOV(o)\,(\otimes_{\varphi \in \Phi}\, \varphi)$.

The definition of the answer to a query covers various decision problems raised in many formalisms. Among the multi-operator ones, one can cite:

1. *Quantified Boolean Formulas* in conjunctive prenex normal form and *Quantified CSPs* [2], looking like $\forall x_1, x_2 \exists x_3 \forall x_4\, (\varphi_{x_1, x_3, x_4} \wedge \varphi_{x_2, x_4})$;
2. *Stochastic Satisfaction* problems (SSAT and Extended-SSAT [3]) and some queries on *Stochastic CSPs* [4] looking like $\max_{d_1, d_2} \sum_{s_1} \max_{d_3} (\prod_{\varphi \in \Phi} \varphi)$, where Φ contains both constraints and conditional probability distributions;
3. some types of finite horizon *Markov Decision Processes* (MDPs [10]), on which queries look like $\max_{d_1} \oplus_{s_1} \ldots \max_{d_n} \oplus_{s_n} (\otimes_{\varphi \in \Phi}\, \varphi)$, where $(\oplus, \otimes)$ equals $(+, \times)$ (MDPs optimizing an expected satisfaction), $(\max, \min)$ (optimistic possibilistic MDPs [9]), or $(\max, +)$ (MDPs based on kappa-rankings [8]).

It also covers queries in other frameworks like Bayesian Networks (BN [7]), or in yet unpublished frameworks such as quantified VCSPs (i.e. VCSPs [11] using an alternation of min and max operations on a combination of soft constraints) or semiring CSPs [11] with multiple elimination operators.

As only one combination operator is involved in the definition of the answer to a query, formalisms such as influence diagrams [12], classical probabilistic MDPs [10], or pessimistic possibilistic MDPs [9] are not basically covered but can be if transformed using so-called "potentials" [13]. However, in these cases, more direct efficient approaches can be proposed. See [14] for further details.

3 Macrostructuring a multi-operator query

Analyzing the *macrostructure* of queries means bringing to light the actual constraints on the elimination order and the possible decompositions. We first give a parameter, the *constrained induced-width*, for quantifying the complexity of a VE algorithm on multi-operator queries and then show how this complexity can be decreased. This leads us to define a systematic method for structuring an unstructured multi-operator query into a tree of mono-operator ones.

3.1 Constrained induced-width

A parameter defining an upper bound on the theoretical complexity of standard VE algorithms on mono-operator queries is the induced-width [15]. In the multi-operator case however, there are constraints on the elimination order because the alternating elimination operators do not generally commute. The complexity can then be quantified using the *constrained induced-width* [16, 17] as defined below.

Definition 6. *Let $G = (V_G, H_G)$ be a hypergraph*[6] *and let $\preceq$ be a partial order on V_G. The constrained induced-width $w_G(\preceq)$ of G with constraints on the elimination order given by $\preceq$ ("$x \prec y$" stands for "y must be eliminated before x") is defined by $w_G(\preceq) = \min_{o \in lin(\preceq)} w_G(o)$, $w_G(o)$ being the induced-width of G for the elimination order o (i.e. the size of the largest hyperedge created when eliminating variables in the order given by o).*[7]

The basic hypergraph associated with a graphical model $\mathcal{N} = (V, \Phi)$ is $G = (V, \{sc(\varphi) \mid \varphi \in \Phi\})$ and the constraints on the elimination order imposed by a query $Q = (SOV, \mathcal{N})$ can be described by $\preceq_{SOV}$ (cf Definition 4). An upper bound on the theoretical complexity of a VE algorithm for answering a query is then $O(|\Phi| \cdot d^{1+w_G(\preceq_{SOV})})$, d being the maximum domain size (for all the complexity results of the paper, we assume that operations like $a \otimes b$ or $a \oplus b$ take a bounded time). Since a linear variation of the constrained induced width yields an exponential variation of the complexity, it is worth working on the two parameters it depends on: the partial order $\preceq_{SOV}$ and the hypergraph G.

[6] V_G is a set of variables and H_G is a set of hyperedges on V_G, i.e. a subset of 2^{V_G}.

[7] To be more formal, we should speak of the induced-width of the primal graph of G (the graph containing an edge $\{x, y\}$ iff there exists $h \in H_G$ s.t. $\{x, y\} \subset h$) since the usual definition of the induced-width holds on graphs (and not on hypergraphs).

Weakening constraints on the elimination order is known to be useless in contexts like Maximum A Posteriori hypothesis [17], where there is only one alternation of max and sum marginalizations. But it can decrease the constrained induced-width as soon as there are more than two levels of alternation.

Indeed, assume that a Stochastic CSP query is equivalent to computing $\max_{x_1,\dots,x_q} \sum_y \max_{x_{q+1}} \left(\varphi_y \times \varphi_{y,x_1} \times \prod_{i\in\{1,\dots,q\}} \varphi_{x_i,x_{q+1}}\right)$ (this may occur if φ_y is a probability distribution on y, the other φ_S model constraints, and the value of y is observed only before making decision x_{q+1}). If one uses $G = (V_G, H_G)$, with $V_G = \{x_1,\dots,x_{q+1},y\}$ and $H_G = \{\{y\},\{y,x_1\}\} \cup \{\{x_i,x_{q+1}\}, i \in \{1,\dots,q\}\}$, together with $\preceq_1 = \preceq_{SOV}$ ($\{x_1,\dots,x_q\} \prec_1 y \prec_1 x_{q+1}$), the constrained induced-width is $w_G(\preceq_1) = q$, because x_{q+1} is then necessarily eliminated first (eliminating x_{q+1} from G creates the hyperedge $\{x_1,\dots,x_q\}$ of size q).

However, the scopes of the functions involved enable us to write the quantity to compute as $\max_{x_1}\left(\left(\sum_y \varphi_y \times \varphi_{y,x_1}\right) \times \left(\max_{x_2,\dots,x_{q+1}}\left(\prod_{i\in\{1,\dots,q\}} \varphi_{x_i,x_{q+1}}\right)\right)\right)$. This rewriting shows that the only constraint on the elimination order is that x_1 must be eliminated before y. This constraint, modeled by $\preceq_2$ defined by $x_1 \prec_2 y$, gives $w_G(\preceq_2) = 1$ (e.g. with the elimination order $x_1 \prec x_{q+1} \prec x_2 \prec x_3 \prec \dots \prec x_q \prec y$). Hence, the complexity decreases from $O((q+2)\cdot d^{1+q})$ to $O((q+2)\cdot d^2)$ (there is a $q+2$ factor because there are $q+2$ scoped functions).

This example shows that defining constraints on the elimination order from the sequence of operator-variables only is uselessly strong and may be exponentially suboptimal compared to a method considering the scopes of the functions involved. It is also obvious that weakening constraints on the elimination order can only decrease the constrained induced-width: if $G = (V_G, H_G)$ is a hypergraph and if $\preceq_1$, $\preceq_2$ are two partial orders on V_G such that $(x \preceq_2 y) \to (x \preceq_1 y)$ ($\preceq_2$ is weaker than $\preceq_1$), then $w_G(\preceq_1) \geq w_G(\preceq_2)$.

Working on the hypergraph There may exist decompositions enabling to use more than just the distributivity of $\otimes$ over $\oplus$.

Indeed, let us consider the QCSP $\exists x_1 \dots \exists x_q \forall y \left(\varphi_{x_1,y} \wedge \dots \wedge \varphi_{x_q,y}\right)$. Using $G_1 = (\{x_1,\dots,x_q,y\}, \{\{x_i,y\}, i \in \{1,\dots,q\}\})$ and $\preceq_1$ defined by $\{x_1,\dots,x_q\} \prec_1 y$ gives $w_{G_1}(\preceq_1) = q$ (because y is then necessarily eliminated first). However, it is possible to *duplicate* y and write $\exists x_1 \dots \exists x_q \forall y \left(\varphi_{x_1,y} \wedge \dots \wedge \varphi_{x_q,y}\right) = \exists x_1,\dots,\exists x_q \left(\left(\forall y_1 \varphi_{x_1,y_1}\right) \wedge \dots \wedge \left(\forall y_q \varphi_{x_q,y_q}\right)\right)$. The complexity is then given by $G_2 = (\{x_1,\dots,x_q,y_1,\dots,y_q\}, \{\{x_i,y_i\}, i \in \{1,\dots,q\}\})$ and $\preceq_2$ defined by $x_i \prec_2 y_i$, leading to the constrained induced-width $w_{G_2}(\preceq_2) = 1$. Therefore, duplicating y decreases the theoretical complexity from $O(q \cdot d^{q+1})$ to $O(q \cdot d^2)$.

Proposition 1 shows that such a duplication mechanism can be used only in one specific case, applicable for eliminations with $\forall$ on QBF, QCSP, or with min on possibilistic optimistic MDPs. Proposition 2 proves that duplicating is always better than not duplicating.

Proposition 1. *Let $(E, \{\oplus^i, i \in I\}, \otimes)$ be a MCS and let $\oplus \in \{\oplus^i, i \in I\}$. Then, $(\oplus_x (\varphi_1 \otimes \varphi_2) = (\oplus_x \varphi_1) \otimes (\oplus_x \varphi_2)$ for all scoped functions φ_1, $\varphi_2) \leftrightarrow (\oplus = \otimes)$.*

Proof. *If $\oplus = \otimes$, then $\oplus_x(\varphi_1 \oplus \varphi_2) = (\oplus_x \varphi_1) \oplus (\oplus_x \varphi_2)$ by commutativity and associativity of $\oplus$. Conversely, assume that for all scoped functions φ_1, φ_2, $\oplus_x(\varphi_1 \otimes \varphi_2) = (\oplus_x \varphi_1) \otimes (\oplus_x \varphi_2)$. As $(E, \{\oplus^i, i \in I\}, \otimes)$ is a MCS, $\otimes$ has an identity $1_\otimes$ and $\oplus$ has an identity $0_\oplus$. Let us consider a boolean variable x and two scoped functions φ_1, φ_2 of scope x, s.t. $\varphi_1((x,t)) = a$, $\varphi_1((x,f)) = \varphi_2((x,t)) = 1_\otimes$, $\varphi_2((x,f)) = b$. Then, the initial assumption implies that $(a \otimes 1_\otimes) \oplus (1_\otimes \otimes b) = (a \oplus 1_\otimes) \otimes (1_\otimes \oplus b)$, i.e. $a \oplus b = (a \oplus 1_\otimes) \otimes (1_\otimes \oplus b)$. Taking $a = b = 0_\oplus$ gives $0_\oplus = 1_\otimes$. Consequently, for all $a, b \in E$, $a \oplus b = (a \oplus 1_\otimes) \otimes (1_\otimes \oplus b) = (a \oplus 0_\oplus) \otimes (0_\oplus \oplus b) = a \otimes b$, i.e. $\oplus = \otimes$.* $\square$

Note that $\oplus = \otimes$ implies that $\oplus$ is idempotent: indeed, given the properties of a MCS, "$\oplus = \otimes$" implies that $a \oplus a = a \otimes (1_\otimes \oplus 1_\otimes) = a \otimes (1_\otimes \oplus 0_\oplus) = a$.

Proposition 2. *Let $(E, \{\oplus^i, i \in I\}, \otimes)$ be a MCS and let $\oplus \in \{\oplus^i, i \in I\}$. Let φ_{x,S_j} be a scoped function of scope $\{x\} \cup S_j$ for all $j \in \{1, \ldots, m\}$. The direct computation of $\psi = \oplus_x(\varphi_{x,S_1} \otimes \cdots \otimes \varphi_{x,S_m})$ always requires more operations than the one of $(\oplus_x \varphi_{x,S_1}) \otimes \cdots \otimes (\oplus_x \varphi_{x,S_m})$. Moreover, the direct computation of ψ results in a time complexity $O(m \cdot d^{1+|S_1 \cup \ldots \cup S_m|})$, whereas the one of the m quantities in the set $\left\{\oplus_x \varphi_{x,S_j} \mid j \in \{1, \ldots, m\}\right\}$ is $O(m \cdot d^{1+\max_{j \in \{1,\ldots,m\}} |S_j|})$.*

Proof. *It can be shown that computing directly $\oplus_x(\varphi_{x,S_1} \otimes \cdots \otimes \varphi_{x,S_m})$ requires $n_1 = |dom(S_1 \cup \ldots \cup S_m)|(m|dom(x)| - 1) = O(md^{1+|S_1 \cup \ldots \cup S_m|})$ operations. Directly computing the quantities in $\left\{\oplus_x \varphi_{x,S_j} | j \in \{1, \ldots, m\}\right\}$ requires $n_2 = (\sum_{j \in \{1,\ldots,m\}} |dom(S_j)|) \cdot (|dom(x)| - 1) = O(m \cdot d^{1+\max_{j \in \{1,\ldots,m\}} |S_j|})$ operations. Directly computing $(\oplus_x \varphi_{x,S_1}) \otimes \cdots \otimes (\oplus_x \varphi_{x,S_m})$ therefore requires $n_3 = n_2 + |dom(S_1 \cup \ldots \cup S_m)|(m-1)$ operations. The result follows from $n_1 - n_3 = (|dom(x)| - 1)(m|dom(S_1 \cup \ldots \cup S_m)| - \sum_{j \in \{1,\ldots,m\}} |dom(S_j)|) \geq 0$.* $\square$

3.2 Towards a tree of mono-operator queries

The constrained induced-width can be decreased and exponential gains in complexity obtained thanks to an accurate multi-operators query analysis. The latter corresponds to determining the actual constraints on the elimination order and the possible additional decompositions using duplication. To systematize it, we introduce rewriting rules transforming an initial unstructured multi-operator query into a tree of mono-operator ones

The basic elements used for such a transformation are computation nodes.

Definition 7. *A computation node n on a MCS $(E, \{\oplus^i, i \in I\}, \otimes)$ is:*

- *either a scoped function φ (*atomic *computation node); the value of n is then $val(n) = \varphi$ and its scope is $sc(n) = sc(\varphi)$;*
- *or a pair (SOV, N) s.t. SOV is a sequence of operator-variables on a set of variables S and N is a set of computation nodes; the value of n is then $val(n) = SOV(\otimes_{n' \in N} val(n'))$, the set of variables it eliminates is $V_e(n) = S$, its scope is $sc(n) = (\cup_{n' \in N} sc(n')) - V_e(n)$, and the set of its sons is $Sons(n) = N$.*

We extend the previous definitions to sets of computation nodes N by $val(N) = \otimes_{n' \in N} val(n')$, $sc(N) = \cup_{n' \in N} sc(n')$, and, if all nodes in N are non-atomic, then $V_e(N) = \cup_{n' \in N} V_e(n')$ and $Sons(N) = \cup_{n' \in N} Sons(n')$. Moreover, for all $\oplus \in \{\oplus^i, i \in I\}$, we define the set of nodes in N performing eliminations with $\oplus$ by $N[\oplus] = \{n \in N \mid n = (\oplus_S, N')\}$.

For example, if $N = \{(\min_{x,y}, N_1), (\sum_z, N_2), (\min_t, N_3)\}$, then $N[\min] = \{(\min_{x,y}, N_1), (\min_t, N_3)\}$ and $N[+] = \{(\sum_z, N_2)\}$. Informally, a computation node (SOV, N) specifies a sequence of eliminations on the combination of its sons and can be seen as the root of a tree of computation nodes. It can be represented as in Figure 1. Given a set of computation nodes N, we define N^{+x} (resp. N^{-x}) as the set of nodes of N whose scope contains x (resp. does not contain x): $N^{+x} = \{n \in N \mid x \in sc(n)\}$ (resp. $N^{-x} = \{n \in N \mid x \notin sc(n)\}$).

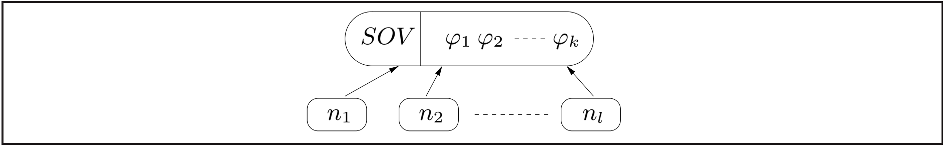


Fig. 1: A computation node (SOV, N). Note that atomic sons (in $N \cap \Phi = \{\varphi_1, \ldots, \varphi_k\}$) and non-atomic ones (in $N - \Phi = \{n_1, \ldots, n_l\}$) are distinguished.

The value of computation nodes can easily be linked to the answer to a query. Indeed, given a query $Q = (SOV, (V, \Phi))$ defined on a MCS $(E, \{\oplus^i, i \in I\}, \otimes)$, $Ans(Q) = val(n_0)$ where $n_0 = (SOV, \Phi)$. The problem consists in rewriting n_0 so as to exhibit the query structure. To do so, we consider each variable from the right to the left of SOV, using an elimination order o *compatible* with SOV (cf Definition 4), and *simulate* the decomposition induced by the elimination of the $|V|$ variables from the right to the left of $SOV(o)$. More precisely, we start from the initial Computation Nodes Tree (CNT):

$$CNT_0(Q, o) = (SOV(o), \Phi)$$

In the example in Figure 2, this initial CNT corresponds to the first node. For all $k \in \{0, \ldots, |V| - 1\}$, the macrostructure at step $k+1$, denoted $CNT_{k+1}(Q, o)$, is obtained from $CNT_k(Q, o)$ by considering the rightmost remaining elimination and by applying two types of rewriting rules:

1. A *decomposition rule DR*, using the distributivity of the elimination operators over $\otimes$ (so that when eliminating a variable x, only scoped functions with x in their scopes are considered) together with possible duplications. Note that DR implements both types of decompositions.

$$\boxed{DR} \quad (sov.\oplus_x, N) \rightsquigarrow \begin{cases} (sov, N^{-x} \cup \{(\oplus_x, \{n\}) \mid n \in N^{+x}\}) \text{ if } \oplus = \otimes \\ (sov, N^{-x} \cup \{(\oplus_x, N^{+x})\}) \text{ otherwise} \end{cases}$$

   In Figure 2, DR transforms the initial structure $CNT_0(Q, o) = (\min_{x_1} \max_{x_2} \max_{x_3} \min_{x_4} \max_{x_5}, \{\varphi_{x_3,x_4}, \varphi_{x_1,x_4}, \varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})$ to $CNT_1(Q, o) = (\min_{x_1} \max_{x_2} \max_{x_3} \min_{x_4}, \{\varphi_{x_3,x_4}, \varphi_{x_1,x_4}, (\max_{x_5}, \{\varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})$ (case $\oplus \neq \otimes$). Eliminating x_4 using $\min = \otimes$ then transforms $CNT_1(Q, o)$ to $CNT_2(Q, o) = (\min_{x_1} \max_{x_2} \max_{x_3}, \{(\min_{x_4}, \{\varphi_{x_3,x_4}\}), (\min_{x_4}, \{\varphi_{x_1,x_4}\}), (\max_{x_5}, \{\varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})$.

2. A *recomposition rule RR* which aims at revealing freedoms in the elimination order for the nodes created by DR.

$$\boxed{RR} \quad (\oplus_x, N) \rightsquigarrow \big(\oplus_{x \cup V_e(N[\oplus])}, (N - N[\oplus]) \cup Sons(N[\oplus])\big)$$

   In Figure 2, RR transforms the computation node $(\min_{x_1} \max_{x_2}, \{(\min_{x_4}, \{\varphi_{x_1,x_4}\}), (\max_{x_3}, \{(\min_{x_4}, \{\varphi_{x_3,x_4}\}), (\max_{x_5}, \{\varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})\})$ into $CNT_3(Q,o) = (\min_{x_1} \max_{x_2}, \{(\min_{x_4}, \{\varphi_{x_1,x_4}\}), (\max_{x_3,x_5}, \{(\min_{x_4}, \{\varphi_{x_3,x_4}\}), \varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})$, because the structure shows that although $x_3 \prec_{SOV} x_5$, there is actually no need to eliminate x_5 before x_3. RR cannot make one miss a better variable ordering, since what is recomposed will always be decomposable again (using the techniques of Section 4).

More formally, for rewriting rule $RR : n_1 \rightsquigarrow n_2$, let us denote $n_2 = RR(n_1)$. Then, for all $k \in \{0, \ldots, |V|-1\}$, $CNT_{k+1}(Q,o) = rewrite(CNT_k(Q,o))$, where

$$rewrite((sov \cdot \oplus_x, N)) = \begin{cases} (sov, N^{-x} \cup \{RR((\oplus_x, \{n\})), n \in N^{+x}\}) \text{ if } \oplus = \otimes \\ (sov, N^{-x} \cup \{RR((\oplus_x, N^{+x}))\}) \text{ otherwise} \end{cases}$$

This means that when eliminating variable x, we decompose the computations (using duplication if $\oplus = \otimes$), and recompose the created nodes in order to reveal freedoms in the elimination order. At each step, a non-duplicated variable appears *once* in the tree and a duplicated one appears at most *once in each branch* of the tree. The final computation nodes tree, denoted $CNT(Q,o)$, is

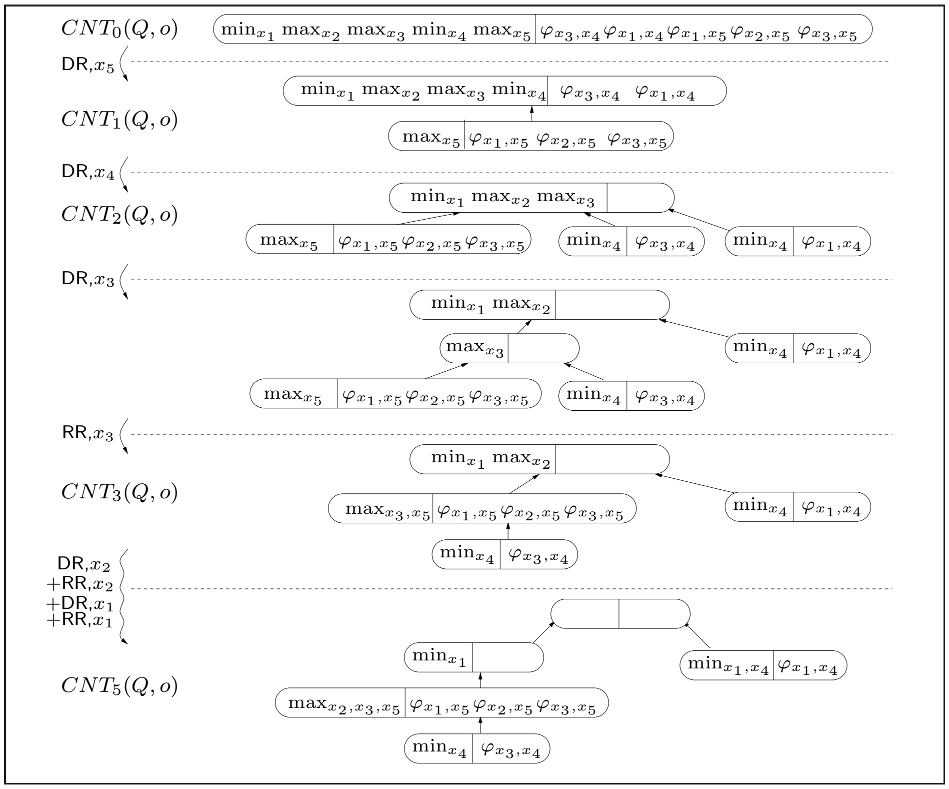
$$CNT(Q,o) = CNT_{|V|}(Q,o) = rewrite^{|V|}(CNT_0(Q,o))$$

3.3 Some good properties of the macrostructure obtained

The soundness of the created macrostructure is provided by Propositions 3 and 4, which show that the rewriting process preserves nodes value.

Proposition 3. *Let $Q = (SOV, \mathcal{N})$ be a query and let $o \in lin(\preceq_{SOV})$. Then, $val(CNT_{k+1}(Q,o)) = val(CNT_k(Q,o))$ for all $k \in \{0, \ldots, |V|-1\}$.*

Proof. We use four lemmas.
Lemma 1. *Rewriting rule $DR : n_1 \rightsquigarrow n_2$ is sound, i.e. $val(n_1) = val(n_2)$ holds.*
Proof of Lemma 1. As $\otimes$ distributes over $\oplus$, $val((sov \cdot \oplus_x, N)) = sov \cdot \oplus_x (\otimes_{n \in N} val(n)) = sov((\otimes_{n \in N^{-x}} val(n)) \otimes \oplus_x (\otimes_{n \in N^{+x}} val(n)))$ (eq1). If $\oplus = \otimes$, Proposition 1 implies that $\oplus_x (\otimes_{n \in N^{+x}} val(n)) = \otimes_{n \in N^{+x}} (\oplus_x val(n)) = val(\{(\oplus_x, \{n\}) \,|\, n \in N^{+x}\})$. Therefore, using (eq1), $val((sov \cdot \oplus_x, N))$ equals $val\big((sov, N^{-x} \cup \{(\oplus_x, n) \,|\, n \in N^{+x}\})\big)$. Otherwise ($\oplus \neq \otimes$), one can just write $\oplus_x (\otimes_{n \in N^{+x}} val(n)) = val\big((\oplus_x, N^{+x})\big)$. This means that (eq1) can be written as $val((sov \cdot \oplus_x, N)) = val\big((sov, N^{-x} \cup \{(\oplus_x, N^{+x})\})\big)$.
Lemma 2. *Let $RR' : (\oplus_S, N_1 \cup \{(\oplus_{S'}, N_2)\}) \rightsquigarrow (\oplus_{S \cup S'}, N_1 \cup N_2)$. If $S' \cap (S \cup sc(N_1)) = \emptyset$ and $N_1 \cap N_2 = \emptyset$, then RR' is a sound rewriting rule.*
Proof of Lemma 2. Given that $\otimes$ distributes over $\oplus$ and $S' \cap sc(N_1) = \emptyset$, one can write $val((\oplus_S, N_1 \cup \{(\oplus_{S'}, N_2)\})) = \oplus_S ((\otimes_{n \in N_1} val(n)) \otimes \oplus_{S'} (\otimes_{n \in N_1} val(n))) = \oplus_S \cdot \oplus_{S'} ((\otimes_{n \in N_1} val(n)) \otimes (\otimes_{n \in N_1} val(n)))$. As $N_1 \cap N_2 = \emptyset$ and $S \cap S' = \emptyset$, the latter quantity also equals $\oplus_{S \cup S'} (\otimes_{n \in N_1 \cup N_2} val(n))$, i.e. $val((\oplus_{S \cup S'}, N_1 \cup N_2))$.


$CNT_0(Q,o)$
$\min_{x_1} \max_{x_2} \max_{x_3} \min_{x_4} \max_{x_5}$ | $\varphi_{x_3,x_4} \varphi_{x_1,x_4} \varphi_{x_1,x_5} \varphi_{x_2,x_5} \varphi_{x_3,x_5}$
DR,x_5
$CNT_1(Q,o)$
$\min_{x_1} \max_{x_2} \max_{x_3} \min_{x_4}$ | φ_{x_3,x_4} φ_{x_1,x_4}
$\max_{x_5}$ | $\varphi_{x_1,x_5} \varphi_{x_2,x_5} \varphi_{x_3,x_5}$
DR,x_4
$CNT_2(Q,o)$
$\min_{x_1} \max_{x_2} \max_{x_3}$
$\max_{x_5}$ | $\varphi_{x_1,x_5} \varphi_{x_2,x_5} \varphi_{x_3,x_5}$
$\min_{x_4}$ | φ_{x_3,x_4}
$\min_{x_4}$ | φ_{x_1,x_4}
DR,x_3
$\min_{x_1} \max_{x_2}$
$\max_{x_3}$
$\min_{x_4}$ | φ_{x_1,x_4}
$\max_{x_5}$ | $\varphi_{x_1,x_5} \varphi_{x_2,x_5} \varphi_{x_3,x_5}$
$\min_{x_4}$ | φ_{x_3,x_4}
RR,x_3
$CNT_3(Q,o)$
$\min_{x_1} \max_{x_2}$
$\max_{x_3,x_5}$ | $\varphi_{x_1,x_5} \varphi_{x_2,x_5} \varphi_{x_3,x_5}$
$\min_{x_4}$ | φ_{x_1,x_4}
$\min_{x_4}$ | φ_{x_3,x_4}
DR,x_2
+RR,x_2
+DR,x_1
+RR,x_1
$CNT_5(Q,o)$
$\min_{x_1}$
$\min_{x_1,x_4}$ | φ_{x_1,x_4}
$\max_{x_2,x_3,x_5}$ | $\varphi_{x_1,x_5} \varphi_{x_2,x_5} \varphi_{x_3,x_5}$
$\min_{x_4}$ | φ_{x_3,x_4}
Fig. 2: Application of the rewriting rules on a QCSP example: $\min_{x_1} \max_{x_2,x_3} \min_{x_4} \max_{x_5}(\varphi_{x_3,x_4} \wedge \varphi_{x_1,x_4} \wedge \varphi_{x_1,x_5} \wedge \varphi_{x_2,x_5} \wedge \varphi_{x_3,x_5})$, with the elimination order $o : x_1 \prec x_2 \prec x_3 \prec x_4 \prec x_5$.

Lemma 3. *$\forall k \in \{0,\dots,|V|\} \forall n = (sov, N) \in CNT_k(Q,o)$, if $\oplus \neq \otimes$, then for all $n' \in N[\oplus]$, $V_e(n') \cap (V_e((N - \{n'\})[\oplus]) \cup sc(N - \{n'\})) = \emptyset$.*

Proof of Lemma 3. The property holds for $k = 0$ since $CNT_0(Q,o) = (SOV, \Phi)$ and $\Phi[\oplus] = \emptyset$. If it holds at step k, it can be shown to hold at $k+1$ (the main point being that DR splits the nodes with x in their scopes and the ones not having x in their scopes)

Lemma 4. *RR is a sound rewriting rule.*

Proof of Lemma 4. If the variable eliminated uses $\oplus \neq \otimes$ as an operator, then, thanks to Lemma 3 and the fact that all computation nodes are distinct, and since variable x considered at step k satisfies $x \notin V_e(N[\oplus])$, it is possible to recursively apply Lemma 2 to nodes in $N[\oplus]$, because the two conditions looking like $S' \cap (S \cup sc(N_1))$ and $N_1 \cap N_2$ then always hold. This shows that RR is sound when $\oplus \neq \otimes$. If $\oplus = \otimes$, then the nodes to recompose look like $(\oplus_x, \{(\oplus_S, N')\})$. As $S \cap \{x\} = \emptyset$, Lemma 3 entails that RR is sound.

As both DR and RR are sound, Proposition 3 holds. □

Proposition 4. *Let $Q = (SOV, \mathcal{N})$ be a query. Then, $val(CNT(Q,o)) = Ans(Q)$ for all $o \in lin(\preceq_{SOV})$.*

Proof. Follows from Proposition 3 and from $val(CNT_0(Q,o)) = Ans(Q)$. □

Independence with regard to the linearization of $\preceq_{SOV}$ Proposition 5 shows that the final tree of computation nodes is independent from the arbitrary elimination order o compatible with SOV chosen at the beginning. In this sense, the structure obtained is a *unique fixed point* which can be denoted simply by $CNT(Q)$.

Proposition 5. *Let $Q = (SOV, \mathcal{N})$ be a query. Then, for all $o, o' \in lin(\preceq_{SOV})$, $CNT(Q, o) = CNT(Q, o')$*

Sketch of the proof. (a) It can be shown that for all $\oplus \in \{\oplus^i, i \in I\}$, if $CNT = (sov \cdot \oplus_x \cdot \oplus_y, N)$ and $CNT' = (sov \cdot \oplus_y \cdot \oplus_x, N)$, then $rewrite^2(CNT) = rewrite^2(CNT')$. (b) Given an elimination order $o \in lin(\preceq_{SOV})$, any elimination order $o' \in lin(\preceq_{SOV})$ can be obtained from o by successive permutations of adjacent eliminations. (a) and (b) entail that $CNT(Q, o) = CNT(Q, o')$. □

3.4 Comparison with an unstructured approach

Building the macrostructure of a query can induce exponential gains in theoretical complexity, as shown in Section 3.1. Stronger results can be stated, proving that the structured approach is always as least as good as existing approaches in terms of constrained induced-width.

Let us define the width w_n of a node $n = (\oplus_S, N)$ as the induced width of the hypergraph $G = (sc(N), \{sc(n'), n' \in N)$ for the elimination of the variables in S (i.e. the minimum size, among all elimination orders of S, of the largest hyperedge created when eliminating variables in S from G). The induced-width of a tree of computation nodes CNT is $w_{CNT} = \max_{n \in CNT} w_n$. One can say that $1 + w_{CNT}$ is the maximum number of variables to consider simultaneously when using an optimal elimination order in a VE algorithm. Theorem 1 shows that the macrostructuration of a query can only decrease the induced-width.

Theorem 1. *Let $Q = (SOV, (V, \Phi))$ be a query and $G = (V, \{sc(\varphi), \varphi \in \Phi\})$. Then, $w_{CNT(Q)} \leq w_G(\preceq_{SOV})$.*

Sketch of the proof. Let o^* be an elimination order s.t. $w_G(\preceq_{SOV}) = w_G(o^*)$. The idea is to apply the rewriting rules on $CNT_0(Q, o^*)$. Let H_k denote the set of hyperedges in the hypergraph G_k obtained after the k first eliminations in o^*. More precisely, $G_0 = G$ and, if $G_k = (V_k, H_k)$ and x is eliminated, then $G_{k+1} = (V_k - \{x\}, (H_k - H_k^{+x}) \cup \{h_{k+1}\})$, where $h_{k+1} = \cup_{h \in H_k^{+x}} h - \{x\}$ is the hyperedge created from step k to $k+1$. It can be proved that for all $k \in \{0, \ldots, |V| - 1\}$, if $CNT_k(Q, o^*) = (sov \cdot \oplus_x, N)$, then for all $n \in N$, there exists $h \in H_k$ s.t. $sc(n) \subset sc(h)$. This property easily holds at step 0, and if it holds at step k, then $sc((\oplus_x, N^{+x})) \subset sc(h_{k+1})$. Moreover, if duplication is used, then for all $n \in N^{+x}$, $sc((\oplus_x, \{n\})) \subset sc(h_{k+1})$. Rewriting rule RR can be shown to be always advantageous in terms of induced-width. This entails the required result. □

For the QCSP example in Figure 2, $w_{CNT(Q)} = 1$, whereas the initial constrained induced-width is $w_G(\preceq_{SOV}) = 3$ (and without duplication, $w_{CNT(Q)}$ would equal 2): the complexity decreases from $O(|\Phi| \cdot d^4)$ to $O(|\Phi| \cdot d^2)$.

More important gaps between $w_{CNT(Q)}$ and $w_G(\preceq_{SOV})$ can be observed on larger problems. More precisely, we performed experiments on instances of the QBF library (only a limited number are reported here). The results are shown in Table 2. In order to compute induced-widths and constrained induced-widths, we use usual junction tree construction techniques with the so-called *min-fill* heuristic. The results show that there can be no gain in analyzing the macrostructure of queries, as is the case for instances of the "robot" problem (which involve only 3 alternations of elimination operators), but that as soon as the number of alternation increases, revealing freedoms in the elimination order can be greatly beneficial. Note that these results provide a theoretical explanation to the experimental gains observed when using *quantifier trees* on QBF [5].

Theorem 1 shows that working directly on the structure obtained can be a good option, because it can decrease the induced-width. However, given an existing solver, an alternative approach is to see the macrostructuration of a query only as a useful preprocessing step revealing freedoms in the elimination order, thanks to Proposition 6.

Proposition 6. *Let $Q = (SOV, (V, \Phi))$ be a query. Assume that duplication is not used. $CNT(Q)$ induces a partial order $\preceq_{CNT(Q)}$ on V, defined by "if $((\oplus_{S_1}, N \cup \{(\oplus'_{S_2}, N')\}) \in CNT(Q))$, then for all $x \in S_1 \cap sc(N')$, $x \prec_{CNT(Q)} S_2$. Then, for all $o \in lin(\preceq_{CNT(Q)})$, $SOV(o)\,(\otimes_{\varphi\in\Phi}\,\varphi) = Ans(Q)$. Moreover, $\preceq_{CNT(Q)}$ is weaker than $\preceq_{SOV}$.*

Sketch of the proof. The idea is that if $o \in lin(\preceq_{CNT(Q)})$, it is possible to do the inverse operations of RR and DR, considering first smallest variables in o. These inverse operations are naturally sound and lead to the structure $(SOV(o), \Phi)$, which proves that $SOV(o)\,(\otimes_{\varphi\in\Phi}\,\varphi) = Ans(Q)$.

If $o \in lin(\preceq_{SOV})$ and $x \preceq_o y$, then, for all $n = (\oplus_{S_1}, N \cup \{(\oplus'_{S_2}, N')\}) \in CNT(Q) = CNT(Q, o)$, it is impossible that $y \in S_1$ and $x \in S_2$ (because y is considered before x during the rewriting process). As this holds for all x, y such that $x \preceq_o y$, this entails that $\neg(y \preceq_{CNT(Q)} x)$. $(x \preceq_o y) \rightarrow \neg(y \preceq_{CNT(Q)} x)$ can also be written $(y \preceq_{CNT(Q)} x) \rightarrow (y \prec_o x)$, which implies that $o \in lin(CNT(Q))$. Therefore, $lin(\preceq_{SOV}) \subset lin(CNT(Q))$, i.e. $\preceq_{CNT(Q)}$ is weaker than $\preceq_{SOV}$ □

Problem instance	w	w'	nbv,nbc,nba	Problem instance	w	w'	nbv,nbc,nba
adder-2-sat	12	24	332, 113, 5	k-branch-n-1	22	43	133, 314, 7
adder-4-sat	28	101	726, 534, 5	k-branch-n-2	39	103	294, 793, 9
adder-8-sat	60	411	1970, 2300, 5	k-branch-n-3	54	185	515, 1506, 11
adder-10-sat	76	644	2820, 3645, 5	k-branch-n-4	70	296	803, 2565, 13
adder-12-sat	92	929	3822, 5298, 5	k-branch-n-5	89	427	1149, 3874, 15
robots-1-5-2-1.6	2213	2213	6916, 23176, 3	k-branch-n-6	107	582	1557, 5505, 17
robots-1-5-2-1.7	1461	1461	7904, 26810, 3	k-branch-n-7	131	761	2027, 7482, 19
robots-1-5-2-1.8	3933	3933	8892, 30444, 3	k-branch-n-8	146	973	2568, 10117, 21
robots-1-5-2-1.9	1788	1788	9880, 34078, 3	k-branch-n-9	166	1201	3163, 12930, 23

Table 2. Comparison between $w = w_{CNT(Q)}$ and $w' = w_G(\preceq_{SOV})$ on some instances of the QBF library (*nbv*, *nbc*, *nba* denote respectively the number of variables, the number of clauses, and the number of elimination operator alternations of an instance).

3.5 Complexity results

The macrostructure is usable only if its computation is tractable. Based on the algorithm in Figure 3, implementing the macrostructuration of a query, Proposition 7 gives an upper bound on the complexity, showing that rewriting a query as a tree of mono-operator computation nodes is easy.

```
begin
    $root \leftarrow \text{newNode}(\emptyset, \emptyset, \Phi, \emptyset)$
    while $(SOV = SOV' \cdot \oplus_x)$ do
        $SOV \leftarrow SOV'$
        if $\oplus \neq \otimes$ then
            $n \leftarrow \text{newNode}(\oplus, \{x\}, \emptyset, \emptyset)$
            foreach $n' \in Sons(root)$ $s.t.$ $x \in sc(n')$ do
                $sc(n) \leftarrow sc(n) \cup sc(n')$
                $Sons(root) \leftarrow Sons(root) - \{n'\}$
                if $op(n') = \oplus$ then
                    $V_e(n) \leftarrow V_e(n) \cup V_e(n')$
                    $Sons(n) \leftarrow Sons(n) \cup Sons(n')$
                else $Sons(n) \leftarrow Sons(n) \cup \{n'\}$
            $sc(n) \leftarrow sc(n) - \{x\}$
            $Sons(root) \leftarrow Sons(root) \cup \{n\}$
        else
            foreach $n' \in Sons(root)$ $s.t.$ $x \in sc(n')$ do
                if $op(n') = \oplus$ then
                    $V_e(n') \leftarrow V_e(n') \cup \{x\}$
                    $sc(n') \leftarrow sc(n') - \{x\}$
                else
                    $n \leftarrow \text{newNode}(\oplus, \{x\}, \{n'\}, sc(n') - \{x\})$
                    $Sons(root) \leftarrow (Sons(root) - \{n'\}) \cup \{n\}$
    return $(root)$
end
```

Fig. 3: MacroStruct$(SOV, (V, \Phi))$ (instruction newNode$(op, V_e, Sons, sc)$ creates a computation node $n = (op_{V_e}, Sons)$ and sets $sc(n)$ to sc.

In the algorithm in Figure 3, the root node of the tree of computation nodes is rewritten. With each node $n = (op_S, N)$ are associated an operator $op(n) = op$, a set of sons $Sons(n) = N$ modeled as a list, and a set of variables eliminated $V_e(n) = S$ modeled as a list too. The scope of n is modeled using a table of $|V|$ booleans. As long as the sequence of operator-variables is not empty, the rightmost remaining elimination is considered. The pseudo-code just implements the rewrite function, which dissociates the cases $\oplus \neq \otimes$ and $\oplus = \otimes$.

Proposition 7. *The time and space complexity of the algorithm in Figure 3 are $O(|V|^2 \cdot |\Phi|)$ and $O(|V| \cdot |\Phi|)$ respectively (if $\Phi \neq \emptyset$ and $V \neq \emptyset$).*

Proof. *At each rewriting step and for each son n' of the root node, tests like "$x \in sc(n')$" and operations like "$sc(n) \leftarrow sc(n) \cup sc(n')$" or "$sc(n') \leftarrow sc(n') - \{x\}$" are $O(|V|)$, since a scope is represented as a table of size $|V|$. Operations like "$Sons(root) \leftarrow Sons(root) - \{n'\}$", "$Sons(root) \leftarrow Sons(root) \cup \{n\}$", "$V_e(n) \leftarrow V_e(n) \cup V_e(n')$" (with $V_e(n) \cap V_e(n') = \emptyset$), or "$V_e(n) \leftarrow V_e(n) \cup \{x\}$" are $O(1)$, since V_e and $Sons$ are represented as lists. Therefore, the operations performed for each rewriting step and for each son of the root are $O(|V|)$. As at each step, $|Sons(root)| \leq |\Phi|$, and as there are $|V|$ rewriting steps, the algorithm is time $O(|V|^2 \cdot |\Phi|)$. As for the space complexity, given that only the scopes of the root sons are used, we need a space $O(|V| \cdot |\Phi|)$ for the scopes. As it can be shown that the number of nodes in the tree of computation nodes is always $O(|V| + |\Phi|)$, recording $op(n)$ and $Sons(n)$ for all nodes n is $O(|V| + |\Phi|)$ too. Last, recording $V_e(n)$ for all nodes n is $O(|V| \cdot |\Phi|)$ because the sum of the number of variables eliminated in each node is lesser than $|V| \cdot |\Phi|$ (the worst case occurs when all variables are duplicated). Hence, the overall space complexity is $O(|V| \cdot |\Phi|)$.* □

4 Decomposing computation nodes

4.1 From computation nodes to multi-operator cluster trees

Once the macrostructure is built (in the form of a tree of mono-operator computation nodes), we use freedoms in the elimination order so as to minimize the induced-width. As $(E, \oplus, \otimes)$ is a commutative semiring for every $\oplus \in \{\oplus^i, i \in I\}$, this can be achieved by decomposing each mono-operator computation node into a cluster tree using usual cluster tree construction techniques. This cluster tree is obtained by considering for each computation node $n = (op_S, N)$ the hypergraph $G(n) = (\cup_{n \in N} sc(n), \{sc(n), n \in N\})$ associated with it.

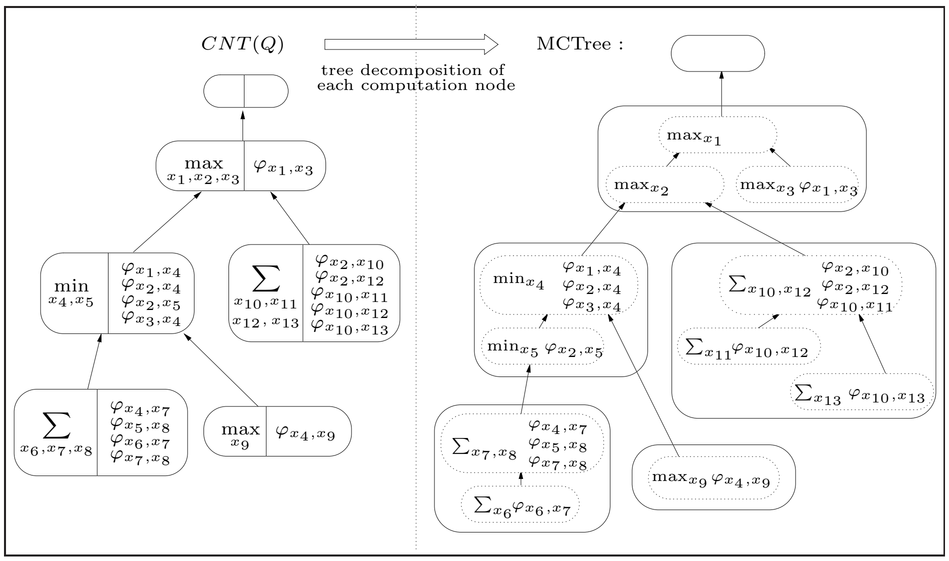
The structure obtained then contains both a macrostructure given by the computation nodes and an internal cluster tree structure given by each of their decompositions. It is then sufficient to choose a root in the cluster tree decomposition [6] of each computation node to obtain a so-called *multi-operator cluster tree* as in Figure 4 (corresponding to an Extended-SSAT [3] problem).

Definition 8. *A Multi-operator Cluster Tree (MCTree) on a MCS $(E, \{\oplus^i, i \in I\}, \otimes)$ is a tree where every vertex c (called a cluster) is labeled with four elements: a set of variables $V(c)$, a set of scoped functions $\Phi(c)$ taking values in E, a set of son clusters $Sons(c)$, and an elimination operator $\oplus(c) \in \{\oplus^i, i \in I\}$. The value of a cluster c is $val(c) = \underset{V(c)-V(pa(c))}{\oplus(c)} \left(\left(\underset{\varphi \in \Phi(c)}{\otimes} \varphi \right) \otimes \left(\underset{s \in Sons(c)}{\otimes} val(s) \right) \right)$.*

It follows from the construction process that if r is the root node of the MCTree associated with a query Q, $val(r) = Ans(Q)$.

4.2 A generic variable elimination algorithm on MCTrees

To define a generic VE algorithm on a MCTree, it suffices to say that as soon as a cluster c has received $val(s)$ from all its children $s \in Sons(c)$, it computes its own value $val(c) = \oplus(c)_{V(pa(c))-V(c)} \left((\otimes_{\varphi \in \Phi(c)} \varphi) \otimes (\otimes_{s \in Sons(c)} val(s)) \right)$ and sends it to $pa(c)$, its parent in the MCTree. The value of the root cluster then equals the answer to the query.



$CNT(Q)$
tree decomposition of each computation node
MCTree :
$\max_{x_1,x_2,x_3}$ φ_{x_1,x_3}
$\min_{x_4,x_5}$ φ_{x_1,x_4} φ_{x_2,x_4} φ_{x_2,x_5} φ_{x_3,x_4}
$\sum_{x_{10},x_{11},x_{12},x_{13}}$ $\varphi_{x_2,x_{10}}$ $\varphi_{x_2,x_{12}}$ $\varphi_{x_{10},x_{11}}$ $\varphi_{x_{10},x_{12}}$ $\varphi_{x_{10},x_{13}}$
$\sum_{x_6,x_7,x_8}$ φ_{x_4,x_7} φ_{x_5,x_8} φ_{x_6,x_7} φ_{x_7,x_8}
$\max_{x_9}$ φ_{x_4,x_9}
$\max_{x_1}$
$\max_{x_2}$
$\max_{x_3} \varphi_{x_1,x_3}$
$\min_{x_4}$ φ_{x_1,x_4} φ_{x_2,x_4} φ_{x_3,x_4}
$\min_{x_5} \varphi_{x_2,x_5}$
$\sum_{x_{10},x_{12}}$ $\varphi_{x_2,x_{10}}$ $\varphi_{x_2,x_{12}}$ $\varphi_{x_{10},x_{11}}$
$\sum_{x_{11}} \varphi_{x_{10},x_{12}}$
$\sum_{x_{13}} \varphi_{x_{10},x_{13}}$
$\sum_{x_7,x_8}$ φ_{x_4,x_7} φ_{x_5,x_8} φ_{x_7,x_8}
$\sum_{x_6} \varphi_{x_6,x_7}$
$\max_{x_9} \varphi_{x_4,x_9}$

Fig. 4: Example of a MCTree obtained from $CNT(Q)$. Note that a cluster c is represented by 1) the set $V(c) - V(pa(c))$ of variables it eliminates, its elimination operator $op(c)$, and the set of function $\Phi(c)$ associated with it, all these elements being put in a pointwise box; 2) the set of its sons, pointing to it in the structure.

5 Conclusion

Solving multi-operator queries using only the sequence of elimination to define constraints on the elimination order is easy but does not take advantage of the actual structure of such queries. Performing a preprocessing finer analysis taking into account both the function scopes and operator properties can reveal extra freedoms in the elimination order as well as decompositions using more than just the distributivity of the combination operator over the elimination operators. This analysis transforms an initial unstructured multi-operator query into a tree of mono-operator computation nodes. The obtained *macrostructure* is always as least as good as the unstructured query in terms of induced-width, which can induce exponential gains in complexity. It is then possible to define a generic VE algorithm on Multi-operator Cluster Trees (MCTrees) by building a cluster-tree decomposition of each mono-operator computation node. Performing such a work using generic algebraic operators makes it applicable to various frameworks (QBF, QCSP, SCSP, SSAT, BN, MDPs).

Other algorithms than VE could be designed on MCTrees, such as a tree search enhanced by branch and bound techniques, e.g. in an AND/OR search [18] or a backtrack bounded by tree decomposition (BTD-like [19]) scheme. Ideas from the game theory field like the alpha-beta algorithm [20] can also be considered. This work was partially conducted within the EU IP COGNIRON ("The Cognitive Companion") funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

References

1. Mackworth, A.: Consistency in Networks of Relations. Artificial Intelligence **8** (1977) 99–118
2. Bordeaux, L., Monfroy, E.: Beyond NP: Arc-consistency for Quantified Constraints. In: Proc. of the 8th International Conference on Principles and Practice of Constraint Programming (CP-02), Ithaca, New York, USA (2002)
3. Littman, M., Majercik, S., Pitassi, T.: Stochastic Boolean Satisfiability. Journal of Automated Reasoning **27** (2001) 251–296
4. Walsh, T.: Stochastic Constraint Programming. In: Proc. of the 15th European Conference on Artificial Intelligence (ECAI-02), Lyon, France (2002)
5. Benedetti, M.: Quantifier Trees for QBF. In: Proc. of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT-05), St. Andrews, Scotland (2005)
6. Kjaerulff, U.: Triangulation of Graphs - Algorithms Giving Small Total State Space. Technical Report R 90-09, Aalborg University, Denmark (1990)
7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
8. Giang, P., Shenoy, P.: A Qualitative Linear Utility Theory for Spohn's Theory of Epistemic Beliefs. In: Proc. of the 16th International Conference on Uncertainty in Artificial Intelligence (UAI-00), Stanford, California, USA (2000) 220–229
9. Sabbadin, R.: A Possibilistic Model for Qualitative Sequential Decision Problems under Uncertainty in Partially Observable Environments. In: Proc. of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI-99), Stockholm, Sweden (1999)
10. Puterman, M.: Markov Decision Processes, Discrete Stochastic Dynamic Programming. John Wiley & Sons (1994)
11. Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., Fargier, H.: Semiring-Based CSPs and Valued CSPs: Frameworks, Properties and Comparison. Constraints **4** (1999) 199–240
12. Howard, R., Matheson, J.: Influence Diagrams. In: Readings on the Principles and Applications of Decision Analysis. Menlo Park, CA, USA (1984) 721–762
13. Ndilikilikesha, P.: Potential Influence Diagrams. International Journal of Approximated Reasoning **10** (1994) 251–285
14. Pralet, C., Verfaillie, G., Schiex, T.: From Influence Diagrams to Multioperator Cluster DAGs. In: Proc. of the 22nd International Conference on Uncertainty in Artificial Intelligence (UAI-06), Cambridge, MA, USA (2006)
15. Dechter, R., Fattah, Y.E.: Topological Parameters for Time-Space Tradeoff. Artificial Intelligence **125** (2001) 93–118
16. Jensen, F., Jensen, F., Dittmer, S.: From Influence Diagrams to Junction Trees. In: Proc. of the 10th International Conference on Uncertainty in Artificial Intelligence (UAI-94), Seattle, WA, USA (1994) 367–373
17. Park, J., Darwiche, A.: Complexity Results and Approximation Strategies for MAP Explanations. Journal of Artificial Intelligence Research **21** (2004) 101–133
18. Marinescu, R., Dechter, R.: AND/OR Branch-and-Bound for Graphical Models. In: Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland (2005)
19. Jégou, P., Terrioux, C.: Hybrid Backtracking bounded by Tree-decomposition of Constraint Networks. Artificial Intelligence **146** (2003) 43–75
20. Knuth, D., Moore, R.: An Analysis of Alpha-Beta Pruning. Artificial Intelligence **8** (1975) 293–326