Semiring Labelled Decision Diagrams, Revisited: Canonicity and Spatial Efficiency Issues*

Hélène Fargier[1], **Pierre Marquis**[2], **Nicolas Schmidt**[1,2]
[1]IRIT-CNRS, Université de Toulouse, France
[2]CRIL-CNRS, Université d'Artois, Lens, France
fargier@irit.fr {marquis,schmidt}@cril.fr

Abstract

Existing languages in the valued decision diagrams (VDDs) family, including ADD, AADD, and those of the SLDD family, prove to be valuable target languages for compiling multivariate functions. However, their efficiency is directly related to the size of the compiled formulae. In practice, the existence of canonical forms may have a major impact on the size of the compiled VDDs. While efficient normalization procedures have been pointed out for ADD and AADD the canonicity issue for SLDD formulae has not been addressed so far. In this paper, the SLDD family is revisited. We modify the algebraic requirements imposed on the valuation structure so as to ensure tractable conditioning, optimization and normalization for some languages of the revisited SLDD family. We show that AADD is captured by this family. Finally, we compare the spatial efficiency of some languages of this family, from both the theoretical side and the practical side.

1 Introduction

In configuration problems of combinatorial objects (like cars), there are two key tasks for which short, guaranteed response times are expected: conditioning (propagating the end-user's choices: version, engine, various options ...) and optimization (maintaining the minimum cost of a feasible car satisfying the user's requirements). When the set of feasible objects and the corresponding cost functions are represented as valued CSPs (VCSPs for short see [Schiex *et al.*, 1995]), the optimization task is NP-hard in the general case, so short response times cannot be ensured.

Valued decision diagrams (VDDs) from the families ADD [Bahar *et al.*, 1993], EVBDD [Lai and Sastry, 1992; Lai *et al.*, 1996; Amilhastre *et al.*, 2002] and their generalization SLDD [Wilson, 2005], and AADD [Tafertshofer and Pedram, 1997; Sanner and McAllester, 2005] do not have such a drawback and appear as interesting representation languages for compiling mappings associating valuations with assignments of discrete variables (including utility functions and probability distributions). Indeed, those languages offer tractable conditioning and tractable optimization (under some conditions in the SLDD case). However, the efficiency of these operations is directly related to the size of the compiled formulae. Following [Darwiche and Marquis, 2002], the choice of the target representation language for the compiled forms must be guided by its succinctness. From the practical side, normalization (and all the more canonicity) are also important: subformulae in normalized form can be more efficiently recognized and the canonicity of the compiled formulae facilitates the search for compiled forms of optimal size (see the discussion about it in [Darwiche, 2011]). Indeed, the ability to ensure a unique form for subformulae prevents them from being represented twice or more.

In this paper, the SLDD family [Wilson, 2005] is revisited, focusing on the canonicity and the spatial efficiency issues. We extend the SLDD setting by relaxing some algebraic requirements on the valuation structure. This extension allows us to capture the AADD language as an element of e−SLDD, the revisited SLDD family. We point out a normalization procedure which extends the AADD 's one to some representation languages of e−SLDD. We also provide a number of succinctness results relating some elements of e−SLDD with ADD and AADD. We finally report some experimental results where we compiled some instances of an industrial configuration problem into each of those languages, thus comparing their spatial efficiency from the practical side.

The rest of the paper is organized as follows. Section 2 gives some formal preliminaries on valued decision diagrams. Section 3 presents the e−SLDD family and describes our normalization procedure. In Section 4, succinctness results concerning ADD, some elements of the e−SLDD family, and AADD are pointed out. Section 5 gives and discusses our empirical results about the spatial efficiency of those languages. Finally, Section 6 concludes the paper.

2 Valued Decision Diagrams

Given a finite set $X = \{x_1, \ldots, x_n\}$ of variables where each variable $x \in X$ ranges over a finite domain D_x, we are interested in representing mappings associating an element from a valuation set E with assignments $\vec{x} = \{(x_i, d_i) \mid d_i \in D_{x_i}, i = 1, \ldots, n\}$ ($\vec{X}$ will denote the set of all assignments over X). E is the carrier of a *valuation structure* $\mathcal{E}$, which

*This work is partially supported by the project BR4CP ANR-11-BS02-008 of the French National Agency for Research.

can be more or less sophisticated from an algebraic point of view. A *representation language* given X w.r.t. a valuation structure $\mathcal{E}$ is mainly a set of data structures. The targeted mapping is called the *semantics* of the data structure and the data structure is a *representation* of the mapping:

Definition 1 (representation language) *(inspired from [Gogic* et al.*, 1995]) Given a valuation structure $\mathcal{E}$, a* representation language *$\mathcal{L}$ over X w.r.t. $\mathcal{E}$ is a 4-tuple $\langle C_\mathcal{L}, Var_\mathcal{L}, I_\mathcal{L}, s_\mathcal{L}\rangle$ where $C_\mathcal{L}$ is a set of data structures α (also referred to as $C_\mathcal{L}$ formulae), $Var_\mathcal{L} : C_\mathcal{L} \rightarrow 2^X$ is a scope function associating with each $C_\mathcal{L}$ formula the subset of X it depends on, $I_\mathcal{L}$ is an interpretation function associating with each $C_\mathcal{L}$ formula α a mapping $I_\mathcal{L}(\alpha)$ from the set of all assignments of $Var_\mathcal{L}(\alpha)$ to E, and $s_\mathcal{L}$ is a size function from $C_\mathcal{L}$ to $\mathbb{N}$ providing the size of any $C_\mathcal{L}$ formula.*

Different formulae can share the same semantics:

Definition 2 (equivalent formulae) *Let $\mathcal{L}_1$ (resp. $\mathcal{L}_2$) be a representation language over X w.r.t. $\mathcal{E}_1$ (resp. $\mathcal{E}_2$) where $E_1 = E_2$. $\alpha \in \mathcal{L}_1$ is* equivalent *to $\beta \in \mathcal{L}_2$ iff $Var_{\mathcal{L}_1}(\alpha) = Var_{\mathcal{L}_2}(\beta)$ and $I_{\mathcal{L}_1}(\alpha) = I_{\mathcal{L}_2}(\beta)$.*

In this paper, we are specifically interested in data structures of the form of *valued decision diagrams*:

Definition 3 (valued decision diagram) *A* valued decision diagram *(VDD) over X w.r.t. $\mathcal{E}$ is a finite DAG α with a single root, s.t. every internal node N is labelled with a variable $x \in X$ and if $D_x = \{d_1, \ldots, d_k\}$, then N has k outgoing arcs $a_1, \ldots, a_k$, so that the arc a_i of α is valued by $v(a_i) = d_i$. We note $out(N)$ (resp. $in(N)$) the arcs outgoing from (resp. incoming to N). Nodes and arcs can also be labelled by elements of E: if N (resp. a_i) is node (resp. an arc) of α, then $\phi(N)$ (resp. $\phi(a_i)$) denotes the label of N (resp. a_i). Finally, each VDD α is a* read-once *formula, i.e., for each path from the root of α to a sink, every variable $x \in X$ occurs at most once as a node label.*

When *ordered* VDDs are considered, a total ordering over X is chosen and for each path from the root of α to a sink, the associated sequence of internal node labels is required to be compatible w.r.t. this variable ordering.

The key problems we focus on are the *conditioning problem* (given a $C_\mathcal{L}$ formula α over X w.r.t. $\mathcal{E}$ and an assignment $\vec{y} \in \vec{Y}$ where $Y \subseteq X$, compute a $C_\mathcal{L}$ formula representing the restriction of $I_\mathcal{L}(\alpha)$ by $\vec{y}$) and the *optimization problem* (given a $C_\mathcal{L}$ formula α over X w.r.t. $\mathcal{E}$, find an assignment $\vec{x^*} \in \vec{X}$ such that $I_\mathcal{L}(\alpha)(\vec{x^*})$ is not dominated w.r.t. some relation $\succeq$ over E – typically, $\succeq$ is a total order). Conditioning is an easy operation on a VDD α. Mainly, for each $(y, d_i) \in \vec{y}$, just by-pass in α every node N labeled by y by linking directly each of its parents to the child N_i of N such that $v((N, N_i)) = d_i$ (N and all its outgoing arcs are thus removed). However, optimization is often more demanding, depending on the family of VDDs under consideration.

ADD, SLDD, and AADD are representation languages composed of valued decision diagrams. The scope functions $Var_{\mathtt{ADD}}$, $Var_{\mathtt{SLDD}}$, and $Var_{\mathtt{AADD}}$ are the same ones and they return the set of variables $Var(\alpha)$ from X where each $x \in Var(\alpha)$ labels at least one node in α. The size functions $s_{\mathtt{ADD}}$, $s_{\mathtt{SLDD}}$, and $s_{\mathtt{AADD}}$ are closely related: the size of a (labelled) decision graph α is the size of the graph (number of nodes plus number of arcs) plus the sizes of the labels in it. The main difference between ADD, SLDD, and AADD lies in the way the decision diagrams are labelled and interpreted.

For ADD, no specific assumption has to be made on the valuation structure $\mathcal{E}$, even if $E = \mathbb{R}$ is often considered:

Definition 4 (ADD) *ADD is the 4-tuple $\langle C_{\mathtt{ADD}}, Var_{\mathtt{ADD}}, I_{\mathtt{ADD}}, s_{\mathtt{ADD}}\rangle$ where $C_{\mathtt{ADD}}$ is the set of ordered VDDs α over X such that sinks S are labelled by elements of E, and the arcs are not labelled; $I_{\mathtt{ADD}}$ is defined inductively by: for every assignment $\vec{x}$ over X,*

- *if α is a sink node S, labelled by $\phi(S) = e$, then $I_{\mathtt{ADD}}(\alpha)(\vec{x}) = e$,*
- *else the root N of α is labelled by $x \in X$; let $d \in D_x$ such that $(x, d) \in \vec{x}$, $a = (N, M)$ the arc such that $v(a) = d$, and β the ADD formula rooted at node M in α; we have $I_{\mathtt{ADD}}(\alpha)(\vec{x}) = I_{\mathtt{ADD}}(\beta)(\vec{x})$.*

Optimization is easy on an ADD formula: every path from the root of α to a sink labelled by a non-dominated valuation among those labeling the sinks of α can be read as a (usually partial) variable assignment which can be extended to a (full) optimal assignment.

In the SLDD framework [Wilson, 2005], the valuation structure $\mathcal{E}$ must take the form of a commutative semiring $\langle E, \oplus, \otimes, 0_s, 1_s\rangle$: $\oplus$ and $\otimes$ are associative and commutative mappings from $E \times E$ to E, with identity elements (respectively) 0_s and 1_s, $\otimes$ left and right distributes over $\oplus$, and 0_s is an annihilator for $\otimes$ ($\forall a \in E$, $a \otimes 0_s = 0_s \otimes a = 0_s$).

Definition 5 (SLDD) *Let $\mathcal{E} = \langle E, \oplus, \otimes, 0_s, 1_s\rangle$ be a commutative semiring. SLDD is the 4-tuple $\langle C_{\mathtt{SLDD}}, Var_{\mathtt{SLDD}}, I_{\mathtt{SLDD}}, s_{\mathtt{SLDD}}\rangle$ where $C_{\mathtt{SLDD}}$ is the set of VDDs α over X with a unique sink S, satisfying $\phi(S) = 1_s$, and such that the arcs are labelled by elements of E, and $I_{\mathtt{SLDD}}$ is defined inductively by: for every assignment $\vec{x}$ over X,*

- *if α is the sink node S, then $I_{\mathtt{SLDD}}(\alpha)(\vec{x}) = 1_s$,*
- *else the root N of α is labelled by $x \in X$; let $d \in D_x$ such that $(x, d) \in \vec{x}$, $a = (N, M)$ the arc such that $v(a) = d$, and β the SLDD formula rooted at node M in α; we have $I_{\mathtt{SLDD}}(\alpha)(\vec{x}) = \phi(a) \otimes I_{\mathtt{SLDD}}(\beta)(\vec{x})$.*

SLDD languages are not specifically suited to optimization w.r.t. any relation $\succeq$. Specifically, [Wilson, 2005] considers the following *addition-is-max-or-min* assumption about $\oplus$:

$$\forall a, b \in E, a \oplus b \in \{a, b\}.$$

Under this assumption, $\oplus$ is idempotent and the relation $\trianglelefteq$ defined by $a \trianglelefteq b$ iff $a \oplus b = a$ is total. [Wilson, 2005] shows that, when $\succeq$ coincides with $\trianglelefteq$, computing the valuation of $I_{\mathtt{SLDD}}(\alpha)$ maximal w.r.t. $\succeq$ amounts to performing $\oplus$-variable elimination; this can be achieved in polynomial time under the linear-time computability assumption for $\otimes$ and $\oplus$.

Sanner and Mc Allester's AADD framework [2005] focuses on the valuation set $E = \mathbb{R}^+$ but enables decision graphs into which the arcs are labelled with *pairs* of values from $\mathbb{R}^+$ and considers *two* operators, namely $+$ and $\times$:

Definition 6 (AADD) *AADD is the 4-tuple* $\langle C_{\texttt{AADD}}, Var_{\texttt{AADD}}, I_{\texttt{AADD}}\ s_{\texttt{AADD}}\rangle$ *where* $C_{\texttt{AADD}}$ *is the set of ordered VDDs* α *over* X *with a unique sink* S*, satisfying* $\phi(S) = 1$*, and such that the arcs are labelled by pairs* $\langle q, f\rangle$ *in* $\mathbb{R}^+ \times \mathbb{R}^+$*;* $I_{\texttt{AADD}}$ *is defined inductively by: for every assignment* $\vec{x}$ *over* X*,*

- *if* α *is the sink node* S*, then* $I_{\texttt{AADD}}(\alpha)(\vec{x}) = 1$*,*
- *else the root* N *of* α *is labelled by* $x \in X$*; let* $d \in D_x$ *such that* $(x, d) \in \vec{x}$*,* $a = (N, M)$ *the arc such that* $v(a) = d$ *and* $\phi(a) = \langle q, f\rangle$*, and* β *the AADD formula rooted at node* M *in* α*; we have*

$$I_{\texttt{AADD}}(\alpha)(\vec{x}) = q + (f \times I_{\texttt{AADD}}(\beta)(\vec{x})).$$

For the normalization purpose, each α *is equipped with a pair* $\langle q_0, f_0\rangle$ *from* $\mathbb{R}^+ \times \mathbb{R}^+$ *(the "offset", labeling the root of* α*); the interpretation function of the resulting "augmented" AADD is given by, for every assignment* $\vec{x}$ *over* X*,* $I^{\langle q_0, f_0\rangle}_{\texttt{AADD}}(\alpha)(\vec{x}) = q_0 + (f_0 \times I_{\texttt{AADD}}(\alpha)(\vec{x}))$*.*

Conditioning and optimization are also tractable on AADD formulae (see [Sanner and McAllester, 2005]).

3 Revisiting the SLDD Framework

In the following, we extend the SLDD framework in two directions: we relax the algebraic requirements imposed on the valuation structure and we point out a normalization procedure which extends the AADD 's one to some representation languages of e-SLDD, the extended SLDD family.

A first useful observation is that, in the SLDD framework, $\oplus$ is not used for defining the SLDD *language*. Actually, different $\oplus$ may be considered over the same formula (e.g., when SLDD is used to compile a Bayesian net, $\oplus = +$ can be used for marginalization purposes and $\oplus = max$ can be considered when a most probable explanation is looked for). This explains why the requirements imposed on $\oplus$ in the SLDD setting can be relaxed. Let us recall that a *monoid* is a triple $\langle E, \otimes, 1_s\rangle$ where E is a set endowed with an associative binary operator $\otimes$ with identity element 1_s:

Definition 7 (e-SLDD) *For any monoid* $\mathcal{E} = \langle E, \otimes, 1_s\rangle$*, e-SLDD is the 4-tuple* $\langle C_{\texttt{e-SLDD}}, Var_{\texttt{e-SLDD}}, I_{\texttt{e-SLDD}_\otimes}, s_{\texttt{e-SLDD}}\rangle$*, defined as the SLDD one, except that, for the normalization purpose, each e-SLDD formula* α *is associated with a value* $q_0 \in E$ *(the "offset" of the data structure, labeling its root); the interpretation function* $I^{q_0}_{\texttt{e-SLDD}}$ *of the extended SLDD setting is given by, for every assignment* $\vec{x}$ *over* X*,*

$$I^{q_0}_{\texttt{e-SLDD}}(\alpha)(\vec{x}) = q_0 \otimes I_{\texttt{e-SLDD}}(\alpha)(\vec{x}).$$

Several choices for $\otimes$ remain usually possible when E is fixed; we sometimes make the notation of the language more precise (but not too heavy) and write $\texttt{e-SLDD}_\otimes$ instead of e-SLDD.

Obviously, the e-SLDD framework captures the SLDD one: when $\langle E, \oplus, \otimes, 0_s, 1_s\rangle$ is a commutative semiring, then $\langle E, \otimes, 1_s\rangle$ is a monoid, and every SLDD formula can be interpreted as an e-SLDD one (choose $q_0 = 1_s$). Interestingly, the e-SLDD framework also captures the AADD language:

Proposition 1 *Let* $E = \mathbb{R}^+ \times \mathbb{R}^+$*,* $1_s = \langle 0, 1\rangle$ *and* $\otimes = \star$ *be defined by* $\forall b, b', c, c' \in E$*,* $\langle b, c\rangle \star \langle b', c'\rangle = \langle b + c \times b', c \times c'\rangle$*.* $\mathcal{E} = \langle E, \otimes, 1_s\rangle$ *is a monoid.*

The correspondence between AADD and $\texttt{e-SLDD}_\star$ is made precise by the following proposition:

Proposition 2 *Let* α *be an AADD formula, also viewed as an* $\texttt{e-SLDD}_\star$ *formula. We have:* $\forall \vec{x} \in \vec{X}$*, if* $I_{\texttt{AADD}}(\alpha)(\vec{x}) = a$ *and* $I_{\texttt{e-SLDD}_\star}(\alpha)(\vec{x}) = \langle b, c\rangle$*, then* $a = b + c$*.*

Observe that $\star$ is *not* commutative: the relaxation of the commutativity assumption is necessary to capture the AADD framework within the e-SLDD family.

Let us now switch to the normalization/canonicity issues for e-SLDD. When compiling a formula, normalization (and all the more canonicity) are important for computational reasons: in practice, subformulae in reduced, normalized form which have been already encountered and cached can be more efficiently recognized. Besides, when the canonicity property is ensured, the recognition issue boils down to a simple equality test. Thus, canonicity is more demanding and is achieved for ordered VDDs, only: reduced ADD formulae and normalized and reduced AADD formulae (which are ordered VDDs) offer the canonicity property. Contrastingly, though some simplification rules have been considered in [Wilson, 2005], no normalization procedure and canonicity conditions for SLDD have been pointed out so far.

The idea at work for normalizing AADD formulae is to propagate from the sink to the root of the diagram the *minimum* valuations of the outgoing arcs. In our more general framework, minimality is characterized by an idempotent, commutative and associative operator $\oplus$, which induces the binary relation $\trianglerighteq$ over E given by:

$$\forall a, b \in E, a \trianglerighteq b \text{ iff } a \oplus b = b.$$

The fact that is $\oplus$ is associative (resp. commutative, idempotent) implies that the induced relation $\trianglerighteq$ is transitive (resp. antisymmetric, reflexive), hence an order over E.

Definition 8 ($\oplus$-normalisation, $\oplus$-reduction) *An e-SLDD formula* α *is* $\oplus$-normalized *iff for any node* N *of* α*,* $\oplus_{a \in out(N)}\phi(a) = 1_s$ *(by convention, we define* $\oplus_{a \in \emptyset}\phi(a) = 1_s$*). An e-SLDD formula* α *is* $\oplus$-reduced *iff it is* $\oplus$*-normalized, and* reduced*, i.e., it does not contain any (distinct) isomorphic nodes*[1] *and any redundant nodes.*[2]

To allow to propagate valuations in VCSPs, where $\trianglerighteq$ is a total order, $\otimes$ is commutative and $\otimes$ is monotonic w.r.t. $\trianglerighteq$ (i.e., $\otimes$ is distributive over $\oplus$), [Cooper and Schiex, 2004] assume a "fairness" property of $\otimes$ w.r.t. $\oplus$: for any valuations $a, b \in E$ such that $a \oplus b = b$, there exists a unique valuation which is the maximal element w.r.t. $\trianglerighteq$ among the $c \in E$ satisfying $b \otimes c = c \otimes b = a$.

Here, we relax these conditions so as to be able to encompass the case of the (possibly partial) relation $\trianglerighteq$ induced

[1] N and M are isomorphic when they are labelled by the same variable and there exists a bijection f from $out(N)$ to $out(M)$ such that $\forall a \in out(N)$, a and $f(a)$ have the same end node and $\phi(a) = \phi(f(a))$.

[2] N is redundant when all outgoing arcs a are labelled by the same value $\phi(a)$ and reach the same end node.

by $\oplus$. Let us state that $\otimes$ is *left-distributive* over $\oplus$ iff $\forall a, b, c \in E$, $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$, and $\otimes$ is *left-fair* w.r.t. $\oplus$ iff $\forall a, b \in E$, if $a \oplus b = b$, then there exists a unique valuation of E, noted $a \otimes^{-1} b$, which is the maximal element w.r.t. $\trianglerighteq$ among the $c \in E$ satisfying $b \otimes c = a$.

Definition 9 (extended SLDD condition) *A valuation structure $\mathcal{E} = \langle E, \oplus, \otimes, 1_s \rangle$ satisfies* the extended SLDD condition *iff $\langle E, \otimes, 1_s \rangle$ is a monoid, $\oplus$ is a mapping from $E \times E$ to E, which is associative, commutative, and idempotent, $\otimes$ is left-distributive over $\oplus$ and left-fair w.r.t. $\oplus$.*

The extended SLDD condition is close to the commutative semiring assumption for SLDD. However, it requires neither the commutativity of $\otimes$, nor an annihilator for $\otimes$, and left-distributivity of $\otimes$ over $\oplus$ is less demanding than (full) distributivity; on the other hand, the left-fairness condition of $\otimes$ w.r.t. $\oplus$ is imposed. The idempotence of $\oplus$ is also less demanding than the "addition-is-max-or-min" condition.

The valuation considered in the AADD framework satisfies the extended SLDD condition:

Proposition 3 *The valuation structure $\mathcal{E} = \langle \mathbb{R}^+ \times \mathbb{R}^+, \oplus, \star, \langle 0, 1 \rangle \rangle$ where $\oplus = min_\star$ is defined by $\forall b, b', c, c' \in E$: $\langle b, b' \rangle$ $min_\star$ $\langle c, c' \rangle = \langle min(b, c), max(b + b', c + c') - min(b, c) \rangle$, satisfies the extended SLDD condition.*

In the AADD case, $E = \mathbb{R}^+ \times \mathbb{R}^+$ is not totally ordered by $\trianglerighteq$ (for instance, none of $\langle 0, 2 \rangle \trianglerighteq \langle 1, 2 \rangle$ and $\langle 1, 2 \rangle \trianglerighteq \langle 0, 2 \rangle$ hold since $\langle 0, 2 \rangle min_\star \langle 1, 2 \rangle = \langle 1, 2 \rangle min_\star \langle 0, 2 \rangle = \langle 0, 3 \rangle$). When $\langle a, a' \rangle \trianglerighteq \langle b, b' \rangle$ holds, we have:

- $\langle a, a' \rangle \star^{-1} \langle b, b' \rangle = \langle 1, 0 \rangle$ if $b' = 0$,
- $\langle a, a' \rangle \star^{-1} \langle b, b' \rangle = \langle \frac{a-b}{b'}, \frac{a'}{b'} \rangle$ if $b' > 0$.

e−$\text{SLDD}_\star$ denotes the corresponding e−SLDD language.

Weighted finite automata and edge-valued binary decision diagrams are captured by using $\mathcal{E} = \langle \mathbb{R}^+, min, +, 0 \rangle$. The following pairs, consisting of a valuation structure – a representation language, can actually be considered:

- $\mathcal{E} = \langle \mathbb{R}^+, min, +, 0 \rangle$ – e−SLDD_+.
- $\mathcal{E} = \langle \mathbb{R}^+, max, \times, 1 \rangle$ – e−$\text{SLDD}_\times$.
- $\mathcal{E} = \langle \mathbb{R}^+ \cup \{+\infty\}, max, min, +\infty \rangle$ – e−SLDD_{min}.
- $\mathcal{E} = \langle \mathbb{R}^+, min, max, 0 \rangle$ – e−SLDD_{max}.

Proposition 4 *The valuation structures $\mathcal{E} = \langle \mathbb{R}^+, min, +, 0 \rangle$, $\mathcal{E} = \langle \mathbb{R}^+, max, \times, 1 \rangle$, $\mathcal{E} = \langle \mathbb{R}^+ \cup \{+\infty\}, max, min, +\infty \rangle$ and $\mathcal{E} = \langle \mathbb{R}^+, min, max, 0 \rangle$ satisfy the extended SLDD condition.*

We are now ready to extend the AADD normalization procedure to the e−SLDD language, under the extended SLDD condition. Algorithm 1 is the normalization procedure. This procedure proceeds backwards (i.e., from the sink to the root). Figure 1 gives an e−$\text{SLDD}_\star$ formula and the corresponding $min_\star$-reduced formula.

Proposition 5 *Assume that $\mathcal{E} = \langle E, \oplus, \otimes, 1_s \rangle$ satisfies the extended SLDD condition. If $\oplus$ satisfies the addition-is-max-or-min property then, for any e−SLDD formula α, a $\oplus$-reduced e−SLDD formula equivalent to α can be computed in polynomial time provided that $\otimes$, $\otimes^{-1}$ and $\oplus$ can be computed in linear time.*

Algorithm 1: normalize(α)

input : an e−$\text{SLDD}_\otimes$ formula α, with offset q_0
output: an e−$\text{SLDD}_\otimes$ formula which is $\oplus$-normalized and equivalent to α

for *each node N of α in inverse topological ordering* do
    $q_{min} := \oplus_{a \in out(N)} \phi(a)$
    for *each* $a \in out(N)$ do
        if $\phi(a) == q_{min}$ then
            $\phi(a) := 1_s$
        else
            $\phi(a) := \phi(a) \otimes^{-1} q_{min}$
    for *each* $a \in in(N)$ do
        $\phi(a) := \phi(a) \otimes q_{\min}$
$q_0 := q_0 \otimes q_{min}$
return α

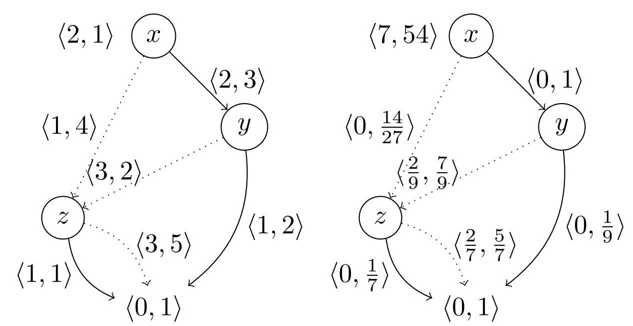

Figure 1: An e−$\text{SLDD}_\star$ formula (left) and the corresponding $min_\star$-reduced e−$\text{SLDD}_\star$ formula (right). x, y and z are Boolean variables. A (resp. plain) edge corresponds to the assignment of the variable labeling its source to 0 (resp. 1).

Clearly, the linear-time computability assumptions are satisfied by the operators $\otimes$, $\otimes^{-1}$, and $\oplus$ associated with e−SLDD_+, e−$\text{SLDD}_\times$, e−SLDD_{min}, e−SLDD_{max}. Thus, the formulae from all these languages can be $\oplus$-reduced in polynomial time.

Interestingly, addition-is-max-or-min is not a necessary condition for ensuring a normalized form; *left-cancellativity* of $\otimes$ ($\forall a, b, c \in E$, if $c \otimes a = c \otimes b$ and c is not an annihilator for $\otimes$, then $a = b$) is also enough:

Proposition 6 *Assume that $\mathcal{E} = \langle E, \oplus, \otimes, 1_s \rangle$ satisfies the extended SLDD condition. If $\otimes$ is left-cancellative, then for any e−SLDD formula α, a $\oplus$-reduced e−SLDD formula equivalent to α it can be computed in polynomial time provided that $\otimes$, $\otimes^{-1}$ and $\oplus$ can be computed in linear time.*

Furthermore, when $\otimes$ is left-cancellative, the canonicity property is ensured for ordered e−SLDD formulae (even if $\trianglerighteq$ is not total):

Proposition 7 *Assume that $\mathcal{E} = \langle E, \oplus, \otimes, 1_s \rangle$ satisfies the extended SLDD condition. If $\otimes$ is left-cancellative, then two ordered e−SLDD formulae are equivalent iff they have the same $\oplus$-reduced form.*

Especially, since $+$, $\times$ and $\star$ are left-cancellative, the ordered $\texttt{e-SLDD}_+$ (resp. $\texttt{e-SLDD}_\times$, $\texttt{e-SLDD}_\star$) formulae offer the canonicity property.

Let us finally switch to conditioning and optimization. First, conditioning does not preserve the $\oplus$-reduction of a formula in the general case, but this is computationally harmless since the $\oplus$-reduction of a conditioned formula can be done in polynomial time. As to optimization, when $\trianglerighteq$ is total, any $\oplus$-reduced $\texttt{e-SLDD}$ formula α contains a path the arcs of which are labelled by 1_s. The (usually partial) variable assignment along this path can be extended to a full minimal solution $\vec{x^*}$ w.r.t. $\trianglerighteq$, and the offset of α is equal to $I_{\texttt{e-SLDD}}(\alpha)(\vec{x^*})$. However, in the general case, the ordering $\succeq$ is not equal to $\trianglerighteq$, so the normalization procedure does not help for determining a minimal solution $\vec{x^*}$ w.r.t. $\succeq$ (or equivalently, a maximal solution w.r.t. the inverse ordering $\preceq$). Nevertheless, a simple left-monotonicity condition over the valuation structure is enough for ensuring that a minimal solution $\vec{x^*}$ w.r.t. $\succeq$ can be computed in time polynomial in the size of the $\texttt{e-SLDD}$ formula, using dynamic programming. The result of [Wilson, 2005] indeed can be extended as follows:

Proposition 8 *For any monoid $\mathcal{E} = \langle E, \otimes, 1_s\rangle$ such that E is totally pre-ordered by $\succeq$, if $\otimes$ is left-monotonic w.r.t. $\succeq$ (for any $a, b, c \in E$, if $a \succeq b$ then $c \otimes a \succeq c \otimes b$), then for any* $\texttt{e-SLDD}$ *formula α, a solution $\vec{x^*}$ minimal w.r.t. $\succeq$ can be computed in time polynomial in the size of α.*

4 Succinctness of VDDs: Theoretical Results

Let $\mathcal{L}_1$ (resp. $\mathcal{L}_2$) be a representation language over X w.r.t. $\mathcal{E}_1$ (resp. $\mathcal{E}_2$). The notion of succinctness and of translations usually considered over propositional languages (see [Darwiche and Marquis, 2002]) can be extended as follows:

Definition 10 (succinctness) *$\mathcal{L}_1$ is* at least as succinct as *$\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_s \mathcal{L}_2$, iff there exists a polynomial p such that for every $\alpha \in C_{\mathcal{L}_2}$, there exists $\beta \in C_{\mathcal{L}_1}$ which is equivalent to α and such that $s_{L_1}(\beta) \leq p(s_{L_2}(\alpha))$.*

Definition 11 (linear / polynomial translation) *$\mathcal{L}_2$ is* linearly (resp. polynomially) translatable into *$\mathcal{L}_1$, denoted $\mathcal{L}_1 \leq_l \mathcal{L}_2$ (resp. $\mathcal{L}_1 \leq_p \mathcal{L}_2$), iff there exists a linear-time (resp. polynomial-time) algorithm f from $C_{\mathcal{L}_2}$ to $C_{\mathcal{L}_1}$ such that for every $\alpha \in C_{\mathcal{L}_2}$, α is equivalent to $f(\alpha)$.*

$<_s$ (resp. $<_p$, $<_l$) denotes the asymmetric part of $\leq_s$ (resp. $\leq_p$, $\leq_l$), and $\sim_s$ (resp. $\sim_p$, $\sim_l$) denotes the symmetric part of $\leq_s$ (resp. $\leq_p$, $\leq_l$). By construction, $\sim_s$, $\sim_p$, $\sim_l$ are equivalence relations.

We have obtained the following result showing that every $\texttt{ADD}$ is linearly translatable into any $\texttt{e-SLDD}$ (sharing the same valuation set E):

Proposition 9 $\texttt{e-SLDD} \leq_l \texttt{ADD}$.

As to the valuation set $E = \mathbb{R}^+$, we get:

Proposition 10

- $\texttt{ADD} \sim_p \texttt{e-SLDD}_{max}$.
- $\texttt{e-SLDD}_\times \not\leq_s \texttt{e-SLDD}_+$ and $\texttt{e-SLDD}_+ \not\leq_s \texttt{e-SLDD}_\times$.
- $\texttt{AADD} <_s \texttt{e-SLDD}_+ <_s \texttt{ADD}$.
- $\texttt{AADD} <_s \texttt{e-SLDD}_\times <_s \texttt{ADD}$.

Similarly, for $E = \mathbb{R}^+ \cup \{+\infty\}$, $\texttt{ADD} \sim_p \texttt{e-SLDD}_{min}$ holds.

5 Succinctness of VDDs: Empirical Results

While succinctness is a way to compare representation languages w.r.t. the concept of spatial efficiency, it does not capture all aspects of this concept, for two reasons (at least). On the one hand, succinctness focuses on the worst case, only. On the other hand, it is of qualitative (ordinal) nature: succinctness indicates when an exponential separation can be achieved between two languages but does not enable to draw any quantitative conclusion on the sizes of the compiled forms. This is why it is also important to complete succinctness results with some size measurements.

To this aim, we made some experiments. We designed a bottom-up ordered $\texttt{e-SLDD}$ compiler. This compiler takes as input VCSP instances in the XML format described in [Roussel and Lecoutre, 2009] or Bayesian networks conforming to the XML format given in [Cozman, 2002]. When VCSP instances are considered, the compiler generates a data structure equivalent to each valued constraint of the instance, under the form of a reduced $\texttt{e-SLDD}_+$ formula, and incrementally combines them w.r.t. $+$ using a simplified version of the $apply(+)$ procedure described in [Sanner and McAllester, 2005]. Similarly, when Bayesian network instances are considered, the conditional probability tables are first compiled into reduced $\texttt{e-SLDD}_\times$ formulae, which are then combined using $\times$. At each combination step, the current $\texttt{e-SLDD}$ formula is reduced. We developed a toolbox which also contains procedures for transforming any $\texttt{e-SLDD}_+$ (resp. $\texttt{e-SLDD}_\times$) formula into an equivalent $\texttt{ADD}$ formula, and any $\texttt{ADD}$ formula into an equivalent $\texttt{e-SLDD}_+$ (resp. $\texttt{e-SLDD}_\times$, $\texttt{AADD}$) formula; the transformation procedure from $\texttt{e-SLDD}_+$ (resp. $\texttt{e-SLDD}_\times$) formulae to $\texttt{ADD}$ formulae roughly consists in pushing the labels from the root to the last arcs of the diagram. The transformation procedures from $\texttt{ADD}$ formulae to $\texttt{e-SLDD}_+$, $\texttt{e-SLDD}_\times$ and $\texttt{AADD}$ formulae are basically normalization procedures.

We considered two families of benchmarks. The VCSP instances we used concern car configurations problems;[3] these instances contain hard constraints and soft constraints, with valuations representing prices, to be aggregated additively. They have the following characteristic features:

- `Small`: #variables=139; max. domain size=16; #constraints=176 (including 29 soft constraints)
- `Medium`: #variables=148; max. domain size=20; #constraints=268 (including 94 soft constraints)
- `Big`: #variables=268; max. domain size=324; #constraints=2157 (including 1825 soft constraints)

We also compiled only the soft constraints of the benchmarks, leading to three other instances, referred to as $\{$`Small, Medium, Big`$\}$ `Price only`. As to Bayesian networks, which are of multiplicative nature (joint

[3]These instances have been built in collaboration with the french car manufacturer Renault; they are described in more depth in [Astesana *et al.*, 2013].

Table 1: Compilation of VCSPs into $\texttt{e-SLDD}_+$, and transformations into ADD, $\texttt{e-SLDD}_\times$ and AADD.

	$\texttt{e-SLDD}_+$		ADD	$\texttt{e-SLDD}_\times$	AADD
Instance	nodes (edges)	time (s)	nodes (edges)	nodes (edges)	nodes (edges)
Small Price only	36 (108)	< 1	4364 (7439)	3291 (7439)	36 (108)
Medium Price only	169 (499)	< 1	37807 (99280)	33595 (99280)	168 (495)
Big Price only	3317 (9687)	18	m-o	-	3317 (9687)
Small	2344 (5584)	1	299960 (637319)	14686 (33639)	2344 (5584)
Medium	6234 (17062)	6	752466 (2071474)	129803 (314648)	6234 (17062)
Big	198001 (925472)	79043	m-o	-	198001 (925472)

Table 2: Compilation of Bayesian networks into $\texttt{e-SLDD}_\times$, and transformations into ADD, $\texttt{e-SLDD}_+$ and AADD.

	$\texttt{e-SLDD}_\times$		ADD	$\texttt{e-SLDD}_+$	AADD
Instance	nodes (edges)	time (s)	nodes (edges)	nodes (edges)	nodes (edges)
Cancer	13 (25)	< 1	38 (45)	23 (45)	11 (21)
Asia	23 (45)	< 1	415 (431)	216 (431)	23 (45)
Car-starts	41 (83)	< 1	42741 (64029)	19632 (39265)	38 (77)
Alarm	1301 (3993)	< 1	m-o	-	1301 (3993)
Hailfinder25	32718 (108083)	8	m-o	-	32713 (108063)

probabilities are products of conditional probabilities), we used some standard benchmarks [Cozman, 2002].

Each configuration (resp. Bayesian net) instance has been compiled into an $\texttt{e-SLDD}_+$ formula (resp. an $\texttt{e-SLDD}_\times$ formula), and then transformed into an ADD formula, an $\texttt{e-SLDD}_\times$ formula (resp. an $\texttt{e-SLDD}_+$ formula), and an AADD formula – the time needed for the compilation and the sizes on the compiled formulae are reported in Table 1 (resp. Table 2). In order to determine a variable ordering, we used the *Maximum Cardinality Search* heuristic [Tarjan and Yannakakis, 1984] in reverse order, as proposed in [Amilhastre, 1999] for the compilation of (classical) CSPs. This heuristic is easy to compute and efficient; experiments reported in [Amilhastre, 1999] show that it typically outperforms several standard CSP variable ordering heuristics.

We ran all our experiments on a computer running at 800MHz with 256Mb of memory. "m-o" means that the available memory has been exhausted, and that the program aborted for this reason.

Our experiments confirm some of the theory-oriented succinctness results, especially the fact that the succinctness of $\texttt{e-SLDD}_+$ and of $\texttt{e-SLDD}_\times$ are incomparable but each of them is strictly more succinct than ADD. Unsurprisingly, when the values of the soft constraints are to be aggregated additively as this is the case for configuration instances (resp. multiplicatively, as this is the case for Bayesian nets), $\texttt{e-SLDD}_+$ (resp. $\texttt{e-SLDD}_\times$) performs better than $\texttt{e-SLDD}_\times$ (resp. $\texttt{e-SLDD}_+$). AADD does not prove to be better than $\texttt{e-SLDD}_+$ in the additive case, or better than $\texttt{e-SLDD}_\times$ in the multiplicative case.[4] Thus, targeting the AADD language does not lead to much better compiled formulae from the spatial efficiency point of view, when the mapping to be represented is additive or multiplicative in essence, but not both.

6 Conclusion

In this paper, we have extended the SLDD family to the e-SLDD family, thanks to a relaxation of some requirements on the valuation structure, which is harmless for the conditioning and optimization purposes. The e-SLDD family is general enough to capture AADD as a specific element. We have pointed out a normalization procedure and a canonicity condition for formulae from some e-SLDD languages, including $\texttt{e-SLDD}_+$ and $\texttt{e-SLDD}_\times$. We have also compared the spatial efficiency of some elements of the e-SLDD family, i.e., $\texttt{e-SLDD}_+$ and $\texttt{e-SLDD}_\times$, with ADD and AADD from both the theoretical side and the practical side. Though $\texttt{e-SLDD}_+$ (resp. $\texttt{e-SLDD}_\times$) is less succinct than AADD from a theoretical point of view, it proves space-efficient enough for enabling the compilation of cost-based configuration problems (resp. Bayesian networks).

Interestingly, one of the conditions pointed out in the e-SLDD setting for tractable normalization (and reduction) does not impose the valuation set E to be *totally* ordered. Clearly, this paves the way for the compilation of multi-criteria objective functions as e-SLDD representations. Investigating this issue is a major perspective for future works. Another important issue for further research is to draw the full knowledge compilation map for VDD languages, which will require to identify the tractable queries and transformations of interest, depending on the algebraic properties of the valuation structure.

[4] On the Bayesian net instances, the resulting ADD and AADD formulae are larger than the ones obtained by [Sanner and McAllester, 2005]. This is due to the way numeric labels are merged (remember that reals are approximated by finite-precision floating-point numbers on a computer). Indeed, in our implementation, e_1 and e_2 are considered identical whenever $e_1 - e_2 < 10^{-9}.e_1$ (where $e_1 \geq e_2$). Since $e_1, e_2 \leq 1$ (they represent probabilities) the standard merging condition $e_1 - e_2 < 10^{-9}$ considered in [Sanner and McAllester, 2005] is subsumed by ours. This explains the size discrepancy.

References

[Amilhastre *et al.*, 2002] Jérôme Amilhastre, Hélène Fargier, and Pierre Marquis. Consistency restoration and explanations in dynamic CSPs application to configuration. *Artif. Intell.*, 135(1-2):199–234, 2002.

[Amilhastre, 1999] Jérôme Amilhastre. *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes*. PhD thesis, Université de Montpellier II, 1999.

[Astesana *et al.*, 2013] Jean-Marc Astesana, Laurent Cosserat, and Hélène Fargier. Business recommendation for configurable products (BR4CP) project: the case study. `http://www.irit.fr/~Helene.Fargier/BR4CPBenchs.html`, March 2013.

[Bahar *et al.*, 1993] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. In *Proc. of ICCAD'93*, pages 188–191, 1993.

[Cooper and Schiex, 2004] Martin C. Cooper and Thomas Schiex. Arc consistency for soft constraints. *Artif. Intell.*, 154(1-2):199–227, 2004.

[Cozman, 2002] Fabio Gagliardi Cozman. JavaBayes Version 0.347, Bayesian Networks in Java, User Manual. Technical report, dec 2002. Benchmarks at `http://sites.poli.usp.br/pmr/ltd/Software/javabayes/Home/node3.html`.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)*, 17:229–264, 2002.

[Darwiche, 2011] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *Proc. of IJCAI'11*, pages 819–826, 2011.

[Gogic *et al.*, 1995] G. Gogic, H.A. Kautz, Ch.H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proc. of IJCAI'95*, pages 862–869, 1995.

[Lai and Sastry, 1992] Yung-Te Lai and Sarma Sastry. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proc. of DAC'92*, pages 608–613, 1992.

[Lai *et al.*, 1996] Yung-Te Lai, Massoud Pedram, and Sarma B. K. Vrudhula. Formal verification using edge-valued binary decision diagrams. *IEEE Trans. on Computers*, 45(2):247–255, 1996.

[Roussel and Lecoutre, 2009] Olivier Roussel and Christophe Lecoutre. XML Representation of Constraint Networks: Format XCSP 2.1. Technical report, CoRR abs/0902.2362, feb 2009.

[Sanner and McAllester, 2005] Scott Sanner and David A. McAllester. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proc. of IJCAI'05*, pages 1384–1390, 2005.

[Schiex *et al.*, 1995] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proc. of IJCAI'95*, pages 631–639, 1995.

[Tafertshofer and Pedram, 1997] Paul Tafertshofer and Massoud Pedram. Factored edge-valued binary decision diagrams. *Formal Methods in System Design*, 10(2/3):243–270, 1997.

[Tarjan and Yannakakis, 1984] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, July 1984.

[Wilson, 2005] Nic Wilson. Decision diagrams for the computation of semiring valuations. In *Proc. of IJCAI'05*, pages 331–336, 2005.

Appendix

Proof:[Proposition 1]

- $\star$ is a mapping from $\mathbb{R}^+ \times \mathbb{R}^+$ to $\mathbb{R}^+$ because $+$ and $\times$ are: when $b, b', c, c' \in \mathbb{R}^+$, both $(b + b' \times c)$ and $(b' \times c')$ belong to $\mathbb{R}^+$.
- Neutral element:
  $\langle b, b'\rangle \star \langle 0, 1\rangle = \langle b + b' \times 0, b' \times 1\rangle = \langle b, b'\rangle$
  $\langle 0, 1\rangle \star \langle b, b'\rangle = \langle 0 + 1 \times b, 1 \times b'\rangle = \langle b, b'\rangle$.
- Associativity:
  $(\langle a, a'\rangle \star \langle b, b'\rangle) \star \langle c, c'\rangle$
  $= \langle a + a' \times b, a' \times b'\rangle \star \langle c, c'\rangle$
  $= \langle (a + a' \times b) + (a' \times b') \times c, (a' \times b') \times c'\rangle$
  $= \langle a + a' \times b + a' \times b' \times c, a' \times b' \times c'\rangle$
  $\langle a, a'\rangle \star (\langle b, b'\rangle \star \langle c, c'\rangle)$
  $= \langle a, a'\rangle \star \langle b + b' \times c, b' \times c'\rangle$
  $= \langle a + a' \times (b + b' \times c), a' \times (b' \times c')\rangle$
  $= \langle a + a' \times b + a' \times b' \times c, a' \times b' \times c'\rangle$

■

Proof:[Proposition 2] The proof is by induction on the height $h(\alpha)$ of α, i.e., the length of a longest path from the root of α to its sink node.

- Base case: $h(\alpha) = 0$. In this case, α reduces to the sink node, so we have $\forall \vec{x} \in \vec{X}$, $I_{\texttt{AADD}}(\alpha)(\vec{x}) = 1$ and $I_{\texttt{e-SLDD}_\star}(\alpha)(\vec{x}) = \langle 0, 1\rangle$, and the equality trivially holds.
- Inductive step: $h(\alpha) > 0$. Suppose that the property is satisfied for every AADD formula of height $\geq k$ and consider an AADD formula α over X s.t. $h(\alpha) = k + 1$. Let $\vec{x} \in \vec{X}$. Suppose w.l.o.g. that the root N of α is labeled with $x \in X$; let $d_x \in D_x$ s.t. $(x, d_x) \in \vec{x}$, and let $a = (N, M)$ be the arc of α s.t. $v(a) = d_x$ and $\phi(a) = \langle q, f\rangle$; finally, let β be the AADD formula rooted at M in α.
  By definition, we have:
  $I_{\texttt{AADD}}(\alpha)(\vec{x}) = q + f \times I_{\texttt{AADD}}(\beta)(\vec{x})$ and
  $I_{\texttt{e-SLDD}_\star}(\alpha)(\vec{x}) = \langle q, f\rangle \star I_{\texttt{e-SLDD}_\star}(\beta)(\vec{x})$.
  By induction hypothesis, if $I_{\texttt{AADD}}(\beta)(\vec{x}) = a$ and $I_{\texttt{e-SLDD}_\star}(\beta)(\vec{x}) = \langle b, c\rangle$, then $a = b + c$.

As a consequence, $I_{\texttt{AADD}}(\alpha)(\vec{x}) = q+f\times a = q+(f\times b)+(f\times c)$, and $I_{\texttt{e-SLDD}_\star}(\alpha)(\vec{x}) = \langle q, f\rangle \star \langle b, c\rangle = \langle q+f\times b, f\times c\rangle$, showing that the equality is satisfied.

■

Proof:[Proposition 3]

- $\langle E = \mathbb{R}^+ \times \mathbb{R}^+, \star, \langle 0, 1\rangle\rangle$ is a monoid (Proposition 1);
- $\oplus = min_\star$ is a mapping from $E \times E$ to E because min and $+$ are mappings from $\mathbb{R}^+ \times \mathbb{R}^+$ to $\mathbb{R}^+$;
- Associativity of $min_\star$:
$\langle a, a'\rangle min_\star(\langle b, b'\rangle min_\star\langle c, c'\rangle)$
$= \langle a, a'\rangle min_\star\langle min(b, c), max(b+b', c+c') - min(b, c)\rangle$
$= \langle min(a, min(b, c)), max(a+a', max(b+b', c+c') - min(a, min(b, c))\rangle$
$= \langle min(a, b, c), max(a+a', b+b', c+c') - min(a, b, c)\rangle$
$(\langle a, a'\rangle min_\star\langle b, b'\rangle) min_\star\langle c, c'\rangle)$
$= \langle min(a, b), max(a+a', b+b') - min(a, b)\rangle min_\star\langle c, c'\rangle)$
$= \langle min(min(a, b), c), max(max(a+a', b+b'), c+c') - min(min(a, b), c)\rangle$
$= \langle min(a, b, c), max(a+a', b+b', c+c') - min(a, b, c)\rangle$
- Commutativity of $min_\star$:
$\langle a, a'\rangle\ min_\star\langle b, b'\rangle$
$= \langle min(a, b), max(a+a', b+b') - min(a, b)\rangle$
$= \langle min(b, a), max(b+b', a+a') - min(b, a)\rangle$
$= \langle b, b'\rangle min_\star\langle a, a'\rangle$
- Idempotence of $min_\star$:
$\langle a, a'\rangle\ min_\star\langle a, a'\rangle$
$= \langle min(a, a), max(a+a', a+a') - min(a, a)\rangle$
$= \langle a, a+a'-a\rangle = \langle a, a'\rangle$
- Left-distributivity of $\star$ over $min_\star$:
$(\langle a, a'\rangle \star \langle b, b'\rangle) min_\star(\langle a, a'\rangle \star \langle c, c'\rangle)$
$= \langle a+a'\times b, a'\times b'\rangle min_\star\langle a+a'\times c, a'\times c'\rangle$
$= \langle min(a+a'\times b, a+a'\times c), max(a+a'\times b+a'\times b', a+a'\times c+a'\times c') - min(a+a'\times b, a+a'\times c)\rangle$
$= \langle a+a'\times min(b, c), a+a'\times max(b+b', c+c') - (a+a'\times min(b, c))\rangle$
$= \langle a+a'\times min(b, c), a'\times max(b+b', c+c') - a'\times min(b, c))\rangle$
$= \langle a+a'\times min(b, c), a'\times (max(b+b', c+c') - min(b, c))\rangle$
$= \langle a, a'\rangle \star \langle min(b, c), max(b+b', c+c') - min(b, c)\rangle$
$= \langle a, a'\rangle \star (\langle b, b'\rangle min_\star\langle c, c'\rangle)$
- Left-fairness of $\star$ w.r.t. $min_\star$: suppose that $\langle a, a'\rangle \neq \langle b, b'\rangle$ and $\langle a, a'\rangle \trianglerighteq \langle b, b'\rangle$ i.e., $\langle a, a'\rangle min_\star\langle b, b'\rangle = \langle b, b'\rangle$ or equivalently $min(a, b) = b$ and $max(a+a', b+b') - min(a, b) = b'$. Let:
$\langle a, a'\rangle \star^{-1} \langle b, b'\rangle = \langle 1, 0\rangle$ if $b' = 0$.
$\langle a, a'\rangle \star^{-1} \langle b, b'\rangle = \langle \frac{a-b}{b'}, \frac{a'}{b'}\rangle$ if $b' > 0$
  - If $b' = 0$, then from $\langle a, a'\rangle \trianglerighteq \langle b, b'\rangle$, we deduce that $min(a, b) = b$ and $max(a+a', b) - min(a, b) = 0$, which shows that $max(a+a', b) = b$. Thus we have both $a \geq b$ and $b \geq a+a'$ with $a' \geq 0$, which implies that $a' = 0$ and $a = b$. So $\langle b, b'\rangle \star \langle 1, 0\rangle = \langle b+b'\times 1, 0\rangle = \langle b+b', 0\rangle = \langle b, 0\rangle = \langle a, a'\rangle$.
  Suppose that there exists $\langle c, c'\rangle \neq \langle 1, 0\rangle$ such that $\langle b, b'\rangle \star \langle c, c'\rangle = \langle a, a'\rangle$. Then it is enough to show that it cannot be the case that $\langle c, c'\rangle \trianglerighteq \langle 1, 0\rangle$ unless $\langle c, c'\rangle = \langle 1, 0\rangle$. Towards a contradiction, assume that $\langle c, c'\rangle min_\star\langle 1, 0\rangle = \langle 1, 0\rangle$. Then we must have $\langle min(c, 1), max(c+c', 1+0) - min(c, 1)\rangle = \langle 1, 0\rangle$, which implies that $min(c, 1) = 1$ and $max(c+c', 1) = min(c, 1)$, hence $c = 1$ and $c' = 0$.
  - If $b' > 0$, then we must prove that $\langle \frac{a-b}{b'}, \frac{a'}{b'}\rangle$ is such that $\langle b, b'\rangle \star \langle \frac{a-b}{b'}, \frac{a'}{b'}\rangle = \langle a, a'\rangle$, which is easy since $\langle b, b'\rangle \star \langle \frac{a-b}{b'}, \frac{a'}{b'}\rangle = \langle b+b'\times \frac{a-b}{b'}, b'\times \frac{a'}{b'}\rangle = \langle a, a'\rangle$. Then it is enough to show that if $\langle c, c'\rangle$ is such that $\langle b, b'\rangle \star \langle c, c'\rangle = \langle a, a'\rangle$, then $\langle c, c'\rangle = \langle \frac{a-b}{b'}, \frac{a'}{b'}\rangle$. The point is that if $\langle b, b'\rangle \star \langle c, c'\rangle = \langle a, a'\rangle$, then we have $b+b'\times c = a$ and $b'\times c' = a'$. Accordingly, we have $c = \frac{a-b}{b'}$ and $c' = \frac{a'}{b'}$, which concludes the proof.

$\star$ is thus left-fair with respect to $min_\star$.

■

Proof:[Proposition 4]

- $\mathcal{E} = \langle \mathbb{R}^+, min, +, 0\rangle$ because $\mathcal{E} = \langle \mathbb{R}^+, +, 0\rangle$ is a monoid, min is a mapping from $\mathbb{R}^+ \times \mathbb{R}^+$ to $\mathbb{R}^+$ that is associative, commutative, and idempotent, $+$ is left-distributive over min ($a+min(b, c) = min(a+b, a+c)$) and $+$ is left-fair with respect to min: if $a \geq b$, then there exists a unique $c = a-b \in \mathbb{R}^+$ such that $b+c = a$.
- $\mathcal{E} = \langle \mathbb{R}^+, max, \times, 1\rangle$ because $\mathcal{E} = \langle \mathbb{R}^+, \times, 1\rangle$ is a monoid, max is a mapping from $\mathbb{R}^+ \times \mathbb{R}^+$ to $\mathbb{R}^+$ that is associative, commutative, and idempotent, $\times$ is left-distributive over max ($a\times max(b, c) = max(a\times b, a\times c)$) and $\times$ is left-fair with respect to max; indeed, let $a, b \in \mathbb{R}^+$ such that $a \neq b$ and $max(a, b) = b$, i.e., $b \geq a$. We must have $b \neq 0$ since otherwise we would also have $a = 0$, contradicting $a \neq b$. So there exists a unique $c = \frac{a}{b}$ such that $b\times c = a$.
- $\mathcal{E} = \langle \mathbb{R}^+ \cup \{+\infty\}, max, min, +\infty\rangle$ because $\mathcal{E} = \langle \mathbb{R}^+ \cup +\infty, min, +\infty\rangle$ is a monoid max is a mapping from $(\mathbb{R}^+ \cup \{+\infty\}) \times (\mathbb{R}^+ \cup \{+\infty\})$ to $\mathbb{R}^+ \cup \{+\infty\}$ that is associative, commutative, and idempotent, min is left-distributive over max and min is left-fair with respect to max with $min^{-1} = min$.
- $\mathcal{E} = \langle \mathbb{R}^+, min, max, 0\rangle$ because $\mathcal{E} = \langle \mathbb{R}^+, max, 0\rangle$ is a monoid, min is a mapping from $\mathbb{R}^+ \times \mathbb{R}^+$ to $\mathbb{R}^+$ that is

associative, commutative, and idempotent, max is left-distributive over min and max is left-fair with respect to min, with $max^{-1} = max$.

■

Proof:[Proposition 5] Let α be an $\texttt{e-SLDD}$ formula over $X = \{x_1, \dots, x_n\}$. We are going to prove that $\mathsf{normalize}(\alpha)$ is an $\oplus$-normalized $\texttt{e-SLDD}$ formula equivalent to α.

The proof is by induction on the height $h(\alpha)$ of α.

- Base case: $h(\alpha) = 0$. In this case α is equal to the sink node 1_s labelled with a given offset. Obviously, we have $\mathsf{normalize}(\alpha) = \alpha$, which is already $\oplus$-normalized (and represent the constant function equal to its offset).
- Inductive step: $h(\alpha) > 0$. Let x_1 be the variable labeling the root N_0 of α. Let $D_{x_1} = \{d_1, \dots, d_m\}$. By induction hypothesis, the property holds for every $\texttt{e-SLDD}$ formula α_{d_j} ($j \in 1, \dots, m$), which is the $\texttt{e-SLDD}$ formula rooted at M_j, where M_j is the child of N_0 such that $v((N_0, M_j)) = d_j$. Let us denote by $\overline{q_0}$ the offset of α and for each $j \in 1, \dots, m$, let $\overline{\phi(a_j}$ be the label of the arc $a_j = (N_0, M_j)$.

  We are going to prove first that $\mathsf{normalize}(\alpha)$ is equivalent to α. By induction hypothesis, for each $j \in 1, \dots, m$, $\mathsf{normalize}(\alpha_{d_j})$ is equivalent to α_{d_j}.

  At the last iteration step of the normalization procedure (i.e., when every internal node of the formula has been considered except its root), α is as depicted at Figure 2.

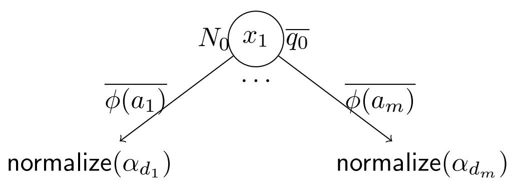

Figure 2: Normalization of an $\texttt{e-SLDD}$ α.

By definition of the semantics of an $\texttt{e-SLDD}$ formula, for every $\vec{x} \in \vec{X}$ such that $\vec{x} = \vec{x'} \cup \{(x_1, d_j)\}$, we have that $I_{\texttt{e-SLDD}}(\alpha)(\vec{x}) = \overline{q_0} \otimes (\overline{\phi(a_j)} \otimes I_{\texttt{e-SLDD}}(\alpha_{d_j})(\vec{x'}))$. By induction hypothesis, this is also equal to $\overline{q_0} \otimes (\overline{\phi(a_j)} \otimes I_{\texttt{e-SLDD}}(\mathsf{normalize}(\alpha_{d_j}))(\vec{x'}))$.

Let $q_{min} = \bigoplus_{j=1}^m \overline{\phi(a_j)}$. The last iteration step of the normalization procedure replaces in α each $\overline{\phi(a_j)}$ ($j \in 1, \dots, m$) by $\phi(a_j) := \overline{\phi(a_j)} \otimes^{-1} q_{min}$ when $\overline{\phi(a_j)} \neq q_{min}$ and by $\phi(a_j) := 1_s$ in the remaining case; finally, $\overline{q_0}$ is replaced by $q_0 := \overline{q_0} \otimes \bigoplus_{j=1}^m \overline{\phi(a_j)}$.

Since we have

$$I_{\texttt{e-SLDD}}(\mathsf{normalize}(\alpha))(\vec{x}) =$$

$$q_0 \otimes (\phi(a_j) \otimes I_{\texttt{e-SLDD}}(\mathsf{normalize}(\alpha_{d_j}))(\vec{x'})),$$

and since $\otimes$ is associative, it is enough to show that for any $j \in 1, \dots, m$, $\overline{q_0} \otimes \overline{\phi(a_j)} = q_0 \otimes \phi(a_j)$ to conclude that $\mathsf{normalize}(\alpha)$ is equivalent to α.

Two cases must be considered:

- $\overline{\phi(a_j)} = q_{min}$. We have $q_0 \otimes \phi(a_j) = (\overline{q_0} \otimes q_{min}) \otimes 1_s = \overline{q_0} \otimes q_{min} = \overline{q_0} \otimes \overline{\phi(a_j)}$.
- $\overline{\phi(a_j)} \neq q_{min}$. We have $q_0 \otimes \phi(a_j) = (\overline{q_0} \otimes \bigoplus_{j=1}^m \overline{\phi(a_j)}) \otimes (\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^m \overline{\phi(a_j)})$. $\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^m \overline{\phi(a_j)}$ is well-defined since $\overline{\phi(a_j)} \trianglerighteq \bigoplus_{j=1}^m \overline{\phi(a_j)}$ holds; indeed, since $\oplus$ is idempotent, we have that $\overline{\phi(a_j)} \oplus \bigoplus_{j=1}^m \overline{\phi(a_j)} = \bigoplus_{j=1}^m \overline{\phi(a_j)}$. Finally, since $\otimes$ is associative and since by definition of $\otimes^{-1}$, we have

  $$\bigoplus_{j=1}^m \overline{\phi(a_j)} \otimes (\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^m \overline{\phi(a_j)}) = \overline{\phi(a_j)},$$

  we also obtain that

  $$q_0 \otimes \phi(a_j) = \overline{q_0} \otimes \overline{\phi(a_j)},$$

  as expected.

We now prove that $\mathsf{normalize}(\alpha)$ is $\oplus$-normalized. By induction hypothesis it is enough to show that $\bigoplus_{j=1}^m \phi(a_j) = 1_s$. To get it, we first demonstrate a couple of intermediate results:

- we prove that $\otimes^{-1}$ is right-monotonic w.r.t. $\trianglerighteq$: $\forall a, b, c \in E$, if $a \trianglerighteq b$, $a \trianglerighteq c$ and $b \trianglerighteq c$, then $a \otimes^{-1} c \trianglerighteq b \otimes^{-1} c$. Towards a contradiction, suppose that $a \trianglerighteq b$, $a \trianglerighteq c$ and $b \trianglerighteq c$ and $a \otimes^{-1} c \not\trianglerighteq b \otimes^{-1} c$. When $\oplus$ satisfies the addition-is-max-or-min condition, $\trianglerighteq$ is total. Hence we have $b \otimes^{-1} c \trianglerighteq a \otimes^{-1} c$. Since $\otimes$ is left-distributive over $\oplus$, it is also left-monotonic w.r.t. $\trianglerighteq$: if $b \trianglerighteq c$ then $a \otimes b \trianglerighteq a \otimes c$; indeed, $b \trianglerighteq c$ holds iff $b \oplus c = c$, hence by left-distributivity of $\otimes$ over $\oplus$, we have $(a \otimes b) \oplus (a \otimes c) = a \otimes (b \oplus c) = a \otimes c$, hence $a \otimes b \trianglerighteq a \otimes c$ holds. Now, from $b \otimes^{-1} c \trianglerighteq a \otimes^{-1} c$, taking advantage of the left-monotony of $\otimes$ w.r.t. $\trianglerighteq$, we get that $c \otimes (b \otimes^{-1} c) \trianglerighteq c \otimes (a \otimes^{-1} c)$, which is equivalent to $b \trianglerighteq a$ by definition of $\otimes^{-1}$. Since we also have $a \trianglerighteq b$, by antisymmetry of $\trianglerighteq$, we get that $a = b$. Since $\trianglerighteq$ is reflexive, we derive that $a \otimes^{-1} c \trianglerighteq b \otimes^{-1} c$, contradiction.
- we prove that $\forall a \in E$, $a \otimes^{-1} a \trianglerighteq 1_s$. Since $\trianglerighteq$ is reflexive, we have $a \trianglerighteq a$. Now, by definition of $\otimes^{-1}$, $a \otimes^{-1} b$ is the maximal element w.r.t. $\trianglerighteq$ among the $c \in E$ satisfying $b \otimes c = a$. Hence, $a \otimes^{-1} a$ is the maximal element w.r.t. $\trianglerighteq$ among the $c \in E$ satisfying $a \otimes c = a$. Since $c = 1_s$ satisfies $a \otimes c = a$, we get that $a \otimes^{-1} a \trianglerighteq 1_s$.
- we prove that $\forall a, b, c \in E$, if $a \trianglerighteq c$ and $b \trianglerighteq c$ then $a \oplus b \trianglerighteq c$. Indeed, $a \trianglerighteq c$ holds iff $a \oplus c = c$ and $b \trianglerighteq c$ holds iff $b \oplus c = c$. So $(a \oplus b) \oplus c = a \oplus (b \oplus c) = a \oplus c = c$, which shows that $a \oplus b \trianglerighteq c$.

– we prove that $\forall a, b \in E$, we have $a \trianglerighteq a \oplus b$ and if $a \oplus b \trianglerighteq a$, then $a \oplus b = a$. This comes immediately from the fact that $\oplus$ is associative, commutative and idempotent.

Let $a_k \in out(N_0)$ and let $q_{min} = \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$. Since $\forall a, b \in E$, we have $a \trianglerighteq a \oplus b$ and since $\oplus$ is associative and commutative, we have $\overline{\phi(a_k)} \trianglerighteq q_{min}$. Because $\otimes^{-1}$ is right-monotonic w.r.t. $\trianglerighteq$, we derive that $\overline{\phi(a_k)} \otimes^{-1} q_{min} \trianglerighteq q_{min} \otimes^{-1} q_{min}$. Now, since $\forall a \in E$, $a \otimes^{-1} a \trianglerighteq 1_s$ and $\trianglerighteq$ is transitive, we get that $\overline{\phi(a_k)} \otimes^{-1} q_{min} \trianglerighteq 1_s$, hence $\phi(a_k) \trianglerighteq 1_s$. When $\overline{\phi(a_k)} = q_{min}$, we have $\phi(a_k) = 1_s$, hence $\phi(a_k) \trianglerighteq 1_s$ since $\trianglerighteq$ is reflexive. Since $\forall a, b, c \in E$, if $a \trianglerighteq c$ and $b \trianglerighteq c$ then $a \oplus b \trianglerighteq c$ and since $\oplus$ is associative, the fact that $\phi(a_k) \trianglerighteq 1_s$ holds for each $a_k \in out(N_0)$ implies that $\bigoplus_{j=1}^{m} \phi(a_j) \trianglerighteq 1_s$.

Finally, since $\oplus$ satisfies the addition-is-max-or-min condition and $\oplus$ is associative, there exists $i \in 1, \ldots, m$ such that $q_{min} = \overline{\phi(a_i)}$. As a consequence $\phi(a_i) = 1_s$. Accordingly, since $\oplus$ is associative and commutative, $\bigoplus_{j=1}^{m} \phi(a_j) = 1_s \oplus \bigoplus_{j=1,\ldots,m \mid j \neq i} \phi(a_j)$. Since $\bigoplus_{j=1}^{m} \phi(a_j) \trianglerighteq 1_s$ holds and since if $a \oplus b \trianglerighteq a$, then $a \oplus b = a$, we derive that $1_s \oplus \bigoplus_{j=1,\ldots,m \mid j \neq i} \phi(a_j) = 1_s$, or equivalently $\bigoplus_{j=1}^{m} \phi(a_j) = 1_s$.

That normalize runs in polynomial time provided that $\otimes$, $\otimes^{-1}$ and $\oplus$ can be computed in linear time is obvious (actually, it is enough to require that $\otimes$ can be computed in linear time and that $\otimes^{-1}$ and $\oplus$ can be computed in polynomial time). Finally, it is also obvious that the reduction (elimination of isomorphic nodes and of redundant nodes) of an e−SLDD formula preserves it semantics and can also be achieved in polynomial time.

■

Proof:[Proposition 6] Let α be an e−SLDD formula over $X = \{x_1, \ldots, x_n\}$. We are going to prove that normalize(α) is an $\oplus$-normalized e−SLDD formula equivalent to α.

The proof is again by induction on the height $h(\alpha)$ of α.

- Base case: $h(\alpha) = 0$. In this case α is equal to the sink node 1_s labelled with a given offset. Obviously, we have normalize(α) $= \alpha$, which is already $\oplus$-normalized (and represent the constant function equal to its offset).
- Inductive step: $h(\alpha) > 0$. We use the same notations as in the proof of Proposition 5. Let x_1 be the variable labeling the root N_0 of α. Let $D_{x_1} = \{d_1, \ldots, d_m\}$. By induction hypothesis, the property holds for every e−SLDD formula α_{d_j} ($j \in 1, \ldots, m$), which is the e−SLDD formula rooted at M_j, where M_j is the child of N_0 such that $v((N_0, M_j)) = d_j$. Let us denote by $\overline{q_0}$ the offset of α and for each $j \in 1, \ldots, m$, let $\overline{\phi(a_j}$ be the label of the arc $a_j = (N_0, M_j)$.

  We first prove that normalize(α) is equivalent to α. By induction hypothesis, for each $j \in 1, \ldots, m$, normalize(α_{d_j}) is equivalent to α_{d_j}.

  By definition of the semantics of an e−SLDD formula, for every $\vec{x} \in \vec{X}$ such that $\vec{x} = \vec{x'} \cup \{(x_1, d_j)\}$, we have that $I_{\texttt{e-SLDD}}(\alpha)(\vec{x}) = \overline{q_0} \otimes (\overline{\phi(a_j)} \otimes I_{\texttt{e-SLDD}}(\alpha_{d_j})(\vec{x'}))$. By induction hypothesis, this is also equal to $\overline{q_0} \otimes (\overline{\phi(a_j)} \otimes I_{\texttt{e-SLDD}}(\text{normalize}(\alpha_{d_j}))(\vec{x'}))$.

  Let $a_k \in out(N_0)$ and let $q_{min} = \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$. When $\overline{\phi(a_k)} = q_{min}$, we have $\overline{\phi(a_k)} \otimes^{-1} q_{min} = q_{min} \otimes^{-1} q_{min}$. By definition of $\otimes^{-1}$, we also have $q_{min} \otimes (q_{min} \otimes^{-1} q_{min}) = q_{min}$. Since 1_s is neutral for $\otimes$, q_{min} is also equal to $q_{min} \otimes 1_s$. Hence we have $q_{min} \otimes (q_{min} \otimes^{-1} q_{min}) = q_{min} \otimes 1_s$. When $\otimes$ is left-cancellative, this implies that $q_{min} \otimes^{-1} q_{min} = 1_s$.

  Thus, when $\otimes$ is left-cancellative, the last iteration step of the normalization procedure replaces in α each $\overline{\phi(a_j)}$ ($j \in 1, \ldots, m$) by $\phi(a_j) := \overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$ and finally $\overline{q_0}$ by $q_0 := \overline{q_0} \otimes \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$. Since we have

  $$I_{\texttt{e-SLDD}}(\text{normalize}(\alpha))(\vec{x}) =$$
  $$q_0 \otimes (\phi(a_j) \otimes I_{\texttt{e-SLDD}}(\text{normalize}(\alpha_{d_j}))(\vec{x'})),$$

  and since $\otimes$ is associative, it is enough to show that for any $j \in 1, \ldots, m$, $\overline{q_0} \otimes \overline{\phi(a_j)} = q_0 \otimes \phi(a_j)$ to conclude that normalize(α) is equivalent to α.

  By definition we have $q_0 \otimes \phi(a_j) = (\overline{q_0} \otimes \bigoplus_{j=1}^{m} \overline{\phi(a_j)}) \otimes (\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^{m} \overline{\phi(a_j)})$. $\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$ is well-defined since $\overline{\phi(a_j)} \trianglerighteq \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$ holds; indeed, since $\oplus$ is idempotent, we have that $\overline{\phi(a_j)} \oplus \bigoplus_{j=1}^{m} \overline{\phi(a_j)} = \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$. Finally, since $\otimes$ is associative and since by definition of $\otimes^{-1}$, we have

  $$\bigoplus_{j=1}^{m} \overline{\phi(a_j)} \otimes (\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^{m} \overline{\phi(a_j)}) = \overline{\phi(a_j)},$$

  we finally obtain that

  $$q_0 \otimes \phi(a_j) = \overline{q_0} \otimes \overline{\phi(a_j)},$$

  as expected.

  We now prove that normalize(α) is $\oplus$-normalized. By induction hypothesis it is enough to show that $\bigoplus_{j=1}^{m} \phi(a_j) = 1_s$.

  – we first prove that $\otimes^{-1}$ is right-distributive over $\oplus$: $\forall a, b, c \in E$, $(a \oplus b) \otimes^{-1} c = (a \otimes^{-1} c) \oplus (b \otimes^{-1} c)$. Observe that $a \oplus b \trianglerighteq c$ precisely when $a \trianglerighteq c$ and $b \trianglerighteq c$. On the one hand, by definition of $\trianglerighteq$, $a \trianglerighteq c$ holds iff $a \oplus c = c$ and $b \trianglerighteq c$ holds iff $b \oplus c = c$. Furthermore, if $a \oplus c = c$, then $(a \oplus c) \oplus b = c \oplus b$. Since $\oplus$ is associative and commutative, we have $(a \oplus c) \oplus b = (a \oplus b) \oplus c$ and $c \oplus b = b \oplus c$, so we get $(a \oplus b) \oplus c = c$, showing that $a \oplus b \trianglerighteq c$. Conversely, if $a \oplus b \trianglerighteq c$ holds, then $(a \oplus b) \oplus c = c$. Hence $a \oplus c = a \oplus ((a \oplus b) \oplus c) = a \oplus a \oplus b \oplus c$ since $\oplus$ is associative, $= a \oplus b \oplus c$ (since $\oplus$ is idempotent), $= c$. This shows that $a \trianglerighteq c$ (showing that $b \trianglerighteq c$ is similar, replacing a by b and b by a). Now, by definition of $\otimes^{-1}$, when $a \otimes^{-1} c$ and $b \otimes^{-1} c$ are well-defined, we have $c \otimes (a \otimes^{-1} c) = a$ and $c \otimes (b \otimes^{-1} c) = b$. Hence $a \oplus b = (c \otimes (a \otimes^{-1} c)) \oplus (c \otimes (b \otimes^{-1} c))$. Since $\otimes$ is left-distributive over $\oplus$, we also have $a \oplus b = c \otimes ((a \otimes^{-1} c) \oplus (b \otimes^{-1} c))$. Besides, by definition of $\otimes^{-1}$, when $(a \oplus b) \otimes^{-1} c$ is well-defined, we have $c \otimes ((a \oplus b) \otimes^{-1} c) = a \oplus b$. Thus we get $c \otimes ((a \otimes^{-1} c) \oplus (b \otimes^{-1} c)) = c \otimes ((a \oplus b) \otimes^{-1} c)$. Since $\otimes$ is left-cancellative, we get that $(a \otimes^{-1} c) \oplus (b \otimes^{-1} c) = ((a \oplus b) \otimes^{-1} c)$.

- we also prove that $\forall a \in E$, $a \otimes^{-1} a = 1_s$. Since $\oplus$ is idempotent, we have $a \oplus a = a$, hence $\trianglerighteq$ is reflexive: $a \trianglerighteq a$. Then, by definition of $\otimes^{-1}$, we have $a \otimes (a \otimes^{-1} a) = a$. Since 1_s is neutral for $\otimes$, we also have $a = a \otimes 1_s$. Hence, $a \otimes (a \otimes^{-1} a) = a \otimes 1_s$. Since $\otimes$ is left-cancellative, we get that $a \otimes^{-1} a = 1_s$.

On this ground, the result follows easily: $\bigoplus_{j=1}^{m} \phi(a_j) = \bigoplus_{j=1}^{m} (\overline{\phi(a_j)} \otimes^{-1} \bigoplus_{i=1}^{m} \overline{\phi(a_i)}) = (\bigoplus_{j=1}^{m} \overline{\phi(a_j)}) \otimes^{-1} (\bigoplus_{i=1}^{m} \overline{\phi(a_i)})$ since $\oplus$ is associative and $\otimes^{-1}$ is right-distributive over $\oplus$. Since $\bigoplus_{j=1}^{m} \overline{\phi(a_j)} = \bigoplus_{i=1}^{m} \overline{\phi(a_i)}$, we get that $\bigoplus_{j=1}^{m} \phi(a_j) = 1_s$.

That normalize runs in polynomial time provided that $\otimes$, $\otimes^{-1}$ and $\oplus$ can be computed in linear time is obvious (actually, it is enough to require that $\otimes$ can be computed in linear time and that $\otimes^{-1}$ and $\oplus$ can be computed in polynomial time). Finally, it is also obvious that the reduction (elimination of isomorphic nodes and of redundant nodes) of an e−SLDD formula preserves it semantics and can also be achieved in polynomial time.

■

Proof:[Proposition 7] For every ordered e−SLDD formula α, let us note $\oplus - \mathsf{reduce}(\alpha)$ the e−SLDD formula obtained by computing the $\oplus$-reduction of α in a bottom-up way: after the $\oplus$-normalization step of node N as achieved by the normalize procedure, one achieves a reduction step which consists in removing N if it is isomorphic to a node previously generated or if it is redundant. Note that the suppression step does not question the fact that the formula is $\oplus$-normalized since it does not modify the arc labels. Thus, the resulting formula is equal to the one obtained by first $\oplus$-normalizing α and then reducing it.

Let α and α' be two ordered e−SLDD formulae. Since the $\oplus$-reduction of a formula preserves its semantics when $\otimes$ is left-cancellative (see Proposition 6), if $\oplus - \mathsf{reduce}(\alpha) = \oplus - \mathsf{reduce}(\alpha')$, then they represent the same function.

Conversely, suppose that α and α' are equivalent. Then they depend on the same variables, say $X = \{x_1, \ldots, x_n\}$; assume w.l.o.g. that the variable ordering under consideration is such that $x_1 < x_2 < \ldots < x_n$.

Let us now prove by induction on n that α and α' have the same $\oplus$-reduced form and satisfy

$$\bigoplus_{\vec{x} \in \vec{X}} I_{\texttt{e-SLDD}}(\oplus - \mathsf{reduce}(\alpha))(\vec{x}) =$$

$$\bigoplus_{\vec{x} \in \vec{X}} I_{\texttt{e-SLDD}}(\oplus - \mathsf{reduce}(\alpha'))(\vec{x}) = 1_s.$$

- Base case: $n = 0$. Since they do not depend on any variable, α and α' are given by a single node, namely the sink node 1_s, and they are labeled by the same offset since α and α' are equivalent. Hence the $\oplus$-reduced forms of α and α' are identical. Furthermore, by convention, $\bigoplus_{a \in \emptyset} a = 1_s$.
- Inductive step: $n > 0$. Let $D_{x_1} = \{d_1, \ldots, d_m\}$. By construction, α (resp. α') has its root N_0 (resp. N'_0) labelled by x_1, an offset q_0 (resp. q'_0) and m outgoing arcs $a_1 = (N_0, M_1), \ldots, a_m = (N_0, M_m)$ (resp. $a'_1 = (N'_0, M'_1), \ldots, a'_m = (N'_0, M'_m)$) such that $\forall j \in 1, \ldots, m$, $v(a_j) = v(a'_j) = d_j$ and the e−SLDD representation rooted at M_j (resp. M'_j) is α_{d_j} (resp. α'_{d_j}).

  Since α and α' are equivalent, then for each $j \in 1, \ldots, m$, α_{d_j} and α'_{d_j} are equivalent. Since none of them depends on x_1, by induction hypothesis, α_{d_j} and α'_{d_j} have the same $\oplus$-reduced form: $\oplus - \mathsf{reduce}(\alpha_{d_j}) = \oplus - \mathsf{reduce}(\alpha'_{d_j})$. Besides, with $X' = \{x_2, \ldots, x_m\}$, we have $\bigoplus_{\vec{x'} \in \vec{X'}} I_{\texttt{e-SLDD}}(\oplus - \mathsf{reduce}(\alpha_{d_j})(\vec{x'}) = \bigoplus_{\vec{x'} \in \vec{X'}} I_{\texttt{e-SLDD}}(\oplus - \mathsf{reduce}(\alpha'_{d_j})(\vec{x'}) = 1_s$.

  At the last iteration step of the $\oplus - \mathsf{reduce}$ procedure (i.e., when every internal node of the formula has been considered except its root), α and α' are as depicted at Figures 3 and 4.

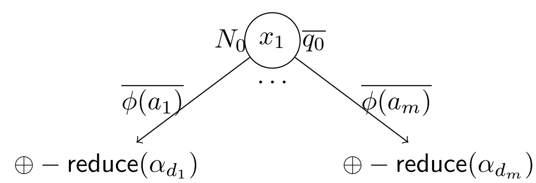

Figure 3: $\oplus$-reduction of the e−SLDD α.

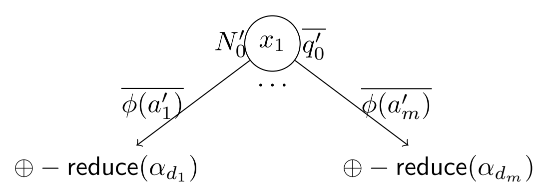

Figure 4: $\oplus$-reduction of the e−SLDD α'.

As explained in the proof of Proposition 6, the normalization part of the last iteration step of the $\oplus - \mathsf{reduce}$ procedure amounts to replacing in α (resp. α') each $\overline{\phi(a_j)}$ (resp. $\overline{\phi(a'_j)}$) ($j \in 1, \ldots, m$) by $\phi(a_j) := \overline{\phi(a_j)} \otimes^{-1} \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$ (resp. $\phi(a'_j) := \overline{\phi(a'_j)} \otimes^{-1} \bigoplus_{j=1}^{m} \overline{\phi(a'_j)}$) and finally $\overline{q_0}$ by $q_0 := \overline{q_0} \otimes \bigoplus_{j=1}^{m} \overline{\phi(a_j)}$ (resp. $\overline{q'_0}$ by $q'_0 := \overline{q'_0} \otimes \bigoplus_{j=1}^{m} \overline{\phi(a'_j)}$).

Let us first prove that for every $j \in 1, \ldots, m$, we have $q_0 \otimes \phi(a_j) = q'_0 \otimes \phi(a'_j)$. Since the $\oplus$-normalizations of α and α' preserve their semantics (see Proposition 6) and since this is also the case of their reductions, the fact that α and α' are equivalent implies that $\oplus-\mathsf{reduce}(\alpha)$ and $\oplus-\mathsf{reduce}(\alpha')$ are equivalent as well. So we have that for each $j \in 1,\ldots,m$, for each $\vec{x} \in \vec{X}$ such that $\vec{x} = \vec{x'} \cup \{(x_1, d_j)\}$, $I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha))(\vec{x}) = I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha'))(\vec{x})$. Hence we have

$$q_0 \otimes (\phi(a_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}))$$
$$= q'_0 \otimes (\phi(a'_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'})).$$

Since this holds for each $\vec{x'} \in \vec{X'}$, we get that

$$\bigoplus_{\vec{x'}\in\vec{X'}} (q_0 \otimes (\phi(a_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'})))$$
$$= \bigoplus_{\vec{x'}\in\vec{X'}} (q'_0 \otimes (\phi(a'_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}))).$$

Since $\otimes$ is associative and since $\otimes$ is left-distributive over $\oplus$, this is equivalent to $q_0 \otimes \phi(a_j) \otimes (\bigoplus_{\vec{x'}\in\vec{X'}} I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'})) = q'_0 \otimes \phi(a'_j) \otimes (\bigoplus_{\vec{x'}\in\vec{X'}} I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}))$. By induction hypothesis, we have

$$\bigoplus_{\vec{x'}\in\vec{X'}} I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}) = 1_s.$$

Since 1_s is the neutral element for $\otimes$, we obtain $q_0 \otimes \phi(a_j) = q'_0 \otimes \phi(a'_j)$, as expected.

It remains to show that $q_0 = q'_0$ and that for each $j \in 1,\ldots,m$, $\phi(a_j) = \phi(a'_j)$. Since $q_0 \otimes \phi(a_j) = q'_0 \otimes \phi(a'_j)$ holds for every $j \in 1,\ldots,m$, we have that $\bigoplus_{j=1}^m (q_0 \otimes \phi(a_j)) = \bigoplus_{j=1}^m (q'_0 \otimes \phi(a'_j))$. Since $\otimes$ is left-distributive over $\oplus$, this is equivalent to $q_0 \otimes \bigoplus_{j=1}^m \phi(a_j) = q'_0 \otimes \bigoplus_{j=1}^m \phi(a'_j)$. Since $\bigoplus_{j=1}^m \phi(a_j) = 1_s = \bigoplus_{j=1}^m \phi(a'_j)$, we get that $q_0 \otimes 1_s = q'_0 \otimes 1_s$. Since 1_s is neutral for $\otimes$, we obtain $q_0 = q'_0$.

Since for each $j \in 1,\ldots,m$, for each $\vec{x} \in \vec{X}$ such that $\vec{x} = \vec{x'} \cup \{(x_1, d_j)\}$, $I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha))(\vec{x}) = I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha'))(\vec{x})$, we get that for each $j \in 1,\ldots,m$, for each $\vec{x'} \in \vec{X'}$,

$$q_0 \otimes (\phi(a_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}))$$
$$= q_0 \otimes (\phi(a'_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'})).$$

As a consequence, for each $j \in 1,\ldots,m$, we have:

$$\bigoplus_{\vec{x'}\in\vec{X'}} (q_0 \otimes (\phi(a_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'})))$$
$$= \bigoplus_{\vec{x'}\in\vec{X'}} (q_0 \otimes (\phi(a'_j) \otimes I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}))).$$

Since $\otimes$ is associative and since $\otimes$ is left-distributive over $\oplus$, this is equivalent to

$$q_0 \otimes \phi(a_j) \otimes (\bigoplus_{\vec{x'}\in\vec{X'}} I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}))$$
$$= q_0 \otimes \phi(a'_j) \otimes (\bigoplus_{\vec{x'}\in\vec{X'}} I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'})).$$

Since by induction hypothesis, $\bigoplus_{\vec{x'}\in\vec{X'}} I_{\texttt{e-SLDD}}(\oplus-\mathsf{reduce}(\alpha_{d_j}))(\vec{x'}) = 1_s$, and 1_s is neutral for $\otimes$, it comes that $q_0 \otimes \phi(a_j) = q_0 \otimes \phi(a'_j)$. Since $\otimes$ is left-cancellative, we get the expected result: $\phi(a_j) = \phi(a'_j)$.

Altogether, we get that $\oplus-\mathsf{reduce}(\alpha) = \oplus-\mathsf{reduce}(\alpha')$.

■

Proof:[Proposition 8] A maximal assignment $\vec{x^*}$ w.r.t. $\succeq$ can be computed in polynomial time by backward induction from the sink of α up to its root N_0. The optimization algorithm consists in computing in a bottom-up way, for every internal node N of α, the values of two synthesized attributes $val(N)$ (a valuation from E) and $suc(N)$ (one of the children nodes of N). More precisely, $val(N)$ is a maximal value w.r.t. $\succeq$ for the e−SLDD formula rooted at N and the $suc(N)$ values aim at tagging a path from N to the sink, corresponding to an assignment leading to this optimal value. Thus $\vec{x^*}$ is associated with the path $N_0, suc(N_0), suc(suc(N_0)), \ldots$ from the root N_0 of α to the sink node in such a way that every variable x labeling a node M in this path is assigned to $d \in D_x$ in $\vec{x^*}$ if and only if $v((M, suc(M)) = d$.

The algorithm is as follows.[5] For the sink node N, we set $val(N)$ to $\otimes$; now for each internal node N of α considered by increasing height (i.e., the length of a longest path from N to the sink), we set $suc(N)$ to one of the children of N such that for every child M of N different from $suc(N)$ we have

$$\phi((N, M)) \otimes val(M) \not\succ \phi((N, suc(N))) \otimes val(suc(N))$$

and we set $val(N)$ to $\phi((N, suc(N))) \otimes val(suc(N))$.

We prove by induction on the height $h(N)$ that the (partial) assignment $\vec{x^*}_N$ associated with the tagged path from N to the sink node is maximal w.r.t. $\succeq$.

- Base case: $h(N) = 0$. N is the sink node, there is no variable to be assigned, hence the optimality of the corresponding empty assignment is obvious.
- Inductive step: suppose that the property holds for every node M of α such that $h(M) \geq k$ and let us prove that it still holds for any node N of α such that $h(N) = k + 1$. It is enough to prove that for every M, if the path from N_0 to the sink node associated with $\vec{x^*}$ contains M, then $\vec{x^*}_M$ is a subset of $\vec{x^*}$ (Bellman's principle of optimality). Towards a contradiction, suppose that this is not the case. Then there exists an assignment $\vec{x'}_M$, different from $\vec{x^*}$, of the variables encountered in a path from M to the sink such that for some valuation $e \in E$ (e is equal to the $\otimes$-combination of the labels $\phi(a)$ of the sequence of arcs from N_0 to M corresponding to $\vec{x^*}$), we have $e \otimes I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x'}_M) \succ e \otimes I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x^*}_M)$, where α_M is the e−SLDD formula rooted at M.

  By induction hypothesis, we also have $I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x'}_M) \not\succ I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x^*}_M)$.

[5] If the purpose is to compute a minimal assignment w.r.t. $\succeq$, then it is enough to replace $\succ$ by $\prec$.

Now, since $\succeq$ is a total pre-order over E, we have that $I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x'}_M) \not\succ I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x^*}_M)$ is equivalent to $I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x^*}_M) \succeq I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x'}_M)$. Then by left monotony of $\otimes$ w.r.t. $\succeq$, we have $e \otimes I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x^*}_M) \succeq e \otimes I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x'}_M)$, which contradicts $e \otimes I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x'}_M) \succ e \otimes I_{\texttt{e-SLDD}}(\alpha_M)(\vec{x^*}_M)$.

■

Proof:[Proposition 9] Any ADD representation α over X can be transformed into an equivalent e-SLDD representation α' in linear time; α' can be generated from α as follows: for any arc $a = (N, M)$ reaching a terminal node M of α labelled by $e \in E$, set $\phi(a)$ to e; for any arc $a = (N, M)$ of α where M is not a leaf, set $\phi(a)$ to 1_s; then merge all leaves of into a single sink (non labelled). By construction, and because $\otimes$ is associative and 1_s is neutral for $\otimes$, the resulting representation α' is a e-SLDD formula such that for every $\vec{x} \in \vec{X}$, $I_{\texttt{ADD}}(\alpha)(\vec{x}) = I_{\texttt{e-SLDD}}(\alpha')(\vec{x})$. ■

Proof:[Proposition 10]

- e-SLDD$_\times \not\leq_s$ e-SLDD$_+$: The point is that the mapping

$$f(x_1, \ldots, x_n) = \Sigma_{i=1}^n 2^{n-i} \times x_i$$

from $\{0,1\}^n$ to $\mathbb{R}^+$ cannot be represented by an e-SLDD$_\times$ formula of size polynomial in n.

Formally, we first show that there is only one max-reduced ordered e-SLDD$_\times$ formula over $X = \{x_1, \ldots, x_n\}$ (with $x_1 < x_2 < \ldots < x_n$) representing $f(x_1, \ldots, x_n) = \Sigma_{i=1}^n 2^{n-i} \times x_i$ and that it has necessarily at least 2^n arcs reaching the sink node, labelled by the integers from $0 to 2^n - 1$. Indeed, the image of f clearly is the set of all integers from 0 to $2^n - 1$. Hence, any ADD formula representing f is tree-shaped and it has 2^n leaves, labelled by those integers. Let us now transform α into an e-SLDD$_\times$ formula β, following the procedure described in the proof of Proposition 9: each of the 2^n valuations labelling the leaves of α are put on their (unique) ingoing arc, and all leaves are merged; all the other arcs a are labelled by $\phi(a) = 1$. Let us now apply the max-normalization procedure to β: for any node N (resp. N') labelled by x_n (i.e., for any node connected directly to the sink), let $\vec{x}_N$ (resp. $\vec{x}_{N'}$) be the corresponding assignment of the $n-1$ first variables and ϕ_N (resp. $\phi_{N'}$) be the value of the image of $\vec{x}_N$ (resp. $\vec{x}_{N'}$) by the restriction of f when $x_n = 0$; N has two outgoing arcs, the one corresponding to the assignment of x_n to 0 (say $a_{N,0}$) and the one corresponding to the assignment of x_n to 1 (say, $a_{N,1}$). It holds that $\phi(a_{N,1}) = \phi_N + 1$ and $\phi(a_{N,0}) = \phi_N$, hence $\phi(a_{N,1}) = \phi(a_{N,0}) + 1$ (see Figure 5). The max-normalization of N starts with the computation of q_{min} equals to $max(\phi_N + 1, \phi_N) = \phi_N + 1$. After the update, $\phi(a_{N,1})$ is equal to 1 and $\phi(a_{N,0})$ is equal to $c = \phi_N \times^{-1} (\phi_N + 1)$; since c is such that $(\phi_N + 1) \times c = \phi_N$, we have $c = \frac{\phi_N}{\phi_N + 1}$. Finally the valuation labeling the arc from the father M of N to N is updated (it is multiplied by $\phi_N + 1$). If $N \neq N'$ then we have $\phi_N \neq \phi_{N'}$. As a consequence, we also have $\frac{\phi_N}{\phi_N+1} \neq \frac{\phi_{N'}}{\phi_{N'}+1}$. Because the initial values $\phi(a_{N,0}) = \phi_N$ of the 0-outgoing arcs of the 2^{n-1} nodes N of β labeled by x_n are pairwise distinct, the corresponding updated labels $\phi(a_{N,0})$ are also pairwise distinct. As a consequence, their father nodes cannot be merged, and since there are 2^{n-1} such nodes, the e-SLDD$_\times$ formula obtained by reducing the resulting max-normalized diagram of β still contains exponentially many nodes.

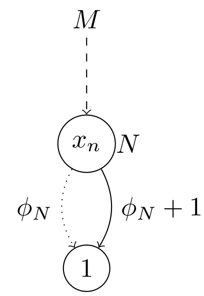

Figure 5: max-normalization of β.

Contrastingly, $f(x_1, \ldots, x_n) = \Sigma_{i=1}^n 2^{n-i} \times x_i$ can be represented by the ordered e-SLDD$_+$ formula given at Figure 6. This e-SLDD$_+$ formula has n nodes labelled by x_1 to x_n, plus the sink. Each internal node N labelled by some x_i has two outgoing arcs $a_{N,1}$ and $a_{N,0}$ with $v(a_{N,1}) = 1$, $v(a_{N,0}) = 0$, $\phi(a_{N,1}) = 2^{n-i}$ and $\phi(a_{N,0}) = 0$. A straightforward induction shows that the interpretation of this e-SLDD$_+$ formula is equal to $\Sigma_{i=1}^n 2^{n-i} \times x_i$.

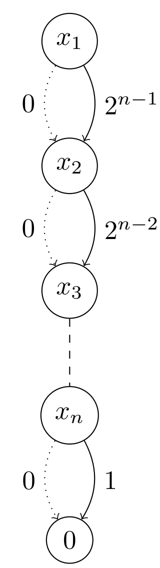

Figure 6: An ordered e-SLDD$_+$ representation of $f(x_1, \ldots, x_n) = \Sigma_{i=1}^n 2^{n-i} \times x_i$.

- e-SLDD$_+$ $\not\leq_s$ e-SLDD$_\times$: The proof is quite similar to the previous one, with

$$f(x_1, \ldots, x_n) = \Pi_{i=1}^{n} \gamma^{2^{n-i} \times x_i},$$

where $0 < \gamma < 1$. Observe that $f(x_1, \ldots, x_n)$ is also equal to $\gamma^{\Sigma_{i=1}^{n} 2^{n-i} x_i}$.

Formally, we first show that there is only one *min*-reduced ordered e-SLDD$_+$ formula over $X = \{x_1, \ldots, x_n\}$ (with $x_1 < x_2 < \ldots < x_n$) representing $f(x_1, \ldots, x_n) = \Pi_{i=1}^{n} \gamma^{2^{n-i} \times x_i}$ and that it has necessarily at least 2^n arcs reaching the sink node, labelled by the numbers of the form γ^i with $i \in 0, \ldots, 2^{n-1}$. Indeed, the image of f clearly is the set $\{\gamma^i \mid i \in 0, \ldots, 2^{n-1}\}$. Hence, any ADD formula representing f is tree-shaped and it has 2^n leaves, labelled by those numbers. Let us now transform α into an e-SLDD$_+$ formula β, following the procedure described in the proof of Proposition 9: each of the 2^n valuations labelling the leaves of α are put on their (unique) ingoing arc, and all leaves are merged; all the other arcs a are labelled by $\phi(a) = 0$. Let us now apply the *min*-normalization procedure to β: for any node N (resp. N') labelled by x_n (i.e., for any node connected directly to the sink), let $\vec{x}_N$ (resp. $\vec{x}_{N'}$) be the corresponding assignment of the $n-1$ first variables and ϕ_N (resp. $\phi_{N'}$) be the value of the image of $\vec{x}_N$ (resp. $\vec{x}_{N'}$) by the restriction of f when $x_n = 0$; N has two outgoing arcs, the one corresponding to the assignment of x_n to 0 (say $a_{N,0}$) and the one corresponding to the assignment of x_n to 1 (say, $a_{N,1}$). It holds that $\phi(a_{N,1}) = \gamma \times \phi_N$ and $\phi(a_{N,0}) = \phi_N$, hence $\phi(a_{N,1}) = \gamma \times \phi(a_{N,0})$ (see Figure 7). The *min*-normalization of N starts with the computation of q_{min} equals to $min(\gamma \times \phi_N, \phi_N) = \gamma \times \phi_N$. After the update, $\phi(a_{N,1})$ is equal to 0 and $\phi(a_{N,0})$ is equal to $c = \phi_N +^{-1} (\gamma \times \phi_N)$; since c is such that $(\gamma \times \phi_N) + c = \phi_N$, we have $c = \phi_N \times (1 - \gamma)$. Finally the valuation labeling the arc from the father M of N to N is updated ($\gamma \times \phi_N$ is added to its current value). If $N \neq N'$ then we have $\phi_N \neq \phi_{N'}$. As a consequence, we also have $\phi_N \times (1-\gamma) \neq \phi_{N'} \times (1-\gamma)$. Because the initial values $\phi(a_{N,0}) = \phi_N$ of the 0-outgoing arcs of the 2^{n-1} nodes N of β labeled by x_n are pairwise distinct, the corresponding updated labels $\phi(a_{N,0})$ are also pairwise distinct. As a consequence, their father nodes cannot be merged, and since there are 2^{n-1} such nodes, the e-SLDD$_+$ formula obtained by reducing the resulting *min*-normalized diagram of β still contains exponentially many nodes.

Contrastingly, $f(x_1, \ldots, x_n) = \Pi_{i=1}^{n} \gamma^{2^{n-i} \times x_i}$ can be represented by the ordered e-SLDD$_\times$ formula given at Figure 8. This e-SLDD$_\times$ formula has n nodes labelled by x_1 to x_n, plus the sink. Each internal node N labelled by some x_i has two outgoing arcs $a_{N,1}$ and $a_{N,0}$ with $v(a_{N,1}) = 1$, $v(a_{N,0}) = 0$, $\phi(a_{N,1}) = \gamma^{2^{n-i}}$ and $\phi(a_{N,0}) = 1$. A straightforward induction shows that the interpretation of this e-SLDD$_\times$ formula is equal to $\Pi_{i=1}^{n} \gamma^{2^{n-i} \times x_i}$.

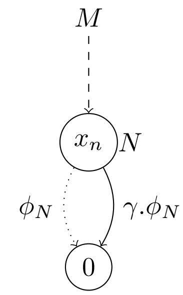

Figure 7: *min*-normalization of β.

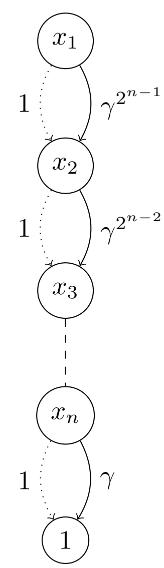

Figure 8: An ordered e-SLDD$_\times$ representation of $f(x_1, \ldots, x_n) = \Pi_{i=1}^{n} \gamma^{2^{n-i} \times x_i}$.

- AADD$<_s$ e−SLDD$_+$: since e−SLDD$_+$ is polynomially translatable into AADD (set all the f to 1), we have AADD$\leq_s$ e−SLDD$_+$; moreover, since e−SLDD$_\times$ is polynomially translatable into AADD (set all the q to 0), a polynomial AADD representation of $f(x_1, \ldots, x_n) = \Pi_{i=1}^n \gamma^{2^{n-i} \times x_i}$ exists, while this is not the case for e−SLDD$_+$. Hence AADD$<_s$ e−SLDD$_+$.
- AADD$<_s$ e−SLDD$_\times$: since e−SLDD$_\times$ is polynomially translatable into AADD (set all the q to 0),we have AADD$\leq_s$ e−SLDD$_\times$; moreover, since e−SLDD$_+$ is polynomially translatable into AADD (set all the f to 1), a polynomial AADD representation of $f(x_1, \ldots, x_n) = \Sigma_{i=1}^n 2^{n-i} \times x_i$ exists, while this is not the case for e−SLDD$_\times$. Hence AADD$<_s$ e−SLDD$_\times$.
- e−SLDD$_+$ $<_s$ ADD: that e−SLDD$_+$ $\leq_s$ ADD is given by Proposition 9; ADD $\not\leq_s$ e−SLDD$_+$ holds since $f(x_1, \ldots, x_n) = \Sigma_{i=1}^n 2^{n-i} \times x_i$ has no polynomially-sized ADD representation while it has a polynomially-sized e−SLDD$_+$ representation.
- e−SLDD$_\times$ $<_s$ ADD: that e−SLDD$_\times$ $\leq_s$ ADD is given by Proposition 9; ADD $\not\leq_s$ e−SLDD$_\times$ holds since $f(x_1, \ldots, x_n) = \Pi_{i=1}^n \gamma^{2^{n-i} \times x_i}$ has no polynomially-sized ADD representation while it has a polynomially-sized e−SLDD$_\times$ representation.
- e−SLDD$_{min}$$\sim_p$ ADD: From Proposition 9, we know that e−SLDD$_{min}$ $\leq_l$ ADD; so obviously e−SLDD$_{min}$ $\leq_p$ ADD.

  Let us now show that ADD $\leq_p$ e−SLDD$_{min}$. Let α be a e−SLDD$_{min}$ representation over X. Let us explain how to generate from α in polynomial time an ADD representationn β representing the same mapping as the one represented by α. The approach consists in parsing α in a top-down way, by decreasing depth. If α is reduced to the sink node, then β is a single-node ADD representation labeled by the offset of α. Otherwise, in the general case, every internal node N^{val_M} of β will be associated with two further labels which are parts of the node identifier: a valuation val from E and a pointer to the corresponding node M in α. At any step, the variable labeling N_M^{val} is the same as the one labeling M. At start, the root node of β is $N^{q_0 N_0}$ where q_0 is the offset of α, and N_0 is the root node of α. Then for every leaf node N_M^{val} of β, for every arc $(M, P) \in out(M)$ in α labelled with $v(a)$, if P is the sink node, then add an arc in β from N_M^{val} to the terminal node of β labelled by the valuation $min(val, \phi(a))$ (this node is not associated with a node in α); otherwise (i.e., if P is not the sink node), then add an arc in β from N_M^{val} to the node $N_P^{min(val,\phi(a))}$. Whatever the case, the arc added to β is also labelled by $v(a)$.

  Figure 9 gives an e−SLDD$_{min}$ formula α and Figure 10 gives the corresponding ADD representation computed using the procedure described above. Observe that this ADD is not reduced in the general case (Figure 11 gives the corresponding reduced ADD).

  The generation algorithm described above rules in time polynomial in the size of α. Especially, the ADD repre-

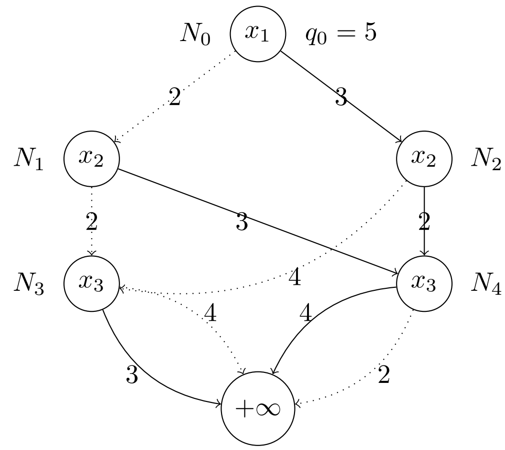

Figure 9: An e−SLDD$_{min}$ representation α. The domain of all variables x_1, x_2, x_3 is $\{0, 1\}$. Every dotted (resp. plain) arc a corresponds to the value $v(a) = 0$ (resp. $v(a) = 1$).

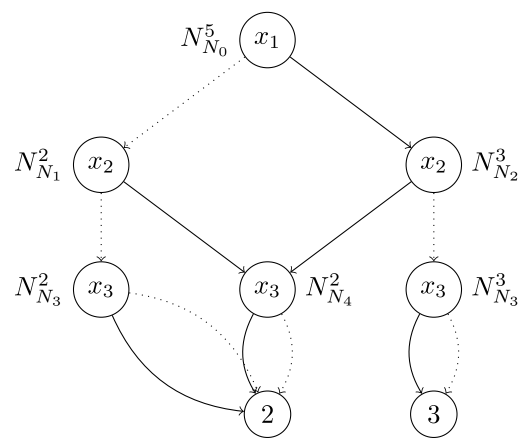

Figure 10: An ADD representation equivalent to α. The domain of all variables is $\{0, 1\}$. Every dotted (resp. plain) arc a corresponds to the value $v(a) = 0$ (resp. $v(a) = 1$).

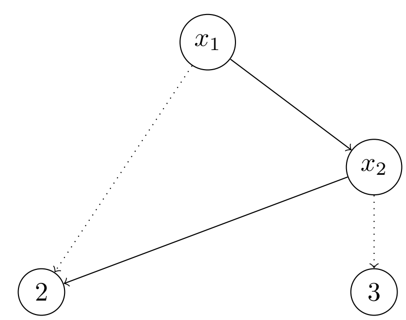

Figure 11: A reduced ADD representation equivalent to α. The domain of all variables is $\{0, 1\}$. Every dotted (resp. plain) arc a corresponds to the value $v(a) = 0$ (resp. $v(a) = 1$).

sentation β which is generated from α contains no more than $n \times k$ nodes and no more than $m \times k$ arcs, where n is the number of nodes of α, m the number of arcs of α and k is the cardinality of $\{\phi(a) \mid a \in \alpha\}$, i.e., the number of different valuations labeling the arcs of α. By construction, for every assignment $\vec{x} \in \vec{X}$, we have $I_{\texttt{e-SLDD}_{min}}(\alpha)(\vec{x}) = I_{\texttt{ADD}}(\beta)(\vec{x})$, which shows that α and β represent the same mapping.

- $\texttt{e-SLDD}_{max} \sim_p \texttt{ADD}$: The proof is the same as the previous one, replacing "min" by "max".

■