Compilation map for configuration

Projet ANR-11-BS02-008

BR4CP

Rapport technique CRIL–2014–002

Sylvie Coste-Marquis

January 7, 2014
Contents

1 Introduction .................................................. 3

2 Choices of languages .......................................... 4
   2.1 Boolean valuation languages .......................... 4
   2.2 Non Boolean valuation languages .................... 6

3 Choices of queries and transformations .................... 8
   3.1 Queries ................................................... 8
   3.2 Transformations ......................................... 10

4 Compilation map ............................................... 10
   4.1 Queries ................................................... 11
   4.2 Transformations ......................................... 12

References ....................................................... 14
1 Introduction

e-commerce tries to match the needs and preferences of customers with the possibilities and goals of the business offer. Configuration softwares provide the customer with a recommendation matching her preferences in order to lead her to buy a product. When products are configurable (such as computers or cars for example), the search space is strongly combinatorial. In order to design an efficient configuration tool, we must deal with the response time of the system - crucial because the configuration process runs online. Many existing configuration systems defines the e-catalog by a constraint satisfaction problem. A valid configuration assigns values to the variables such that all compatibility constraints hold. In such configuration systems, the customer describes her choices by assigning values to some configuration variables (for instance, she wants a red car without air conditioning). The tool generates a solution, if there exists one or else indicates that the choices made so far do not lead to any suitable product.

We propose to use techniques from works on compilation in order to address the problem of response time. The idea is to build off-line a compact compiled form of the e-catalog. This compiled form is then used to propagate the choices made by the customer. Turning the model into a compiled form off-line allows for more efficient treatment on-line afterwards. This compilation step occurs only once and off-line so it doesn’t matter if the compilation time is important. Many languages exist for compiling the e-catalog. Among them, we can distinguish two kinds of languages : classical languages with boolean valuation and non-boolean valuations languages allowing for the representation of the prices of the products (for instance, the price of the configured car).

For propositional languages, many known results establish a knowledge compilation map: Such a map analyzes the compilation target languages w.r.t. succinctness (a form of space efficiency) and the queries (and transformations) they can handle in polynomial time (time efficiency). For non-boolean languages, the compilation map exists only partially. However the existing maps are general and contain all standard queries and transformations. Configuration requests new queries: explaining why a choice of the user does not lead to any possible product or coming back on previous choices that lead to an inconsistency. Furthermore, we need to consider new languages allowing for computing the minimum price of a product compatible with the current choice of the customer, not only the fact that a solution exists or not.

We propose in this report:

1. To identify the queries and transformations useful for configuration;
2. To study the complexity of these queries and transformations for different
target compilation languages;
3. To identify new target compilation languages offering a good computational behavior (spatial and computational efficiency).

2 Choices of languages

A target compilation language has to be distinguished from a representation language: a representation language must be easily understandable by a human reader. A target compilation language must allow for efficient queries and transformations and is not concerned by such a problem of being understandable.

Two kinds of target compilation languages are studied in this review: Boolean valuation languages and non-Boolean valuation languages.

Retained languages for this report are: NNF, DNNF, OBDD<, OMDD<, ToOBDD< and ToMODS for the Boolean valuation languages and ADD, SLDD+, AADD and vCSP for non-Boolean valuation ones.

2.1 Boolean valuation languages

Let us define the considered languages.

Considered languages are mainly subsets of the NNF language [Darwiche, 1999]. Definitions come mainly from [Darwiche, 1999; Darwiche and Marquis, 2002].

Definition 1 (NNF). Let PS be a denumerable set of propositional variables. A sentence in NNF is a rooted, directed acyclic graph (DAG) where each leaf node is labeled with true, false, X or ¬X, X ∈ PS; and each internal node is labeled with ∨ or ∧ and can have arbitrarily many children.

Definition 2 (CNF and DNF). The language CNF (resp. DNF) is the subset of NNF satisfying flatness (the height of each sentence is at most 2) and simple disjunction (resp. simple conjunction).

We consider also nested subsets of the NNF language that do not have any restriction on the height of the sentences. The DNNF language although impose one of the following properties.

Properties.

Decomposability An NNF satisfies this property if for each conjunction C in the NNF, the conjuncts of C do not share variables. That is, if C_1, . . . , C_n are the children of an and-node C, then \( \text{Vars}(C_i) \cap \text{Vars}(C_j) = \emptyset \) for \( i \neq j \).
**Determinism** An NNF satisfies this property if for each disjunction $C$ in the NNF, each two disjuncts of $C$ are logically contradictory. That is, if $C_1, \ldots, C_n$ are the children of or-node $C$, then $C_i \land C_j$ false for $i \neq j$.

**Smoothness** An NNF satisfies this property if for each disjunction $C$ in the NNF, each disjunct of $C$ mentions the same variables. That is, if $C_1, \ldots, C_n$ are the children of or-node $C$, then $\text{Vars}(C_i) = \text{Vars}(C_j)$ for $i \neq j$.

**Definition 3 (DNNF).** The language $\text{DNNF}$ is the subset of NNF satisfying decomposability.

**Definition 4 (Decision node).** A decision node $N$ in an NNF sentence is one which is labeled with true, false, or is an or-node having the form $(X \land \alpha) \lor (\neg X \land \beta)$, where $X$ is a variable, $\alpha$ and $\beta$ are decision nodes. In the latter case, $d\text{Var}(N)$ denotes the variable $X$.

**Definition 5 (BDD).** The language $\text{BDD}$ is the set of NNF sentences, where the root of each sentence is a decision node.

**Definition 6 (FBDD).** $\text{FBDD}$ is the intersection of $\text{DNNF}$ and $\text{BDD}$.

**Definition 7 (OBDD$<_<$).** Let $<$ be a total ordering on the variables PS. The language $\text{OBDD}_<^<$ is the subset of $\text{FBDD}$ satisfying the following property: if $N$ and $M$ are or-nodes, and if $N$ is an ancestor of $M$, then $d\text{Var}(N) < d\text{Var}(M)$.

Another language considered is the $\text{OMDD}_<^<$ language. As the necessary queries and transformations remain NP-hard for CSP, the $\text{OMDD}_<^<$ language has been proposed as a way to compile CSPs. This language has been defined in [Amilhastre et al., 2012].

**Definition 8 (MDD).** A multivalued decision diagram on a set of finite domain variables $X$ is a directed acyclic graph $\phi = < N, E >$ where $N$ is a set of nodes containing at most one root and at most one leaf (denoted the sink). Except for the sink, each node $n \in N$ is labeled by a variable $\text{Var}(N)$ in $X$ and each edge $E \in E$ going out of $n$ is labeled by a value $\text{Lbl}(E)$ in the domain of $\text{Var}(N)$.

**Definition 9 (OMDD$<_<$).** Let $<$ be a total ordering on the variables PS. The language $\text{OMDD}_<^<$ is the language of MDD ordered w.r.t. $<$. Another suited language as target compilation language is the $\text{TOBDD}_<^<$ one [Subbarayan et al., 2007].

**Definition 10 (tree-of-BDDs).**
A decomposition tree of a CNF formula $\Sigma$ is a (finite) labelled tree $T$ whose set of nodes is $N$. Each node $n \in N$ is labelled with $\text{Var}(n)$, a subset of $\text{Var}(\Sigma)$. For each $n \in N$, let $\text{clauses}(n) = \{ \text{clause } \delta \text{ s.t } \text{Var}(\delta) \subseteq \text{Var}(n) \}$; $T$ satisfies two conditions: for every clause $\delta$ of $\Sigma$ there exists $n \in N$ such that $\delta \in \text{clauses}(n)$, and for every $x \in \text{Var}(\Sigma)$, $\{ n \in N | x \in \text{Var}(n) \}$ forms a connected subtree of $T$.

Let $<$ be a total strict ordering over $PS$. A tree-of-BDDs of a CNF formula $\Sigma$ given $<$ consists of a decomposition tree $T$ of $\Sigma$ equipped with a further labelling function $B$ such that for every $n \in N$, $B(n)$ is the OBDD representation of $\exists \text{Var}(n).I(\Sigma)$.

We have $\text{Var}(T) = \bigcup_{n \in N} \text{Var}(n)$ and $I(T) = \bigwedge_{n \in N} I(B(n))$.

### 2.2 Non Boolean valuation languages

For non-Boolean valuation languages, we consider: $v$CSP, ADD, SLDD+ and AADD. We define here these languages with the definitions given in [Fargier et al., 2013].

Given a finite set $X = \{ x_1, ..., x_n \}$ 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 assignment $\vec{x} = \{ (x_i, d_i) | d_i \in D_{x_i}, i = 1, ..., n \}$. $E$ is the carrier of a valuation structure $\mathcal{E}$.

**Definition 11 (VDD).** A valued decision diagram (VDD) over $X$ w.r.t. $\mathcal{E}$ is a finite DAG $\alpha$ with a single root, s.t. every internal node $n$ is labelled with a variable $x \in X$ and if $D_x = \{ d_1, ..., d_k \}$, then $n$ has $k$ outgoing arcs $a_1, ..., a_k$ so that the arc $a_i$ of $\alpha$ is valued by $\nu(a_i) = d_i$. We note $\text{out}(n)$ (resp. $\text{in}(n)$) the arcs outgoing from (resp. incoming) to $n$. Nodes and arcs can also be labelled by elements of $E$: if $\phi(n)$ (resp. $\phi(a_i)$) is a node (resp. an arc) of $\alpha$, then $\phi(n)$ (resp. $\phi(a_i)$) denotes the label of $n$ (resp. $a_i$). Finally, each VDD is a read-once formula, i.e., for each path from the root of $\alpha$ to a sink, every variable $x \in X$ occurs at most once as a node label.

No specific assumption is made on the valuation structure $\mathcal{E}$ for defining ADD [Bahar et al., 1993].

**Definition 12 (ADD).** ADD is the 4-tuple $\langle C_{\text{ADD}}, \text{Var}_{\text{ADD}}, I_{\text{ADD}}, s_{\text{ADD}} \rangle$ where $C_{\text{ADD}}$ is the set of ordered VDDs over $X$ such that sinks $S$ are labelled by elements of $E$, and the arcs are not labelled; $I_{\text{ADD}}$ is defined inductively by: for every assignment $\vec{x}$ over $X$,

- if $\alpha$ is a sink node $S$, labelled by $\phi(S) = e$, then $I_{\text{ADD}}(\alpha)(\vec{x}) = e$,
• else the root $N$ of $\alpha$ 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 $\beta$ the ADD formula rooted at node $M$ in $\alpha$; we have $I_{ADD}(\alpha)(\vec{x}) = I_{ADD}(\beta)(\vec{x})$.

For defining $\text{SLDD}^+$, the valuation structure $\mathcal{E}$ must be a commutative semiring [Wilson, 2005].

**Definition 13 (SLDD).** Let $\mathcal{E} = \langle E, \oplus, \otimes, 0_s, 1_s \rangle$ be a commutative semiring. $\text{SLDD}$ is the 4-tuple $\langle C_{\text{SLDD}}, \text{Var}_{\text{SLDD}}, I_{\text{SLDD}}, s_{\text{SLDD}} \rangle$ where $C_{\text{SLDD}}$ is the set of VDDs over $X$ with a unique sink $S$, satisfying $\phi(S) = 1_s$, and such that the arcs are labelled by elements of $E$, and $I_{\text{SLDD}}$ is defined inductively by: for every assignment $\vec{x}$ over $X$,

- if $\alpha$ is the sink node $S$, then $I_{\text{SLDD}}(\alpha)(\vec{x}) = 1_s$,
- else the root $N$ of $\alpha$ 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 $\beta$ the SLDD formula rooted at node $M$ in $\alpha$; we have $I_{\text{SLDD}}(\alpha)(\vec{x}) = \phi(a) \otimes I_{\text{SLDD}}(\beta)(\vec{x})$.

We consider $\text{SLDD}^+$ language that impose $\oplus = +$.

In the AADD framework [Sanner and McAllester, 2005], the valuation set $E = \mathbb{R}^+$. 

**Definition 14 (AADD).** $\text{AADD}$ is the 4-tuple $\langle C_{\text{AADD}}, \text{Var}_{\text{AADD}}, I_{\text{AADD}}, s_{\text{AADD}} \rangle$ where $C_{\text{AADD}}$ is the set of ordered VDDs over $X$ with a unique sink $S$, satisfying $\phi(S) = 1$, and such that the arcs are labelled by pairs $(q, f)$ in $\mathbb{R}^+ \times \mathbb{R}^+$, $I_{\text{AADD}}$ is defined inductively by: for every assignment $\vec{x}$ over $X$,

- if $\alpha$ is the sink node $S$, then $I_{\text{AADD}}(\alpha)(\vec{x}) = 1$,
- else the root $N$ of $\alpha$ 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 $v(a) = d$, $\phi(a) = (q, f)$, and $\beta$ the AADD formula rooted at node $M$ in $\alpha$; we have $I_{\text{AADD}}(\alpha)(\vec{x}) = q + (f \times I_{\text{AADD}}(\beta)(\vec{x}))$.

Finally, we include into our report the $\text{vCSP}$ language [Schiex et al., 1995]. We need the definition of a valuation structure.

**Definition 15 (Valuation structure).** A valuation structure $S = \langle E, \otimes, > \rangle$ verifies:

- $E$ is a set, whose elements are called valuations, which is totally ordered by $>$, with a maximum element noted $\top$ and a minimum element noted $\bot$;
• \( \otimes \) is a commutative, associative closed binary operation on \( E \) which verifies:
  
  – Identity;
  – Monotonicity;
  – And has an absorbing element.

A valued CSP is thus defined by adding to each constraint of a classical CSP a valuation.

**Definition 16 (vCSP).** A valued CSP is defined by a classical CSP \( \langle X, D, C \rangle \), a valuation structure \( S = \langle E, \otimes, > \rangle \) and an application \( \phi \) from \( C \) to \( E \). It is noted \( \langle X, D, C, S, \phi \rangle \).

## 3 Choices of queries and transformations

The requests and transformations we identified useful for configuration [Lhomme, 2013] are the following ones. They are formed of some of the classical ones identified in [Darwiche and Marquis, 2002] and new ones more oriented towards configuration [Astesana et al., 2010; Amilhastre et al., 2002; 2012].

### 3.1 Queries

Here are the queries considered with their meaning in configuration.

**Definitions in Boolean valuation languages**

**CO** Checking the consistency of the configurator

**VA** Is the configurator valid?

**CT** Counting the number of solutions

**MX** Extracting one solution

**CX** Providing the user with all possible values of a variable

**EQ** Equivalence test

**REST** Computing a restorated choice of variables (maximal)

**EXPL** Finding an explanation of an inconsistent choice of variables (minimal)
We define formally these queries [Darwiche and Marquis, 2002]. Let $\mathcal{C}$ denote a representation language.

- $\mathcal{C}$ satisfies CO (resp. VA) iff there exists a polytime algorithm that maps every $\mathcal{C}$ representation $\alpha$ to 1 if $\alpha$ is consistent (resp. valid), and to 0 otherwise.

- $\mathcal{C}$ satisfies EQ iff there exists a polytime algorithm that maps every pair of $\mathcal{C}$ representations $\alpha, \beta$ to 1 if $\alpha \equiv \beta$ holds, and to 0 otherwise.

- $\mathcal{C}$ satisfies CT iff there exists a polytime algorithm that maps every $\mathcal{C}$ representation $\alpha$ to a nonnegative integer that represents the number of models of $\alpha$ over $\text{Var}(\alpha)$ (in binary notation).

- $\mathcal{C}$ satisfies MX iff there exists a polytime algorithm that maps every $\phi \in \mathcal{C}$ to one model of $\phi$ if there is one and stops without returning anything otherwise.

- $\mathcal{C}$ satisfies CX iff there exists a polytime algorithm that outputs, for any $\phi \in \mathcal{C}$ and any $x \in \text{Var}(\phi)$, the set of values taken by $x$ in at least one model of $\phi$.

- $\mathcal{C}$ satisfies REST iff there exists a polytime algorithm that outputs, for every inconsistent $\phi \in \mathcal{C}$ and a set of value taken by $x \in \text{Var}(\phi)$, a maximal consistent set of variables $x \in \text{Var}(\phi)$ (from [Amilhastre et al., 2002]).

**Meaning in non-boolean valuation languages**

- **Prix**\textsubscript{min/max} Finding the minimal or maximal price of the models of the configuration

- **CT**\textsubscript{min} Counting the number of solutions having minimal price

- **CT**\textsubscript{$\leq$} Counting the number of solutions with a price inferior or equal to a value

- **MX**\textsubscript{min} Extracting one solution with minimal price

- **MX**\textsubscript{$\leq$} Extracting one solution with a price inferior to a given value

- **CX**\textsubscript{min} Providing the user with all possible value of interesting variables leading to minimal price

- **CX**\textsubscript{$\leq$} Providing the user with all possible value of interesting variables leading to a price inferior or equal to a value
3.2 Transformations

The transformations we retained useful for configuration are the following.

CD Assigns some values to some variables

FO Eliminates some variables

SFO Eliminates one variable

∧BC Closure under bounded conjunction

Definitions in Boolean valuation languages

- C satisfies CD iff there exists a polytime algorithm that maps every C representation \( \alpha \) and every consistent term \( \gamma \) to a C representation \( \beta \) which is logically equivalent to \( \alpha|\gamma \), the conditioning of \( \alpha \) on \( \gamma \).

- C satisfies FO iff there exists a polytime algorithm that maps every C representation \( \alpha \) and every subset \( X \) of variables from PS to a C representation \( \beta \) equivalent to \( \exists X.\alpha \), the forgetting of \( X \) from \( \alpha \). If the property holds for each singleton \( X \), we say that \( C \) satisfies SFO.

- C satisfies \( \land BC \) iff there exists a polytime algorithm that maps every pair of C representations \((\alpha, \beta)\) to a C representation \( \gamma \) which is logically equivalent to \( \alpha \land \beta \).

Meaning and definitions in non-Boolean valuation languages

In non-Boolean valuation languages, FO and SFO are replaced by the following transformations.

\( \Sigma_e \) Eliminates some variables (sum)

\( \Sigma_{se} \) Eliminates one variable (sum)

4 Compilation map

We first present results for the queries and then results for the transformations.
Table 1: Polytime queries for retained boolean valuation languages. ✓ means ”satisfies” and ◦ means ”does not satisfy unless P = NP”.

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>VA</th>
<th>CT</th>
<th>MX</th>
<th>CX</th>
<th>EQ</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBDD_&lt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OMDD_&lt;</td>
<td>✓</td>
<td>◦</td>
<td>◦</td>
<td>✓</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓</td>
<td>◦</td>
<td>◦</td>
<td>✓</td>
<td>◦</td>
<td>◦</td>
<td>✓</td>
</tr>
<tr>
<td>CNF</td>
<td>◦</td>
<td>✓</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>ToOBDD_&lt;</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

4.1 Queries

Table 1 recalls the results for queries for Boolean valuation languages. We give here the references of the papers where the presented results were proven. When the color of the symbols is red, the result has not been published yet.

OBDD_< Results for CO, VA, CT and EQ come from [Gergov and Meinel, 1994; Bryant, 1992].

OMDD_< Results come from [Amilhastre et al., 2012].

DNNF Results for CO, VA, CT and EQ come from [Darwiche and Marquis, 2002].

CNF Results for CO, VA, CT and EQ come from [Darwiche and Marquis, 2002].

ToOBDD_< Results for CO and VA come from [Subbarayan et al., 2007].

Table 2 presents the results for queries in non-Boolean valuation languages. Note that not all of these results have been published yet and some of them are still under work.

Prix_{min/max} These results come from the fact that optimization is feasible in polynomial time for ADD, SLDD+ and AADD[Bahar et al., 1993; Wilson, 2005; Sanner and McAllester, 2005].

MX_{min} idem.

MX_{≤} idem.

vCSP Results are established (or direct consequences of) in [Schiex et al., 1995].
Table 2: Polytime queries for non-boolean valuation languages. ✓ means "satisfies" and ◦ means "does not satisfy unless P = NP".

<table>
<thead>
<tr>
<th></th>
<th>Prix_{min/max}</th>
<th>CT_{min}</th>
<th>CT_{&lt;}</th>
<th>MX_{min}</th>
<th>MX_{&lt;}</th>
<th>CX_{min}</th>
<th>CX_{&lt;}</th>
<th>EQ</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLDD+</td>
<td>✓</td>
<td>✓</td>
<td>◦</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADD</td>
<td>✓</td>
<td>✓</td>
<td>◦</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AADD</td>
<td>✓</td>
<td>✓</td>
<td>◦</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ψCSP</td>
<td>◦</td>
<td>?</td>
<td>?</td>
<td>o</td>
<td>o</td>
<td>?</td>
<td>?</td>
<td>o</td>
<td>?</td>
</tr>
</tbody>
</table>

4.2 Transformations

Table 3 presents the results for transformation and Boolean valuation languages.

Table 3: Polytime transformations for boolean valuation languages. ✓ means "satisfies", • means "does not satisfy" and ◦ means "does not satisfy unless P = NP".

<table>
<thead>
<tr>
<th></th>
<th>CD</th>
<th>∧BC</th>
<th>FO</th>
<th>SFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBDD&lt;</td>
<td>✓</td>
<td>✓</td>
<td>•</td>
<td>✓</td>
</tr>
<tr>
<td>OMDD&lt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓</td>
<td>◦</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CNF</td>
<td>✓</td>
<td>✓</td>
<td>◦</td>
<td>✓</td>
</tr>
<tr>
<td>ToOBDD&lt;</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>?</td>
</tr>
</tbody>
</table>

We give here the origins of the presented results.

OBDD< CD and ∧BC are well-known since [Bryant, 1986]. Other results have been established in [Darwiche and Marquis, 2002].

OMDD< Results have been established in [Amilhastre et al., 2012].

DNNF Results for CD and ∧BC come from [Darwiche and Marquis, 2002]. FO has been proven in [Darwiche, 2001], SFO is thus straightforward.

CNF [Darwiche and Marquis, 2002] proves CD, FO, SFO and ∧BC.

ToOBDD< CD, FO and ∧BC were proven in [Fargier and Marquis, 2009].

Table 4 presents the results for transformations and non-Boolean languages. Note that most of these results have not been published yet.
Table 4: Polytime transformations for non-boolean valuation languages. ✓ means "satisfies" and ◦ means "does not satisfy unless P = NP".

<table>
<thead>
<tr>
<th></th>
<th>CD</th>
<th>∧BC</th>
<th>Σ_e</th>
<th>Σ_sp</th>
<th>Σ_se</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLDD+</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ADD</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AADD</td>
<td>✓</td>
<td>◦</td>
<td>◦</td>
<td>✓</td>
<td>◦</td>
</tr>
<tr>
<td>vCSP</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

References


