The Expressive Power of Ad-Hoc Constraints for Modelling CSPs

Ruiwei Wang and Roland H.C. Yap

School of Computing, National University of Singapore,
13 Computing Drive, 117417, Singapore
{ruiwei,ryap}@comp.nus.edu.sg

Abstract

Ad-hoc constraints (also called generic constraints) are important for modelling Constraint Satisfaction Problems (CSPs). Many representations have been proposed to define ad-hoc constraints, such as tables, decision diagrams, binary constraint trees, automata and context-free grammars. However, prior works mainly focus on efficient Generalized Arc Consistency (GAC) propagators of ad-hoc constraints using the representations. In this paper, we ask a more fundamental question which bears on modelling constraints in a CSP as ad-hoc constraints, how the choice of constraints and operations affect tractability. Rather than ad-hoc constraints and their GAC propagators, our focus is on their expressive power in terms of succinctness (polysize) and cost of operations/queries (polytime). We use a large set of constraint families to investigate the expressive power of 14 existing ad-hoc constraints. We show a complete map of the succinctness of the ad-hoc constraints. We also present results on the tractability of applying various operations and queries on the ad-hoc constraints. Finally, we give case studies illustrating how our results can be useful for questions in the modelling of CSPs.

Introduction

A wide range of combinatorial problems in real life can be modelled with Constraint Satisfaction Problems (CSPs) using a diverse of constraints. We will focus on ad-hoc constraints (also called generic constraints) that can be defined with various representations, such as tables (Bessière and Régin 1997; Lecoutre 2011; Lecoutre, Likitvivatanavong, and Yap 2015; Wang et al. 2016; Demeulenaere et al. 2016; Verhaeghe, Lecoutre, and Schaus 2017; Wang and Yap 2019, 2020), automatas (Pesant 2004; Quimper and Walsh 2006; Cheng, Xia, and Yap 2012), context-free grammars (Sellmann 2006; Quimper and Walsh 2006, 2007; Kadioglu and Sellmann 2008), decision diagrams (Cheng and Yap 2010; Gange, Stuckey, and Szymanek 2011; Perez and Régin 2014; Vion and Piechowiak 2018; Verhaeghe, Lecoutre, and Schaus 2018, 2019), and binary constraint trees (Wang and Yap 2022b). Ad-hoc constraints are very useful for modelling specific constraints which do not fit well with common global constraints.

Many algorithms (Yap, Xia, and Wang 2020) have been proposed to efficiently enforce Generalized Arc Consistency (GAC) on ad-hoc constraints. However, there is less work studying the constraint expressive power that can be useful for modelling CSPs, such as the constraint succinctness and the cost of operations/queries on constraints. In this paper, we investigate ad-hoc constraints with a knowledge compilation (KC) approach which has been mainly developed to study Boolean functions (Darwiche and Marquis 2002). Note that although we use a KC approach, we are concerned about questions arising from modelling CSPs which do not normally occur in the KC literature.

To solve a problem using CSP solvers, one needs to model the problem as constraints, where ad-hoc constraints provide the most modelling flexibility. However, there are many different choices for the ad-hoc constraints and their specific representation. A basic question we address is when is an ad-hoc constraint defined with a certain representation worse than another one. For this, we first use 15 families of constraints to analyze 14 existing ad-hoc constraints, showing ad-hoc constraints have very different properties, e.g. some can model a constraint family in polysize while others cannot. Going further, we give a complete map of succinctness for the 14 ad-hoc constraints.

Besides primitive constraints, constraint modelling languages (Nethercote et al. 2007; Frisch et al. 2008) also provide a higher level of modelling, such as the use of logical operators to combine constraints. We study whether the ad-hoc constraints can provide tractable conjunction, disjunction, negation, projection and conditioning operations. This can give guidance on how to model combinations of constraints in a CSP solver supporting certain ad-hoc constraints taking into account tractability considerations.

In addition, we may want to ask various queries on the constraints in a CSP model. For instance, we may want to count the solutions/tuples of a constraint (Pesant 2005) or check whether some constraints are equivalent. We also study whether the ad-hoc constraints provide tractable consistency, entailment, validity, implicant, equivalence, counting and model enumeration queries.

In summary, we present a KC approach to investigate the succinctness of ad-hoc constraints and associated operations/queries. Our results deepen our understanding of the fundamental properties of ad-hoc constraints. It also helps to give insights into when a certain ad-hoc constraint can be used given the context of its usage.

Preliminaries

A CSP P is a pair (X, C) where X is a set of variables, $\mathcal{D}(x)$ is the domain of a variable x, and C is a set of constraints. A *literal* of a variable x is a pair (x, a). A *tuple* over variables $\{x_{i_1}, x_{i_2}, \dots, x_{i_r}\}$ is a set of literals $\{(x_{i_1}, a_1), (x_{i_2}, a_2), \dots, (x_{i_r}, a_r)\}$. A tuple τ is an *assignment* if $a \in \mathcal{D}(x)$ for all $(x, a) \in \tau$. Then $U(S)$ denotes the set of all assignments over variables S. Each constraint c has a constraint scope $scp(c) \subseteq X$ and a constraint relation $rel(c) \subseteq U(scp(c))$. The arity of a constraint c is $|scp(c)|$. c is a *binary constraint* if $|scp(c)| = 2$. A CSP is called a *binary CSP* if the largest constraint arity is 2. A binary CSP is *normalized* if all constraints have different scopes.

Given any variables V and literals τ, we use $\tau[V] = \{(x, a) \in \tau | x \in V\}$ to denote a subset of τ, and $T[V] = \{\tau[V] | \tau \in T\}$ is the *projection* of the tuples T on V. An assignment τ over X is a solution of P if $\tau[scp(c)] \in rel(c)$ for all $c \in C$. $sol(X, C)$ or $sol(P)$ denotes the solutions of P, and $sol(C) = sol(\bigcup_{c \in C} scp(c), C)$.

Ad-Hoc Constraints

Many representations R can be used to encode a set $rel(R)$ of tuples over some variables $scp(R)$, and R is regarded as a kind of ad-hoc constraints (generic constraints) if $rel(R)$ can have arbitrary tuples over $scp(R)$. The *size of the variables* $scp(R)$ is defined as $size(scp(R)) = \sum_{x \in scp(R)}(1 + |\mathcal{D}(x)|)$. We introduce 14 existing ad-hoc constraints defined with different representations which come with efficient GAC propagators. We classified them into 4 categories.

Table and Non-ordinary Tables. A *table* R over variables X is a set of tuples over X, we also call this *ordinary tuples*. In addition, many generalizations of tuples have been proposed, which we call *non-ordinary tuples over* X. Table 1 summarises the non-ordinary tuples used in the non-ordinary tables: c-table (c-T), short table (shoT), smart table (smaT), basic smart table (bsmaT) and segmented table (segT).

The *c-T* (Katsirelos and Walsh 2007; Xia and Yap 2013), *shoT* (Jefferson and Nightingale 2013), *sliT* (Gharbi et al. 2014), *smaT* (Mairy, Deville, and Lecoutre 2015), *bsmaT* (Verhaeghe et al. 2017) and *segT* (Audemard, Lecoutre, and Maamar 2020) over variables X are respectively defined as a set of c-tuples, short supports, entries, smart tuples, basic smart tuples and segmented tuples X. These various representations R all encode the tuples $rel(R) = \bigcup_{t \in R} rel(t)$.

The *size of a table or a non-ordinary table constraint* R is the sum of $size(scp(R))$ and the number of literals, unary equalities, unary tautologies, and comparisons used in R.

Example 1. The constraint $\bigvee_{k=1}^{3} \bigwedge_{i=0}^{2} x_i = y_{(i+k)\%3}$ can be modelled as a smaT R over 6 variables $\{x_0, x_1, x_2, y_0, y_1, y_2\}$, where R consists of 3 smart tuples $\{x_0 = y_0, x_1 = y_1, x_2 = y_2\}$ and $\{x_0 = y_1, x_1 = y_2, x_2 = y_0\}$ and $\{x_0 = y_2, x_1 = y_0, x_2 = y_1\}$. Each smart tuple includes 3 binary constraints of which the constraint scopes do not share any variables. Correspondingly, these smart tuples can also be encoded as segmented tuples by encoding each binary constraint in the smart tuples as a binary table constraint.

Tuple	Possible constraints in a tuple	Structure
entry	unary equalities and a table	separated
short support	unary equalities and unary tautologies	separated
c-tuple	unary tables	separated
segmented tuple	tables, unary equalities and unary tautologies	separated
basic smart tuple	unary comparisons, unary tables, unary equalities and unary tautologies	separated
smart tuple	comparisons, unary tables, unary equalities and unary tautologies	acyclic

Table 1: Non-ordinary tuples t over variables X: t is a set of constraints and X=$\bigcup_{c \in t} scp(c)$ and t encodes tuples $rel(t)$= $sol(t)$. "separated" means $scp(c_i) \cap scp(c_j)$=$\emptyset$ for all $c_i, c_j \in t$ such that $c_i \neq c_j$. "acyclic" means the constraint graph of (X, t) is acyclic. Then $x = a$, $x = *$ (and $x - y \triangleright a$, $x \triangleright a$) are *unary equality*, *tautology* (and *comparisons*) relations, where $x, y \in X$ and a is a constant and $\triangleright \in \{=, \neq, <, >, \leq, \geq\}$ and $rel(x = *)$ is $\{\{(x, b)\} | b \in \mathcal{D}(x)\}$.

Decision Diagrams. We study 3 kinds of decision diagrams w.r.t. an order O over a set of r variables X, i.e. OMVD (Amilhastre et al. 2014), OMDD (Srinivasan et al. 1990) and sMDD (Verhaeghe, Lecoutre, and Schaus 2018):

- An *Ordered Multi-valued Variable Diagram (OMVD)* is a directed acyclic graph which has $r + 1$ layers $\{L_1, \cdots, L_{r+1}\}$ of nodes and a root node in L_1 and a terminal node in L_{r+1} such that each arc pointing from a node in L_i to a node in L_{i+1} is labelled with a value in $\mathcal{D}(O_i)$ for $1 \leq i \leq r$. $out(v, a)$ ($in(v, a)$) denotes the arcs labelled with a which point from v (point to v).
- An *Ordered Multi-valued Decision Diagram (OMDD)* is an OMVD such that $|out(v, a)| \leq 1$ for any $1 \leq i \leq r$ and a value $a \in \mathcal{D}(O_i)$ and a node v in L_i.
- A *semi-OMDD (sMDD)* is an OMVD such that for any $1 \leq i \leq \lfloor \frac{r}{2} \rfloor$ ($\lfloor \frac{r}{2} \rfloor + 2 < i \leq r + 1$) and $a \in \mathcal{D}(O_i)$ and a node v in L_i, $|out(v, a)| \leq 1$ ($|in(v, a)| \leq 1$).

where O_i denotes the i^{th} variable in O and every path from root to the terminal node corresponds to a tuple over X. We remark that OMVD is also called non-deterministic OMDD. A representation R w.r.t. an order O_1 *uses an order* O_2 if O_1 is a subsequence of O_2.

Following (Amilhastre et al. 2014), the *size of an OMVD constraint* R is defined as the sum of $size(scp(R))$ and the number of edges and nodes in R.

In this paper, OMVD does not have any *jumping arcs* between nodes in 2 non-consecutive layers. Note that w.l.o.g. jumping arcs can be removed by adding polysize new arcs.

Example 2. Figure 1 gives two decision diagrams encoding $x_1 = x_2 \wedge h_1 = a_{12}$ w.r.t. the variable order $h_1 < x_1 < x_2 < x_3$ with $\mathcal{D}(h_1) = \{a_{12}\}$ and $\mathcal{D}(x_i) = \{1, 2, 3\}$. Figure 1(a) is an OMDD. For each node v in the OMDD and a value a, $|out(v, a)| \leq 1$. Then Figure 1(b) is a sMDD. sMDD can be regarded as the combination of an OMDD and the transpose of another OMDD. We can see that for each node v in the layers L_1, L_2, the arcs pointing from v have different labels, and for each node v in the L_5 layer, the arcs pointing to v have different labels.

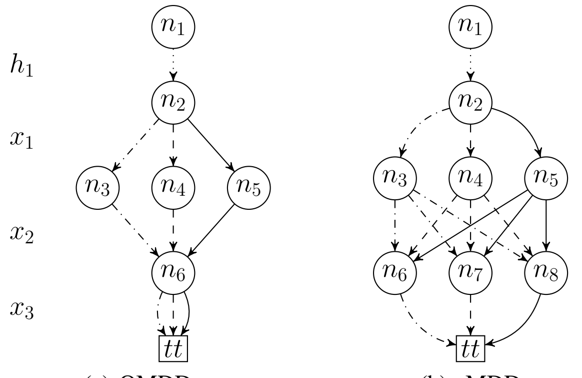

Figure 1: An OMDD and sMDD, where the dash-dotted (dashed, solid, dotted) lines denote the values 1 (2, 3, a_{12}).

Binary Constraint Tree. A *Binary Constraint Tree* (BCT) (Wang and Yap 2022b) is a normalized binary CSP which has a tree structured constraint graph (Dechter 1987, 1990a). It is well known that BCT can be solved by Arc Consistency algorithms (Freuder 1982).

A BCT (V, C) can also be used to model the constraints R over variables $scp(R)$ such that $scp(R)$ is a subset of V and the constraint relation $rel(R)$ is equal to $sol(V, C)[scp(R)]$ where the variables in $V \setminus scp(R)$ and $scp(R)$ are respectively called *hidden* variables and *original* variables. The *size of a BCT* (V, C) is the sum of $size(V)$ and the sizes of binary constraints in C, where binary constraint size follows table constraint size.

A BCT $R = (V, C)$ over variables $scp(R)$ *uses a graph* $G = (H \cup Y, E)$, where G is a tree and the vertices in Y are variables, if the variables $scp(R)$ is a subset of the leaves Y in G,[1] and there is a BCT $G(R)$=$(G(V), G(C))$ encoding $rel(R)$ and a bijective function f^R from the vertices $H \cup Y$ and edges E in G to the variables $G(V)$ and binary constraints $G(C)$ in $G(R)$ such that

- $V \subseteq G(V)$, $Y \subseteq G(V)$, $C \subseteq G(C)$, and
- $rel(c) = U(scp(c))$ for all $c \in G(C) \setminus C$, and
- every v in $H \cup Y$ is mapped to a variable $f^R(v) \in G(V)$ with $f^R(v) = v$ for all $v \in Y$, and
- each e in E is mapped to a constraint $f^R(e) \in G(C)$ with $scp(f^R(e)) = \{f^R(v) | v \in e\}$,

We remark that a BCT R over variables $scp(R)$ denotes a constraint over $scp(R)$ encoded by a BCT.

Example 3. The constraint $x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3 \wedge x_2 \neq x_4 \wedge x_3 \neq x_4$, where $\mathcal{D}(x_1)$=$\mathcal{D}(x_4)$={1,2,3} and $\mathcal{D}(x_2)$=$\mathcal{D}(x_3)$={1,2}, can be encoded as a BCT ($\{x_1, x_2, x_3, x_4, h\}$, $\{c_1, c_2, c_3, c_4\}$) such that $scp(c_i) = \{h, x_i\}$, $\mathcal{D}(h)$ = {1,2}, $rel(c_1)$ = $\{t^1_{13}, t^1_{23}\}$, $rel(c_2)$ = $\{t^2_{11}, t^2_{22}\}$, $rel(c_3)$ = $\{t^3_{12}, t^3_{21}\}$ and $rel(c_4)$ = $\{t^4_{13}, t^4_{23}\}$, where t^i_{jk} = {$(h$,$j)$, $(x_i$,$k)$}, and h is the only hidden variable in the BCT.

[1]Note that any non-leaf variable x in a BCT can be encoded as a leaf by replacing x with a copy x' and adding an equality $x = x'$.

Grammars. The strings with a length r in a language can also be used to model a constraint over r variables. Formally, a constraint c can be encoded as a language L w.r.t. an order O over r variables $scp(c)$ such that $\{(O_1, a_1),\cdots, (O_r, a_r)\}$ is in $rel(c)$ iff $a_1 \cdots a_r$ is a string in L. As the order is significant, a constraint can be encoded as different languages depending on the order.

A *context free language (CFL)* is generated by a *context free grammar (CFG)* consisting of terminals, non-terminals, productions and a start non-terminal. The *size of a CFG constraint* R is defined as the sum of $size(scp(R))$ and the number of terminals, non-terminals, productions used in R.

A CFG R w.r.t. O can be encoded in Chomsky normal form (Lange and Leiß 2009) which can be encoded into an acyclic CFG $\mathcal{A}^R$ (Katsirelos, Narodytska, and Walsh 2009) such that $rel(\mathcal{A}^R) = rel(R)$, the right hand site (RHS) of each production in $\mathcal{A}^R$ has 2 non-terminals or 1 terminal, and each non-terminal α in $\mathcal{A}^R$ encodes the tuples $rel(\alpha)$ over variables $scp(\alpha) \subseteq scp(R)$ where $scp(\alpha) \cap scp(\beta) = \emptyset$ if α, β are in the same RHS of a production in $\mathcal{A}^R$.

Deterministic finite state automata (DFA) (Pesant 2004) and non-deterministic finite state automata (NFA) (Quimper and Walsh 2006) are two subsets of CFG, where any OMVD (OMDD) constraint can be expressed as a NFA (DFA) constraint, while the NFA (DFA) constraint can be directly expanded into the OMVD (OMDD) constraint using the same order (Amilhastre et al. 2014).

Example 4. The CFG ({0,1, #}, $\{S_0, S_1\}$, $\{S_0 \rightarrow S_1 \# S_1$, $S_1 \rightarrow 0S_10$, $S_1 \rightarrow 1S_11$, $S_1 \rightarrow \epsilon\}$, S_0) generates the language $L = \{aa^R \# bb^R | a, b \in \{0, 1\}*\} \subseteq \{0, 1, \#\}*$, where a^R and b^R are the reverse of a and b, {0,1, #} and $\{S_0, S_1\}$ are the terminals and non-terminals, and S_0 is the start non-terminal. The constraint, which is encoded as L w.r.t. an order $x_1 < x_2 < x_3$, can also be encoded into an acyclic CFG ({0,1,#}, $\{S_3, \cdots, S_{13}\}$, $\{S_3 \rightarrow S_4S_5$, $S_3 \rightarrow S_6S_7$, $S_5 \rightarrow \#$, $S_6 \rightarrow \#$, $S_4 \rightarrow S_8S_{10}$, $S_4 \rightarrow S_9S_{11}$, $S_7 \rightarrow S_{10}S_{12}$, $S_7 \rightarrow S_{11}S_{13}$, $S_8 \rightarrow 0$, $S_9 \rightarrow 1$, $S_{10} \rightarrow 0$, $S_{11} \rightarrow 1$, $S_{12} \rightarrow 0$, $S_{13} \rightarrow 1\}$, S_3). The non-terminals in an acyclic CFG encode tuples over different variables, e.g. the non-terminals S_8, S_9 encode tuples over the variable x_1, and the non-terminals S_{10}, S_{11} (S_{12}, S_{13}) encode tuples over the variable x_2 (variable x_3).

Constraint Families (Counterexamples)

We first investigate the modelling power of the ad-hoc constraints on 15 families of constraints given in Table 2. These families being ad-hoc constraints, can be essentially arbitrary, and have been chosen as counterexamples to prove various properties of the ad-hoc constraints. Furthermore, many of these families are hard to model with common global constraints. Some families also correspond to well known constraints or variants thereof. The results given in Figure 2 show that the modelling powers of the ad-hoc constraints are significantly different.

We summarize some of the families given in Table 2. $\mathbf{F}_1$ expresses a permutation constraint (Puget 1998). $\mathbf{F}_2$ is the negation of $\mathbf{F}_1$. Projecting variable h_1 from $\mathbf{F}_3$ gets $\mathbf{F}_2$. Projecting variables $h_2, y_1, \cdots, y_n$ from $\mathbf{F}_4$ and $\mathbf{F}_5$ gets $\mathbf{F}_3$.

$\mathbf{F}_1$	$\bigwedge_{i=1}^{n} \bigwedge_{j=i+1}^{n} x_i \neq x_j$ where $\mathcal{D}(x_i) = \{1, \cdots, n\}$
$\mathbf{F}_2$	$\bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} x_i = x_j$ where $\mathcal{D}(x_i) = \{1, \cdots, n\}$
$\mathbf{F}_3$	$\bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} x_i = x_j \wedge h_1 = a_{ij}$ $\mathcal{D}(h_1) = \{a_{ij} \| 1 \leq i < j \leq n\}$ and $\mathcal{D}(x_i) = \{1, \cdots, n\}$
$\mathbf{F}_4$	$(\bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} x_i = x_j \wedge h_1 = a_{ij}) \wedge (\bigwedge_{k=1}^{n} y_k = 1)$ $\mathcal{D}(h_1) = \{a_{ij} \| 1 \leq i < j \leq n\}$ $\mathcal{D}(x_i) = \{1, \cdots, n\}$ and $\mathcal{D}(y_k) = \{1\}$
$\mathbf{F}_5$	$\bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} x_i = x_j \wedge h_1 = a_{ij} \wedge h_2 = a_{ij}$ $\mathcal{D}(h_1), \mathcal{D}(h_2)$ are $\{a_{ij} \| 1 \leq i < j \leq n\}$ $\mathcal{D}(x_i) = \{1, \cdots, n\}$
$\mathbf{F}_6$	$\bigwedge_{i=1}^{n} x_i = 1 \vee y_i = 1 \vee z_i = 1$ where domains are $\{0, 1\}$
$\mathbf{F}_7$	$\bigwedge_{c \in F} h_c - x \neq -1{+}l_x \wedge h_c - y \neq 1{+}l_y \wedge h_c - z \neq 3{+}l_z$ c is a clause $(x = l_x) \vee (y = l_y) \vee (z = l_z)$ in a 3-SAT F $l_x, l_y, l_z \in \{0, 1\}$ $\mathcal{D}(h_c) = \{0, 2, 4\}$ and $\mathcal{D}(x), \mathcal{D}(y), \mathcal{D}(z)$ are $\{0, 1\}$
$\mathbf{F}_8$	$\bigwedge_{i=2}^{n} x_i \neq x_{i-1}$ where $\mathcal{D}(x_i) = \{1, 2, 3\}$
$\mathbf{F}_9$	$\bigvee_{O \in \mathcal{O}} \mathcal{F}(O)$ where $\mathcal{O}$ denotes all orders over $\{x_1, \cdots, x_{2n}\}$ $\mathcal{F}(O) = (\bigwedge_{i=1}^{n} O_i = O_{(i+n)}) \wedge v = a_O$ $\mathcal{D}(v) = \{a_O \| O \in \mathcal{O}\}$ and $\mathcal{D}(x_i)$ is $\{1, \cdots, n\}$
$\mathbf{F}_{10}$	$\bigwedge_{i=1}^{n} x_i \neq 0$ where variable domains are $\{0, 1, 2\}$
$\mathbf{F}_{11}$	$(\bigwedge_{i=1}^{n} x_i = 0) \vee (\bigwedge_{i=1}^{n} y_i = 0)$ where domains are $\{0, 1\}$
$\mathbf{F}_{12}$	$\bigvee_{i=1}^{n} x_i \neq 0$ where variable domains are $\{0, 1\}$
$\mathbf{F}_{13}$	Disjunctive Normal Form (DNF)
$\mathbf{F}_{14}$	CSPs with a treewidth bounded by a fixed k
$\mathbf{F}_{15}$	$\bigvee_{k=0}^{l} x_{2k+1} = \# \wedge S_1^{2k} \wedge S_{2k+2}^{n}$ where domain is $\{\#, 0, 1\}$ $S_j^u = \bigwedge_{i=j}^{\lfloor \frac{j+u}{2} \rfloor} x_i = x_{j+u-i} \wedge x_i \in \{0, 1\}$ and $n = 2l + 1$

Table 2: Families of constraints ($\mathbf{F}_i$).

The family $\mathbf{F}_{14}$ is the set of CSPs (X, C) with a treewidth bounded by a fixed integer k, where the arity of all constraints in C are also bounded by k. $\mathbf{F}_{15}$ models the language discussed in Example 4 over n variables.

$\mathbf{F}_7$ is used to model the 3-SAT problem F. Each clause c in F, defined with $(x = l_x) \vee (y = l_y) \vee (z = l_z)$, is modelled as the constraint c': $(h_c - x \neq -1 + l_x) \wedge (h_c - y \neq 1 + l_y) \wedge (h_c - z \neq 3 + l_z)$, where $l_x, l_y, l_z \in \{0, 1\}$ and $rel(c) = rel(c')[\{x, y, z\}]$. Then $sol(\mathbf{F}_7)[X]$=$sol(F)$ and F is satisfiable iff $\mathbf{F}_7$ is satisfiable, so $\mathbf{F}_7$ is NP-hard.

In addition, in Figure 2, a family F can be encoded as an ad-hoc constraint R in polysize if the size of R can be polynomial in the sum of $size(scp(R))$ and the length of the expression defining F, where the expression length is polynomial in the number of bounded arity constraints used by the expression. For example, the expression defining the family $\mathbf{F}_1$ includes $n(n-1)$ binary inequalities.

Theorem 1. The results in Figure 2 hold.

The results of the green cells in Figure 2 are straightforward. The results of red cells can be verified by using operations on the ad-hoc constraints, see Section 9. The results of the cells not colored can be verified by using of the arcs in Figure 3. We now prove the results of the blue cells.

In (Moshier and Rounds 1987), they show there is no polysize CFG encoding all permutations of n values. For any variable order, the constraint relation of $\mathbf{F}_1$ corresponds to all permutations of n values, so there is no polysize CFG encoding $\mathbf{F}_1$. Propositions 1, 2, 3 show $\mathbf{F}_3$, $\mathbf{F}_6$, $\mathbf{F}_8$ cannot be respectively encoded as polysize sMDD, smaT, segT.

	$\mathbf{F}_1$	$\mathbf{F}_2$	$\mathbf{F}_3$	$\mathbf{F}_4$	$\mathbf{F}_5$	$\mathbf{F}_6$	$\mathbf{F}_7$	$\mathbf{F}_8$	$\mathbf{F}_9$	$\mathbf{F}_{10}$	$\mathbf{F}_{11}$	$\mathbf{F}_{12}$	$\mathbf{F}_{13}$	$\mathbf{F}_{14}$	$\mathbf{F}_{15}$
Table	●	●	●	●	●	●	○	●	●	●	●	●	●	●	●
shoT	●	✓	✓	✓	✓	●	○	●	●	●	✓	✓	✓	●	●
c-T	●	✓	✓	✓	✓	●	○	●	●	✓	✓	✓	✓	●	●
segT	?	✓	✓	✓	✓	✓	○	●	✓	✓	✓	✓	✓	●	✓
smaT	?	✓	✓	✓	✓	●	○	✓	✓	✓	✓	✓	✓	●	✓
sMDD	●	●	●	✓	✓	✓	○	✓	●	✓	✓	✓	●	●	●
OMDD	●	●	✓	✓	✓	✓	○	✓	●	✓	✓	✓	●	●	●
OMVD	●	✓	✓	✓	✓	✓	○	✓	●	✓	✓	✓	✓	●	●
BCT	●	✓	✓	✓	✓	✓	○	✓	●	✓	✓	✓	✓	✓	●
CFG	●	✓	✓	✓	✓	✓	○	✓	●	✓	✓	✓	✓	✓	✓

Figure 2: Expressive power: ✓(●) means a family can (cannot) be encoded in polysize and ○ means a family is NP-hard.

The family $\mathbf{F}_3$ can be encoded as a polysize OMDD w.r.t. the order $h_1 < x_1 < \cdots < x_n$. Then $\mathbf{F}_4$ and $\mathbf{F}_5$ can be encoded as polysize sMDD using the order $h_1 < x_1 < \cdots < x_n < h_2 < y_1 < \cdots < y_n$. The family $\mathbf{F}_7$ can be used to model the 3-SAT problem, so $\mathbf{F}_7$ is NP-hard. Correspondingly, it is NP-hard to encode $\mathbf{F}_7$ as the representations in polysize, since they have polytime GAC propagators which can be used to check whether a constraint relation is empty. This result will be used to prove the NP-hardness of the conjunction operations on various representations (see the results given in Figure 4).

Proposition 6 shows that there is no CFG encoding $\mathbf{F}_9$ in polysize. Then (Amilhastre et al. 2014) shows $\mathbf{F}_{13}$ cannot be encoded as polysize OMDDs, since an OMDD over Boolean variables can be regarded as an Ordered Binary Decision Diagram (OBDD) and DNF cannot be encoded as polysize OBDD (Darwiche and Marquis 2002).

The family $\mathbf{F}_{14}$ can be encoded as polysize BCT (see Proposition 4) but not polysize OMVD (see Proposition 5). $\mathbf{F}_{15}$ is the family given by Theorem 6 in (Kadioglu and Sellmann 2010) which can be encoded as a polysize CFG but not a BCT (see Proposition 7). The families $\mathbf{F}_6$,$\mathbf{F}_8$ can be regarded as special cases of $\mathbf{F}_{14}$, so the family $\mathbf{F}_{14}$ cannot be encoded as polysize segT and smaT.

Proposition 1. There is no polysize sMDD encoding $\mathbf{F}_3$.

Proof. Let $n > 10$ and R be a sMDD encoding the family $\mathbf{F}_3$ w.r.t. an order O over the variables $\{h_1, x_1, \cdots, x_n\}$ and h_1 is the k^{th} variable by the order O.

Assume $k \leq \lfloor \frac{n}{2} \rfloor + 1$ and V is the set of variables $\{O_i | \lfloor \frac{n}{2} \rfloor + 2 \leq i \leq n\}$. Given any subset S of V. $\tau(S)$ denotes the tuple $\{(x_i, 1) | x_i \in S\} \cup \{(x_i, 2) | x_i \in V \setminus S\}$, and $A(S)$ denotes the set $\{(h_1, a_{ij}) | \; \{x_i, x_j\} \subseteq V, |\{x_i, x_j\} \cap S| = 1\}$ which includes the literals l of h_1 such that $\tau(S) \cup \{l\}$ is not in $rel(R)[V \cup \{h_1\}]$. In addition, there must be exactly 1 node $n(S)$ in the $(\lfloor \frac{n}{2} \rfloor + 2)^{th}$ layer of R such that $\tau(S)$ is encoded by a path from $n(S)$ to the terminal node, since R is a sMDD. For any subsets S_1, S_2 of V, if S_1 is not equal to S_2, then $A(S_1) \neq A(S_2)$ and the nodes $n(S_1)$ and $n(S_2)$ are different. Then V has at least $2^{\lfloor \frac{n}{2} \rfloor - 1}$ different subsets, hence, R has at least $2^{\lfloor \frac{n}{2} \rfloor - 1}$ nodes.

Similarly, for $k \geq \lfloor \frac{n}{2} \rfloor + 1$, R also has at least $2^{\lfloor \frac{n}{2} \rfloor - 1}$ nodes. So there is no polysize sMDD encoding $\mathbf{F}_3$. $\square$

Proposition 2. There is no polysize smaT encoding $\mathbf{F}_6$.

Proof. Let R be a smaT encoding $\mathbf{F}_6$. For any smart tuple τ in R and $1 \leq i \leq n$, τ must include a binary constraint or unary constraint which makes sure that all variables in $\{x_i, y_i, z_i\}$ cannot be assigned with 0 at the same time, where the constraint relation of a binary constraint or unary constraint has at most $6 = 2^3 - 2$ tuples, thus, the smart tuple τ encodes at most 6^n tuples. The clauses used in $\mathbf{F}_6$ do not share any variables, therefore, $|rel(R)| = 7^n$ and the number of smart tuples in R is at least $(\frac{7}{6})^n$. So there is no polysize smaT encoding the family $\mathbf{F}_6$. $\square$

Proposition 3. There is no polysize segT encoding $\mathbf{F}_8$.

Proof. Assume $n > 10$ and R is a segT encoding $\mathbf{F}_8$ and t is a segmented tuple in R, where $|rel(R)| = 3(2^{n-1})$ and $|rel(t)| \neq \emptyset$. Let $X(t)$ be the set of variables x_i such that x_i, x_{i+1} are included by different constraints in t.

For any $x_i \in X(t)$ and $a \in \{1, 2, 3\}$, if $(x_i, a) \in rel(t)[\{x_i\}]$, then $(x_{i+1}, a) \notin rel(t)[\{x_{i+1}\}]$. So there are at least $\lfloor \frac{|X(t)|}{2} \rfloor$ variables x_i such that $rel(t)[\{x_i\}] \leq 1$. For any nonempty $V \subseteq scp(R)$ and a tuple τ over V, the size of $\{\tau_1 \in rel(R) | \tau \subseteq \tau_1\}$ is $2^{n-|V|}$.

If $|t| > \lfloor \sqrt{n} \rfloor$, then $|X(t)| \geq \lfloor \sqrt{n} \rfloor$ and $|rel(t)| \leq k = 2^{n - \lfloor \frac{\sqrt{n}}{2} \rfloor}$. If $|t| \leq \lfloor \sqrt{n} \rfloor$, there is a constraint c in t and a polynomial p such that $|scp(c)| \geq \lfloor \sqrt{n} \rfloor$ and $|rel(c)| \leq p$ and for any $\tau \in rel(c)$, $|\{\tau_1 \in rel(t) | \tau \subseteq \tau_1\}| \leq 2^{n - \lfloor \sqrt{n} \rfloor}$, so $|rel(t)| \leq pk$. So $p|R| \geq \frac{3(2^{n-1})}{k} \geq 2^{\lfloor \frac{\sqrt{n}}{2} \rfloor}$ and there is no polysize segT encoding $\mathbf{F}_8$. $\square$

Proposition 4. A k treewidth CSP $P = (X, C)$ with domain size d can be encoded as a BCT (V, C') such that $|V| \leq 2|X|$ and $|\mathcal{D}(v)| \leq d^{k+1}$ for all $v \in V$.

Proof. (Dechter and Pearl 1989) shows that there is a BCT P_1=(H, C_1) such that $rel(P)$=$\{(\bigcup_{(h,\tau) \in t} \tau) | t \in rel(P_1)\}$ where $|H| \leq |X|$ and for any $h \in H$, existing $var(h) \subseteq X$ such that $\mathcal{D}(h) \subseteq U(var(h))$ and $|var(h)| \leq k + 1$, i.e. $|\mathcal{D}(h)| \leq d^{k+1}$. Then $rel(P)$ can be encoded as a BCT (V, C') where $V = X \cup H$ and $C' = C_1 \cup \{c_x | x \in X\}$ and c_x is between x and a variable $h \in H$ such that $x \in var(h)$ and $rel(c_x) = \{\{(x, a), (h, \tau) | \tau \in \mathcal{D}(h), (x, a) \in \tau\}\}$. So P can be encoded as a BCT (V, C') such that $|V| \leq 2|X|$ and $|\mathcal{D}(v)| \leq d^{k+1}$ for all $v \in V$. $\square$

Proposition 5. There is no polysize OMVD encoding $\mathbf{F}_{14}$.

Proof. Let F be the CSPs over Boolean variables X such that $|X| \geq 2^w$ where w is the treewidth of the CSPs. From Proposition 4, assume F' is the set of polysize BCTs encoding F. The treewidth of a BCT is 1, so F' is a subset of $\mathbf{F}_{14}$

Theorem 5 in (Razgon 2016) shows there is no polysize NROBP (nondeterministic read-once branching program) encoding F. So there is no polysize OMVD R encoding F' and $\mathbf{F}_{14}$, otherwise removing the hidden variables labels from the arcs in R gets a polysize NROBP encoding F (this is impossible), where an OMVD over Boolean variables with some unlabeled arcs can be regarded as a NROBP. So there is no polysize OMVD encoding $\mathbf{F}_{14}$. $\square$

Proposition 6. There is no polysize CFG encoding $\mathbf{F}_9$.

Proof. Assume R is a CFG w.r.t. any order O over $\{v\} \cup X$ encoding $\mathbf{F}_9$ and O' is a subsequence of O over X, where $X = \{x_1, \cdots, x_{2n}\}$. The interchange Lemma (Ogden, Ross, and Winklmann 1985) shows that there is $T \subseteq sol(\mathcal{F}(O'))$ and $W = \{O_k | 1 \leq k < i\}$ and $V = \{O_k | i \leq k < j\}$ and $U = \{O_k | j \leq k \leq 2n + 1\}$ such that (i) $|T| \geq \frac{n^n}{c_R(2n+1)^2}$ and $\frac{n}{2} \leq j - i < n$, where c_R is the number of non-terminals used in the acyclic CFG $\mathcal{A}^R$; and (ii) for any $\tau_1, \tau_2 \in T$, the tuples $\tau_1[W] \cup \tau_2[V] \cup \tau_1[U]$ and $\tau_2[W] \cup \tau_1[V] \cup \tau_2[U]$ are also in the constraint relation of $\mathbf{F}_9$. If $v = a_{O'}$, then every variable in $V \cap X$ equals to a variable in $W \cup U$, therefore, $\tau_1[V]$ is equal to $\tau_2[V]$. For all tuples τ in T, $\tau[V]$ is common, thus, $|T| \leq n^{\frac{3n}{2}+1}$ and $c_R(2n + 1)^2 \geq n^{\frac{n}{2}-1}$. c_R is not polynomial in the length $O(n2^{2n})$ of the expression defining $\mathbf{F}_9$, so there is no polysize CFG encoding $\mathbf{F}_9$. $\square$

Lemma 1. For any BCT $R = (V, C)$ and variables $X \subseteq V$, there is $v \in V$ such that deleting v partitions the constraint graph of R into connected components (CC) $com(v)$ where each CC in $com(v)$ has at most $\frac{|X|}{2}$ variables in X.

Proof. Let $mv(v)$ be the CC in $com(v)$ having the most variables in X and $next(v)$ be the variable in $mv(v)$ connecting to v. For any $v \in V$, if both $mv(v)$ and $mv(next(v))$ have more than $\frac{|X|}{2}$ variables in X, then $next(next(v))$ is not equal to v, since v connects to less than $\frac{|X|}{2}$ variables in X without passing $next(v)$. If $mv(v)$ has more than $\frac{|X|}{2}$ variables in X for all $v \in V$, then R has infinite variables $\{v, next(v), next(next(v)) \cdots\}$ due to R is acyclic, so there is $v \in V$ such that each CC in $S(v)$ has at most $\frac{n}{2}$ variables in X. $\square$

Proposition 7. There is no polysize BCT encoding $\mathbf{F}_{15}$.

Proof. Let $R = (V, C)$ be a BCT encoding the family $\mathbf{F}_{15}$. Based on Lemma 1, there must be $v \in V$ such that deleting v partitions the constraint graph of R into connected components (CC) $com(v)$ of which each CC has at most $\frac{l}{2}$ variables x_j where j is an even integer.

$\{x_i = x_j | 1 \leq i, j \leq n, i \text{ is odd}, j \text{ is even}\}$ is the set of $l(l + 1)$ equalities used in the family $\mathbf{F}_{15}$. For each variable x_i with an odd i, there are at most $\frac{l}{2}$ variables x_j with an een j such that x_i, x_j are in the same CC. Therefore, there are at least $\frac{(l+1)l}{2} = (l+1)(l - \lfloor \frac{l}{2} \rfloor)$ pairs x_i, x_j with an odd i and an even j such that x_i, x_j are in different CC. Then based on the pigeonhole principle, there is $0 \leq k \leq l$ such that S_1^{2k}, S_{2k+2}^{n} use at least $\frac{l}{2}$ equalities E where x_i, x_j are in different CC in $com(v)$ for all $x_i = x_j$ in E. The equalities in E do not share any variables.

For any tuples $\tau_1, \tau_2 \in sol(R)$ including $(x_{2k+1}, \#)$ and any equality $x_i = x_j$ in E, if $\tau_1[\{x_i\}] \neq \tau_2[\{x_i\}]$, then $\tau_1[\{v\}] \neq \tau_2[\{v\}]$, since the tuple $(\tau_1 \setminus \tau_1[W]) \cup \tau_2[W]$ is not in $sol(R)$, where W is the set of variables of a CC in $com(v)$ such that $x_j \in W$. So $|\mathcal{D}(v)| \geq 2^{\frac{l}{2}}$ and there is no polysize BCT encoding $\mathbf{F}_{15}$. $\square$

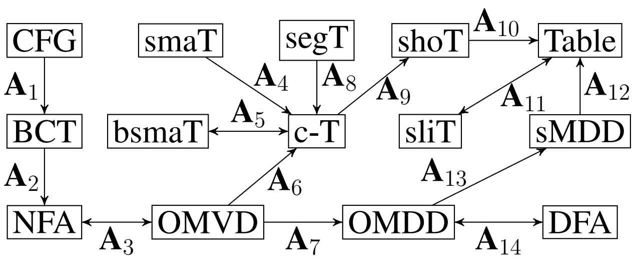

Figure 3: Succinctness of the 14 representations. An arc from A to B means $A \leq B$. If there is a path from A to B, then $A \leq B$, otherwise $A \nleq B$.

Succinctness of Ad-Hoc Constraints

We use *succinctness* (Darwiche and Marquis 2002) to measure the compactness of various ad-hoc constraints. The notation $A \leq B$ denotes a representation A is *at least as succinct as* another representation B.[2] It means there is no super-polynomial blow-up when encoding B into A but it does not mean that A is always smaller than B.

Figure 3 shows the succinctness of the 14 representations (ad-hoc constraints) studied—each arc from a node A to another node B in the figure denotes A is at least as succinct as B. The arcs in Figure 3 make it easy to see when a representation may be used such as identifying which ad-hoc constraints may be more suitable to model a constraint. For example, $\mathbf{F}_{15}$ cannot be encoded as a polysize BCT, and BCT $\leq$ OMVD and BCT $\leq$ c-T, so $\mathbf{F}_{15}$ cannot also be encoded as a polysize OMVD and c-T. Consider also the constraints c with a bounded arity, e.g. binary constraints, which can be encoded as polysize tables. Figure 3 shows that the other 13 representations are more succinct than table, thus, c can be encoded as all 14 representations in polysize. In addition, the cells in Figure 2 not colored can also be directly verified from the arcs in Figure 3.

Theorem 2. The results in Figure 3 hold.

Proof. We first sketch the proofs for the arcs in Figure 3:

- $\mathbf{A}_1$: Proposition 8 shows CFG$\leq$BCT;
- $\mathbf{A}_2$: (Wang and Yap 2022a) shows BCT$\leq$NFA;
- $\mathbf{A}_3$, $\mathbf{A}_{14}$: see section and also (Amilhastre et al. 2014);
- $\mathbf{A}_{13}$: Proposition 9 shows OMDD$\leq$sMDD;
- $\mathbf{A}_4$, $\mathbf{A}_8$: c-T is both smaT and segT, so c-T$\leq$ smaT, segT;
- $\mathbf{A}_5$: c-T is a special case of bsmaT and each baic smart tuple can be encoded as a polysize c-tuple, so bsmaT=c-T;
- $\mathbf{A}_6$: each c-tuple can be encoded as a path in an OMVD;
- $\mathbf{A}_7$: OMDD is a special case of OMVD;
- $\mathbf{A}_9$: each short support can be encoded as a c-tuple;
- $\mathbf{A}_{10}$: see (Jefferson and Nightingale 2013);
- $\mathbf{A}_{11}$: an entry t encodes at most 1 or $|rel(c)|$ tuples where c is the table in t and a table can be an entry, so table=sliT.
- $\mathbf{A}_{12}$: see (Verhaeghe, Lecoutre, and Schaus 2018).

Then based on the results in Figure 2, we can verify that if a representation A cannot reach another representation B via a path in Figure 3, there are families which can be encoded as polysize B but not A, i.e. in some columns of Figure 2, A is a $\bullet$ and B is a $\checkmark$.

For example, in the $\mathbf{F}_6$ column of Figure 2, smaT is a $\bullet$ and segT, OMVD, BCT, CFG are $\checkmark$, therefore, there is no path from smaT to segT, OMVD, BCT and CFG, which means smaT$\nleq$segT, smaT$\nleq$OMVD, smaT$\nleq$BCT and smaT$\nleq$CFG. In addition, in the $\mathbf{F}_{14}$ column of Figure 2, OMVD is a $\bullet$ and BCT, CFG are $\checkmark$, thus, OMVD$\nleq$BCT and OMVD$\nleq$CFG. Similarly, the other ad-hoc constraints can also be verified based on the results in Figure 2.

So the results in Figure 3 hold. $\square$

Recently, a BCT GAC propagator (Wang and Yap 2022b) was shown to outperform the state-of-the-art OMDD and table GAC propagators. A natural question that arises is "which ad-hoc constraints can be handled by the BCT GAC propagator?" From Figure 3, we immediately see that except for the CFG, smaT and segT constraints, the other 11 ad-hoc constraints can be encoded as polysize BCT and directly handled by the BCT GAC propagator.

Proposition 8. CFG is at least as succinct as BCT.

Proof. Let $A = (V, C)$ be a BCT. If $|V| = 1$, $sol(A)$ can be encoded as a polysize CFG. Assume for $1 \leq |V| < k$, $sol(A)$ can be encoded as a polysize CFG. Let $x \in V$ and c be the only constraint in C including x and $scp(c) = \{x, y\}$.

If $|V| = k$, there is a polysize CFG R encoding $sol(V \setminus \{x\}, C \setminus \{c\})$. Then a CFG encoding $sol(A)$ can be constructed by replacing each production $\alpha \to a$ in $\mathcal{A}^R$ such that $scp(\alpha) = \{y\}$ with the productions: $\alpha \to a\beta^{(y,a)}$ and $\{\beta^{(y,a)} \to b | \{(x, b), (y, a)\} \in rel(c)\}$.

By induction on $|V|$, there is a polysize CFG R encoding $sol(A)$ for any BCT A. Then $rel(A) = sol(A)[scp(A)]$ can be encoded as a CFG by eliminating all non-terminals α in $\mathcal{A}^R$ such that $scp(\alpha) \cap scp(A) = \emptyset$. So CFG $\leq$ BCT. $\square$

Proposition 9. OMDD is at least as succinct as sMDD.

Proof. Assume R is a sMDD w.r.t. an order O over $scp(R)$ and $k = \lfloor \frac{n}{2} \rfloor + 1$ and $n = |scp(R)| > 10$ and for any node v in the layer L_k of R and $a \in \mathcal{D}(O_k)$, $R(v, a)$ is the subgraph in R consisting of the paths from the nodes in $out(v, a)$ to the terminal node and G_v^a is a copy of the transpose of $R(v, a)$.

Let G_1 be the subgraph induced in R by $\bigcup_{i=1}^{k} L_i$ and G_2 be a graph combining G_1 with all G_v^a, such that $v \in L_k$ and $a \in \mathcal{D}(O_k)$ and G_v^a is not empty, by adding an arc labelled with a pointing from v to the root of G_v^a.

An OMDD encoding $rel(R)$ can be constructed by merging all terminal nodes in G_2, where G_2 is up to $|L_k| \times |\mathcal{D}(O_k)|$ times larger than R. So OMDD $\leq$ sMDD. $\square$

[2]In more detail, $A \leq B$, if for any constraint c encoded by B with size n, there exists an encoding A of c whose size is polynomial in n. A and B are *equally succinct*, denoted as $A = B$, if $A \leq B$ and $B \leq A$. In addition, A and B are *incomparable, denoted as* $A \neq B$, if $A \nleq B$ and $B \nleq A$. A is *strictly more succinct* than B, denoted as $A < B$, if $A \leq B$ and $B \nleq A$.

Operation	Description	Definition
$A \vee B$	disjunction	$\{\tau \in U(scp(A) \cup scp(B)) \mid \tau[scp(A)] \in rel(A)$ or $\tau[scp(B)] \in rel(B)\}$
$\bigvee S$	big disj.	$\bigvee_{R \in S} R$
$A \wedge B$	conjunction	$\{\tau \in U(scp(A) \cup scp(B)) \mid \tau[scp(A)] \in rel(A)$ and $\tau[scp(B)] \in rel(B)\}$
$\bigwedge S$	big conj.	$\bigwedge_{R \in S} R$
$\neg A$	negation	$U(scp(A)) \setminus rel(A)$
$A[-Y]$	FO	$rel(A)[scp(A) \setminus Y]$
$A[-y]$	SFO	$rel(A)[scp(A) \setminus \{y\}]$
$A\vert\tau$	CD	$\{\tau' \setminus \tau \mid \tau' \in rel(A), \tau'[Y] \subseteq \tau\}$

Table 3: Operations: A, B are representations, S is a set of representations, y is a variable in $scp(A)$, Y is a subset of $scp(A)$, a is a value in $\mathcal{D}(y)$ and τ is a tuple over Y.

Operations on Ad-Hoc Constraints

We now consider 8 operations inspired by KC (Darwiche and Marquis 2002) which are useful for building CSP models: (i) The CD (polytime conditioning), FO (polytime forgetting) and SFO (polytime singleton forgetting) operations can be used to eliminate variables, e.g. the variables assigned during search can be eliminated; (ii) Logical operations can be used to combine constraints in CSP models, e.g. the applications given in (Bacchus and Walsh 2005). The definitions of the operations can be found in Table 3.

In knowledge compilation, a representation is said to *satisfy an operation* if the time complexity of computing the operation is polynomial in the representation size (Darwiche and Marquis 2002). Figure 4 shows the results of the operations. DFA and OMDD (NFA and OMVD) can be encoded as each other in polysize with the same order, so they satisfy the same operations. In addition, sliT = table and bsmaT = c-T, thus, sliT and bsmaT respectively satisfy the same operations as table and c-T.

Theorem 3. The results in Figure 4 hold.

In Figure 4, the green cells are straightforward, and the red cells can be directly verified by using of the families given in the cells. The proofs of OMDD/OMVD are given in (Amilhastre et al. 2014). We now prove the blue cells.

The family $\mathbf{F}_8$ can be encoded as a conjunction of 2 segTs encoding $\bigwedge_{i=1}^{n}\{x_i \neq x_{i+1} | i\%2 = 1\}$ and $\bigwedge_{i=1}^{n}\{x_i \neq x_{i+1} | i\%2 = 0\}$, so segT does not satisfy $A \wedge B$.

Given a 3-SAT F, assume g is a function which maps clause variable pairs in F to an integer in $\{1, 2, 3\}$ such that for any clause c in F and 2 variables $v_1, v_2 \in scp(c)$, $g(c, v_1) \neq g(c, v_2)$ if $v_1 \neq v_2$. Let gc_k be the constraint $\bigwedge\{h_c - v \neq a + l_v \mid h_c - v \neq a + l_v$ is included in the expression of $\mathbf{F}_7$, $c \in F$, $g(c, v) = k\}$ where $k \in \{1, 2, 3\}$ and $a \in \{-1, 1, 3\}$. Then the family $\mathbf{F}_7$ can be modelled as $gc_1 \wedge gc_2 \wedge gc_3$, where gc_k is a smart tuple which can be encoded as a smaT or sMDD. $\mathbf{F}_7$ is NP-hard, so the smaT, sMDD, BCT, CFG constraints do not satisfy the $A \wedge B$ operation unless NP=P. In addition, sMDD satisfies the $\neg A$ operation (see Proposition 10), so the sMDD constraint does not satisfy $A \wedge B$ and $A \vee B$ unless NP=P.

Let τ be a tuple over variables Y. For any BCT $A = (V, C)$, $A|\tau$ is encoded as a BCT $(V, \{c' \mid c \in C\})$ over $scp(A) \setminus Y$ where $rel(c') = \{\tau_1 \in rel(c) | \tau_1[Y] \subseteq \tau\}$ and $scp(c') = scp(c)$. For any CFG A, $A|\tau$ is encoded by removing all non-terminals α such that $\tau[scp(\alpha)] \notin rel(\alpha)[Y]$ or $scp(\alpha) \subseteq Y$ from $\mathcal{A}^A$. So BCT, CFG satisfy CD. $\mathbf{F}_3$ can be defined as $A|\{(y_k, 1)|1 \leq k \leq n\}$ such that A is a sMDD encoding $\mathbf{F}_4$, so sMDD does not satisfy CD.

Any BCT A can also encoding $rel(A)[scp(A) \setminus Y]$ over $scp(A) \setminus Y$. For any CFG A, $A[-Y]$ can be encoded as a CFG by removing all non-terminals α such that $scp(\alpha) \subseteq Y$ from the acyclic CFG $\mathcal{A}^R$. So BCT, CFG satisfy the SFO and FO operations. $\mathbf{F}_6$ is a 3-SAT such that all clauses do not share any variables, thus, there is a smaT A encoding a subset of $\mathbf{F}_7$ with 1 smart tuple such that $A[-V]$ encodes $\mathbf{F}_6$ where $V = \{h_c | c \in F\}$. Then for any smaT A and a variable $y \in scp(A)$, $A[-y] = \bigvee_{a \in \mathcal{D}(y)}(A|\{(y, a)\})$. So smaT satisfies the SFO operation but not the FO operation.

	$A \wedge B$	$A \vee B$	$\bigwedge S$	$\bigvee S$	$\neg A$	SFO	FO	CD
Table	✓	● $\mathbf{F}_{11}$	● $\mathbf{F}_1$	● $\mathbf{F}_{11}$	● $\mathbf{F}_{12}$	✓	✓	✓
shoT	✓	✓	● $\mathbf{F}_1$	✓	● $\mathbf{F}_{1,2}$	✓	✓	✓
c-T	✓	✓	● $\mathbf{F}_1$	✓	● $\mathbf{F}_{1,2}$	✓	✓	✓
segT	● $\mathbf{F}_8$	✓	● $\mathbf{F}_8$	✓	● $\mathbf{F}_8$	✓	✓	✓
smaT	○ $\mathbf{F}_7$	✓	● $\mathbf{F}_6$	✓	● $\mathbf{F}_6$	✓	● $\mathbf{F}_{6,7}$	✓
sMDD	○ $\mathbf{F}_7$	○ $\mathbf{F}_7$	● $\mathbf{F}_1$	● $\mathbf{F}_2$	✓	● $\mathbf{F}_{3,5}$	● $\mathbf{F}_{3,5}$	● $\mathbf{F}_4$
BCT	○ $\mathbf{F}_7$	?	● $\mathbf{F}_1$	● $\mathbf{F}_9$	● $\mathbf{F}_{1,2}$	✓	✓	✓
CFG	○ $\mathbf{F}_7$	?	● $\mathbf{F}_1$	● $\mathbf{F}_9$	● $\mathbf{F}_{1,2}$	✓	✓	✓
OMVD	○ $\mathbf{F}_7$	?	● $\mathbf{F}_1$	● $\mathbf{F}_9$	● $\mathbf{F}_{1,2}$	✓	✓	✓
OMDD	○ $\mathbf{F}_7$	○ $\mathbf{F}_7$	● $\mathbf{F}_1$	● $\mathbf{F}_2$	✓	● $\mathbf{F}_{2,3}$	● $\mathbf{F}_{2,3}$	✓

Figure 4: Satisfied operations: ✓ means a representation satisfies a operation; ● (○) means a representation does not satisfy an operation (operation unless NP=P); '?' is unknown. Reasons for unsatisfied operations are given with families.

Proposition 10. sMDD satisfies $\neg A$.

Proof. Let A be a sMDD w.r.t. an order O and $n = |scp(A)|$ and $k = \lfloor \frac{n}{2} \rfloor$. Then the negation of a sMDD A can be encoded as a sMDD by reconstructing A with:

1. for each $1 < i \leq k+1$, adding a node tt_i to L_i if existing $j < i$, $v \in L_j$ and $a \in \mathcal{D}(O_j)$ such that $out(v, a) = \emptyset$;
2. for each $k+1 < i \leq n$, adding a node tt_i to L_i if existing $i < j$, $v \in L_j$, and $a \in \mathcal{D}(O_{j-1})$ such that $in(v, a) = \emptyset$;
3. for each $1 \leq i \leq k$, $a \in \mathcal{D}(O_i)$, and $v \in L_i$ such that $out(v, a) = \emptyset$, adding an arc labelled with the value a pointing from v to tt_{i+1};
4. for each $k + 2 \leq i \leq n, a \in \mathcal{D}(O_i), v \in L_{i+1}$ such that $in(v, a) = \emptyset$, adding an arc labelled with the value a pointing from tt_i to v;
5. resetting the arcs pointing from nodes in L_{k+1} to nodes in L_{k+2} with the arcs which are not in the original A.

So the sMDD constraint satisfies the $\neg A$ operation. □

The red cells in Figure 2 can be verified with the results in Figure 4. OMDD satisfies $\neg A$ and the negation of $\mathbf{F}_2$ is $\mathbf{F}_1$, so there is no polysize OMDD encoding $\mathbf{F}_2$. For any representation R encoding $\mathbf{F}_4$ or $\mathbf{F}_5$, $R[-\{h_1, h_2, y_1, \cdots, y_n\}]$ encodes $\mathbf{F}_2$. As table satisfies FO, thus, $\mathbf{F}_4$ and $\mathbf{F}_5$ cannot be encoded as polysize table.

	$A \wedge B$	$A \vee B$	$\bigwedge S$	$\bigvee S$	$\neg A$	SFO	FO	CD
sMDD	$\bullet$ $\mathbf{F}_4$	$\bullet$ $\mathbf{F}_4$	$\bullet$ $\mathbf{F}_1$	$\bullet$ $\mathbf{F}_2$	✓	$\bullet$ $\mathbf{F}_{3,5}$	$\bullet$ $\mathbf{F}_{3,5}$	$\bullet$ $\mathbf{F}_4$
BCT	✓	✓	$\bullet$ $\mathbf{F}_1$	✓	$\bullet$ $\mathbf{F}_{1,2}$	✓	✓	✓
CFG	?	✓	$\bullet$ $\mathbf{F}_1$	✓	$\bullet$ $\mathbf{F}_{1,2}$	✓	✓	✓
OMVD	✓	✓	$\bullet$ $\mathbf{F}_1$	✓	$\bullet$ $\mathbf{F}_{1,2}$	✓	✓	✓
OMDD	✓	✓	$\bullet$ $\mathbf{F}_1$	$\bullet$ $\mathbf{F}_2$	✓	$\bullet$ $\mathbf{F}_{2,3}$	$\bullet$ $\mathbf{F}_{2,3}$	✓

Figure 5: Representations using the same graph/order.

Representations Using the Same Graph/Order

We then look at the representations using the same graph or order. In Figure 5, we only give the results of the sMDD, OMDD, OMVD, BCT and CFG using the same graph/order, as the table and non-ordinary tables do not require any order/graph. We also see the usefulness of the families in showing the negative results.

Theorem 4. The results in Figure 5 hold.

The proofs of OMDD and OMVD can be found in (Amilhastre et al. 2014). The proofs of yellow cells are the same as those in Figure 4, and red cells can be directly verified by the families given in the cells. We now prove the blue cells.

Let A and B be the polysize sMDDs that respectively encode $\mathbf{F}_4$ and $\bigwedge_{i=1}^{n} v_i = 1$ using the order $O = h_1 < x_1 < \cdots < x_{\lfloor \frac{n}{2} \rfloor} < v_1 < \cdots < v_n < x_{\lfloor \frac{n}{2} \rfloor + 1} < \cdots < x_n < y_1 < \cdots < y_n$. There is no polysize sMDD R encoding $A \wedge B$ w.r.t. O, otherwise we can get a polysize sMDD encoding $\mathbf{F}_3$ by removing the single value variables y_1,v_1,$\cdots$,y_n,v_n from R. In addition, sMDD satisfies the negation operation, so sMDDs using the same order do not satisfy the $A \wedge B$ and $A \vee B$ operations.

For any CFG R and a variable $x \notin scp(R)$, x can be added behind (before) any $y \in scp(R)$ by replacing the productions $\alpha \to a$ in $\mathcal{A}^R$ such that $scp(\alpha) = \{y\}$ with the productions: $\alpha \to a\beta$ ($\alpha \to \beta a$) and $\{\beta \to b | b \in \mathcal{D}(x)\}$. So CFGs using an order O can be extended to the CFGs w.r.t. O which satisfies the $A \vee B$ and $\bigvee S$ operations, since CFG is closed under union.

Propositions 11 and 12 show BCTs using the same graph satisfy the $A \wedge B$, $A \vee B$ and $\bigvee S$ operations.

Proposition 11. BCTs using the same graph satisfy $A \wedge B$.

Proof. Let A, B be 2 BCTs using a graph $(H \cup Y, E)$. The $A \wedge B$ operation can be encoded as a BCT $R = (\{f^R(v) | v \in H \cup Y\}, \{f^R(e) | e \in E\})$ by mapping the vertices v and edges e in the graph to variables $f^R(v)$ and binary constraints $f^R(e)$ such that $\mathcal{D}(f^R(v))$=$U(\{f^A(v), f^B(v)\})$ for $v \notin Y$ and $scp(f^R(e))$=$\{f^R(v) | v \in e\}$ and $rel(f^R(e))$ is $\{\bigcup_{v \in e} t_\tau^v | \tau \in sol(\{f^A(e), f^B(e)\})\}$ where t_τ^v=$\tau[\{v\}]$ if $v \in Y$, otherwise $t_\tau^v = \{(f^R(v), \tau[\{f^A(v), f^B(v)\}])\}$.

Every tuple τ_R in $sol(R)$ corresponds to two tuples $\tau_A \in sol(G(A))$ and $\tau_B \in sol(G(B))$ such that $\tau_R[Y] \cup \bigcup \{a | (v, a) \in \tau_R, v \notin Y\} = \tau_A \cup \tau_A$. Then R also uses the graph G, so BCTs using the same graph satisfy $A \wedge B$. $\square$

Proposition 12. BCTs using the same graph satisfy $\bigvee S$.

Proof. Let S be a set of BCTs using $(H \cup Y, E)$. $\bigvee_{A \in S} A$ can be encoded as a BCT $(\{f^R(v) | v \in H \cup Y\}, \{f^R(e) | e \in E\})$ such that $scp(f^R(e)) = \{f^R(v) | v \in e\}$ and $rel(f^R(e)) = \bigcup_{A \in S} \{\{t_{v,a}^A | (v, a) \in \tau\} | \tau \in rel(f^A(e))\}$ where $t_{v,a}^A$=(v, a) if $v \in Y$, otherwise $t_{v,a}^A$=$(f^R(v), a^A)$ and $\mathcal{D}(f^R(v)) = \bigcup_{A \in S} \{a^A | a \in \mathcal{D}(f^A(v))\}$.

For each tuple τ_R in $sol(R)$, there is $A \in S$ such that the tuple $\tau_R[Y] \cup \{(v, a) | (v, a^A) \in \tau_R, v \notin Y\}$ is a tuple in $sol(G(A))$. So BCTs using the same graph satisfy $\bigvee S$. $\square$

Query	Definition	Usage
CO	is $rel(A) \neq \emptyset$?	necessary condition of GAC
VA	is $rel(A) = U(scp(A))$?	remove universal constraints
CE	is $rel(t) \cap rel(A) = \emptyset$?	enforce GAC on A
IM	is $rel(t) \subseteq rel(A)$?	enforce GAC on $\neg A$
EQ	is $rel(A) = rel(B)$?	identify equivalent/implied
SE	is $rel(A) \subseteq rel(B)$?	constraints A
CT	what is the size of $rel(A)$?	applications in (Pesant 2005)
ME	enumerating the tuples in $rel(A)$	improve CSP models by tabling (Dekker et al. 2017)

Table 4: Queries, where A, B are representations and t is a c-tuple over $scp(A)$ and CO, VA, CE, IM, EQ, SE, CT, ME stand for "consistency", "validity", "clausal entailment", "implicant", "equivalence", "sentential entailment", " model counting", "model enumeration".

Queries on Ad-Hoc Constraints

Finally, we investigate 8 queries (the notation comes from the KC literature (Darwiche and Marquis 2002)) which may be asked on the constraints in a CSP model. Table 4 shows the queries and example usage of the queries. A representations R *satisfy a query* (not ME) if the time complexity of computing the query is polynomial in the size of R, in addition, R *satisfy ME* if the cost of computing ME is polynomial in the size of R and $rel(R)$.

The current domain of the variables in a constraint A can be denoted as a c-tuple t over $scp(A)$, and enforcing GAC on A (on $\neg A$) with the current domain t is iteratively removing the tuples $\{(x, a)\}$ from the relation of constraints $c \in t$ such that $rel(t') \cap rel(A) = \emptyset$ ($rel(t') \subseteq rel(A)$) where $t' = \{c'\} \cup (t \setminus \{c\})$ and $rel(c') = \{(x, a)\}$ and $x \in scp(c)$. Hence, $rel(t) \cap rel(A) = \emptyset$ ($rel(t) \subseteq rel(A)$) iff enforcing GAC on A (on $\neg A$) with t removes all tuples from t. So CE (IM) on A corresponds to enforcing GAC on A (on $\neg A$).

The satisfied queries of OMDD and OMVD can be found in (Amilhastre et al. 2014). In addition, sliT, bsmaT, DFA and NFA respectively satisfy the same queries as table, c-T, OMDD and OMVD. The satisfied queries of the other representations can be found in Figure 6.

Theorem 5. The results in Figure 6 hold.

The ad-hoc constraints have polytime GAC propagators, so they satisfy CE and CO.

Then sMDD<OMDD and the ad-hoc constraints satisfying CO and CD also satisfy ME (Amilhastre et al. 2014). So all 14 ad-hoc constraints satisfy ME.

$\mathbf{F}_{13}$ can be encoded as polysize shoT, so shoT, c-T, segT, smaT, BCT, CFG do not satisfy VA, IM, EQ, EQ*, SE, CT, SE* unless NP=P. Table can be encoded as polysize OMDD w.r.t. any order, so table satisfies VA, IM, EQ, SE, CT.

	CO	VA	CE	IM	EQ	EQ*	SE	SE*	CT	ME
Table	✓	✓	✓	✓	✓	-	✓	-	✓	✓
shoT	✓	○	✓	○	○	-	○	-	○	✓
c-T	✓	○	✓	○	○	-	○	-	○	✓
segT	✓	○	✓	○	○	-	○	-	○	✓
smaT	✓	○	✓	○	○	-	○	-	○	✓
sMDD	✓	✓	✓	✓	✓	✓	○	✓	✓	✓
BCT	✓	○	✓	○	○	○	○	○	○	✓
CFG	✓	○	✓	○	○	○	○	○	○	✓
OMVD	✓	○	✓	○	○	○	○	○	○	✓
OMDD	✓	✓	✓	✓	✓	✓	○	✓	✓	✓

Figure 6: Satisfied Queries: ✓(○) means a representation satisfies (does not satisfy) a query (unless NP=P), and EQ*, SE* require the representations use the same graph/order.

sMDD does not satisfy SE unless NP=P, since sMDD satisfies $\neg A$ but not $A \wedge B$ unless NP=P. Then sMDD<OMDD, so sMDD satisfies VA, IM, EQ, CT. $\neg A \wedge B$ is a polysize OMVD using O for any polysize sMDDs A, B using an order O. So sMDD satisfies EQ*, SE*.

Case Studies

We now briefly (due to space) give case studies showing applications of our results to modelling and constraint usage.

The MaxOrder Constraint (Papadopoulos, Roy, and Pachet 2014). A MaxOrder constraint c over $\{x_1,\cdots,x_n\}$ is defined as $\bigwedge_{i=1}^{n-1} L_1^i \wedge (\bigwedge_{j=1}^{n-l+1} \neg L_2^j)$, where L_1 and L_2 are 2 languages, L_2^j is defined by L_2 w.r.t. $x_j < \cdots < x_{j+l-1}$ and L_1^i is defined by L_1 w.r.t. $x_i < x_{i+1}$.

As shown in (Papadopoulos, Roy, and Pachet 2014; Perez and Régin 2015), L_1^i,L_2^j can be encoded as DFAs/OMDDs using the same order which satisfy $\neg A$ and $A \wedge B$, and c can be obtained by combining the DFAs/OMDDs.

Then we can also model the DFAs encoding L_1^i,$\neg L_2^j$ as BCTs (NFAs,OMVDs) using the same graph/order, since they satisfy $A \wedge B$ and BCT<NFA=OMVD<DFA.

The C&Lex(X,Y,Regular) Constraint (Katsirelos, Narodytska, and Walsh 2008). The constraint is defined as $c_1 \wedge c_2 \wedge c_3$ such that c_1,c_2 are DFAs and c_3 is the $Lex(X, Y)$ constraint (Frisch et al. 2002) where X= $scp(c_1)$ and Y=$scp(c_2)$ and $X \cap Y = \emptyset$.

The constraint can be created by joining the DFAs encoding c_1,c_2,c_3 (Katsirelos, Narodytska, and Walsh 2008). Then the DFAs can also be encoded as BCTs/OMVDs using a graph/order, since they satisfy $A \wedge B$ and OMVD<DFA.

Constraints Used in Bucket Elimination. The bucket elimination algorithm (Dechter 1999) works by iteratively eliminating variables x and constraints C_x including x from a CSP (X, C) by adding a constraint $c_x = (\bigwedge_{c \in C_x} c)[-x]$ where the constraints in C are encoded as tables.

Since BCT<c-T<shoT<table and BCT (shoT, c-T) using the same graph/order satisfies $A \wedge B$ and SFO, we can also use BCTs (shoTs, c-Ts) to encode the constraints.

We see that representations which do not satisfy SFO or $A \wedge B$ would be less suited to encode the constraints in C.

The Root Constraint (Bessiere et al. 2006). The Root constraint Roots($[x_1, \cdots, x_n]$, S, T) is NP-hard but it can be decomposed into the constraints $i \in S \leftrightarrow x_i \in T$, where $i \in [1, \cdots, n]$ and S, T are set variables (Bessiere et al. 2009). The set variables can be implemented as Boolean variables (Hawkins, Lagoon, and Stuckey 2005), and then the constraints can be encoded as $c_1 \wedge c_2$, where c_1 is $\neg s_i \vee (\bigvee_{j \in \mathcal{D}(x_i)} x_i = j \wedge t_j)$ and c_2 is $\bigwedge_{j \in \mathcal{D}(x_i)} s_i \vee x_i \neq j \vee \neg t_j$ and s_i, t_j are Boolean variables.

BCTs using the same graph satisfy the $\bigvee S$ and $A \wedge B$ operations, and the treewidth of c_2 is 2, thus, it is straightforward that $c_1 \wedge c_2$ can be encoded as a polysize BCT.

Related Work

The KC properties, i.e. the succinctness, operations and queries, have been used to analyze KC forms (Darwiche and Marquis 2002), such as the OBDD (Bryant 1986), d-DNNF (Darwiche 2001) and Pseudo-Boolean constraints (Le Berre et al. 2018). However, most of the focus in KC is on compiling Boolean functions. In this paper, we focus on investigating existing ad-hoc constraints over finite domain variables—these are incomparable with or more succinct than the Boolean function representations.

There are also other representations which can be used to compile CSPs, such as AOMDD (Mateescu and Dechter 2006) and MDDG (Koriche et al. 2015). Their works mainly focus on compiling the solution space of CSPs but not modelling CSPs. Furthermore, $\mathbf{F}_2$ cannot be encoded as a polysize AOMDD, since AOMDD reduces to just OMDD on encoding $\mathbf{F}_2$ which has a complete constraint graph. Then MDDG over Boolean variables is decision-DNNF (Fargier and Marquis 2006) and d-DNNF, thus, there is no polysize MDDG encoding DNF unless the polynomial hierarchy collapses (Darwiche and Marquis 2002). So shoT, c-T, smaT, segT, OMVD, BCT and CFG are incomparable with or are more succinct than AOMDD and MDDG.

Both intensional constraints and binary CSPs with hidden variables (Dechter 1990b) can also be regarded as ad-hoc constraints. However, it is NP-hard to enforce GAC on intensional constraints (Bessiere et al. 2007) and the constraints defined by binary CSPs with hidden variables.

Conclusion

Ad-hoc constraints (also called generic constraints) are very useful for modelling many constraint problems. Numerous compact representations have been proposed to define ad-hoc constraints, such as the 14 representations investigated here. In this paper, we give a succinctness map to show whether there exists any super-polynomial blow-ups between the representations of ad-hoc constraints. We then analyze what operations/queries are tractable on these ad-hoc constraints. We believe this paper is the first comprehensive work to study the tractability of these ad-hoc constraints from a space and time perspective. Our results not only improve our understanding of the foundations of these ad-hoc constraints but also give guidance when they are used in new ways. We believe it can be used to better understand choices and tradeoffs when building a CSP model to a problem.

Acknowledgements

This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0004] and grant T1 251RES2219.

References

Amilhastre, J.; Fargier, H.; Niveau, A.; and Pralet, C. 2014. Compiling CSPs: A complexity map of (non-deterministic) multivalued decision diagrams. *International Journal on Artificial Intelligence Tools*, 23(04): 1460015.

Audemard, G.; Lecoutre, C.; and Maamar, M. 2020. Segmented Tables: An Efficient Modeling Tool for Constraint Reasoning. In *European Conference on Artificial Intelligence*, 315–322. IOS Press.

Bacchus, F.; and Walsh, T. 2005. Propagating logical combinations of constraints. In *International Joint Conference on Artificial Intelligence*, 35–40.

Bessiere, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2006. The ROOTS Constraint. In *International Conference on Principles and Practice of Constraint Programming*, 75–90.

Bessiere, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2009. Range and Roots: Two common patterns for specifying and propagating counting and occurrence constraints. *Artificial Intelligence*, 173(11): 1054–1078.

Bessiere, C.; Hebrard, E.; Hnich, B.; and Walsh, T. 2007. The complexity of reasoning with global constraints. *Constraints*, 12(2): 239–259.

Bessière, C.; and Régin, J.-C. 1997. Arc Consistency for General Constraint Networks: Preliminary Results. In *International Joint Conference on Artificial Intelligence*, 398–404.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8): 677–691.

Cheng, K.; and Yap, R. H. C. 2010. An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. *Constraints*, 15(2): 265–304.

Cheng, K. C.; Xia, W.; and Yap, R. H. 2012. Space-time tradeoffs for the regular constraint. In *International Conference on Principles and Practice of Constraint Programming*, 223–237.

Darwiche, A. 2001. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2): 11–34.

Darwiche, A.; and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17: 229–264.

Dechter, R. 1987. Decomposing an N-ary Relation into a Tree of Binary Relations. In Vardi, M. Y., ed., *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 185–189.

Dechter, R. 1990a. Decomposing a relation into a tree of binary relations. *Journal of Computer and System Sciences*, 41(1): 2–24.

Dechter, R. 1990b. On the Expressiveness of Networks with Hidden Variables. In *AAAI National Conference on Artificial Intelligence*, 556–562.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2): 41–85.

Dechter, R.; and Pearl, J. 1989. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3): 353–366.

Dekker, J. J.; Björdal, G.; Carlsson, M.; Flener, P.; and Monette, J.-N. 2017. Auto-tabling for subproblem presolving in MiniZinc. *Constraints*, 22(4): 512–529.

Demeulenaere, J.; Hartert, R.; Lecoutre, C.; Perez, G.; Perron, L.; Régin, J.-C.; and Schaus, P. 2016. Compact-Table: efficiently filtering table constraints with reversible sparse bit-sets. In *International Conference on Principles and Practice of Constraint Programming*, 207–223.

Fargier, H.; and Marquis, P. 2006. On the Use of Partially Ordered Decision Graphs in Knowledge Compilation and Quantified Boolean Formulae. In *AAAI National Conference on Artificial Intelligence*, 42–47.

Freuder, E. C. 1982. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1): 24–32.

Frisch, A.; Hnich, B.; Kiziltan, Z.; Miguel, I.; and Walsh, T. 2002. Global constraints for lexicographic orderings. In *International Conference on Principles and Practice of Constraint Programming*, 93–108.

Frisch, A. M.; Harvey, W.; Jefferson, C.; Martinez-Hernandez, B.; and Miguel, I. 2008. Essence: A constraint language for specifying combinatorial problems. *Constraints*, 13(3): 268–306.

Gange, G.; Stuckey, P. J.; and Szymanek, R. 2011. MDD propagators with explanation. *Constraints*, 16(4): 407.

Gharbi, N.; Hemery, F.; Lecoutre, C.; and Roussel, O. 2014. Sliced Table Constraints: Combining Compression and Tabular Reduction. In *International Conference on Integration of Artificial Intelligence and Operations Research techniques in Constraint Programming*, 120–135.

Hawkins, P.; Lagoon, V.; and Stuckey, P. J. 2005. Solving Set Constraint Satisfaction Problems using ROBDDs. *J. Artif. Intell. Res.*, 24: 109–156.

Jefferson, C.; and Nightingale, P. 2013. Extending simple tabular reduction with short supports. In *International Joint Conferences on Artificial Intelligence*, 573–579.

Kadioglu, S.; and Sellmann, M. 2008. Efficient Context-Free Grammar Constraints. In *AAAI National Conference on Artificial Intelligence*, 310–316.

Kadioglu, S.; and Sellmann, M. 2010. Grammar constraints. *Constraints*, 15(1): 117–144.

Katsirelos, G.; Narodytska, N.; and Walsh, T. 2008. Combining symmetry breaking and global constraints. In *International Workshop on Constraint Solving and Constraint Logic Programming*, 84–98.

Katsirelos, G.; Narodytska, N.; and Walsh, T. 2009. Reformulating global grammar constraints. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 132–147.

Katsirelos, G.; and Walsh, T. 2007. A compression algorithm for large arity extensional constraints. In *International Conference on Principles and Practice of Constraint Programming*, 379–393.

Koriche, F.; Lagniez, J.-M.; Marquis, P.; and Thomas, S. 2015. Compiling constraint networks into multivalued decomposable decision graphs. In *International Joint Conference on Artificial Intelligence*, 332–338.

Lange, M.; and Leiß, H. 2009. To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm. *Informatica Didactica*, 8(2009): 1–21.

Le Berre, D.; Marquis, P.; Mengel, S.; and Wallon, R. 2018. Pseudo-Boolean Constraints from a Knowledge Representation Perspective. In *International Joint Conference on Artificial Intelligence*, 1891–1897.

Lecoutre, C. 2011. STR2: optimized simple tabular reduction for table constraints. *Constraints*, 16(4): 341–371.

Lecoutre, C.; Likitvivatanavong, C.; and Yap, R. H. 2015. STR3: A path-optimal filtering algorithm for table constraints. *Artificial Intelligence*, 220: 1–27.

Mairy, J.-B.; Deville, Y.; and Lecoutre, C. 2015. The Smart Table Constraint. In *International Conference on the Integration of AI and OR Techniques in Constraint Programming*, 271–287.

Mateescu, R.; and Dechter, R. 2006. Compiling constraint networks into AND/OR multi-valued decision diagrams (AOMDDs). In *International Conference on Principles and Practice of Constraint Programming*, 329–343.

Moshier, M. D.; and Rounds, W. C. 1987. On the succinctness properties of unordered context-free grammars. In *Annual Meeting of the Association for Computational Linguistics*, 112–116.

Nethercote, N.; Stuckey, P. J.; Becket, R.; Brand, S.; Duck, G. J.; and Tack, G. 2007. MiniZinc: Towards a standard CP modelling language. In *International Conference on Principles and Practice of Constraint Programming*, 529–543.

Ogden, W.; Ross, R. J.; and Winklmann, K. 1985. An "interchange lemma" for context-free languages. *SIAM Journal on Computing*, 14(2): 410–415.

Papadopoulos, A.; Roy, P.; and Pachet, F. 2014. Avoiding plagiarism in Markov sequence generation. In *AAAI National Conference on Artificial Intelligence*, 2731–2737.

Perez, G.; and Régin, J.-C. 2014. Improving GAC-4 for table and MDD constraints. In *International Conference on Principles and Practice of Constraint Programming*, 606–621.

Perez, G.; and Régin, J.-C. 2015. Efficient operations on MDDs for building constraint programming models. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 374–380.

Pesant, G. 2004. A regular language membership constraint for finite sequences of variables. In *International Conference on Principles and Practice of Constraint Programming*, 482–495.

Pesant, G. 2005. Counting Solutions of CSPs: A Structural Approach. In *International Joint Conference on Artificial Intelligence*, 260–265.

Puget, J.-F. 1998. A fast algorithm for the bound consistency of alldiff constraints. In *AAAI National Conference on Artificial Intelligence*, 359–366.

Quimper, C.-G.; and Walsh, T. 2006. Global grammar constraints. In *International Conference on Principles and Practice of Constraint Programming*, 751–755.

Quimper, C.-G.; and Walsh, T. 2007. Decomposing global grammar constraints. In *International Conference on Principles and Practice of Constraint Programming*, 590–604.

Razgon, I. 2016. On the read-once property of branching programs and CNFs of bounded treewidth. *Algorithmica*, 75(2): 277–294.

Sellmann, M. 2006. The theory of grammar constraints. In *International Conference on Principles and Practice of Constraint Programming*, 530–544.

Srinivasan, A.; Ham, T.; Malik, S.; and Brayton, R. K. 1990. Algorithms for discrete function manipulation. In *International Conference on Computer-Aided Design*, 92–95.

Verhaeghe, H.; Lecoutre, C.; Deville, Y.; and Schaus, P. 2017. Extending Compact-Table to basic smart tables. In *International Conference on Principles and Practice of Constraint Programming*, 297–307.

Verhaeghe, H.; Lecoutre, C.; and Schaus, P. 2017. Extending Compact-Table to Negative and Short Tables. In *AAAI National Conference on Artificial Intelligence*, 3951–3957.

Verhaeghe, H.; Lecoutre, C.; and Schaus, P. 2018. Compact-MDD: Efficiently Filtering (s)MDD Constraints with Reversible Sparse Bit-sets. In *International Joint Conference on Artificial Intelligence*, 1383–1389.

Verhaeghe, H.; Lecoutre, C.; and Schaus, P. 2019. Extending Compact-Diagram to Basic Smart Multi-Valued Variable Diagrams. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 581–598.

Vion, J.; and Piechowiak, S. 2018. From MDD to BDD and Arc consistency. *Constraints*, 23(4): 451–480.

Wang, R.; Xia, W.; Yap, R. H. C.; and Li, Z. 2016. Optimizing Simple Tabular Reduction with a Bitwise Representation. In *International Joint Conference on Artificial Intelligence*, 787–795.

Wang, R.; and Yap, R. H. C. 2019. Arc Consistency Revisited. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 599–615.

Wang, R.; and Yap, R. H. C. 2020. Bipartite Encoding: A New Binary Encoding for Solving Non-Binary CSPs. In *International Joint Conference on Artificial Intelligence*, 1184–1191.

Wang, R.; and Yap, R. H. C. 2022a. CNF Encodings of Binary Constraint Trees. In *International Conference on Principles and Practice of Constraint Programming*, 40:1–40:19.

Wang, R.; and Yap, R. H. C. 2022b. Encoding Multi-valued Decision Diagram Constraints as Binary Constraint Trees. In *AAAI National Conference on Artificial Intelligence*, 3850–3858.

Xia, W.; and Yap, R. H. C. 2013. Optimizing STR algorithms with tuple compression. In *International Conference on Principles and Practice of Constraint Programming*, 724–732.

Yap, R. H. C.; Xia, W.; and Wang, R. 2020. Generalized Arc Consistency Algorithms for Table Constraints: A Summary of Algorithmic Ideas. In *AAAI National Conference on Artificial Intelligence*, 13590–13597.