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. 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 Szymbek 2011; Perez and Régin 2014; Vion and Piechoiaki 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, $D(x)$ is the domain of a variable $x$, and $C$ is a set of constraints. A literal of a variable is a pair $(x, a)$. A tuple over variables $\{x_{1}, x_{2}, \ldots, x_{r}\}$ is a set of literals $\{(x_{1}, a_{1}), (x_{2}, a_{2}), \ldots, (x_{r}, a_{r})\}$. A tuple $\tau$ is an assignment if $a \in D(x)$ 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 $scp(c)$ in $X$ and a constraint relation $rel(c) \subseteq U(scp(c))$. The arity of a constraint $c$ is $|scp(c)|$. A CSP is called a binary CSP if the largest constraint arity is 2.

Given any variables $V$ and literals $\tau$, we use $\tau[V] = \{(x, a) \in \tau | x \in V\}$ to denote a subset of $\tau$, and $T[\tau] = \{\tau[V] | \tau \in T\}$ is the projection of the tuples on $V$. An assignment $\tau$ 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 $sol(C) = \{\tau | \forall c \in C, \tau[scp(c)] \in rel(c)\}$.

Table: 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 $x = a$, $x \neq a$ and $x \nRightarrow a$, $x \swarrow a$ are unary equality, tautology (and comparisons) relations, where $x, y \in X$ and $a$ is a constant and $\nRightarrow \in \{=, \neq, <, >, \leq, \geq\}$ and $rel(x = a) = \{(y, b) | b \in D(x)\}$.

<table>
<thead>
<tr>
<th>Tuple</th>
<th>Possible constraints in a tuple</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry</td>
<td>unary equalities and a table</td>
<td>separated</td>
</tr>
<tr>
<td>short support</td>
<td>unary equalities and unary tautologies</td>
<td>separated</td>
</tr>
<tr>
<td>c-tuple</td>
<td>unary tables</td>
<td>separated</td>
</tr>
<tr>
<td>segmented tuple</td>
<td>tables, unary equalities and unary tautologies</td>
<td>separated</td>
</tr>
<tr>
<td>basic smart tuple</td>
<td>unary comparisons, unary tables, unary equalities and unary tautologies</td>
<td>separated</td>
</tr>
<tr>
<td>smart tuple</td>
<td>comparisons, unary tables, unary equalities and unary tautologies</td>
<td>acyclic</td>
</tr>
</tbody>
</table>

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 $x = a$, $x \neq a$ and $x \nRightarrow a$, $x \swarrow a$ are unary equality, tautology (and comparisons) relations, where $x, y \in X$ and $a$ is a constant and $\nRightarrow \in \{=, \neq, <, >, \leq, \geq\}$ and $rel(x = a) = \{(y, b) | b \in D(x)\}$.

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 (Amilhatre 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 $\{L_{0}, \ldots, L_{r+1}\}$ 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 $D(O_i)$ for $1 \leq i \leq r$. $out(v, a)$ 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 $a \in D(O_i)$ and a node in $L_i$, $|out(v, a)| \leq 1$ (w.r.t. $O_i$).
- A semi-OMDD (sMDD) is an OMVD such that for any $1 \leq i \leq \left\lfloor \frac{r}{2} \right\rfloor$, $\left\lfloor \frac{r}{2} \right\rfloor + 1 \leq i \leq r + 1$ and $a \in D(O_i)$ and a node in $L_i$, $|out(v, a)| \leq 1$ (w.r.t. $O_i$).

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 $x_1 = x_2 \land h_1 = a_{12}$ w.r.t. the variable order $h_1 < x_1 < x_2 < a_{12}$ with $D(h_1) = \{a_{12}\}$ and $D(x_1) = \{1, 2, 3\}$. 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.
Example 3. The constraint \( \{O_1, a_1, \ldots, O_2, a_2\}\) is in \(rel(c)\) iff \(a_1 \cdots a_2\) 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 \(size(scpt(R))\) 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 Leiβ 2009) which can be encoded into an acyclic CFG \(A^{R}\) (Katsirelos, Narodytska, and Walsh 2009) such that \(rel(A^{R}) = rel(R)\), the right hand side (RHS) of each production in \(A^{R}\) has 2 non-terminals or 1 terminal, and each non-terminal in \(A^{R}\) encodes the tuples \(rel(\alpha)\) over variables \(scpt(\alpha)\) such that \(scpt(\alpha)\cap scpt(\beta) = \emptyset\) if \(\alpha, \beta\) are in the same RHS of a production in \(A^{R}\).

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 \(((0, 1), \{S_0, S_1\}, \{S_0 \rightarrow S_1 \# S_1, S_1 \rightarrow 0 S_1 0, S_1 \rightarrow 1 S_1 1, S_1 \rightarrow \epsilon\}, S_0)\) 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 \(b\). \(\{S_0, S_1\}\) 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 \(((0,1), \{S_3, \ldots, S_{13}\}, \{S_3 \rightarrow S_4 S_5, S_5 \rightarrow S_6 S_7, S_7 \rightarrow \# S_6 \rightarrow \# S_4 \rightarrow S_8 S_{10} S_{11}, S_{12} \rightarrow S_{10} S_{12}, S_{13} \rightarrow S_{11} S_{13}, S_8 \rightarrow 0, S_9 \rightarrow 1, S_{10} \rightarrow 0, S_{11} \rightarrow 1, S_{12} \rightarrow 0, S_{13} \rightarrow 1\}, S_3)\). 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. \(F_1\) expresses a permutation constraint (Puget 1998). \(F_2\) is the negation of \(F_1\). Projecting variable \(h_1\) from \(F_2\) gets \(F_3\). Projecting variables \(h_2, y_1, \ldots, y_n\) from \(F_4\) and \(F_5\) gets \(F_3\).

---

**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 \(scpt(R)\) such that \(scpt(R)\) is a subset of \(V\) and the constraint relation \(rel(R)\) is equal to \(sol(V, C)\cap scpt(R)\) where the variables in \(V\) \(\setminus scpt(R)\) and \(scpt(R)\) 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 \(scpt(R)\) uses a graph \(G = (H \cup Y, E)\), where \(G\) is a tree and the vertices in \(Y\) are variables, if the variables \(scpt(R)\) is a subset of the leaves \(Y\) in \(G\), and there is a BCT \(G(R) = (G(V), G(C))\) encoding \(rel(R)\) and a bijection 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
- \(rel(c) = U(scpt(e))\) 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 \(scpt(f^{R}(e)) = \{f^{R}(v)\mid v \in e\}\).

We remark that a BCT \(R\) over variables \(scpt(R)\) denotes a constraint over \(scpt(R)\) encoded by a BCT.

---

1Note 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 \(x = x'\).
The family $F_{14}$ 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 $k$. $F_{15}$ models the language discussed in Example 4 over $n$ variables.

$F_7$ is used to model the 3-SAT problem $F$. Each clause $c$ in $F$, defined with $(x = l_x) \lor (y = l_y) \lor (z = l_z)$, is modelled as the constraint $c': (h_x - x \neq -1 + l_x) \land (h_y - y \neq -1 + l_y) \land (h_z - z \neq -1 + l_z)$, where $l_x, l_y, l_z \in \{0, 1\}$, and $rel(c) = rel(c')(\{x, y, z\})$. Then $sol(F_7)[X] = sol(F)$ and $F$ is satisfiable iff $F_7$ is satisfiable, so $F_7$ is NP-hard.

In addition, in Figure 2, a family $F$ can be encoded as an ad-hoc constraint $R$ in poly-size if the size of $R$ can be polynomial in the sum of $\text{size}(scp(R))$ 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 $F_1$ includes $n(n-1)$ inequality clauses.

**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 poly-size CFG encoding all permutations of $n$ values. For any variable order, the constraint relation of $F_1$ corresponds to all permutations of $n$ values, so there is no poly-size CFG encoding $F_1$. Propositions 1, 2, 3 show $F_3, F_6, F_8$ cannot be respectively encoded as poly-size sMDD, smaT, segT.

The family $F_{13}$ can be encoded as a poly-size OMDD w.r.t. the order $h_1 < x_1 < \cdots < x_n$. Then $F_2$ and $F_5$ can be encoded as poly-size sMDD using the order $h_1 < x_1 < \cdots < x_n < h_2 < y_1 < \cdots < y_n$. The family $F_7$ can be used to model the 3-SAT problem, so $F_7$ is NP-hard. Correspondingly, it is NP-hard to encode $F_7$ as the representations in poly-size, 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 $F_9$ in poly-size. Then (Amilhastre et al. 2014) shows $F_{15}$ cannot be encoded as poly-size OMDDs, since an OMDD over Boolean variables can be regarded as an Ordered Binary Decision Diagram (OBDD) and DNF cannot be encoded as poly-size OBDD (Darwiche and Marquis 2002).

The family $F_{14}$ can be encoded as poly-size BCT (see Proposition 4) but not poly-size OMVD (see Proposition 5). $F_{15}$ is the family given by Theorem 6 in (Kadioglu and Sellmann 2010) which can be encoded as a poly-size CFG but not a BCT (see Proposition 7). The families $F_9, F_{10}$ can be regarded as special cases of $F_{14}$, so the family $F_{14}$ cannot be encoded as poly-size segT and smaT.

**Proposition 1.** There is no poly size sMDD encoding $F_3$.

**Proof.** Let $n > 10$ and $R$ be a sMDD encoding the family $F_3$ 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 $k \leq \lceil \frac{n}{2} \rceil + 1$ and $V$ is the set of variables $\{O_i[i < \frac{n}{2} + 2 \leq i \leq n]\}$. Given any subset $S$ of $V$, $\tau(S)$ denotes the tuple $(\{x_i, 1\}, \{x_i\}, 2, x_i \in S \setminus V)$, and $A(S)$ denotes the set $\{(h_1, a_0) \mid \{x_i, x_j\} \subseteq V, \{x_i, x_j\} \cap S = 1\}$ 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 $(\frac{n}{2} + 1)$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 $2^{\frac{n}{2} + 1}$ different subsets, hence, $R$ has at least $2^{\frac{n}{2} + 1}$ nodes.

Similarly, for $k \geq \lceil \frac{n}{2} \rceil + 1$, $R$ also has at least $2^{\frac{n}{2} + 1}$ nodes. So there is no poly size sMDD encoding $F_3$. 

\[\square\]
Proposition 2. There is no polysize smaT encoding $F_6$.

Proof. Let $R$ be a smaT encoding $F_6$. For any smart tuple $\tau$ in $R$ and $1 \leq n \leq n$, $\tau$ must include a binary constraint or unary constraint which makes sure that all variables in \{x_1, x_2, x_3\} 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 $\tau$ encodes at most $6^n$ tuples. The clauses used in $F_6$ do not share any variables, therefore, $|rel(R)| = 7^n$ and the number of smart tuples in $R$ is at least $\left(\frac{7}{6}\right)^n$. So there is no polysize smaT encoding the family $F_6$.

Proposition 3. There is no polysize segT encoding $F_8$.

Proof. Assume $n > 10$ and $R$ is a segT encoding $F_8$ and $t$ is a segmented tuple in $R$, where $|rel(R)| = 3(2^n - 1)$ and $|rel(t)| \neq 0$. 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 $\left\lfloor \frac{|X(t)|}{2} \right\rfloor$ variables $x_i$ such that $rel(t)[\{x_i\}] \leq 1$. For any nonempty $V \subseteq scp(R)$ and a tuple $\tau$ over $V$, the size of $\{\tau \in rel(R) \mid \tau \subseteq V\} = 2^{n-|V|}$.

If $|t| > \sqrt{n}$, then $|X(t)| \geq \sqrt{n}$ and $|rel(t)| \leq k = 2^{n-\frac{|U|}{2}}$. If $|t| \leq \sqrt{n}$, there is a constraint $c$ in $t$ and a polynomial $p$ such that $|scp(c)| \geq \sqrt{n}$ and $|rel(c)| \leq p$ and for any $\tau \in rel(c)$, $|\{\tau \in rel(t) \mid \tau \subseteq \tau \} | \leq 2^{n-\frac{|U|}{2}}$, so $|rel(t)| \leq pk$. So $|P| \geq \frac{3(2^{n-1})}{k} \geq \frac{2^{n-|U|}}{k}$ and there is no polysize segT encoding $F_8$.

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 $|D(v)| \leq d^{k+1}$ for all $v \in V$.

Proof. (Dechter and Pearl 1989) shows that there is a BCT $P_1=(H, C_1)$ such that $rel(P) = \{(\cup_{(a, b) \in \tau} \in rel(P_1)) \mid \forall h \in H \text{ and } x \in X \text{ such that } D(h) \subseteq U(var(h)) \text{ and } |var(h)| \leq k + 1, \text{ i.e. } |D(h)| \leq d^{k+1}\}$. Then rel($P_1$) can be encoded as a BCT $(V, C'')$ where $V = X \cup H$ and $C'' = C_1 \cup \{c_x \mid x \in X\}$ and $c_x$ is between $X$ and a variable $h \in H$ such that $x \in var(h)$ and $rel(c_x) = \{(x, a), (h, \tau) \mid \tau \in D(h) \text{ and } x_a \in \tau\}$. So $P$ can be encoded as a BCT $(V, C''')$ such that $|V| \leq 2|X|$ and $|D(v)| \leq d^{k+1}$ for all $v \in V$.

Proposition 5. There is no polysize OMVD encoding $F_{14}$.

Proof. Let $F$ be the CSPs over Boolean variables $X$ such that $|X| \geq 2^n$ where $w$ is the treewidth of the CSP. 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 $F_{14}$ Theorem 5 in (Razgon 2016) shows there is no polysize NROBP (non-deterministic read-once branching program) encoding $F$. So there is no polysize OMVD $R$ encoding $F'$ and $F_{14}$, 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 $F_{14}$.

Proposition 6. There is no polysize CFG encoding $F_9$.

Proof. Assume $R$ is a CFG w.r.t. any order $O$ over $\{v\} \cup X$ encoding $F_9$ and $O'$ is a subsequence of $O$ over $X$, where $X = \{x_1, \ldots, x_{2n}\}$. The interchange Lemma (Odden, Ross, and Winklmann 1985) shows that there is $T \subseteq sol(F(O'))$ and $W = \{O_k \mid k < l\}$ and $V = \{O_i \mid i \leq j\}$ and $U = \{O_k \mid k \leq 2n + 1\}$ such that (i) $|T| \geq \frac{n^n}{c_{rel}(2n+1)}$, and $\frac{n}{2} < i < n$, where $c_{rel}$ is the number of non-terminals used in the acyclic CFG $A^R$; and (ii) for any $\tau_1, \tau_2 \in T$, the tuples $\tau_1[W], \tau_2[V], \tau_1[U]$ and $\tau_2[W] \cup \tau_1[V] \cup \tau_2[U]$ are also in the constraint relation of $F_9$. If $v = a_0'$, then every variable in $V$ connects to a variable in $W$, therefore, $\tau_1[V]$ is equal to $\tau_2[V]$. For all tuples $\tau$ in $T$, $\tau[V]$ is common, thus, $|T| \leq n^{2n+1}$ and $c_{rel}(2n+1)^2 \geq n^{2n-1}$. $c_{rel}$ is not polynomial in the length $O(n^{2n})$ of the expression defining $F_9$, so there is no polysize CFG encoding $F_9$.

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 $\frac{|X|}{2}$ variables in $X$.

Proposition 7. There is no polysize BCT encoding $F_{15}$.

Proof. Let $R = (V, C)$ be a BCT encoding the family $F_{15}$. 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 $\frac{|X|}{2}$ variables $x_j$ where $j$ is an even integer.

$\{x_i = x_j \mid \frac{i}{2} \leq i \leq n \text{, i is odd, j is even}\}$ is the set of all $(l + 1)$ equalities used in the family $F_{15}$. For each variable $x_i$ with an odd $i$, there are at most $\frac{|X|}{2}$ variables $x_j$ with an even $j$ such that $x_i, x_j$ are in the same CC. Therefore, there are at least $\frac{2^{n+1}}{2} = (l + 1)(\frac{1}{2} + \frac{1}{2})$ 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 $S_{k}^{2k}, S_{k}^{2k+2}$ use at least $\frac{1}{2}$ 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 $\tau_1, \tau_2 \in \text{sol}(R)$ including $(x_{2k+1}, \#)$ and any equality $x_i = x_j$ in $E$, if $\tau_1[\{x_i\}] \neq \tau_2[\{x_i\}]$, then $\tau_1[\{x_i\}] \neq \tau_2[\{x_j\}]$, since the tuple $(\tau \setminus \tau_1[W]) \cup \tau_2[W]$ is not in sol(R), where $W$ is the set of variables of a CC in $com(v)$ such that $x_i \in W$. So $|D(v)| \geq 2^2$ and there is no polysize BCT encoding $F_{15}$.
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 $B$. 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, $F_{15}$ cannot be encoded as a polysize BCT, and BCT $\leq$ OMVD and BCT $\leq$ c-T, so $F_{15}$ 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:

- $A_1$: Proposition 9 shows $OMDD \leq sMDD$;
- $A_3$, $A_{14}$: see section and also (Amilhastre et al. 2014);
- $A_{13}$: Proposition 9 shows $OMDD \leq sMDD$;
- $A_4$, $A_8$: c-T is both $smaT$ and $segT$, so $c-T \leq smaT$, $segT$;
- $A_5$: c-T is a special case of $bsmaT$ and each baic smart tuple can be encoded as a polysize c-tuple, so $bsmaT=c-T$;
- $A_6$: each c-tuple can be encoded as a path in an OMVD;
- $A_7$: OMDD is a special case of OMVD;
- $A_9$: each short support can be encoded as a c-tuple;
- $A_{10}$: see (Jefferson and Nightingale 2013);
- $A_{11}$: 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 $\equiv sliT$.
- $A_{12}$: 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 $F_6$ column of Figure 2, $smaT$ is a $\bullet$ and $segT$, OMVD, BCT and CFG are $\checkmark$, therefore, there is no path from $smaT$ to $segT$, OMVD, BCT and CFG, which means $smaT \not\leq segT$, $smaT \not\leq OMVD$, $smaT \not\leq BCT$ and $smaT \not\leq CFG$. In addition, in the $F_{14}$ column of Figure 2, OMVD is a $\bullet$ and BCT, CFG are $\checkmark$, thus, OMVD $\not\leq BCT$ and OMVD $\not\leq CFG$. Similarly, the other ad-hoc constraints can also be verified based on the results in Figure 2.

So the results in Figure 3 hold.

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 $e$ be the only constraint in $C$ including $x$ and $scp(e) = \{x, y\}$.

If $|V| = k$, there is a polysize CFG $R$ encoding $sol(V \setminus \{x\}, C \setminus \{e\})$. Then a CFG encoding $sol(A)$ can be constructed by replacing each production $a \rightarrow b \in \Gamma$ with $a \rightarrow a b^{(y,a)}$ and $\{b^{(y,a)} \rightarrow b\} \in rel(c)$.

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 $\alpha$ in $\Gamma$ such that $scp(\alpha) \cap scp(A) = \emptyset$. So CFG $\leq BCT$.

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 $k = \lfloor \frac{k}{2} \rfloor + 1$ and $n = |scp(R)| > 10$ and for any node $v$ in the layer $L_k$ of $R$ and $a \in D(O_k)$, $R(v, a)$ 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 transpore 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 $a \in L_k$ and $a \in D(O_k)$ 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 $|L_k| \times |D(O_k)|$ times larger than $R$. So OMDD $\leq sMDD$. 


4109
### 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 polytime with the same order, so they satisfy the same operations. In addition, sIIT = table and bsmT = c-T, thus, sIIT and bsmT 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 (Amitha et al. 2014). We now prove the blue cells.

The family $F_3$ can be encoded as a conjunction of 2 segTs encoding $\wedge_{i=1}^{n} \{x_i \neq x_{i+1} \mid |x_{i+1}| = 1 \}$ and $\wedge_{i=1}^{n} \{x_i \neq x_{i+1} \mid |x_{i+1}| = 2 \}$, so segT does not satisfy $A \land B$.

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 $v_1, v_2 \in \text{scp}(c)$, $g(c, v_1) \neq g(c, v_2)$ if $v_1 \neq v_2$. Let $g_{c_1}$ be the constraint $\wedge \{h_c \neq v \mid v \neq a + 1 \}$ and $g_{c_2}$ be the constraint $\wedge \{h_c \neq v \mid v \neq a + 1 \}$ included in the expression of $F_{2}, g(c, v) = k$ for $k \in \{1, 2, 3\}$ and $a \in \{1, 1, 3\}$. Then the family $F_{7}$ can be modelled as $g_{c_1} \land g_{c_2} \land g_{c_3}$, where $g_{c_3}$ is a smart tuple which can be encoded as a smaT or smMD. $F_{2}$ is NP-hard, so the smaT, smMD, BCT, CFG constraints do not satisfy the $A \land B$ operation unless NP=NP. In addition, smMD satisfies the $\neg A$ operation (see Proposition 10), so the smMD constraint does not satisfy $A \land B$ and $A \lor B$ unless NP=NP.

Let $\tau$ 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 $\text{scp}(A) \setminus Y$ where $\text{rel}(c') = \{\tau_1 \in \text{rel}(c) \mid \tau_1[Y] \subseteq \tau\}$ and $\text{scp}(c') = \text{scp}(c)$. For any CFG $A$, $A[\tau]$ is encoded by removing all non-terminals $\alpha$ such that $\text{scp}(\alpha) \notin \text{rel}(\alpha) \cup \text{scp}(\alpha) \subseteq Y$ from $A^\alpha$. So BCT, CFG satisfy sCD. $F_{3}$ can be defined as $A_{\{\{y_k, 1\} \mid 1 \leq k \leq n\}}$ such that $A$ is a SMD encoding $F_{4}$, so SMD does not satisfy c-T.

Any BCT $A$ can also encoding $\text{rel}(A) \setminus \text{scp}(A) \setminus Y$ over $\text{SCP}(A) \setminus Y$. For any CFG $A$, $A[\neg Y]$ can be encoded as a CFG by removing all non-terminals $\alpha$ such that $\text{scp}(\alpha) \subseteq Y$ from the acyclic CFG $A^\alpha$. So BCT, CFG satisfy the SFO and FO operations. $F_{6}$ is a 3-SAT such that all clauses do not share any variables, thus, there is a smaT $A$ encoding a subset of $F_{7}$ with 1 smart tuple such that $A[\neg Y]$ encodes $F_{6}$ where $V = \{h_c \mid c \in F\}$. Then for any smaT $A$ and a variable $y \in \text{scp}(A), A[\neg y] = \bigvee_{a \in \text{D}(y)} A\{\{y, a\}\}$. So smT satisfies the SFO operation but not the FO operation.

**Proposition 10.** smMD satisfies $\neg A$.

**Proof.** Let $A$ be a smMD w.r.t. an order $O$ and $n = |\text{scp}(A)|$ and $k = \left\lceil \frac{n}{2} \right\rceil$. Then the negation of a smMD $A$ can be encoded as a smMD by reconstructing $A$ with:

1. for each $1 < i < k + 1$, adding a node $tt_i$ to $L_i$ if existing $j < i, v \in L_j$ and $a \in D(O_j)$ such that $\text{out}(v, a) = \emptyset$;
2. for each $k + 1 < i < n$, adding a node $tt_i$ to $L_i$ if existing $i < j, v \in L_j$, and $a \in D(O_{i-1})$ such that $\text{in}(v, a) = \emptyset$;
3. for each $1 \leq i \leq k$, $a \in D(O_i)$, and $v \in L_i$ such that $\text{out}(v, a) = \emptyset$, adding an arc labelled with the value $a$ pointing from $v$ to $tt_{i+1}$;
4. for each $k + 2 \leq i \leq n$, $a \in D(O_i), v \in L_{i+1}$ such that $\text{in}(v, a) = \emptyset$, 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 smMD 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 $F_{3}$ is $F_{1}$, so there is no polysize OMDD encoding $F_{2}$. For any representation $R$ encoding $F_{1}$ or $F_{3}$, $R[\{h_1, h_2, y_1, \ldots, y_n\}]$ encodes $F_{2}$. As table satisfies FO, thus, $F_{4}$ and $F_{5}$ cannot be encoded as polysize table.

---

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \lor B$</td>
<td>disjunction</td>
<td>${\tau \in U(\text{scp}(A) \cup \text{scp}(B)) \mid \tau[\text{scp}(A)] \notin \text{rel}(A) \lor \tau[\text{scp}(B)] \notin \text{rel}(B)}$</td>
</tr>
<tr>
<td>$\lor S$</td>
<td>big $</td>
<td>$</td>
</tr>
<tr>
<td>$A \land B$</td>
<td>conjunction</td>
<td>${\tau \in U(\text{scp}(A) \cap \text{scp}(B)) \mid \tau[\text{scp}(A)] \in \text{rel}(A) \land \tau[\text{scp}(B)] \in \text{rel}(B)}$</td>
</tr>
<tr>
<td>$\land S$</td>
<td>big conj.</td>
<td>$\land_{B \in S} R$</td>
</tr>
<tr>
<td>$\neg A$</td>
<td>negation</td>
<td>$U(\text{scp}(A)) \setminus \text{rel}(A)$</td>
</tr>
<tr>
<td>$A[\neg Y]$</td>
<td>SFO</td>
<td>$\text{rel}(A)[\text{scp}(A) \setminus {y}]$</td>
</tr>
<tr>
<td>$A[\tau]$</td>
<td>CD</td>
<td>${\tau' \mid \tau' \in \text{rel}(A), \tau'[Y] \subseteq \tau}$</td>
</tr>
</tbody>
</table>

### Table 4: Satisfied operations: $\checkmark$ means a representation satisfies an operation; $\bullet \circ$ means a representation does not satisfy an operation (operation unless NP=NP); ‘?’ is unknown. Reasons for unsatisfied operations are given with families.
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 poly-size sMDDs that respectively encode \( F_1 \) and \( \sum_{i=1}^n v_i = 1 \) using the order \( h_1 < x_1 < \cdots < x_i < \cdots < x_n < y_1 < \cdots < y_n \). There is no poly-size sMDD \( R \) encoding \( A \land B \) w.r.t. \( O \), otherwise we can get a poly-size sMDD encoding \( F_1 \) 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 \( A \land B \) for \( A \lor B \) 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 \( \alpha \rightarrow a \) in \( A^R \) such that \( scp(\alpha) = \{ y \} \) with the productions: \( \alpha \rightarrow a \beta \) (\( \beta \rightarrow a \beta \)) and \( \{ \beta \rightarrow b \beta \} \in \mathcal{D}(x) \). So CFGs using an order \( O \) can be extended to the CFGs w.r.t. \( O \) which satisfies the \( A \lor B \) and \( \lor S \) operations, since \( C \) is closed under union.

Propositions 11 and 12 show BCTs using the same graph satisfy the \( A \land B \) for \( A \lor B \) and \( \lor S \) operations.

**Proposition 11.** BCTs using the same graph satisfy \( A \land B \).

**Proof.** Let \( A, B \) be 2 BCTs using a graph \( (H \lor Y, E) \). The \( A \land B \) operation can be encoded as a BCT \( R = \{ (f^R(v) \mid v \in H \lor Y), \{ f^R(e) \mid e \in E \} \} \) by mapping the vertices \( v \) and edges \( e \) in the graph to variables \( f^R(v) \) and binary constraints \( f^R(e) \) such that \( \mathcal{D}(f^R(v)) = \{ f^A(v), f^B(v) \} \) for \( v \notin Y \) and \( scp(f^R(e)) = \{ f^R(v) \mid v \in e \} \) and \( rel(f^R(e)) = \{ \tau \in \mathcal{S}(f^A(e), f^B(e)) \mid \tau \in \mathcal{S}(v, a) \} \) for \( v \in Y \), otherwise \( \tau^R \) is \( \{ f^R(v), \tau \{ f^A(v), f^B(v) \} \} \).

Every tuple \( \tau_R \) in \( sol(R) \) corresponds to two tuples \( \tau_A \in sol(G(A)) \) and \( \tau_B \in sol(G(B)) \) such that \( \tau_R \) satisfies \( \{ v \mid v \notin Y \} \) and \( \{ v \mid v \notin Y \} \). Then \( R \) also uses the graph \( G \), so BCTs using the same graph satisfy \( A \land B \). \( \square \)

**Proposition 12.** BCTs using the same graph satisfy \( \lor S \).

**Proof.** Let \( S \) be a set of BCTs using \( (H \lor Y, E) \). \( \lor S \) can be encoded as a BCT \( \{ f^R(v) \mid v \in H \lor Y \}, \{ f^R(e) \mid e \in E \} \) such that \( scp(f^R(e)) = \{ f^R(v) \mid v \in e \} \) and \( rel(f^R(e)) = \{ \tau \in \mathcal{S}(f^A(e), f^B(e)) \mid \tau \in \mathcal{S}(v, a) \} \) for \( v \notin Y \), otherwise \( \tau^R \) is \( \{ f^R(v), \tau \{ f^A(v), f^B(v) \} \} \).

Every tuple \( \tau_R \) in \( sol(R) \) corresponds to two tuples \( \tau_A \in sol(G(A)) \) and \( \tau_B \in sol(G(B)) \) such that \( \tau_R \) satisfies \( \{ v \mid v \notin Y \} \) and \( \{ v \mid v \notin Y \} \). Then \( R \) also uses the graph \( G \), so BCTs using the same graph satisfy \( A \land B \). \( \square \)
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 $\text{BCT} < \text{shoT} < \text{table}$ and $\text{BCT} (\text{shoT}, c-T)$ using the same graph/order satisfies $A \land B$ and $\text{SFO}$, we can also use $\text{BCTs (shoT, c-Ts)}$ to encode the constraints.

We see that representations which do not satisfy $\text{SFO}$ or $\text{AOMDD}$ satisfies $\text{V A, IM, EQ, CT}$. So $\text{sMDD}$ satisfies $\text{EQ*, SE*}$. 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 $\{x_1, \ldots, x_n\}$ is defined as $\bigwedge_{i=1}^{n-1} L_1 \land \bigwedge_{j=1}^{n-1} \neg L_2$, where $L_1$ and $L_2$ are 2 languages, $L_2$ is defined by $L_2 \text{ w.r.t. } x_j < \cdots < x_{j+t-1}$ and $L_1$ is defined by $L_1 \text{ w.r.t. } x_i < x_{i+1}$.

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

Then we can also model the DFAs encoding $L_1, L_2$ as BCTs (NFAs,OMVDs) using the same graph/order, since they satisfy $A \land B$ and BCT<$\text{NEA=OMVD}<\text{DFAs}$.

The C&Lex$(X,Y,\text{Regular})$ Constraint (Katsirelos, Narodytska, and Walsh 2008). The constraint is defined as $c_1 \land c_2 \land c_3$ such that $c_1,c_2$ are DFAs and $c_3$ is the $\text{Lex}(X,Y)$ constraint (Frisch et al. 2002) where $X=\text{scp}(c_1)$ and $Y=\text{scp}(c_2)$ 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 $A \land B$ and OMVD<$\text{NEA=OMVD}$.

The Root Constraint (Bessiere et al. 2006). The Root constraint $\text{Roots([x_1, \ldots, x_n], S, T)}$ is NP-hard but it can be decomposed into the constraints $i \in S \leftrightarrow x_i \in T$, where $i \in [1, \ldots, n]$ 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 $c_1 \land c_2$, where $c_1 is \neg s_i \lor \bigvee_{j \in \text{DF}(x_i)} x_i = j \land t_j$ and $c_2 is \bigwedge_{j \in \text{DF}(x_i)} s_i \lor x_i \neq j \lor \neg t_j$ and $s_i, t_j$ are Boolean variables.

BCTs using the same graph satisfy the $\lor S$ and $A \land B$ operations, and the treewidth of $c_2$ is 2, thus, it is straightforward that $c_1 \land c_2$ can be encoded as a poly-size BCT.

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 $\{x_1, \ldots, x_n\}$ is defined as $\bigwedge_{i=1}^{n-1} L_1 \land \bigwedge_{j=1}^{n-1} \neg L_2$, where $L_1$ and $L_2$ are 2 languages, $L_2$ is defined by $L_2 \text{ w.r.t. } x_j < \cdots < x_{j+t-1}$ and $L_1$ is defined by $L_1 \text{ w.r.t. } x_i < x_{i+1}$.

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

Then we can also model the DFAs encoding $L_1, L_2$ as BCTs (NFAs,OMVDs) using the same graph/order, since they satisfy $A \land B$ and BCT<$\text{NEA=OMVD}$.

The C&Lex$(X,Y,\text{Regular})$ Constraint (Katsirelos, Narodytska, and Walsh 2008). The constraint is defined as $c_1 \land c_2 \land c_3$ such that $c_1,c_2$ are DFAs and $c_3$ is the $\text{Lex}(X,Y)$ constraint (Frisch et al. 2002) where $X=\text{scp}(c_1)$ and $Y=\text{scp}(c_2)$ 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 $A \land B$ and OMVD<$\text{DFAs}$.

The Root Constraint (Bessiere et al. 2006). The Root constraint $\text{Roots([x_1, \ldots, x_n], S, T)}$ is NP-hard but it can be decomposed into the constraints $i \in S \leftrightarrow x_i \in T$, where $i \in [1, \ldots, n]$ 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 $c_1 \land c_2$, where $c_1 is \neg s_i \lor \bigvee_{j \in \text{DF}(x_i)} x_i = j \land t_j$ and $c_2 is \bigwedge_{j \in \text{DF}(x_i)} s_i \lor x_i \neq j \lor \neg t_j$ and $s_i, t_j$ are Boolean variables.

BCTs using the same graph satisfy the $\lor S$ and $A \land B$ operations, and the treewidth of $c_2$ is 2, thus, it is straightforward that $c_1 \land c_2$ can be encoded as a poly-size 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 coping with the space of CSPs but not modelling CSPs. Furthermore, $F_3$ cannot be encoded as a poly-size AOMDD, since AOMDD reduces to just OMDD on encoding $F_3$ 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 poly-size 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 generically 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


