New Canonical Representations by Augmenting OBDDs with Conjunctive Decomposition (Extended Abstract)*

Yong Lai and **Dayou Liu**
College of Computer Science and Technology
Jilin University, China
{laiy, dyliu}@jlu.edu.cn

Minghao Yin
Department of Computer Science
Northeast Normal University, China
ymh@nenu.edu.cn

Abstract

We identify two families of canonical representations called ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ by augmenting ROBDD with two types of conjunctive decompositions. These representations cover the three existing languages ROBDD, ROBDD with as many implied literals as possible (ROBDD-L_∞), and AND/OR BDD. We introduce a new time efficiency criterion called rapidity which reflects the idea that exponential operations may be preferable if the language can be exponentially more succinct. Then we demonstrate that the expressivity, succinctness and operation rapidity do not decrease from ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ to ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$, and then to ROBDD$[\wedge_{\widehat{i+1}}]_{\mathcal{C}}$. We also demonstrate that ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$ ($i > 1$) and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ are not less tractable than ROBDD-L_∞ and ROBDD, respectively. Finally, we develop a compiler for ROBDD$[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ which significantly advances the compiling efficiency of canonical representations.

1 Introduction

Canonicity, an important property of knowledge compilation (KC) languages, provides equivalence tests with constant time complexity and plays a critical role in the performance of compiling methods [Darwiche, 2011; Van den Broeck and Darwiche, 2015]. The reduced ordered binary decision diagram (ROBDD) [Bryant, 1986] is one of the most influential canonical languages in the KC literature.

Despite its current success, ROBDD has a well-known weakness in succinctness, which reflects the explosion in size for many types of knowledge bases in practice. Deterministic, decomposable negation normal form (d-DNNF) [Darwiche, 2001] is a strict non-canonical superset of ROBDD, and most efficient compilers (e.g., OBDD compilers) can be seen as special d-DNNF compilers [Huang and Darwiche, 2007]. A recent trend in the KC field is to identify new canonical representations in d-DNNF to mitigate the size explosion problem of ROBDD without sacrificing its main advantages. Researchers has proposed many canonical languages, including AND/OR BDD (AOBDD) [Mateescu *et al.*, 2008], compressed and trimmed SDD over a fixed vtree $\mathcal{V}$ (CSDD$_{\mathcal{V}}$) [Darwiche, 2011], ROBDD with as many implied literals as possible (ROBDD-L_∞) [Lai *et al.*, 2013]. However, the corresponding compilers cannot yet compile many problems that the state-of-the-art d-DNNF compiler DSHARP can compile [Muise *et al.*, 2012]. Furthermore, the relationships between these canonical representations, which are indispensable for choosing appropriate representations in practical applications, are not well studied.

Decomposability is an important factor behind the strong succinctness and tractability of d-DNNF. The ideas of ROBDD-L_∞ and AOBDD are to use two special types of conjunctive decomposability to relax the linear variable ordering of ROBDD. We generalize these two types of $\wedge$-decompositions to propose bounded $\wedge$-decomposition parameterized by integer i ($\wedge_i$-decomposition), and $\wedge_i$-decomposition respecting tree $\mathcal{T}$ ($\wedge_{\mathcal{T},i}$-decomposition). Then we identify a family of canonical languages in d-DNNF called ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$ by imposing reducedness, the finest $\wedge_i$-decomposability, and ordered decision respecting a chain $\mathcal{C}$; and another family of canonical languages called ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ by imposing reducedness, the finest $\wedge_{\mathcal{T},i}$-decomposability, and ordered decision respecting $\mathcal{T}$. We demonstrate that these two families of languages cover the three previous languages ROBDD, AOBDD and ROBDD-L_∞, as depicted in Figure 1.

We evaluate the theoretical properties of the two families of canonical languages from four aspects, and the obtained results significantly extend the current KC map:

(a) We analyze the expressivity and demonstrate that ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$ is complete while ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ is incomplete. We also demonstrate that if $i \leq j$, ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ is not more expressive than ROBDD$[\wedge_{\widehat{\mathcal{T}},j}]_{\mathcal{T}}$ (resp. CSDD$_{\mathcal{V}}$).

(b) We analyze the succinctness and demonstrate that ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$ (resp. MODS) is strictly less succinct than ROBDD$[\wedge_{\widehat{j}}]_{\mathcal{C}}$ if $i < j$. We also demonstrate that ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_{\mathcal{T}}$ is at most as succinct as ROBDD$[\wedge_{\widehat{i}}]_{\mathcal{C}}$.

*This paper is an extended abstract of an article in the Journal of Artificial Intelligence Research [Lai *et al.*, 2017], and was supported by the National Natural Science Foundation of China under grants 61402195 and 61370156, the China Postdoctoral Science Foundation under grant 2014M561292, and Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education.

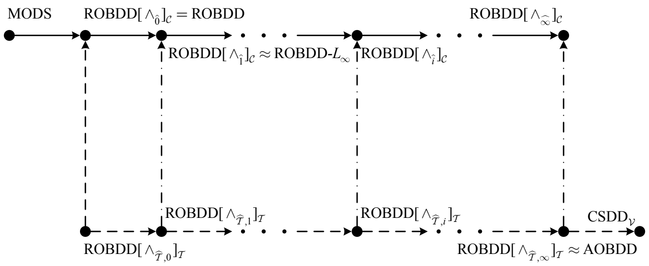

Figure 1: The relationship between many canonical representations in d-DNNF, where each formula in MODS is a set of models, $\mathcal{T}$ is not a chain, $\mathcal{C}$ is a topological order of $\mathcal{T}$, and $\mathcal{V}$ is the vtree corresponding to $\mathcal{T}$

(c) We analyze the time efficiency in terms of tractability. We demonstrate that ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is not less tractable than ROBDD. We also demonstrate that ROBDD$[\wedge_{\hat{i}}]_\mathcal{C}$ ($i \geq 2$) maintains the same tractability as ROBDD-L_∞, which is more tractable than d-DNNF.

(d) We also analyze the time efficiency in terms of a new notion called rapidity. We prove that each operation on ROBDD$[\wedge_{\hat{i}}]_\mathcal{C}$ (resp. ROBDD$[\wedge_{\widehat{\mathcal{T}},j}]_\mathcal{T}$ and MODS) is at most as rapid as the same operation on ROBDD$[\wedge_{\hat{j}}]_\mathcal{C}$.

For practical purposes, we developed an ROBDD$[\wedge_{\widehat{\infty}}]_\mathcal{C}$ compiler with high efficiency. Preliminary experimental results indicate that our compiler is significantly more efficient than the state-of-the-art compilers of ROBDD-L_∞ and CSDD$_\mathcal{V}$; moreover, only our compiler is comparable to D-SHARP in both compiling time and resulting sizes.

2 Basic Concepts

This paper uses x to denote a propositional or Boolean variable, and $PV = \{x_1, \ldots, x_n, \ldots\}$ to denote a countably infinite set of variables. A formula φ is constructed from constants $true$, $false$ and variables in PV using negation operator $\neg$, conjunction operator $\wedge$, disjunction operator $\vee$, equality operator $\leftrightarrow$ and decision operator $\psi \diamond_x \psi' = (\neg x \wedge \psi) \vee (x \wedge \psi')$, and we use $Vars(\varphi)$ to denote the set of variables appearing in φ.

We assume that PV is associated with some strict (partial) order $\prec$. We focus on the tree-structured orders which are defined as the ancestor-descendant relationships on trees of variables. Given a tree $\mathcal{T}$ over variables, we denote its depth by $dep(\mathcal{T})$, and the corresponding order by $\prec_\mathcal{T}$.

Definition 1 ($\wedge$-decomposition, $\wedge_i$-decomposition and $\wedge_\mathcal{T}$-decomposition)**.** Given a formula φ, a formula set Ψ is its $\wedge$-*decomposition*, iff $\varphi \equiv \bigwedge_{\psi \in \Psi} \psi$ and $\{Vars(\psi) : \psi \in \Psi\}$ partitions $Vars(\varphi)$. Ψ is *bounded* by an integer $0 \leq i \leq \infty$ ($\wedge_i$-decomposition) iff there exists at most one $\psi \in \Psi$ with $|Vars(\psi)| > i$. Ψ *respects* a tree $\mathcal{T}$ over variables ($\wedge_\mathcal{T}$-decomposition), iff any two formulas $\psi, \psi' \in \Psi$ satisfy that $Vars(\psi)$ and $Vars(\psi')$ are from two disjoint subtrees.

[Lai *et al.*, 2013] and [Mateescu *et al.*, 2008] implicitly discussed $\wedge_1$-decomposition and $\wedge_\mathcal{T}$-decomposition, respectively. Given two $\wedge$-decompositions Ψ and Ψ' of a formula, Ψ is *strict* iff $|\Psi| > 1$; and Ψ is *finer* than Ψ' iff $\{Vars(\psi) : \psi \in \Psi\}$ is a finer partition than $\{Vars(\psi) : \psi \in \Psi'\}$.

Proposition 1. *From the viewpoint of equivalence, each non-trivial formula has exactly one finest $\wedge_i$-decomposition (resp. $\wedge_\mathcal{T}$-decomposition).*

The finest $\wedge_i$-decomposition (resp. $\wedge_\mathcal{T}$-decomposition) is hereafter denoted by $\wedge_{\hat{i}}$-decomposition (resp. $\wedge_{\widehat{\mathcal{T}}}$-decomposition), and a $\wedge_{\widehat{\mathcal{T}}}$-decomposition bounded by integer i is denoted by $\wedge_{\widehat{\mathcal{T}},i}$-decomposition.

3 ROBDD$[\wedge_{\hat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$

We first define binary decision diagram with conjunctive decomposition (BDD$[\wedge]$):

Definition 2 (BDD$[\wedge]$)**.** A BDD$[\wedge]$ is a rooted DAG. Each vertex v is labeled with a symbol $sym(v)$. If v is a leaf, $sym(v) = \bot$ or $\top$; otherwise, $sym(v)$ is a variable or operator $\wedge$. Each internal vertex v has a set of children $Ch(v)$. For a vertex labeled with variable, $Ch(v) = \{lo(v), hi(v)\}$, where $lo(v)$ and $hi(v)$ are called *low* and *high* children, and are connected by dashed and solid arcs, respectively; for a $\wedge$-vertex, $\{\vartheta(w) : w \in Ch(v)\}$ is a strict $\wedge$-decomposition of $\vartheta(v)$. Each vertex represents a formula defined as follows:

$$\vartheta(v) = \begin{cases} false/true & sym(v) = \bot/\top; \\ \bigwedge_{w \in Ch(v)} \vartheta(w) & sym(v) = \wedge; \\ \vartheta(lo(v)) \diamond_{sym(v)} \vartheta(hi(v)) & \text{otherwise.} \end{cases}$$

The formula represented by the BDD$[\wedge]$ is defined as the one represented by its root.

Given two vertices, we say that they are *identical* with each other, if they are leaf vertices with the same symbol, or they are internal vertices with the same symbol and children. Next we define some constraints on BDD$[\wedge]$:

Definition 3 (constraints on BDD$[\wedge]$)**.** Given an integer i, a partial order $\prec$ over variables, and a tree $\mathcal{T}$ over variables,

- A BDD$[\wedge]$ is *ordered* over $\prec$ (OBDD$[\wedge]_\prec$) iff each $\diamond$-vertex u and its $\diamond$-descendant v satisfy $sym(u) \prec sym(v)$;
- A BDD$[\wedge]$ is *reduced* (RBDD$[\wedge]$) iff no two vertices are identical and no $\diamond$-vertex has two identical children;
- A BDD$[\wedge]$ is, respectively, $\wedge_i$-*decomposable* and $\wedge_{\mathcal{T},i}$-*decomposable* (BDD$[\wedge_i]$ and BDD$[\wedge_{\mathcal{T},i}]$) iff each $\wedge$-vertex is a $\wedge_i$-decomposition and a $\wedge_{\mathcal{T},i}$-decomposition;
- A BDD$[\wedge]$ is, respectively, $\wedge_{\hat{i}}$-*decomposable* and $\wedge_{\widehat{\mathcal{T}},i}$-*decomposable* (BDD$[\wedge_{\hat{i}}]$ and BDD$[\wedge_{\widehat{\mathcal{T}},i}]$) iff each $\wedge$-vertex is a $\wedge_{\hat{i}}$-decomposition and a $\wedge_{\widehat{\mathcal{T}},i}$-decomposition, and the $\wedge_{\hat{i}}$-decomposition and $\wedge_{\widehat{\mathcal{T}},i}$-decomposition of each $\diamond$-vertex v are $\{\vartheta(v)\}$.

In the subsequent sections, we focus on OBDD$[\wedge]_\prec$ where $\prec$ is tree-structured: the ancestor-descendant relationship $\prec_\mathcal{T}$ on a tree $\mathcal{T}$ and particularly $\prec_\mathcal{C}$ over a chain $\mathcal{C}$. We assume that $\mathcal{C}$ and $\mathcal{T}$ always satisfy $\prec_\mathcal{T} \subset \prec_\mathcal{C}$, and we sometimes use $\mathcal{C}$ and $\mathcal{T}$ to denote $\prec_\mathcal{C}$ and $\prec_\mathcal{T}$, respectively. We will analyze the canonicity, expressivity, and space-time efficiency of ROBDD$[\wedge_{\hat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$. Note that each ROBDD over $\mathcal{C}$ is an ROBDD$[\wedge_{\hat{0}}]_\mathcal{C}$, and the mutual transformation between an ROBDD-L_∞ over $\mathcal{C}$ (resp. AOBDD over $\mathcal{T}$) and the equivalent ROBDD$[\wedge_{\widehat{1}}]_\mathcal{C}$ (resp. ROBDD$[\wedge_{\widehat{\mathcal{T}},\infty}]_\mathcal{T}$) can be performed in linear time.

4 Canonicity and Expressivity

We first state that ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ is canonical and complete:

Theorem 1. *Fixing integer i and chain $\mathcal{C}$ over PV, there is exactly one ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ representing a given formula.*

ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is also canonical, but incomplete:

Theorem 2. *Fixing tree $\mathcal{T}$ over PV, there is at most one ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ to represent a given formula, and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is incomplete when $\mathcal{T}$ is not a chain.*

Due to the incompleteness of ROBDD$[\wedge_{\widehat{\mathcal{T}}}]_\mathcal{T}$, we draw the expressivity relation between different ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$:

Theorem 3. *Given a tree $\mathcal{T}$ over variables and two integers i and j, ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is at most as expressive as ROBDD$[\wedge_{\widehat{\mathcal{T}},j}]_\mathcal{T}$ if $i \leq j$.*

Given a vtree $\mathcal{V}$ corresponding to $\mathcal{T}$, ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is at most as expressive as CSDD$_\mathcal{V}$, because the former is a subset of the latter.

5 Succinctness

In this section, we first analyze the succinctness relationship between ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{j}}]_\mathcal{C}$, and we then analyze the succinctness relationship between ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$. Note that the standard definition of succinctness in the KC map only applies to complete languages. We can extend it to handle the case of incomplete languages by only comparing the sizes of formulas that can be represented in both languages.

We can present an algorithm (see Algorithm DECOMPOSE in [Lai *et al.*, 2017]) to transform an OBDD$[\wedge_i]_\mathcal{C}$ into the equivalent ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$. Along with some counter-examples, we can state the following succinctness results:

Theorem 4. *ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ is at most as succinct as ROBDD$[\wedge_{\widehat{j}}]_\mathcal{C}$ iff $i \leq j$.*

The above theorem indicates that ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ can further mitigate the size explosion problem of ROBDD from a theoretical perspective. We then analyze the succinctness relationship between ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ and ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$:

Theorem 5. *ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is as succinct as ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ if $i = 0$ or $dep(\mathcal{T}) < \infty$, and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is strictly less succinct than ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ otherwise.*

The succinctness relationship between ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ (resp. ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$) and the ones in $\{$ROBDD, ROBDD-L_∞, AOBDD$\}$ is immediate from Theorems 4–5. The succinctness relationship between ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ ($i \geq 1$) and CSDD$_\mathcal{V}$ is incomparable, because some class of circular bit-shift functions can be represented in ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ but not in CSDD$_\mathcal{V}$ in polysize [Pipatsrisawat, 2010]. Finally, ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is strictly more succinct than MODS if some variable in $\mathcal{T}$ has an infinite number of disjoint subtrees each of which has at least two vertices.

Table 1: Polytime queries and transformations of ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$, where $\mathcal{C}$ and $\mathcal{T}$ are over PV, $\surd$ means "satisfies", $\bullet$ means "does not satisfy", and $\circ$ means "does not satisfy unless P = NP"

L	**CO**	**VA**	**CE**	**IM**	**EQ**	**SE**	**CT**	**ME**
ROBDD$[\wedge_{\widehat{0}}]_\mathcal{C}$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ ($i > 0$)	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$
ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
L	**CD**	**FO**	**SFO**	**$\wedge$C**	**$\wedge$BC**	**$\vee$C**	**$\vee$BC**	**$\neg$C**
ROBDD$[\wedge_{\widehat{0}}]_\mathcal{C}$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$	$\bullet$	$\surd$	$\surd$
ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ ($i > 0$)	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$
ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ ($dep(\mathcal{T}) < \infty$)	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$
ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ ($dep(\mathcal{T}) = \infty$)	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$	$\surd$	$\surd$

6 Operating Efficiency

We now analyze the time efficiency of operating ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ in terms of tractability and a new notion called *rapidity*. We can write an operation as a set of triples with the form $(\varphi_1, \ldots, \varphi_n, \alpha, \beta)$, where $\varphi_1, \ldots, \varphi_n$ represent the input formulas, α represents the supplementary input, and β represents the output.

6.1 Tractability Evaluation

We examine the tractability of ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ with respect to the criteria proposed in [Darwiche and Marquis, 2002] in Table 1.

According to the results in Table 1, we know that ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ ($i \geq 2$) is as tractable as ROBDD-L_∞. In other words, compared with ROBDD-L_∞ over $\mathcal{C}$, ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ indeed improves the succinctness under the premise of maintaining the same tractability. Moreover, we know that ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is at least as tractable as ROBDD; in particular, ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ even has more tractability than ROBDD if $dep(\mathcal{T}) < \infty$. Finally, we know that ROBDD$[\wedge_{\widehat{i}}]_\mathcal{C}$ is more tractable than d-DNNF, and ROBDD$[\wedge_{\widehat{\mathcal{T}},i}]_\mathcal{T}$ is more tractable than CSDD$_\mathcal{V}$ [Van den Broeck and Darwiche, 2015].

6.2 New Perspective on Time Efficiency

Due to distinct succinctness, it can sometimes be insufficient to compare the time efficiency of two languages solely by comparing their tractability. Consider an operation OP, and two languages L and L$'$ such that L does not satisfy **OP** but L$'$ satisfy **OP**. Assume that the number of basic arithmetic operations involved in performing OP on $(\varphi_1, \ldots, \varphi_n, \alpha)$ (resp. $(\varphi'_1, \ldots, \varphi'_n, \alpha)$) be 2^m (resp. n), where $\varphi_i \in$ L (resp. $\varphi'_i \in$ L$'$), and $m = |\alpha| + \sum_{1 \leq i \leq n} |\varphi_i|$ (resp. $n = |\alpha| + \sum_{1 \leq i \leq n} |\varphi'_i|$). Then, performing OP on $(\varphi_1, \ldots, \varphi_n, \alpha)$ can be exponentially (in m) more time-consuming than performing OP on $(\varphi'_1, \ldots, \varphi'_n, \alpha)$ when $n = 2^{m^2}$. To overcome this problem, we define a new notion to compare time efficiency from a different perspective, supplementing the concept of tractability:

Definition 4 (rapidity)**.** An operation OP on a canonical language L$_1$ is *at most as rapid as OP* on another canonical language L$_2$ (L$_1 \leq_r^{OP}$ L$_2$), iff for each algorithm ALG performing OP on L_1, there exists some polynomial p and some algorithm $\textsc{Alg}'$ performing OP on L_2 such that for every valid input $(\varphi_1, \ldots, \varphi_n, \alpha)$ of OP on L_1 and every valid input $(\varphi'_1, \ldots, \varphi'_n, \alpha)$ of OP on L_2 satisfying $\varphi_i \equiv \varphi'_i$, $\textsc{Alg}'(\varphi'_1, \ldots, \varphi'_n, \alpha)$ can be done in time $p(t + |\varphi_1| + \cdots + |\varphi_n| + |\alpha|)$, where α is any element of supplementary information and t is the running time of $\textsc{Alg}(\varphi_1, \ldots, \varphi_n, \alpha)$.

Assume that an operation OP satisfies $\mathrm{L}_1 \leq_r^{OP} \mathrm{L}_2$. Let $(\varphi_1, \ldots, \varphi_n, \alpha)$ be a valid input of OP on L_1 and $(\varphi'_1, \ldots, \varphi'_n, \alpha)$ be a valid input of OP on L_2, where $\varphi_i \equiv \varphi'_i$ for $1 \leq i \leq n$. We know that if performing OP on $(\varphi_1, \ldots, \varphi_n, \alpha)$ can be done in polytime, then performing OP on $(\varphi'_1, \ldots, \varphi'_n, \alpha)$ can also be done in polytime in $|\alpha| + \sum_{1 \leq i \leq n} |\varphi_i|$. Therefore, for applications needing canonical languages, we suggest that users choose a language by the following steps rather than by the traditional viewpoint of the KC map: first, identify the set $\mathcal{L}$ of canonical languages meeting the expressivity requirement; second, identify the set $\mathcal{OP}$ of necessary operations and identify the subset $\mathcal{L}'$ of $\mathcal{L}$ meeting the tractability requirement; third, add each language $\mathrm{L} \in \mathcal{L}$ satisfying $\exists \mathrm{L}' \in \mathcal{L}' \forall OP \in \mathcal{OP}.\mathrm{L}' \leq_r^{OP} \mathrm{L}$ to $\mathcal{L}'$; and finally, choose one of the most succinct languages in $\mathcal{L}'$.

We can propose algorithms (see Algorithms ConvertDown and ConvertTree in [Lai *et al.*, 2017]) which respectively transform $\text{ROBDD}[\wedge_{\widehat{j}}]_{\mathcal{C}}$ and $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$ into $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$ and $\text{ROBDD}[\wedge_{\widehat{\mathcal{T},i}}]_{\mathcal{T}}$ $(i \leq j)$ and whose time complexities are polynomial in the sizes of outputs. Then we can obtain the following rapidity results:

Theorem 6. *Given two integers i and j, a chain $\mathcal{C}$ and a tree $\mathcal{T}$ over variables, and an operation OP, MODS $\leq_r^{OP}$ $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}} \leq_r^{OP} \text{ROBDD}[\wedge_{\widehat{j}}]_{\mathcal{C}}$ if $i \leq j$; and $\text{ROBDD}[\wedge_{\widehat{\mathcal{T},i}}]_{\mathcal{T}} \leq_r^{OP} \text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$.*

It was mentioned that for the operation OP corresponding to **SE**, **SFO**, $\wedge$**BC** or $\vee$**BC**, OP on $\text{ROBDD}[\wedge_{\widehat{0}}]_{\mathcal{C}}$ can be performed in polytime but OP on $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$ $(i > 0)$ cannot be performed in polytime unless $\mathrm{P} = \mathrm{NP}$. Therefore, if we only consider the tractability of OP, it may create the illusion that the time efficiency of performing OP on $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$ is pessimistically lower than that of performing OP on $\text{ROBDD}[\wedge_{\widehat{0}}]_{\mathcal{C}}$. Actually, OP on $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$ can also be performed in polytime in the sizes of equivalent $\text{ROBDD}[\wedge_{\widehat{0}}]_{\mathcal{C}}$s. According to this new perspective, an application requiring OP prefers $\text{ROBDD}[\wedge_{\widehat{i}}]_{\mathcal{C}}$ to $\text{ROBDD}[\wedge_{\widehat{0}}]_{\mathcal{C}}$.

7 Preliminary Experimental Results

We developed an $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler, and compare it with three compilers of two canonical languages ROBDD-L_∞ and $\text{CSDD}_{\mathcal{V}}$ and a non-canonical language d-DNNF. The three canonical languages can be seen as subsets of d-DNNF. The state-of-the-art ROBDD-L_∞, $\text{CSDD}_{\mathcal{V}}$ and d-DNNF compilers are reported in [Lai *et al.*, 2013; Oztok and Darwiche, 2015; Muise *et al.*, 2012], and called BDDjLu, miniC2D and DSharp, respectively. Individual runs were limited to a one-hour time-out. Table 2 shows the overall performance of the four compilers over the eight domains. The experimental results show that the $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler outperformed both BDDjLu and miniC2D on three domains; and it succeeded in 18 (resp. 30) instances more than BDDjLu (resp. miniC2D). The $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler and DSharp outperformed each other on two domains. However, DSharp succeeded in six more instances than the $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler, since the latter was relatively inefficient in sortnet. The reason behind the inefficiency of $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler in sortnet is that the ordering heuristic has a negative effect on this domain. Specifically, the $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler succeeded in all instances in sortnet when we used the lexicographical order of variables.

Table 2: Comparative compiling performance between $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$, ROBDD-$L_\infty$, $\text{CSDD}_{\mathcal{V}}$ and d-DNNF

domain (#)	$\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$	ROBDD-L_∞	$\text{CSDD}_{\mathcal{V}}$	d-DNNF
Bejing (16)	**6**	5	4	5
blocksworld (7)	**7**	**7**	6	**7**
emptyroom (28)	**28**	**28**	**28**	**28**
flat200 (100)	**100**	**100**	**100**	**100**
grid (33)	31	16	11	**33**
II (41)	**15**	14	9	13
iscas89 (35)	**24**	23	**24**	**24**
sortnet (12)	5	5	4	**12**
total (261)	216	198	186	**222**

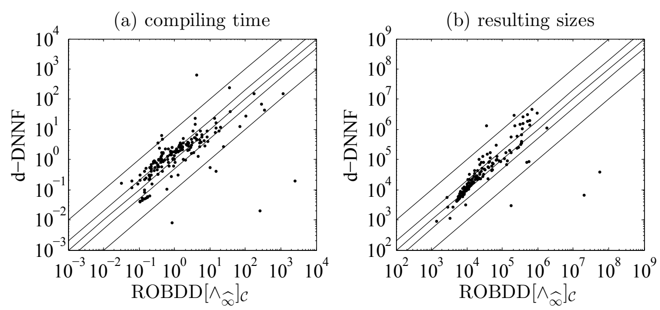

Figure 2: Compiling time and resulting sizes of non-trivial instances using d-DNNF and $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compilers

Figure 2 analyzes the detailed compiling time and resulting size performance between our $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler and DSharp across the eight domains in Table 2. The experimental results show that the performance of the $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler is comparable with that of DSharp.

8 Conclusions

We proposed two families of canonical languages, and analyzed their theoretical properties in terms of the existing criteria of expressivity, succinctness and tractability, as well as the new criterion rapidity. These results provide an important complement to the existing KC map, and the notion of rapidity sheds new light on identifying more succinct canonical representations without the worry of losing the tractability of KC languages. We also developed an efficient $\text{ROBDD}[\wedge_{\widehat{\infty}}]_{\mathcal{C}}$ compiler, which significantly advances the state-of-the-art of compiling efficiency of canonical representations, and has compiling efficiency even comparable with that of DSharp.

References

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

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

[Darwiche, 2011] Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 819–826, 2011.

[Huang and Darwiche, 2007] Jinbo Huang and Adnan Darwiche. The language of search. *Journal of Artificial Intelligence Research*, 29:191–219, 2007.

[Lai *et al.*, 2013] Yong Lai, Dayou Liu, and Shengsheng Wang. Reduced ordered binary decision diagram with implied literals: A new knowledge compilation approach. *Knowledge and Information Systems*, 35(3):665–712, 2013.

[Lai *et al.*, 2017] Yong Lai, Dayou Liu, and Minghao Yin. New canonical representations by augmenting OBDDs with conjunctive decomposition. *Journal of Artificial Intelligence Research*, 58:in press, 2017.

[Mateescu *et al.*, 2008] Robert Mateescu, Rina Dechter, and Radu Marinescu. AND/OR Multi-Valued Decision Diagrams (AOMDDs) for Graphical Models. *Journal of Artificial Intelligence Research*, 33:465–519, 2008.

[Muise *et al.*, 2012] Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck, and Eric I. Hsu. Dsharp: Fast d-DNNF compilation with sharpSAT. In *Proceedings of the 25th Canadian Conference on Artificial Intelligence*, pages 356–361, 2012.

[Oztok and Darwiche, 2015] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3141–3148, 2015.

[Pipatsrisawat, 2010] Thammanit Pipatsrisawat. *Reasoning with Propositional Knowledge: Frameworks for Boolean Satisfiability and Knowledge Compilation*. PhD thesis, University of California, Los Angeles, 2010.

[Van den Broeck and Darwiche, 2015] Guy Van den Broeck and Adnan Darwiche. On the role of canonicity in knowledge compilation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1641–1648, 2015.