

Towards a Knowledge Compilation Map for Heterogeneous Representation Languages

Hélène Fargier¹ Pierre Marquis² Alexandre Niveau²

1. IRIT-CNRS, Toulouse, France — fargier@irit.fr

2. CRIL-CNRS, Lens, France — [{marquis,niveau}@cril.fr">{marquis,niveau}@cril.fr](mailto)

IJCAI — August 6th, 2013



Introductory Example

- A product configuration problem: customized tee-shirts. [Had08]
- Parameters:
 - print – “Men in Black” (MiB) or “Save the Whales” (StW)
 - color – black or blue
 - size – small or large
 - sleeves – with or without

[Had08] Tarik Hadzic, Esben Rune Hansen, and Barry O’Sullivan. “On Automata, MDDs and BDDs in Constraint Satisfaction”. In: *Proceedings of the ECAI Workshop on Inference methods based on Graphical Structures of Knowledge (WIGSK)*. 2008

Introductory Example

- A product configuration problem: customized tee-shirts. [Had08]
- Parameters:
 - print – “Men in Black” (MiB) or “Save the Whales” (StW)
 - color – black or blue
 - size – small or large
 - sleeves – with or without
- Rules:
 - “MiB” tee-shirts must be black
 - “StW” picture does not fit on small-sized tee-shirts
 - large-sized tee-shirts cannot be sleeveless
- the program must be able to tell whether each choice respects the rules

[Had08] Tarik Hadzic, Esben Rune Hansen, and Barry O’Sullivan. “On Automata, MDDs and BDDs in Constraint Satisfaction”. In: *Proceedings of the ECAI Workshop on Inference methods based on Graphical Structures of Knowledge (WIGSK)*. 2008

Problem

- Configurable product → **Constraint Satisfaction Problem (CSP)**

$$\left\{ \begin{array}{l} \textit{print} = \textit{MiB} \quad \rightarrow \quad \textit{color} = \textit{black} \\ \textit{print} = \textit{StW} \quad \rightarrow \quad \textit{size} > S \\ \textit{size} = S \quad \vee \quad \textit{sleeves} = \textit{with} \end{array} \right.$$

- each variable corresponds to a choice
- each solution is a feasible configuration

Problem

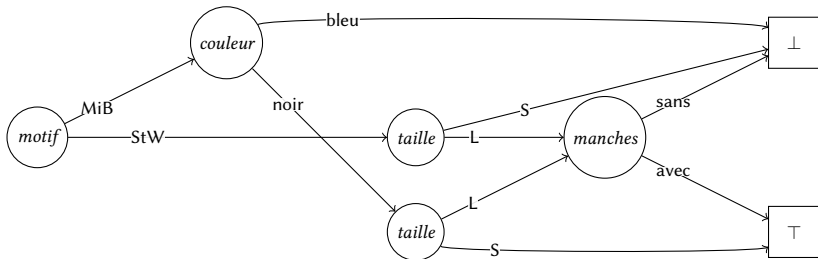
- Configurable product → **Constraint Satisfaction Problem (CSP)**

$$\left\{ \begin{array}{l} \textit{print} = \textit{MiB} \quad \rightarrow \quad \textit{color} = \textit{black} \\ \textit{print} = \textit{StW} \quad \rightarrow \quad \textit{size} > S \\ \textit{size} = S \quad \vee \quad \textit{sleeves} = \textit{with} \end{array} \right.$$

- each variable corresponds to a choice
- each solution is a feasible configuration
- Configuration process:
 - each choice fixes a value for some variable
 - is the CSP still **consistent**?
- NP-complete problem... but the user doesn't want to wait too long after each choice

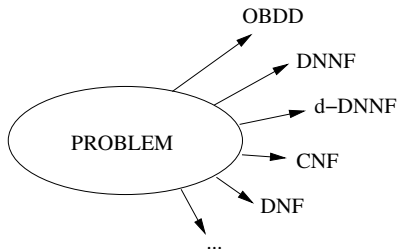
A Solution: Knowledge Compilation

- The CSP is a **fixed part** of the problem
- **Compile** this CSP into a suitable data structure, such as an OBDD:



- Assigning values to variables (conditioning) and checking consistency are **tractable** operations on OBDDs
- the user's wait is reduced

Choosing a Compilation Language



- What is the best language for my application?
- use the **knowledge compilation map** [Dar02]
- Compares languages according to two criteria:
 - 1 efficiency of operations
 - 2 succinctness

Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary **queries** and **transformations**

L	CO (consistency)	VA (validity)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○
DNNF	✓	○	✓	○	○	○	○	✓
BDD	○	○	○	○	○	○	○	○
FBDD	✓	✓	✓	✓	?	○	✓	✓
OBDD	✓	✓	✓	✓	✓	○	✓	✓
DNF	✓	○	✓	○	○	○	○	✓
CNF	○	✓	○	✓	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	$\wedge C$ (conjunction)	$\wedge BC$ (bounded conj.)	$\vee C$ (disjunction)	$\vee BC$ (bounded disj.)	$\neg C$ (negation)
NNF	✓	○	✓	✓	✓	✓	✓	✓
DNNF	✓	✓	✓	○	○	✓	✓	○
BDD	✓	○	✓	✓	✓	✓	✓	✓
FBDD	✓	○	✓	●	○	●	○	✓
OBDD	✓	●	○	●	○	●	○	✓
DNF	✓	✓	✓	●	✓	✓	✓	●
CNF	✓	○	✓	✓	✓	●	✓	●

- ✓ polynomial
- not polynomial unless $P = NP$
- not polynomial

Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary **queries** and **transformations**

L	CO (consistency)	VA (validity)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○
DNNF	✓	○	✓	○	○	○	○	✓
BDD	○	○	○	○	○	○	○	○
FBDD	✓	✓	✓	✓	?	○	✓	✓
OBDD	✓	✓	✓	✓	✓	○	✓	✓
DNF	✓	○	✓	○	○	○	○	✓
CNF	○	✓	○	✓	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	$\wedge C$ (conjunction)	$\wedge BC$ (bounded conj.)	$\vee C$ (disjunction)	$\vee BC$ (bounded disj.)	$\neg C$ (negation)
NNF	✓	○	✓	✓	✓	✓	✓	✓
DNNF	✓	✓	✓	○	○	✓	✓	○
BDD	✓	○	✓	✓	✓	✓	✓	✓
FBDD	✓	○	✓	●	○	●	○	✓
OBDD	✓	●	○	●	○	●	○	✓
DNF	✓	✓	✓	●	✓	✓	✓	●
CNF	✓	○	✓	✓	✓	●	✓	●

- ✓ polynomial
- not polynomial unless $P = NP$
- not polynomial

Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary **queries** and **transformations**

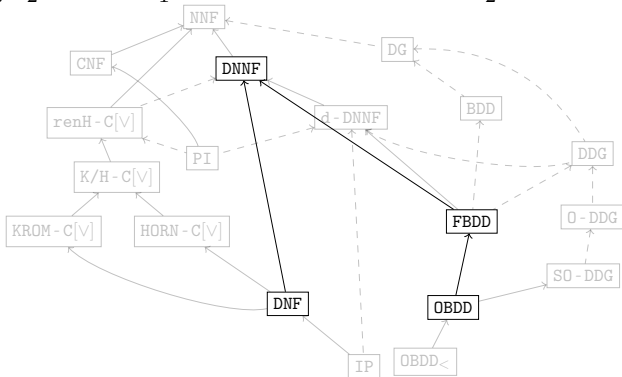
L	CO (consistency)	VA (validity)	CE (clause entailmt.)	IM (implicant check)	EQ (equivalence)	SE (entailment)	CT (model count)	ME (model enum.)
NNF	○	○	○	○	○	○	○	○
DNNF	✓	○	✓	○	○	○	○	✓
BDD	○	○	○	○	○	○	○	○
FBDD	✓	✓	✓	✓	?	○	✓	✓
OBDD	✓	✓	✓	✓	✓	○	✓	✓
DNF	✓	○	✓	○	○	○	○	✓
CNF	○	✓	○	✓	○	○	○	○

L	CD (conditioning)	FO (forgetting)	SFO (single forg.)	$\wedge C$ (conjunction)	$\wedge BC$ (bounded conj.)	$\vee C$ (disjunction)	$\vee BC$ (bounded disj.)	$\neg C$ (negation)
NNF	✓	○	✓	✓	✓	✓	✓	✓
DNNF	✓	✓	✓	○	○	✓	✓	○
BDD	✓	○	✓	✓	✓	✓	✓	✓
FBDD	✓	○	✓	●	○	●	○	✓
OBDD	✓	●	○	●	○	●	○	✓
DNF	✓	✓	✓	●	✓	✓	✓	●
CNF	✓	○	✓	✓	✓	●	✓	●

- ✓ polynomial
- not polynomial unless $P = NP$
- not polynomial

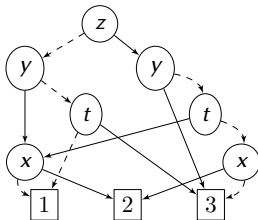
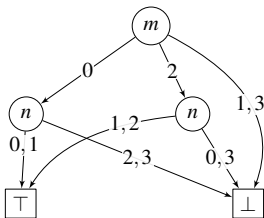
Knowledge Compilation Map: Succinctness

- Succinctness relation: orders languages w.r.t. their ability to represent knowledge compactly
- $L_1 \leq_s L_2$ means “ L_1 is at least as succinct as L_2 ”



Beyond Boolean Languages

- The map is drawn for lots of languages representing **Boolean functions over Boolean variables**
- There exists maps for languages with **multivalued** variables (family of MDDs) or **continuous** variables



- Other languages, representing functions with **non-Boolean values** (ADDs)
- How to compare these heterogeneous languages? How to unify all these maps?

Plan

- 1 Knowledge Compilation
- 2 Representation Languages**
- 3 Comparing Heterogeneous Languages
- 4 Result Inheritance

Languages of the Classical Compilation Map

- In the classical compilation map, the notion of “language” designates a **formal language**:
 - A propositional formula is a **word** over the alphabet $P_S \cup \{\vee, \wedge, \neg, (,)\}$
 - It is in CNF if it verifies some specific properties
 - The CNF language is **the set of all CNFs**
 - The notion of “language” concerns only syntax
- the semantics is implicitly given by the **interpretation function** of propositional formulæ

Limitations

- This notion of language is limited:
 - implicit interpretation function
 - implicit variable domains
- Easily adaptable to other families of data structures...
- ... but implicit aspects prevent a **unified** presentation
- We need a more general notion

Representation Language

- Definition of a **representation language**, as general as possible
- Universe of discourse \mathfrak{U} : contains all objects that we could intend to represent (Boolean functions, real functions, etc.)
- Generic alphabet Σ : no *a priori* restriction on formulæ $\varphi \in \Sigma^*$

Definition

A representation language is a pair $L = \langle \Phi_L, \mathcal{I}_L \rangle$, where

- Φ_L is the **syntax** of L : $\Phi_L \subseteq \Sigma^*$;
- \mathcal{I}_L is the **semantics** of L : $\mathcal{I}_L: \Sigma^* \rightarrow \mathfrak{U}$ (partial function, defined at least on all formulæ in Φ_L).

Examples

- Language of propositional logic: $\text{PROP} = \langle \Phi_{\text{PROP}}, \mathcal{I}_{\text{PROP}} \rangle$
 - Φ_{PROP} : set of well-formed propositional formulæ
 - $\mathcal{I}_{\text{PROP}}$: classical interpretation function
- $\text{CNF} = \langle \Phi_{\text{CNF}}, \mathcal{I}_{\text{PROP}} \rangle$, with Φ_{CNF} the set of CNFs
- $\text{HORN-C} = \langle \Phi_{\text{HORN-C}}, \mathcal{I}_{\text{PROP}} \rangle$, with $\Phi_{\text{HORN-C}}$ the set of Horn-CNFs
- $\text{OMDD} = \langle \Phi_{\text{OMDD}}, \mathcal{I}_{\text{MDD}} \rangle$
 - Φ_{OMDD} : set of ordered MDDs
 - \mathcal{I}_{MDD} : interpretation function of multivalued decision diagrams

Interpretation Space

- Semantics of L: way of **interpreting** some formulæ of Σ^*
- Associates with each formula $\varphi \in \Phi_L$ its interpretation $\llbracket \varphi \rrbracket_L$

Interpretation Space

- Semantics of L: way of **interpreting** some formulæ of Σ^*
 - Associates with each formula $\varphi \in \Phi_L$ its interpretation $\llbracket \varphi \rrbracket_L$
 - ... but it also interprets other formulæ
(semantics of CNF: $\mathcal{I}_{\text{PROP}}$, interprets also DNFs, for example)
- **interpretation space** Ω_L : set of all objects represented by the semantics of L
- Example : $\Omega_{\text{PROP}} = \Omega_{\text{CNF}} = \Omega_{\text{HORN-C}} =$ set of Boolean functions over Boolean variables

Interpretation Space

- Semantics of L: way of **interpreting** some formulæ of Σ^*
 - Associates with each formula $\varphi \in \Phi_L$ its interpretation $\llbracket \varphi \rrbracket_L$
 - ... but it also interprets other formulæ
(semantics of CNF: $\mathcal{I}_{\text{PROP}}$, interprets also DNFs, for example)
- **interpretation space** Ω_L : set of all objects represented by the semantics of L
- Example : $\Omega_{\text{PROP}} = \Omega_{\text{CNF}} = \Omega_{\text{HORN-C}} =$ set of Boolean functions over Boolean variables
 - **Completeness** of L: relative to its interpretation space
(CNF is complete, HORN-C is incomplete)

Plan

- 1 Knowledge Compilation
- 2 Representation Languages
- 3 Comparing Heterogeneous Languages**
- 4 Result Inheritance

Encoding MDDs into BDDs

- In practice, MDDs are often compiled into BDDs
- Use of classical encodings (also used to go from CSP to SAT [Wal00; Pre04])
 - Direct encoding: one Boolean variable per multivalued variable and per value in the domain
 - Multivalued encoding: like the direct encoding, but no “at-most-one” constraint
 - Log encoding: Boolean variables used as bits
- Encoding an MDD into a BDD is polynomial

[Wal00; Pre04] Toby Walsh. “SAT v CSP”. In: *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*. 2000, pp. 441–456; Steven Prestwich. “Local Search on SAT-Encoded Colouring Problems”. In: *Theory and Applications of Satisfiability Testing* (2004), pp. 26–29

Translatability of MDD into BDD

- MDDs can thus be “translated” into BDDs in polynomial time
- One would like to write $\text{MDD} \geq_p \text{BDD}$...
- But it is not the case: $\text{MDD} \not\geq_p \text{BDD}$, because they represent different kinds of functions
- The classical relation of polynomial translatability requires languages to have the same **interpretation space**
- We would like the compilation map to take **translations** into account

Translation

- We extend classical comparison relations
 - Possibility of using a **semantical correspondence** between interpretation spaces: $\mathcal{T} \subseteq \Omega_{L_1} \times \Omega_{L_2}$
- indicates objects considered as “equivalent”
- Examples: direct encoding \mathcal{T}_{dir} , multivalued encoding \mathcal{T}_{multi} , log encoding \mathcal{T}_{log}
 - \mathcal{T} induces a syntactic **translation** between formulæ of L_1 and formulæ of L_2

Extended Polynomial Translatability

- If there exists a polynomial algorithm transforming any formula φ_1 of L_1 into a formula φ_2 of L_2 such that $\llbracket \varphi_1 \rrbracket_{L_1} \mathcal{T} \llbracket \varphi_2 \rrbracket_{L_2}$, then L_1 is said to be **polynomially translatable into L_2 modulo \mathcal{T}**
 - We denote it as $L_1 \geq_p^{\mathcal{T}} L_2$
- Generalization of the classical polynomial translatability:
 $L_1 \geq_p L_2$ corresponds to $L_1 \geq_p^{\text{Id}} L_2$
- We also extend the **succinctness** and **expressiveness** relations to the use of a correspondence: $L_1 \geq_s^{\mathcal{T}} L_2$ and $L_1 \geq_e^{\mathcal{T}} L_2$

Examples

- Thanks to the extended relations, one can compare heterogeneous languages:
 - $\text{MDD} \geq_p^{\mathcal{T}_{dir}} \text{BDD}$ and $\text{MDD} \geq_p^{\mathcal{T}_{log}} \text{BDD}$
 - $\text{MDD} \not\leq_s^{\mathcal{T}_{dir}} \text{CNF}$
- One can also compare homogeneous languages of incomparable expressiveness (e.g., HORN-C and AFF), via a well-chosen semantical correspondence
- One can extend succinctness results from one family of languages to another via some translation:

$$\text{BDD} <_s \text{OBDD}$$



$$\text{MDD} <_s \text{OMDD}$$

Plan

- 1 Knowledge Compilation
- 2 Representation Languages
- 3 Comparing Heterogeneous Languages
- 4 Result Inheritance**

Polynomial Translatability and Operations

- The classical polynomial translatability allows one to easily infer results about queries and transformations
 - $\text{MODS} \geq_p \text{OBDD}$
 - ⇒ MODS satisfies all queries that OBDD satisfies
 - $\text{NNF} \sim_p \text{PROP}$
 - ⇒ NNF and PROP satisfy the exact same set of queries and transformations
- What properties of this kind hold on languages “equivalent modulo some translation”, like OBDD and OMDD?

Query Inheritance

- Classical case: if $L_1 \geq_p L_2$, then all queries satisfied by L_2 are satisfied by L_1 .
- Extended case: suppose $L_1 \geq_p^T L_2$.
What can we say about queries satisfied by L_1 ?

Query Inheritance

- Classical case: if $L_1 \geq_p L_2$, then all queries satisfied by L_2 are satisfied by L_1 .

- Extended case: suppose $L_1 \geq_p^{\mathcal{T}} L_2$.

What can we say about queries satisfied by L_1 ?

→ **Nothing** in the general case: it depends on the \mathcal{T} used

- Let L_2 be a language satisfying CT
- \mathcal{T}_{dir} maintains the number of models, so if $L_1 \geq_p^{\mathcal{T}_{dir}} L_2$ holds, then L_1 also satisfies CT
- \mathcal{T}_{multi} does not maintain the number of models: $L_1 \geq_p^{\mathcal{T}_{multi}} L_2$ can hold without L_1 satisfying CT
- Same problem for transformations

Inheritance Theorem

- We define (in the paper) a notion of **suitability to a semantical correspondence** for queries and transformations
 - CT is suitable to \mathcal{T}_{dir} , but not to \mathcal{T}_{multi}
 - CO and CD are suitable to both
 - SFO is not suitable to any of the two

Inheritance Theorem

- We define (in the paper) a notion of **suitability to a semantical correspondence** for queries and transformations
 - CT is suitable to \mathcal{T}_{dir} , but not to \mathcal{T}_{multi}
 - CO and CD are suitable to both
 - SFO is not suitable to any of the two

Theorem

If $L_1 \geq_p^{\mathcal{T}} L_2$, then all queries **suitable to \mathcal{T}** and satisfied by L_2 are satisfied by L_1 .

If $L_1 \sim_p^{\mathcal{T}} L_2$, then all transformations **suitable to \mathcal{T}** and satisfied by L_2 are satisfied by L_1 .

- Most queries and transformations in the map are suitable to \mathcal{T}_{dir} and/or \mathcal{T}_{multi}
- One can **extend the results** of some language over Boolean variables to some language over multivalued variables

Example of Application

- Family of “bounded MDDs”
 - k -MDD: restriction of MDD to domains of cardinality k ;
 - k -FMDD: read-once fragment of k -MDD;
 - k -OMDD and k -OMDD_<: ordered fragments of k -MDD
- \mathcal{T}_k : direct encoding on domains of cardinality k
 - \mathcal{T}_k is a bijection
 - all queries and transformations are suitable to \mathcal{T}_k

Example of Application

- Families of BDD and k -MDD are equivalent modulo \mathcal{T}_k
 $(k\text{-MDD} \sim_p^{\mathcal{T}_k} \text{BDD}, \quad k\text{-FMDD} \sim_p^{\mathcal{T}_k} \text{FBDD},$
 $k\text{-OMDD} \sim_p^{\mathcal{T}_k} \text{OBDD}, \quad k\text{-OMDD}_{<} \sim_p^{\mathcal{T}_k} \text{OBDD}_{<})$
- Compilation map of BDD :

$$\text{BDD} <_s \text{FBDD} <_s \text{OBDD} <_s \text{OBDD}_{<}$$

L	CO	VA	CE	IM	EQ	SE	CT	ME	CD	FO	SFO	$\wedge C$	$\wedge BC$	$\vee C$	$\vee BC$	$\neg C$
BDD	○	○	○	○	○	○	○	○	✓	○	○	✓	○	✓	○	✓
FBDD	✓	✓	✓	✓	○	○	✓	✓	✓	●	○	●	○	●	○	✓
OBDD	✓	✓	✓	✓	✓	○	✓	✓	✓	●	○	●	○	●	○	✓
OBDD _{<}	✓	✓	✓	✓	✓	○	✓	✓	✓	●	○	●	○	●	○	✓

Example of Application

- Families of BDD and k -MDD are equivalent modulo \mathcal{T}_k
 $(k\text{-MDD} \sim_p^{\mathcal{T}_k} \text{BDD}, \quad k\text{-FMDD} \sim_p^{\mathcal{T}_k} \text{FBDD},$
 $k\text{-OMDD} \sim_p^{\mathcal{T}_k} \text{OBDD}, \quad k\text{-OMDD}_{<} \sim_p^{\mathcal{T}_k} \text{OBDD}_{<})$
- Compilation map of k -MDD :

$$k\text{-MDD} <_s k\text{-FMDD} <_s k\text{-OMDD} <_s k\text{-OMDD}_{<}$$

L	CO	VA	CE	IM	EQ	SE	CT	ME	CD	FO	SFO	$\wedge C$	$\wedge BC$	$\vee C$	$\vee BC$	$\neg C$
k -MDD	○	○	○	○	○	○	○	○	✓	○	○	✓	○	✓	○	✓
k -FMDD	✓	✓	✓	✓	?	○	✓	✓	✓	●	○	●	○	●	○	✓
k -OMDD	✓	✓	✓	✓	✓	○	✓	✓	✓	●	✓	●	○	●	○	✓
k -OMDD _{<}	✓	✓	✓	✓	✓	○	✓	✓	✓	●	✓	●	✓	●	✓	✓

Conclusion

- General framework for the comparison of representation languages
 - Adaptation of concepts of the knowledge compilation map
- makes it possible to formally compare heterogeneous languages
- Mechanism to extend results from one language hierarchy to another
 - First step towards a general compilation map, presenting the various hierarchies of heterogeneous languages in a unified manner (quad-trees, R^* -trees, qualitative formalisms...)