Towards a Knowledge Compilation Map for Heterogeneous Representation Languages

Alexandre Niveau\textsuperscript{1}

joint work with

Hélène Fargier\textsuperscript{2}  Pierre Marquis\textsuperscript{3}

published in IJCAI'13

1. GREYC-CNRS, Caen, France — alexandre.niveau@unicaen.fr
2. IRIT-CNRS, Toulouse, France — fargier@irit.fr
3. CRIL-CNRS, Lens, France — marquis@cril.fr

June 4th, 2015
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

<table>
<thead>
<tr>
<th>L</th>
<th>CO (consistency)</th>
<th>VA (validity)</th>
<th>CE (clause entailment)</th>
<th>IM (implicant check)</th>
<th>EQ (equivalence)</th>
<th>SE (entailment)</th>
<th>CT (model count)</th>
<th>ME (model enum.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>BDD</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>DNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>CNF</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>CD (conditioning)</th>
<th>FO (forgetting)</th>
<th>SFO (single forg.)</th>
<th>∧C (conjunction)</th>
<th>∧BC (bounded conj.)</th>
<th>∨C (disjunction)</th>
<th>∨BC (bounded disj.)</th>
<th>¬C (negation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>BDD</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>DNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>CNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
</tbody>
</table>

✓ polynomial
○ not polynomial unless P = NP
● not polynomial
Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary queries and transformations

<table>
<thead>
<tr>
<th>L</th>
<th>CO (consistency)</th>
<th>VA (validity)</th>
<th>CE (clause entail.)</th>
<th>IM (implicant check)</th>
<th>EQ (equivalence)</th>
<th>SE (entailment)</th>
<th>CT (model count)</th>
<th>ME (model enum.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>BDD</td>
<td>○ ○</td>
<td>✓ ○</td>
<td>○ ○</td>
<td>? ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>DNF</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
<tr>
<td>CNF</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
<td>○ ✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>CD (conditioning)</th>
<th>FO (forgetting)</th>
<th>SFO (single forg.)</th>
<th>∧C (conjunction)</th>
<th>∧BC (bounded conj.)</th>
<th>∨C (disjunction)</th>
<th>∨BC (bounded disj.)</th>
<th>¬C (negation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>✓</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>BDD</td>
<td>✓</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○ ○</td>
<td>○</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓</td>
<td>● ○</td>
<td>● ○</td>
<td>● ○</td>
<td>● ○</td>
<td>● ○</td>
<td>● ○</td>
<td>● ○</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓</td>
<td>● ●</td>
<td>● ●</td>
<td>● ●</td>
<td>● ●</td>
<td>● ●</td>
<td>● ●</td>
<td>● ●</td>
</tr>
<tr>
<td>DNF</td>
<td>✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>CNF</td>
<td>✓</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
<td>✓ ○</td>
</tr>
</tbody>
</table>

✓  polynomial
○  not polynomial unless \( P = NP \)
●  not polynomial
Knowledge Compilation Map: Operations

- All online manipulations boil down to elementary queries and transformations

<table>
<thead>
<tr>
<th>L</th>
<th>CO (consistency)</th>
<th>VA (validity)</th>
<th>CE (clause entailment)</th>
<th>IM (implicant check)</th>
<th>EQ (equivalence)</th>
<th>SE (entailment)</th>
<th>CT (model count)</th>
<th>ME (model enum.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
</tr>
<tr>
<td>BDD</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>? ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
<td>◦ ◦</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
</tr>
<tr>
<td>DNF</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
<td>✓ ◦</td>
</tr>
<tr>
<td>CNF</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
<td>◦ ✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>CD (conditioning)</th>
<th>FO (forgetting)</th>
<th>SFO (single forg.)</th>
<th>∧C (conjunction)</th>
<th>∧BC (bounded conj.)</th>
<th>∨C (disjunction)</th>
<th>∨BC (bounded disj.)</th>
<th>¬C (negation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNF</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>◦</td>
</tr>
<tr>
<td>DNNF</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>✓</td>
<td>○</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>BDD</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>◦</td>
<td>○</td>
<td>○</td>
<td>◦</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>◦</td>
<td>○</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>◦</td>
<td>○</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>DNF</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>◦</td>
<td>○</td>
<td>◦</td>
<td>◦</td>
</tr>
<tr>
<td>CNF</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>✓</td>
<td>◦</td>
<td>○</td>
<td>◦</td>
<td>◦</td>
</tr>
</tbody>
</table>

✓ polynomial
○ not polynomial unless $P = NP$
bullet 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 \( L_2 \)"
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 \( L_2 \)”
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 \( L_2 \)”

- Other relations: expressiveness (\( \leq_e \)), polynomial translatability (\( \leq_p \))
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, and for languages representing functions with non-Boolean values (VDDs)

Languages close in essence: generalizations of the BDD family in several directions
   → some are “equivalent”
   → similarities in maps
Motivation

• However, these languages are *heterogeneous*, i.e., they represent different kinds of objects
  • their maps are distinct
  • their “equivalence” is not formally stated within the framework

• We would like to **unify** the maps, in order to
  • allow the comparison of heterogeneous languages
  • factorize the common parts of the maps
  • inherit results between “close” heterogeneous languages
  • enable the diversification of the KC map setting

→ We propose a generalized framework for comparing representation languages
Plan

1. Introduction
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 $P \cup \{\lor, \land, \neg, (, )\}$
  • It is in CNF if it verifies some specific properties
  • The CNF language is the set of all CNFs

• The notion of “language” concerns syntax only
  → 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 $\mathcal{U}$: contains all objects that we could intend to represent (Boolean functions, real functions, etc.)
- Generic alphabet $\Sigma$: no \emph{a priori} restriction on formulæ $\varphi \in \Sigma^*$

**Definition**

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

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

- Language of propositional logic: \( \text{PROP} = \langle \Phi_{\text{PROP}}, \mathcal{I}_{\text{PROP}} \rangle \)
  - \( \Phi_{\text{PROP}} \): set of well-formed propositional formulae
  - \( \mathcal{I}_{\text{PROP}} \): usual interpretation function
- \( \text{CNF} = \langle \Phi_{\text{CNF}}, \mathcal{I}_{\text{PROP}} \rangle \), with \( \Phi_{\text{CNF}} \) 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
- \( \text{OMDD} = \langle \Phi_{\text{OMDD}}, \mathcal{I}_{\text{MDD}} \rangle \)
  - \( \Phi_{\text{OMDD}} \): set of ordered MDDs
  - \( \mathcal{I}_{\text{MDD}} \): interpretation function of multivalued decision diagrams
Interpretation Space

- Semantics of $L$: way of interpreting some formulæ of $\Sigma^*$
- Associates with each formula $\varphi \in \Phi_L$ its interpretation $[\varphi]_L$
**Interpretation Space**

- Semantics of \( L \): way of **interpreting** some formulæ of \( \Sigma^* \)
- Associates with each formula \( \varphi \in \Phi_L \) its interpretation \( [\varphi]_L \)
- ... but it also interprets other formulæ
  (semantics of CNF: \( \mathcal{I}_{PROP} \), interprets also DNFs, for example)

\[ \rightarrow \text{interpretation space } \Omega_L \text{: set of all objects represented by the semantics of } L \]

- Example : \( \Omega_{PROP} = \Omega_{CNF} = \Omega_{HORN-C} = \text{set of Boolean functions over Boolean variables} \)
**Interpretation Space**

- Semantics of L: way of **interpreting** some formulæ of $\Sigma^*$
- Associates with each formula $\varphi \in \Phi_L$ its interpretation $[\varphi]_L$
- ... but it also interprets other formulæ
  (semantics of CNF: $\mathcal{I}_\text{PROP}$, interprets also DNFs, for example)

→ **interpretation space** $\Omega_L$: set of all objects represented by the semantics of L

- Example: $\Omega_{\text{PROP}} = \Omega_{\text{CNF}} = \Omega_{\text{HORN-C}} = \text{set of Boolean functions over Boolean variables}$

- **Completeness** of L: relative to its interpretation space
  (CNF is complete, HORN-C is incomplete)
Plan

1. Introduction
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

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: $\text{MDD} \nleq_p \text{BDD}$, 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 semantic correspondence between interpretation spaces: $\mathcal{T} \subseteq \Omega_{L_1} \times \Omega_{L_2}$

→ indicates objects considered as “equivalent”

- Example: given $f: \mathbb{N}^n \rightarrow \mathbb{B}$ and $g: \mathbb{B}^m \rightarrow \mathbb{B}$,

  \[ f \mathcal{T}_{dir} g \iff g \text{ is a direct encoding of } f \]

- Similarly for 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 $\varphi_1$ of $L_1$ into a formula $\varphi_2$ of $L_2$ such that $\left[ \varphi_1 \right]_{L_1} T \left[ \varphi_2 \right]_{L_2}$, then $L_1$ is said to be polynomially translatable into $L_2$ modulo $T$

• We denote it as $L_1 \geq_p^T L_2$

→ Generalization of the classical polynomial translatability:
$L_1 \geq_p L_2$ corresponds to $L_1 \geq_{p}^{id} L_2$

• We also extend the succinctness and expressiveness relations to the use of a correspondence: $L_1 \geq_s^T L_2$ and $L_1 \geq_e^T L_2$
Examples

- Thanks to the extended relations, one can compare heterogeneous languages:
  - \( \text{MDD} \geq^p \text{BDD} \) and \( \text{MDD} \geq^p \text{BDD} \)
  - \( \text{MDD} \not<^s \text{CNF} \)
- One can also compare homogeneous languages of incomparable expressiveness (e.g., HORN-C and AFF), via a well-chosen semantic correspondence.
- One can extend succinctness results from one family of languages to another via some translation:
  
  \[
  \begin{align*}
  \text{BDD} & \prec_s \text{OBDD} \\
  \Rightarrow & \\
  \text{MDD} & \prec_s \text{OMDD}
  \end{align*}
  \]
Plan

1. Introduction

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
  • MODS $\geq_p$ OBDD
    $\Rightarrow$ MODS satisfies all queries that OBDD satisfies
  • NNF $\sim_p$ PROP
    $\Rightarrow$ 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 $L_1 \geq_T L_2$.
  What can we say about queries satisfied by $L_1$?
  → **Nothing** in the general case: it depends on the $T$ used
    • Let $L_2$ be a language satisfying $CT$
    • $T_{dir}$ maintains the number of models, so if $L_1 \geq_{p\ T_{dir}} L_2$ holds, then $L_1$ also satisfies $CT$
    • $T_{multi}$ does not maintain the number of models: $L_1 \geq_{p\ T_{multi}} L_2$ can hold without $L_1$ satisfying $CT$

• Same problem for transformations
Inheritance Theorem

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

**Theorem**

If $L_1 \geq_T L_2$, then all queries *suitable to $T$ and satisfied by $L_2$ are satisfied by $L_1$.*

If $L_1 \sim_T L_2$, then all transformations *suitable to $T$ and satisfied by $L_2$ are satisfied by $L_1$.*

- Most queries and transformations in the map are suitable to $T_{dir}$ and/or $T_{multi}$

→ 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 $\mathcal{T}_k$
  
  \[(k\text{-MDD} \sim_{\mathcal{T}_k} BDD, \quad k\text{-FMDD} \sim_{\mathcal{T}_k} FBDD, \quad k\text{-OMDD} \sim_{\mathcal{T}_k} OBDD, \quad k\text{-OMDD} < \sim_{\mathcal{T}_k} OBDD <)\]

- Compilation map of BDD:

  $$BDD <_s FBDD <_s OBDD <_s OBDD <$$

<table>
<thead>
<tr>
<th></th>
<th>CO</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
<th>CD</th>
<th>FO</th>
<th>SFO</th>
<th>$\land$C</th>
<th>$\land$BC</th>
<th>$\lor$C</th>
<th>$\lor$BC</th>
<th>$\lor$C</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDD</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>√</td>
<td>o</td>
<td>o</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FBDD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>o</td>
<td>√</td>
<td>√</td>
<td>✓</td>
<td>o</td>
<td>o</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OBDD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>o</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OBDD&lt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>o</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Example of Application

- Families of BDD and $k$-MDD are equivalent modulo $T_k$
  ($k$-MDD $\sim_{T_k} BDD$, $k$-FMDD $\sim_{T_k} FBDD$
  $k$-OMDD $\sim_{T_k} OBDD$, $k$-OMDD $\prec\sim_{T_k} OBDD\prec$)

- Compilation map of $k$-MDD:

\[
k-MDD \prec_s k-FMDD \prec_s k-OMDD \prec_s k-OMDD \prec
\]
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 and \(R^*\)-trees, qualitative formalisms, languages representing preferences...)