A Few Words about Knowledge Compilation

Pierre Marquis

Université d'Artois
CRIL UMR CNRS 8188
France

Third Workshop on Kernelization (WorKer 2011), Vienna, September 2nd, 2011

What is ”Knowledge Compilation”?

- A family of approaches for addressing the **intractability of a number of AI problems**
- Is concerned with **pre-processing** pieces of available information for **improving some tasks from a computational point of view**
- Amounts to a **translation** issue:
  - **Off-line phase:** Turn some pieces of information Σ into a **compiled form** $comp(\Sigma)$
  - **On-line phase:** Exploit the compiled form $comp(\Sigma)$ (and the remaining pieces of information α) to achieve the task(s) under consideration

Knowledge Compilation: A Recent Research Topic

- Identified as a **research topic in AI** in the ”recent” past (say, 20 years ago)
- The name “knowledge compilation” dates back to the late 80’s/beginning of the 90’s (the purpose was to improve propositional reasoning, especially clausal entailment)
- **Many developments** from there
  - From the theoretical side (concepts, algorithms, etc.)
  - From the practical side (benchmarks, pieces of software, applications, etc.)

Knowledge Compilation: An Old Idea

- Pre-processing pieces of information for improving computations is an **old idea!**
- Many applications in Computer Science (even before the modern computer era)
- Tables of logarithms (John Napier) date back from the 17^{th} century...
- Indexes for DBs, etc.
- ”Improving computations” means (typically) “saving computation time”

What is ”Knowledge”?

- Taken in a rather broad sense
  (and not necessarily as “true belief”)
- Pieces of information and **ways to exploit** them
- Same meaning as “knowledge” in “knowledge representation”
- Pieces of information are typically encoded as formulae Σ, α, ... in a **logic-based framework**

$$\langle L, \vdash \rangle$$

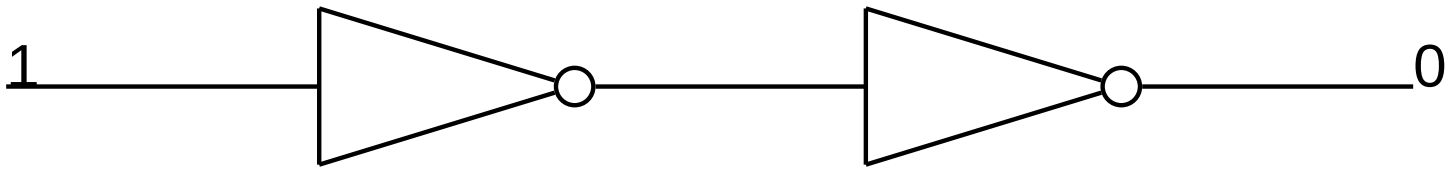
What is ”Exploiting Knowledge”?

- What are the **tasks** to be computationally improved via knowledge compilation?
- A **domain-dependent issue** in general
- Typically **combinations of basic queries and transformations**

Basic Queries and Transformations

- **Queries**
  - **Consistency:** Does there exist α such that $\Sigma \nvdash \alpha$ holds?
  - **Sentential Entailment:** Does $\Sigma \vdash \alpha$ hold?
  - ...
- **Transformations**
  - **Conditioning:** Make some elementary propositions true (or false) in Σ
  - **Closures under connectives:** Compute a representation in L of $\alpha \oplus \beta$ from $\alpha \in L$ and $\beta \in L$
  - **Forgetting:** When defined, compute a representation of the most general consequence w.r.t. $\vdash$ of $\Sigma \in L$ not containing some given elementary propositions
  - ...

Example: Consistency-Based Diagnosis



Example: Consistency-Based Diagnosis

- $S = (SD, OBS)$ gathers a **system description** SD and some **observations** OBS
- SD describes the behaviour of the system components and how they are connected

$$\neg ab - inv_1 \Rightarrow (out - inv_1 \Leftrightarrow \neg in - inv_1)$$

$$\neg ab - inv_2 \Rightarrow (out - inv_2 \Leftrightarrow \neg in - inv_2)$$

$$out - inv_1 \Leftrightarrow in - inv_2$$

- OBS describes the inputs and outputs of the system

$$in - inv_1 \wedge \neg out - inv_2$$

- Δ is a **consistency-based (c-b) diagnosis** for S iff it is a conjunction of ab-literals such that $\Delta \wedge SD \wedge OBS$ is consistent

$$ab - inv_1, ab - inv_2, ab - inv_1 \wedge ab - inv_2$$

Example: Consistency-Based Diagnosis

- Generating the **consistency-based diagnoses** of a system $S = (SD, OBS)$ for many observations OBS

$$mod(\exists(PS \setminus AB).(SD \mid OBS))$$

$$\exists(PS \setminus AB).(SD \mid OBS) \equiv$$

$$(\exists(PS \setminus AB).SD) \mid OBS \equiv ab - inv_1 \vee ab - inv_2$$

- The task can be viewed as a combination of **conditioning**, **forgetting** and **model enumeration**
- Forgetting **FO** and model enumeration **ME** are NP-hard in the general case

When is Knowledge Compilation Useful?

Two conditions are necessary:

- At least one of the tasks under consideration is **computationally hard** (NP-hard)
- Some pieces of information are **more subject to change than others**
- The archetypal **inference problem**: a set of pairs $\{\langle \Sigma, \alpha \rangle\}$
  - A "knowledge" base Σ (the fixed part)
  - Queries α about it: $\alpha_1, \ldots, \alpha_n$ (the varying part)
- Pre-processing Σ before answering queries makes sense if Σ does not often change and n is sufficiently large
- Some queries/transformations of interest become **"less intractable"**, provided that the computational effort spent during the off-line phase is **"reasonable"**

Example: Consistency-Based Diagnosis

- Fixed part $\Sigma = SD$, varying part $\alpha = OBS$ (and Δ)
- Forgetting **FO** and model enumeration **ME** are NP-hard in the general case
- If conditioning **CO**, forgetting **FO** and model enumeration **ME** are easy (i.e. in P) from $comp(\Sigma)$, then if $comp(\Sigma)$ is "small enough", the computational effort spent in generating it can be balanced over many OBS (and Δ)

In the Following: Two Central Issues

- How to **evaluate** at the problem level whether KC can prove useful?
  - Is consistency-based diagnosis compilable (to P)?
- How to **choose** a target language for the KC purpose?
  - Into which language should $comp(SD)$ be represented?

The Compilability Issue

Cadoli *et al.* [AIJ96, AI Comm.98, AIJ99, JAIR00, Inf.Comp.02]
Liberatore [Ph.D.98, JACM01]

- Evaluating KC **at the problem level**
- **Intuition:** A (decision) problem is **compilable to a complexity class** C if it is in C once the fixed part Σ of any instance has been pre-processed, i.e., turned off-line into a data structure **of size polynomial** in $|\Sigma|$
- Several **compilability classes** organized into hierarchies (which echo PH)
- Enable to classify problems as **compilable to** C, or as **non-compilable to** C (usually under standard assumptions of complexity theory)

Decision Problems = Languages L of Pairs

- $\langle \Sigma, \alpha \rangle \in L$
- Σ: The **fixed part**
- α: The **varying part**
- **Examples:**

CLAUSE ENTAILMENT = $\{\langle \Sigma, \alpha \rangle \mid \Sigma$ a propositional formula and α a clause s.t. $\Sigma \models \alpha\}$

C-B DIAGNOSIS = $\{\langle \Sigma = SD, \alpha = (OBS, \Delta) \rangle \mid \Delta$ is a c-b diagnosis for $(SD, OBS)\}$

compC

- C = a complexity class closed under polynomial reductions and admitting complete problems for such reductions
- A language of pairs L **belongs to** compC if and only if there exists a **polysize function** *comp* and a language of pairs $L' \in$ C such that for every pair $\langle \Sigma, \alpha \rangle$, $\langle \Sigma, \alpha \rangle \in L$ iff $\langle comp(\Sigma), \alpha \rangle \in L'$
- For every admissible complexity class C, we have the inclusion C $\subseteq$ compC

compC

- Membership to compC: Follow the definition!
- Non-membership to compC: A more complex issue in general
- Classes C/poly are useful

Advice-Taking Turing Machines

- An **advice-taking Turing machine** is a Turing machine that has associated with it a special "advice oracle" A, which can be any function (not necessarily a recursive one)
- On input s, a special "advice tape" is automatically loaded with $A(|s|)$ and from then the computation proceeds as normal, based on the two inputs, s and $A(|s|)$

C/poly

- An advice-taking Turing machine uses **polynomial advice** if its advice oracle A is **polysize**
- If C is a class of languages defined in terms of resource-bounded Turing machines, then C/poly is the class of languages defined by Turing machines with the same resource bounds but augmented by polynomial advice
- C/poly contains all languages L for which there exists a polysize function A from $\boldsymbol{N}$ to the set of strings s.t. the language $\{\langle A(|s|), s\rangle \mid s \in L\}$ belongs to C

P/poly vs. PH

Karp and Lipton [ACM STOC'98], Yap [TCS83]

- **If** NP $\subseteq$ P/poly **then** $\Pi_2^p = \Sigma_2^p$ **(hence** PH **collapses at the second level)**
- If NP $\subseteq$ coNP/poly then $\Pi_3^p = \Sigma_3^p$ (hence PH collapses at the third level)

CLAUSE ENTAILMENT $\notin$ compP

... unless PH collapses at the second level
(Kautz and Selman [AAAI'92])

- Let n be any non-negative integer
- Let Σ_n^{max} be the CNF formula

$$\bigwedge_{\gamma_i \in 3-C_n} \neg holds_i \vee \gamma_i$$

- $3 - C_n$ is the set of all 3-literal clauses that can be generated from $\{x_1, \ldots, x_n\}$ and the $holds_i$ are new variables, not among $\{x_1, \ldots, x_n\}$
- $|\Sigma_n^{max}| \in \mathcal{O}(n^3)$

CLAUSE ENTAILMENT $\notin$ compP

- Each 3-CNF formula α_n built up from the set of variables $\{x_1, \ldots, x_n\}$ is in bijection with the subset S_{α_n} of the variables $holds_i$ s.t. γ_i is a clause of α_n if and only if $holds_i \in S_{\alpha_n}$
- α_n is unsatisfiable iff

$$\Sigma_n^{max} \models \gamma_{\alpha_n} = \bigvee_{holds_i \in S_{\alpha_n}} \neg holds_i$$

CLAUSE ENTAILMENT $\notin$ compP

- Assume that we have a polysize compilation function *comp* such that for any CNF Σ and any clause γ, determining whether $comp(\Sigma) \models \gamma$ belongs to P
- Then 3-SAT would be in P/poly:
  - Let α be a 3-CNF formula
  - If $|Var(\alpha)| = n$, then the machine loads

$$A(n) = comp(\Sigma_n^{max})$$

  - Finally it determines in polynomial time whether $comp(\Sigma_n^{max}) \models \gamma_\alpha$
- Since 3-SAT is complete for NP, this would imply NP $\subseteq$ P/poly
- Similarly, C-B DIAGNOSIS $\notin$ compP unless PH collapses at the second level

The Impact of the Language of Varying Parts

- If for each fixed part Σ, there are only **polynomially many** possible varying parts α, then the corresponding language of pairs L belongs to compP
- During the off-line phase, consider successively every α and store it in a lookup-table $comp(\Sigma)$ whenever $\langle \Sigma, \alpha \rangle$ belongs to L
- For every Σ, $|comp(\Sigma)|$ is polynomially bounded in the size of Σ and determining on-line whether $\langle \Sigma, \alpha \rangle \in L$ amounts to a lookup operation

Example: LITERAL ENTAILMENT $\in$ compP

- $\{\langle \Sigma, \alpha \rangle \mid \alpha \in L_{Var(\Sigma)}, \Sigma \models \alpha\} \in$ compP
- Only the literals occurring in Σ have to be considered during the off-line phase
- LITERAL ENTAILMENT is coNP-complete: intractable when viewed "all-at-once", tractable as a language of pairs

A Sufficient, yet Non-Necessary Restriction

- The fact that only polynomially many varying parts α are possible is **not a necessary condition** for the membership to compP
- TERM ENTAILMENT $\in$ compP
- A **separability property** at the query level (the dual of the disjunctive one)

$$\Sigma \models \alpha_1 \wedge \ldots \wedge \alpha_n \text{ iff } \forall i \in 1 \ldots n, \Sigma \models \alpha_i$$

- Can be extended to the K-CNF ENTAILMENT problem

compC-Reductions are not General Enough!

- A notion of comp**-reduction** suited to the compilability classes compC has been pointed out
- The existence of **complete problems** for such classes has been proven
- Many non-compilability results from the literature **cannot** be rephrased as compC-completeness results
- E.g. it is unlikely that CLAUSE ENTAILMENT is compcoNP-complete (this would imply P = NP)
- There is a need for **more general compilability classes:** nu-compC

nu-compC

- C = a complexity class closed under polynomial reductions and admitting complete problems for such reductions
- A language of pairs L belongs to nu-compC if and only if there exists a **binary polysize function** *comp* and a language of pairs $L' \in$ C such that for all $\langle \Sigma, \alpha \rangle \in L$, we have:

$$\langle \Sigma, \alpha \rangle \in L \text{ iff } \langle comp(\Sigma, |\alpha|), \alpha \rangle \in L'$$

- "nu" stands for "non-uniform", which indicates that the compiled form of Σ may also **depend on the size of the varying part** α

Links between Compilability Classes

For each admissible C:

- C $\subseteq$ compC $\subseteq$ nu-compC
- C $\subseteq$ C/poly $\subseteq$ nu-compC
- Under reasonable assumptions on C, all those inclusions are **strict ones**

Reduction and Complete Problems

For each admissible C:

- A notion of **non-uniform** comp**-reduction** suited to the compilability classes nu-compC has also been pointed out (it includes the notion of (uniform) comp-reduction)
- The existence of **complete problems** for such classes has been proven
- CLAUSE ENTAILMENT is nu-compcoNP-complete

Compilability Hierarchies

For each admissible C:

- Inclusions of compilability classes C/poly, compC, nu-compC similar to those holding in PH exist
- It is strongly believed that **the corresponding compilability hierarchies are proper**: if one of them collapses, then PH collapses at well
- For instance, if CLAUSE ENTAILMENT is in nu-compP, then PH collapses

Proving Non-Compilability

- In order to show that a problem is not in compC, it is enough **to prove that it is** nu-compC'**-hard, where** C' **is located "higher" than** C in the polynomial hierarchy
- **Complete problems** for any nu-compC class can be derived from complete problems for C
- Hence nu-compC-complete problems appear as a very interesting tool for proving non-compilability results

Further Readings

Compilability of a number of AI problems: diagnosis, planning, abduction, belief revision, closed-world reasoning, paraconsistent inference from belief bases, etc.

- Liberatore [PhD98, KR'98, ACM TOCL00, IJIS05]
- Cadoli *et al.* [AIJ99, Inf.Comp.02]
- Liberatore and Schaerf [ACM TCL07]
- Nebel [JAIR00]
- Coste and Marquis [AMAI02]
- Darwiche and Marquis [AIJ04]
- Chen [IJCAI'05]
- ...

The KC Map

Darwiche and Marquis [IJCAI'01, JAIR02]
A **multi-criteria evaluation** of target languages for propositional KC

- **Queries:** operations returning information from a compiled form without changing it
- **Transformations:** operations modifying the compiled form
- **Succinctness:** the ability of a language to represent information using little space
- ...

Queries

Decision or functions problems / properties of fragments

- **CO** (consistency)
- **CE** (clause entailment: implicates)
- **VA** (validity)
- **EQ** (equivalence)
- **SE** (sentential entailment)
- **IM** (implicants)
- **CT** (model counting)
- **ME** (model enumeration)
- ...

Transformations

Function problems / properties of fragments

- **CD** conditioning
- $\wedge$ **C** ($\wedge$**BC**) (closure under $\wedge$)
- $\vee$**C** ($\vee$**BC**) (closure under $\vee$)
- $\neg$**C** (closure under $\neg$)
- **FO** (**SFO**) (forgetting)
- ...

Queries and Transformations are not Independent

Let **L** be a propositional language

- If a language **L** satisfies **SE**, then it also satisfies **CE** and **EQ**
- If **L** satisfies **ME**, then it satisfies **CO**
- If **L** satisfies **CO** and **CD**, then it satisfies **CE**
- If **L** satisfies **CT**, then it satisfies **CO** and **VA**
- If **L** satisfies **CO**, $\wedge$**C** and $\neg$**C**, then it satisfies **SE**
- If **L** satisfies **VA**, $\vee$**C** and $\neg$**C**, then it satisfies **SE**
- If **L** contains L_{PS} and satisfies $\wedge$**C** and $\vee$**BC**, then it does not satisfy **CO** unless P = NP
- If **L** satisfies **FO**, then it satisfies **CO**
- ...

Succinctness

Succinctness captures **the ability of a language to represent information using little space**

- $\mathbf{L}_1$ is **at least as succinct as** $\mathbf{L}_2$, denoted $\mathbf{L}_1 \leq_s \mathbf{L}_2$, iff there exists a polynomial p such that for every formula $\alpha \in \mathbf{L}_2$, there exists an equivalent formula $\beta \in \mathbf{L}_1$ where $|\beta| \leq p(|\alpha|)$
- A **pre-order** $\leq_s$ over propositional languages

Many Languages in the KC Map

- ”Flat” Languages
  - CNF
  - DNF
  - PI (prime implicates)
  - IP (prime implicants)
  - MODS
  - ...
- ”Nested” Languages
  - DNNF
  - d-DNNF
  - $\text{OBDD}_{<}$
  - ...

DNNF

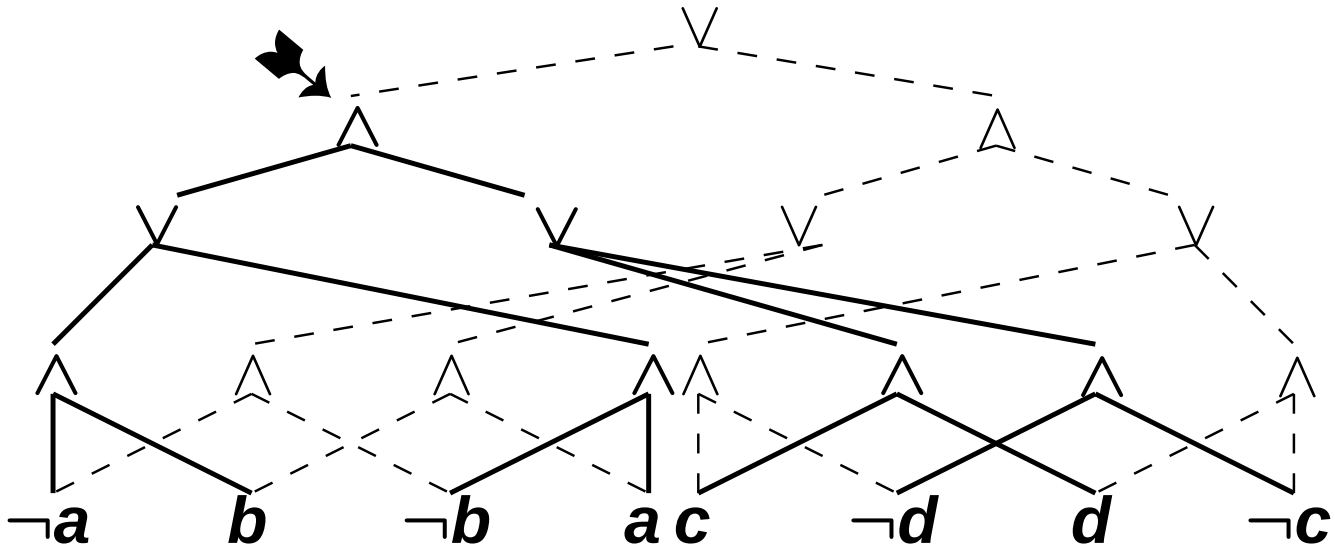


Figure: A DNNF formula.

d-DNNF

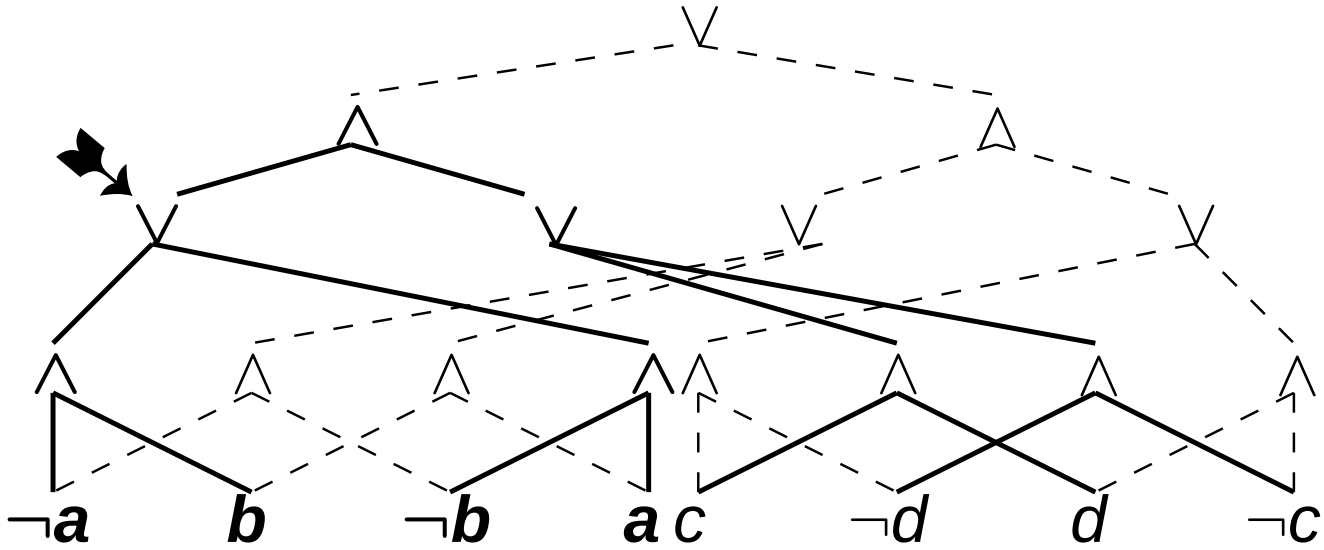


Figure: A d-DNNF formula.

$\mathrm{OBDD}_<$

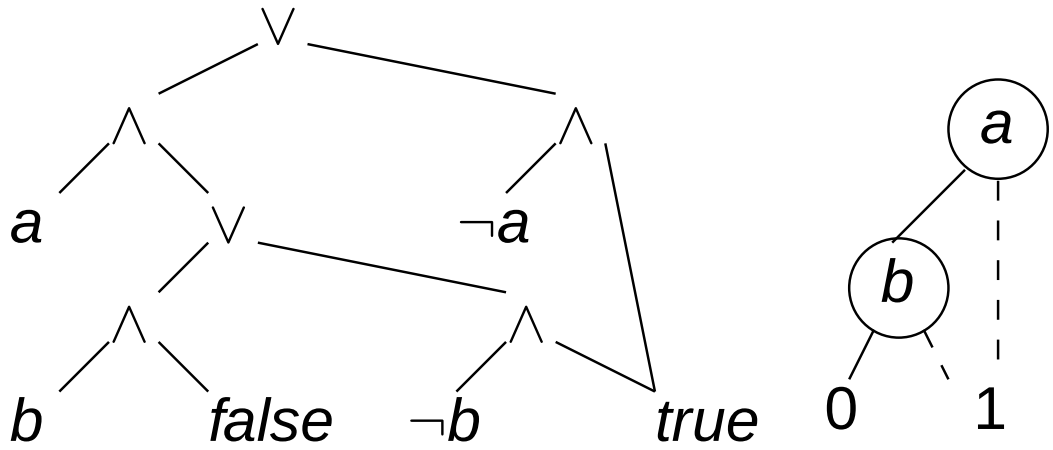


Figure: On the left part, a formula in the $\mathrm{OBDD}_<$ language. On the right part, a more standard notation for it.

Fragments, Queries and Transformations

“Positive” results

- DNNF satisfies **CO**, **CE**, **ME**, **CD**, **FO**, $\vee$**C**
- d-DNNF satisfies **CO**, **VA**, **CE**, **IM**, **CT**, **ME**, **CD**
- $\mathrm{OBDD}_{<}$ satisfies **CO**, **VA**, **CE**, **IM**, **EQ**, **CT**, **ME**, **CD**, **SFO**, $\wedge$**BC**, $\vee$**BC**, $\neg$**C**
- DNF satisfies **CO**, **CE**, **ME**, **CD**, **FO**, $\wedge$**BC**, $\vee$**C**
- PI satisfies **CO**, **VA**, **CE**, **IM**, **EQ**, **SE**, **ME**, **CD**, **FO**, $\vee$**BC**
- IP satisfies **CO**, **VA**, **CE**, **IM**, **EQ**, **SE**, **ME**, **CD**, $\wedge$**BC**
- MODS satisfies **CO**, **VA**, **CE**, **IM**, **EQ**, **SE**, **CT**, **ME**, **CD**, **FO**, $\wedge$**BC**

DNNF satisfies **FO**

An **inductive** characterization:

- $\exists X.\mathit{false} \equiv \mathit{false}$
- $\exists X.\mathit{true} \equiv \mathit{true}$
- $\exists X.l \equiv \mathit{true}$ if $\mathit{var}(l) \in X$, $\equiv l$ otherwise
- $\exists X.(\alpha_1 \vee \ldots \vee \alpha_n) \equiv (\exists X.\alpha_1) \vee \ldots \vee (\exists X.\alpha_n)$
- $\exists X.(\alpha_1 \wedge \ldots \wedge \alpha_n) \equiv (\exists X.\alpha_1) \wedge \ldots \wedge (\exists X.\alpha_n)$ since $\alpha_1 \wedge \ldots \wedge \alpha_n$ is decomposable

Polytime Reductions, Queries and Transformations

- If L_2 is polytime reducible to L_1 and L_1 satisfies a given query **QU** then L_2 satisfies **QU**
- If L_2 is polytime reducible to L_1 and L_2 does not satisfy a given query **QU** unless P = NP then L_1 does not satisfy **QU** unless P = NP
- There is no similar results for transformations **TR** in the general case
  - $\mathtt{OBDD}_<$ is polytime reducible to DNNF
  - DNNF satisfies **FO**
  - $\mathtt{OBDD}_<$ does not satisfy **FO**

Fragments, Queries and Transformations

“Negative” results

- DNNF **does not satisfy** any of **VA**, **IM**, **EQ**, **SE**, **CT**, $\wedge$**BC**, $\neg$**C** unless P = NP
- **VA**: the validity problem for DNF formulae is coNP-complete
- ...

The Succinctness of Propositional Fragments

- $\mathtt{DNNF} <_s \mathtt{d\text{-}DNNF} <_s \mathtt{OBDD}_< <_s \mathtt{MODS}$
- $\mathtt{CNF} \not\leq_s \mathtt{DNF}$
- $\mathtt{DNNF} \not\leq_s^* \mathtt{CNF}$
- ...

Succinctness vs. Non-Succinctness Results

Different kinds of proof

- DNNF $\leq_s$ DNF: Easy since DNNF $\supseteq$ DNF
- DNF $\not\leq_s$ DNNF: **Combinatorial arguments**

$$\bigwedge_{i=0}^{n-1} (\neg x_{2i} \vee x_{2i+1}) \in \text{DNNF}$$

- DNNF $\not\leq_s^*$ CNF: **Exploit non-compilability results!**
  - DNNF satisfies **CE**
  - CLAUSE ENTAILMENT from CNF formulae Σ is not in compP unless PH collapses

Taking Advantage of the KC Map

- **Identify** the queries and transformations required by the application
- **Select** the fragments satisfying them
- **Choose** one of the most succinct fragments among the selected ones

Example: Consistency-Based Diagnosis

Generating the **consistency-based diagnoses** of a system

- **ME**, **FO**, **CD** are required
- `DNNF`, `DNF`, `PI`, `MODS` offer them
- `DNNF` and `PI` are the most succinct ones among them
- Explain the success of `DNNF` and `PI` for model-based diagnosis?

Further Readings

- Waechter and Haenni [KR'06] (PDAG)
- Fargier and Marquis [AAAI'06] (DDG)
- Subbarayan *et al.* [AAAI'07] (`tree-of-BDDs`)
- Fargier and Marquis [IJCAI'09] (`tree-of-C`)
- Pipatsrisawat and Darwiche [AAAI'08] ($\mathtt{DNNF}_T$)
- Mateescu*et al.* [JAIR08] (AOMDD)
- Darwiche [IJCAI'11] (SDD)
- Fargier and Marquis [ECAI'08, AAAI'08], Marquis [IJCAI'11] (closure principles)
- ...

Conclusion

A Few Words about ...

- Knowledge compilation
  - The compilability issue
  - The KC map

KC in AI: Other Topics

(see the ECAI'08 tutorial notes on my Web page for references)

- KC for reasoning under inconsistency
- KC for closed-world reasoning and default reasoning
- KC languages based on other formal settings, like propositional epistemic logic, CSPs, Bayesian networks, valued CSPs, description logics, etc.
- Applications of KC to diagnosis
- Applications of KC to planning
- ...

Some Issues for Further Work

- The **decomposition of other problems** (from AI or considered by other communities) into queries and transformations.
- The **development of KC maps** for more expressive settings than propositional logic
- The **design of additional compilers** and their evaluation on benchmarks.
- The **successful exploitation of KC for other applications**
- ...

A Few Words about Knowledge Compilation

Pierre Marquis

Université d'Artois
CRIL UMR CNRS 8188
France

Third Workshop on Kernelization (WorKer 2011), Vienna, September 2nd, 2011