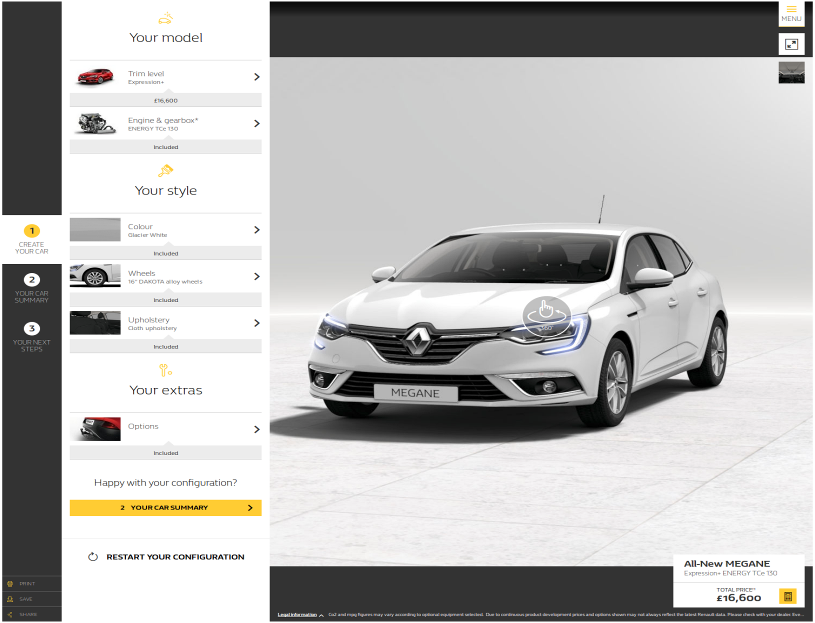
Knowledge Compilation in Artificial Intelligence and Databases

Stefan Mengel

CNRS, CRIL UMR 8188

16/03/2017


MENU
Your model
Trim level
Expression+
£16,600
Engine & gearbox*
ENERGY TCe 130
Included
Your style
Colour
Glacier White
Included
Wheels
16" DAKOTA alloy wheels
Included
Upholstery
Cloth upholstery
Included
Your extras
Options
Included
Happy with your configuration?
2 YOUR CAR SUMMARY
RESTART YOUR CONFIGURATION
1 CREATE YOUR CAR
2 YOUR CAR SUMMARY
3 YOUR NEXT STEPS
PRINT
SAVE
SHARE
360°
MEGANE
All-New MEGANE
Expression+ ENERGY TCe 130
TOTAL PRICE
£16,600
Legal information
Co2 and mpg figures may vary according to optional equipment selected. Due to continuous product development prices and options shown may not always reflect the latest Renault data. Please check with your dealer. Eve...

The Configurator as a Computational Problem

- in the applet
  - configure parts of car ($\sim$ 160 parts, some hidden, 10^{26} combinations)
  - constraints on combinations (mostly marketing, 10^{21} possible cars)
  - after partial choice by user, check for legal car (and minimal price, restrict future choices, take back choices, ...)

The Configurator as a Computational Problem

- in the applet
  - configure parts of car ($\sim$ 160 parts, some hidden, 10^{26} combinations)
  - constraints on combinations (mostly marketing, 10^{21} possible cars)
  - after partial choice by user, check for legal car (and minimal price, restrict future choices, take back choices, ...)
- slightly abstracted
  - input: CNF formula and partial assignement
  - question: is there satisfying assignment?
  - clearly NP-hard

The Configurator as a Computational Problem

- in the applet
  - configure parts of car ($\sim$ 160 parts, some hidden, 10^{26} combinations)
  - constraints on combinations (mostly marketing, 10^{21} possible cars)
  - after partial choice by user, check for legal car (and minimal price, restrict future choices, take back choices, ...)
- slightly abstracted
  - input: CNF formula and partial assignement
  - question: is there satisfying assignment?
  - clearly NP-hard
- so is there a SAT/CSP-solver in this flash-applet? NO, because...
  - not enough computational power
  - want guaranteed response time

Knowledge compilation and d-DNNF

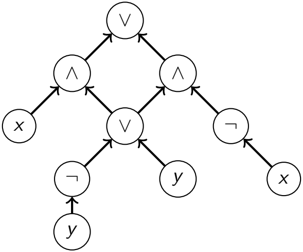
- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning

Knowledge compilation and d-DNNF

- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF

Knowledge compilation and d-DNNF

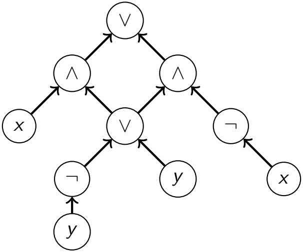
- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF



- d-DNNF (deterministic decomposable negation normal form)

Knowledge compilation and d-DNNF

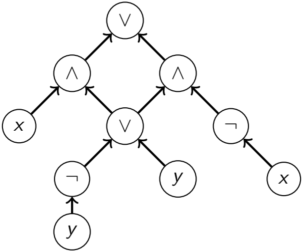
- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF



- d-DNNF (deterministic decomposable negation normal form)
  - negation normal form: ¬ only on inputs

Knowledge compilation and d-DNNF

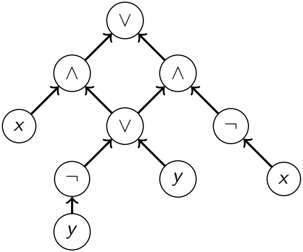
- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF



- d-DNNF (deterministic decomposable negation normal form)
  - negation normal form: $\neg$ only on inputs
  - decomposable: children of $\wedge$ have disjoint variables

Knowledge compilation and d-DNNF

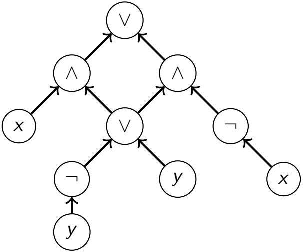
- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF



- d-DNNF (deterministic decomposable negation normal form)
  - negation normal form: $\neg$ only on inputs
  - decomposable: children of $\wedge$ have disjoint variables
  - deterministic: every assignment satisfies at most one child of $\vee$

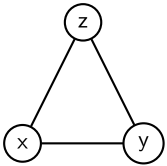
Knowledge compilation and d-DNNF

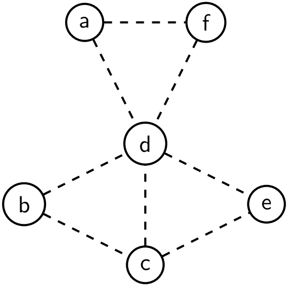
- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF



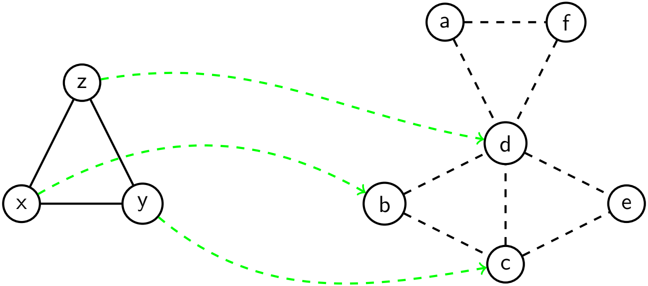
- d-DNNF (deterministic decomposable negation normal form)
  - negation normal form: $\neg$ only on inputs
  - decomposable: children of $\wedge$ have disjoint variables
  - deterministic: every assignment satisfies at most one child of $\vee$
- generalizes other data structures like OBDDs

Probabilistic Embedding

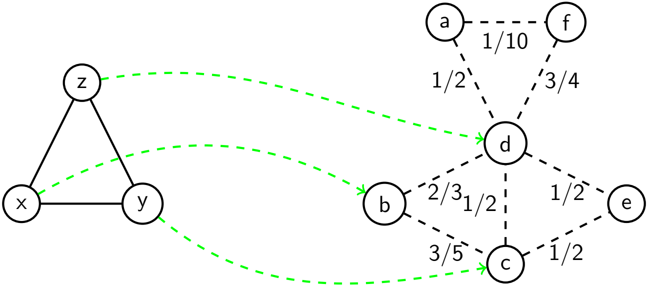




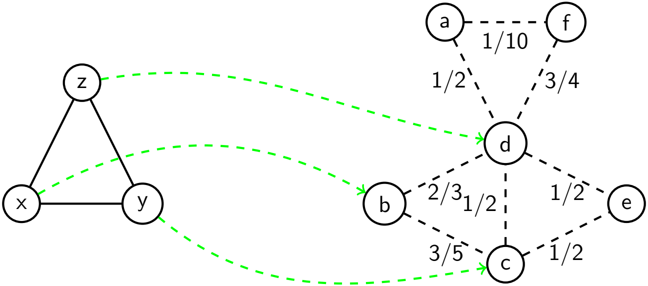
Probabilistic Embedding



Probabilistic Embedding

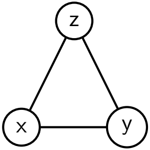


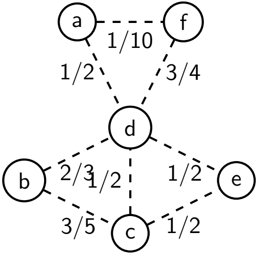
Probabilistic Embedding



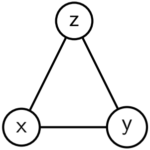
- probability of having mapping that maps edges on edges?
- models problem on probabilistic databases
- #P-hard

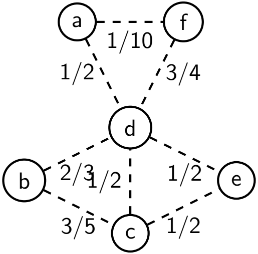
The Practical Approach: Grounding





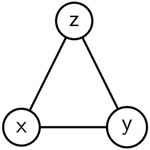
The Practical Approach: Grounding

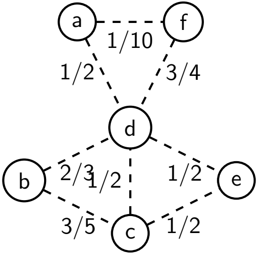




- write down all potential matches, answer “yes” if and only if one of them there

$$\{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\}$$





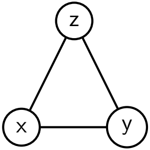
- write down all potential matches, answer "yes" if and only if one of them there

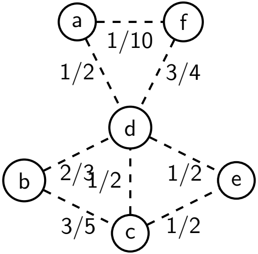
$$\{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\}$$

- translate to DNF

$$(x_{af} \wedge x_{ad} \wedge x_{df}) \vee (x_{bd} \wedge x_{bc} \wedge x_{cd}) \vee (x_{cd} \wedge x_{ce} \wedge x_{de})$$

The Practical Approach: Grounding





- write down all potential matches, answer "yes" if and only if one of them there

$$\{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\}$$

- translate to DNF

$$(x_{af} \wedge x_{ad} \wedge x_{df}) \vee (x_{bd} \wedge x_{bc} \wedge x_{cd}) \vee (x_{cd} \wedge x_{ce} \wedge x_{de})$$

- compute probability with model counter (typically CNF)

Model Counters: Exhaustive DPLL

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Model Counters: Exhaustive DPLL

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Theorem (Huang, Darwiche '05)

Traces of runs of exhaustive DPLL are d-DNNF.

Model Counters: Exhaustive DPLL

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Theorem (Huang, Darwiche '05)

Traces of runs of exhaustive DPLL are d-DNNF.

- so model counters can also compile: sharpSAT $\rightsquigarrow$ Dsharp
- bounds on d-DNNF
  $\approx$ bounds on grounding approach to probabilistic embedding

Lower Bounds for d-DNNF

Theorem (Bova, Capelli, M., Slivovsky, IJCAI 2016)

There are CNF-formulas whose smallest d-DNNF encoding has exponential size.

Communication Complexity

Communication Complexity

- Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- Alice knows assignment to $x_1, \ldots, x_n$, Bob knows assignment to $y_1, \ldots, y_n$

Definition: Communication Complexity $CC(f)$

minimal number of bits that Alice and Bob have to exchange

Communication Complexity

- Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- Alice knows assignment to $x_1, \ldots, x_n$, Bob knows assignment to $y_1, \ldots, y_n$

Definition: Communication Complexity $CC(f)$

minimal number of bits that Alice and Bob have to exchange

Examples

- $CC(PARITY) = 1$
- $CC(EQUAL(x_1, \ldots, x_n, y_1, \ldots, y_n)) = n$

Communication Complexity

- Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- Alice knows assignment to $x_1, \ldots, x_n$, Bob knows assignment to $y_1, \ldots, y_n$

Definition: Communication Complexity $CC(f)$

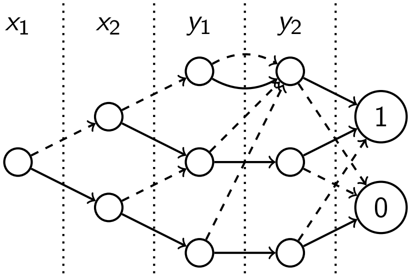
minimal number of bits that Alice and Bob have to exchange

Examples

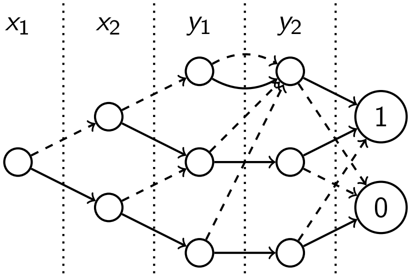
- $CC(PARITY) = 1$
- $CC(EQUAL(x_1, \ldots, x_n, y_1, \ldots, y_n)) = n$

- well studied, many variations, techniques and results

Lower Bounds for OBDD by Communication Complexity



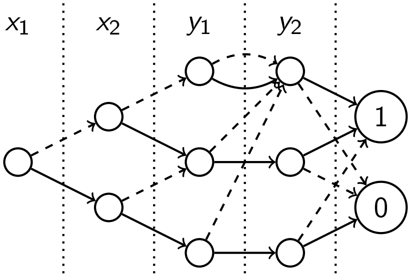
Lower Bounds for OBDD by Communication Complexity



communication protocol from small OBDD:

- assume small OBDD for $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- cut OBDD in the middle
- Alice sends node she reaches on her half of assignment

Lower Bounds for OBDD by Communication Complexity



communication protocol from small OBDD:

- assume small OBDD for $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- cut OBDD in the middle
- Alice sends node she reaches on her half of assignment
  $\Rightarrow \log(\#\text{ nodes in middle}) \geq CC(f)$

Generalization to d-DNNF

Connection (Bova, Capelli, M, Slivovsky '16)

Functions with high multipartition communication complexity have no small d-DNNF.

Generalization to d-DNNF

Connection (Bova, Capelli, M, Slivovsky '16)

Functions with high multipartition communication complexity have no small d-DNNF.

Theorem (essentially Duris et al. '04)

There are CNF with high multipartition communication complexity.

Generalization to d-DNNF

Connection (Bova, Capelli, M, Slivovsky '16)

Functions with high multipartition communication complexity have no small d-DNNF.

Theorem (essentially Duris et al. '04)

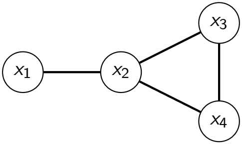
There are CNF with high multipartition communication complexity.

- lets us show lower bounds for d-DNNF "easily" by using CC literature
- several variations of this for other data structures from knowledge compilation

Graphs and CNF

- assign graph to every CNF formula

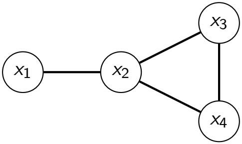
$$(\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



Graphs and CNF

- assign graph to every CNF formula

$$(\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$



- idea: if graphs are "easy", maybe we can compile efficiently

Compilation on Trees

Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Compilation on Trees

Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Proof (sketch).

Assume that CNF is monotone, so all clauses $x \vee y$

Compilation on Trees

Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Proof (sketch).

Assume that CNF is monotone, so all clauses $x \vee y$
Compute for root x d-DNNF $C_{x,0}$, $C_{x,1}$ for solutions when x is $0/1$

Compilation on Trees

Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Proof (sketch).

Assume that CNF is monotone, so all clauses $x \vee y$

Compute for root x d-DNNF $C_{x,0}$, $C_{x,1}$ for solutions when x is 0/1

- only edge xy: $C_{x,0} = \neg x \wedge y$ and $C_{x,1} = x$
- otherwise x with children $y_1, \ldots, y_s$

$$C_{x,o} = \neg x \wedge C_{y_1,1} \wedge \cdots \wedge C_{y_s,1}$$

$$C_{x,1} = x \wedge (C_{y_1,0} \vee C_{y_1,1}) \wedge \cdots \wedge (C_{y_s,0} \vee C_{y_s,1})$$

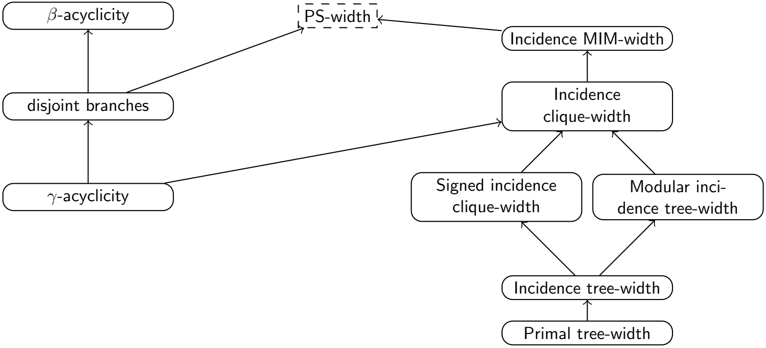
compilation result $C = C_{x,0} \vee C_{x_1}$ □

Graph Decomposition Techniques and Knowledge Compilation

generally, assigning (hyper)graphs to CNF gives tractable classes

Graph Decomposition Techniques and Knowledge Compilation

generally, assigning (hyper)graphs to CNF gives tractable classes



Parameter of the hypergraph **Parameter of the incidence graph**

(Samer, Szeider '10; Paulusma, Slivovsky, Szeider '13; Fisher, Makowsky, Ravve '08; Slivovsky, Szeider '14; Capelli, M, Durand '14; Brault-Baron, Capelli, M '15; Sæther, Telle, Vatshelle '14, Bova, Capelli, M, Slivovsky '15)

Treewidth vs. Cliquewidth

- d-DNNF size for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT size far preferable

Treewidth vs. Cliquewidth

- d-DNNF size for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT size far preferable

Question

Can DNNF size for cliquewidth be improved to FPT?

Treewidth vs. Cliquewidth

- d-DNNF size for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT size far preferable

Question

Can DNNF size for cliquewidth be improved to FPT?

Theorem (roughly, M '16)

Cliquewidth k formulas generally require DNNF size n^k

Treewidth vs. Cliquewidth

- d-DNNF size for formula F
  - treewidth k FPT: $2^k|F|$
  - cliquewidth k not FPT: $|F|^k$
- FPT size far preferable

Question

Can DNNF size for cliquewidth be improved to FPT?

Theorem (roughly, M '16)

Cliquewidth k formulas generally require DNNF size n^k

- proved with communication complexity approach
- hard instances random systems of linear equations

Conclusion

- data structures from artificial intelligence with many connections to other areas
  - database theory
  - (communication/circuit) complexity
  - graph theory
  - lower bounds for algorithms

Conclusion

- data structures from artificial intelligence with many connections to other areas
  - database theory
  - (communication/circuit) complexity
  - graph theory
  - lower bounds for algorithms
- some things I could not talk about
  - factorized databases
  - constant delay enumeration algorithms (with updates)
  - qualitative differences between counting algorithms

Conclusion

- data structures from artificial intelligence with many connections to other areas
  - database theory
  - (communication/circuit) complexity
  - graph theory
  - lower bounds for algorithms
- some things I could not talk about
  - factorized databases
  - constant delay enumeration algorithms (with updates)
  - qualitative differences between counting algorithms

Thank you for your attention!