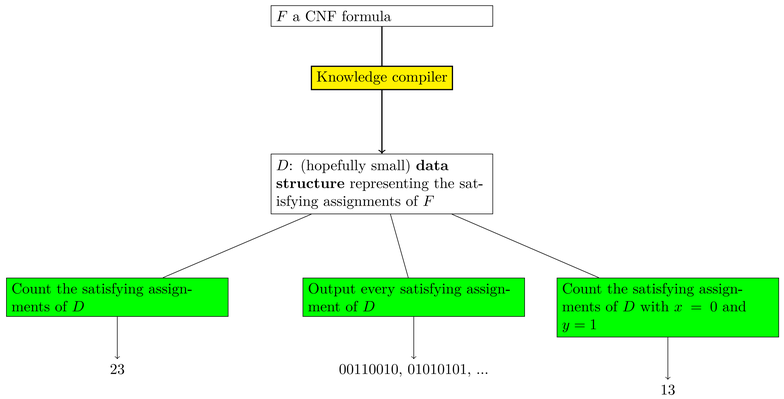
Certifying Top-Down Decision-DNNF Compilers

Florent Capelli[1], Jean-Marie Lagniez[2], Pierre Marquis[2,3]

[1] CRIStAL, Université de Lille & Inria & CNRS France
[2] CRIL, Université d'Artois & CNRS France
[3] IUF, France

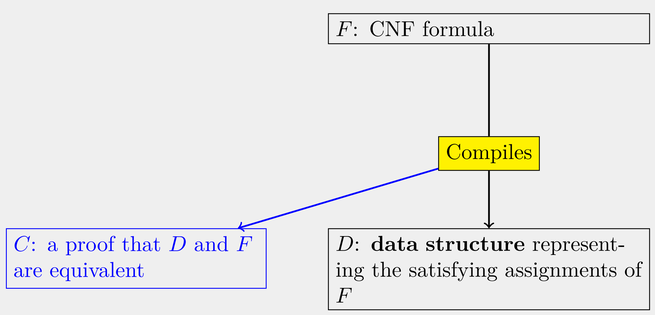
Certifying Knowledge Compiler



Can we trust the compilation procedure?

Problem

Can we check whether the constructed data structure represents the input CNF formula? **coNP-hard** for most interesting data structures

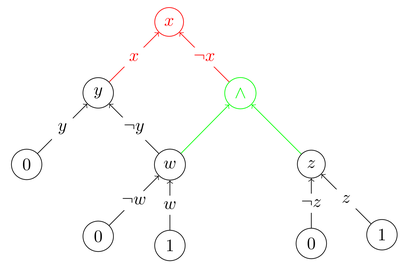


Using C, one can check in time polynomial in $|C|, |D|$ and $|F|$ whether D represents F.

Top-Down Knowledge Compiler

- Based on exhaustive DPLL: $F = (x \wedge F[x = 0]) \vee (\neg x \wedge F[x = 1])$
- Detect independent formulas $F = F_1 \wedge F_2$ with $\mathsf{var}(F_1) \cap \mathsf{var}(F_2) = \emptyset$
- Oracle call to SAT-solver:
  - Avoid UNSAT branches.
  - Use **learnt clauses** to discover unit propagation earlier

Compiling $F = (\neg x \vee \neg y) \wedge (x \vee z) \wedge (x \vee w) \wedge (\neg x \vee y \vee w)$ into a Decision-DNNF circuit :



To check for equivalence, we can replay the computation... but we need to record more than the Decision-DNNF circuit.

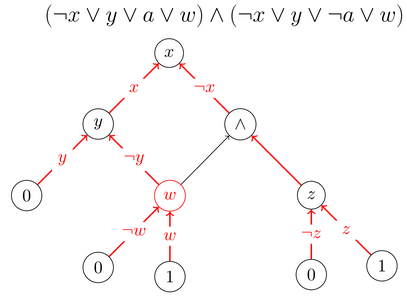
Certifiable Decision-DNNF

A certifiable Decision-DNNF circuit:

- A Decision-DNNF circuit
- A set of **learnt clauses**
- For each gate v, a canonical father $e(v)$

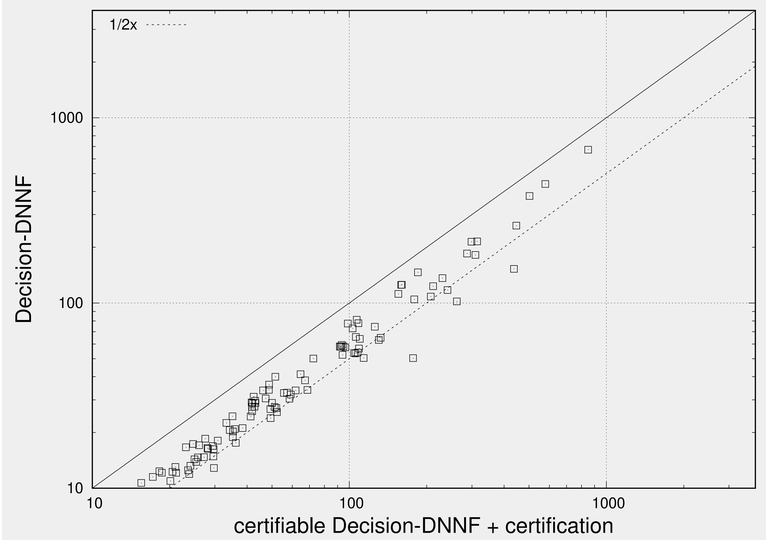
Intuitively, the canonical father records the order of gates creations by the solver

$$F = (\neg x \vee \neg y) \wedge (x \vee z) \wedge (x \vee a \vee w) \wedge (x \vee \neg a \vee w) \wedge (\neg x \vee y \vee a \vee w) \wedge (\neg x \vee y \vee \neg a \vee w)$$



Results on a Modification of D4

Time



Size