Knowledge Compilation

Simone Bova

Algorithms and Complexity Group Retreat
Semmering, June 29-May 1, 2015

Area

- Artificial Intelligence

  AAAI, IJCAI, KR, . . .
- Logic in Computer Science

  SAT, . . .
- Theory

  CCC, ICALP, STACS, . . .

Generalities

- FWF Project currently led by Stefan
  (supporting Ronald and Simone)
- International Workshop recently organized at TU
  (`http://www.vcla.at/kc2015`)

Compilation

$$\phi \longmapsto c(\phi)$$

where:

- ϕ represents some propositional knowledge encoded in some *source* language (say CNFs);
- $c(\phi)$ represents the same knowledge *compiled* in some *target* language (say OBDDs).

Compilation

$$\phi \longmapsto c(\phi)$$

where:

- ϕ represents some propositional knowledge encoded in some *source* language (say CNFs);
- $c(\phi)$ represents the same knowledge *compiled* in some *target* language (say OBDDs).

Allow a computationally hard compilation map $c(\cdot)$, as long as $c(\phi)$ is "succinct" and "tractable".

Practice

A query of interest is posed multiple times
to the knowledge base.

Practice

A query of interest is posed multiple times to the knowledge base.

The query is hard on the source language, but polytime on the target language.

Practice

A query of interest is posed multiple times
to the knowledge base.

The query is hard on the source language,
but polytime on the target language.

The high compilation cost is amortized,
by reusing the compiled knowledge over multiple queries.

Example

Clause entailment (CE) queries are hard on CNFs,
but OBDDs support CE in polytime.

Example

Clause entailment (CE) queries are hard on CNFs,
but OBDDs support CE in polytime.

In practice,

$$\phi \overset{?}{\models} \delta_1, \phi \overset{?}{\models} \delta_2, \ldots, \phi \overset{?}{\models} \delta_i, \ldots$$

Example

Clause entailment (CE) queries are hard on CNFs,
but OBDDs support CE in polytime.

In practice,

$$\phi \overset{?}{\models} \delta_1, \phi \overset{?}{\models} \delta_2, \ldots, \phi \overset{?}{\models} \delta_i, \ldots$$

is outperformed by

$$c(\phi) = \chi, \chi \overset{?}{\models} \delta_1, \chi \overset{?}{\models} \delta_2, \ldots, \chi \overset{?}{\models} \delta_i, \ldots$$

Theory

Let S and T be representation languages
(a *representation language* is a class of circuits).

S is *polysize compilable* into T (or T *is at least as succinct as S*) if for all $C \in$ S there exists $D \in$ T st:

- D is equivalent to C;
- D is polysize in C.

Example

$\mathrm{XOR}_n(x_1, \ldots, x_n) = 1$ iff $x_1 + \ldots + x_n \equiv 1 \pmod 2$:

Example

$\mathrm{XOR}_n(x_1, \ldots, x_n) = 1$ iff $x_1 + \ldots + x_n \equiv 1 \pmod 2$:

- CNF size is $2^{\Omega(n)}$ (claim);

Example

$\mathrm{XOR}_n(x_1, \ldots, x_n) = 1$ iff $x_1 + \ldots + x_n \equiv 1 \pmod 2$:

- CNF size is $2^{\Omega(n)}$ (claim);
- OBDD size is $2n + 1$ (board).

Example

$\mathrm{XOR}_n(x_1, \ldots, x_n) = 1$ iff $x_1 + \ldots + x_n \equiv 1 \pmod 2$:

- CNF size is $2^{\Omega(n)}$ (claim);
- OBDD size is $2n + 1$ (board).

On the other hand, there is a class $\{f_n\}_{n \in \mathbb{N}}$ of functions with

- $O(n)$ CNF size;
- $2^{\Omega(n)}$ OBDD size.

Example

$\mathrm{XOR}_n(x_1, \ldots, x_n) = 1$ iff $x_1 + \ldots + x_n \equiv 1 \pmod 2$:

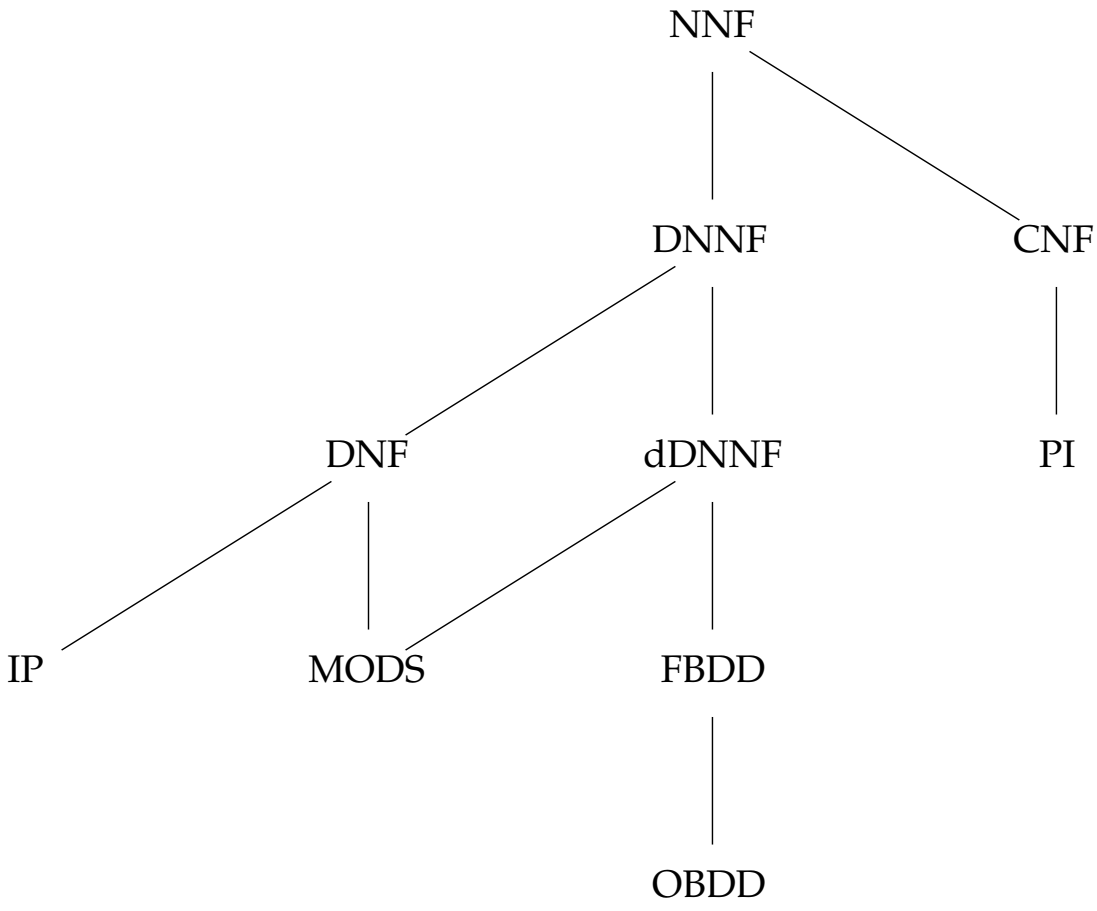
- CNF size is $2^{\Omega(n)}$ (claim);
- OBDD size is $2n + 1$ (board).

On the other hand, there is a class $\{f_n\}_{n \in \mathbb{N}}$ of functions with

- $O(n)$ CNF size;
- $2^{\Omega(n)}$ OBDD size.

CNFs and OBDDs are incomparable in succinctness.

Representation Languages



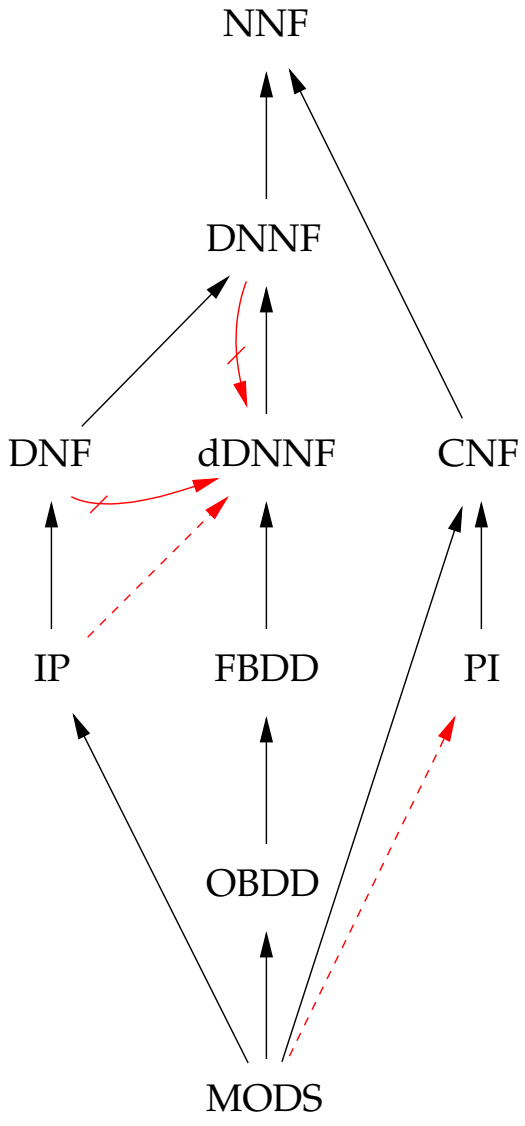
Inclusion relation on representation languages.

Succinctness Relation

The succinctness relation is (partially) established in a series of results:

- Quine (1959),
- Chandra and Markowsky (1978),
- Bryant (1986),
- Wegener (1987),
- Gergov and Meinel (1994),
- Gogic, Kautz, Papdimitriou, and Selman (1995),
- Selman and Kautz (1996),
- Cadoli and Donini (1997),
- Darwiche (1999), and
- B, Capelli, Mengel, and Slivovsky (2015).

Succinctness Relation



S $\dashrightarrow$ T means S $\rightsquigarrow$ T unknown. S $\not\to$ T means S $\not\rightsquigarrow$ T unless PH collapses.

Questions

In choosing a representation language,
there is a tradeoff between “succinctness” and “tractability”.

Questions

In choosing a representation language,
there is a tradeoff between “succinctness” and “tractability”.

Important research questions:

Questions

In choosing a representation language,
there is a tradeoff between "succinctness" and "tractability".

Important research questions:

1. Improve knowledge of existing languages.

Questions

In choosing a representation language,
there is a tradeoff between "succinctness" and "tractability".

Important research questions:

1. Improve knowledge of existing languages.
   - Complexity of equivalence on FBDDs?
     Negation on deterministic DNNFs?

Questions

In choosing a representation language,
there is a tradeoff between "succinctness" and "tractability".

Important research questions:

1. Improve knowledge of existing languages.
   - Complexity of equivalence on FBDDs?
     Negation on deterministic DNNFs?
   - Separate SDDs and OBBDs,
     DNFs and deterministic DNNFs.

Questions

In choosing a representation language,
there is a tradeoff between "succinctness" and "tractability".

Important research questions:

1. Improve knowledge of existing languages.
   - Complexity of equivalence on FBDDs?
     Negation on deterministic DNNFs?
   - Separate SDDs and OBBDs,
     DNFs and deterministic DNNFs.
2. Find new languages challenging existing ones.

Questions

In choosing a representation language,
there is a tradeoff between "succinctness" and "tractability".

Important research questions:

1. Improve knowledge of existing languages.
   - Complexity of equivalence on FBDDs?
     Negation on deterministic DNNFs?
   - Separate SDDs and OBBDs,
     DNFs and deterministic DNNFs.
2. Find new languages challenging existing ones.
   - Find language supporting counting
     but incomparable to deterministic DNNFs.

Parameterizations

S is *fpt size compilable* into T wrt parameterization $\kappa\colon \mathrm{S} \to \mathbb{N}$ if for all $C \in \mathrm{S}$ there exists $D \in \mathrm{T}$ st:

- D is equivalent to C;
- D is fpt size in C wrt parameterization κ.

Parameterizations

S is *fpt size compilable* into T wrt parameterization $\kappa\colon \mathrm{S} \to \mathbb{N}$ if for all $C \in \mathrm{S}$ there exists $D \in \mathrm{T}$ st:

- D is equivalent to C;
- D is fpt size in C wrt parameterization κ.

Research questions:

Parameterizations

S is *fpt size compilable* into T wrt parameterization $\kappa\colon \mathrm{S} \to \mathbb{N}$ if for all $C \in \mathrm{S}$ there exists $D \in \mathrm{T}$ st:

- D is equivalent to C;
- D is fpt size in C wrt parameterization κ.

Research questions:

- Study the succinctness/tractability tradeoff in the parameterized setting.

Parameterizations

S is *fpt size compilable* into T wrt parameterization $\kappa\colon \mathrm{S} \to \mathbb{N}$ if for all $C \in \mathrm{S}$ there exists $D \in \mathrm{T}$ st:

- D is equivalent to C;
- D is fpt size in C wrt parameterization κ.

Research questions:

- Study the succinctness/tractability tradeoff in the parameterized setting.
- Find language supporting fpt size compilation of functions wrt *expression width* (smallest treewidth over equivalent circuits).

Thank you for your attention!