Succinct Compilation of Propositional Theories

Simone Bova

Vienna University of Technology

Universidad del País Vasco
February 6, 2013
Outline

Classical Compilation

Parameterized Compilation

Research Agenda
Outline

Classical Compilation

Parameterized Compilation

Research Agenda
Idea

Many reasoning tasks in artificial intelligence (inference, decision) are \textit{computationally intractable}. 
Idea

Many reasoning tasks in artificial intelligence (inference, decision) are computationally intractable.

Example (LATINCOMPLETION)

<table>
<thead>
<tr>
<th>1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Problem  LATINCOMPLETION

Instance  A partial function $f: [n] \times [n] \to [n]$. 
Many reasoning tasks in artificial intelligence (inference, decision) are \textit{computationally intractable}.

\textbf{Example (LATINCOMPLETION)}

\begin{center}
\begin{tabular}{|c|c|}
\hline
1 & \  \\
\hline
2 & \  \\
\hline
\end{tabular} \quad \sim \quad 
\begin{tabular}{|c|c|c|}
\hline
1 & 3 & 2 \\
\hline
3 & 2 & 1 \\
\hline
2 & 1 & 3 \\
\hline
\end{tabular}
\end{center}

\textbf{Problem} \quad \text{LATINCOMPLETION}

\textbf{Instance} \quad A partial function \( f : [n] \times [n] \to [n] \).

\textbf{Question} \quad Does there exist a \( n \times n \) Latin square extending \( f \)?

\textit{LATINCOMPLETION} is computationally intractable (Colbourn, 1984).
However, part of the information specifying such tasks is typically background knowledge, ie:
### Idea

However, part of the information specifying such tasks is typically *background knowledge*, i.e:

1. *known before* the execution of individual tasks;
However, part of the information specifying such tasks is typically
\textit{background knowledge}, ie:

\begin{enumerate}
\item \textit{known before} the execution of individual tasks;
\item \textit{remains stable} through the execution of several individual tasks.
\end{enumerate}
A proposition is a Boolean formula
(Boolean variables combined by \(\neg, \wedge, \vee\)).

Example (Cont’d)

\[
\begin{array}{ccc}
1 & & \\
 & 2 & \\
 & & \\
\end{array}
\]
Idea

A proposition is a Boolean formula (Boolean variables combined by $\neg$, $\land$, $\lor$).

Example (Cont’d)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is the proposition $\phi_3 \land x_{111} \land x_{222}$ satisfiable?
A proposition is a Boolean formula (Boolean variables combined by $\neg$, $\land$, $\lor$).

Example (Cont’d)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\Rightarrow$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Is the proposition $\phi_3 \land x_{111} \land x_{222}$ satisfiable? Yes!
A proposition is a Boolean formula (Boolean variables combined by $\neg$, $\land$, $\lor$).

Example (Cont’d)

\[
\begin{array}{ccc}
1 & & \\
& 2 & \\
\end{array}
\quad \sim \quad \begin{array}{ccc}
1 & 3 & 2 \\
3 & 2 & 1 \\
2 & 1 & 3
\end{array}
\]

Is the proposition $\phi_3 \land x_{111} \land x_{222}$ satisfiable? Yes!

In the above LatinCompletion instance:

1. $\phi_3$, the propositional theory of the $3 \times 3$ Latin square, is background knowledge (known, stable);
Idea

A proposition is a Boolean formula (Boolean variables combined by $\neg$, $\land$, $\lor$).

Example (Cont’d)

Is the proposition $\phi_3 \land x_{111} \land x_{222}$ satisfiable? Yes!

In the above LATINCOMPLETION instance:

1. $\phi_3$, the propositional theory of the $3 \times 3$ Latin square, is background knowledge (known, stable);
2. $x_{111} \land x_{222}$, the given partial function, is online information (unknown, varying).
Idea

A proposition is a Boolean formula
(Boolean variables combined by ¬, ∧, ∨).

Example (Cont’d)

Is the proposition $\phi_3 \land x_{111} \land x_{222}$ satisfiable? Yes!

In the above LATINCOMPLETION instance:

1. $\phi_3$, the propositional theory of the $3 \times 3$ Latin square,
   is background knowledge (known, stable);

2. $x_{111} \land x_{222}$, the given partial function,
   is online information (unknown, varying).

Infer the solution by combining 1 and 2.
**Idea**

*Example (Cont’d)*

For $(i, j, k) \in [n]^3$, the propositional variable $x_{ijk}$ means, "$(i, j)$ maps to $k$".

$\varphi_n$ is the propositional theory of the $n \times n$ Latin square, i.e., satisfying assignments to $\varphi_n$ correspond to $n \times n$ Latin squares (mapping $(i, j)$ to $k$ iff the assignment maps variable $x_{ijk}$ to $\top$).

$\varphi_n = \varphi_n^1 \land \varphi_n^2 \land \varphi_n^3$ where:

- $\varphi_n^1 = \bigwedge (i, j) \in [n]^2 \left( \bigvee k \in [n] x_{ijk} \land \bigwedge k \in [n] \left( x_{ijk} \rightarrow \bigwedge j \in [n] \left( \neg x_{ij k} \right) \right) \right)$
- $\varphi_n^2 = \bigwedge (i, k) \in [n]^2 \left( \bigvee j \in [n] x_{ijk} \land \bigwedge j \in [n] \left( x_{ijk} \rightarrow \bigwedge i \in [n] \left( \neg x_{i j k} \right) \right) \right)$
- $\varphi_n^3 = \bigwedge (j, k) \in [n]^2 \left( \bigvee i \in [n] x_{ijk} \land \bigwedge i \in [n] \left( x_{ijk} \rightarrow \bigwedge j \in [n] \left( \neg x_{i j k} \right) \right) \right)$
Example (Cont’d)
For \((i, j, k) \in [n]^3\), the propositional variable \(x_{ijk}\) means, “\((i, j)\) maps to \(k\)”.
Idea

Example (Cont’d)

For \((i, j, k) \in [n]^3\), the propositional variable \(x_{ijk}\) means, “\((i, j)\) maps to \(k\)”.

\(\phi_n\) is the propositional theory of the \(n \times n\) Latin square, i.e.,
satisfying assignments to \(\phi_n\) correspond to \(n \times n\) Latin squares
(mapping \((i, j)\) to \(k\) iff the assignment maps variable \(x_{ijk}\) to \(\top\)).
Idea

Example (Cont’d)

For \((i, j, k) \in [n]^3\), the propositional variable \(x_{ijk}\) means, \”\((i, j)\) maps to \(k\)\”.

\(\phi_n\) is the propositional theory of the \(n \times n\) Latin square, ie, satisfying assignments to \(\phi_n\) correspond to \(n \times n\) Latin squares (mapping \((i, j)\) to \(k\) iff the assignment maps variable \(x_{ijk}\) to \(\top\)).

\(\phi_n = \phi_{n1} \land \phi_{n2} \land \phi_{n3}\) where:

\[
\phi_{n1} = \bigwedge_{(i,j) \in [n]^2} \left( \left( \bigvee_{k \in [n]} x_{ijk} \right) \land \left( x_{ijk} \rightarrow \left( \bigwedge_{k \neq k' \in [n]} \neg x_{ij'k} \right) \right) \right),
\]

\[
\phi_{n2} = \bigwedge_{(i,k) \in [n]^2} \left( \left( \bigvee_{j \in [n]} x_{ijk} \right) \land \left( x_{ijk} \rightarrow \left( \bigwedge_{j \neq j' \in [n]} \neg x_{ij'k} \right) \right) \right),
\]

\[
\phi_{n3} = \bigwedge_{(j,k) \in [n]^2} \left( \left( \bigvee_{i \in [n]} x_{ijk} \right) \land \left( x_{ijk} \rightarrow \left( \bigwedge_{i \neq i' \in [n]} \neg x_{i'jk} \right) \right) \right).
\]
Idea

Exploit *background knowledge against computational intractability:*
Idea

Exploit background knowledge against computational intractability:

1. preprocess the background knowledge into a compiled knowledge that allows for solving the reasoning task easily (in polynomial time);
Exploit *background knowledge against computational intractability*:

1. preprocess the background knowledge into a *compiled knowledge* that allows for solving the reasoning task easily (in polynomial time);
2. process *many* individual tasks using the shared compiled knowledge together with task specific online information.
Idea

Exploit *background knowledge against computational intractability*:

1. preprocess the background knowledge into a *compiled knowledge* that allows for solving the reasoning task easily (in polynomial time);

2. process *many* individual tasks using the shared compiled knowledge together with task specific online information.

*Compilation cost is amortized by reusing compiled knowledge* to ease a large number of individual executions.
Entailment

The key problem in knowledge compilation since the 90s:

**Problem**  ClauseEntailment

**Instance**  A proposition $\phi$ (theory) and a clause $\delta$ (query).

**Question**  $\phi \models \delta$?
Entailment

The key problem in knowledge compilation since the 90s:

**Problem**  **Clause Entailment**

**Instance** A proposition $\phi$ (theory) and a clause $\delta$ (query).

**Question** $\phi \models \delta$?

<table>
<thead>
<tr>
<th>$w$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$(x \lor z) \land (x \lor y) \land (\neg w \lor y \lor \neg z) \land (\neg w \lor \neg x \lor \neg y)$</th>
<th>$\neg w \lor \neg y \lor z$</th>
<th>$y \lor z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Entailment

ClauseEntailment is computationally intractable (coNP-hard).
Entailment

ClauseEntailment is computationally intractable (coNP-hard).

Take $\phi$, the theory, as background knowledge and $\delta$, the query, as online information (practical case in artificial intelligence).
Entailment

Clause Entailment is computationally intractable (coNP-hard).

Take $\phi$, the theory, as *background knowledge* and $\delta$, the query, as *online information* (practical case in artificial intelligence).

**Definition (Compilation)**

A *compilation* is a (computable) map $c$ st for all $\phi$ and $\delta$:

1. $c(\phi) \models \delta$ iff $\phi \models \delta$ (i.e., $c(\phi)$ logically equivalent to $\phi$);
2. $c(\phi) \models \delta$ is poly-time decidable.
Entailment

Clause Entailment is *computationally intractable* (coNP-hard).

Take $\phi$, the theory, as *background knowledge* and $\delta$, the query, as *online information* (practical case in artificial intelligence).

**Definition (Compilation)**

A *compilation* is a (computable) map $c$ st for all $\phi$ and $\delta$:

1. $c(\phi) \models \delta$ iff $\phi \models \delta$ (ie, $c(\phi)$ logically equivalent to $\phi$);
2. $c(\phi) \models \delta$ is poly-time decidable.

A series of *hard* instances,

$$(\phi, \delta_1)$$
$$(\phi, \delta_2)$$
$$(\phi, \delta_3)$$
$$\vdots$$
Entailment

Clause Entailment is computationally intractable (coNP-hard).

Take $\phi$, the theory, as background knowledge and $\delta$, the query, as online information (practical case in artificial intelligence).

Definition (Compilation)

A compilation is a (computable) map $c$ st for all $\phi$ and $\delta$:

1. $c(\phi) \models \delta$ iff $\phi \models \delta$ (ie, $c(\phi)$ logically equivalent to $\phi$);
2. $c(\phi) \models \delta$ is poly-time decidable.

A series of hard instances, compiles into a series of easy equivalent instances:

$$(\phi, \delta_1) \leadsto (c(\phi), \delta_1)$$
$$(\phi, \delta_2) \leadsto (c(\phi), \delta_2)$$
$$(\phi, \delta_3) \leadsto (c(\phi), \delta_3)$$
$$\vdots \quad \vdots \quad \vdots$$
Entailment

Example (Compilation into DNF)

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:
Entailment

Example (Compilation into DNF)
Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
Entailment

Example (Compilation into DNF)

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$
Entailment

Example (Compilation into DNF)

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$: 
Entailment

Example (Compilation into DNF)

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$

$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$c(\phi) \models \delta$ iff $c(\phi) \land \neg \delta$ unsatisfiable,
Entailment

Example (Compilation into DNF)
Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$c(\phi) \models \delta$ iff $c(\phi) \land \neg \delta$ unsatisfiable,
iff $c(\phi) \land (x_3 \land \neg x_4)$ unsatisfiable,
**Entailment**

**Example (Compilation into DNF)**

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$
\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),
$$

$$
c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).
$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$c(\phi) \models \delta$ iff $c(\phi) \land \neg \delta$ unsatisfiable,

iff $c(\phi) \land (x_3 \land \neg x_4)$ unsatisfiable,

iff $((x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4)) \land (x_3 \land \neg x_4)$ unsatisfiable,
Entailment

Example (Compilation into DNF)

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$c(\phi) \models \delta$ iff $c(\phi) \land \neg \delta$ unsatisfiable,
iff $c(\phi) \land (x_3 \land \neg x_4)$ unsatisfiable,
iff $((x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4)) \land (x_3 \land \neg x_4)$ unsatisfiable,
iff $x_1 \lor x_2$ unsatisfiable (false).
Example (Compilation into DNF)

Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$c(\phi) \models \delta$ iff $c(\phi) \land \neg \delta$ unsatisfiable,

iff $c(\phi) \land (x_3 \land \neg x_4)$ unsatisfiable,

iff $((x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4)) \land (x_3 \land \neg x_4)$ unsatisfiable,

iff $x_1 \lor x_2$ unsatisfiable (false).

Thus, CLAUSE_ENTAILMENT compiles via such $c$: 
Entailment

Example (Compilation into DNF)
Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$$c(\phi) \models \delta \text{ iff } c(\phi) \land \neg \delta \text{ unsatisfiable,}$$

$$\text{iff } c(\phi) \land (x_3 \land \neg x_4) \text{ unsatisfiable,}$$

$$\text{iff } ((x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4)) \land (x_3 \land \neg x_4) \text{ unsatisfiable,}$$

$$\text{iff } x_1 \lor x_2 \text{ unsatisfiable (false).}$$

Thus, CLAUSEENTAILMENT compiles via such $c$:

**1.** $c(\phi) \models \delta \text{ iff } \phi \models \delta$ for all $\delta$;
Entailment

Example (Compilation into DNF)
Compile $\phi$ into DNF $c(\phi)$ logically equivalent to $\phi$, eg:

$$\phi = (x_1 \lor x_2) \land (x_3 \lor x_4),$$
$$c(\phi) = (x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4).$$

Check $c(\phi) \models \delta$, eg, $\delta = \neg x_3 \lor x_4$:

$c(\phi) \models \delta$ iff $c(\phi) \land \neg \delta$ unsatisfiable,
iff $c(\phi) \land (x_3 \land \neg x_4)$ unsatisfiable,
iff $((x_1 \land x_3) \lor (x_1 \land x_4) \lor (x_2 \land x_3) \lor (x_2 \land x_4)) \land (x_3 \land \neg x_4)$ unsatisfiable,
iff $x_1 \lor x_2$ unsatisfiable (false).

Thus, CLAUSEENTAILMENT compiles via such $c$:

1. $c(\phi) \models \delta$ iff $\phi \models \delta$ for all $\delta$;
2. $c(\phi) \models \delta$ is poly-time decidable (reduction to DNF satisfiability, easy).
Succinctness

Example (Compilation into DNF, Cont’d)

- $\phi = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \cdots \land (x_{n-1} \lor x_n)$ is size $|\phi| = n$;
Succinctness

Example (Compilation into DNF, Cont’d)

- $\phi = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \cdots \land (x_{n-1} \lor x_n)$ is size $|\phi| = n$;
- $|c(\phi)| \geq |(x_1 \land x_3 \land \cdots \land x_{n-1}) \lor \cdots \lor (x_2 \land x_4 \land \cdots \land x_n)| \geq 2^{n/2} \cdot n/2$;
**Succinctness**

*Example (Compilation into DNF, Cont’d)*

- $\phi = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \ldots \land (x_{n-1} \lor x_n)$ is *size* $|\phi| = n$;
- $|c(\phi)| \geq |(x_1 \land x_3 \land \ldots \land x_{n-1}) \lor \ldots \lor (x_2 \land x_4 \land \ldots \land x_n)| \geq 2^{n/2} \cdot n/2$;
- $|c(\phi)|$ is *not polynomially bounded* in the size of $|\phi|$.
Succinctness

Example (Compilation into DNF, Cont’d)

- \( \phi = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \cdots \land (x_{n-1} \lor x_n) \) is size \( |\phi| = n \);
- \( |c(\phi)| \geq |(x_1 \land x_3 \land \cdots \land x_{n-1}) \lor \cdots \lor (x_2 \land x_4 \land \cdots \land x_n)| \geq 2^{n/2} \cdot n/2 \);
- \( |c(\phi)| \) is not polynomially bounded in the size of \( |\phi| \).

Definition (Succinctness)

A compilation \( c \) is succinct if \( |c(\phi)| \) is polynomially bounded in \( |\phi| \), ie, there exists \( d \) st for all \( \phi \),

\[
|c(\phi)| \in O(|\phi|^d).
\]
**Succinctness**

*Example (Compilation into DNF, Cont’d)*

- $\phi = (x_1 \lor x_2) \land (x_3 \lor x_4) \land \cdots \land (x_{n-1} \lor x_n)$ is size $|\phi| = n$;
- $|c(\phi)| \geq |(x_1 \land x_3 \land \cdots \land x_{n-1}) \lor \cdots \lor (x_2 \land x_4 \land \cdots \land x_n)| \geq 2^{n/2} \cdot n/2$;
- $|c(\phi)|$ is not polynomially bounded in the size of $|\phi|$.

**Definition (Succinctness)**

A compilation $c$ is succinct if $|c(\phi)|$ is polynomially bounded in $|\phi|$, ie, there exists $d$ s.t for all $\phi$,

$$|c(\phi)| \in O(|\phi|^d).$$

**Remark**

*Without succinctness, ClauseEntailment compiles even requiring that $c(\phi) \models \delta$ is decidable in time $O(|\phi|^d)$.*
LITERAL Entailment is Clause Entailment restricted to instances $(\phi, \delta)$ where $\delta$ is a literal.
Compilability

\textsc{LiteralEntailment} is \textsc{ClauseEntailment} restricted to instances \((\phi, \delta)\) where \(\delta\) is a literal.

\textit{Fact}
\textsc{LiteralEntailment} \textit{compiles succinctly}.
Compilability

LiteralEntailment is ClauseEntailment restricted to instances \((\phi, \delta)\) where \(\delta\) is a literal.

**Fact**

LiteralEntailment compiles succinctly.

**Proof.**

The map \(c\) sends \(\phi\) to \(c(\phi)\), the conjunction of all literals entailed by \(\phi\) (computing \(c\) involves solving \(\leq |\phi|\) many instances of a coNP-hard problem). For all literals \(\delta\), clearly \(c(\phi) \models \delta\) is poly-time decidable (check \(\delta\) occurs in \(c(\phi)\) as a conjunct), \(c(\phi) \models \delta\) iff \(\phi \models \delta\). Moreover, \(|c(\phi)| \leq |\phi|\), thus \(c\) is succinct.
Classical Compilability | Incompilability

Theorem (Selman and Kautz, 1996)

ClauseEntailment does not compile succinctly
(under standard assumptions in complexity theory).
Theorem (Selman and Kautz, 1996)

**Clause Entailment** does not compile succinctly
(under standard assumptions in complexity theory).

**Proof.**

Suppose not. Let \( n \in \mathbb{N} \).

Key observation (easy). There exists a proposition \( \tau_n \) of size \( O(n^3) \) st for all 3CNF \( \chi \) on \( n \) variables, there exists a clause \( \delta_\chi \) st \( \tau_n \models \delta_\chi \) if and only if \( \chi \) is unsatisfiable.

Let \( \tau_n \mapsto c(\tau_n) \) be a succinct compilation of \( \tau_n \).

We give a polynomial-time algorithm for the satisfiability of 3CNFs on \( n \) variables, ie, 3SAT in \( \text{P/poly} \) which implies \( \text{NP} \subseteq \text{P/poly} \) and thus \( \text{PH} \) collapses to \( \Sigma_2^p \) (Karp and Lipton, 1980).

The algorithm, given a propositional formula \( \chi \) on \( n \) variables, decides in polynomial-time the question \( c(\tau_n) \models \delta_\chi \) (here \( c(\tau_n) \) is the advice), and reports that \( \chi \) is satisfiable if and only if the answer is negative.
### Outline

<table>
<thead>
<tr>
<th>Classical Compilation</th>
<th>Parameterized Compilation</th>
<th>Research Agenda</th>
</tr>
</thead>
</table>

**Classical Compilation**

**Parameterized Compilation**

**Research Agenda**
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:

1. solvable in exponential time $O(d^n)$ with $d < 2$;
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:

1. solvable in exponential time $O(d^n)$ with $d < 2$;
2. believed not solvable in subexponential time $2^{o(n)}$. 
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:

1. solvable in exponential time $O(d^n)$ with $d < 2$;
2. believed not solvable in subexponential time $2^{o(n)}$.

**Theorem**

3SAT is solvable in time $O(k2^k \cdot n)$ where $k$ is the treewidth of the instance.
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:
1. solvable in exponential time $O(d^n)$ with $d < 2$;
2. believed not solvable in subexponential time $2^{o(n)}$.

Theorem
3SAT is is solvable in time $O(k2^k \cdot n)$ where $k$ is the treewidth of the instance

$O(k2^k \cdot n)$ faster than $O(d^n)$ if $k$ is much smaller than $n$ ($k \ll n$).
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:
1. solvable in exponential time $O(d^n)$ with $d < 2$;
2. believed not solvable in subexponential time $2^{o(n)}$.

**Theorem**
3SAT is is solvable in time $O(k2^k \cdot n)$ where $k$ is the treewidth of the instance

$O(k2^k \cdot n)$ faster than $O(d^n)$ if $k$ is much smaller than $n$ ($k \ll n$).

**Example**
Treewidth $\text{tw}(\phi)$ of typical industrial instance $\phi$ on 2000 vars is $< 10$. 
Fixed-Parameter Tractability

3SAT: Given a 3CNF $\phi$ on $n$ variables, is $\phi$ satisfiable?

3SAT is NP-hard:
1. solvable in exponential time $O(d^n)$ with $d < 2$;
2. believed not solvable in subexponential time $2^{o(n)}$.

**Theorem**
3SAT is solvable in time $O(k2^k \cdot n)$ where $k$ is the treewidth of the instance, ie, 3SAT is fixed-parameter tractable wrt parameterization $tw$, ie, it has a runtime of the form $f(tw(\phi))|\phi|^d$ for some constant $d$ and function $f$.

$O(k2^k \cdot n)$ faster than $O(d^n)$ if $k$ is much smaller than $n$ ($k \ll n$).

**Example**
Treewidth $tw(\phi)$ of typical industrial instance $\phi$ on 2000 vars is $< 10$. 
Example

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]
Treewidth

Example

$$\phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7).$$

Figure: Primal graph of $\phi$. 
Example

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

**Figure:** \{1, 4\}, \{2, 3\}, \{5, 6, 8\}, \{7\} 4-bramble implies \( \text{tw}(\phi) \geq 3 \).
**Example**

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

*Figure:* Primal graph of \( \phi \). Elimination 2, 1, 6, 5, 4, 3, 8, 7 gives \( \text{tw}(\phi) \leq 3 \).
Treewidth

Example

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

Figure: Eliminating 2, neighborhood size |\{3, 4\}| = 2 \ldots
**Example**

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

*Figure:* Eliminating 1, neighborhood size \(|\{4, 7\}| = 2 \ldots*
**Example**

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

*Figure:* Eliminating 6, neighborhood size \(|\{5, 8\}| = 2\ldots*
Treewidth

Example

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

Figure: Eliminating 5, neighborhood size \( |\{3, 7, 8\}| = 3 \ldots \)
Example

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

**Figure:** Eliminating 4, neighborhood size \(|\{3, 7, 8\}| = 3\).
Treewidth

Example

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

Figure: Eliminating 4, neighborhood size \(|\{3, 7, 8\}| = 3\). Done.
**Treewidth**

**Example**

\[ \phi = (\neg x_7 \lor \neg x_5 \lor \neg x_3) \land (x_4 \lor x_2 \lor \neg x_3) \land (\neg x_3 \lor \neg x_8 \lor \neg x_4) \land (\neg x_8 \lor x_6 \lor \neg x_5) \land (x_4 \lor \neg x_1 \lor \neg x_7). \]

*Figure:* \( \text{tw}(\phi) = 3. \)
Clause Entailment: Given \((\phi, \delta)\), does \(\phi \models \delta\)?
**Parameterized Compilation**

**Clause Entailment:** Given $(\phi, \delta)$, does $\phi \models \delta$?

A *parameterization* is a map $\kappa$ sending pairs $(\phi, \delta)$ into $\mathbb{N}$. 
**Parameterized Compilation**

**Clause Entailment:** Given $(\phi, \delta)$, does $\phi \models \delta$?

A *parameterization* is a map $\kappa$ sending pairs $(\phi, \delta)$ into $\mathbb{N}$.

**Definition (Parametrically Succinct Compilation)**

Let $\kappa$ be a parameterization. A compilation $c$ is (wrt parameterization $\kappa$):
**Clause Entailment:** Given $(\phi, \delta)$, does $\phi \models \delta$?

A parameterization is a map $\kappa$ sending pairs $(\phi, \delta)$ into $\mathbb{N}$.

**Definition (Parametrically Succinct Compilation)**
Let $\kappa$ be a parameterization. A compilation $c$ is (wrt parameterization $\kappa$):

1. **kernel-size** if $|c(\phi)| \leq f(\kappa(\phi, \delta))$ for some function $f$;
**Clause Entailment:** Given \((\phi, \delta)\), does \(\phi \models \delta\)?

A parameterization is a map \(\kappa\) sending pairs \((\phi, \delta)\) into \(\mathbb{N}\).

**Definition (Parametrically Succinct Compilation)**

Let \(\kappa\) be a parameterization. A compilation \(c\) is (wrt parameterization \(\kappa\)):

1. **kernel-size** if \(|c(\phi)| \leq f(\kappa(\phi, \delta))\) for some function \(f\);

2. **fpt-size** (or fixed-parameter tractable in size) if
   \[|c(\phi)| \leq f(\kappa(\phi, \delta)) \cdot |(\phi, \delta)|^d\] for some function \(f\) and constant \(d\).
Parameterized Compilation

\textsc{ClauseEntailment} fails classical compilation, ie, does not compile succinctly (unless PH collapses).
Parameterized Compilation

ClauseEntailment fails classical compilation, i.e., does not compile succinctly (unless PH collapses).

Can we relativize classical incompilability by parametrized compilability? I.e:
Parameterized Compilation

CLAUSEENTAILMENT fails classical compilation, ie, does not compile succinctly (unless PH collapses).

Can we relativize classical incompilability by parameterized compilability? Ie:

1. find parameterizations $\kappa$ st CLAUSEENTAILMENT compiles in kernel-size (wrt $\kappa$);
Parameterized Compilation

ClauseEntailment fails classical compilation, ie, does not compile succinctly (unless PH collapses).

Can we relativize classical incompilability by parameterized compilability? Ie:

1. find parameterizations \( \kappa \) st ClauseEntailment compiles in kernel-size (wrt \( \kappa \));

2. find parameterizations \( \kappa \) st ClauseEntailment compiles in fpt-size (wrt \( \kappa \)).
Parameterized Compilation

ClauseEntailment fails classical compilation, ie, does not compile succinctly (unless PH collapses).

Can we relativize classical incompilability by parameterized compilability? Ie:

1. find parameterizations $\kappa$ st ClauseEntailment compiles in kernel-size (wrt $\kappa$);
2. find parameterizations $\kappa$ st ClauseEntailment compiles in fpt-size (wrt $\kappa$).

Remark

1. There are examples witnessing (1) kernel-size compilability, (2 and not 1) fpt-size compilability but kernel-size incompilability, and (not 2) fpt-size incompilability.
Parameterized Compilation

CLAUSE.Entailment fails classical compilation, ie, does not compile succinctly (unless PH collapses).

Can we relativize classical incompilability by parametrized compilability? Ie:

1. find parameterizations $\kappa$ st CLAUSE.Entailment compiles in kernel-size (wrt $\kappa$);
2. find parameterizations $\kappa$ st CLAUSE.Entailment compiles in fpt-size (wrt $\kappa$).

Remark

1. There are examples witnessing (1) kernel-size compilability, (2 and not 1) fpt-size compilability but kernel-size incompilability, and (not 2) fpt-size incompilability.
2. Parameterizations $\kappa$ yielding fixed-parameter tractability of CLAUSE.Entailment are uninteresting wrt parameterized compilation.
### Implicates

\( \phi \) is a proposition, \( \delta \) is a clause:

1. \( \delta \) implicate of \( \phi \) if \( \phi \models \delta \) and \( \top \not\models \delta \);
2. \( \delta \) prime implicate of \( \phi \) if, \( \phi \models \delta' \) implies \( \delta \models \delta' \) for all implicates \( \delta' \) of \( \phi \).

\( \pif(\phi) \), prime implicate form of \( \phi \), is conjunction of prime implicates of \( \phi \).

1. For all clauses \( \delta \), \( \phi \models \delta \) iff \( \delta_i \models \delta \) for some clause \( \delta_i \) of \( \pif(\phi) \).
2. \( \pif(\phi) \models \delta \) is poly-time.
3. \( \pif(\phi) \) is logically equivalent to \( \phi \).

**Remark**
Prime implicate forms can be redundant. Irredundant prime implicate forms are not unique.
Implicates

\( \phi \) is a proposition, \( \delta \) is a clause:

1. \( \delta \) implicate of \( \phi \) if \( \phi \vdash \delta \) and \( \top \nvdash \delta \);
Implicates

φ is a proposition, δ is a clause:

1. δ implicate of φ if φ ⪰ δ and ⊤ ⪯ δ;
2. δ prime implicate of φ if, φ ⪰ δ′ ⪰ δ implies δ ⪯ δ′ for all implicates δ′ of φ.
Implicates

φ is a proposition, δ is a clause:

1. δ implicate of φ if φ ⊨ δ and ⊤ \not\models δ;
2. δ prime implicate of φ if, 
   φ ⊨ δ′ ⊨ δ implies δ ⊨ δ′ for all implicates δ′ of φ.

pif(φ), prime implicate form of φ, is conjunction of prime implicates of φ.
Implicates

\(\phi\) is a proposition, \(\delta\) is a clause:

1. \(\delta\) implicate of \(\phi\) if \(\phi \models \delta\) and \(\top \not\models \delta\);
2. \(\delta\) prime implicate of \(\phi\) if,
   \(\phi \models \delta' \models \delta\) implies \(\delta \models \delta'\) for all implicates \(\delta'\) of \(\phi\).

\(\text{pif}(\phi)\), prime implicate form of \(\phi\), is conjunction of prime implicates of \(\phi\).

**Fact**

1. For all clauses \(\delta\), \(\phi \models \delta\) iff \(\delta_i \models \delta\) for some clause \(\delta_i\) of \(\text{pif}(\phi)\).
Implicates

φ is a proposition, δ is a clause:

1. δ implicate of φ if φ ⊨ δ and ⊤ ⊭ δ;
2. δ prime implicate of φ if,
   φ ⊨ δ′ ⊨ δ implies δ ⊨ δ′ for all implicates δ′ of φ.

pif(φ), prime implicate form of φ, is conjunction of prime implicates of φ.

Fact

1. For all clauses δ, φ ⊨ δ iff δi ⊨ δ for some clause δi of pif(φ).
2. pif(φ) ⊨ δ is poly-time.
Implicates

φ is a proposition, δ is a clause:

1. δ implicate of φ if φ ⊨ δ and ⊤ \not\models δ;
2. δ prime implicate of φ if,
   φ ⊨ δ′ ⊨ δ implies δ ⊨ δ′ for all implicates δ′ of φ.

pif(φ), prime implicate form of φ, is conjunction of prime implicates of φ.

Fact

1. For all clauses δ, φ ⊨ δ iff δ_i ⊨ δ for some clause δ_i of pif(φ).
2. pif(φ) ⊨ δ is poly-time.
3. pif(φ) is logically equivalent to φ.
Implicates

φ is a proposition, δ is a clause:

1. δ implicate of φ if φ \models \delta and \top \not\models \delta;
2. δ prime implicate of φ if,
   φ \models \delta' \models \delta implies δ \models \delta' for all implicates δ' of φ.

pif(φ), prime implicate form of φ, is conjunction of prime implicates of φ.

Fact

1. For all clauses δ, φ \models \delta iff δ_i \models \delta for some clause δ_i of pif(φ).
2. pif(φ) \models \delta is poly-time.
3. pif(φ) is logically equivalent to φ.

Remark

Prime implicate forms can be redundant.
Irredundant prime implicate forms are not unique.
### Implicates

The table below shows the evaluation of the expression $\phi$ with respect to its variables $w, x, y, z$ under various combinations of these variables. The expression is given by:

$$\phi = w \lor x \lor \neg z \lor \neg y \lor \neg x$$

The table indicates the truth values of $\phi$ under different assignments to its variables and their complements. The 0 and 1 entries correspond to the true and false values, respectively.

<table>
<thead>
<tr>
<th>$w$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$\phi$</th>
<th>$x \lor z$</th>
<th>$x \lor y$</th>
<th>$\neg w \lor y \lor \neg z$</th>
<th>$\neg w \lor \neg y \lor z$</th>
<th>$\neg w \lor \neg x \lor \neg z$</th>
<th>$\neg w \lor \neg x \lor \neg y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The expression $\phi$ has 3 irredundant prime implicate forms.
Parameterization

\[ \text{minvar}(\phi, \delta) \] is the smallest \( k \in \mathbb{N} \) such that \( \phi \) is logically equivalent to a proposition on \( k \) variables.
**Kernel-Size Compilation**

**Parameterization**  \( \minvar(\phi, \delta) \) is the smallest \( k \in \mathbb{N} \) such that 
\( \phi \) is logically equivalent to a proposition on \( k \) variables.

**Observation**

**ClauseEntailment** compiles in kernel-size wrt parameterization \( \minvar \).
**Kernel-Size Compilation**

**Parameterization** \( \text{minvar}(\phi, \delta) \) is the smallest \( k \in \mathbb{N} \) such that 
\( \phi \) is logically equivalent to a proposition on \( k \) variables.

**Observation**
ClauseEntailment compiles in kernel-size wrt parameterization \( \text{minvar} \).

**Proof.**
Let \( \phi \) be a proposition. Take \( c(\phi) \) be the prime implicate normal form of \( \phi \) 
(computable by Quine and McCluskey algorithm, hard).
Then \( c(\phi) \) uses exactly \( \text{minvar}(\phi, \delta) = k \) variables, thus \( |c(\phi)| \leq k2^k \). \( \square \)
Parameterization \( \minvar(\phi, \delta) \) is the smallest \( k \in \mathbb{N} \) such that 
\( \phi \) is logically equivalent to a proposition on \( k \) variables.

Observation

\textsc{ClauseEntailment} compiles in kernel-size wrt parameterization \( \minvar \).

Proof.

Let \( \phi \) be a proposition. Take \( c(\phi) \) be the prime implicate normal form of \( \phi \) 
(computable by Quine and Mccluskey algorithm, hard).

Then \( c(\phi) \) uses exactly \( \minvar(\phi, \delta) = k \) variables, thus \( |c(\phi)| \leq k2^k \). \hfill \Box

Conjecture

\textsc{ClauseEntailment} not in fpt-time wrt parameterization \( \minvar \).
Kernel-Size Compilation

\( \mathcal{F} \) class of propositions, \( \kappa \) parameterization.
\( \mathcal{F} \) is \( \kappa \)-bounded if there exists \( k \) s.t. for all \( \phi \in \mathcal{F}, \kappa(\phi) \leq k \).
Kernel-Size Compilation

\( \mathcal{F} \) class of propositions, \( \kappa \) parameterization. \( \mathcal{F} \) is \( \kappa \)-bounded if there exists \( k \) st for all \( \phi \in \mathcal{F} \), \( \kappa(\phi) \leq k \).

\texttt{ClauseEntailment}(\( \mathcal{F} \)) is \texttt{ClauseEntailment} restricted to instances \((\phi, \delta)\) with \( \phi \in \mathcal{F} \).
Kernel-Size Compilation

\[ \mathcal{F} \] class of propositions, \( \kappa \) parameterization.
\( \mathcal{F} \) is \( \kappa \)-bounded if there exists \( k \) st for all \( \phi \in \mathcal{F} \), \( \kappa(\phi) \leq k \).

\text{ClauseEntailment}(\mathcal{F}) \) is \text{ClauseEntailment}
restricted to instances \( (\phi, \delta) \) with \( \phi \in \mathcal{F} \).

**Conjecture**

\text{ClauseEntailment}(\mathcal{F}) \) compiles in constant-size if and only if \( \mathcal{F} \) is minvar-bounded.
Kernel-Size Compilation

\( \mathcal{F} \) class of propositions, \( \kappa \) parameterization.
\( \mathcal{F} \) is \( \kappa \)-bounded if there exists \( k \) st for all \( \phi \in \mathcal{F} \), \( \kappa(\phi) \leq k \).

**Conjecture**

\textsc{ClauseEntailment}(\( \mathcal{F} \)) is \textsc{ClauseEntailment} restricted to instances \( (\phi, \delta) \) with \( \phi \in \mathcal{F} \).

The proposition gives sufficiency (necessity is open).
**Parameterization**  
\( \text{mintw}(\phi, \delta) \) is the smallest \( k \in \mathbb{N} \) such that \( \phi \) is logically equivalent to a CNF of treewidth \( k \).
Parameterization  \( \text{mintw}(\phi, \delta) \) is the smallest \( k \in \mathbb{N} \) such that \( \phi \) is logically equivalent to a CNF of treewidth \( k \).

Observation  \text{CLAUSE}\text{ENTAILMENT} \text{ compiles in fpt-size wrt parameterization mintw.}
Parameterization\quad mintw(\phi, \delta) is the smallest \( k \in \mathbb{N} \) such that
\( \phi \) is logically equivalent to a CNF of treewidth \( k \).

Observation
ClauseEntailment compiles in fpt-size wrt parameterization mintw.

Proof.
Let \( \phi \) be a proposition using \( n \) variables. Let \( \phi' \) be an irredundant prime implicate normal form of \( \phi \) with minimum treewidth (among all irredundant prime implicate normal forms of \( \phi \)). Then, \( tw(\phi') = mintw(\phi, \delta) = k \). Take \( c(\phi) \) to be the join tree form (a certain CNF) of a small tree decomposition of \( \phi' \) (computable, hard). Then 
\[ |c(\phi)| \leq k^2 k \cdot n. \]
Parameterization \( \text{mintw}(\phi, \delta) \) is the smallest \( k \in \mathbb{N} \) such that \( \phi \) is logically equivalent to a CNF of treewidth \( k \).

Observation

**CLAUSEENTAILMENT** compiles in fpt-size wrt parameterization \( \text{mintw} \).

Proof.

Let \( \phi \) be a proposition using \( n \) variables. Let \( \phi' \) be an irredundant prime implicate normal form of \( \phi \) with minimum treewidth (among all irredundant prime implicate normal forms of \( \phi \)). Then, \( \text{tw}(\phi') = \text{mintw}(\phi, \delta) = k \). Take \( c(\phi) \) to be the join tree form (a certain CNF) of a small tree decomposition of \( \phi' \) (computable, hard). Then \( |c(\phi)| \leq k2^k \cdot n \).

Conjecture

**CLAUSEENTAILMENT** not in fpt-time neither compiles in kernel-size wrt parameterization \( \text{mintw} \).
Parameterization  \[ \text{clsize}(\phi, \delta) = |\delta| \] is the number of literals in clause \( \delta \).
Fpt-Size Incompilability

Parameterization  \( \text{clsize}(\phi, \delta) = |\delta| \) is the number of literals in clause \( \delta \).

Observation  
\text{ClauseEntailment} does not compile in fpt-size prime implicate form \wrt parameterization \text{clsize}. 
**Fpt-Size Incomplilability**

**Parameterization** \( \text{clsize}(\phi, \delta) = |\delta| \) is the number of literals in clause \( \delta \).

**Observation**

**CLAUSE**\( \text{ENTAILMENT} \) does not compile in fpt-size prime implicate form wrt parameterization clsize.

**Proof.**

Assume \( f \) and \( d \) witness fpt-size compilation \( c \) in prime implicate form, i.e., \(|c(\phi)| \leq f(|\delta|)|\phi|^d \) for all \( \phi \) and \( \delta \). For all \( m, n \in \mathbb{N} \), let

\[
\phi_{mn} = \left( \bigwedge_{(i,j) \in [m] \times [n]} (x_i \lor y_{ij}) \right) \land \left( \bigvee_{i \in [m]} \neg x_i \right).
\]

Then \( |\phi_{mn}| = O(mn) \). Moreover, \( \phi_{mn} \) has \( mn + (n + 1)^m \geq n^m \) prime implicates \( \{y_{11}, \ldots, y_{1n}, \neg x_1\} \times \{y_{21}, \ldots, y_{2n}, \neg x_2\} \times \cdots \times \{y_{m1}, \ldots, y_{mn}, \neg x_m\} \). Therefore \( |c(\phi_{mn})| \geq n^m \). Let \( |\delta| = k \) and \( m, n \in \mathbb{N} \) st \( f(k)|\phi_{mn}|^d < n^m \leq |c(\phi_{mn})|. \) \( \square \)
Fpt-Size Incompilability

**Parameterization** \( \text{clsize}(\phi, \delta) = |\delta| \) is the number of literals in clause \( \delta \).

**Observation**

**ClauseEntailment** does not compile in fpt-size prime implicate form wrt parameterization \( \text{clsize} \).

**Proof.**

Assume \( f \) and \( d \) witness fpt-size compilation \( c \) in prime implicate form, ie, \(|c(\phi)| \leq f(|\delta|)|\phi|^d\) for all \( \phi \) and \( \delta \). For all \( m, n \in \mathbb{N} \), let

\[
\phi_{mn} = \left( \bigwedge_{(i,j) \in [m] \times [n]} (x_i \lor y_{ij}) \right) \land \left( \bigvee_{i \in [m]} \neg x_i \right).
\]

Then \(|\phi_{mn}| = O(mn)|. Moreover, \( \phi_{mn} \) has \( mn + (n + 1)^m \geq n^m \) prime implicants \( \{y_{11}, \ldots, y_{1n}, \neg x_1\} \times \{y_{21}, \ldots, y_{2n}, \neg x_2\} \times \cdots \times \{y_{m1}, \ldots, y_{mn}, \neg x_m\} \). Therefore \(|c(\phi_{mn})| \geq n^m\). Let \(|\delta| = k \) and \( m, n \in \mathbb{N} \) st \( f(k)|\phi_{mn}|^d < n^m \leq |c(\phi_{mn})| \). \( \square \)

**Conjecture**

**ClauseEntailment** does not compile in fpt-size wrt parameterization \( \text{clsize} \).
Outline

Classical Compilation

Parameterized Compilation

Research Agenda
Propositional Logic

Compilation map (Darwiche and Marquis, 2002):

1. propositional reasoning tasks (entailment et cetera);
2. propositional logic formalisms (formulas et cetera).

A certain formalism supports certain tasks in poly-time.
Propositional Logic

Compilation map (Darwiche and Marquis, 2002):

1. propositional reasoning tasks (entailment et cetera);
2. propositional logic formalisms (formulas et cetera).

A certain formalism supports certain tasks in poly-time.

Typical complexity issues within the compilation map (under standard hypotheses in classical complexity):

1. a formalism does not support a task in poly-time;
2. a formalism does not compile into another formalism in poly-size.
Propositional Logic

Compilation map (Darwiche and Marquis, 2002):

1. propositional reasoning tasks (entailment et cetera);
2. propositional logic formalisms (formulas et cetera).

A certain formalism supports certain tasks in poly-time.

Typical complexity issues within the compilation map (under standard hypotheses in classical complexity):

1. a formalism does not support a task in poly-time;
2. a formalism does not compile into another formalism in poly-size.

Revisit complexity issues of the compilation map within parameterized tractability and parameterized compilability.
M. Cadoli, F.M. Donini, P. Liberatore, and M. Schaerf.
Preprocessing of Intractable Problems.

H. Chen.
Parameterized Compilability.

C. Colbourn.
The Complexity of Completing Partial Latin Squares.

A. Darwiche and P. Marquis.
A Knowledge Compilation Map.

The Comparative Linguistics of Knowledge Representation.

P. Mathieu and J.-P. Delahaye.
A Kind of Logical Compilation for Knowledge Bases.

B. Selman and H.A. Kautz.
Knowledge Compilation and Theory Approximation.
Gracias por su atención!