Parameterized Compilability Revisited
Combining parameterized complexity and knowledge compilation

Ronald de Haan
(joint work with Simone Bova, Neha Lodha, and Stefan Szeider)
Setting

KB $D$

yes/no

$q_1$?

$q_2$?

$q_i$?

for $D$:

$q$

$q'$

...
Compilation problems are problems of pairs:

\[ L \subseteq \Sigma^* \times \Sigma^* \]

**Offline part:** fixed knowledge base
**Online part:** differing queries

Graphically:

\[(D, q) \in L \quad \text{iff} \quad q? \text{ yes}\]
Knowledge Compilation

In a picture

1. poly-time queries

2. size: poly in $|\mathcal{D}|$

3. $q$? yes iff $q$? yes

KB $\mathcal{D}$

computable (no time bound)

KB $\mathcal{D}'$
Knowledge Compilation

Formally

$L \subseteq \Sigma^* \times \Sigma^*$ is (poly-size) compilable if there exists a computable function $c : \Sigma^* \rightarrow \Sigma^*$ and a problem $L' \subseteq \Sigma^* \times \Sigma^*$ such that:

1. $L'$ is poly-time decidable
2. $|c(D)| \leq \text{poly}(|D|)$
3. $(D, q) \in L$ if and only if $(c(D), q) \in L'$
Negative compilation results

In a picture

either: compiled knowledge base too large

or: no tractable query answering
Parameterized Compilation

Idea: be more generous for both requirements of compilation by using a problem parameter that captures structure in the input.

- Allow fpt-size compiled knowledge bases,
- and allow fpt-time query answering.
Parameterized Compilation

(see Chen, 2005)

Formally

Problem:

\[ L \subseteq \Sigma^* \times \Sigma^* \]

Parameterization:

\[ \kappa : \Sigma^* \rightarrow \mathbb{N} \]

\( L \) is \textit{fpt-size compilable} if there exists a computable function \( c : \Sigma^* \rightarrow \Sigma^* \), computable functions \( f, g : \mathbb{N} \rightarrow \mathbb{N} \), and a problem \( L' \subseteq \Sigma^* \times \Sigma^* \) with a parameterization \( \kappa' : \Sigma^* \rightarrow \mathbb{N} \) such that:

1. \( L' \) is \textit{fpt-time} decidable (w.r.t. \( \kappa' \))
2. \( |c(D)| \leq f(\kappa(D)) \cdot \text{poly}(|D|) \)
3. \( (D, q) \in L \) if and only if \( (c(D), q) \in L' \)
4. \( \kappa'(c(D)) \leq g(\kappa(D)) \)
Why this definition?

Why not just be more generous on the compilation size, and stick to poly-time query answering?

**Answer:** poly-time and fpt-time query answering turn out to coincide when allowing fpt-size compilations.
More powerful parameterizations

For fixed-parameter tractability: the parameterization has access to the entire input (and is assumed to be tractably computable).

\[ K \{ D \quad q? \} \rightarrow \text{poly-time} \quad N \]

In parameterized compilation: the parameterization has access only to the offline part of the input. Lifting the time restrictions for computing the parameter does not trivialize the problem, and makes sense in the setting of compilation.

\[ K \rightarrow \text{no time bounds} \quad N \]

As a result: we can allow more powerful parameters.
Clause Entailment (CE)

As an example, we consider the problem of clause entailment, which is a core problem in knowledge compilation.

Offline instance: a CNF formula $\varphi$

Online instance: a clause $\delta$

Question: $\varphi \models \delta$?

Theorem (Selman & Kautz, 1996)

CE has no poly-size compilation, unless the PH collapses.

(See also Cadoli, Donini, Liberatore & Schaerf, 2002.)
Parameters for CE

\[
\begin{align*}
\text{itw} & \quad \text{incidence treewidth of } \varphi \\
\text{itweup} & \quad \text{itw after propagating entailed unit clauses} \\
\text{itwsf} & \quad \text{minimum itw over all equivalent “sub-CNFs”} \\
& \quad \text{(sub-CNFs are obtained by deleting clauses and/or literals)} \\
\text{itweq} & \quad \text{minimum itw over all equivalent CNF formulas}
\end{align*}
\]

Dominance relation between these parameters and computational cost of computing them:

\[
\begin{align*}
\text{itw} <_{\text{dom}} \text{itweup} <_{\text{dom}} \text{itwsf} <_{\text{dom}} \text{itweq}
\end{align*}
\]

\[
\text{easy} \quad \text{hard} \quad \text{(but computable)}
\]
Parameterized compilation for CE

These parameters lead to different complexity and compilability behavior:

<table>
<thead>
<tr>
<th></th>
<th>poly-size compilable</th>
<th>fpt-time computable</th>
<th>fpt-size compilable</th>
</tr>
</thead>
<tbody>
<tr>
<td>itw</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>itweup</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>itwsf</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>itweq</td>
<td>NO</td>
<td>NO</td>
<td>NO?</td>
</tr>
</tbody>
</table>

We can move to more powerful parameters (whose values are smaller), in order to find the boundary of fpt-size compilability.
Parameter values in practice

More powerful parameters $\rightarrow$ smaller values in practice?

(Important question that needs further research.)

There are instances where \textit{itweup} is smaller than \textit{itw}:

<table>
<thead>
<tr>
<th>File</th>
<th>#vars</th>
<th>#clauses</th>
<th>itw</th>
<th>itweup</th>
</tr>
</thead>
<tbody>
<tr>
<td>3blocks</td>
<td>283</td>
<td>9690</td>
<td>35</td>
<td>22</td>
</tr>
<tr>
<td>4blocksb</td>
<td>410</td>
<td>24758</td>
<td>58</td>
<td>7</td>
</tr>
<tr>
<td>AProVE09-08</td>
<td>8564</td>
<td>28927</td>
<td>85</td>
<td>12</td>
</tr>
<tr>
<td>AProVE09-13</td>
<td>7606</td>
<td>26317</td>
<td>44</td>
<td>15</td>
</tr>
<tr>
<td>medium</td>
<td>116</td>
<td>953</td>
<td>52</td>
<td>7</td>
</tr>
<tr>
<td>satellite2_v01i.shuffled-4055</td>
<td>853</td>
<td>27249</td>
<td>191</td>
<td>31</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ev</td>
<td># of essential variables</td>
<td>fpt-size compilable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sbup</td>
<td>strong backdoor size to UP</td>
<td>fpt-size compilable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sbpl</td>
<td>strong backdoor size to PL</td>
<td>not fpt-size compilable, unless PH collapses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wgt</td>
<td>assignment weight</td>
<td>not fpt-size compilable, unless $W[1] \subseteq \text{FPT}/\text{fpt}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cls</td>
<td>size of queries (clauses)</td>
<td>not fpt-size compilable, unless nu-few-NP $\subseteq \text{FPT}/\text{fpt}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We considered fpt-size compilation with the aim of relativizing negative incompilability results.

As an example, we looked at parameterized variants of the clause entailment problem.

This approach opens the possibility for new parameters, and new positive compilability results.

This approach also introduces new theoretical questions.
References

