Parameterized Compilability Revisited

Combining parameterized complexity and knowledge compilation

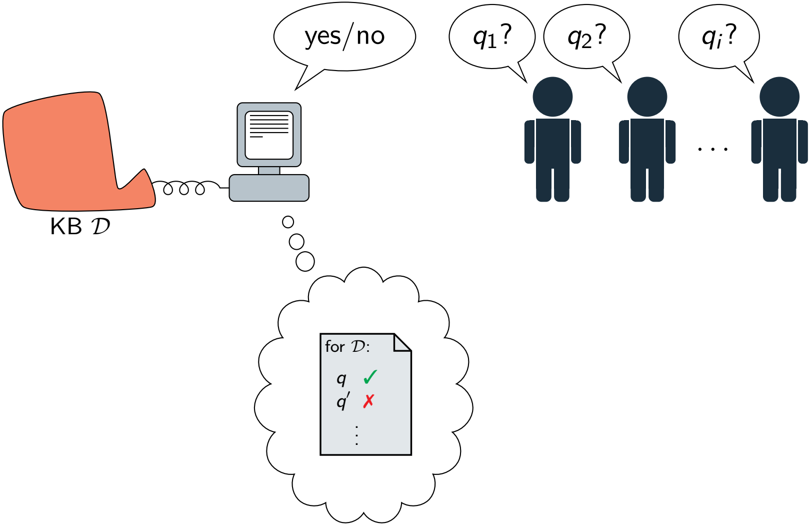
Ronald de Haan

(joint work with Simone Bova, Neha Lodha, and Stefan Szeider)

TU WIEN

ALGORITHMS AND
COMPLEXITY GROUP

Setting



Setting

Formally

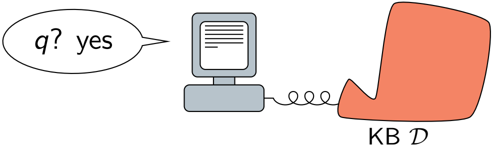
Compilation problems are problems of pairs:

$$L \subseteq \overset{\text{offline}}{\Sigma^*} \times \overset{\text{online}}{\Sigma^*}$$

Offline part: fixed knowledge base
Online part: differing queries

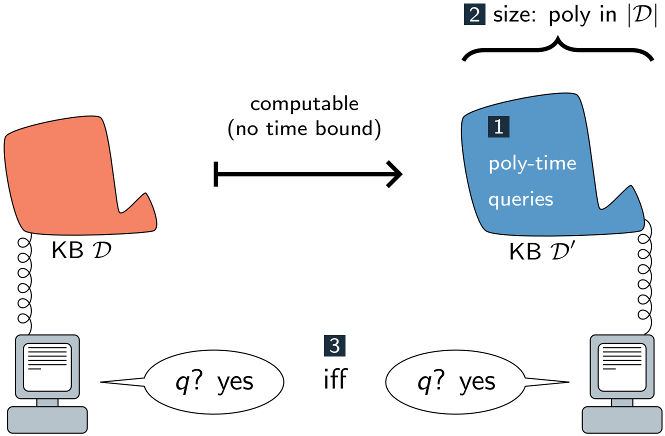
Graphically:

$(\mathcal{D}, q) \in L$ iff



Knowledge Compilation

In a picture



Knowledge Compilation

Formally

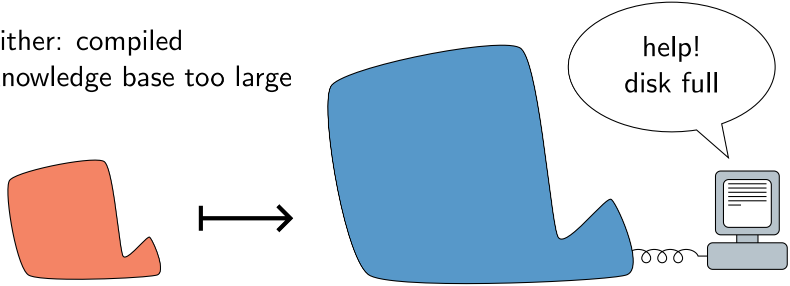
$L \subseteq \Sigma^* \times \Sigma^*$ is *(poly-size) compilable* if there exists a computable function $c : \Sigma^* \to \Sigma^*$ and a problem $L' \subseteq \Sigma^* \times \Sigma^*$ such that:

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

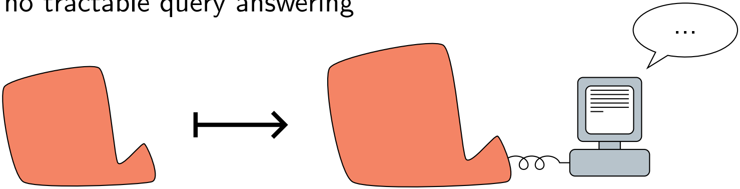
Negative compilation results

In a picture

either: compiled
knowledge base too large



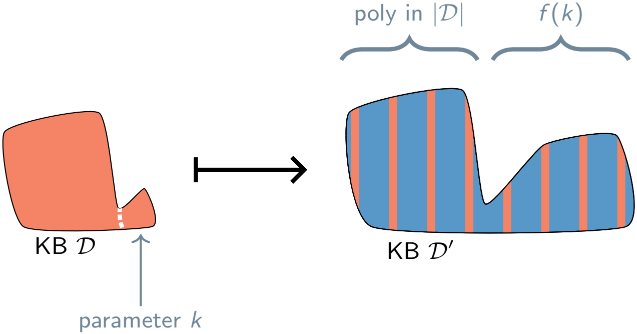
or: no tractable query answering



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:

$$L \subseteq \overset{\text{offline}}{\Sigma^*} \times \overset{\text{online}}{\Sigma^*}$$

Parameterization:

$$\kappa : \overset{\text{offline instance}}{\Sigma^*} \to \overset{\text{parameter value}}{\mathbb{N}}$$

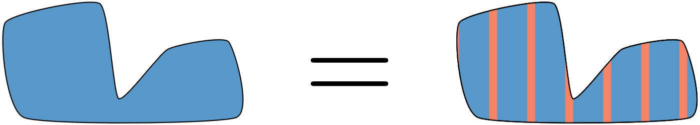
L is *fpt-size compilable* if there exists a computable function $c : \Sigma^* \to \Sigma^*$, computable functions $f, g : \mathbb{N} \to \mathbb{N}$, and a problem $L' \subseteq \Sigma^* \times \Sigma^*$ with a parameterization $\kappa' : \Sigma^* \to \mathbb{N}$ such that:

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

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.

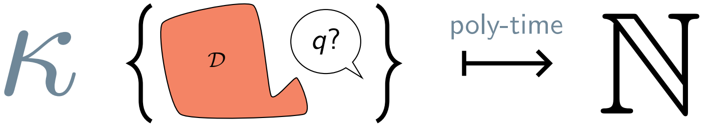


fpt-size compilation
poly-time online

fpt-size compilation
fpt-time online

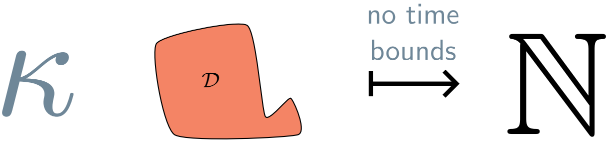
More powerful parameterizations

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



κ $\mathcal{D}$ q? poly-time $\mathbb{N}$

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.



κ $\mathcal{D}$ no time bounds $\mathbb{N}$

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 φ
Online instance:	a clause δ
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

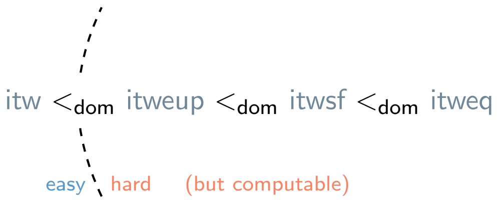
itw incidence treewidth of φ

itweup itw after propagating entailed unit clauses

itwsf minimum itw over all equivalent “sub-CNFs”
(sub-CNFs are obtained by deleting clauses and/or literals)

itweq minimum itw over all equivalent CNF formulas

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



itw $<_{\text{dom}}$ itweup $<_{\text{dom}}$ itwsf $<_{\text{dom}}$ itweq

easy hard (but computable)

Parameterized compilation for CE

These parameters lead to different complexity and compilability behavior:

	poly-size compilable	fpt-time computable	fpt-size compilable
itw	NO	YES	YES
itweup	NO	NO	YES
itwsf	NO	NO	YES
itweq	NO	NO	NO?

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 ⟶ smaller values in practice?

(Important question that needs further research.)

There are instances where itweup is smaller than itw:

Preliminary investigation (approximations)

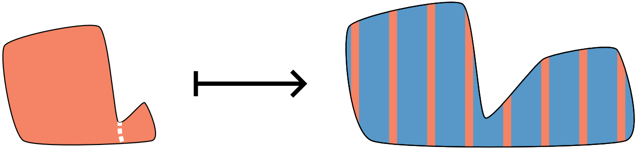
File	*#vars*	*#clauses*	itw	itweup
3blocks	283	9690	35	22
4blocksb	410	24758	58	7
AProVE09-08	8564	28927	85	12
AProVE09-13	7606	26317	44	15
medium	116	953	52	7
satellite2_v01i.shuffled-4055	853	27249	191	31

Other parameters for CE

ev	# of essential variables	fpt-size compilable
sbup	strong backdoor size to UP	fpt-size compilable
sbpl	strong backdoor size to PL	not fpt-size compilable, unless PH collapses
wgt	assignment weight	not fpt-size compilable, unless W[1] $\subseteq$ FPT/fpt
cls	size of queries (clauses)	not fpt-size compilable, unless nu-few-NP $\subseteq$ FPT/fpt

Conclusion

- 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

- M. Cadoli, F.M. Donini, P. Liberatore, and M. Schaerf.
  Preprocessing of Intractable Problems.
  Information and Computation, 176(2):89–120, 2002.

- H. Chen.
  Parameterized Compilability.
  Proceedings of IJCAI, 2005.

- B. Selman and H.A. Kautz.
  Knowledge Compilation and Theory Approximation.
  Journal of the ACM, 43:193–224, 1996.