



# Resolution-based approximate reasoning for OWL DL

*Pascal Hitzler*



*Denny Vrandecic*

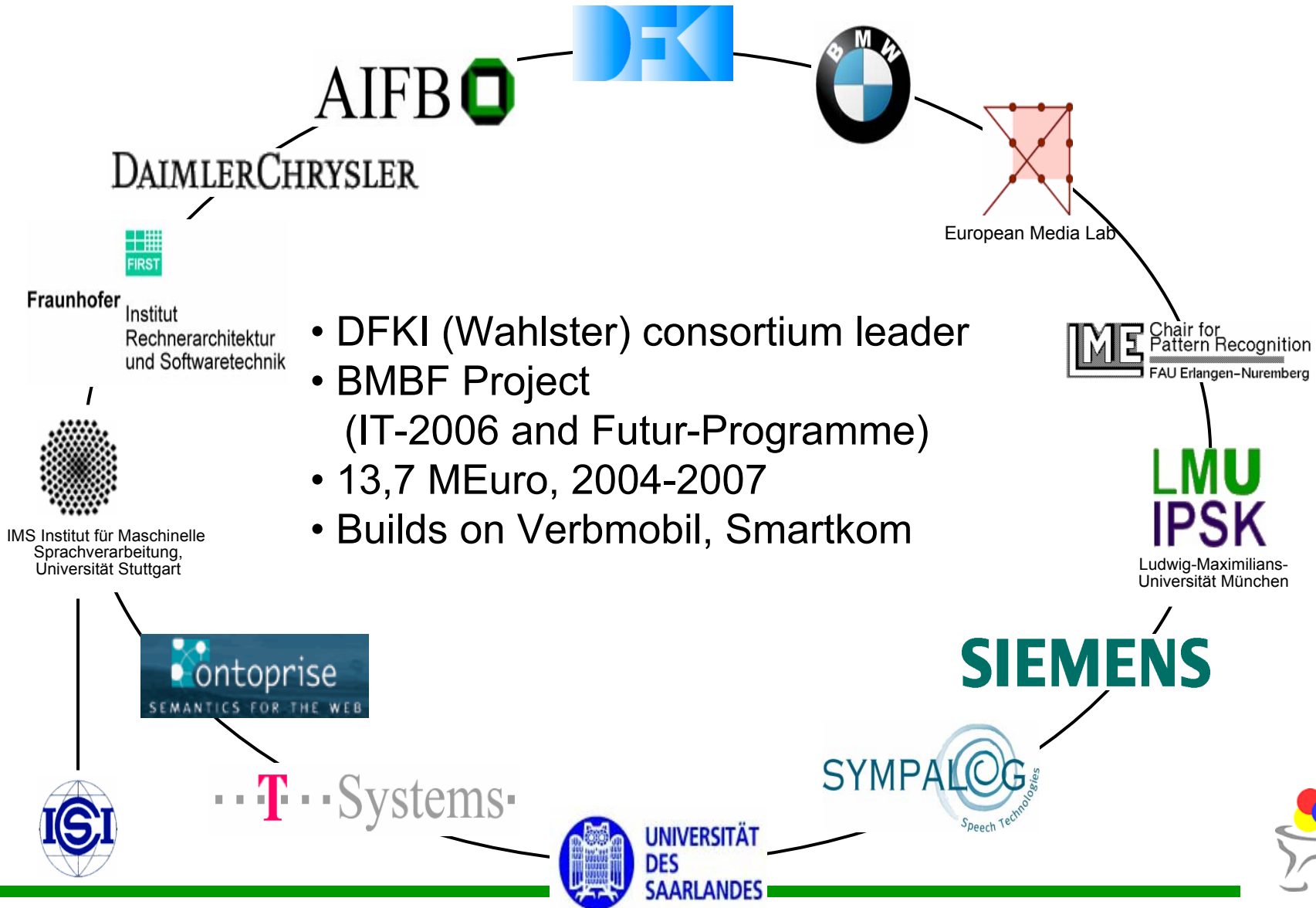


**AIFB, University of Karlsruhe, Germany**

# Problem Description

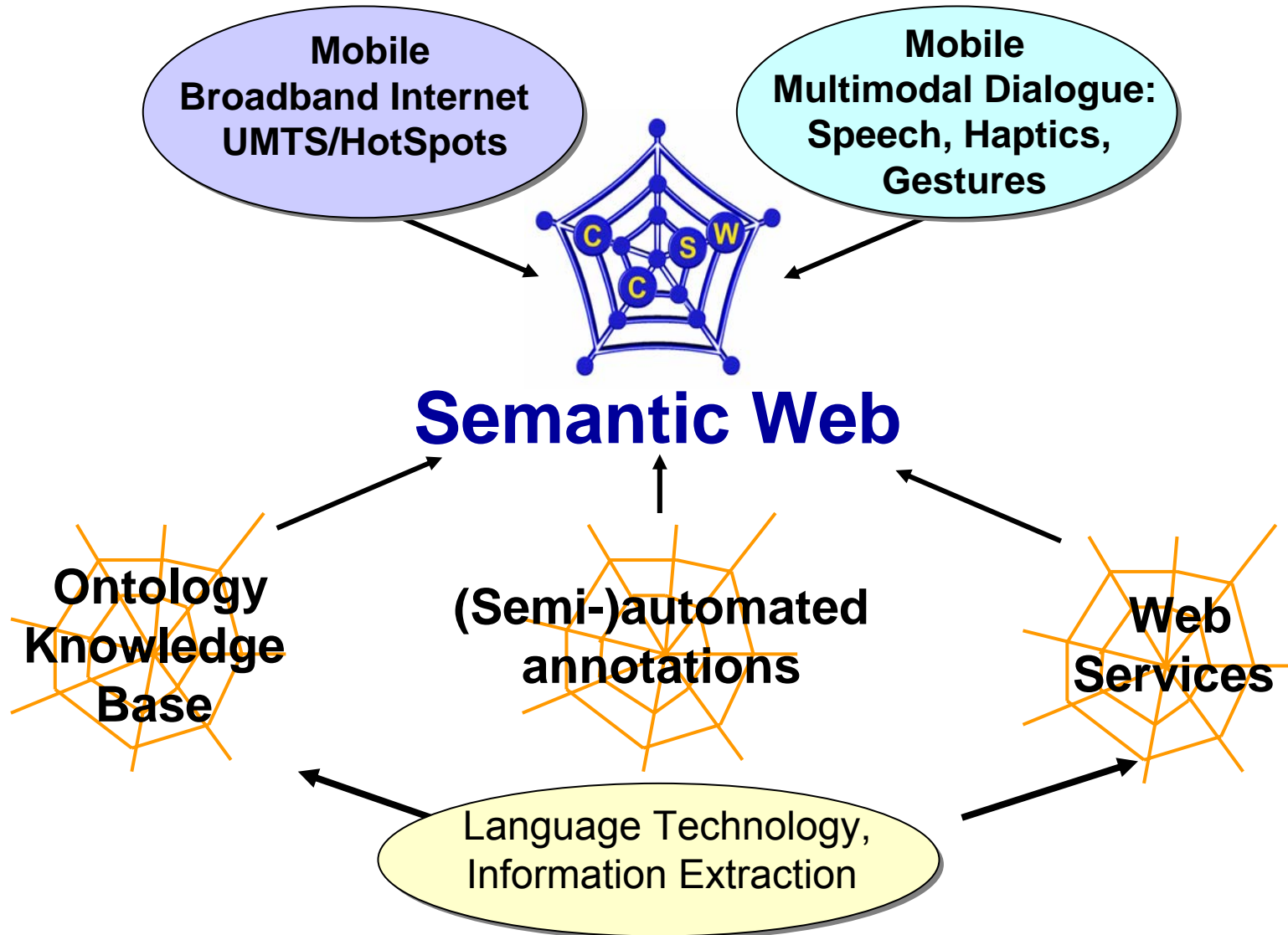
- Reasoning with OWL DL is **hard**.
- For certain Semantic Web applications quick responses are more important than absolute accuracy of answering.  
e.g. *SmartWeb* scenario.

# SmartWeb: Mobile querying of the Semantic Web

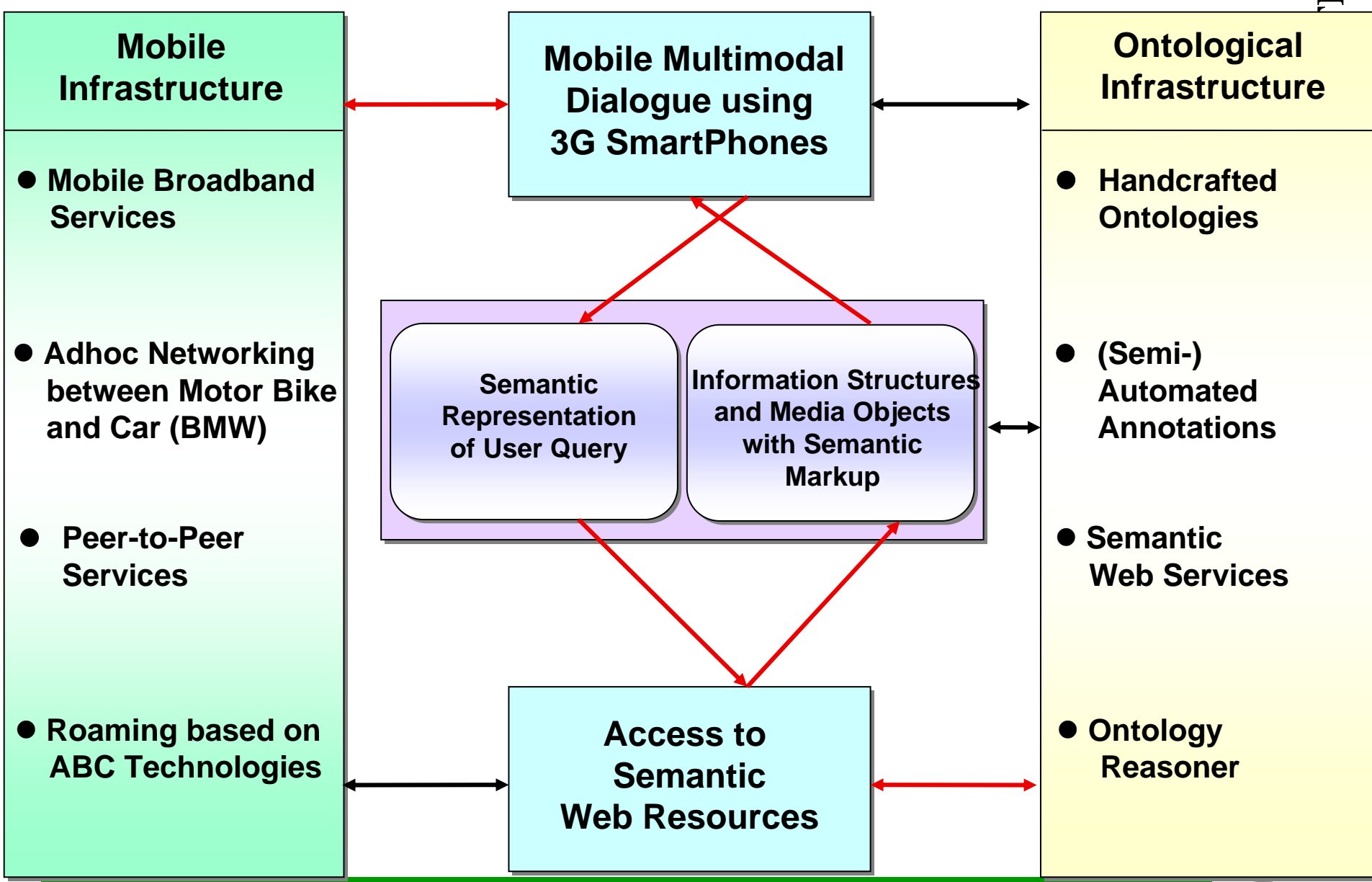


- DFKI (Wahlster) consortium leader
- BMBF Project (IT-2006 and Futur-Programme)
- 13,7 MEuro, 2004-2007
- Builds on Verbmobil, Smartkom





# The Basic Architecture of SmartWeb



# Problem Description

- Reasoning with OWL DL is **hard**. (Expressivity vs. scalability)
- For certain Semantic Web applications quick responses are more important than absolute accuracy of answering.  
e.g. *SmartWeb* scenario.
  - Answering of human queries in an open domain.
- We **trade soundness for time**, using **approximate reasoning**.

# Approximate Reasoning

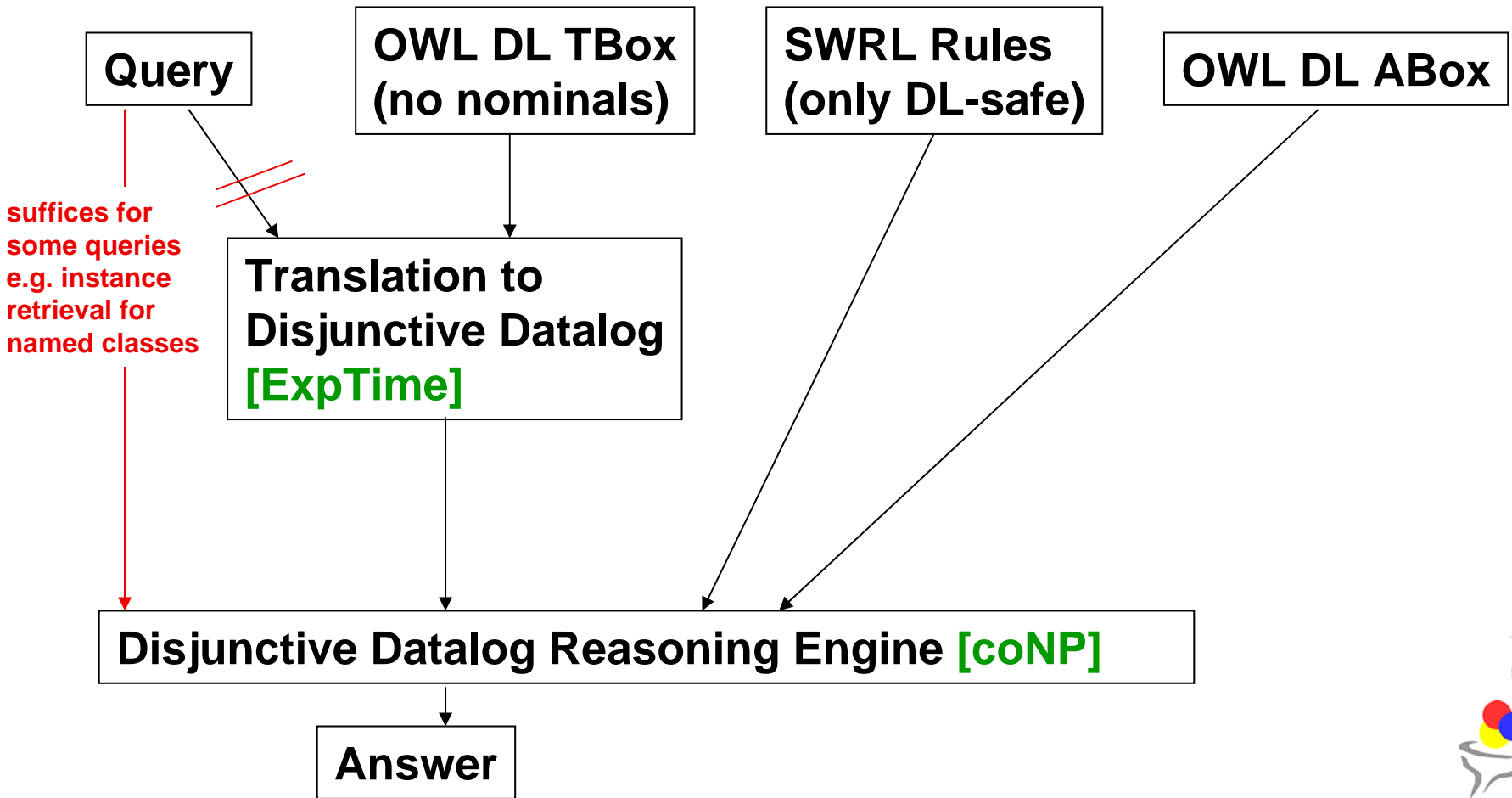
- do not confuse with *fuzzy* or *probabilistic* reasoning!
- speed up obtained by
  - modifying the underlying inference relation
  - in a semantically controlled and well-understood way.
- e.g. by decreasing the complexity class of a reasoning task
- e.g. by utilizing intimate knowledge of the bottlenecks in a reasoning algorithm.

# Our Concrete Approach

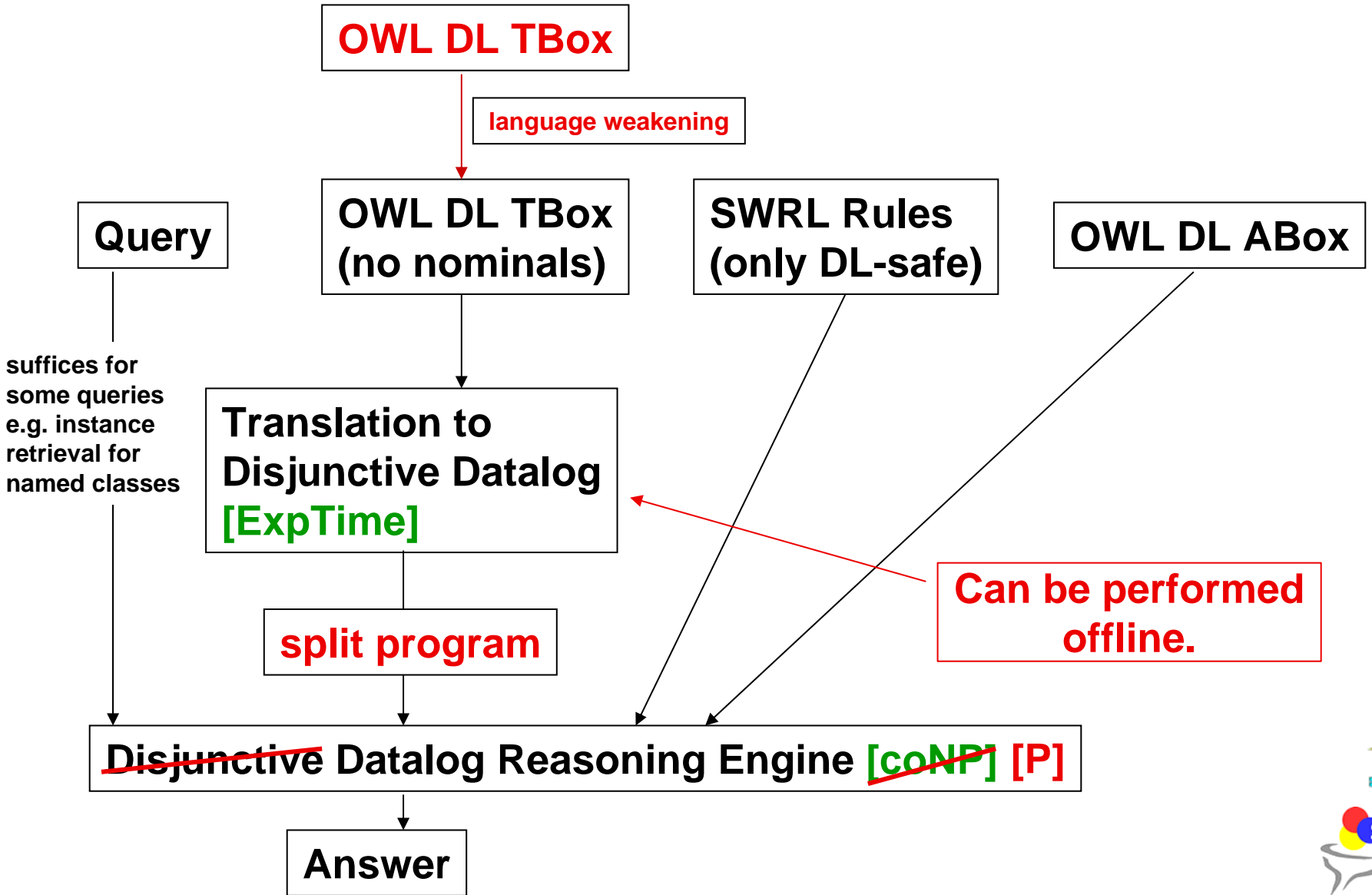
We facilitate recent results and algorithms due to  
*Hustadt, Motik, Sattler, Studer 2003/2004/2005*

on **casting OWL-DL into disjunctive Datalog**.  
(currently being implemented in **KAON2**  
see <http://kaon2.semanticweb.org>)

# KAON2 Reasoner Core Architecture



# KAON2 Reasoner Core Architecture



## Screech simple example

serbian  $\sqsubset$  croatian  $\sqsubseteq$  european

eucitizen  $\sqsubseteq$  european

german  $\sqsubset$  french  $\sqsubset$  beneluxian  $\sqsubseteq$  eucitizen

**beneluxian  $\equiv$  luxembourgian  $\sqsubset$  dutch  $\sqsubset$  belgian**

serbian(ljiljana).

serbian(nenad).

german(pascal).

french(julien).

croatian(boris).

german(markus).

german(stephan).

croatian(denny).

indian(sudhir).

**belgian(saartje).**

german(rudi).

german(york).

## Screech simple example

beneluxian  $\equiv$  luxembourgian  $\sqcup$  dutch  $\sqcup$  belgian

translates into the following four clauses:

~~luxembourgian(x)  $\vee$  dutch(x)  $\vee$  belgian(x)  $\leftarrow$  beneluxian(x)~~

beneluxian(x)  $\leftarrow$  luxembourgian(x)

beneluxian(x)  $\leftarrow$  dutch(x)

beneluxian(x)  $\leftarrow$  belgian(x)

### split of first clause:

luxembourgian(x)  $\leftarrow$  beneluxian(x)

dutch(x)  $\leftarrow$  beneluxian(x)

belgian(x)  $\leftarrow$  beneluxian(x)

$\vdash$  luxembourgian(saartje)

$\vdash$  dutch(saartje)

$\vdash$  belgian(saartje)

# Screech reasoning

- data complexity is **P**
- complete
- but unsound
- inference can be described in terms of standard notions from *non-monotonic reasoning*

## Semantic description

Inference boils down to  
*brave reasoning with well-supported models.*

**Variant of standard notion for non-disjunctive programs. Shown by Fages (1994) to be equivalent to stable models.**

**Reiter's Default Logic**

**Answer Set Programming**

## Screech Performance (not optimized yet)

Galen ontology

673 axioms, 175 classes

randomly populated with 500 individuals

267 disjunctions in 133 rules eliminated

Complete run:

- queried for the extensions of all 175 Galen classes
- resulting in 5809 classifications (Screech)
  - 5353 (i.e. **92.2%**) **correct**
- For 138 out of 175 classes: computed extension correct
- Average **time saved: 39.0%**

# Screech Performance example run

Time (DD)	Time (SPLIT)	Instances	Class Name
11036 ms	6489 ms	154/154	Biological_object
11026 ms	5959 ms	9/9	Specified_set
11006 ms	6219 ms	9/13	Multiple
11015 ms	5898 ms	16/16	Probe_structural_part_of_heart
11036 ms	7711 ms	4/4	Human_red_blood_cell_mature
11055 ms	5949 ms	24/58	Biological_object_that...

**Table 2.** Performance comparison for instance retrieval using disjunctive datalog (DD) vs. the corresponding split program (SPLIT), on the KAON2 datalog engine. *Instances* indicates the number of instances retrieved using DD versus SPLIT, e.g. class *Multiple* contained 9 individuals, while the split program allowed to retrieve 13 (i.e. the 9 correct individuals plus 4 incorrect ones). The full name of the class in the last row is `Biological_object_that_has_left_right_symmetry`.

# Conclusions

- Screech reasoner based on KAON2
- complete but unsound
- 40% speed-up with only 8% wrong answers
- future work
  - optimise further
  - tackle other parts of the KAON2 algorithms
  - combine with other approximate reasoning and optimization techniques

**Thanks!**

<http://logic.aifb.uni-karlsruhe.de/screetch>