

Towards Formally Verified Rule Language Compilers

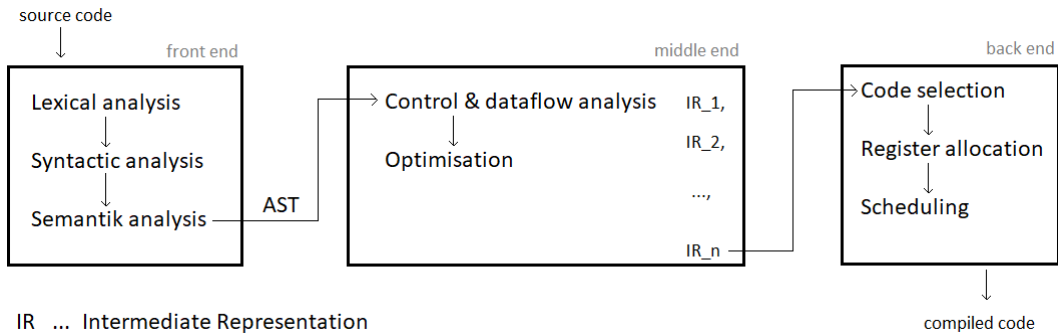
Antonio Hentschke

September 12, 2024

What is the problem

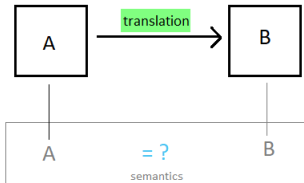
- ▶ optimisations need formal verification
 - ▶ search for a fixpoint may not terminate, e.g. Skolem chase
 - ▶ some algorithms optimize based on edge cases
- ▶ correctness guarantees should not compromise efficiency

Short trip to the world of compilers



Formally verified compilers

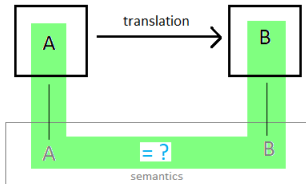
- ▶ 2 approaches
 - ▶ internally verified - directly from scratch
 - ▶ external verification aka. translation validation (more flexible)



- ▶ formalize representations; formalize transitions; show preservation of semantics

Formally verified compilers

- ▶ 2 approaches
 - ▶ internally verified - directly from scratch
 - ▶ external verification aka. translation validation (more flexible)



- ▶ formalize representations; formalize transitions; show preservation of semantics

Semantic preservation

- ▶ formalize representations; formalize transitions; show preservation of semantics
- ▶ theorem provers like Coq or Lean can be used

"If the compiler produces compiled code C from source code S , without reporting compile-time errors, then every observable behavior (B) of C is either identical to an allowed behavior of S , or improves over such an allowed behavior of S by replacing undefined behaviors with more defined behaviors."
[2]

Semantic preservation

- ▶ formalize representations; formalize transitions; show preservation of semantics
- ▶ theorem provers like Coq or Lean can be used

$$S \text{ safe} \Rightarrow \forall B, S \Downarrow B \iff C \Downarrow B$$

Semantic preservation

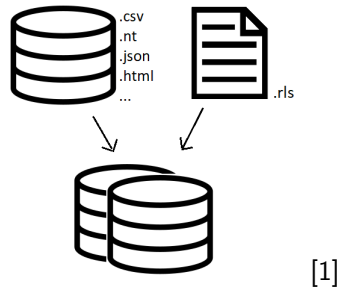
- ▶ formalize representations; formalize transitions; show preservation of semantics
- ▶ theorem provers like Coq or Lean can be used

$$\forall S, C, \text{Comp}(S) = \text{OK}(C) \Rightarrow S \approx C$$

Rule reasoning systems

1. logic programming systems
2. KG and deductive database engines
3. specialised data-analytics systems
4. data management frameworks

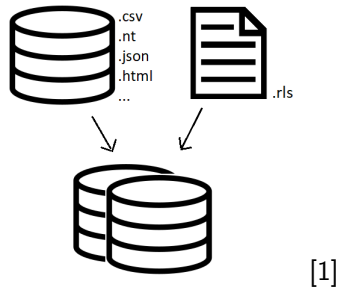
according to *Nemo: Your Friendly and Versatile Rule Reasoning Toolkit* [0]



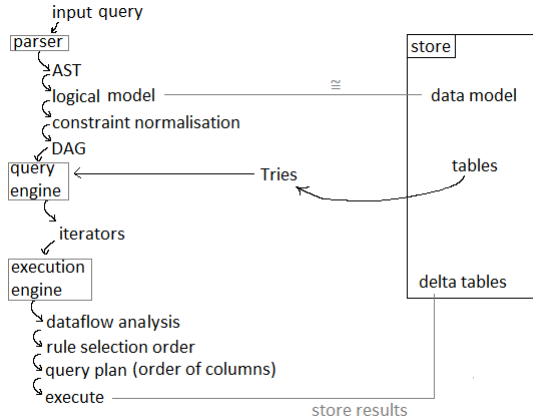
Rule reasoning systems

1. logic programming systems
2. **KG and deductive database engines**
3. **specialised data-analytics systems**
4. data management frameworks

according to *Nemo: Your Friendly and Versatile Rule Reasoning Toolkit* [0]



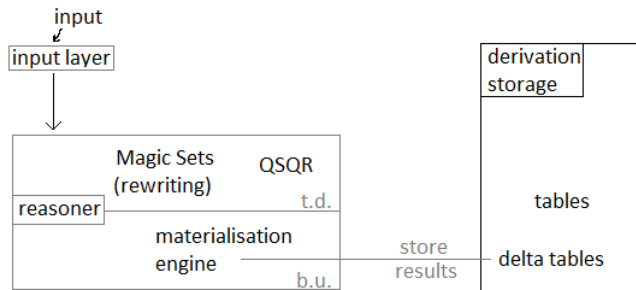
Nemo



Nemo

```
a(x) :- b("foo", x).  
a(x) :- b(var, x), var = "foo".
```

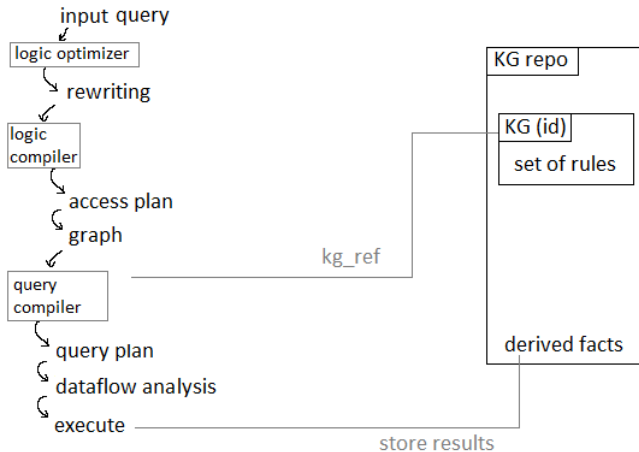
VLog



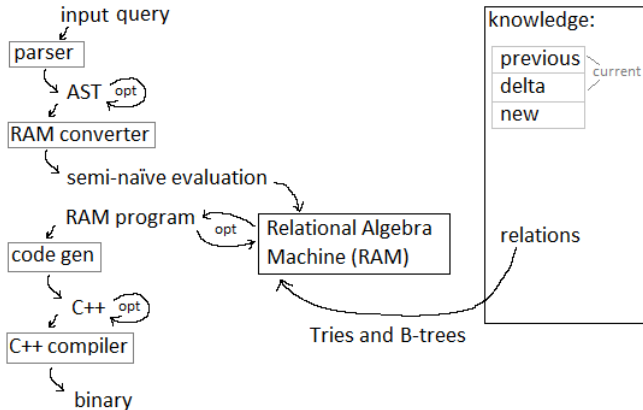
t.d. = top-down (query-driven reasoning)

b.u. = bottom-up (materialisation)

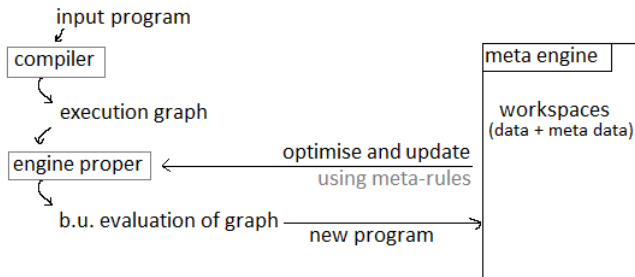
Vadalog



Soufflé



LogicBlox



Structural similarities

- ▶ top-down evaluation (query-driven reasoning) and/or bottom-up (materialisation)
- ▶ distinction between IDB and EDB (e.g. delta tables)
- ▶ some initial parsing into an AST
- ▶ rewriting (e.g. constraint normalisation)
- ▶ a graph representation of the database and some structure to access it
- ▶ query engine or materialisation engine
- ▶ usage of relational algebra to process tables

Concrete plans for verification

- ▶ semi-naive evaluation (since it's optimized)
- ▶ LFTJ (since it's optimized)
- ▶ 1-parallel restricted chase
- ▶ (skolem chase)

Time for questions :)

References

- [0] **classification of rule reasoning systems** from *Nemo: Your Friendly and Versatile Rule Reasoning Toolkit*, Alex Ivliev and Lukas Gerlach and Simon Meusel and Jakob Steinberg and Markus Krötzsch, <https://iccl.inf.tu-dresden.de/w/images/f/fb/KR-2024-CR.pdf>
- [1] **database icon** <https://linearicons.com/>, created by <https://perxis.com/> - <https://cdn.linearicons.com/free/1.0.0/Linearicons-Free-v1.0.0.zip>, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=77236210>;
- note icon** SimpleIcon <http://www.simpleicon.com/>, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons; changes were made to the graphics
- [2] **definition of semantic preservation** taken from *CompCert - A Formally Verified Optimizing Compiler*, Leroy, Xavier and Blazy, Sandrine and Kästner, Daniel and Schommer, Bernhard and Pister, Markus and Ferdinand, Christian, https://inria.hal.science/hal-01238879/file/erts2016_compcert.pdf