KAON2 Algorithm

Non-tableau reasoning with description logics

C. Braun

Department of Computer Science
University of Dresden

20th January 2006

Outline

Outline

Semantic Web

Ambition:

Represent Web metadata in a uniformed way
$\Rightarrow$ Answer complex queries much more efficiently

The current situation:

- RDFS and OWL: Web-ontology languages recently standardized
- Rely on the DL formalism:
  - TBox: relations between descriptions
  - ABox: individuals in the knowledge base
- $\nearrow$ expressivity $\Rightarrow$ tractability $\searrow$!

Traditional systems

What about the existing DL reasoners ?

- query answering with tableau-based algorithms
- optimizations focused on TBoxes

ABox: tableau-based reasoning perform poorly !

- every element in the ABox is treated individually
  $\Rightarrow$ no consideration for sets with similarities
- every element in the ABox must be taken into account
  $\Rightarrow$ no consideration for "irrelevant" ones

Relevant issue: currently, ABoxes are getting larger...

Outline

Idea : borrow some good recipes...

DL systems can be related to the field of DB management :

- TBox $\leftrightarrow$ DB Schema
- ABox $\leftrightarrow$ DB Instance

Advantages : sophisticated techniques available

- Join-order optimization : consider sets with similarities
- Magic sets tranformation : consider only relevant elements

Consequence :

Drawbacks of tableau-based algorithms are avoided !

Idea: from DL to DB

Input: A $\mathcal{SHIQ}$ knowledge base $SHIQ(KB)$

- $\mathcal{SHOIN}$ corresponds to OWL-DL
- $\mathcal{SHIQ}$ fast as expressive as $\mathcal{SHOIN}$
- $\Rightarrow$ Good choice for ontology representation !

Output: A disjunctive datalog program $DD(KB)$

- main language for logic databases
  $\Rightarrow$ optimizations available
- if the same ground facts are derived from $SHIQ(KB)$ and $DD(KB)$...
  ... then query $DD(KB)$ instead of $SHIQ(KB)$!

$\Rightarrow$ Answering query over large ABoxes is much more efficient

Underlying strategy $SHIQ(KB) \rightarrow DD(KB)$

Use only the best side of each...

1. First, TBox and RBox reasoning only:
   Starting with $SHIQ(KB)$, compute all non-ground consequences.
   $\Rightarrow$ Enough for TBox reasoning tasks (computing the subsumption hierarchy . . .)
2. Then, translation:
   Transform the clauses into a disjunctive datalog program.
3. Now, ABox reasoning:
   Use $DD(KB)$ for assertional reasoning.
   $\Rightarrow$ Query answering (instance checking . . .) is much more efficient !

Why is it (so) complicated ?

Requirement

The decision procedure must be efficiently “simulated” from one formalism to the other:

$\Rightarrow$ Use a resolution decision procedure, but “highly optimized”

$\Rightarrow$ Basic Superposition

Some more tasks...

Additional steps to ensure a correct and terminating transformation.

Overview of the whole algorithm

5 steps

1. Translate $SHIQ(KB)$ into $ALCHIQ^{-}(KB)$
2. Transform $ALCHIQ^{-}(KB)$ in clausal normal form $\chi(KB)$
3. Saturate $\chi(KB)$ by Basic Superposition
4. Remove function symbols to get $FF(KB)$
5. Transform $FF(KB)$ into a disjunctive datalog $DD(KB)$

Outline

Some words about $\mathcal{SHIQ}$

The $\mathcal{SHIQ}$ knowledge base $SHIQ(KB)$:

SHIQ(KB) $= KB_R \cup KB_T \cup KB_A$

- RBox KB_R:
  - Transitivity: $Trans(R)$
  - Role inclusions: $R \sqsubseteq S$
- TBox KB_T:
  - Concept inclusion: $C \sqsubseteq D$
  - Concept equivalence: $C \equiv D$
- ABox KB_A:
  - Membership: $C(a), \neg R(a, b) \ldots$
  - (In)equality: $a \approx b, a \not\approx b$

Input: $\mathcal{SHIQ}$

A very expressive DL:

Many constructors supported:

- boolean operations : $C \sqcap D$, $C \sqcup D$, $\neg C$
- quantifiers : $\exists R.C$, $\forall R.C$
- number restrictions : $\leq nR.C$, $\geq nR.C$
- inverse roles : $Inv(R)$

Sometimes a bit too expressive $\Rightarrow$ use fragments:

- $\mathcal{ALCHIQ}$: $\mathcal{SHIQ}$ without transitivity axioms in RBoxes.
- $\mathcal{SHIQ}^{-}$ ($\mathcal{ALCHIQ}^{-}$): $\mathcal{SHIQ}$ ($\mathcal{ALCHIQ}$) with only simple roles (roles without subroles) in number restrictions.

First task: $\mathcal{SHIQ} \rightarrow \mathcal{ALCHIQ}$

Why is $\mathcal{SHIQ}$ too expressive ?

Transitive roles :may threaten the termination of the algorithm
$\Rightarrow$ use $\mathcal{ALCHIQ}$ instead

Remove transitive roles:

1. Compute the concept closure $clos(KB)$:
   $\sim$ smallest set of "relevant concepts"
2. Adapt the RBox : remove all transitivity axioms $Trans(S)$
3. Adapt the TBox : add for the concepts in $clos(KB)$ new inclusion axioms encoding transitivity
   $\Rightarrow$ if $S \sqsubseteq R \in KB_R$, then add $\forall R.C \sqsubseteq \forall S.\forall S.C$
4. Keep the initial ABox

Output :

An equisatisfiable $\mathcal{ALCHIQ}$ knowledge base

How hard ?

Polynomial time in the size of $SHIQ(KB)$

Second task: $\mathcal{ALCHIQ} \rightarrow \mathcal{ALCHIQ}^{-}$

Why is $\mathcal{ALCHIQ}$ still too expressive ?

- subroles in number restrictions :may also lead to infinite derivations
- use $\mathcal{ALCHIQ}^{-}$ instead, but that comes only later (Step 3)...

Outline

Clausification step (1)

Why do we need FOL clauses in normal form ?

Input required by the clausal calculus Basic Superposition (Step 3)
$\Rightarrow ALCHIQ^{-}(KB) \rightsquigarrow \chi(KB)$

First idea: direct clausification

1. directly transform $ALCHIQ^{-}(KB)$ into a FOL formula $\Pi(KB)$
   $\pi_y(C \sqcup D, X) = \pi_y(C, X) \vee \pi_y(D, X)$,
   $\pi(R \sqsubseteq S) = \forall x, y : R(x, y) \rightarrow S(x, y)$,...
2. skolemize $\Pi(KB)$
3. use well-known equivalences to put $\Pi(KB)$ in CNF

Clausification step (2)

2 drawbacks :

- exponential blow-up of the size of the KB
- lost of the nice syntactic structure

Second (better) idea: Structural transformation (*renaming*)

1. first put the KB in definitorial normal form:
   $\exists P.A \sqsubseteq \forall Q.B$: find a simpler expression...
   $\sim \{\exists P.A \sqsubseteq N_1, N_2 \sqsubseteq \forall Q.B, N_1 \sqsubseteq N_2\}$
2. apply (extended) direct clausification to this set

A new additional difficulty

Output :

- equisatisfiable set of clauses $\chi(KB)$
- achieved in polynomial time
- all clauses classifiable in (only) 8 different syntactic categories
  $\Rightarrow$ will be useful in Step 3

Side-effect: new function symbols

The skolemization may introduce new function symbols.
$\Rightarrow$ Query answering may not terminate

Clausification step (4)

Example :

$C \sqsubseteq \exists R.C$ translated to the rules:
$\{R(x, f(x)) \leftarrow C(x), C(f(x)) \leftarrow C(x)\}$
If $C(a) \in KB_A$, we derive $\{C(f(a)), C(f(f(a))), ...\} \rightarrow \infty$

Solution:

Before translating the clauses into $DD(KB)$:

- Render function symbols redundant $\Rightarrow$ Step 3
- Remove all function symbols $\Rightarrow$ Step 4

Outline

Basic Superposition : how did we get there ?

1. Resolution (1965)

Proves theorems in FOL by refutation via inference rules
$\rightarrow$yes but... search-space may be huge !

2. Ordered Resolution

Reduces the search space for resolution with ordering constraints
$\rightarrow$yes but... no support for logics with equality !

3. Paramodulation

Adds explicit rules to resolution calculus for equality reasoning
$\rightarrow$yes but... too many (unnecessary) inferences !

4. Superposition

Refines ordering restrictions to deal with equality
$\rightarrow$yes but... still not efficient !

5. Basic Superposition

Avoids the need to apply superposition in terms introduced by previous unification steps.
$\rightarrow$much better ! and still sound and complete...

Parameters of BS:

BS needs 2 parameters:

- ordering $\succ$ on terms and literals
- selection function f

Some room left for a clever choice:

- ordering:
  $S = \{C(a), \neg C(x) \vee C(f(x))\}$
  If $\neg C(x) \succ C(f(x))$ we derive $\{C(f(a)), C(f(f(a))), ...\}$.
  $\Rightarrow$ choose $C(f(x)) \succ \neg C(x)$
- selection function:
  f selects only negative literals.

Helpful to ensure termination.

Basic Superposition: 5 inference rules (1)

1. Positive Superposition

$$\frac{(C \vee s \approx t).\rho \qquad (D \vee w \approx v).\rho}{(C \vee D \vee w[t]_p \approx v).\theta}$$

where $\sigma = MGU(s\rho, w\rho\,|_p)$ and $\theta = \rho\sigma$ $(+\ CONSTRAINTS)$
Select some equal terms to be replaced in other terms and remove the corresponding equality axiom after unification.

Example: Positive superposition

$$(C(x) \vee s \approx t).\{s \mapsto g(x), t \mapsto h(x)\} \qquad (C(g(x)))\{x \mapsto f(y)\}$$

$$\equiv$$

$$\frac{C(x) \vee [g(x)] \approx [h(x)] \qquad C(g([f(y)]))}{C(f(y)) \vee C([h(f(y))])}$$

Basic Superposition: 5 inference rules (2)

2. Negative Superposition

$$\frac{(C \vee s \approx t).\rho \qquad (D \vee w \not\approx v).\rho}{(C \vee D \vee w[t]_p \not\approx v).\theta}$$

where $\sigma = MGU(s\rho, w\rho\,|_p)$ and $\theta = \rho\sigma\ (+\ CONSTRAINTS)$

Same idea with inequality axioms.

3. Reflexivity resolution

$$\frac{(C \vee s \not\approx t).\rho}{C.\theta}$$

where $\sigma = MGU(s\rho, t\rho)$ and $\theta = \rho\sigma\ (+\ CONSTRAINTS)$

Equality predicates are replaced by their consequences after unification of both equals.

Basic Superposition: 5 inference rules (3)

4. Equality factoring

$$\frac{(C \vee s \approx t \vee s' \approx t').\rho}{C \vee t \not\approx t' \vee s' \approx t').\theta}$$

where $\sigma = MGU(s\rho, s'\rho)$ and $\theta = \rho\sigma$ $(+ \ CONSTRAINTS)$
A set of equalities is simplified after the unification of terms involved in different equalities.

Basic Superposition: 5 inference rules (4)

5. Ordered hyperresolution

$$\frac{E_1 ... E_n \qquad N}{(C_1 \vee ... \vee C_n \vee D).\theta}$$

where: E_i are of the form $(C_i \vee A_i).\rho$, for $1 \leq i \leq n$, N is of the form $(D \vee \neg B_1 \vee ... \vee \neg B_n).\rho$, σ is the most general substitution such that $A_i\theta = B_i\theta$ for $1 \leq i \leq n$ and $\theta = \rho\sigma$,
$(+ \ CONSTRAINTS)$
An optimized version of classical resolution.

Example: Ordered Hyperresolution

$$\frac{(C(y) \vee \neg R(x,y)).\{\} \qquad (D(x) \vee R(x,f(x))).\{\}}{(C(y) \vee D(x)).\{y \mapsto f(x)\}}$$

Outcome of the saturation step (1)

By case-analysis

Recall: the input clauses had a well-defined syntactic structure $\Rightarrow$ a case-analysis reaveals all possible forms of an inferred clause C

Conclusion: only 2 possibilities

- C belongs again to a syntactic category
- C is redundant (i.e. removable)

Consequence

Termination is guaranteed !

Outcome of the saturation step (2)

How hard ?

Exponential time in the size of the input KB.

Be careful !

Does not work so well with $\mathcal{ALCHIQ}$ instead of $\mathcal{ALCHIQ}^{-}$.
$\Rightarrow$ needs an additional transformation, decomposition, during Basic Superposition.

Outline

Recall: function symbols are annoying

Problem

Function symbols in recursive rules threaten termination of query answering.

Idea

- $f(a)$ replaced by a_f, $f(x)$ replaced by x_f.
- A new predicate S_f is added to encode the relations.

At the same time, unsafe variables in a clause get bound.

Output

$FF(KB)$ function-free and equisatisfiable with $\chi(KB)$

Outline

Some words about Disjunctive Datalog Programs

LP language with some particularities:

- FOL signature Σ with equality $\approx$
- only constants (no function symbol with arity > 0)
- Finite set of rules of the form:
  $A_1 \vee ... \vee A_n \leftarrow B_1, ..., B_m$
  with $n, m \geq 0$, A_i and B_i are atoms over Σ.

Prerequisite: safe rules

All variables in the head appear also in the body
$\Rightarrow$ number of constants remain finite
$\Rightarrow$ all models are finite

Straightforward rewriting

Use a well-known logical equivalence :

$(A \leftarrow B) \equiv (\neg B \vee A)$

- move the positive literals in the head
- move the negative literals in the body

Is it really a disjunctive datalog program ? $\Rightarrow$ Recall Step 4

- Only constants: Yes, all function symbols removed.
- Safe rules: Yes, unsafe variables bound.

So finally...

Our goal is reached !

Summary

- A novel algorithm which reduces a $\mathcal{SHIQ}$ knowledge base to an equisatisfiable disjunctive datalog program, using Basic Superposition
- Efficiently implemented in KAON2, a promising ontology reasoner developed in Karlsruhe
- Many challenges remain: support for nominals, non-monotonicity . . .

For Further Information

Boris Motik
Reasoning in Description Logics using Resolution and Deductive Databases.
PhD, 2006.

http://kaon2.semanticweb.org

KAON2

Motivation

- New DL reasoner based on this algorithm
- Better performance on large ABoxes than existing systems
- Very promising for further optimizations

Main features

- Java 1.5
- API for ontology manipulation (OWL XML, OWL RDF)
- Storage of ABox assertions in RDBMS
- Support for SPARQL

Architecture

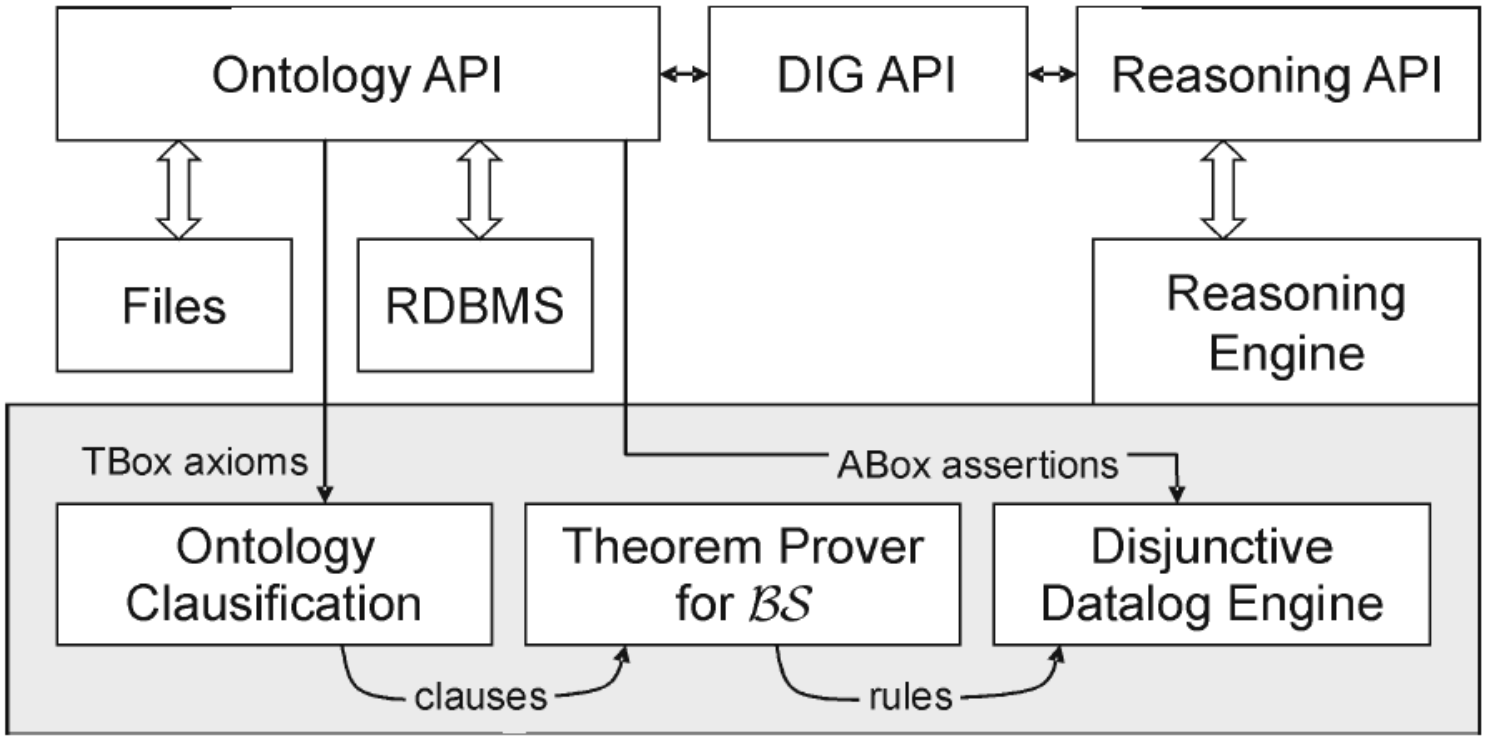

Figure: KAON2 Architecture

Theorem Prover: inference loop

Fairness

Closures split in 2 sets:

1. U: unprocessed closures
2. W: worked-off closures

$\Rightarrow$ all closures are eventually handled

Initially

W empty, all closures in U.

Theorem Prover: inference loop

Repeat:

1. **Non-deterministic choice** of a closure C in U.
2. **Redundancy elimination** ($C \rightarrow C'$) 3 possible outcomes:
   - $C' = \{\} \Rightarrow KB$ unsatisfiable !
   - C' redundant $\Rightarrow C'$ removed
   - Otherwise: go on
3. **Backward subsumption**: remove all D in W with $D \sqsubseteq C'$
4. **Inferences computation**: saturation $\Rightarrow$ non-tautological conclusions added to U

KAON2

Test it yourself !

http://kaon2.semanticweb.org/