Automated Deduction Techniques for Knowledge Representation Applications

Peter Baumgartner
Max-Planck-Institute for Informatics
The Big Picture

Knowledge Base

Ontologies
- OWL DL
  (Tambis, Wine, Galen)
- First-Order
  (SUMO/MILO, OpenCyc)
- FrameNet

Rules (SWRL)

Data (ABox)

Reasoning

Tasks
- TBox: (Un)satisfiability, Subsumption
- ABox: Instance, Retrieve
- General entailment tasks

Theorem provers for:
- Classical FO (ME: Darwin)
- FO with Default Negation
  (Hyper Tableaux: KRHyper)

Robust Reasoning Services?

- Issues: undecidable logic, model computation, equality, size
- Approach: transformation of KB tailored to exploit prover features
Contents

• Transforming the knowledge base for reasoning
  – Transformation of OWL to clause logic: about equality
  – Treating equality
  – Blocking
• Theorem proving
  – KRHyper model generation prover
  – Experimental evaluation
• Rules: an application for reasoning on FrameNet
Transformation of OWL to Clause Logic

- We use the WonderWeb OWL API to get FO Syntax first
- Then apply standard clause normalform trafo (except for "blocking")
- Equality comes in, e.g., for
  - **nominals** ("oneOf")
    
    \[
    \text{WhiteLoire} \sqsubseteq \forall \text{madeFromGrape} . \text{Sauvignon} \sqcup \text{Chenin} \sqcup \text{Pinot}
    \]
    
    \[
    \text{WhiteLoire}(x) \land \text{madeFromGrape}(x, y) \implies \begin{align*}
    y &= \text{Sauvignon} \\
    y &= \text{Chenin} \\
    y &= \text{Pinot}
    \end{align*}
    \]

  - **cardinality restrictions**
    
    \[
    \text{Cation} \sqsubseteq \leq 4 \text{ hasCharge}
    \]
    
    \[
    \text{Cation}(x) \land \text{hasCharge}(x, x1) \land \cdots \land \text{hasCharge}(x, x5) \implies \begin{align*}
    x1 &= x2 \lor x1 &= x3 \lor \cdots \lor x4 = x5
    \end{align*}
    \]

- \(\rightarrow\) Need an (efficient) way to treat **equality**
Equality

• **Option 1:** use equality axioms
  But substitution axioms $x = y \Rightarrow f(x) = f(y)$ - cumbersome

• **Option 2:** use a (resolution) prover with built-in equality
  But how to extract a model from a failed resolution proof?
  We focus on systems for model generation

• **Option 3:** Transform equality away a la Brand's transformation
  **Problem:** Brand's Transformation is not "efficient enough"
  **Solution:** Use a suitable, modified Brand transformation
Brand's Transformation Revisited

Extension of Brand’s Method: UNA for constants (optional)

Add $\neg (a = b)$ for all different constants $a$ and $b$

Modified Flattening

Given: $P(f(x)) \leftarrow f(g(a)) = h(a, x)$

Brand: $P(z_1) \leftarrow f(z_2) = z_3, h(z_4, x) = z_3, f(x) = z_1, g(z_4) = z_2, a = z_4$

Our trafo: $P(f(x)) \leftarrow f(z_1) = h(a, x), g(a) = z_1$

A clause is flatt iff all proper subterms are constants or variables

Our Transformation
- modified flattening
- add equivalence relation axioms for $=$
- add predicate substitution axioms $P(y) \leftarrow P(x), x = y$

It works much better in practice!
Blocking

- **Problem:** Termination in case of satisfiable input
  Specifically: cyclic definitions in TBox
  Example from Tambis KB:

  ![Graph](image)

  TBox

  - **Solution:** Learn from blocking technique from description logics
    "Re-use" previously introduced individual to satisfy exist-quantifier
    Here: encode search for model with finite domain in clause logic:

    \[
    \begin{align*}
    aC(a) \land \text{dom}(a) & \\
    aC(P(A(a))) \land P(A(a)) = a & \\
    aC(P(A(a))) \land \text{dom}(P(A(a))) &
    \end{align*}
    \]

    Try this first

  - **Issue:** Make it work fast: don't be too ambitious on speculating
KRHyper

- **Semantics**
  - Classical predicate logic (refutational complete)
  - Stable models of normal programs (with transformation)
  - Possible models for disjunctive programs (with transformation)

- **Efficient Implementation** (in Ocaml):
  Transitive closure of 16,000 facts -> 217,000 facts:
  
  KRHyper: 17 sec, 63 Mb
  Otter (pos. hyperres) 37 min, 124 Mb
  Compiling SATCHMO: 2:14 h, 271 Mb
  smodels: -

- User manual
- Proof tree output
Computing Models with KRHyper

- Disjunctive logic programs
- Stratified default negation

\[
\begin{align*}
(1) & \quad b ; c ::= a. \\
(2) & \quad a ; d ::= c. \\
(3) & \quad \text{false} ::= a, b. \\
(4) & \quad e ::= c, \text{not d.}
\end{align*}
\]

- Variant for predicate logic
- Extensions: minimal models, abduction, default negation
### Experimental Evaluation

**OWL Test Cases**

<table>
<thead>
<tr>
<th>System</th>
<th>Consistent (56)</th>
<th>Inconsistent (72)</th>
<th>Entailment (111)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRHyper with blocking</td>
<td>86%</td>
<td>89%</td>
<td>93%</td>
</tr>
<tr>
<td>KRHyper w/o blocking</td>
<td>79%</td>
<td>94%</td>
<td>93%</td>
</tr>
<tr>
<td>Fact</td>
<td>42%</td>
<td>85%</td>
<td>7%</td>
</tr>
<tr>
<td>Hoolet</td>
<td>78%</td>
<td>94%</td>
<td>72%</td>
</tr>
<tr>
<td>FOWL</td>
<td>53%</td>
<td>4%</td>
<td>32%</td>
</tr>
<tr>
<td>Pellet</td>
<td>96%</td>
<td>98%</td>
<td>86%</td>
</tr>
<tr>
<td>Euler</td>
<td>0%</td>
<td>98%</td>
<td>100%</td>
</tr>
<tr>
<td>OWLP</td>
<td>50%</td>
<td>26%</td>
<td>53%</td>
</tr>
<tr>
<td>Cerebra</td>
<td>90%</td>
<td>59%</td>
<td>61%</td>
</tr>
<tr>
<td>Surnia</td>
<td>-</td>
<td>0%</td>
<td>13%</td>
</tr>
<tr>
<td>ConsVISor</td>
<td>77%</td>
<td>65%</td>
<td>-</td>
</tr>
</tbody>
</table>
Realistically Sized Ontologies

• **Tambis**
  – About chemical structures, functions, processes, etc within a cell
  – 345 concepts, 107 roles
  – KRHyper: 2 sec per subsumption test

• **Wine**
  – Wine and food ontology, from the OWL test suite
  – 346 concepts, 16 roles, 150 GCIs, ABox
  – KRHyper: 80 sec / 3 sec per negative / positive subsumption test

• **Galen Common Reference Model**
  – Medical terminology ontology
  – big: 24,000 concepts, 913,000 relations, 400 GCIs, transitivity
  – KRHyper: 5 sec per subsumption test

• **OpenCyc**
  – 480,000 (simple) rules. Darwin: 60 sec for satisfiability
Rules

- Adding logic programming style rules is currently discussed in the Semantic Web context (SWRL and many others)

- **Example:**
  
  HomeWorker\(x\) ← work\(x, y\) ∧ live\(x, z\) ∧ loc\(y, w\) ∧ loc\(z, w\)

  Cannot be expressed in description logics

- Adding rules to the input language is trivial in approaches that transform ontologies to clause logic

- **Problem:** can simulate Role-Value maps, leading to undecidability

- **Rationale** of doing it nevertheless:
  
  - Better have only a semi-decision procedure than nothing
  
  - In many cases have termination nonetheless (with blocking)
  
  - Really useful in some applications
From Natural Language Text To Frame Representation

FrameNet
550 Frames
7000 Lex Units

Linguistic Method

Text

BMW bought Rover from BA

Logic

Deduction System

Frame Representation
Com GT
Buyer: BMW
Seller: BA
Goods: Rover
Money: unknown

Work in Collaboration with Computer Linguistics Department (Prof. Pinkal)
The plane manufacturer has from Great Britain the order for 25 transport planes received.

**Task:** Fill in the missing elements of „Request“ frame
Transfer of Role Fillers

The plane manufacturer has from Great Britain the order for 25 transport planes received.

Parsing gives partially filled FrameNet frame instances of „receive“ and „request“:

receive1: receive
  target: „received“
  donor: „Great Britain“
  recipient: manufacturer1
  theme: request1

request1: request
  target: „order“
  speaker: „Great Britain“
  addressee: manufacturer
  message: „transport plane“

- Transfer of role fillers done so far manually
- Can be done automatically. By „model generation“
Transfer of Role Fillers by Rules

receive1:
  receive
  target: "received"
  donor: "Great Britain"
  recipient: manufacturer1
  theme: request1

request1:
  request
  target: "order"
  speaker: "Great Britain"
  addressee: 
  message: "transport plane"

Rules
speaker(Request, Donor) :-
  receive(Receive),
  donor(Receive, Donor),
  theme(Receive, Request),
  request(Request).

Facts
receive(receive1).
  donor(receive1, "Great Britain").
  theme(receive1, request1).
  request(request1).
Exploiting Nonmonotonic Negation: Default Values

Insert default value as a role filler in absence of specific information

receive1:  
  receive  
  target: „received“  
  donor: „Great Britain“  
  recipient: manufacturer1  
  theme: request1

request1:  
  request  
  target: „order“  
  speaker: „Great Britain“  
  addressee:  
  message: „transport plane“

Should transfer "donor" role filler only if "speaker" is not already filled:

default_request_speaker(Request, Donor) :-
  receive(Receive),
  donor(Receive, Donor),
  theme(Receive, Request),
  request(Request).
Default Values

Insert default value as a role filler in absence of specific information

Example:
In Stock Market context use default "share" for "goods" role of "buy":

```
default_buy_goods(Buy, "share") :-
'Buy is an event in a stock market context'.
```

Example:
Disjunctive (uncertain) information

Linguistic analysis is uncertain whether "Rover" or "Chrysler" was bought:

```
default_buy_goods(buy1,"Rover").
default_buy_goods(buy1,"Chrysler").
```

This amounts to two models, representing the uncertainty.
They can be analyzed further.
Default Value – General Transformation

Technique:

\[
\begin{align*}
  a & : = \text{not } \text{not}_a. \\
  \text{not}_a & : = \text{not } a. \\
\end{align*}
\]

has two stable models: one where \( a \) is true and one where \( a \) is false

Choice to fill with default value or not:

\[
\begin{align*}
  \text{goods}(F,R) & : = \\
  & \quad \text{not } \text{not}_\text{goods}(F,R), \\
  & \quad \text{buy}(F), \\
  & \quad \text{default}_\text{buy}_\text{goods}(F,R). \\
  \text{not}_\text{goods}(F,R) & : = \\
  & \quad \text{not } \text{goods}(F,R), \\
  & \quad \text{buy}(F), \\
  & \quad \text{default}_\text{buy}_\text{goods}(F,R). \\
\end{align*}
\]

Require at least one filler for role:

\[
\begin{align*}
  \text{false} & : = \\
  & \quad \text{buy}(F), \\
  & \quad \text{default}_\text{buy}_\text{goods}(F,R_1), \\
  & \quad \text{goods}(F,R_1), \\
  & \quad \text{goods}(F,R_2), \\
  & \quad \text{not equal}(R_1,R_2). \\
  \text{equal}(X,X). & \\
\end{align*}
\]

Case of waiving default value:

\[
\begin{align*}
  \text{false} & : = \\
  & \quad \text{buy}(F), \\
  & \quad \text{default}_\text{buy}_\text{goods}(F,R_1), \\
  & \quad \text{goods}(F,R_1), \\
  & \quad \text{goods}(F,R_2), \\
  & \quad \text{not equal}(R_1,R_2). \\
  \text{equal}(X,X). & \\
\end{align*}
\]

Role is filled:

\[
\begin{align*}
  \text{some}_\text{buy}_\text{goods}(F) & : = \\
  & \quad \text{buy}(F), \\
  & \quad \text{goods}(F,R). \\
\end{align*}
\]
Conclusions

• Objective: "robust" reasoning support beyond description logics
• Method
  – FO theorem prover, specifically model generation paradigm
  – Tailor translation to capitalize on prover features
  – Exploit nonmonotonic features (for KB with FO semantics!)
• Practice
  – Experimental evaluation on OWL test suite "promising"
  – Need more experiments with e.g. OpenCyc and FrameNet