Classification & Experimentation

Bijan Parsia
<bijan.parsia@manchester.ac.uk>
Today

• Some brief hints on engineering
• Thinking about the problem
• Classification without heat death
• Experimentation on reasoning
A Tale of Three Rules

Wherein we see that a little thought goes a long way
Consider the □-rule

□-rule: if \( C_1 \sqcap C_2 \in L(a) \) for some \( a \) and \( \{ C_1, C_2 \} \not\subseteq L(a) \)

then add \( \{ C_1, C_2 \} \) to \( L(a) \)

- How do you implement this?
- When do you fire this?
Consider the clash rule

\[ \{A, \neg A\} \not\subseteq L(a), \bot \not\in L(a) \text{ for all } a, A \]

- How do you test for clashes?
- How often do you test for clashes?
Consider the $\square$-rule

$\square$-rule: if $C_1 \sqcup C_2 \in L(a)$ for some $a$ and $\{C_1, C_2\} \cap L(a) = \emptyset$
then replace $G$ with $G_1$ and $G_2$ s.t.
$C_1 \in L(a)$ in $G_1$ and $C_2 \in L_2(a)$ in $G_2$

- How do you create $G_1$ and $G_2$?
  - Deep copy?
There’s other stuff in your code!

```java
debugString = currentClass.toString();
if (debug)
    System.out.println(debugString);

debugString = currentClass.toString();
if (debug)
    System.out.println(debugString);
```

- What’s wrong with this?
- What should it look like?
- This appeared in real code
  - by one of the top 4 programmers I know
Classify Classification

Wherein we despair
Classification: Formulations

• Most standard formulation:
  – For all class names A, B in O, determine whether
    • $O \models A \sqsubseteq B$ (or not)

• Slight generalisation
  – For all class names (plus $\bot$ and $\top$), A, B in O, determine whether
    • $O \models A \sqsubseteq B$
  – This neatly includes
    • the consistency check (i.e., whether $O \models \top \sqsubseteq \bot$)
    • concept satisfiabilities (i.e., whether $O \models A \sqsubseteq \bot$)
    • concept trivialities (i.e., whether $O \models \top \sqsubseteq A$)

• Full generalisation
  – For all predicate names (user defined or built in) A, B in O, determine whether
    • $O \models A \sqsubseteq B$
  – This includes roles/properties/binary predicates!
  – Few systems do this! (Only HermiT?)
Classification: For implementation (1)

• Close Functional Formulation (CFF)
  – Input: An ontology $O$ (i.e., a set of axioms)
  – Output: A new ontology $O'$ st
    • $O' = \{A \sqsubseteq B \mid A, B \in (\hat{O} \cup \{\bot, T\}) \land O \models A \sqsubseteq B\}$ or
      – Transitive closure
    • $O' = \{A \sqsubseteq B \mid A, B \in (\hat{O} \cup \{\bot, T\}) \land O \models A \sqsubseteq B$
      & $\nexists C \text{ s.t. } \{A \sqsubseteq C, C \sqsubseteq B\} \subseteq O'$
      – Transitive reduct...almost
    • (Pick your favorite data structure to represent this)

• Problems?
CFF Problems: \( \bot \) and \( \top \)

- Many subsumptions involving \( \bot \) and \( \top \) are trivial
  - \( A \subseteq \top \)
  - \( \bot \subseteq A \)
  - Other key trivial subsumption:
    - \( A \subseteq A \) (and circular!)

- Non trivial examples “blow up” the transitive closure
  - If \( A \subseteq \top \), then \( A \) is subsumed by every other term
CFF Problems: Equivalences

- In CFF, if $A \equiv B$, then
  - it shows up as $A \sqsubseteq B$ and $B \sqsubseteq A$
- But what happens with longer chains?
- Problems even
  - if we allow equivalences

\[
\begin{align*}
A \equiv B & \quad A \equiv B & \quad A \equiv B \\
B \equiv C & \quad B \equiv C & \quad B \equiv C \\
A \equiv C & \quad A \equiv C & \quad A \equiv D
\end{align*}
\]
Transitive Reduct Saves the Day?

• Modified Functional Formulation (CFF)
  – Input: An ontology $O$ (i.e., a set of axioms)
  – Output: A new ontology $O'$ st
    • $O' = \{A \sqsubseteq B \mid A, B \in (\bar{O} \cup \{\bot, \top\}) \& O \vdash A \sqsubseteq B^*$
      & $\not\exists C$ s.t. $\{A \sqsubseteq C, C \sqsubseteq B\} \subseteq O'$
      & $O \nvdash A \equiv B\} \cup$
    
    $\{\equiv(A_1...A_n)** \mid A_1...A_n \in (\bar{O} \cup \{\bot, \top\}) \&$
    
    \[1\leq i,j\leq n^*, \ O \vdash A_i \equiv A_j\}$

  *And a few more side conditions

  **Where $A_1...A_n$ is “appropriately” sorted

  – (Pick your favorite data structure to represent this)

• Great for some applications
  – But the downstream application should know your particulars!

• Bad for some applications
Downstream apps...oy!

Declaration(Class(:A))
EquivalentClasses(:A owl:Nothing)
Declaration(Class(:B))
EquivalentClasses(owl:Nothing :A)

Declaration(Class(:A))
EquivalentClasses(:A owl:Nothing)
SubClassOf(:A owl:Thing)
Declaration(Class(:B))
SubClassOf(:B owl:Thing)
EquivalentClasses(owl:Nothing :A)
Counting Entailments

- **Goal:**
  - Given $O_1$ and $O_2$, determine
    - whether $O_1$ has “more entailments” than $O_2$
    - restrict our attention to atomic subsumptions

- Easy if one entails the other

- Transitive reduct fails to be monotonic

\[
\begin{align*}
  X_1 &\sqsubseteq A \\
  X_2 &\sqsubseteq A \\
  X_3 &\sqsubseteq A \\
  X_1 &\sqsubseteq B \\
  X_2 &\sqsubseteq B \\
  X_3 &\sqsubseteq B \\
  A &\sqsubseteq B
\end{align*}
\]

\[
\begin{array}{c}
  \overset{B}{A} \\
  \overset{X_1}{X_2} \\
  \overset{X_3}{4}
\end{array}
\]
Counting Entailments

- Goal:
  - Given $O_1$ and $O_2$, determine
    - whether $O_1$ has “more entailments” than $O_2$
    - restrict our attention to atomic subsumptions

- Easy if one entails the other
- Transitive reduct fails to be monotonic
Extended notions

- What about disjointnesses?
  - Negative literals as well!
    - \( L = \emptyset \cup \{\neg A \mid A \in \emptyset\} \)
    - \( \{A \sqsubseteq B \mid A, B \in L \land \emptyset \vdash A \sqsubseteq B\} \)
    - Note redundancy and choice!
      - \( A \sqsubseteq B \iff \neg B \sqsubseteq \neg A \)
      - \( A \sqsubseteq \neg B \iff B \sqsubseteq \neg A \)

- Beyond literals!
  - We could classify sub-expressions (Sub)
    - \( A \in \text{Sub} \)
      - \( C \sqsubseteq D \in \text{Sub} \rightarrow C,D \in \text{Sub} \)
      - \( \neg C \in \text{Sub} \rightarrow C \in \text{Sub} \)
      - \( C \sqcup D \in \text{Sub} \rightarrow C,D \in \text{Sub} \)
      - \( C \sqcap D \in \text{Sub} \rightarrow C,D \in \text{Sub} \)
      - \( \exists P.C \in \text{Sub} \rightarrow C \in \text{Sub} \)
      - \( \forall P.C \in \text{Sub} \rightarrow C \in \text{Sub} \)
  - \( \{A \sqsubseteq B \mid A, B \in \text{Sub} \land \emptyset \vdash A \sqsubseteq B\} \)
Classify before you die

Wherein we avoid work
3 RoughClassification Approaches

- Reduction to SAT
  - All tableaux & hypertableaux systems
    - Dominant, covers arbitrary languages
- Consequence based
  - Currently for fragments, esp. EL and horn-SHIQ
- Meta/Modular
Subsumption tests

• There can always be $n^2$ subsumption tests
  – Four possible states
    1. $O \models A \subseteq B$
       – In all models of $O$, $A^I \subseteq B^I$
    2. $O \not\models A \subseteq B$
       – In at least one model, $A^I \not\subseteq B^I$
    3. $O \models A \subseteq \lnot B$
       – In all models of $O$, $A^I \cap B^I = \emptyset$
    4. $O \models \lnot(A \subseteq B)$
       – In every model, $A^I \not\subset B^I$

• We look for 1 & 2
  – 3 and 4 entail 2
  – Handy fact!
Key issues

• There can always be $n^2$ subsumptions
  – Consider $O \models T \sqsubseteq \bot$!
    • But this case doesn’t require $n^2$ tests

• 1 subsumption test
  – Can dominate
  – Easiest to see in SAT based procedures
    • If SAT is NP-hard (EXPTIME, NEXPTIME, 2NEXPTIME), then one such test can kill you
      • $A \sqcap \neg B$
    – But even with a PTIME SAT test...
      • The quadratic factor can kill you
The quadratic factor

- Consider the SNOMED CT ontology
  - contains about 300,000 terms.

- Presume the naive approach
  - Perform $\approx n^2$ subsumption tests

- Let your test we wicked fast
  - 1 millisecond per test

- Classification time
  $300,000 \times 300,000$ milliseconds
  $= 25,000$ hours
  $\approx 2.8$ years

Any practically scalable classification implementation must prune the subsumption test space
SAT based procedures

Wherein we get satisfaction
SAT based procedures

- **Refutation procedure**
  - Via reduction to an concept
    - \( C \sqcap \neg D \)

- **Individual SAT tests**
  - Positive: Concept is unsatisfiable; subsumption holds
  - Negative: Concept is satisfiable; nonsubsumption

- **Basic strategy**
  1. Avoid tests
  2. Substitute cheap (generally sound, but incomplete) tests
    - SAT procedure independent (some are part of 1)
    - Exploit extra info from the SAT test
  3. Worst case, do a “full” SAT test
    - And complain about it!
Enhanced Traversal

• Data structure:
  – A DAG where
    • Nodes are (sets of) concept names
    • Edges indicate subsumption relations
    • Initialize with $\bot \rightarrow \top$

• General idea
  – DAG represents the transitive reduct of atomic subsumption
  – Add subsumptions as you find them
  – Don’t look for subsumptions that are
    • implicit in the graph
    • impossible in the graph
  – Defer looking for subsumptions
    • where they are unlikely
ET: Top search (Top down)

- Given a fresh concept, \( C \), to classify
- Starting from \( \top \) check whether
  \[ -C \subseteq \top \]
ET: Top search (Top down)

- Given a fresh concept, C, to classify
- Starting from $\top$ check whether
  - $C \sqsubseteq \top$
    - Easy yes!
  - Only candidate left is $\bot$
    - SAT test!??!?!
    - (In some cases)
    - Answer (let’s say): no
    - No other candidates for subsumers
    - Done Top search for C
ET: Bottom search (Bottom Up)

- Given our placed concept C
- Starting from $\perp$ check whether $\neg \perp \sqsubseteq C$
ET: Bottom search (Bottom Up)

• Given our placed concept C
• Starting from \( \bot \) check whether
  
  \(-\ \bot \subseteq C \)

  • Easy yes!
  
  \(-\text{What’s left?} \)

  • Only candidate is \( \top \)
  
  • \( \top \) subsumes all subsumees of C

  – Potential SAT test!!!

  – In this case, \( \top \not\subseteq C \)

  – So we’re done!
Information reuse

- **Top Down**
  - If we know
    - $E \subseteq D$
    - $C \not\subseteq D$
  - Then we know
    - $C \not\subseteq E$
    - *No need to perform a test!*

- **Bottom up**
  - If we know
    - $E \subseteq D$
    - $E \not\subseteq C$
  - Then we know
    - $D \not\subseteq C$
    - *No need to perform a test!*
Savings

• Possible tests (assuming consistency)
  – Total
    • \( n = 5 \)
    • \( n^2 = 25 \)

• Count (order C, D, E)
Savings

• Possible tests (assuming consistency)
  – Total
    • $n = 5$
    • $n^2 = 25$
  – Count (order C, D, E)
    – (1) trivial ($\perp \subseteq \top$)
    – (2) non trivial ($\top \subseteq \perp$)
      • Consistent! (SAT)!
  – C
    • Top Down
Savings

- Possible tests (assuming consistency)
  - Total
    - \( n = 5 \)
    - \( n^2 = 25 \)
  - Count (order C, D, E)
    - (1) trivial (\( \bot \leq \top \))
    - (2) non trivial (\( \top \leq \bot \))
      - Consistent! (SAT)
    - C
      - Top Down
        - (3) \( C \leq \top \) (trivial!)
        - (4) \( C \leq \bot \) (hard!)
          » C is is satisfiable (SAT)
Savings

- Possible tests
  - Total
    - \( n = 5 \)
    - \( n^2 = 25 \text{ SAT!} \)

- Count (order C, D, E)
  - (1) trivial (\( \bot \subseteq \top \) )
  - (2) non trivial (\( \top \subseteq \bot \) )
    - Consistent! (SAT)
  - C [1 SAT, 1 Trivial]
    - Bottom up
      - (5) \( \bot \subseteq C \) (trivial)
      - (6) \( \top \subseteq C \) (SAT!)
Savings

• Possible tests
  – Total
    • $n = 5$
    • $n^2 = 25$ SAT!?
  – Count (order C, D, E)
    – (1) trivial ($\bot \in \top$)
    – (2) non trivial ($\top \in \bot$)
      • Consistent! (SAT)
    – C [2 SAT, 2 Trivial]
    – D
      • Top Down
        – (7) $D \in \top$ (trivial!)
        – (8) $D \in C$ (SAT!)
        – (9) $D \in \bot$ AVOIDED
Savings

- Possible tests
  - Total
    - $n = 5$
    - $n^2 = 25$ SAT!
- Count (order C, D, E)
  - (1) trivial ($\bot \subseteq \top$)
  - (2) non trivial ($\top \subseteq \bot$)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [1 SAT, 1 Trivial, 1 Avoided]
    - Bottom up
      - (10) $\bot \subseteq D$ Trivial
      - (11) $C \subseteq D$ (SAT)
      - (12) $\top \subseteq D$ AVOIDED!
Savings

- Possible tests
  - Total
    - $n = 5$
    - $n^2 = 25$ SAT!
- Count (order C, D, E)
  - (1) trivial ($\bot \subseteq \top$)
  - (2) non trivial ($\top \subseteq \bot$)
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [2 SAT, 2 Trivial, 2 Avoided]
  - E
    - Top Down
      - (13) $E \subseteq \top$ (trivial)
      - (14) $E \subseteq C$ (SAT)
      - (15) $E \subseteq D$ (SAT)
      - (16) $E \subseteq \bot$ (AVOIDED)
Savings

- Possible tests
  - Total
    - \( n = 5 \)
    - \( n^2 = 25 \) SAT!
- Count (order C, D, E)
  - (1) trivial (\( \bot \subseteq \top \))
  - (2) non trivial (\( \top \subseteq \bot \))
    - Consistent! (SAT)
  - C [2 SAT, 2 Trivial]
  - D [2 SAT, 2 Trivial, 2 Avoided]
  - E [1 Trivial, 2 SAT, 1 Avoided]
    - Bottom up
      - (17) \( \bot \subseteq E \) (trivial)
      - (18) C \( \subseteq E \) AVOIDED
      - (19) D \( \subseteq E \) (SAT)
      - (20) \( \top \subseteq E \) AVOIDED
Savings

• Possible tests
  – Total
    • $n = 5$
    • $n^2 = 25$ SAT!? 

• Count (order C, D, E)
  – (1) trivial ($\bot \subseteq \top$)
  – (2) non trivial ($\top \subseteq \bot$)
    • Consistent! (SAT)
  – C [2 SAT, 2 Trivial]
  – D [2 SAT, 2 Trivial, 2 Avoided]
  – E [2 Trivial, 3 SAT, 3 Avoided]
  – Reflexive!
    • $C \subseteq C$
      – All avoided = 5
  • Total [8 SAT, 7 Trivial, 5+5 avoided] = 25
9 SAT seem like a lot!

• Assertions!
  – If our ontology contains $E \sqsubseteq D$
    • We can just enter that! No sat!
  – If our ontology contains (or implies) $C \sqsubseteq \neg D$
    • Then we don’t need to test $D \sqsubseteq C$, $C \sqsubseteq D$, $E \sqsubseteq C$
    • 4 tests gone!
  – We can look for cheap consequences
    • E.g., $A \sqsubseteq C \sqcap D$ immediately gives $A \sqsubseteq C$, $A \sqsubseteq D$
      – Must take care about $\top \sqsubseteq A$, $C$, $D$ or $\top \sqsubseteq A$, $C$, $D$

• Exploit internals
  – For any SAT test we can
    • extract a representation of a model
    • if we have such a “pseudo-model” of $C$ and of $\neg D$
      – We can see if they merge to form a new model
        » Done!