The Mobile Semantic Web

Shonali Krishnaswamy\textsuperscript{a} & Yuan-Fang Li\textsuperscript{b}

\textsuperscript{a} Institute of Infocomm Research, A*STAR, Singapore
\textsuperscript{b} Faculty of Information Technology, Monash University, Australia

\{shonali.krishnaswamy,yuanfang.li\}@monash.edu
Agenda

1. Introduction & Motivation

2. A brief introduction to ontology languages & reasoning

3. Strategies & systems for mobile semantic reasoning

4. Future Directions & Discussions
Mobile Semantic Reasoning:

Why ?
Mobile Semantic Web

What is happening?

- **Proliferation** of Mobile Devices
- **Pervasive** Connectivity
- **Phenomenal** increase in Mobile Content

What is it leading to?

- **Monetisation** of Mobile Data and Content
  - Mobile Crowdsourcing & Crowdsensing
- **Context-Aware** Information & Service Delivery
- **KYC** – Know Your Customer
Why Mobile Semantic Web?

- Managing Data & Content On-Board Devices
  - Semantic Search on Mobile Devices
Service Matching in Mobile Environments

- Keyword / String-based Matching On-board a Mobile Device
- Semantic Matching on a Remote Server
- Hybrid
- Semantic Matching On-board a Mobile Device
Why Mobile Semantic Web?

- Semantics of Locations & Trajectories
  - Can we infer and manage location semantics on-board devices?
Why Mobile Semantic Web?

- The Tell-Tale Phone
  - KYC & Mobile Users

Where is the user now, the semantic meaning of the current place?

Home, office, friend’s home, transportation location, … ?

Where is the next place that the user would go to?

Is the user a male or female? How old is he/she? What types of job is he/she doing? …
Why Mobile Semantic Web?

- Mobile Data Mining – Here and Now!
- The Performance & Privacy Story
  - **Privacy** – Localised management of user data and applications (when needed)
  - **Battery Usage** – Continuous communication is more expensive on energy usage than processing
  - **Disconnections** – Mobile users face intermittent connectivity
  - **Network Bottlenecks/Scalability** – The Cloud can provide storage and compute resources, but mobile networks are still a bottleneck
Mobile Hosted Web Services

- **Key Challenges**

- Identify services that are relevant to the user’s changing context: *location, device connectivity levels, QoS*

- Small mobile devices are typically resource constrained in terms of *processing power, memory capabilities, screen size, battery life*.

- Mobile users require quick feedback
  - Attention Span approx 15 s

- Information needs at a high level: *result accuracy*.
Mobile Semantic Reasoning

- Mobile reasoning is important
- Mobile reasoning is hard
  - Computationally complex
  - Mobile devices have limited resources
  - Humans have limited attention span

![Bar Chart: Classification Time in Seconds (log-scale)]

![Image: Out Of Memory Error]
Semantic Reasoning:

What?
Ontology Reasoning

Basic Description Logics
Reasoning Tasks & Their Complexity
Tableau Algorithms
Description Logics

- A family of knowledge representation formalisms (Baader et al (2003))
  - Successor of semantic networks & Minsky frames
  - Describes domains using *concepts, roles & individuals*
- Formal semantics
  - A decidable fragment of first-order logic (generally speaking)
  - Some DLs are variants of modal logic

Knowledge Base

- **TBox (schema)**
  - \( \text{Animal} \sqsubseteq \text{LivingThing} \)
  - \( \text{Carnivore} \sqsubseteq \text{Animal} \lor \forall \text{eats. Animal} \)
  - ...
- **Rbox (roles)**
  - \( \text{eats} \sqsubseteq \text{consumes} \)
- **ABox (instance)**
  - \( \text{simba} \in \text{Carnivore} \)
  - ...

© Disney Inc.
Description Logics: Main Ingredients

- Used for knowledge representation

- Three types of main entities
  - **Concepts** (classes): *first-class citizen*, representing sets of abstract entities
    
    \[ \text{MeatyPizza} \sqsubseteq \text{Pizza} \]

  - **Roles** (properties, predicates): binary relations
    
    \[ \text{hasTopping} \sqsubseteq \text{hasIngredient} \]

  - **Individuals**: entities
    
    \[ \text{Country} \equiv \{ \text{America, England, France, Germany, Italy} \} \]
Description Logics: Expressions & Axioms

- **Expressions** used to construct complex concepts & roles
  
  \[
  \text{Food} \sqcap \exists \text{hasBase}. \text{PizzaBase} \\
  \text{Pizza} \sqcap \exists \text{hasTopping}. \text{MeatTopping}
  \]

- **Axioms** used to express relationship between concepts, roles & individuals
  
  – Subsumption, equivalence, disjointness, assertions
  
  \[
  \text{Pizza} \sqsubseteq \text{Food} \sqcap \exists \text{hasBase}. \text{SomeBase} \\
  \text{MeatyPizza} \equiv \text{Pizza} \sqcap \exists \text{hasTopping}. \text{MeatTopping} \\
  \text{VegetarianPizza} \sqcap \text{NonVegetarianPizza} \sqsubseteq \bot \\
  \text{Country}(\text{America})
  \]
**Description Logics: Syntax & Semantics**

- The DL syntax for concept expressions ($\mathcal{ALC}$):
  - $C ::= \top \mid \bot \mid A \mid \neg C \mid C \cap D \mid C \cup D \mid \exists R.C \mid \forall R.C$
  - Other syntaxes available: RDF/XML, etc.
- Set-based semantics: $(\Delta^I, \cdot^I)$
  - The domain of interpretation $\Delta^I$: the universal set of individuals
  - The interpreting function $\cdot^I$: maps concepts/roles/individuals into $\Delta^I$

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Interpretation</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top$</td>
<td>$\Delta^I$</td>
<td>Everything</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\emptyset$</td>
<td>Nothing (empty set)</td>
</tr>
<tr>
<td>$A$</td>
<td>$A^I \subseteq \Delta^I$</td>
<td>A class is a set of individuals</td>
</tr>
<tr>
<td>$C_1 \cap C_2$</td>
<td>$C^I_1 \cap C^I_2$</td>
<td>Intersection of 2 sets of individuals</td>
</tr>
<tr>
<td>$C_1 \cup C_2$</td>
<td>$C^I_1 \cup C^I_2$</td>
<td>Union of 2 sets of individuals</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>$\Delta^I \setminus C^I$</td>
<td>Subtraction from the domain</td>
</tr>
</tbody>
</table>
A Spectrum of Expressivity (1)

- More expressive **extensions**
  - OWL Lite: $\text{SHIF}(D)$ (Horrocks & Patel-Schneider (2003))
    - Transitive, inverse, functional roles & role hierarchy,
  - OWL DL: $\text{SHOIN}(D)$ (Horrocks, Sattler, Tobies (1999))
    - Nominals & qualified number restrictions
  - OWL 2 DL: $\text{SROIQ}(D)$ (Horrocks, Kutz, Sattler (2006))
    - Complex role inclusions

<table>
<thead>
<tr>
<th>DL</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL Lite</td>
<td>$\text{hasTopping} \sqsubseteq \text{hasIngedient}$</td>
</tr>
<tr>
<td>OWL DL</td>
<td>$\geq 3 \text{hasTopping}$</td>
</tr>
<tr>
<td>OWL 2 DL</td>
<td>$\geq 3 \text{hasTopping}.\text{MeatTopping}$</td>
</tr>
</tbody>
</table>
A Spectrum of Expressivity (2)

- Less expressive **subsets**
  - Syntactic restrictions on class expressions
  - OWL 2 EL: $\mathcal{EL}^{++}$ (Baader, Brandt, Lutz (2005))
    - Designed for efficient reasoning over large TBoxes
    - Especially biomedical ontologies
  - OWL 2 QL: $DL-Lite_R$ (Calvanese et al (2005))
    - Designed for *very* efficient query answering over large Aboxes
    - Ontology-based data access
  - OWL 2 RL: $DLP \ & pD^*$ (Grosof et al (2003); Horst (2005))
    - Designed for efficient rule-based reasoning
Description Logics: Reasoning Tasks

- Based on the interpretation, for TBox & Abox
- Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
- For TBox $\mathcal{T}$
  - Concept satisfiability $\mathcal{I} \vDash C \text{ iff } C^\mathcal{I} \neq \emptyset$
  - Subsumption $\mathcal{I} \vDash C \subseteq D \text{ iff } C^\mathcal{I} \subseteq D^\mathcal{I}$
  - TBox consistency $\mathcal{I} \vDash \mathcal{T} \text{ iff } \mathcal{I} \vDash \varphi$, for every axiom $\varphi \in \mathcal{T}$
- For ABox $\mathcal{A}$
  - Concept assertions $\mathcal{I} \vDash C(a) \text{ iff } a^\mathcal{I} \in C^\mathcal{I}$
  - Role assertions $\mathcal{I} \vDash R(a, b) \text{ iff } (a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$
  - ABox consistency $\mathcal{I} \vDash \mathcal{A} \text{ iff } \mathcal{I} \vDash \varphi$, for every axiom $\varphi \in \mathcal{A}$
- Can all be reduced to ABox consistency
## Reasoning Complexity

<table>
<thead>
<tr>
<th>DL</th>
<th>TBox consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALC</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>SHIF(D) (OWL Lite)</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>SHOIN(D) (OWL DL)</td>
<td>NEXPTIME-complete</td>
</tr>
<tr>
<td>SROIQ(D) (OWL 2 DL)</td>
<td>2NXPTIME-complete</td>
</tr>
<tr>
<td>EL++ (OWL 2 EL)</td>
<td>PTIME-complete</td>
</tr>
<tr>
<td>DL-LiteR (OWL 2 QL)</td>
<td>NLOGSPACE-complete</td>
</tr>
<tr>
<td>OWL 2 RL</td>
<td>PTIME-Complete</td>
</tr>
</tbody>
</table>

*With great expressivity comes great complexity*
Reasoning is Hard

- Especially for large ontologies and/or resource-constrained devices
Reasoning is **Hard**

- Difficulty a result of:
  - Ontology:
    - Size (measured in different ways)
    - Language constructs used (union, at most, *etc.*)
  - Reasoner:
    - Tableau algorithms: exhaustive exploration until all ABoxes checked
      - Sound & complete, but **expensive**
    - Optimisation techniques employed
Reasoning Paradigms

- **Tableau** algorithms (Baader & Sattler (2001))
  - Mostly widely used, suitable for very expressive DLs
  - Reasoners: FaCT++ (Tsarkov & Horrocks (2006)), Racer (Haarslev & Möller (2001)), HermiT (Shearer, Motik & Horrocks (2008)), Pellet (Sirin et al (2007))

- **Completion rules-based** algorithms (Baader, Brandt, Lutz (2005))
  - Suitable for less expressive DLs: the $\mathcal{EL}$ family
  - Reasoners: CEL (Baader, Lutz & Suntisrivaraporn (2006)), REL (Ren & Pan (2010)), Snorocket (Lawley & Bousquet (2010))

- **Consequence-driven** algorithms (Kazakov (2009))
  - A recent, efficient algorithm for Horn $\mathcal{SHIQ}$ and beyond
  - Reasoners: ELK (Kazakov Krötzsch & Simančík (2011)), CB, ConDOR (Simančík, Kazakov, Horrocks (2011))
Completion Rules Algorithm

- Many large biomedical ontologies don’t use/need the full expressivity of OWL 2 DL:
  - Gene Ontology, NCI Thesaurus, Gazetteer, and many, many more
  - They can be expressed using a very limited set of constructs

- $\mathcal{EL}$ was designed to exploit this fact:
  - Simpler logic, faster reasoning!

- Completion rules-based algorithm
  - An algorithm for calculating classification
    - The subsumption hierarchy of an ontology
    - Apply completion rules to saturate the subsumption graph
    - PTIME-complete, optimal!
Tableau Algorithms

- Foundation of several highly optimised reasoners
  - FaCT++, Racer, Pellet, HermiT, etc.

- Support TBox reasoning, *sound & complete*
  - Through reduction to *ABox consistency checking*
  - E.g., concept satisfiability: build a *tree-like model* for \( C \) by applying *expansion rules* on an ABox: \( A \models C(a) \) iff \( A \cup \{ \neg C(a) \} \) is inconsistent
  - Models are ABoxes

- An ABox is
  - Complete: if no more rules apply
  - Closed: if it contains a *clash*
  - Open: if it doesn’t contain a clash
Tableau Algorithm: Basic Idea

- Basic idea: given a concept $C$
  - Apply expansion rules repeatedly to syntactically decompose $C$
  - Stop when
    - A clash occurs: $C$ is unsatisfiable, or
    - No more rules apply & no clash detected: $C$ is satisfiable
- Clash (contradiction) — many forms
  - $\{C(a), \neg C(a)\} \in \mathcal{A}$
  - $\{\bot(a)\} \in \mathcal{A}$
  - $\{(\leq n \ r)(a)\} \cup \{r(a, b_i)|1 \leq i \leq n + 1\} \cup \{y_i \neq y_j|1 \leq i <\leq n + 1\} \subseteq \mathcal{A}$
    - for $a, y_1, \ldots, y_n + 1 \in \mathcal{N}_I, r \in \mathcal{N}_R, n \in \mathbb{N}$
  - ... 
- Transformation (expansion) rules
  - Can be deterministic or nondeterministic
  - Source of complexity
Expansion Rules

- Each language construct has a rule: \( \Pi, \sqcup, \exists, \forall, \leq, \geq, \text{etc.} \)
  - Consistency preserving
  - Can generate new individuals \( \exists \)-rule, \( \geq \)-rule
  - Can be non-deterministic \( \sqcup \)-rule, \( \leq \)-rule
  - Can be applied simultaneously – order matters greatly!

\[ \Pi \text{-rule} \]
\[
\begin{align*}
(C_1 \cap C_2)(a) & \in A \\
C_1(a) & \notin A \land C_2(a) \notin A \\
\hline
A' & := A \cup \{C_1(a), C_2(a)\}
\end{align*}
\]

\[ \sqcup \text{-rule} \]
\[
\begin{align*}
(C_1 \sqcup C_2)(a) & \in A \\
C_1(a) & \notin A \land C_2(a) \notin A \\
\hline
A' & := A \cup \{C_1(a)\}, A'' := A \cup \{C_2(a)\}
\end{align*}
\]

\[ \exists \text{-rule} \]
\[
\begin{align*}
(\exists r. C)(a) & \in A \\
\forall z \in A \bullet C(z) & \in A \land r(a, z) \in A \\
\hline
A' & := A \cup \{C(b), r(a, b)\}
\end{align*}
\]

\[ \forall \text{-rule} \]
\[
\begin{align*}
(\forall r. C)(a) & \in A, r(a, b) \in A \\
C(b) & \notin A \\
\hline
A' & := A \cup \{C(b)\}
\end{align*}
\]
Tableau Algorithm: An Example

- **Individuals** \( \{x_0, x_1, \ldots, x_7\} \subseteq A \)
- **Assertions** \( \mathcal{L}(x_3) = \{C_1, \neg C_4, C_4 \sqcup C_5\} \)
  \( \mathcal{L}(x_4) = \{C_2\} \)
  \( \mathcal{L}(x_5) = \{C_2, C_3\} \)
  \( \mathcal{L}(x_6) = \{\forall R_3.(\neg C_1 \sqcup \neg C_2)\} \)
  \( \mathcal{L}(x_7) = \{C_1\} \)
- **Task** \( A \models C_0(x_0) \), where \( C_0 \equiv \exists R_1.(\geq 1 R_2) \cap \exists R_1.(C_1 \cap \exists R_1.(C_2 \cap C_3)) \)
- **Negated** \( A \cup \{((\forall R_2.(\leq 0 R_2)) \cup (\forall R_1.(\neg C_1 \sqcup \forall R_1.(\neg C_2 \sqcup \neg C_3))))(x_0)\} \) closed?
Tableau Algorithm: An Example

∀-rule: ∀R₃. (¬C₁ ⊢ ¬C₂) ∈ ℒ(x₆)

L(x₇) ∪ {¬C₁ ⊢ ¬C₂}

□-rule: ∀R₂. (≤ 0R₂) □
∀R₁.(...) ∈ ℒ(x₀)

L(x₀) ∪ {∀R₂. (≤ 0R₂)}

∀-rule: ∀R₂. (≤ 0R₂) ∈ ℒ(x₀)

L(x₁) ∪ {≤ 0R₂}

≤ -rule: (≤ 0R₂) ∈ ℒ(x₁)

Clash #R₂[{x₁}] > 0

□-rule: ∀R₂. (≤ 0R₂) □
∀R₁. (¬C₁ ⊢ ∀R₁.(...)) ∈ ℒ(x₀)

L(x₀) ∪ {∀R₁. (¬C₁ ⊢ ∀R₁.(...))}

∀-rule: ∀R₁. (¬C₁ □
∀R₁. (¬C₂ ⊢ ¬C₃)) ∈ ℒ(x₀)

L(x₃) ∪ {¬C₁ ⊢ ∀R₁. (¬C₂ ⊢ ¬C₃)}

backjump

∀-rule: ∀R₃. (¬C₁ ⊢ ¬C₂) ∈ ℒ(x₆)

L(x₇) ∪ {¬C₁ ⊢ ¬C₂}

□-rule: ∀R₂. (≤ 0R₂) □
∀R₁. (¬C₁ ⊢ ∀R₁.(...)) ∈ ℒ(x₀)

L(x₀) ∪ {∀R₁. (¬C₁ ⊢ ∀R₁.(...))}

∀-rule: ∀R₁. (¬C₁ □
∀R₁. (¬C₂ ⊢ ¬C₃)) ∈ ℒ(x₀)

L(x₃) ∪ {¬C₁ ⊢ ∀R₁. (¬C₂ ⊢ ¬C₃)}
Tableau Algorithm: An Example

∀-rule: \( \forall R_1. (\neg C_1 \sqcup \forall R_1. (\neg C_2 \sqcup \neg C_3)) \in \mathcal{L}(x_0) \)
\( \mathcal{L}(x_3) \cup \{\neg C_1 \sqcup \forall R_1. (\neg C_2 \sqcup \neg C_3)\} \)

\( \square\)-rule: \( C_4 \sqcup C_5 \in \mathcal{L}(x_3) \)
\( \mathcal{L}(x_3) \cup \{C_4\} \)
\( \{C_4, \neg C_4\} \subseteq \mathcal{L}(x_3) \)

Clash

\( \square\)-rule: \( \neg C_1 \sqcup \neg C_2 \in \mathcal{L}(x_7) \)
\( \mathcal{L}(x_7) \cup \{\neg C_1\} \)
\( \{C_1, \neg C_1\} \in \mathcal{L}(x_7) \)

Clash

\( \square\)-rule: \( \neg C_1 \sqcup \forall R_1. (\neg C_2 \sqcup \neg C_3) \in \mathcal{L}(x_3) \)
\( \mathcal{L}(x_3) \cup \{\neg C_1\} \)
\( \{C_1, \neg C_1\} \subseteq \mathcal{L}(x_3) \)

Clash

backjump

\( \square\)-rule: \( C_4 \sqcup C_5 \in \mathcal{L}(x_3) \)
\( \mathcal{L}(x_3) \cup \{C_5\} \)

backjump

\( \square\)-rule: \( \neg C_1 \sqcup \neg C_2 \in \mathcal{L}(x_7) \)
\( \mathcal{L}(x_7) \cup \{\neg C_1\} \)
\( \{C_1, \neg C_1\} \in \mathcal{L}(x_7) \)

backjump

\( \square\)-rule: \( \neg C_1 \sqcup \forall R_1. (\neg C_2 \sqcup \neg C_3) \in \mathcal{L}(x_3) \)
\( \mathcal{L}(x_3) \cup \{\forall R_1. (\neg C_2 \sqcup \neg C_3)\} \)

\( \mathcal{L}(x_3) = \{C_3\} \)
\( \mathcal{L}(x_3) = \{C_2, C_3\} \)
\( \mathcal{L}(x_7) = \{C_1\} \)
Tableau Algorithm: An Example

∀-rule: \( \forall R_1. (\neg C_2 \sqcup \neg C_3) \in \mathcal{L}(x_3) \)
\[ \mathcal{L}(x_4) \cup \{\neg C_2 \sqcup \neg C_3\} \]
\[ \mathcal{L}(x_5) \cup \{\neg C_2 \sqcup \neg C_3\} \]

\( \sqcap \)-rule: \( \neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_4) \)
\[ \mathcal{L}(x_4) \cup \{\neg C_2\} \]
\( \{C_2, \neg C_2\} \in \mathcal{L}(x_4) \)

\( \sqcap \)-rule: \( \neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_5) \)
\[ \mathcal{L}(x_5) \cup \{\neg C_2\} \]
\( \{C_2, \neg C_2\} \in \mathcal{L}(x_5) \)

Backjump

\( \sqcap \)-rule: \( \neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_4) \)
\[ \mathcal{L}(x_4) \cup \{\neg C_3\} \]

\( \sqcap \)-rule: \( \neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_5) \)
\[ \mathcal{L}(x_5) \cup \{\neg C_3\} \]
\( \{C_3, \neg C_3\} \in \mathcal{L}(x_5) \)

Backjump

\( \sqcap \)-rule: \( \neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_4) \)
\[ \mathcal{L}(x_4) \cup \{\neg C_3\} \]
\( \{C_3, \neg C_3\} \in \mathcal{L}(x_4) \)

 contradiction is given to the right of the red clash highlight. If the rule action which was completed by the rule is highlighted in purple. If the rule the assertion it is applied to, are highlighted in blue. The transformation / are illustrated in the figure, such that an individual's
Observations

- Many transformation rules do not contribute to the inference problem
- Especially disjunctions

$\forall$-rule: $\forall R_3. (\neg C_1 \sqcup \neg C_2) \in \mathcal{L}(x_6)$

$\Box$-rule: $C_4 \sqcup C_5 \in \mathcal{L}(x_3)$

$\Box$-rule: $\neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_4)$

$\Box$-rule: $\neg C_1 \sqcup \neg C_2 \in \mathcal{L}(x_7)$

- Hence, eliminate rule applications can improve efficiency
  - Completeness trade-off
Mobile Semantic Reasoning:

How?
Optimising mobile reasoning: 3 Strategies

1. Persistent, adaptive and incremental reasoning
2. Predictive models for reasoning performance
3. Logical optimisation
Mobile Semantic Reasoning:

Strategy 1:

Persistent, Adaptive and Incremental Reasoning
Caching Strategy (1)

- Tableaux transformation rules which contribute to a clash (i.e. a positive match) are cached: mTableaux (Steller & Krishnaswamy (2008, 2009))
Caching Strategy (2)

- There may still be transformation rules that don’t contribute to the task

\[ \forall \text{rule: } \forall R_1. (\neg C_2 \sqcup \neg C_3) \in \mathcal{L}(x_3) \]

- Only one of the above needs to generate a clash
  - But both need to be evaluated
  - Remembering previous clashes may help

- Caching Strategy (CS) – avoid re-evaluation of previous assertions
  - Across inference tasks
  - In the service matching setting
  - Time sensitive
  - Sound, incomplete
Caching Strategy (3)

- Caching already employed by existing OWL reasoners
  - Main memory only
- In a mobile setting
  - Similar requests may be made *successively*
  - Limited main memory
  - Substantial *fast* secondary storage (flash-based)
  - Hence, *persistent* caching *across* tasks may improve performance

<table>
<thead>
<tr>
<th>Cache entry</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid</td>
</tr>
<tr>
<td>Matching</td>
<td>Full</td>
</tr>
<tr>
<td></td>
<td>Partial</td>
</tr>
</tbody>
</table>
Adaptive Inference

- Typically current reasoners:
  - Depth-first, arbitrary ordering of user requirements
  - Complete entire inference process before a result is provided: “all or nothing” principle

- **mTableaux** adaptive Inference strategy (Steller & Krishnaswamy (2008, 2009)):
  - Leveraging priority order of matching in reasoning
  - Persistence to support incremental and partial reasoning
  - Resource-adaptation as a control mechanism
  - A degree of match metric
Standard vs Adaptive Inference

User Request: \((\neg \text{WiFi} \lor \neg \text{Internet}) \lor \neg \text{Coffee})\)

**Standard Tableaux:**

- A
- B
- C
- D
- E

\(\neg \text{WiFi} \quad \neg \text{Internet} \quad \neg \text{Coffee}\)

**Adaptive Inference:**

Order: \(\neg \text{Internet}, \neg \text{Coffee}, \neg \text{WiFi}\)

- A
- B
- C
- D (0.3)
- E

\(\neg \text{Internet} \quad \neg \text{WiFi} \quad \neg \text{Coffee}\)
Adaptive Inference - Considerations

- Priority expansion of disjunction elements
  - Queuing and expansion ordering

- Simultaneously open / unfinished branches
  - Branch identifiers
  - State management

- Degree of match computation
Adaptive Inference Strategy
Degree of Match – Product Scenario (after timeout)

Degree of Match: After Time Out Period

Product Service to Match Against
ProductRequest1

ProductServiceH does not match ProductRequest1
Adaptive Inference Strategy
Degree of Match (after timeout) - Product Scenario

Degree of Match After Time Out Period:
to Match ProductRequest2 against ProductServiceA

Stop After Time (seconds)

Degree of Match

Adaptive Inference Strategy
Standard Inference (Theoretical)
Standard Inference (Actual)
No Result
Semantic Reasoners for Embedded Devices

- An ontology reasoner on Programmable Logic Controllers: Grimm (2012)
  - For $\mathcal{EL}^+$: without concept disjointness & nominals (individuals)
  - Used for industrial diagnostics

- Programmable logic controllers:
  - Widely used in industrial automation
  - Are resource-constrained:
    - Fixed-length execution cycles
    - 4MB memory (S7-300, S7-400)

- Reasoning strategies
  - A compact representation for axioms
  - An interruption-safe implementation of completion rules
Optimising mobile reasoning: 3 Strategies

1. Persistent, adaptive and incremental reasoning
2. Predictive models for reasoning performance
3. Logical optimisation
Mobile Semantic Reasoning:

Strategy 2:

Predictive Models for Reasoning Performance
A Need to Understand Reasoning Performance

- Recall, reasoning can be very hard

Kang, Y.-B. et al (2012)
A Need to Predict Reasoning Performance

- Ontologies are different
  - Size doesn’t tell the whole story
  - Interactions of expressions/axioms important source of complexity

- Reasoners are different
  - Different reasoning paradigms & optimisation techniques
  - Drastic performance differences on the same ontology

- Hence,
  - What to do when facing design choices?
  - What to do when efficiency is a hard constraint?
  - How long will a reasoner take?
Performance Prediction: Take 1

- Prediction models for ontology reasoner performance (Kang et al 2012)
  - Given an (ontology, reasoner) pair, predict the performance of the reasoner on the ontology
  - Predict discretised reasoning time: classification
- Use ontology metrics as features:
  - Rationale: ontologies may be difficult for different reasons
  - Metrics capture different aspects of complexity
- Train prediction models on subsets of features
  - Not every feature is equally useful
- Identify key metrics that impact performance the most
Ontology Metrics

- 27 metrics organised into 4 categories
- **ONT**: ontology-level metrics
  - Overall size & complexity
- **CLS**: class-level metrics
  - Complexity about named classes
- **ACE**: anonymous class expressions
  - Complexity of different types of expressions
- **PRO**: property expressions & axioms
  - Complexity related to properties

The values of all metrics are calculated; and the distribution of 8 representative metrics are shown in Figure 1, where the metric values are plotted in log scale and ranked by the values. As can be seen quite clearly, the values for these metrics span a large spectrum, ranging from 0 to more than $10^{5}$, and to more than $10^{7}$ for DIT. Moreover, as expected the majority of ontologies have metric values in the middle of the range, with a few having values closer to the boundary.

Classification time for all ontologies is also collected. All the experiments are performed on a high-performance server running OS Linux 2.6.18 and Java 1.6 on an Intel (R) Xeon X7560 CPU at 2.27GHz with a maximum of 40GB allocated to the 4 reasoner.OWLAPI version 3.2.4 are used to load ontologies and interface with the reasoners. The reasoners that are invoked are: FaCT++, HermiT 1.3.5, Pellet 2.3.0 and TrOWL 0.8. REL is the underlying reasoner used by TrOWL. These metrics will be revisited in Section 7.
Prediction Model Construction (1)

1. Data collection
   - 350+ ontologies of various sizes & difficulty levels
   - 4 reasoners
   - Collect metric values for all ontologies
   - Collect reasoning performance for all (ontology, reasoner) pairs

2. Reasoning time discretisation

<table>
<thead>
<tr>
<th>Label</th>
<th>Classification time</th>
<th>FaCT++</th>
<th>HermiT</th>
<th>Pellet</th>
<th>TrOWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.01s &lt; A ≤ 1s</td>
<td>75</td>
<td>154</td>
<td>126</td>
<td>105</td>
</tr>
<tr>
<td>B</td>
<td>1s &lt; B ≤10s</td>
<td>16</td>
<td>35</td>
<td>38</td>
<td>17</td>
</tr>
<tr>
<td>C</td>
<td>10s &lt; C ≤ 100s</td>
<td>6</td>
<td>12</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>D</td>
<td>100s &lt; D</td>
<td>11</td>
<td>13</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>269</td>
<td>291</td>
<td>330</td>
<td>337</td>
</tr>
</tbody>
</table>
Prediction Model Construction (2)

1. Feature selection
   - Not all features are equally useful
   - Apply 6 ranking-based feature selection algorithms

2. Model training
   - Not all models are equally accurate
   - Train many models, not only one: 9 models (Bayesian, decision tree, rules, etc.)
   - Prediction performance measure by accuracy
   - 10-fold cross validation to evaluate prediction performance
Prediction Model Evaluation (1)

- Model accuracy

![Accuracy Comparison Graphs](image-url)
Prediction Model Evaluation (2)

- Average prediction accuracy for all models exceed 80%
- Metrics make good features for reasoning performance prediction – worst accuracy is still close to 80% (Naïve Bayes)
- Random forest the most accurate
- Random forest the most stable
Key Metrics Identification

- What makes reasoning hard?
- Identify key metrics that impact reasoning performance the most
- Calculation: combining metric frequency & weight in prediction models
Performance Prediction: Take 2

- Prediction models for ontology reasoner performance
  - Instead *discretised* reasoning time, prediction *actual* reasoning time
  - *Regression analysis*

- Use *ontology metrics* as features:
  - Same 4 categories: ONT, CLS, ACE & PRO
  - Significantly expanded metrics set: 91 metrics
  - Capture complexity more comprehensively

- Training: a regression model for each reasoner

- Application: identify performance hotspots
Prediction Model Construction

1. Data collection
   - 450+ ontologies
   - 6 OWL reasoners: FaCT++, HermiT, JFact, Pellet & TrOWL

2. Data preprocessing
   - Cleansing, normalisation, metrics removal, splitting

3. Regression model building
   - Training a random forest model for each reasoner
Prediction Model Evaluation

- Prediction performance metrics:
  - Coefficient of determination $R^2$: the higher the better
  - Root mean square error RMSE: the lower the better
- The models are highly accurate

<table>
<thead>
<tr>
<th>Model</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R^2$</td>
<td>RMSE</td>
</tr>
<tr>
<td>FaCT++</td>
<td>0.853</td>
<td>1.24</td>
</tr>
<tr>
<td>HermiT</td>
<td>0.868</td>
<td>1.13</td>
</tr>
<tr>
<td>JFact</td>
<td>0.834</td>
<td>1.37</td>
</tr>
<tr>
<td>MORe</td>
<td>0.882</td>
<td>0.81</td>
</tr>
<tr>
<td>Pellet</td>
<td>0.835</td>
<td>1.15</td>
</tr>
<tr>
<td>TrOWL</td>
<td>0.942</td>
<td>0.89</td>
</tr>
</tbody>
</table>
Application: Performance Hotspot Identification

- Hotspots (Gonçalves, Parsia, and Sattler 2012) represent performance bottlenecks
  - A small subset of axioms (generated from a *signature*) that significantly impact performance, i.e., for a subset $M$ of ontology $O$, $M$ is a hotspot if $\#M \ll \#O$ and $RT(O \setminus M, R) \ll RT(O, R)$
    - $\#$ represents the size, $RT$ represents reasoning time on ontology $O$ by reasoner $R$
  - Their identification provides insights for ontology design & maintenance

- The current identification algorithm (Gonçalves, Parsia, and Sattler 2012) requires exhaustive concept satisfiability checking: can be *expensive*
Regression-based Hotspot Identification: Algorithm

- We apply the regression model to this problem
- Given ontology $O$, reasoner $R$, number of candidates $k$

1. For each concept $C$ in ontology $O$
   1. Adaptively generate candidate $M_C$ from $C$
   2. Predict reasoning time $t_{M_C}$ of $M_C$ using the prediction model for $R$
2. Rank all candidates $C$ by $t_{M_C}$ in descending order
3. Return the top $k$ candidates
Regression-based Hotspot Identification: Dataset

- Dataset: 8 large bio-ontologies known to contain hotspots (Gonçalves, Parsia, and Sattler 2012)

<table>
<thead>
<tr>
<th>Ontology</th>
<th># axioms</th>
<th># classes</th>
<th>Reasoner</th>
<th>RT(O, R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChEBI</td>
<td>60,085</td>
<td>28,869</td>
<td>Pellet</td>
<td>174.1</td>
</tr>
<tr>
<td>EFO</td>
<td>7,493</td>
<td>4,143</td>
<td>HermiT</td>
<td>78.8</td>
</tr>
<tr>
<td>IMGT</td>
<td>1,122</td>
<td>112</td>
<td>HermiT</td>
<td>122.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MORe</td>
<td>108.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pellet</td>
<td>&gt; 3,600</td>
</tr>
<tr>
<td>VO</td>
<td>8,488</td>
<td>3,530</td>
<td>HermiT</td>
<td>65.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pellet</td>
<td>&gt; 3,600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TrOWL</td>
<td>115.31</td>
</tr>
<tr>
<td>NCIt</td>
<td>116,587</td>
<td>83,722</td>
<td>HermiT</td>
<td>1,020.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MORe</td>
<td>643.2</td>
</tr>
</tbody>
</table>
Regression-based Hotspot Identification: Results

- Evaluation criteria: no. of hotspots identified & no. of tests required

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Reasoner</th>
<th># hotspots</th>
<th># tests</th>
<th>% avg # hotspot</th>
<th>% avg RT boost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChEBI</td>
<td>Pellet</td>
<td>10</td>
<td>10</td>
<td>0.5%</td>
<td>96%</td>
</tr>
<tr>
<td>EFO</td>
<td>HermiT</td>
<td>10</td>
<td>16</td>
<td>1.2%</td>
<td>75.5%</td>
</tr>
<tr>
<td>IMGT</td>
<td>HermiT</td>
<td>3</td>
<td>14</td>
<td>6.8%</td>
<td>79.4%</td>
</tr>
<tr>
<td></td>
<td>MORe</td>
<td>10</td>
<td>16</td>
<td>8.7%</td>
<td>76.6%</td>
</tr>
<tr>
<td></td>
<td>Pellet</td>
<td>2</td>
<td>14</td>
<td>6.8%</td>
<td>&gt; 99.98%</td>
</tr>
<tr>
<td>VO</td>
<td>HermiT</td>
<td>10</td>
<td>10</td>
<td>4.6%</td>
<td>98.5%</td>
</tr>
<tr>
<td></td>
<td>Pellet</td>
<td>10</td>
<td>10</td>
<td>4.5%</td>
<td>&gt; 98.1%</td>
</tr>
<tr>
<td></td>
<td>TrOWL</td>
<td>10</td>
<td>10</td>
<td>4.5%</td>
<td>97.8%</td>
</tr>
<tr>
<td>NCIt</td>
<td>HermiT</td>
<td>10</td>
<td>10</td>
<td>3.3%</td>
<td>90.8%</td>
</tr>
<tr>
<td></td>
<td>MORe</td>
<td>10</td>
<td>10</td>
<td>2.5%</td>
<td>88.7%</td>
</tr>
</tbody>
</table>
Regression-based Hotspot Identification: Observations

- Hotspot identification
  - Our method identifies more hotspots using fewer tests for most \((O, R)\) pairs than (Gonçalves, Parsia, and Sattler 2012)
  - Generating candidates is fast (polynomial to \# O)
  - Making prediction is really fast
  - Most pairs are identified with very few tests (less than 20)

- Reasoning performance
  - Different reasoners may have dramatically different performance on the same ontology
  - Reinforces the need for understanding & predicting performance
Optimising mobile reasoning: 3 Strategies

1. Persistent, adaptive and incremental reasoning
2. Predictive models for reasoning performance
3. Logical optimisation
Mobile Semantic Reasoning:

Strategy 3:

Logic-level Optimisation and Tuning
**KRHyper**

- **KRHyper**: Kleemann and Sinner (2006)
  - Novel Tableaux reasoner for First Order Logic (FOL) for deployment on resource constrained devices
  - It implements the standard Tableaux optimisation strategies of backjumping, semantic branching, Boolean constraint propagation, lazy unfolding and absorption used by today’s commercial and open source reasoners.

- **Performance**
  - Better performance than RacerPro for small test cases, not performing as well for larger tests.
  - Still exhausts all memory when the reasoning task becomes too large for a small device to handle and fails to provide any result.
Semantic Reasoners for Mobile Devices

- Gu et al. (2007)
  - Framework which provides an RDF/OWL parser, reasoner and sRDQL query engine for information matching
  - Example: a shopping assistance application which uses a user’s context to provide suggestions to the user about products which may suit their needs based on previous usage patterns.
  - Runs on the user’s mobile device using J2ME
- This framework provides acceptable performance by supporting OWL Lite
- The reasoner in the framework uses a forward chaining approach which supports rule based reasoning
Mini-ME

- Mini-ME: Ruta et al. (2008a,b,2012)
  - supports distance based, ranked, matching of requests to services using a DL mobile reasoner implemented in Java 2 Micro Edition (J2ME)
  - supports short range (bluetooth) ad-hoc networks
  - Achieves acceptable performance by restricting the OWL-DL language to a subset
  - Ontology structure is constrained so that it can be reduced to set comparisons for reducing computational overhead
  - Mini-ME: an $\text{ALCN}$ reasoner Android (Ruta et al. 2012)
  - Improved memory footprint compared to previous version
**Delta-Reasoner**

- Delta-Reasoner: Motik, Horrocks & Kim (2012)
  - A reasoner to support context-awareness
  - Incremental OWL 2 RL reasoning
  - Continuous conjunctive query answering
- Performance evaluated on a laptop
  - 3 ontologies
  - Axioms added & removed

### Table 1: Test Ontologies

<table>
<thead>
<tr>
<th></th>
<th>TBox axioms</th>
<th>Class assertions</th>
<th>Property assertions</th>
<th>DL expressivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICODI</td>
<td>223</td>
<td>33,238</td>
<td>82,943</td>
<td>$\textit{ALHI}(\text{D})$</td>
</tr>
<tr>
<td>SEMINTEC</td>
<td>219</td>
<td>17,941</td>
<td>47,299</td>
<td>$\textit{ALHIF}$</td>
</tr>
<tr>
<td>LUBM</td>
<td>93</td>
<td>18,128</td>
<td>82,415</td>
<td>$\textit{ALEHT}^+(\text{D})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Loading</th>
<th>Initial reasoning</th>
<th>Incremental reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>VICODI</td>
<td>2127</td>
<td>2820</td>
<td>156</td>
</tr>
<tr>
<td>SEMINTEC</td>
<td>1048</td>
<td>1123</td>
<td>157</td>
</tr>
<tr>
<td>LUBM</td>
<td>2597</td>
<td>818</td>
<td>135</td>
</tr>
</tbody>
</table>
Optimisation Techniques in mTableaux: At a Glance

- mTableaux: a framework for light-weight mobile DL inference (Steller & Krishnaswamy (2008, 2009))
  - In a service matching setting

- Focus: class membership checking for a single individual $C(a) \in A$
  - Matching a request against a service description

- New optimisation strategies
  - Selective application of transformation rules (ST)
  - Selective application of the disjunction rule (SD)

- Caveat: realisation/consistency checking not performed
  - Ontology assumed to be consistent
  - Hence no completeness guarantee
    - False negatives possible, false positives not
Selective Transformation Rule Application (ST)

- Tableaux transformation (expansion) rules are applied to only a subset of individuals, which relate to the service description being checked
  - E.g., a user searches for an Internet café that sells both Internet and coffee $\exists \text{sells.Coffee} \cap \exists \text{sells.Internet}$
  - That a venue hasPlayground is not of interest
- More formally, given an ABox A, tableaux transformation (expansion) rules are only applied to a subset of individuals ST include
  - the individual of interest $x$,
  - iteratively, other individuals that are connected to individuals in ST through properties in universally qualified class expressions, and their super properties, and
  - nothing else.
Selective Disjunction Rule Application (SD)

- The disjunction rule (|-rule) is non-deterministic, and expensive
  - It generates new ABoxes, all of which need to be explored
  - Hence increases search space
- Aim of SD: reduce the no. of |-rule applications
- Tableaux disjunction expansion rule is only applied to disjunctions which contain class concepts which relate to the user request
  - Hence reducing search space
- E.g., looking for Internet café? \( \exists \text{sells.} \text{Internet} \sqcap \exists \text{sells.} \text{Coffee} \)
  - Apply |-rule to Tea \( \sqcap \) Coffee, not to Pizza \( \sqcap \) VideoGame
Evaluation of mTableaux Optimisation: Comparison with Other Reasoners

![Graph comparing reasoners on Galen ontology](image-url)
Evaluation of mTableaux Optimisation: Combinations of Techniques

- Evaluated on resource-constrained mobile devices

<table>
<thead>
<tr>
<th>Technique</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caching (CSa)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caching (CSb)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation

Product Case Study: Positive Match (Processing Time)

- CSa
- CSb
- ST
- SD

Out of Mem
300+ Secs

Time (seconds)

Test: Combination of mTableaux strategies enabled

ST = Selective Trans. Rules, SD = Selective Disjunctions, CS = Caching Strategy
Evaluation

Test: Combination of mTableaux strategies enabled

ST = Selective Trans. Rules, SD = Selective Disjunctions, CS = Caching Strategy

Same result for both fully and partially cached requests
Summary of Evaluation

- mTableaux reduces the size of the inference problem by reducing the number of expansions performed by the reasoner.
- mTableaux successfully enables the completion of a matching task on a small, resource constrained device without exceeding available memory.
- mTableaux significantly improves the response time to perform matching when the optimisation and caching strategies were enabled compared to when these were not enabled. In fact without the strategies enabled the task could not be completed.
- The combination of adaptive, caching and logical optimisation techniques provides the fastest / most efficient response-time.
Related Work & Recent Development
## ELK on Android

- Direct porting of ELK on Android 4.2: Kazakov & Klinov (2013)
  - ELK supports OWL 2 EL reasoning
- Evaluation
  - 5 OWL 2 EL ontologies: ChEBI, EMAP, Fly Anatomy, Gene Ontology & EL-GALEN
  - *Acceptable* performance
  - Still magnitudes slower than on desktop computers

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Workers</th>
<th>Google Nexus 4</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Load./Index.</td>
<td>Classif.</td>
</tr>
<tr>
<td>ChEBI</td>
<td>1</td>
<td>31,370</td>
<td>207,020</td>
</tr>
<tr>
<td>ChEBI</td>
<td>2</td>
<td>29,423</td>
<td>160,334</td>
</tr>
<tr>
<td>ChEBI</td>
<td>3</td>
<td>32,213</td>
<td>148,369</td>
</tr>
<tr>
<td>ChEBI</td>
<td>4</td>
<td>32,443</td>
<td>147,868</td>
</tr>
<tr>
<td>ChEBI</td>
<td>5</td>
<td>32,900</td>
<td>114,054</td>
</tr>
<tr>
<td>ChEBI</td>
<td>6</td>
<td>29,997</td>
<td>107,033</td>
</tr>
</tbody>
</table>
What about Other Reasoners?

- Comparison analysis of modern reasoners on Android: Yus et al. (2013)
  - JFaCT, CB, HermiT, Pellet

<table>
<thead>
<tr>
<th></th>
<th>JFact</th>
<th>CB</th>
<th>HermiT</th>
<th>Pellet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pizza</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>0.37</td>
<td>0</td>
<td>0.57</td>
<td>0.97</td>
</tr>
<tr>
<td>Android1</td>
<td>4.90</td>
<td>0</td>
<td>14.88</td>
<td>33.22</td>
</tr>
<tr>
<td>Android2</td>
<td>3.42</td>
<td>0</td>
<td>10.43</td>
<td>20.77</td>
</tr>
<tr>
<td>Wine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>10.39</td>
<td>0</td>
<td>6.54</td>
<td>2.22</td>
</tr>
<tr>
<td>Android1</td>
<td>2196.05</td>
<td>0</td>
<td>511.97</td>
<td>194.12</td>
</tr>
<tr>
<td>Android2</td>
<td>1609.32</td>
<td>0</td>
<td>361.38</td>
<td>131.80</td>
</tr>
<tr>
<td>DBpedia</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>UDT!</td>
<td>0</td>
<td>0.10</td>
<td>1.39</td>
</tr>
<tr>
<td>Android1</td>
<td>UDT!</td>
<td>0</td>
<td>8.87</td>
<td>115.30</td>
</tr>
<tr>
<td>Android2</td>
<td>UDT!</td>
<td>0</td>
<td>5.13</td>
<td>63.15</td>
</tr>
<tr>
<td>GO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>7.77</td>
<td>0.11</td>
<td>1.56</td>
<td>1.96</td>
</tr>
<tr>
<td>Android1</td>
<td>OOM!</td>
<td>1.95</td>
<td>OOM!</td>
<td>OOM!</td>
</tr>
<tr>
<td>Android2</td>
<td>435.60</td>
<td>1.47</td>
<td>487.98</td>
<td>83.97</td>
</tr>
<tr>
<td>NCI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>2.61</td>
<td>0.24</td>
<td>2.23</td>
<td>4.24</td>
</tr>
<tr>
<td>Android1</td>
<td>OOM!</td>
<td>3.31</td>
<td>OOM!</td>
<td>OOM!</td>
</tr>
<tr>
<td>Android2</td>
<td>OOM!</td>
<td>2.69</td>
<td>2020.48</td>
<td>OOM!</td>
</tr>
</tbody>
</table>
Mobile Semantic Reasoning:

What’s Next?
Future Directions

- Meta-reasoning
  - Given an ontology, find a reasoner that will be the most efficient
  - Making use of prediction models: work in progress

- Hybrid reasoning – mobile + cloud
  - Determine the most appropriate platform for a reasoning job
  - Task decomposition

- Comparative evaluation of reasoners and reasoning strategies for resource utilisation
  - Adaptation to other constraints – resources, confidence
  - Understand and optimise for reasoning impact on CPU, RAM, power consumption
Thank you!

Questions?
References


References


References


References


References


References


References


References


For your reading:

Service Matching in Mobile Environments
Service Matching in Mobile Environments

- Keyword / String-based Matching Onboard a Mobile Device
- Semantic Matching on a Remote Server
- Hybrid
- Semantic Matching Onboard a Mobile Device
Keyboard/Interface Matching on Mobile Device

- Large body of existing research

- Simple matching techniques – deployed on-board the mobile device

- Can operate in a P2P manner as well (e.g. UPnP)

- Techniques:
  - String matching of keywords, Interface matching (WSDL), Extra-functional properties based (e.g QoS), convert service descriptions into a 128-bit integer using a hash function and compare these integers for service matching, location matching etc.
Keyboard/Interface Matching on Mobile Device

- **Advantages:**
  - No remote infrastructure is required
  - Matching process occurs on-board the mobile devices
  - Uses the information gathered from other devices within network range

- **Disadvantage:**
  - Do not support semantic reasoning
    - Less accurate than semantic techniques
Semantic Matching on Remote Servers

- Large body of existing research
- Key Examples:
  - CoBrA (Chen et al., 2004) was one of the early middleware architectures which utilises semantics and context to reason about users
  - Task Computing Project (TCP) (Masuoka et al., 2003) utiliseS OWL-S for modelling services
  - Integrated Global Pervasive Computing Framework (IGPCF) offers web service discovery using semantics on the web but assumes that pervasive users are permanently connected to the Internet.
Semantic Matching on Remote Servers

- Luo et al. (2005, 2006) adds OWL-S descriptions to the UDDI service registry which performs inferences when a service description is published to it.

- DReggie Chakraborty et al. (2001) extends Jini to support semantic matching using Prolog for reasoning.

- LARKS (Sycara et al., 2002) is designed to match service descriptions with requests, using its own semantic description language.

- The CMU Matchmaker (Srinivasan et al., 2005) provides inference based reasoning to compare OWL-S service profiles with requests, which can be stored in a back-end UDDI registry.
Semantic Matching on Remote Servers

- Bener et al. (2009) proposes a matchmaking algorithm which also takes preconditions and effects into consideration using SWRL.

- Stuckenschmidt and Kolb (2008) defines a reasoning approach which supports partial matching of services against requests where there is insufficient time to complete the full matching process. However, this is achieved by reducing the number of conditions in the request. Reasoning is with Pellet.

- Semantic Web Engineering - Environment and Tools (SWE-ET) (Brambilla et al., 2006) combines the CEFREIEL Glue42 discovery engine with the WebRatio framework to support WSMO Semantic Web Service discovery.
Semantic Matching on Remote Servers

- The Internet Reasoning Service (IRS)-III 46 (Domingue et al., 2008) is a Semantic Web Service broker and reasoning environment which is again based on WSMO but has
  - The added functionality of importing OWL ontologies.

- DIANE (Kuster and König-Ries, 2008) is an environment for automated service discovery and matching which uses its own service profile language to describe a service as a set of effects. It supports a subset of logic without any rules or quantifiers and provides “fuzzy” matching of conditions in the user request against the service description, to provide ranked service results.
Semantic Matching on Remote Servers – Support Mobile Clients

- Broens et al. (2004); and Doulkeridis and Vazirgiannis (2008) utilise semi-OWL and RDF documents to express service
- Baousis et al. (2008); de Andrade et al. (2007); and Chen et al. (2006), support matching of mobile services, but rely on the CMU matchmaker
- Jeon et al. (2008) matches semantically described personal preferences using OWL and SWRL roles on an external server.
- Suraci et al. (2007); and Srirama et al. (2007) support
  - OWL-S service matching on a high-end node.
- Bianchini et al. (2006) provides UDDI based matchmaking of semantically described services in a mobile environment based on location and device capabilities
Semantic Matching on Remote Servers – Support Mobile Clients

- Veijalainen et al. (2006) performs mobile semantic service matching on laptops which are not resource constrained.
- Wang and Hu (2008) is a P2P semantic OWL-S matching architecture which attempts to reduce the number of inference checks required.
- Niazi and Mahmoud (2009); Wolowski et al. (2007); Zoric et al. (2007a); El-Sayed and Black (2006); Almeida et al. (2006); and Sycara et al. (2002) matches services described in OWL using the Jena.
- Peng et al. (2008) delegates OWL service matching to a resource capable machine and uses RacerPro.
- AIDAS (Toninelli et al., 2008) performs matching of mobile user preferences and device capabilities for services using Pellet.
Semantic Matching on Remote Servers – Support Mobile Clients

- Gaia (Ranganathan and Campbell, 2003) is a semantically driven context mobile middleware which performs matching utilising the FaCT++ reasoner.

- Patel and Chaudhary (2009); De and Moessner (2008); and Wei et al. (2008) support semantic queries and matching by making use of Jess First Order Logic (FOL) reasoner.

- Agostini et al. (2007); Mokhtar et al. (2008); and Bouillet et al. (2008) perform all inferences offline on a high-performance server, before matching is later completed on-board a resource constrained device
  - Hybrid Approach
Semantic Matching on Remote Servers – Support Mobile Clients

- All of the semantically driven approaches operate on a high-performance machine
  - require such a machine to perform pre-processing before the matching occurs.
- Reasoners Used: Jena inference prover, Jess, RacerPro, Fact++, Pellet, Prolog or Lisp
- The HermiT (Motik et al., 2009) reasoner has been developed to provide more optimised Tableaux semantic DL reasoning using OWL
  - HermiT has been developed for the desktop / server environment
- Very few perform semantic reasoning/inference on resource-constrained mobile devices
Semantic Matching on-board Mobile Devices

- Chakraborty et al. (2006); Nedos et al. (2006)
  - match services based on semantic service types defined in a hierarchy
  - However these approaches use explicit subclass relations only (such as OWL Lite)
  - Do not support semantic reasoning and inference proof
  - Current open source / commercial reasoners such as Pellet, KAON2, FaCT++, RacerPro considerably resource intensive
  - Cannot function on small resource constrained devices