Android goes Semantic: DL Reasoners on Smartphones

Fernando Bobillo, fbobillo@unizar.es

Department of Computer Science and Systems Engineering
University of Zaragoza, Spain

with Roberto Yus, Carlos Bobed, Guillermo Esteban and Eduardo Mena

ORE 2013
Ulm (Germany)
July 2013

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Outline

Outline

Motivation

- Objective: enable **mobile devices with semantic reasoning** capabilities
- **Local reasoners** are needed to be able to manage knowledge even when network disconnections make impossible to rely on other devices/computer
- Michele Ruta et al. implemented a mobile $\mathcal{ALCN}$ reasoner from scratch
- We try to avoid these rewritings and investigate how to **adapt existing DL reasoners to** work on **Android devices**

Android and Java

- **Android**: popular operative system for smartphones and tablets (about 64 % of the share)
- Most semantic reasoners are implemented in **Java**
- Android libraries, and APIs are written in C, but it supports Java code as it uses a Java-like virtual machine (**Dalvik**)
- Dalvik runs **dex-code** and Java bytecodes can be converted to Dalvik-compatible .dex files
- However, **Dalvik does not align to Java SE** and so it does not support Java ME classes, AWT or Swing
- Running semantic APIs and reasoners on Android could require some **rewriting efforts**!

Outline

Running Semantic APIs on Android

- **OWL API 3.4.3** could be converted to Dalvik without **any modifications** and so, imported into an Android project directly.
- **OWL API 2.2.0** (automatically imported by Pellet along with the OWL API 3.2.4) uses Java classes that are **not supported** by Dalvik
- **Jena** cannot be directly imported into an Android project, but there is a project called **Androjena** to port it to the Android platform

Running Reasoners on Android: Success

- **JFact** is a port of FaCT++ (written in C++): it **works fine**
- **CB** (in OCaml): can be compiled to native Android code and **run using** the tool **adb** (Android Debug Bridge)
- **HermiT cannot be converted directly** to Dalvik
  - References to unsupported Java classes
  - Runtime error of Dalvik when unmarshalling the objects serialized by *dk.brics.automaton*

  Our solutions
  - Remove the **debug** package of HermiT and its references
  - Reimplement some methods of *JAutomata* class
  - Reimplement the marshalling/unmarshalling methods

Running Reasoners on Android: Success

- **Pellet cannot be converted directly** to Dalvik as it uses 3 libraries that reference unsupported Java classes:
  - **Jena**: can be replaced by Androjena
  - **OWL API 2.2.0**: can be removed
  - **JAXB**: can be fixed
    - remove the JAXB .jar file
    - add to our Android project the source code of the *java.xml.bind* package and the *Xerces* library
    - Due to the limit of 65536 methods references per .dex file, add **only the 9 classes** that Pellet needs

Running Reasoners on Android: Fail

- **RacerPro, KAON2, QUEST, TrOWL, fuzzyDL**: reference unsupported Java classes
  - They cannot be directly converted and **their source code** was not publicly available for modifications
  - We noticed that they use some problematic libraries:
    - **Jena** (QUEST, TrOWL): could be replaced e.g. **Androjena**
    - **Java RMI** (KAON2): could be replaced e.g. **LipeRMI**
    - **Xerces** (QUEST): **could be solved** as in Pellet
    - **Gurobi** (fuzzyDL): **no replacement**

Outline

Experiments

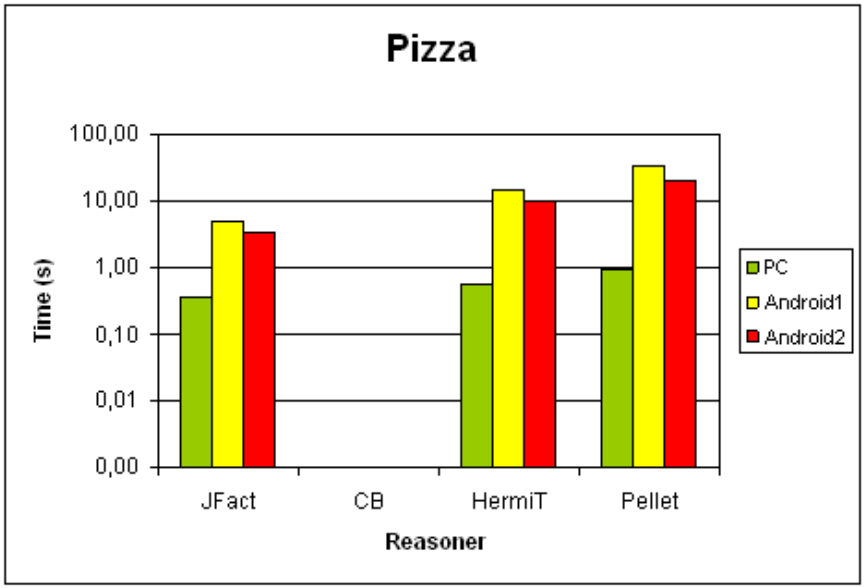
- Very preliminary evaluation of the performance
- Task: **classification of 5 ontologies**
  - Pizza, Wine, DBpedia, GO, NCI
- 3 devices:
  - **PC**: Windows 64-bits, i5-2320 3.00GHz, 16GB RAM
  - **Android1**: Samsung Galaxy Tab, 1.0GHz, 512MB RAM, Android 2.3.3
  - **Android2**: Galaxy Nexus, 1.2GHz dual-core, 1GB RAM, Android 4.2.1

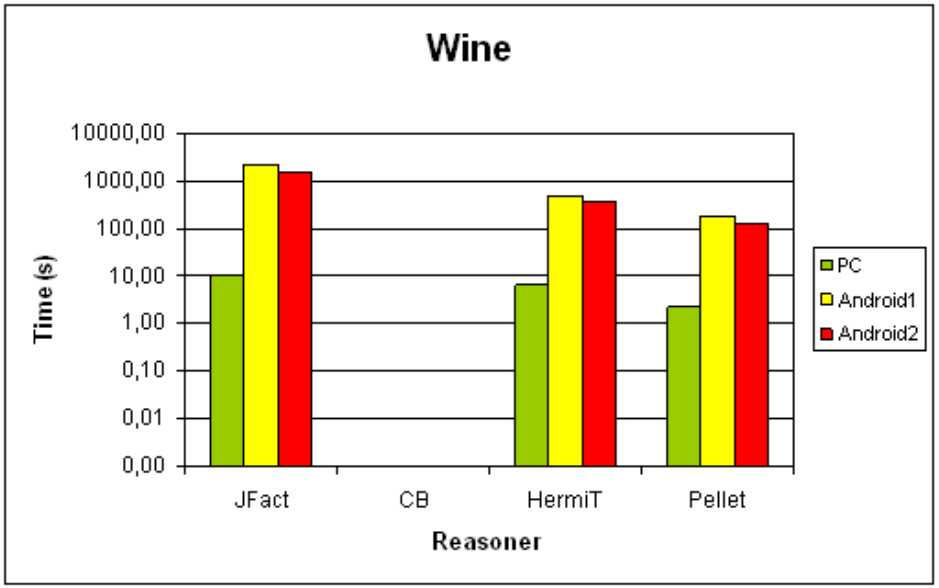
Classification time in PC and Android

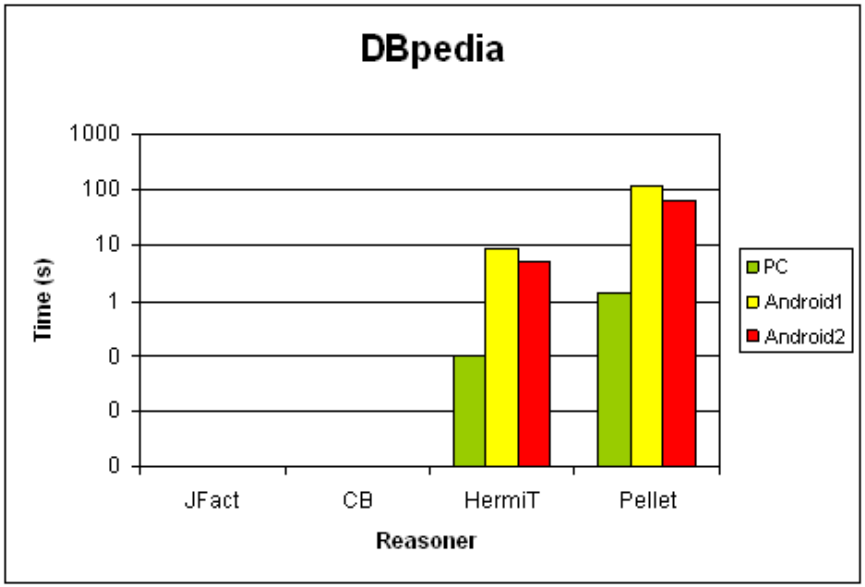
		JFact	CB	HermiT	Pellet
Pizza	PC	0.37	≈0◊	0.57	0.97
	Android1	4.90	≈0◊	14.88	33.22
	Android2	3.42	≈0◊	10.43	20.77
Wine	PC	10.39	≈0◊	6.54	2.22
	Android1	2196.05	≈0◊	511.97	194.12
	Android2	1609.32	≈0◊	361.38	131.80
DBpedia	PC	**UDT!**	≈0	0.10	1.39
	Android1	**UDT!**	≈0	8.87	115.30
	Android2	**UDT!**	≈0	5.13	63.15
GO	PC	7.77	0.11	1.56	1.96
	Android1	**OOM!**	1.95	**OOM!**	**OOM!**
	Android2	435.60	1.47	487.98	83.97
NCI	PC	2.61	0.24	2.23	4.24
	Android1	**OOM!**	3.31	**OOM!**	**OOM!**
	Android2	**OOM!**	2.69	2020.48	**OOM!**

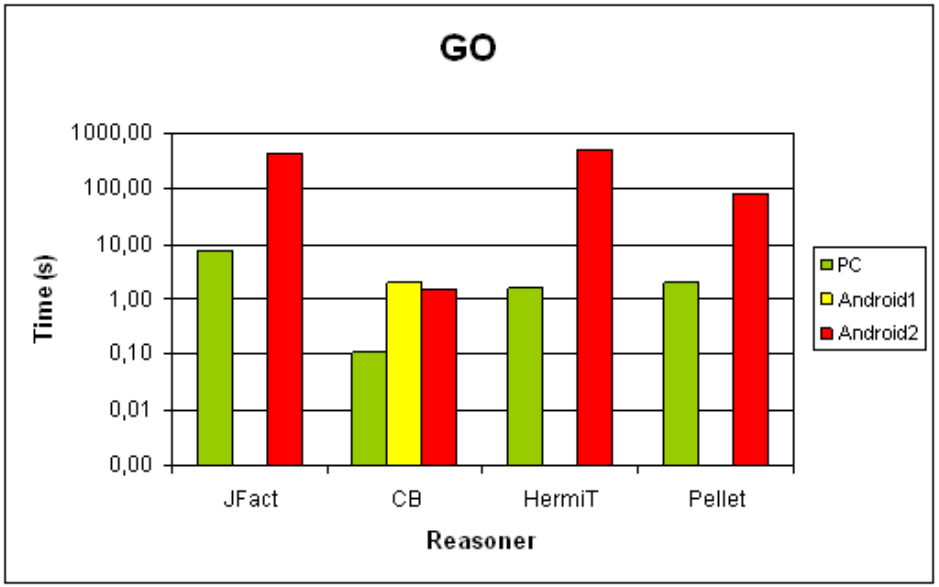
- ◊: incomplete reasoning
- **OOM!**: Out of Memory
- **UDT!**: Unsupported Data Type

Classification time in PC and Android









Outline

Conclusions and future work

- We have shown that current Android devices could be able to use most of the semantic reasoners although some **manual work** is usually needed
- The **relative reasoning times** are **similar in mobile and non-mobile devices**
- **Main limitation** on current smartphones/tablets: memory usage and processing requirements
- **Possible trend**: **performance improved a 30%** when updating from a 2010 Samsung Galaxy Tab to a 2011 Google Galaxy Nexus
- **Future work**: test current semantic reasoners on Android measuring other important aspects e.g. **memory and battery usage**

Comments?

Thank you very much for your attention