A Framework for Easing the Development of Applications Embedding Answer Set Programming

Davide Fuscà **Stefano Germano** Jessica Zangari
Marco Anastasio Francesco Calimeri Simona Perri

18th International Symposium on Principles and Practice of Declarative Programming

Department of Mathematics and Computer Science, University of Calabria, Italy
embasp@mat.unical.it

Outline

Introduction

Answer Set Programming (ASP)

The Framework

- Abstract Architecture
- Implementing EMBASP

Embedding ASP Programs

ASP-based Applications

Related Work

Conclusions

Introduction

Motivations

- *Declarative* and *Imperative* languages integration
- *Answer Set Programming (ASP)* is mature for practical applications and it is used all around the world
- Ease the development of *ASP-based applications*, in both educational and real-world contexts
- Separation of Concerns (or Levels of Analysis)

- ICT industry is moving towards the mobile scenario
- Lack of works about ASP systems natively running on *mobile devices*

Contributions

- EMBASP: an abstract framework for the integration of ASP in external systems for generic applications
- An actual Java implementation of the framework with specialized libraries for two state-of-the-art ASP systems
- Some fully functional applications developed in the educational context

Freely available at

`https://www.mat.unical.it/calimeri/projects/embasp/`

Answer Set Programming (ASP)

ASP - Introduction

A purely *declarative* AI formalism for *Knowledge Representation and Reasoning* developed in the field of *Logic Programming and Nonmonotonic Reasoning*

- language based on *rules*, allowing for both *disjunction in rule heads* and *nonmonotonic negation in the body*
- use *logic program* to represent a given computational problem
- an *answer set solver* is used to find the *models*, called *answer sets*, which correspond one-to-one to solutions of the computational problem

ASP - Syntax I

As in the ASP-Core-2 standard [CFG+12]

- A *term* is a variable or a constant
- An *atom* is $a(t_1, \ldots, t_n)$, where
    - a is a *predicate* of arity n
    - $t_1, \ldots, t_n$ are *terms*
- A *literal* is either
    - *positive literal* p
    - or a *negative literal* `not` p

  where p is an *atom*.

ASP - Syntax II

A (*disjunctive*) *rule* r is of the form

$$a_1 \mid \cdots \mid a_n \text{ :- } b_1, \cdots, b_k, \texttt{ not } b_{k+1}, \cdots, \texttt{ not } b_m.$$

where:

- $a_1, \cdots, a_n, b_1, \cdots, b_m$ are atoms and $n \geq 0$, $m \geq k \geq 0$
- $a_1 \mid \cdots \mid a_n$ is the *head* of r
- $b_1, ..., b_k, \texttt{ not } b_{k+1}, ..., \texttt{ not } b_m$ is the *body* of r
- If the *head* is empty (i.e. $n = 0$), it is called an *integrity constraint*
- If the *body* is empty (i.e. $k = m = 0$), it is called a *fact*
- $H(r)$ denotes the set $\{a_1, ..., a_n\}$ of the head atoms
- $B(r)$ the set $\{b_1, ..., b_k, \texttt{not } b_{k+1}, \ldots, \texttt{not } b_m\}$ of the body literals
- $B^+(r)$ (resp., $B^-(r)$) denotes the set of atoms occurring positively (resp., negatively) in $B(r)$
- A rule r is *safe* if each variable appearing in r appears also in $B^+(r)$

One of the most common ASP programming methodology is the “Guess&Check” (GC) paradigm [EFLP00]

- a **Guessing Part**, that defines the search space (for instance, by means of disjunctive rules)
- a **Checking Part** (optional), that checks solution admissibility (usually, by means of *integrity constraints*)

One of the most common ASP programming methodology is the "Guess&Check" (GC) paradigm [EFLP00]

- a **Guessing Part**, that defines the search space (for instance, by means of disjunctive rules)
- a **Checking Part** (optional), that checks solution admissibility (usually, by means of *integrity constraints*)

That can be further extended to match the "Guess/Check/Optimize" (GCO) paradigm [BLR97]

- **Optimizing Part** (optional), that specifies preference criteria (usually, by means of *weak constraints* [BLR97, CFG+12])

ASP example - SUDOKU - Input

A set of facts F is given representing the *schema* to be completed:

- a binary predicate pos encoding possible position coordinates;
- a unary predicate $symbol$ encoding possible symbols (numbers);
- facts of the form $sameblock(x1, y1, x2, y2)$ state that two positions $(x1, y1)$ and $(x2, y2)$ are within the same block;
- facts of the form $cell(x, y, n)$ represent that a position (x, y) is filled with symbol n.

ASP example - SUDOKU - logic program

An ASP program P_{sudoku} such that the answer sets of $P_{sudoku} \cup F$ correspond to the solutions of the Sudoku *schema* at hand:

$$r_1: \quad cell(X, Y, N) \mid nocell(X, Y, N) \text{ :- } pos(X), pos(Y), symbol(N).$$

$$r_2: \quad \text{:- } cell(X, Y, N), cell(X, Y, N1), N1 <> N.$$

$$r_3: \quad assigned(X, Y) \text{ :- } cell(X, Y, N).$$

$$r_4: \quad \text{:- } pos(X), pos(Y), not\ assigned(X, Y).$$

$$r_5: \quad \text{:- } cell(X, Y1, Z), cell(X, Y2, Z), Y1 <> Y2.$$

$$r_6: \quad \text{:- } cell(X1, Y, Z), cell(X2, Y, Z), X1 <> X2.$$

$$r_7: \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), Y1 <> Y2, sameblock(X1, Y1, X2, Y2).$$

$$r_8: \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), X1 <> X2, sameblock(X1, Y1, X2, Y2).$$

ASP example - SUDOKU - logic program

An ASP program P_{sudoku} such that the answer sets of $P_{sudoku} \cup F$ correspond to the solutions of the Sudoku *schema* at hand:

$$r_1 : \quad cell(X, Y, N) \mid nocell(X, Y, N) \text{ :- } pos(X), pos(Y), symbol(N).$$

$$r_2 : \quad \text{:- } cell(X, Y, N), cell(X, Y, N1), N1 <> N.$$

$$r_3 : \quad assigned(X, Y) \text{ :- } cell(X, Y, N).$$

$$r_4 : \quad \text{:- } pos(X), pos(Y), not\ assigned(X, Y).$$

$$r_5 : \quad \text{:- } cell(X, Y1, Z), cell(X, Y2, Z), Y1 <> Y2.$$

$$r_6 : \quad \text{:- } cell(X1, Y, Z), cell(X2, Y, Z), X1 <> X2.$$

$$r_7 : \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), Y1 <> Y2, sameblock(X1, Y1, X2, Y2).$$

$$r_8 : \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), X1 <> X2, sameblock(X1, Y1, X2, Y2).$$

ASP example - SUDOKU - logic program

An ASP program P_{sudoku} such that the answer sets of $P_{sudoku} \cup F$ correspond to the solutions of the Sudoku *schema* at hand:

$$r_1: \quad cell(X, Y, N) \mid nocell(X, Y, N) \text{ :- } pos(X), pos(Y), symbol(N).$$

$$r_2: \quad \text{:- } cell(X, Y, N), cell(X, Y, N1), N1 <> N.$$

$$r_3: \quad assigned(X, Y) \text{ :- } cell(X, Y, N).$$

$$r_4: \quad \text{:- } pos(X), pos(Y), not\ assigned(X, Y).$$

$$r_5: \quad \text{:- } cell(X, Y1, Z), cell(X, Y2, Z), Y1 <> Y2.$$

$$r_6: \quad \text{:- } cell(X1, Y, Z), cell(X2, Y, Z), X1 <> X2.$$

$$r_7: \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), Y1 <> Y2, sameblock(X1, Y1, X2, Y2).$$

$$r_8: \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), X1 <> X2, sameblock(X1, Y1, X2, Y2).$$

An ASP program P_{sudoku} such that the answer sets of $P_{sudoku} \cup F$ correspond to the solutions of the Sudoku *schema* at hand:

$$r_1: \quad cell(X, Y, N) \mid nocell(X, Y, N) \text{ :- } pos(X), pos(Y), symbol(N).$$

$$r_2: \quad \text{:- } cell(X, Y, N), cell(X, Y, N1), N1 <> N.$$
$$r_3: \quad assigned(X, Y) \text{ :- } cell(X, Y, N).$$
$$r_4: \quad \text{:- } pos(X), pos(Y), not\ assigned(X, Y).$$

$$r_5: \quad \text{:- } cell(X, Y1, Z), cell(X, Y2, Z), Y1 <> Y2.$$
$$r_6: \quad \text{:- } cell(X1, Y, Z), cell(X2, Y, Z), X1 <> X2.$$

$$r_7: \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), Y1 <> Y2, sameblock(X1, Y1, X2, Y2).$$
$$r_8: \quad \text{:- } cell(X1, Y1, Z), cell(X2, Y2, Z), X1 <> X2, sameblock(X1, Y1, X2, Y2).$$

ASP example - SUDOKU - logic program

An ASP program P_{sudoku} such that the answer sets of $P_{sudoku} \cup F$ correspond to the solutions of the Sudoku *schema* at hand:

$$r_1 : \quad cell(X, Y, N) \mid nocell(X, Y, N) :- \ pos(X), pos(Y), symbol(N).$$

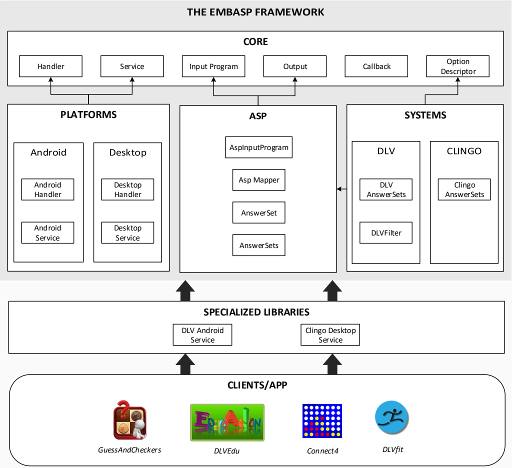
$$r_2 : \quad :- \ cell(X, Y, N), cell(X, Y, N1), N1 <> N.$$
$$r_3 : \quad assigned(X, Y) :- \ cell(X, Y, N).$$
$$r_4 : \quad :- \ pos(X), pos(Y), not\ assigned(X, Y).$$

$$r_5 : \quad :- \ cell(X, Y1, Z), cell(X, Y2, Z), Y1 <> Y2.$$
$$r_6 : \quad :- \ cell(X1, Y, Z), cell(X2, Y, Z), X1 <> X2.$$

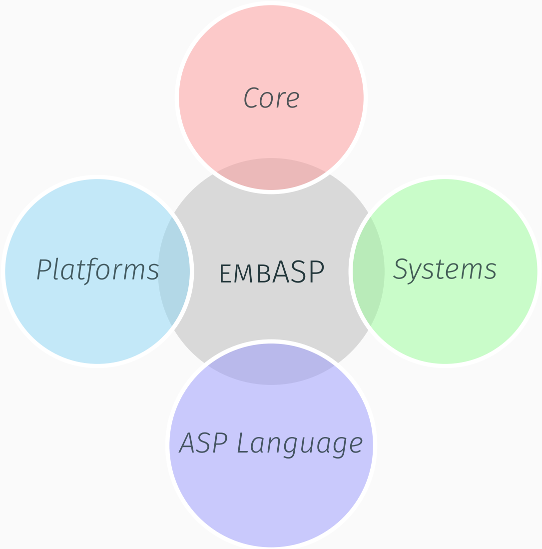
$$r_7 : \quad :- \ cell(X1, Y1, Z), cell(X2, Y2, Z), Y1 <> Y2, sameblock(X1, Y1, X2, Y2).$$
$$r_8 : \quad :- \ cell(X1, Y1, Z), cell(X2, Y2, Z), X1 <> X2, sameblock(X1, Y1, X2, Y2).$$

The Framework

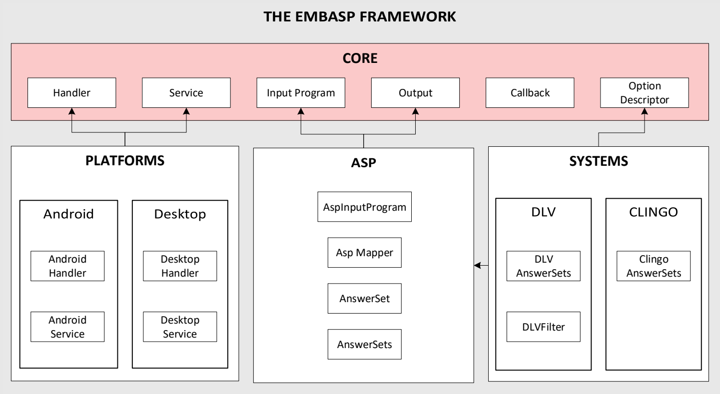
EMBASP - A visual overview



EMBASP Abstract Architecture

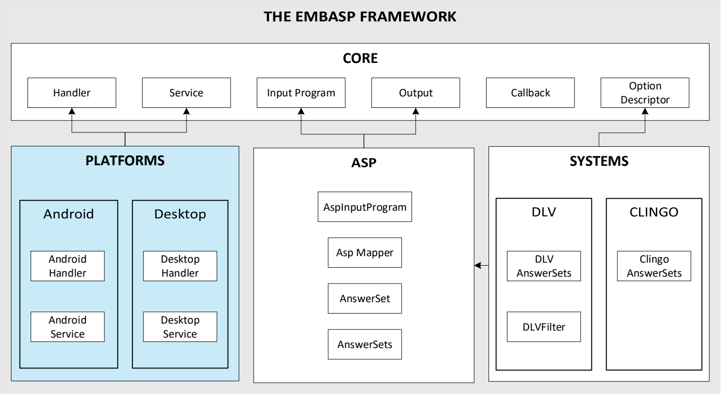


EMBASP - *Core*

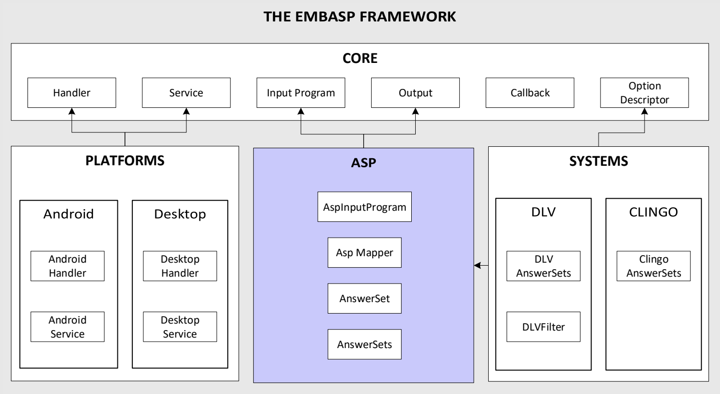


Defines the basic components of the *Framework*

EMBASP - *Platforms*

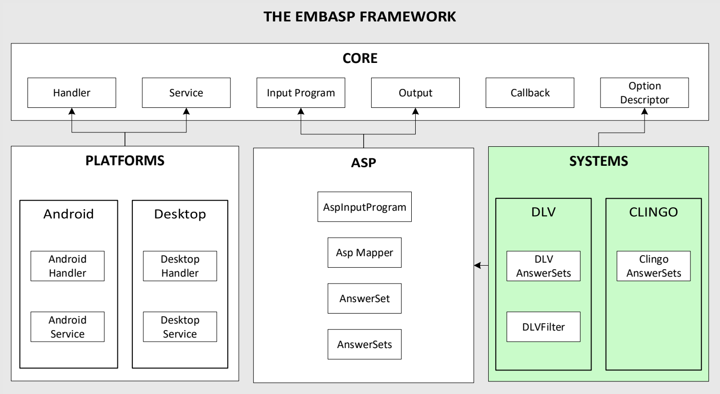


Contains what is platform-dependent



Defines specific facilities for ASP

EMBASP - *Systems*

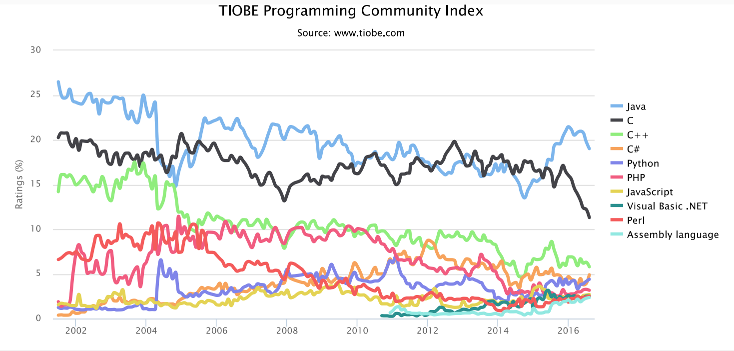


Defines what is system-dependent

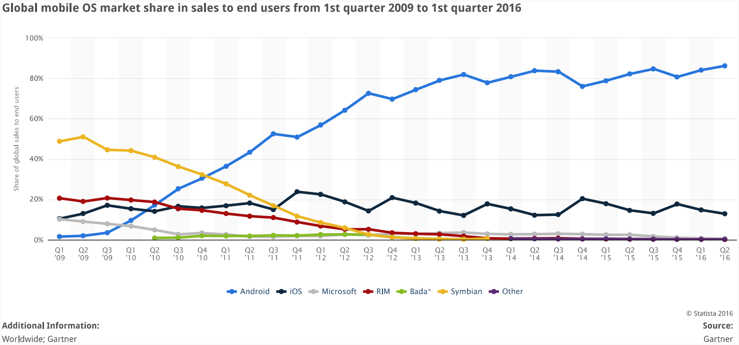
Implementing EMBASP

- Java implementation of the Framework
- Specializations for two of the state-of-the-art ASP systems

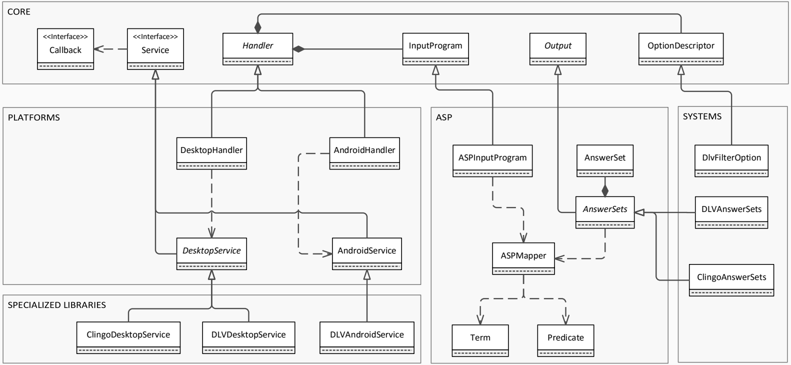
Why Java? I


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
Java
C
C++
C#
Python
PHP
JavaScript
Visual Basic .NET
Perl
Assembly language

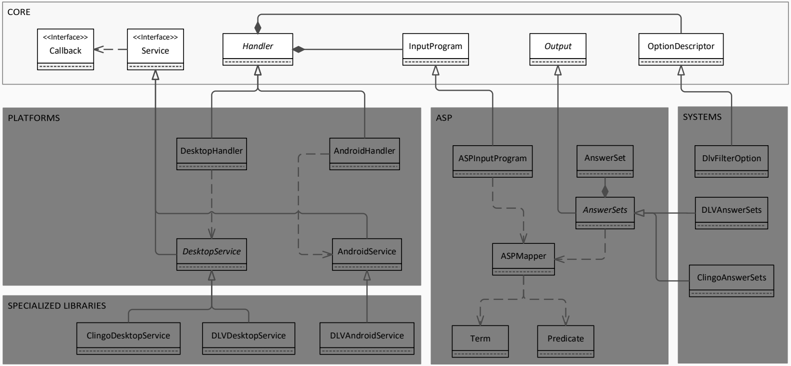
Why Java? II


Global mobile OS market share in sales to end users from 1st quarter 2009 to 1st quarter 2016
100%
80%
60%
40%
20%
0%
Share of global sales to end users
Q1 '09
Q2 '09
Q3 '09
Q1 '10
Q2 '10
Q3 '10
Q4 '10
Q1 '11
Q2 '11
Q3 '11
Q4 '11
Q1 '12
Q2 '12
Q3 '12
Q4 '12
Q1 '13
Q2 '13
Q3 '13
Q4 '13
Q1 '14
Q2 '14
Q3 '14
Q4 '14
Q1 '15
Q2 '15
Q3 '15
Q4 '15
Q1 '16
Q2 '16
Android
iOS
Microsoft
RIM
Bada*
Symbian
Other
© Statista 2016
Additional Information:
Worldwide; Gartner
Source:
Gartner

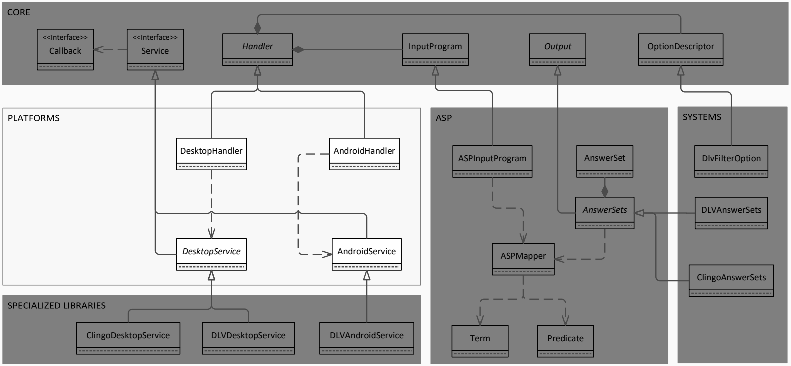
Architecture



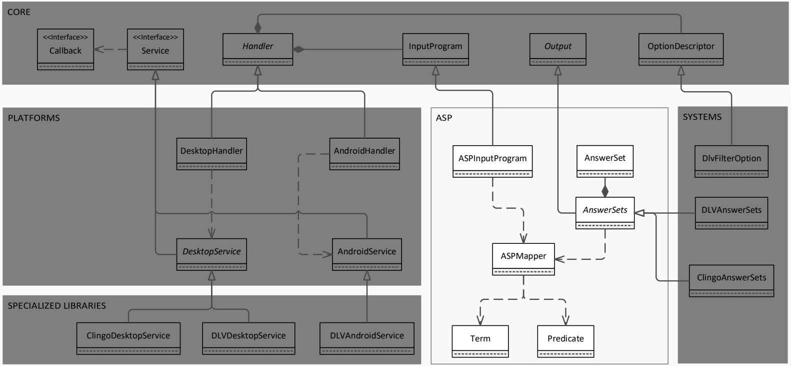
EMBASP - *Core*


CORE
<<Interface>>
Callback
<<Interface>>
Service
Handler
InputProgram
Output
OptionDescriptor
PLATFORMS
DesktopHandler
AndroidHandler
DesktopService
AndroidService
SPECIALIZED LIBRARIES
ClingoDesktopService
DLVDesktopService
DLVAndroidService
ASP
ASPInputProgram
AnswerSet
AnswerSets
ASPMapper
Term
Predicate
SYSTEMS
DlvFilterOption
DLVAnswerSets
ClingoAnswerSets

EMBASP - *Platforms*



EMBASP - *ASP Language*



The `ASPMapper`

Two-way "translator" between strings recognizable by the ASP solver at hand and Java objects directly employable within the application

- Guided by the following Java Annotations:

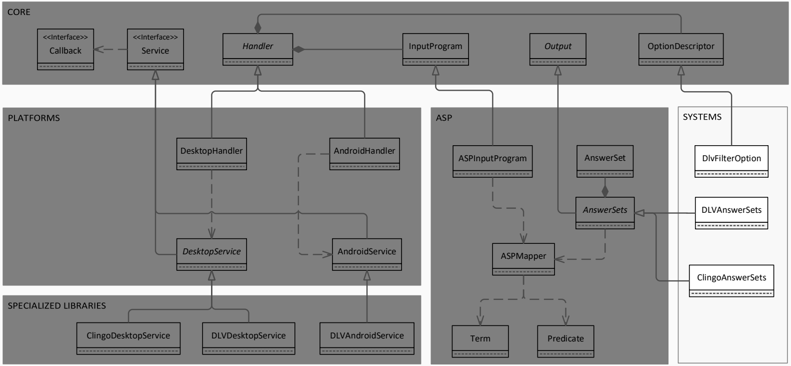
  @Predicate (string_name)

  Defines the predicate name a class is mapped to

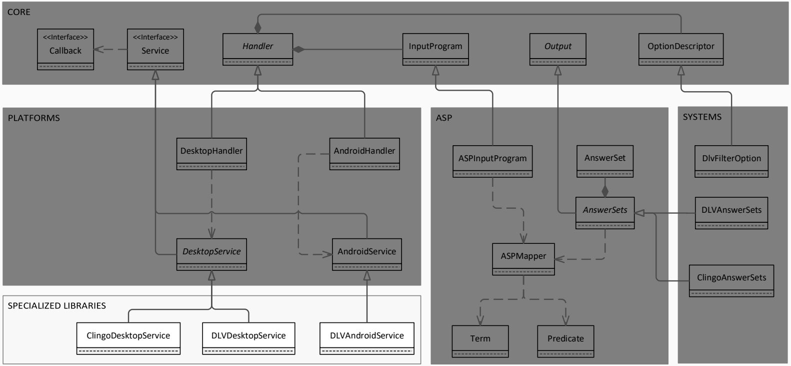
  @Term (integer_position)

  Defines the term (and its position) in the ASP atom the field is mapped to
- Uses the Java Reflection mechanisms to examine the Annotation at run-time and perform the translation
- Give developers the possibility to work separately on the ASP-based modules and on the Java side

EMBASP - *Systems*



EMBASP - *Specialized Libraries*



Specializing the Framework - The Android case

JNI (Java Native Interface) [Ora] and *Android NDK (Native Development Kit)* [Gooa]

- The use of *JNI* grants the access to the API provided by the *Android NDK*, and to the exposed DLV functionalities directly from the Java code of an Android application
- The *NDK* allows developers to implement parts of an Android application as “native-code” languages, such as *C* and *C++*
- These technologies represent the general and standard way to realize the porting of a *C++* software in an Android context

Embedding ASP Programs

How to use EMBASP to build an app

Build an (Android) app for solving Sudoku puzzles using EMBASP

- We have a proper logic program to solve a sudoku puzzle
- We have also an initial schema

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Full code available at:

https://www.mat.unical.it/calimeri/projects/embasp/

How to use EMBASP to build an app - The Mapping

The class **`Cell`**

```
@Predicate("cell")
public class Cell {

  @Term(1)
  private int row;

  @Term(2)
  private int column;

  @Term(3)
  private int value;

  [...]

}
```

Thanks to the *annotations* the **`ASPMapper`** will be able to map **`Cell`** objects into strings properly recognizable from the ASP solver as *logic facts* of the form:

$$cell(Row, Column, Value)$$

How to use EMBASP to build an app - The Activity I

```
public class MainActivity extends AppCompatActivity {
  [...]

  private Handler handler;

  @Override
  protected void onCreate(Bundle bundle) {
    handler = new AndroidHandler(getApplicationContext(),
        DLVAndroidService.class);
    [...]
  }

  public void onClick(final View view){
    [...]
    startReasoning();
  }
[...]
```

How to use EMBASP to build an app - The Activity II

```
[...]
  public void startReasoning() {

    InputProgram inputProgram = new ASPInputProgram();
    for (int i = 0; i < 9; i++)
      for (int j = 0; j < 9; j++)
        try {
          if(sudokuMatrix[i][j] != 0)
            inputProgram.addObjectInput(new Cell(i, j,
                sudokuMatrix[i][j]));
        } catch (Exception e) { // Handle Exception }
    handler.addProgram(inputProgram);

    String sudokuEncoding = getEncodingFromResources();
    handler.addProgram(new ASPInputProgram(sudokuEncoding));

    Callback callback = new MyCallback();
    handler.startAsync(callback);
}}
```

How to use EMBASP to build an app - The Callback

```
private class MyCallback implements Callback {
  @Override
  public void callback(Output o) {
    if(!(o instanceof AnswerSets)) return;

    AnswerSets answerSets = (AnswerSets)o;
    if(answerSets.getAnswersets().isEmpty()) return;

    AnswerSet as = answerSets.getAnswersets().get(0);
    try {
      for(Object obj : as.getAtoms()) {
        Cell cell = (Cell) obj;
        sudokuMatrix[cell.getRow()][cell.getColumn()] = cell.
            getValue();
      }
    } catch (Exception e) { // Handle Exception }

    displaySolution();
}}
```

Other Language Implementations of EMBASP

The *abstract architecture* of EMBASP can be made concrete by means of other *object-oriented* programming languages

- It uses features that are typical of any object-oriented language, such as *inheritance* and *polymorphism*
- The unique exception is the *ASPMapper* component which uses *annotations* and *reflection*
  - Some languages have similar constructs
  - In other these constructs can be simulated applying typical *Software Engineering patterns* [GHJV94]

ASP-based Applications: some Examples in the Educational Setting

Some Examples in the Educational Setting

ASP-based applications developed by means of EMBASP for educational purposes, and, in particular, in the context of a university course that covers ASP topics

- Engagement of university undergraduate students in ASP capabilities
- ASP looks well-fitted for the use in the development of educational/training software

GuessAndCheckers

A native mobile application that works as an helper for users that play “live” games of the (Italian) checkers (i.e., by means of physical board and pieces)

A native mobile application that works as an helper for users that play “live” games of the (Italian) checkers (i.e., by means of physical board and pieces)

- by means of the device camera a picture of the board is taken
- the information about the current status of the game is properly inferred thanks to the *OpenCV* library
- an ASP-based artificial intelligence module then suggests the move

DLVEdu

An educational Android App for children, that is able to guide the child throughout the learning tasks, by proposing a series of educational games

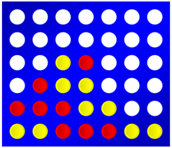
DLVEdu

An educational Android App for children, that is able to guide the child throughout the learning tasks, by proposing a series of educational games

- dynamically builds and updates a customized educational path along the different games
- uses well-known mobile technologies, such as voice or drawn text recognition
- features a “Parent Area”, that allows parents to monitor child’s achievements and to express some preferences

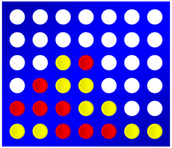
Connect4

An Android application that allows a user to play the game against an ASP-based artificial player



Connect4

An Android application that allows a user to play the game against an ASP-based artificial player



- different AIs designed and implemented
    - from the most powerful one (with advanced techniques for the perfect play)
    - to the simplest one (with some classical heuristic strategies)
- using EMBASP, two different versions of the same app have been built:
    - one for Android, making use of DLV
    - one for Java-enabled desktop platforms, making use of clingo.

DLVfit

A health app that aims at suggesting the owner of a mobile device the “best” way to achieve some fitness goals

DLVfit

A health app that aims at suggesting the owner of a mobile device the “best” way to achieve some fitness goals

- goals and preferences about habits and activities can be expressed in a customizable way
- using the Google Activity Recognition APIs [Goob], the app, in the background, constantly detects the current user activity
- at any time, the user might ask for a suggestion about a workout plan for the rest of the day

Advantages of the approach used in these apps

- Wide range of customization possibilities thanks to the modelling capabilities and the declarative nature of ASP
- Flexibility and possibility to build the ASP program(s) at runtime and to customize the modules ease the developer's job of make the app comply to the user's desiderata

Related Work

Related Work

Clingo4 [GKKS14]

- Enables a form of control over the computational tasks of the embedded ASP solver *Clingo* with scripting languages *lua* and *python*
- The main purpose is the support of dynamic and incremental reasoning

Java Wrapper [Ric03]

- Acts like a versatile wrapper wherewith the Java developers can interact with the ASP solver (DLV)
  - Differently, EMBASP makes use of Java Annotations, allowing an easy mapping of input/output to Java Objects

Related Work

JDLV [FGLR12]

- Based on JASP, an hybrid language that allows a bilateral interaction between ASP and Java
- Uses JPA annotations to define how Java classes map to relations, similarly to ORM frameworks
  - Differently, EMBASP exploits custom annotations, almost effortless to define, in order to deal with the mapping

Moreover, EMBASP is not specifically bound to a single or specific solver and it can be easily extended to deal with any solver, and with different solvers at the same time.

Related Work

Tweety [Thi14]

- A set of Java libraries that allow to make use of several knowledge representation systems supporting different logic formalisms
- The use is very similar to EMBASP, both provide libraries to incorporate proper calls to external declarative systems from within “traditional” applications
- Tweety implementation is very rich, covering a wide range of KR formalisms, yet looking less general
  - Differently, EMBASP is mainly focused on fostering the use of ASP in the widest range of contexts and supports the mobile setting

Conclusions

Conclusions

- A general framework for embedding the reasoning capabilities of ASP into external systems
- The fully abstract architecture makes the framework general enough to be adapted to a wide range of scenarios
- Actual Java implementation and two specialized libraries for embedding *DLV* on Android applications and *clingo* on any Java-based desktop application are provided
- Has been tested within some university courses featuring ASP topics, for implementing a set of applications, ranging from AI-based games to educative apps

The framework, documentation, an application showcase and further details are freely available at:

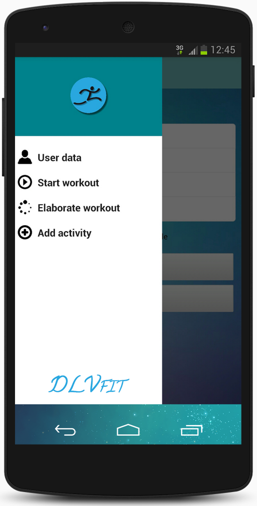
`https://www.mat.unical.it/calimeri/projects/embasp/`

Questions?

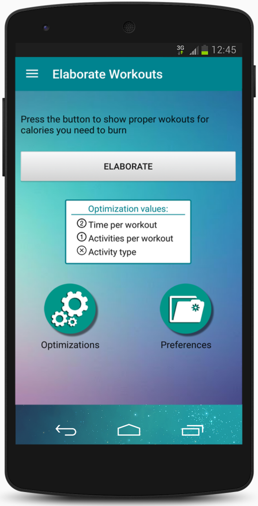
Questions?

Thank you for your attention.

DLVFIT



12:45
User data
Start workout
Elaborate workout
Add activity
DLVFIT



12:45
Elaborate Workouts
Press the button to show proper wokouts for calories you need to burn
ELABORATE
Optimization values:
2 Time per workout
1 Activities per workout
Activity type
Optimizations
Preferences

DLVfit - ASP Reasoning Module

- The app *dynamically* builds a suitable ASP program whose answer sets represent workout plans that comply with the *very personal goals* and *preferences* previously expressed
- A classic *Guess/Check/Optimize* paradigm is used:

Guess Compute how much time should be spent on each exercise

Check Find only admissible workout plans

Optimize Try to satisfy the user's preferences to the largest possible extent

DLVFIT- ASP program: relevants concepts

`calories_burnt_per_activity(A, C)`

the calories burnt (C), in each unit of time, per each Activity (A)

DLVFIT- ASP program: relevants concepts

`calories_burnt_per_activity(A, C)`

the calories burnt (C), in each unit of time, per each Activity (A)

`remaining_calories_to_burn(R)`

the calories that remain to burnt in the current day

DLVFIT- ASP program: relevants concepts

`calories_burnt_per_activity(A, C)`

the calories burnt (C), in each unit of time, per each Activity (A)

`remaining_calories_to_burn(R)`

the calories that remain to burnt in the current day

`how_long(A, D)`

the amount of the time that can be spent for each activity

DLVFIT- ASP program: relevants concepts

`calories_burnt_per_activity(A, C)`

the calories burnt (C), in each unit of time, per each Activity (A)

`remaining_calories_to_burn(R)`

the calories that remain to burnt in the current day

`how_long(A, D)`

the amount of the time that can be spent for each activity

`max_time(T)`

the duration of the workout

DLVfit- ASP program: relevants concepts

`calories_burnt_per_activity(A, C)`

the calories burnt (C), in each unit of time, per each Activity (A)

`remaining_calories_to_burn(R)`

the calories that remain to burnt in the current day

`how_long(A, D)`

the amount of the time that can be spent for each activity

`max_time(T)`

the duration of the workout

`surplus(C)`

the maximum surplus of calories to burn of the suggested workouts

DLVfit- ASP program: relevants concepts

`calories_burnt_per_activity(A, C)`

the calories burnt (C), in each unit of time, per each Activity (A)

`remaining_calories_to_burn(R)`

the calories that remain to burnt in the current day

`how_long(A, D)`

the amount of the time that can be spent for each activity

`max_time(T)`

the duration of the workout

`surplus(C)`

the maximum surplus of calories to burn of the suggested workouts

`optimize(T, W, P)`

the specific optimization operation(s) that the user wants to perform

DLVfit - An example of Input I (Basic Concepts)

```
calories_burnt_per_activity("ON_BICYCLE", 5).
calories_burnt_per_activity("WALKING", 2).
calories_burnt_per_activity("RUNNING", 11).

remaining_calories_to_burn(200).

how_long("ON_BICYCLE", 10).
how_long("ON_BICYCLE", 20).
how_long("WALKING", 10).
how_long("WALKING", 20).
how_long("RUNNING", 10).
how_long("RUNNING", 20).

max_time(20).

surplus(100).
```

DLVfit - An example of Input II (Custom Optimizations)

```
optimize("RUNNING", 1, 3).
optimize("WALKING", 2, 3).
optimize("ON_BICYCLE", 3, 3).
```

maximize the number of favourite activities to perform

DLVfit - An example of Input II (Custom Optimizations)

```
optimize("RUNNING", 1, 3).
optimize("WALKING", 2, 3).
optimize("ON_BICYCLE", 3, 3).
```

maximize the number of favourite activities to perform

```
optimize(time,0,2).
```

minimize total time spent exercising

DLVfit - An example of Input II (Custom Optimizations)

```
optimize("RUNNING", 1, 3).
optimize("WALKING", 2, 3).
optimize("ON_BICYCLE", 3, 3).
```

maximize the number of favourite activities to perform

```
optimize(time,0,2).
```

minimize total time spent exercising

```
optimize(activities, 0, 1).
```

minimize total number of activities to perform

DLVFIT

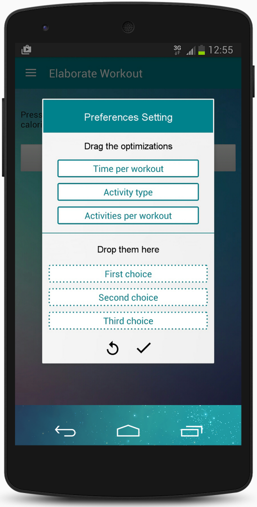


Figure 1: Expressing priorities

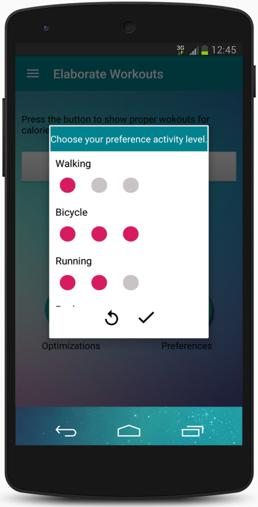


Figure 2: Expressing preferences

DLVfit - Logic rules composing the ASP program

```
activity_to_do(A, HL) | not_activity_to_do(A, HL) :- how_long(A, HL).
```

DLVfit - Logic rules composing the ASP program

```
activity_to_do(A, HL) | not_activity_to_do(A, HL) :- how_long(A, HL).
```

```
:- activity_to_do(A, HL1), activity_to_do(A, HL2), HL1 != HL2.

:- remaining_calories_to_burn(RC), total_calories_activity_to_do(CB),
   RC > CB.
:- remaining_calories_to_burn(RC), total_calories_activity_to_do(CB),
   CB > RCsurplus, RCsurplus = RC + surplus.

:- max_time(MTS), total_time_activity_to_do(TS), MTS < TS.
```

DLVfit - Logic rules composing the ASP program

```
activity_to_do(A, HL) | not_activity_to_do(A, HL) :- how_long(A, HL).
```

```
:- activity_to_do(A, HL1), activity_to_do(A, HL2), HL1 != HL2.

:- remaining_calories_to_burn(RC), total_calories_activity_to_do(CB),
   RC > CB.
:- remaining_calories_to_burn(RC), total_calories_activity_to_do(CB),
   CB > RCsurplus, RCsurplus = RC + surplus.

:- max_time(MTS), total_time_activity_to_do(TS), MTS < TS.
```

```
:~ optimize(A, W, P), activity_to_do(A, _). [W:P]

:~ optimize(time, _, P), activity_to_do(_, HL). [HL:P]

:~ optimize(activities, _, P), #int(HM),
        HM = #count{A, HL : activity_to_do(A, HL)}. [HM:P]
```

References I

Francesco Buccafurri, Nicola Leone, and Pasquale Rullo.
Strong and Weak Constraints in Disjunctive Datalog.
In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'97)*, volume 1265 of *Lecture Notes in AI (LNAI)*, pages 2–17, Dagstuhl, Germany, July 1997. Springer Verlag.

Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Francesco Ricca, and Torsten Schaub.
Asp-core-2: Input language format, 2012.

Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer.
Declarative problem-solving using the dlv system.
In *Logic-based artificial intelligence*, pages 79–103. Springer, 2000.

References II

Onofrio Febbraro, Giovanni Grasso, Nicola Leone, and Francesco Ricca.
JASP: a framework for integrating Answer Set Programming with Java.
In *Proc. of KR2012*. AAAI Press, 2012.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.
Design patterns: elements of, 1994.

M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub.
***Clingo* = ASP + control: Preliminary report.**
In M. Leuschel and T. Schrijvers, editors, *Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14)*, volume arXiv:1405.3694v1, 2014. Theory and Practice of Logic Programming, Online Supplement.

Google.
Android NDK.
`https://developer.android.com/ndk/index.html`.

References III

Google.

Google Activity Recognition API.
https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognition.

Oracle.

JNI.
http://docs.oracle.com/javase/8/docs/technotes/guides/jni.

Francesco Ricca.

The DLV Java Wrapper.
In Marina de Vos and Alessandro Provetti, editors, *Proceedings ASP03 - Answer Set Programming: Advances in Theory and Implementation*, pages 305–316, Messina, Italy, September 2003. Online at http://CEUR-WS.org/Vol-78/.

References IV

Matthias Thimm.
Tweety: A comprehensive collection of java libraries for logical aspects of artificial intelligence and knowledge representation.
In *KR*, 2014.