Datalog Development Tools

O. Febbraro\textsuperscript{2}, G. Grasso\textsuperscript{3}, N. Leone\textsuperscript{1}, K. Reale\textsuperscript{1} and Francesco Ricca\textsuperscript{1}

\textsuperscript{1} Dipartimento di Matematica, Università della Calabria - 87036 Rende, Italy
\textsuperscript{2} DLVSystem s.r.l. - P.zza Vermicelli, Polo Tecnologico, 87036 Rende, Italy
\textsuperscript{3} Department of Computer Science, Oxford University - Parks Road, Oxford, UK

Acknowledgments. This work has been partially supported by the Calabrian Region under PIA project DLVSYSTEM approved in BURC n. 20 parte III del 15/05/2009 - DR n. 7373 del 06/05/2009.

Datalog 2.0 2012
Outline

1. Introduction
2. Language Overview
3. ASPIDE: Integrated Development Environment for ASP
4. JASP: Integrating ASP with Java
5. Conclusion and Related Work
Context (1)

- Disjunctive Datalog under the stable model semantics
  
  - Answer Set Programming (ASP)
  - Declarative programming paradigm
  - Non-monotonic reasoning and logic programming

- Idea:
  1. Logic programs represent computational problems
  2. Answer sets correspond to solutions
  3. Use a solver to find solutions

[Gelfond-Lifschitz ’91]
Context (2)

After more than 20 years:

1. **ASP formal properties are well-understood**
   → **ASP is expressive... beyond NP** [Eiter et.al ’97]

2. **Robust and efficient implementations**
   - DLV [Leone et.al ’06], Clasp [Gebser et.al ’07],
   - CModels [Lierler ’05], IDP[Wittocx et al.], etc.
   - *continuous improvement witnessed by ASP competitions*

3. **Applications in several fields**
   - Artificial Intelligence, Knowledge Representation & Reas.,
   - Information Integration, Data cleaning, Bioinformatics, ...
   - Recently employed for developing industrial applications
     → Workforce Management [Ricca et al. 2010-tplp]
     → E-Tourism [Ricca et al. 2010-fi]
Context (2)

After more than 20 years:

1. ASP formal properties are well-understood → ASP is expressive... beyond NP [Eiter et.al ’97]
2. Robust and efficient implementations
   - DLV [Leone et.al ’06], Clasp [Gebser et.al ’07],
   - CModels [Lierler ’05], IDP[Wittocx et al.], etc.
   - continuous improvement witnessed by ASP competitions
3. Applications in several fields
   - Artificial Intelligence, Knowledge Representation & Reas.,
   - Information Integration, Data cleaning, Bioinformatics, ...
   - Recently employed for developing industrial applications
     → Workforce Management [Ricca et al. 2010-tplp]
     → E-Tourism [Ricca et al. 2010-fi]
After more than 20 years:

1. **ASP formal properties are well-understood**
   → **ASP is expressive... beyond NP** [Eiter et.al ’97]

2. **Robust and efficient implementations**
   - DLV [Leone et.al ’06], Clasp [Gebser et.al ’07],
   - CModels [Lierler ’05], IDP[Wittocx et al.], etc.
   - *continuous improvement witnessed by ASP competitions*

3. **Applications in several fields**
   - Artificial Intelligence, Knowledge Representation & Reas.,
   - Information Integration, Data cleaning, Bioinformatics, ...
   - **Recently employed for developing industrial applications**
     → Workforce Management [Ricca et al. 2010-tplp]
     → E-Tourism [Ricca et al. 2010-fi]
Lessons learned

- Viability of the exploitation of ASP in real-world apps.
  - complex business-logic at a lower (implementation) price
  - flexibility, readability, extensibility, ease of maintenance, etc.

- Practical obstacles to ASP-based development:
  1. ASP programmers needs an IDE
     → programmers accustomed to Workbenches (e.g. eclipse,...)
     → tools for simplifying development and maintenance
     → graphic tools simplify the approach of novice users
  2. ASP is not a full general-purpose language
     → some components better built with O.-O. Programming
     → ASP solutions must be embedded at some point
  3. ASP is not integrated in development processes and platforms
Lessons learned

- Viability of the exploitation of ASP in real-world apps.
  - complex business-logic at a lower (implementation) price
  - flexibility, readability, extensibility, ease of maintenance, etc.

- Practical obstacles to ASP-based development:
  1. **ASP programmers needs an IDE**
     - programmers accustomed to Workbenches (e.g. eclipse,...)
     - tools for simplifying development and maintenance
     - graphic tools simplify the approach of novice users
  2. **ASP is not a full general-purpose language**
     - some components better built with O.-O. Programming
     - ASP solutions must be embedded at some point
  3. **ASP is not integrated in development processes and platforms**
Lessons learned

- Viability of the exploitation of ASP in real-world apps.
  - complex business-logic at a lower (implementation) price
  - flexibility, readability, extensibility, ease of maintenance, etc.

- Practical obstacles to ASP-based development:
  1. **ASP programmers needs an IDE**
     - programmers accustomed to Workbenches (e.g. eclipse,...)
     - tools for simplifying development and maintenance
     - graphic tools simplify the approach of novice users
  2. **ASP is not a full general-purpose language**
     - some components better built with O.-O. Programming
     - ASP solutions must be embedded at some point
  3. **ASP is not integrated in development processes and platforms**
Development Tools for ASP

1. **ASPIDE**: IDE for ASP... *the most comprehensive*
   - Cutting-edge editing tool
     → textual/graphical (assisted) composition of programs
   - Development tools
     → debugging, profiling, testing, run configuration, output-handling
   - Application configuration and deployment tools
     → DBMS access, solver execution configuration, ...
   - Extensible with plugins

2. **A framework integrating ASP with Java**
   - The hybrid language \texttt{JASP}
     - *simply embed ASP code in a Java program*
       → bilateral interaction between ASP and Java
   - The Eclipse plug-in \texttt{JDLV}
     - compiler from \texttt{JASP} to Java
Development Tools for ASP

1. **ASPIDE**: IDE for ASP... *the most comprehensive*
   - Cutting-edge editing tool
     - textual/graphical (assisted) composition of programs
   - Development tools
     - debugging, profiling, testing, run configuration, output-handling
   - Application configuration and deployment tools
     - DBMS access, solver execution configuration, ...
   - Extensible with plugins

2. A framework integrating *ASP with Java*
   - The hybrid language *JASP*
     - *simply embed ASP code in a Java program*
     - bilateral interaction between ASP and Java
   - The Eclipse plug-in *JDLV*
     - compiler from *JASP* to Java
Development Tools for ASP

1. **ASPIDE**: IDE for ASP... *the most comprehensive*
   - Cutting-edge editing tool
     - textual/graphical (assisted) composition of programs
   - Development tools
     - debugging, profiling, testing, run configuration, output-handling
   - Application configuration and deployment tools
     - DBMS access, solver execution configuration, ...
   - Extensible with plugins

2. **A framework integrating ASP with Java**
   - The hybrid language **JASP**
     - *simply embed ASP code in a Java program*
       - bilateral interaction between ASP and Java
   - The Eclipse plug-in **JDLV**
     - compiler from **JASP** to Java
ASP Syntax and Semantics

**Syntax:**

\[
a_1 \lor \ldots \lor a_n \; \leftarrow \; b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m.
\]

- head
- body

**Semantics:**

- Consider the Ground Instantiation \( P = \text{ground}(\pi) \)
- Apply the Gelfond-Lifschitz semantics on \( P \)

**Informal meaning:**

“At least one \( a_i \) is true if \( b_1, \ldots, b_k \) are true and \( b_{k+1}, \ldots, b_m \) are false”.

Francesco Ricca

Datalog Development Tools
ASP Example

Example (3-col)

Input: a Graph represented by node( _) and edge(_, _).
Problem: assign one color out of 3 colors to each node such that two adjacent nodes have always different colors.

% guess a coloring for the nodes
(r) col(X, red) v col(X, yellow) v col(X, green) :- node(X).

% discard colorings where adjacent nodes have the same color
(c) :- edge(X, Y), col(X, C), col(Y, C).
ASPIDE: Integrated Development Environment for Answer Set Programming

O. Febbraro, K. Reale, F. Ricca
“ASPIDE: Integrated Development Environment for Answer Set Programming.”
Motivation

- **Diffused programming languages come with SDKs**
  - tools for simplifying development and maintenance
  - programmers accustomed to Workbenches (e.g. eclipse, ...)
  - graphic tools simplify the approach of novice users

- **Integrated Development Environment (IDE)**
  - tools for the entire life-cycle of development
    - from (assisted) programs editing to application deployment

- **No IDE → usage of ASP may be discouraged**
Motivation

- **Diffused programming languages come with SDKs**
  - tools for simplifying development and maintenance
  - programmers accustomed to Workbenches (e.g. eclipse,...)
  - graphic tools simplify the approach of novice users

- **Integrated Development Environment (IDE)**
  - tools for the entire life-cycle of development
    - from (assisted) programs editing to application deployment

- No IDE $\rightarrow$ usage of ASP may be discouraged
Motivation

- **Diffused programming languages come with SDKs**
  - tools for simplifying development and maintenance
  - programmers accustomed to Workbenches (e.g. eclipse,...)
  - graphic tools simplify the approach of novice users

- **Integrated Development Environment (IDE)**
  - tools for the entire life-cycle of development
    - from (assisted) programs editing to application deployment

- **No IDE → usage of ASP may be discouraged**
**Goal:** An *Integrated Development Environment supporting the entire life-cycle of ASP development*

- **ASPIDE**
  - Cutting-edge editing tool
    - textual/graphical (assisted) composition of programs
  - Development tools
    - debugging, profiling, testing,
    - run configuration, output-handling
    - visual representation of results, plug-ins
  - Application configuration and deployment tools
    - DBMS access, execution configuration, ...
Goal: An Integrated Development Environment supporting the entire life-cycle of ASP development

ASPIDE

- Cutting-edge editing tool
  - textual/graphical (assisted) composition of programs
- Development tools
  - debugging, profiling, testing,
  - run configuration, output-handling
  - visual representation of results, plug-ins
- Application configuration and deployment tools
  - DBMS access, execution configuration, ...
Main Features

- **Workspace management**
  - organize projects and related resources

- **Advanced Program Editing**
  - **Text Editor**
    - Text coloring, automatic completion, refactoring, code templates, code checking, errors highlighting, quick fixes, etc.
  - **Visual Editor with Reverse-engineering**
    - Drawing logic programs in a QBE-like style
  - **Outline navigation**

- **Dependency graph visualizer**

- **Run Configuration and Presentation of Results**
  - Setup execution & friendly browsing of results
Main Features (2)

- **Debugger, and Profiler**
  - GUI for the debugging tool *Spock* [Brain-etal-07]
  - GUI for the *DLV Profiler* [Calimeri-etal-09]

- **Unit Testing for ASP**
  - a framework in the *style of JUnit*
  - a new language for unit tests in ASP
  - graphical interface for creating unit tests

- **Annotations for ASP programs**
  - annotations start with @ (eg. @name, @schema, ...)
  - rule names, predicate schemas and database specification
Main Features (3)

- **User-defined Plugins → SDK free for download**
  1. Input data is not encoded in ASP → **Input Plugin**
     - needs to be translated
  2. Rules might need “refactorings” → **Rewriting Plugin**
     - performance, input languages etc.
  3. Specific output formats → **Output Plugin**
     - solver result as set of ASP facts

- **Database Access**
  - Connect to DBMSs via JDBC, DLV \(^DB\) TYP files, ODBC import/export
  - Attribute names and data types for predicates
  - *Data Integration scenario*
  - **Implemented as an Input Plugin**
ASPIDE at work

DEMO

1. Create/Modify/Test Programs
2. Create a program that runs on a RDBMS
JASP: integrating Java with Answer Set Programming

O. Febbraro, N. Leone, G. Grasso, F. Ricca
“JASP: A Framework for Integrating Answer Set Programming with Java.”
ASP and Object-Oriented Programming

- **Existing Application Programming Interfaces (API)**
  - the DLV Wrapper [Ricca 2003], OntoDLV API [Gallucci ’07]
    - libraries for interacting with an ASP solver from a Java program
    - control the execution of an external solver and
    - convert data from logic-based to Java representations

- **Shortcomings:**
  1. the programmer has the burden of the integration → repetitive and time-consuming ad-hoc procedures
  2. no support from programming tools and workbenches
  3. no support for enterprise applications standards → persistency of complex object-oriented domain models

- Developers are hindered from adopting a poorly-supported non standard technology
ASP and Object-Oriented Programming

- **Existing Application Programming Interfaces (API)**
  - the DLV Wrapper [Ricca 2003], OntoDLV API [Gallucci '07]
    - libraries for interacting with an ASP solver from a Java program
    - control the execution of an external solver and
    - convert data from logic-based to Java representations

- **Shortcomings:**
  1. the programmer has the burden of the integration
     → repetitive and time-consuming ad-hoc procedures
  2. no support from programming tools and workbenches
  3. no support for enterprise applications standards
     → persistency of complex object-oriented domain models

- Developers are hindered from adopting a poorly-supported non standard technology
ASP and Object-Oriented Programming

- **Existing Application Programming Interfaces (API)**
  - the DLV Wrapper [Ricca 2003], OntoDLV API [Gallucci ’07]
    - libraries for interacting with an ASP solver from a Java program
    - control the execution of an external solver and
    - convert data from logic-based to Java representations

- **Shortcomings:**
  1. the programmer has the burden of the integration
     → repetitive and time-consuming ad-hoc procedures
  2. no support from programming tools and workbenches
  3. no support for enterprise applications standards
     → persistency of complex object-oriented domain models

- Developers are hindered from adopting a poorly-supported non standard technology
The needs that inspired \textit{JASP}

\textbf{Needs:}

- Seamless embedding of ASP
- Integration in standard software processes/technologies
- Support for development tools/workbenches

\textbf{A framework integrating ASP with Java}

1. The hybrid language \textit{JASP}
   - simply embed ASP code in a Java program
   - bilateral interaction between ASP and Java
   - exploit standard ORM technologies

2. An implementation of \textit{JASP}
   - compiler from \textit{JASP} to Java
   - plugin for the Eclipse platform
The needs that inspired **JASP**

- **Needs:**
  - Seamless embedding of ASP
  - Integration in standard software processes/technologies
  - Support for development tools/workbenches

- **A framework integrating ASP with Java**
  1. The hybrid language **JASP**
     - *simply embed ASP code in a Java program*
       → bilateral interaction between ASP and Java
     - exploit standard ORM technologies
  2. An implementation of **JASP**
     - compiler from **JASP** to Java
     - plugin for the Eclipse platform
Common Scenario

Develop an application: (e.g., Map coloring tool)

- Adopt (say) Java
  → libraries, programming tools, O-O programming...
- Complex problem to solve (e.g., 3-Col)
  1 Write a complex backtracking procedure
  2 Write a two-rule ASP program!

Developer Desiderata:

1 I do not want to write ASP to Java glue code
   - I need to know an API
   - Repetitive tasks: Variables → Facts + Build ASP Rules →
     → Call solver + Parse Answer Sets → Variables

2 I do not want to change my programming environment
   (e.g., Eclipse)
Common Scenario

**Develop an application:** (e.g., Map coloring tool)

- **Adopt (say) Java**
  - libraries, programming tools, O-O programming...
- **Complex problem to solve (e.g., 3-Col)**
  1. Write a complex backtracking procedure
  2. Write a two-rule ASP program!

**Developer Desiderata:**

1. **I do not want to write ASP to Java glue code**
   - I need to know an API
   - Repetitive tasks: Variables → Facts + Build ASP Rules → Call solver + Parse Answer Sets → Variables
2. **I do not want to change my programming environment** (e.g., Eclipse)
Common Scenario

Develop an application: (e.g., Map coloring tool)

- Adopt (say) Java
  → libraries, programming tools, O-O programming...
- Complex problem to solve (e.g., 3-Col)
  1. Write a complex backtracking procedure
  2. Write a two-rule ASP program!

Developer Desiderata:

1. I do not want to write ASP to Java glue code
   - I need to know an API
   - Repetitive tasks: Variables → Facts + Build ASP Rules →
     → Call solver + Parse Answer Sets → Variables

2. I do not want to change my programming environment
   (e.g., Eclipse)
Common Scenario

Develop an application: (e.g., Map coloring tool)

- Adopt (say) Java
  - libraries, programming tools, O-O programming...
- Complex problem to solve (e.g., 3-Col)
  1. Write a complex backtracking procedure
  2. Write a two-rule ASP program!

Developer Desiderata:

1. I do not want to write ASP to Java glue code
   - I need to know an API
   - Repetitive tasks: Variables → Facts + Build ASP Rules →
     Call solver + Parse Answer Sets → Variables
2. I do not want to change my programming environment (e.g., Eclipse)
Common Scenario

**Develop an application:** (e.g., Map coloring tool)

- Adopt (say) Java
  - libraries, programming tools, O-O programming...
- Complex problem to solve (e.g., 3-Col)
  1. Write a complex backtracking procedure
  2. Write a two-rule ASP program!

**Developer Desiderata:**

1. I do not want to write ASP to Java glue code
   - I need to know an API
   - Repetitive tasks: Variables → Facts + Build ASP Rules → Call solver + Parse Answer Sets → Variables
2. I do not want to change my programming environment (e.g., Eclipse)
Common Scenario

Develop an application: (e.g., Map coloring tool)

- Adopt (say) Java
  → libraries, programming tools, O-O programming...
- Complex problem to solve (e.g., 3-Col)
  1. Write a complex backtracking procedure
  2. Write a two-rule ASP program!

Developer Desiderata:

1. I do not want to write ASP to Java glue code
   - I need to know an API
   - Repetitive tasks: Variables → Facts + Build ASP Rules →
     → Call solver + Parse Answer Sets → Variables

2. I do not want to change my programming environment (e.g., Eclipse)
Common Scenario

Develop an application: (e.g., Map coloring tool)

- Adopt (say) Java
  → libraries, programming tools, O-O programming...
- Complex problem to solve (e.g., 3-Col)
  1. Write a complex backtracking procedure
  2. Write a two-rule ASP program!

- Developer Desiderata:
  1. I do not want to write ASP to Java glue code
     - I need to know an API
     - Repetitive tasks: Variables → Facts + Build ASP Rules →
       → Call solver + Parse Answer Sets → Variables
  2. I do not want to change my programming environment
     (e.g., Eclipse)
JASP-core by example

```java
class Graph {
    private Set<Arc> arcs = new HashSet<Arc>();
    private Set<String> nodes = new HashSet<String>();

    public void addNode(String id) {
        nodes.insert(id);
    }

    public void addArc(String from, String to) {
        arcs.insert(new Arc(from, to));
    }

    public Set<Colored> compute3Coloring() {
        Set<Colored> res = new HashSet<Colored>();
        <# in=arcs::arc, nodes::node out=res::col
            col(X, red) v col(X, green)
            v col(X, blue) :- node(X).
            :- col(X, C), col(Y, C), arc(X, Y).
        #>
        if_no_answerset { res = null; }
        return res;
    }

    public class Arc {
        public String start; public String end;
    }

    public class Colored {
        public String node; public String color;
    }
}
```

Java World
JASP-core by example

```java
1  class Graph {
2      private Set<Arc> arcs = new HashSet<Arc>();
3      private Set<String> nodes = new HashSet<String>();
4  
5  public void addNode(String id) {
6      nodes.insert (id);
7  }
8  
9  public void addArc(String from, String to) {
10     arcs.insert (new Arc(from, to));
11  }
12  
13  public Set<Colored> compute3Coloring() {
14     Set<Colored> res = new HashSet<Colored>();
15     <# in=arcs::arc nodes::node out=res::col
16        col(x,red) v col(x,green)
17        v col(x,blue) :- node(x).
18        :- col(x,C), col(y,C), arc(x,y).
19     }
20  }
21  
22  if_no_answerset { res = null; }
23  return res; }
24  
25  public class Arc {
26      public String start; public String end;  }
27  
28  public class Colored {
29      public String node; public String color;  }
30
```

The image shows a class definition for `Graph` in Java, with an implementation for adding nodes and arcs. It also demonstrates a Prolog rule for computing a 3-coloring of a graph, with a comment indicating the use of JASP-core for generating Prolog rules from Java code.
**JASP-core by example**

```java
1 class Graph {
2   private Set<Arc> arcs = new HashSet<Arc>();
3   private Set<String> nodes = new HashSet<String>();
4   public void addNode(String id){
5     nodes.insert(id); }
6   public void addArc(String from, String to){
7     arcs.insert(new Arc(from, to)); }
8   public Set<Colored> compute3Coloring(){
9     Set<Colored> res = new HashSet<Colored>();
10    <# in=arcs::arc, nodes::node out=res::col
11       col(X, red) v col(X, green)
12       v col(X, blue) :- node(X).
13       :- col(X, C), col(Y, C), arc(X, Y).
14     if_no_answerset { res = null; }  
15     return res; }
16  }
17 public class Arc {
18   public String start; public String end;  
19  }
20 public class Colored {
21   public String node; public String color;}
```
Introduction
Language Overview

ASPIDE: Integrated Development Environment for ASP

JASP: Integrating ASP with Java

Conclusion and Related Work

JASP\text{-}core by example

```java
1. class Graph {
2.     private Set<Arc> arcs = new HashSet<Arc>();
3.     private Set<String> nodes = new HashSet<String>();
4.     public void addNode(String id) {
5.         nodes.insert(id);
6.     }
7.     public void addArc(String from, String to) {
8.         arcs.insert(new Arc(from, to));
9.     }
10.    public Set<Colored> compute3Coloring() {
11.        Set<Colored> res = new HashSet<Colored>();
12.        for (Arc arc : arcs) {
13.            if (!res.contains(arc)) {
14.                res.add(arc);
15.            }
16.        }
17.    }
18. }
19.
20. class Arc {
21.     public String start; public String end;
22. }
23. class Colored {
24.     public String node; public String color;
25. }
```

Java World

ASP World

Object to Relational

Relational to Object
**JASP-core by example**

```java
1  class Graph {
2      private Set<Arc> arcs = new HashSet<Arc>();
3      private Set<String> nodes = new HashSet<String>();
4  public void addNode(String id) {
5          nodes.insert(id);
6      }
7  public void addArc(String from, String to) {
8          arcs.insert(new Arc(from, to));
9      }
10  public Set<Colored> compute3Coloring() {
11      Set<Colored> res = new HashSet<Colored>();
12      <# in=arcs::arc, nodes::node out=res::col
13          col(X, red) v col(X, green)
14          v col(X, blue) :- node(X).
15          :- col(X, C), col(Y, C), arc(X, Y).
16      #>
17      if_no_answerset { res = null; }
18      return res; }
19  public class Arc {
20      public String start; public String end;  }
21  public class Colored {
22      public String node; public String color;}
```
Default ORM Mapping

Map Java objects to ASP facts (and vice-versa)

- **Impedance Mismatch problem** [Maier '90, Keller et al. '93]
  → shared with object persistence frameworks

- **Basic Object-Relational Mapping (ORM) Strategy**
  - For classes with:
    - no-arguments constructor, non-recursive type definition
    - getters and setters, one-to-one associations only

  - Intuitively:
    - one attribute per (simple) field
    - several fields for one-to-one associations
    - *compound key* of all class attributes + *embedded value*
    - attributes are filled according to the declaration order of fields

- ... custom mappings with Java Persistence API (JPA)
public class Arc {
    private String start;
    private String end;

    public Arc(String from, String to) {
        start = from; end = to;
    }

    public String getStart() {
        return start;
    }

    public void setStart(String start) {
        this.start = start;
    }

    public String getEnd() {
        return end;
    }

    public void setEnd(String end) {
        this.end = end;
    }
}

public static void main(String[] args) {
    //...
    ArrayList<Arc> arcs = new ArrayList<Arc>();

    arcs.add(new Arc("1", "2"));
    arcs.add(new Arc("2", "3"));
    //...
    <# in=arcs::arc

    arc("1", "2").
    arc("2", "3").
    ...
}
Semantics

```java
public Set<Colored> compute3Coloring()
{
    Set<Colored> res = new HashSet<Colored>();
    <# in=arcs::arc, nodes::node out=res::col
        col(X,red) v col(X,green)
            v col(X,blue) :- node(X).
        :- col(X,C), col(Y,C), arc(X,Y).
    #>
    if_no_answerset { res = null; } return res; }
```
Enterprise Applications:

AkA: “information systems” or “data processing”

They involve:

- persistent data... a lot of data
- a lot of user interface screens
- complex business (il)logic

Widely employed technologies

- Relational Database Systems
- Persistence via ORM Tools (e.g., Hibernate, TopLink)
- Component based Frameworks: (e.g., EJB, Spring)
- Web-based Interfaces
- Fully-featured Workbenches (e.g., Eclipse)
Enterprise Applications Architecture

Presentation Layer

Business Logic Layer

Persistence Layer

Database Layer
The **JASP** Language

- **The basic language** *JASP-core*
  - is ok in most cases
  - improved for easing (enterprise) application development

- **JASP (full) features:**
  - Java Persistence API (JPA) Mappings
  - Named Non-positional Notation
  - Dynamic Composition of Modules
  - Access Java variables in logic rules
  - Database Access
JPA Mappings

Custom ORM mappings according with Java Persistence API

```java
package it.unical.mat.teambuilding.persistence;
import java.util.ArrayList;

@Entity
public class Shift implements Comparable<Shift>{
    @Id
    Integer id;
    @Column
    String name;
    @OneToMany(cascade=CascadeType.ALL)
    List<NeededEmployee> neededEmployees=new ArrayList<NeededEmployee>();
    @Column
    Date date;
    @Column
    Integer duration;
    @OneToMany(cascade=CascadeType.ALL)
    List<Employee> excludedEmployees=new ArrayList<Employee>();
    public Shift()
}

//...
```
Supported JPA Annotations

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Entity</td>
<td>Indicates a class with mapping. Class name is the predicated name.</td>
</tr>
<tr>
<td>@Table (name=&quot;pred-name&quot;)</td>
<td>In conjunction with @Entity, to rename the default predicate name.</td>
</tr>
<tr>
<td>@Column</td>
<td>Identifies a class member, to be included in the mapping.</td>
</tr>
<tr>
<td>@Id</td>
<td>Marks a class member as identifier (key) of the relative table.</td>
</tr>
<tr>
<td>@OneToMany</td>
<td>On class members to denote associations multiplicity</td>
</tr>
<tr>
<td>@ManyToOne</td>
<td></td>
</tr>
<tr>
<td>@ManyToOne</td>
<td></td>
</tr>
<tr>
<td>@ManyToMany</td>
<td></td>
</tr>
<tr>
<td>@OneToOne</td>
<td></td>
</tr>
<tr>
<td>@JoinColumn (name=&quot;pred-name&quot;)</td>
<td>In conjunction with @OneToMany or @ManyToOne to specify a mapping</td>
</tr>
<tr>
<td></td>
<td>realized through an associative predicate</td>
</tr>
</tbody>
</table>
Non-positional Notation & Dynamic Composition

1. Access attributes by name → compact and intuitive rules
2. Compose ASP programs at runtime

```java
public void createTeam(boolean forceMixG) {
    List<Person> personList = loadPeople();
    <#+ (ml) in=personList::people
    inTeam(X,G) v outTeam(X,G) :-
        people(name: X, gender:G).
    :- #count{X: inTeam(X,G)} >5.
    #>
    if (forceMixG) {
        <#+ (ml)
        :- inTeam(X,GX), not inTeam(Y, GY),
        people(name: Y, gender: GY), GX != GY.
    #>
    }
    Set<Team> res = new HashSet<Team> ();
    <# (ml) out=res::inTeam #>
    for_each_answerset {
        //do something with res
    }
}```
Access Java variables

1. Access Java variables’ content to build ASP rules
2. Dot notation for object properties $\rightarrow$ Efficient conversion

---

1. 
   ```
   for (int i = 0; i<10; i++)
   
   <#+ (dyn)
   a($\{i\},$\{i+1\}). #>
   
   a(1,2). a(2,3). ... a(10,11).
   ```

2. 
   ```
   <#+ (m2) in=b
   p(N):-[b.flights.passengers.passport](N). #>
   ```

   Equivalent to:
   
   ```
   b(X), person(id:X, flight:F), flights(id:F, passenger:P), passports(id:P, num:N)
   ```

   actually generates only specific facts from variable b:

   ```
   "[b.flights.passengers.passport]"("JKH87687SDFHS349728").
   "[b.flights.passengers.passport]"("FSDJHRMEW749382MDS7").
   "[b.flights.passengers.passport]"("VXMNCWIEU84390MN4B").
   ...
Database Access

- Support data intensive applications → database access
- Interact directly with DBMSs, possibly exploit DLV$^{DB}$

```datalog
<# fromDB=Arc::arc
    reachable(X,Y) :- arc(X,Y).
    reachable(X,Y) :- arc(X,Y), reachable(Y,Y).
    reaches(Y) :- reachable("Rome",Y).
#>
```

- Access tables `fromTable = Table@DBURL :: predname`
Database Access

- Support data intensive applications → database access
- Interact directly with DBMSs, possibly exploit DLV$^{DB}$

<# fromDB=Arc:::arc
    reachable(X,Y) :- arc(X,Y).
    reachable(X,Y) :- arc(X,U), reachable(U,Y).
    reaches(Y) :- reachable("Rome",Y).
#>

- Access tables fromTable = Table@DBURL :: predname
Implementation

- **The Jdlvc compiler**
  - generates plain Java classes from \texttt{JASP} files
  - support ASP-core and DLV syntax
  - Implemented in Java, based on the DLV Wrapper API

- **The JDLV Eclipse plug-in**
  - Editing Features
    - automatic completion
    - dynamic code checking and errors highlighting
    - outline view
  - Includes \texttt{Jdlvc}
    - automatic generation of Java code, by means of our \texttt{Jdlvc} compiler.
  - Java and ASP on the same environment!
Compiler and Generated Code Performance

### Reachability - Scatter Plot (msec)

<table>
<thead>
<tr>
<th>File Size (MB)</th>
<th>Executable Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>330</td>
<td>93</td>
</tr>
<tr>
<td>660</td>
<td>115</td>
</tr>
<tr>
<td>990</td>
<td>137</td>
</tr>
<tr>
<td>1320</td>
<td>159</td>
</tr>
<tr>
<td>1650</td>
<td>181</td>
</tr>
<tr>
<td>1980</td>
<td>203</td>
</tr>
<tr>
<td>2310</td>
<td>225</td>
</tr>
<tr>
<td>2640</td>
<td>247</td>
</tr>
<tr>
<td>2970</td>
<td>269</td>
</tr>
<tr>
<td>3300</td>
<td>291</td>
</tr>
<tr>
<td>3630</td>
<td>313</td>
</tr>
<tr>
<td>3960</td>
<td>335</td>
</tr>
<tr>
<td>4290</td>
<td>357</td>
</tr>
<tr>
<td>4620</td>
<td>379</td>
</tr>
<tr>
<td>4950</td>
<td>401</td>
</tr>
<tr>
<td>5280</td>
<td>423</td>
</tr>
<tr>
<td>5610</td>
<td>445</td>
</tr>
<tr>
<td>5940</td>
<td>467</td>
</tr>
<tr>
<td>6270</td>
<td>489</td>
</tr>
<tr>
<td>6600</td>
<td>511</td>
</tr>
<tr>
<td>6930</td>
<td>533</td>
</tr>
<tr>
<td>7260</td>
<td>555</td>
</tr>
<tr>
<td>7590</td>
<td>577</td>
</tr>
<tr>
<td>7920</td>
<td>599</td>
</tr>
<tr>
<td>8250</td>
<td>621</td>
</tr>
</tbody>
</table>

### Reachability - Execution Times

<table>
<thead>
<tr>
<th>File Size (MB)</th>
<th>Execution Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>330</td>
<td>489</td>
</tr>
<tr>
<td>660</td>
<td>511</td>
</tr>
<tr>
<td>990</td>
<td>533</td>
</tr>
<tr>
<td>1320</td>
<td>555</td>
</tr>
<tr>
<td>1650</td>
<td>577</td>
</tr>
<tr>
<td>1980</td>
<td>599</td>
</tr>
<tr>
<td>2310</td>
<td>621</td>
</tr>
<tr>
<td>2640</td>
<td>643</td>
</tr>
<tr>
<td>2970</td>
<td>665</td>
</tr>
<tr>
<td>3300</td>
<td>687</td>
</tr>
<tr>
<td>3630</td>
<td>709</td>
</tr>
<tr>
<td>3960</td>
<td>731</td>
</tr>
<tr>
<td>4290</td>
<td>753</td>
</tr>
<tr>
<td>4620</td>
<td>775</td>
</tr>
<tr>
<td>4950</td>
<td>797</td>
</tr>
<tr>
<td>5280</td>
<td>819</td>
</tr>
<tr>
<td>5610</td>
<td>841</td>
</tr>
<tr>
<td>5940</td>
<td>863</td>
</tr>
<tr>
<td>6270</td>
<td>885</td>
</tr>
<tr>
<td>6600</td>
<td>907</td>
</tr>
<tr>
<td>6930</td>
<td>929</td>
</tr>
<tr>
<td>7260</td>
<td>951</td>
</tr>
<tr>
<td>7590</td>
<td>973</td>
</tr>
<tr>
<td>7920</td>
<td>995</td>
</tr>
<tr>
<td>8250</td>
<td>1017</td>
</tr>
</tbody>
</table>

### Compiler Performance

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size (KB)</th>
<th>Size (LOC)</th>
<th>DLV Calls</th>
<th>Compile Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SokobanDecision.jdlv</td>
<td>9</td>
<td>205</td>
<td>1</td>
<td>70 ms</td>
</tr>
<tr>
<td>WeightAssignmentTree.jdlv</td>
<td>4</td>
<td>123</td>
<td>1</td>
<td>30 ms</td>
</tr>
<tr>
<td>TwentyCalls.jdlv</td>
<td>47</td>
<td>1509</td>
<td>20</td>
<td>51 ms</td>
</tr>
<tr>
<td>Huge.jdlv</td>
<td>7456</td>
<td>188938</td>
<td>1</td>
<td>4871 ms</td>
</tr>
</tbody>
</table>

Francesco Ricca
Datalog Development Tools
Use-case: Workforce Management via ASP

- **The Gioia Tauro seaport**
  - the largest transshipment terminal of the Mediterranean Sea
  - main activity: container transshipment [Vacca et. al]
  - recently become an *automobile hub*

- **Automobile Logistics by ICO B.L.G.** (subsidiary of BLG Logistics Group)
  - several ships of different size shore the port every day,
  - transported vehicles are handled, warehoused, technically processed and then delivered to their final destination.

- **Management Goal:** promptly serve shoring boats!
  - **Crucial task:** arranging suitable teams of employees
    - teams are subject to many constraints
      (contract, required skills, fair distribution of load)

- **Designed a Team Building system with ASP**
JASP ORM Mapping

```java
@Entity public class Employee
{
  @Id Integer id;
  String name;
  @OneToMany
  @JoinColumn(name="hasSkill")
  Set<Skill> skills;
}

@Entity
public class Allocation
{
  @Id Integer id;
  @OneToOne Employee employee;
  @OneToOne Shift shift;
  @OneToOne Skill skill;
}

@Entity public class Calendar
{
  @Id Integer id;
  @OneToOne Employee employee;
  Date date;
  Boolean isAbsent;
  Integer dayHours, weekHours, weekOvertime;
}

@Entity public class Skill
{
  @Id String name;
  Boolean isCrucial, isHeavy;
}

@Entity public class Shift
{
  @Id Integer id;
  Date date;
  Integer duration;
  @OneToMany
  Map<Skill, Integer> neededEmp;
  @OneToMany
  @JoinColumn(name="excluded")
  Set<Employee> excluded;
}
```


**JASP Encoding**

```java
public Allocation computeTeam(Shift s,
    Set<Employee> e, Integer maxGap){
    Set<Allocation> teams = Sets.newHashSet();
    <#+(teambuilding)
        fromDB=AllocationDiary::lastAllocation,
        Calendar:@
        in=s::shift, e::employee
        out=teams::assign
        assign(employee:Em, shift:Sh, skill:Sk)
        ▼ nAssign(Em, Sh, Sk) :-
            canBeAssigned(Em, Sh, Sk).

    canBeAssigned(Em, Sh, Sk) :-
        [e.hasSkill](Em, Sk),
        [s.neededEmp](Sh, Sk, _),
        not exceedTimeLimit(Em, Sh),
        not absent(Em, Sh), not [s.excluded](Sh, Em).

    absent(E, Sh) :- shift(id:Sh, date:D),
        Calendar(date:D, employee:E, absent:true).
    #>
```

Francesco Ricca

Datalog Development Tools
JASP at work

DEMO

1. Application to Workforce Management
2. Modify a simple program
Conclusion

On the field experience → Development Tools

1. **ASPIDE:** The complete IDE for ASP
   - Advanced editing features... extensible via plugins
     → textual/graphical editing tool, debugging, testing, profiling, etc.
     → *supports the entire life-cycle of logic programs development*
   - [http://www.mat.unical.it/ricca/aspide](http://www.mat.unical.it/ricca/aspide)

2. **A framework for integrating Java and ASP**
   - The **JASP** language
     → *ASP code in-lined within Java programs*
     → complies with application development standards (i.e., JPA)
   - **JDLV:** a plugin for the Eclipse platform
Related Work

- **API-Based approaches**
  - JSetL [Rossi et al. ’07], ILOG, Jess, ...

- **Multi-paradigm Languages** [Hailpern 1986; Placer 1991]
  - Alma-0 [Apt et al. ’98], DJ [Zhou ’99],
  - Oz [Roy 2005], PROVA [Koslenkov et al. ’06],
  - etc. [Spinellis 1994]

- **ASP and Java**
  - DLVWrapper [Ricca 2003]
  - OntoDLV API [Gallucci and Ricca 2007]
  - Recent proposal of an hybrid language [Oetsc et al. 2011]