ASP in Industrial Contexts: Applications and Toolchain

Francesco Ricca
Department of Mathematics and Computer Science
University of Calabria
RuleML Webminar 2018
Outline

1. Introduction
2. Industrial Applications of DLV
3. Development Tools for ASP
4. Lessons learned and current developments
5. References
Answer Set Programming (ASP)

- Declarative programming paradigm
- Non-monotonic reasoning and logic programming
- Stable model semantics [GL91]

Expressive KR Language

- Roots in Datalog and Nonmonotonic Logic
- Default negation, Disjunction, Constraints, Aggregates,
- Weak constraints, Functions, Lists, Sets...
- Can model problems up to $\Sigma^P_2/\Pi^P_2$ [EGM97, DEGV01]
  - even problems not (polynomially) translatable to SAT or CSP
Problem solving in ASP

**Idea:**

1. Represent a computational problem by a Logic program
2. Answer sets correspond to problem solutions
3. Use an ASP solver to find these solutions
Applications of ASP

Applications in several fields

- Artificial Intelligence, Knowledge Representation & Reas.,
- Information Integration, Data cleaning, Bioinformatics, Robotics...
- see e.g. [Lif02, EFLP99, EFLP99, EIST06, EEB10, Sak11, SN99, DGH09, CHO^+09, RDG^+10, RGA^+12, GNA13]

Robust and efficient implementations

- DLV [LPF^+06], Wasp [ADLR15], Clasp [GKNS07],
- CModels [LM04], IDP [WMD08], etc.
- continuous improvement (see ASP competitions [CIR^+11])
Applications of ASP

Applications in several fields
- Artificial Intelligence, Knowledge Representation & Reas.,
- Information Integration, Data cleaning, Bioinformatics, Robotics...
- see e.g. [Lif02, EFLP99, EFLP99, EIST06, EEB10, Sak11, SN99, DGH09, CHO+09, RDG+10, RGA+12, GNA13]

Robust and efficient implementations
- DLV [LPF+06], Wasp [ADLR15], Clasp [GKNS07],
- CModels [LM04], IDP [WMD08], etc.
- continuous improvement (see ASP competitions [CIR+11])

We used DLV and WASP in industrial applications
The ASP Tools we use

The DLV System

- One of the most popular ASP systems
  ...more than fifteen years of research and development
- Actively maintained
  - University of Calabria: Research and Extension
  - DLV System s.r.l.: Maintenance & Commercialization
    - Spin-Off of University of Calabria

The Wasp solver

- Latest research product
- Winner of the Marathon track in ASPCOMP’15
- Next major release of DLV will be based on it
The ASP Tools we use

The DLV System
- One of the most popular ASP systems
  ...more than fifteen years of research and development
- Actively maintained
  - University of Calabria: Research and Extension
  - DLV System s.r.l.: Maintenance & Commercialization
    - Spin-Off of University of Calabria

The Wasp solver
- Latest research product
- Winner of the Marathon track in ASPCOMP’15
- Next major release of DLV will be based on it
Routing and classification of call-center customers

- ZLog platform employed by Telecom Italia call-centers
  *(Telecom Italia is the largest Italian carrier)*

**Team Building in the Gioia-Tauro Seaport** [RGA+12]
- Team builder for ICO BLG in the Gioia Tauro seaport

**Tools for the touristic industry**
- Intelligent allotment of touristic packages [DLNR15]
- A mediator system for e-tourism [RDG+10]

**Automatic Diagnosis of Headache Disorders**
- The International Headache Society (IHS) Classification
Cleaning medical archives [GNA13]
- Distributed archives of the Italian Healthcare System

Intelligent Data Extraction
- DIADEM Project (U. Oxford)

Business Simulation Games
- ASP-based Autonomous Players

Automatic Itinerary Search

... and many others! [LR15, GLMR11]
An introduction to ASP
The language of ASP is:
- Datalog
- Default negation
- Disjunction
- Integrity Constraints
- Weak Constraints
- Aggregate atoms

Programming Methodology:
- Guess, Check, Optimize
ASP Syntax

Rule: \[ a_1 \mid \ldots \mid a_n \quad \vdash \quad b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m. \]

Atoms and Literals: \( a_i, b_i, \text{not } b_i \)
Positive Body: \( b_1, \ldots, b_k \)
Negative Body: \( \text{not } b_{k+1}, \ldots, \text{not } b_m. \)

Fact: A rule with empty body
Constraint: A rule with empty head

Variables: allowed in atom’s arguments

- Must occur in the positive body (Safety)
- Are placeholders for constants
- “Replace variables by constants in all possible ways” (Instantiation)
ASP Syntax

Rule: \[ a_1 \mid \ldots \mid a_n : \begin{array}{c} b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m. \end{array} \]

- **Atoms and Literals:** \( a_i, b_i, \text{not } b_i \)
- **Positive Body:** \( b_1, \ldots, b_k \)
- **Negative Body:** \( \text{not } b_{k+1}, \ldots, \text{not } b_m. \)

- **Fact:** A rule with empty body
- **Constraint:** A rule with empty head

**Variables:** allowed in atom’s arguments

- Must occur in the positive body (Safety)
- Are placeholders for constants
- “Replace variables by constants in all possible ways” (Instantiation)
Informal Semantics

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

Informal Semantics:

“If all \( b_1, \ldots, b_k \) are true and all \( b_{k+1}, \ldots, b_m \) are not true, then at least one among \( a_1, \ldots, a_n \) is true”.

Example (Datalog + Disjunction + Negation)

% Disjunctive knowledge: “A parent \( P \) is either a father or a mother”

\[
\text{mother} (P, S) \mid \text{father} (P, S) : - \text{parent} (P, S). \\
\]

% Constrains: “Ensure that none is the parent of himself.”

\[
- \text{parent} (F, S), \text{not } \text{parent} (S, F). \\
\]
Informal Semantics

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

Informal Semantics:

“If all \( b_1, \ldots, b_k \) are true and all \( b_{k+1}, \ldots, b_m \) are not true, then at least one among \( a_1, \ldots, a_n \) is true”.

Example (Datalog + Disjunction + Negation)

% Disjunctive knowledge: “A parent \( P \) is either a father or a mother”
\[ \text{mother}(P, S) \mid \text{father}(P, S) \leftarrow \text{parent}(P, S). \]

% Constrains: “Ensure that none is the parent of himself.”
\[ \leftarrow \text{parent}(F, S), \text{not } \text{parent}(S, F). \]
Informal Semantics (2)

Example

\[
\text{isInterestedinASP}(\text{john}) \mid \text{isCurious}(\text{john}) : - \text{attendsASP}(\text{john}).
\text{attendsASP}(\text{john}).
\]

Two (minimal) models encoding two plausible scenarios:

- \( M_1 : \{ \text{isInterestedinASP}(\text{john}), \text{attendsASP}(\text{john}) \}. \)
- \( M_2 : \{ \text{isCurious}(\text{john}), \text{attendsASP}(\text{john}) \}. \)
Constraint:

\[ \neg b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m. \]

Informal Semantics:

“It is not possible that all \( b_1, \ldots, b_k \) are true and all \( b_{k+1}, \ldots, b_m \) are false”.

Example

\begin{align*}
\text{isInterestedinASP}(\text{john}) \mid \text{isCurious}(\text{john}) & :\neg \text{attendsASP}(\text{john}). \\
\neg \text{hatesASP}(\text{john}), \text{isInterestedinASP}(\text{john}) & \neg \text{attendsASP}(\text{john}). \text{hatesASP}(\text{john}).
\end{align*}

Only one plausible scenario:

\begin{itemize}
  \item \( M_1:\{\text{isInterestedinASP}(\text{john}), \text{attendsASP}(\text{john}), \text{hatesASP}(\text{john}).\}\)
  \item \( M_2:\{\text{isCurious}(\text{john}), \text{attendsASP}(\text{john}), \text{hatesASP}(\text{john}).\}\)
\end{itemize}
Informal Semantics (3)

Constraint:

\[ :- b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m. \]

Informal Semantics:

“It is not possible that all \( b_1, \ldots, b_k \) are true and all \( b_{k+1}, \ldots, b_m \) are false”.

Example

\[ isInterestedinASP(john) \lor isCurious(john) :- attendsASP(john). \]
\[ :- hatesASP(john), isInterestedinASP(john). \]
\[ attendsASP(john). \quad hatesASP(john). \]

Only one plausible scenario:

- \( M_1:\{isInterestedinASP(john), attendsASP(john), hatesASP(john)\} \)
- \( M_2:\{isCurious(john), attendsASP(john), hatesASP(john)\} \)
Weak Constraints:

- Express desiderata
- *Constraints which should possibly be satisfied (as soft constraints in CSP)*

Syntax: $\sim b(X, Y)$.

Intuitive meaning: “set $b$ as false, if possible”
Weak Constraints:

- Express desiderata
- \textit{Constraints which should possibly be satisfied (as soft constraints in CSP)}

Syntax: \( \sim b(\overline{X}, \overline{Y}). \ [w@p] \)

Weight and Priority: \( ([w@p]) \)

- higher weights/priorities \( \Rightarrow \) higher importance
- “@p” can be omitted

“minimize the sum of the weights of the violated constraints in the highest priority level, and so on”

Declarative specification of optimization problems
Aggregate atoms: Express functions calculated over sets of elements:

\[ L_g <_{op} f\{S\} <_{op} U_g \]

**Example**

\[ 5 < \#\text{count}\{\text{EmpId} : \text{emp}(\text{EmpId, male, Skill, Salary})\} \leq 10 \]

The atom is true if the number of male employees is greater than 5 and does not exceed 10.
Example (Team Building)

% An employee is either included in the team or not
\( inTeam(I) \mid outTeam(I) :\neg \{ emp(I, Sx, Sk, Sa) \} \).

% The team consists of a certain number of employees
\( nEmp(N), \not \#count\{ I : inTeam(I) \} = N \).

% At least a given number of different skills must be present in the team
\( nSkill(M), \not \#count\{ Sk : emp(I, Sx, Sk, Sa), inTeam(I) \} \leq M \).

% The sum of the salaries of the employees working in the team must not exceed the given budget
\( budget(B), \not \#sum\{ Sa, I : emp(I, Sx, Sk, Sa), inTeam(I) \} \leq B \).

% The salary of each individual employee is within a specified limit
\( maxSal(M), \not \#max\{ Sa : emp(I, Sx, Sk, Sa), inTeam(I) \} \leq M \).
Programming Methodology

Guess & Check & Optimize (GCO)

1. **Guess** solutions → using disjunctive rules
2. **Check** admissible ones → using strong constraints

Optimization problem?

3. Specify **Preference** criteria → using weak constraints

In other words...

1. disjunctive rules → generate candidate solutions
2. constraints → test solutions discarding unwanted ones
3. weak constraints → single out optimal solutions
**Programming Methodology**

**Guess & Check & Optimize (GCO)**

1. **Guess** solutions → using disjunctive rules
2. **Check** admissible ones → using strong constraints

*Optimization problem?*

3. **Specify Preference** criteria → using weak constraints

In other words...

1. disjunctive rules → generate candidate solutions
2. constraints → test solutions discarding unwanted ones
3. weak constraints → single out optimal solutions
### Guess & Check & Optimize (GCO)

1. **Guess** solutions → using disjunctive rules
2. **Check** admissible ones → using strong constraints

*Optimization problem?*

3. Specify **Preference** criteria → using weak constraints

### In other words...

1. disjunctive rules → generate candidate solutions
2. constraints → test solutions discarding unwanted ones
3. weak constraints → single out optimal solutions
Example (3-col)

**Problem:** Given a graph, assign one color out of 3 colors to each node such that two adjacent nodes have always different colors.

**Input:** a Graph is represented by \( \textit{node}(\_\_\_) \) and \( \textit{edge}(\_, \_, \_) \).

\[
\begin{align*}
\text{% guess a coloring for the nodes} \\
(r) & \quad \text{\textit{col}}(X, \text{red}) \mid \text{\textit{col}}(X, \text{yellow}) \mid \text{\textit{col}}(X, \text{green}) :\! : \! \text{\textit{node}}(X). \\
\text{% discard colorings where adjacent nodes have the same color} \\
(c) & \quad :\! : \! \text{\textit{edge}}(X, Y), \text{\textit{col}}(X, C), \text{\textit{col}}(Y, C). \\
\text{% NB: answer sets are subset minimal} & \quad \rightarrow \text{only one color per node}
\end{align*}
\]
Guess and Check (Example 1)

Example (3-col)

**Problem:** Given a graph, assign one color out of 3 colors to each node such that two adjacent nodes have always different colors.

**Input:** a Graph is represented by node(\textunderscore) and edge(\textunderscore, \textunderscore).

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

% discard colorings where adjacent nodes have the same color
\( (c) \) :- edge(X, Y), col(X, C), col(Y, C).

% NB: answer sets are subset minimal \( \rightarrow \) only one color per node
Example (3-col)

**Problem:** Given a graph, assign one color out of 3 colors to each node such that two adjacent nodes have always different colors.

**Input:** a Graph is represented by `node(__)` and `edge(__,__)`.

% guess a coloring for the nodes
\[(r) \text{ col}(X, \text{red}) \mid \text{col}(X, \text{yellow}) \mid \text{col}(X, \text{green}) : \text{node}(X).\]

% discard colorings where adjacent nodes have the same color
\[(c) : \text{edge}(X, Y), \text{col}(X, C), \text{col}(Y, C).\]

% NB: answer sets are subset minimal \(\rightarrow\) only one color per node
Example (3-col)

**Problem:** Given a graph, assign one color out of 3 colors to each node such that two adjacent nodes have always different colors.

**Input:** a Graph is represented by `node(_)` and `edge(_,_)`.

% guess a coloring for the nodes

\[(r) \quad \text{col}(X, \text{red}) \mid \text{col}(X, \text{yellow}) \mid \text{col}(X, \text{green}) \leftarrow \text{node}(X).\]

% discard colorings where adjacent nodes have the same color

\[(c) \leftarrow \text{edge}(X, Y), \text{col}(X, C), \text{col}(Y, C).\]

% NB: answer sets are subset minimal → only one color per node
**Guess, Check and Optimize (Example 2)**

**Example (Traveling Salesman Person)**

**Problem:** Find a path of minimum length in a Weighted Graph beginning at the starting node which contains all nodes of the graph.

**Input:** node( _) and edge(_, _, _), and start(_).

% Guess a path

\[\text{inPath}(X, Y) \mid \text{outPath}(X, Y) :\neg \text{edge}(X, Y, \_).\]

% Ensure that it is Hamiltonian

\[\neg \text{inPath}(X, Y), \text{inPath}(X, Y1), Y <> Y1.\]

\[\neg \text{inPath}(X, Y), \text{inPath}(X1, Y), X <> X1.\]

\[\neg \text{node}(X), \neg \text{reached}(X).\]

\[\neg \text{inPath}(X, Y), \text{start}(Y).\]

\[\text{reached}(X) :\neg \text{reached}(Y), \text{inPath}(Y, X).\]

\[\text{reached}(X) :\neg \text{start}(X).\]

% Minimize the sum of distances

\[\neg \text{inPath}(X, Y), \text{edge}(X, Y, C). [C]\]
Guess, Check and Optimize (Example 2)

Example (Traveling Salesman Person)

**Problem:** Find a path of minimum length in a Weighted Graph beginning at the starting node which contains all nodes of the graph.

**Input:** node(\_\_) and edge(\_, \_, \_), and start(\_\_).

% Guess a path

\[ inPath(X, Y) \mid outPath(X, Y) :\!\!\!: edge(X, Y, \_). \]

% Ensure that it is Hamiltonian

\[ \neg inPath(X, Y), inPath(X, Y1), Y <> Y1. \]

\[ \neg inPath(X, Y), inPath(X1, Y), X <> X1. \]

\[ \neg node(X), \text{not} \text{ reached}(X). \]

\[ \neg inPath(X, Y), \text{start}(Y). \]

\[ reached(X) :\!\!\!: reached(Y), inPath(Y, X). \]

\[ reached(X) :\!\!\!: \text{start}(X). \]

% Minimize the sum of distances

\[ :\!\!\!: inPath(X, Y), edge(X, Y, C). [C] \]
Guess, Check and Optimize (Example 2)

Example (Traveling Salesman Person)

**Problem:** Find a path of **minimum length** in a Weighted Graph beginning at the starting node which contains all nodes of the graph.

**Input:** `node(_,_)` and `edge(_,_,_)`, and `start(_)`.

% Guess a path

\[ \text{inPath}(X, Y) \mid \text{outPath}(X, Y) :\leftarrow \text{edge}(X, Y, _). \]

% Ensure that it is Hamiltonian

\[ \leftarrow \text{inPath}(X, Y), \text{inPath}(X, Y1), Y <> Y1. \]
\[ \leftarrow \text{inPath}(X, Y), \text{inPath}(X1, Y), X <> X1. \]
\[ \leftarrow \text{node}(X), \text{not reached}(X). \]
\[ \leftarrow \text{inPath}(X, Y), \text{start}(Y). \]

\( \text{reached}(X) :\leftarrow \text{reached}(Y), \text{inPath}(Y, X). \)
\( \text{reached}(X) :\leftarrow \text{start}(X). \)

% Minimize the sum of distances

\[ \sim \text{inPath}(X, Y), \text{edge}(X, Y, C). [C] \]
Routing and classification of call-center customers
Call center routing problem

Domain description

- Call centers provide remote assistance to a variety of services
- Front-ends are flooded by a huge number of telephone calls every day
- Customers should be routed to the most appropriate service

Goals: Improve the quality of service

- Reduce the average call response times
- Quickly find solutions for customers
Customer profiling for routing phone calls

- Based on DLV
- Developed by Exeura s.r.l, a spin-off company of the University of Calabria


- In production on call centers of Telecom Italia

Key Ideas

- Classify customer profiles
- Try to anticipate their actual needs
  → Exploit experience of customer care service
Customer classification

Customer’s routing

1. a customer calls the contact center
2. he/she is automatically assigned to a category (based on his/her profile)
3. then routed to an appropriate human operator or automatic responder

Categories based on
- customer behavioral aspects
  - recent history of contacts, telephone calls to the contact center, messages sent to customer assistance, etc.
- basic customer demographics
  - age, residence, type of contract, etc.
Customer classification

Customer’s routing

1. a customer calls the contact center
2. he/she is automatically assigned to a category (based on his/her profile)
3. then routed to an appropriate human operator or automatic responder

Categories based on

- customer behavioral aspects
  - recent history of contacts, telephone calls to the contact center, messages sent to customer assistance, etc.
- basic customer demographics
  - age, residence, type of contract, etc.
ZLog Architecture
Customer classification

Contact center operators define categories

- Customer categories created with an user-friendly user interface
  - Added to the call routing system in real time
  - Automatically translated into ASP rules
  - Fed as input to DLV with the customer data DBs
  - DLV quickly computes the new class of customers

The customer call is routed

- Customer care of the appropriate branch is contacted
- The user is faced with an automatic responder
ZLog interface: Class Definition
ZLog Class Encoding

\( \text{varTipoAbbonato}(CLI) \leftarrow \text{OR1}(CLI). \)

\( \text{OR1}(CLI) \leftarrow \text{AND1}(CLI). \quad \text{OR1}(CLI) \leftarrow \text{AND2}(CLI). \)

\( \text{OR1}(CLI) \leftarrow \text{Abbonati_on_line1}(CLI). \)

\( \text{AND1}(CLI) \leftarrow \text{Clienti_Linee}(CLI, \ldots), \text{not Abbonati_on_line2}(CLI). \)

\( \text{AND2}(CLI) \leftarrow \text{Clienti_Linee1}(CLI), \text{not Abbonati_on_line2}(CLI). \)

\( \text{Abbonati_on_line1}(CLI) \leftarrow \)

\( \quad \text{Abbonati_on_line}(CLI, \ldots, \text{ESITO_OPSC}, \text{ESITO_TGDS}, \ldots), \)

\( \quad \text{ESITO_OPSC} = "2", \text{ESITO_TGDS} = "0". \)

\( \text{Abbonati_on_line2}(CLI) \leftarrow \)

\( \quad \text{Abbonati_on_line}(CLI, \ldots, \text{ESITO_OPSC}, \text{ESITO_TGDS}, \ldots), \)

\( \quad \text{DatiOPSC(ESITO_OPSC)}. \)

\( \text{DatiOPSC(codifica : "11"}. \quad \text{DatiOPSC(codifica : "12"}). \)

\( \quad \text{DatiOPSC(codifica : "13"}. \)

\( \text{Clienti_Linee1}(CLI) \leftarrow \text{Clienti_Linee}(CLI, \ldots, \text{TIPO_CLIENTE}, \text{STATO}, \ldots), \)

\( \quad \text{TIPO_CLIENTE} = "ABB", \text{STATO} = "A". \)
The system is in production

- It runs in a production system at Telecom Italia
- It handles *over one million telephone calls every day*
  - Customer categories are detected in less than 100 ms
  - The system manages over 400 calls/sec.

Users Feedback

- “ZLog made possible huge time savings”
- “ZLog sensibly reduced the average call response times”
- “We improved our customer support quality”
Teambuilding in Gioia Tauro Seaport
Context and Motivation

The Gioia Tauro seaport
- the largest transshipment terminal of the Mediterranean Sea
- main activity: container transhipment [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*
- The impossibility of arranging teams
  $\rightarrow$ contract violations $\rightarrow$ pecuniary sanctions for B.L.G.
Requirements (1)

Team Building Process

1. **Data regarding shoring boats available one day in advance**
   (arrival/departure date, number and kind of vehicles, etc.)

2. **Manager determines requirement on skills (plans)**
   (setting the number of required employees per skill per shift)

3. **Available employees are assigned to shifts**
   (respecting constraints)
Team Building Requirements

- Shift requirements (e.g., number of workers per role)
- Employee contract (e.g. max 36 hours per week, etc.)
- Turnover of heavy/dangerous roles
- Fair distribution of workload
- and others (e.g. preserve crucial skills, etc.)
Team-Building Encoding (simplified)

% Guess the assignment of available employees to shifts in appropriate roles

\( r \) \( \text{assign}(Em, Sh, Sk) \mid nAssign(Em, Sh, Sk) :\neg \text{canBeAssigned}(Em, Sh, Sk). \)

% Workers potentially allocable on the given shift.

\( r_{aux1} \) \( \text{canBeAssigned}(Em, Sh, Sk) :\neg \text{neededEmployees}(Sh, Sk, _), \)
\( \text{hasSkill}(Em, Sk), \neg \text{exceedTimeLimit}(Em, Sh), \)
\( \neg \text{absent}(Em, Sh), \neg \text{excluded}(Em, Sh), \)

% Workers not allocable due to contract constraints.

\( r_{aux2} \) \( \text{exceedTimeLimit}(Em, Sh) :\neg \text{shift}(Sh, _, Dur), \)
\( \text{workedWeeklyHours}(Em, Wh), \neg Dur + Wh > 36. \)

% Similarly for daily hours (max 8h) and weekly overtime (max 12h).

\( r_{aux3} \) \( \text{exceedTimeLimit}(Em, Sh) :\neg ...........
\)
\( r_{aux4} \) \( \text{exceedTimeLimit}(Em, Sh) :\neg ............ \)
Team-Building Encoding (simplified)

% Guess the assignment of available employees to shifts in appropriate roles

\[
(r) \text{assign}(Em, Sh, Sk) \mid nAssign(Em, Sh, Sk) :\neg \text{canBeAssigned}(Em, Sh, Sk).
\]

% Workers potentially allocable on the given shift.

\[
(r_{aux1}) \text{canBeAssigned}(Em, Sh, Sk) :\neg \text{neededEmployees}(Sh, Sk, _),
\text{hasSkill}(Em, Sk), \not\text{exceedTimeLimit}(Em, Sh),
\not\text{absent}(Em, Sh), \not\text{excluded}(Em, Sh),
\]

% Workers not allocable due to contract constraints.

\[
(r_{aux2}) \text{exceedTimeLimit}(Em, Sh) :\neg \text{shift}(Sh, _, Dur),
\text{workedWeeklyHours}(Em, Wh), \not\text{Dur} + Wh > 36.
\]

% Similarly for daily hours (max 8h) and weekly overtime (max 12h).

\[
(r_{aux3}) \text{exceedTimeLimit}(Em, Sh) :\neg ........
\]

\[
(r_{aux4}) \text{exceedTimeLimit}(Em, Sh) :\neg ........
\]
% Discard assignments with a wrong number of employees in some skill.

\((c_1) : \neg \text{shiftPlan}(Sh, Sk, EmpNum, _), \#	ext{count}\{Em : \text{assign}(Em, Sh, Sk)\} \neq EmpNum.\)

% Avoid that an employee covers two roles in the same shift.

\((c_2) : \neg \text{assign}(Em, Sh, Sk1), \text{assign}(Em, Sh, Sk2), Sk1 \neq Sk2.\)
Introduction

Industrial Applications of DLV

Development Tools for ASP

Lessons learned and current developments

F. Ricca

ASP in industrial contexts: Applications and Toolchain
Manual team composition required several hours!

→ costly and risky management task

We developed a Team Builder using ASP

→ the user exploits a friendly User Interface

→ full warranty of respecting all constraints!

Performance

→ 130 employees, 36 meta-plans per week

→ all the employee-allocation constraints were enabled

→ Few seconds to generate a shift plan for one day

→ Some minutes (≈500s) for a complete allocation (one month)
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 $[DGM^{+}15]$, 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 [DGM+15], 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** [DGM+15], 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
ASPIDE: Integrated Development Environment for Answer Set Programming

[FRR11] O. Febbraro, K. Reale, F. Ricca

ASPIDE: Integrated Development Environment for Answer Set Programming

[FRR11] O. Febbraro, K. Reale, F. Ricca

http://www.mat.unical.it/ricca/aspine
Embedding ASP in concrete systems

Existing Application Programming Interfaces (API)
- the DLV Wrapper [Ric03], OntoDLV API [RGS+09]
  - 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
Embedding ASP in concrete systems

Existing Application Programming Interfaces (API)

- the DLV Wrapper [Ric03], OntoDLV API [RGS+09]
  - 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
JASP: integrating Java with Answer Set Programming

[FLGR12] O. Febbraro, N. Leone, G. Grasso, F. Ricca
“JASP: A Framework for Integrating Answer Set Programming with Java.”
Lessons learned (1)

**ASP → model domain + reasoning**
- Executable Specification → Fast prototyping
- Purely Declarative Language → Flexibility, Ease of Maintenance and Extensibility

**ASP applied for solving real-world problems**
- Complex business-logic at a lower (implementation) price
- Develop and test reasoning "on-site" → great advantage!
- Effective implementations, rapid development
- Raw solver performance might not be a problem
Lessons learned (2)

Development tools play a key role

→ Just as for any other programming language
→ Integrate ASP in programming environments
→ *Give the feeling ASP is not only for researchers*

Is ASP a *silver bullet*?

→ Can it be used for everything?
→ Should it be used as a black box?
→ Obtaining the required performance is for experts only?
→ Application-oriented extensions!
Development tools play a key role

→ Just as for any other programming language
→ Integrate ASP in programming environments
→ *Give the feeling ASP is not only for researchers*

Is ASP a *silver bullet*?

→ Can it be used for everything?
→ Should it be used as a black box?
→ Obtaining the required performance is for experts only?
→ *Application-oriented extensions!*
Application-oriented Extensions of ASP solvers

[DGL+16] C. Dodaro and P. Gasteiger and N. Leone and B. Musitsch and F. Ricca and K. Schekotihin
“Combining Answer Set Programming and domain heuristics for solving hard industrial problems (Application Paper)”
In: Theory and Practice of Logic Programming v.16 issue 5-6 (2016)

[CDRS17] B. Cuteri and C. Dodaro and F. Ricca and P. Schüller
“Constraints, lazy constraints, or propagators in ASP solving: An empirical analysis”
In: Theory and Practice of Logic Programming v.17 issue 5-6 (2017)
Domain Heuristics

“The power of problem-specific heuristics turned out to be the key in many applications of problem solvers”

[G. Friedrich at CP/ICLP 2015]
Hard Problems by Siemens

Challenging real-world applications from Siemens

1. The Partner Units Problem (PUP)
   - Railway systems control

2. Combined Configuration Problem (CCP).
   - Abstracts complex problem composed of a set of subproblems
   - Railway interlocking systems, safety automation, and resource distribution

The hardest instances of PUP and CCP are out of reach for state-of-the-art ASP solvers!!
Hard Problems by Siemens

Challenging real-world applications from Siemens

1. The Partner Units Problem (PUP)
   - Railway systems control

2. Combined Configuration Problem (CCP).
   - Abstracts complex problem composed of a set of subproblems
   - Railway interlocking systems, safety automation, and resource distribution

The hardest instances of PUP and CCP are out of reach for state-of-the-art ASP solvers!!
<table>
<thead>
<tr>
<th>Tool</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBC</td>
<td>8</td>
</tr>
<tr>
<td>CPLEX</td>
<td>10</td>
</tr>
<tr>
<td>CP</td>
<td>24</td>
</tr>
<tr>
<td>SAT</td>
<td>25</td>
</tr>
<tr>
<td>VBS-ALT</td>
<td>34</td>
</tr>
</tbody>
</table>
Can we improve the performance of ASP solvers?

1. Try better encodings
   - Requires deep knowledge of language and systems
   - Proposed by the organizers of the latest ASP competition
   - Better performance but still not satisfactory

2. Improve the implementation of the solvers
   - State-of-the-art solvers are very optimized
   - A slow process requiring years of development

3. Push additional domain knowledge in the solving process
   - Domain heuristics are easily provided by domain experts
   - The integration into existing ASP solvers is not obvious
Can we improve the performance of ASP solvers?

1. Try better encodings
   - Requires deep knowledge of language and systems
   - Proposed by the organizers of the latest ASP competition
   - Better performance but still not satisfactory

2. Improve the implementation of the solvers
   - State-of-the-art solvers are very optimized
   - A slow process requiring years of development

3. Push additional domain knowledge in the solving process
   - Domain heuristics are easily provided by domain experts
   - The integration into existing ASP solvers is not obvious
ASP Solving with Domain Heuristics

Extension of Wasp
- Interface for specifying heuristics
- Fast prototyping (Python/Perl)
- Performance oriented (C++)

Experiment with PUP and CPP
- Integration of existing and novel domain heuristics for PUP and CCP into Wasp
- Experimental analysis on real-world instances provided by Siemens
Example of heuristics (1)

# Input:
# * a set of pigeons $P=\{1, \ldots, n\}$, defined by means of the predicate pigeon
# * a set of holes $H=\{1, \ldots, m\}$, defined by means of the predicate hole

\[\text{pigeon}(1) \leftarrow \text{hole}(1) \leftarrow\]
\[\vdots \quad \vdots\]
\[\text{pigeon}(n) \leftarrow \text{hole}(m) \leftarrow\]

# Guess an assignment

\[\text{inHole}(p, h) \leftarrow \text{not} \ \text{outHole}(p, h) \ \forall p \in P, \forall h \in H\]
\[\text{outHole}(p, h) \leftarrow \text{not} \ \text{inHole}(p, h) \ \forall p \in P, \forall h \in H\]

# A hole contains at most one pigeon

\[\leftarrow \text{inHole}(p_i, h), \text{inHole}(p_j, h) \ \forall p_i, p_j \in P \ | \ i \neq j, \forall h \in H\]

# A pigeon is assigned to at most one hole

\[\leftarrow \text{inHole}(p, h_i), \text{inHole}(p, h_j) \ \forall h_i, h_j \in H \ | \ i \neq j, \forall p \in P\]

# A pigeon must be in some hole

\[\text{inSomeHole}(p) \leftarrow \text{inHole}(p, h) \ \forall p \in P, \forall h \in H\]
\[\leftarrow \text{not} \ \text{inSomeHole}(p) \ \forall p \in P\]
Example of heuristics (2)

```python
var = {1: 'false', 'false': 1}
P = []  # list of all pigeon-constants
H = []  # list of all hole-constants

def addedVarName(v, name):
    # invoked when WASP parses the atom table of the gringo numeric format
    global var, H, P
    var.update({v: name, name: v})
    if name.startswith("pigeon"):
        P.append(name[7:-1])
    if name.startswith("hole"):
        H.append(name[5:-1])

def onFinishedParsing():
    # disable simplifications of variables
    return [v for v in var.keys() if isinstance(v, int)]

def choiceVars():  # event onChoiceRequired, invoked when a choice is needed
    global var, H, P
    if len(P) > len(H):
        return [4, 0]  # force incoherence
    # assign pigeon i to hole i
    return [var["inHole(:,s)" % (i, i)] for i in range(1, len(P))]
```
PUP: state of the art

- CLASP
- CLASP (10 threads)
- CLASP portfolio (10 threads)
- CLASPfolio (10 threads)
- MEASP
- VBS
PUP: Wasp + domain heuristics
CCP: state of the art

- CLASP
- CLASP (10 threads)
- CLASP portfolio (10 threads)
- CLASPfolio (10 threads)
- MEASP
- VBS

Time (s)
CCP: Wasp + domain heuristics
External Propagators

Why do we need new propagators?

- Improve the performance of solvers
- Implement language extensions
  - e.g., Aggregates [? ], Constraint ASP [? ? ],
  - e.g., Acyclic constraints [? ], ASPMT [? ]
- Overcome drawbacks of “ground+solve” approach

Challenges

- Hard to integrate new propagators into existing solvers
- The knowledge of internal solver details is needed
Wasp interface for External Propagators

- Implementation
  → Synchronous message passing protocol

- Multi-language support
  - Python and Perl interface requiring no changes to WASP
    → Fast Prototyping
  - C++ interface for performance-oriented implementations
    → Efficient implementation
Propagation functions

Type of propagators

- Eager propagators (e.g., unit)
- Post propagators (e.g., unfounded set, aggregates)
- Lazy propagators (e.g., check answer set)

**Function Propagate(I)**

1. for $\ell \in I$ do $I := I \cup \text{Propagation}(\ell)$;
2. $I' := \text{PostPropagation}(I)$;
3. if $I' \neq \emptyset$ then $I := I \cup I'$; goto 1;
4. return $I$;
WASP interface for propagators (simplified)

- Common methods
  - getLiterals()
  - simplifyAtLevelZero()

- Eager propagators
  - onLiteralTrue(ℓ)
  - getReasonForLiteral(ℓ)
  - onLiteralsUndefined(L)

- Post propagators
  - onLiteralsTrue(L)
  - getReasonForLiteral(ℓ)
  - onLiteralsUndefined(L)

- Lazy propagators
  - checkAnswerSet(I)
  - getReasonsForCheckFailure()
Abduction is a popular formalism for NLU

- Finding explanations for sentences
- Recently modeled and implemented in ASP [Schüller 2016]

Example

Given the text:

“Mary lost her father. She is depressed.”

use background knowledge to understand that

“Mary is depressed because of the death of her father.”
Abduction in NLU can be modeled in ASP

- Formulations for different objective functions [Schüller 2016]
  - Cardinality Minimality (minimal observations)
  - Coherence [Ng, H.T. and Mooney, R.J 1992]
  - Weighted Abduction [Hobbs et al. 1993]

- Pure ASP is not effective [Schüller 2016]
  → Due to the grounding blow-up of few constraints

Can ASP systems be made efficient?

- We experiment with external propagators
  → Avoid the instantiation bottleneck!
- Good results on real-world NLU benchmarks
Abduction in NLU can be modeled in ASP

- Formulations for different objective functions [Schüller 2016]
  - Cardinality Minimality (minimal observations)
  - Coherence [Ng, H.T. and Mooney, R.J 1992]
  - Weighted Abduction [Hobbs et al. 1993]

- Pure ASP is not effective [Schüller 2016]
  → Due to the grounding blow-up of few constraints

Can ASP systems be made efficient?

- We experiment with external propagators
  → Avoid the instantiation bottleneck!
- Good results on real-world NLU benchmarks
Case of study

Abduction in NLU can be modeled in ASP

- Formulations for different objective functions [Schüller 2016]
  - Cardinality Minimality (minimal observations)
  - Coherence [Ng, H.T. and Mooney, R.J 1992]
  - Weighted Abduction [Hobbs et al. 1993]

- Pure ASP is not effective [Schüller 2016]
  → Due to the grounding blow-up of few constraints

Can ASP systems be made efficient?

- We experiment with external propagators
  → Avoid the instantiation bottleneck!
- Good results on real-world NLU benchmarks
Abduction in NLU can be modeled in ASP

- Formulations for different objective functions [Schüller 2016]
  - Cardinality Minimality (minimal observations)
  - Coherence [Ng, H.T. and Mooney, R.J 1992]
  - Weighted Abduction [Hobbs et al. 1993]

- Pure ASP is not effective [Schüller 2016]
  → Due to the grounding blow-up of few constraints

Can ASP systems be made efficient?

- We experiment with external propagators
  → Avoid the instantiation bottleneck!
- Good results on real-world NLU benchmarks
Experiment Setup

Software Setup
- Timeout: 10 minutes
- Memory limit: 5GB
- 50 natural language understanding instances

Compared methods

1. **Constraint**
   - Pure ASP
   - Grounder instantiates all constraints

2. **Lazy Propagator** (in Python)
   - Omit transitivity constraints
   - Lazily instantiate if violation is detected in an answer set candidate

3. **Eager Propagator** (in Python)
   - Simulate inferences
Experimental results: NLU

NLU Benchmark: Number of solved instances and average running time (in seconds).

<table>
<thead>
<tr>
<th>Obj. Func.</th>
<th>WASP</th>
<th>WASP-LAZY</th>
<th>WASP-EAGER</th>
<th>WASP-POST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sol.</td>
<td>avg t</td>
<td>sol.</td>
<td>avg t</td>
</tr>
<tr>
<td>Card.</td>
<td>43</td>
<td>39.7</td>
<td>50</td>
<td>2.3</td>
</tr>
<tr>
<td>Coh.</td>
<td>43</td>
<td>40.1</td>
<td>50</td>
<td>18.5</td>
</tr>
<tr>
<td>W. Abd.</td>
<td>43</td>
<td>49.3</td>
<td>50</td>
<td>26.6</td>
</tr>
</tbody>
</table>

- Eager/Post propagator good performance
- Lazy propagator is the best approach!
- Memory consumption drops from 2GB to 119MB
Conclusion

1. **ASP Applications**
   - Call-center routing, Workforce Management, PUP, CCP, ...

2. **Lessons learned**
   - Rapid development of advanced components
   - Flexibility, easy maintenance, etc.
   - Tools play an important role
   - The golden hammer temptation...

3. **Application-oriented extensions**
   - User-defined domain heuristics and propagators
   - Successfully applied to CCP, PUP, NLU
Thanks for your attention!

Research group at UNICAL and DLVSystem
- Nicola Leone, Mario Alviano, Francesco Calimeri, Gelsomina Catalano, Susanna Cozza, Carmine Dodaro, Onofrio Febbraro, Gianluigi Greco, Giovambattista Ianni, Salvatore Maria Ielpa, Wolfgang Faber, Marco Manna, Alessandra Martello, Kristian Reale, Simona Perri, Giorgio Terracina, Pierfrancesco Veltri

DLVSystem
- A spin-off company developing solutions based on ASP
References


References (cont.)


References (cont.)


References (cont.)


References (cont.)


References (cont.)


References (cont.)


