

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)

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 Σ_2^P/Π_2^P [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

Industrial Applications of ASP

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

Industrial Applications of ASP

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

ASP Basics

The language of ASP is:

Datalog

- + Default negation
- + Disjunction
- + Integrity Constraints
- + Weak Constraints
- + Aggregate atoms

Programming Methodology:

- Guess, Check, Optimize

ASP Syntax

Rule: $\underbrace{a_1 \mid \dots \mid a_n}_{\text{head}} \text{ :- } \underbrace{b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m}_{\text{body}}.$

Atoms and Literals: a_i , b_i , $\text{not } b_i$

Positive Body: b_1, \dots, b_k

Negative Body: $\text{not } b_{k+1}, \dots, \text{not } b_m.$

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: $\underbrace{a_1 \mid \dots \mid a_n}_{\text{head}} \text{ :- } \underbrace{b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m}_{\text{body}}.$

Atoms and Literals: a_i , b_i , $\text{not } b_i$

Positive Body: b_1, \dots, b_k

Negative Body: $\text{not } b_{k+1}, \dots, \text{not } b_m.$

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:

$$\underbrace{a_1 \mid \dots \mid a_n}_{\text{head}} \text{ :- } \underbrace{b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m}_{\text{body}}.$$

Informal Semantics:

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

Informal Semantics

Rule:

$$\underbrace{a_1 \mid \dots \mid a_n}_{\text{head}} \text{ :- } \underbrace{b_1, \dots, b_k, \text{ not } b_{k+1}, \dots, \text{ not } b_m}_{\text{body}}.$$

Informal Semantics:

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

Example (Datalog + Disjunction + Negation)

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

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

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

$\text{:- } \text{parent}(F, S), \text{ not } \text{parent}(S, F).$

Informal Semantics (2)

Example

*isInterestedinASP(john) | isCurious(john) :- attendsASP(john).
attendsASP(john).*

Two (minimal) models encoding two plausible scenarios:

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

Informal Semantics (3)

Constraint:

$\vdash b_1, \dots, b_k, \text{not } b_{k+1}, \dots, \text{not } b_m.$

Informal Semantics:

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

Example

*isInterestedinASP(john) | isCurious(john) \vdash attendsASP(john).
 \vdash hatesASP(john), isInterestedinASP(john).
attendsASP(john). hatesASP(john).*

Only one plausible scenario:

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

Informal Semantics (3)

Constraint:

$$\vdash b_1, \dots, b_k, \text{not } b_{k+1}, \dots, \text{not } b_m.$$

Informal Semantics:

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

Example

isInterestedinASP(john) | isCurious(john) \vdash attendsASP(john).
 \vdash hatesASP(john), isInterestedinASP(john).
attendsASP(john). hatesASP(john).

Only one plausible scenario:

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

Weak Constraints

Weak Constraints:

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

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

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

Weak Constraints

Weak Constraints:

- Express desiderata
- *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 Atom (Informal)

Aggregate atoms: Express functions calculated over sets of elements:

$$L_g <_{op} f\{S\} <_{op} U_g$$

Example

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

The atom is true if the number of male employees is greater than 5 and does not exceed 10.

Aggregate Usage

Example (Team Building)

% An employee is either included in the team or not

$inTeam(I) \mid outTeam(I) \text{ :- } emp(I, Sx, Sk, Sa).$

% The team consists of a certain number of employees

$\text{ :- } nEmp(N), \text{ not } \#count\{I : inTeam(I)\} = N.$

% At least a given number of different skills must be present in the team

$\text{ :- } nSkill(M), \text{ not } \#count\{Sk : emp(I, Sx, Sk, Sa), inTeam(I)\} \leq M.$

% The sum of the salaries of the employees working in the team must not exceed the given budget

$\text{ :- } budget(B), \text{ not } \#sum\{Sa, I : emp(I, Sx, Sk, Sa), inTeam(I)\} \leq B.$

% The salary of each individual employee is within a specified limit

$\text{ :- } maxSal(M), \text{ not } \#max\{Sa : emp(I, Sx, Sk, Sa), inTeam(I)\} \leq M.$

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

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 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*(_) and *edge*(_,_).

% 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 → only one color per node

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*(_) and *edge*(_,_).

% 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 → only one color per node

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*(_) and *edge*(_,_).

% 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 → only one color per node

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*(_) and *edge*(_,_).

% 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 → 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

inPath(X, Y) | *outPath*(X, Y) :- *edge*(X, Y, _).

| Guess

% Ensure that it is Hamiltonian

:- *inPath*(X, Y), *inPath*(X, Y1), Y <> Y1.

:- *inPath*(X, Y), *inPath*(X1, Y), X <> X1.

| Check

:- *node*(X), not *reached*(X).

:- *inPath*(X, Y), *start*(Y).

| Aux. Rules

reached(X) :- *reached*(Y), *inPath*(Y, X).

reached(X) :- *start*(X).

% Minimize the sum of distances

| Optimize

:~ *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

inPath(X, Y) | *outPath*(X, Y) :- *edge*(X, Y, _).

| Guess

% Ensure that it is Hamiltonian

:- *inPath*(X, Y), *inPath*(X, Y1), Y <> Y1.

| Check

:- *inPath*(X, Y), *inPath*(X1, Y), X <> X1.

:- *node*(X), not *reached*(X).

:- *inPath*(X, Y), *start*(Y).

| Aux. Rules

reached(X) :- *reached*(Y), *inPath*(Y, X).

reached(X) :- *start*(X).

% Minimize the sum of distances

| Optimize

~ *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

inPath(X, Y) | *outPath*(X, Y) :- *edge*(X, Y, _).

| Guess

% Ensure that it is Hamiltonian

:- *inPath*(X, Y), *inPath*(X, Y1), Y <> Y1.

:- *inPath*(X, Y), *inPath*(X1, Y), X <> X1.

:- *node*(X), not *reached*(X).

:- *inPath*(X, Y), *start*(Y).

| Check

reached(X) :- *reached*(Y), *inPath*(Y, X).

reached(X) :- *start*(X).

| Aux. Rules

% Minimize the sum of distances

:~ *inPath*(X, Y), *edge*(X, Y, C). [C]

| Optimize

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

The ZLog platform

Customer profiling for routing phone calls

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

`http://www.exeura.eu/en/archives/solution/customer-profiling`

- 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

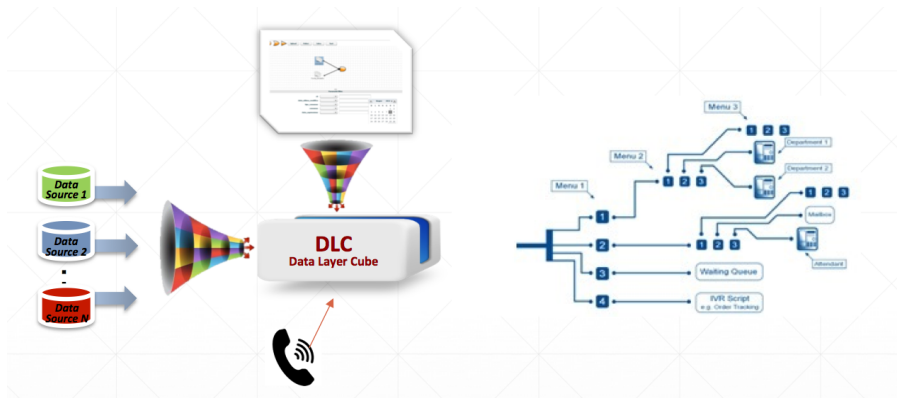
Customer's routing

- 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




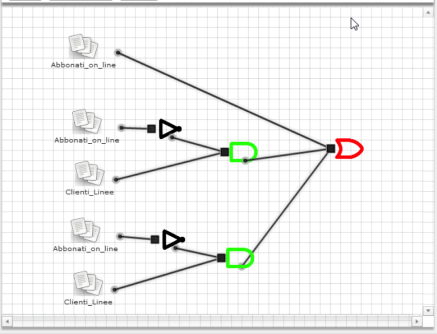
- Customer categories created with an user-friendly user interface
 - Added to the call routing system in real time
 - **Automatically translated into ASP rules**
 - Fed as input to DLV with the customer data DBs
 - **DLV quickly computes the new class of customers**

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

Classi
▼ Contattabilità
Accessi_40915
Accessi_40916
Accessi_40920
Campagne_IVR
Circolanta_Oper_IVR
Email_certificate
Frequent Caller
Multicaller
Recall
SMS_Caring
Servizi_CSP
Servizi_VAS
Tratte_di_chiamata_con_T
Utilizzo_WebCallBack
▼ Caring
Abbonati_on_line
Assistenza_Dealer
Campagne_Push
Clienti_Linee
Credito_Residuo
Esigenze
Linee_Particolari
Offerte
Pending_Issue
Prenotazione_offerte
Pro_routing

Editor
Test Pulisci tutto Salva   

Parametri - Offerte

| Attributo | Operatore | Valore | Raggruppamento |
|-------------------|-----------|--------------|---------------------|
| cli | = | | |
| cod_offerta | = | | Offerte Serie A Tim |
| desc_offerta | = | | |
| data_att_offerta | = | | |
| data_scad_offerta | >= | 30 giorni fa | |
| stato | = | | Attivo e Sospeso |
| canone | = | | |

Cluster pubblici
varMarcNEWMNP
varMarcPREMIUM
varMarcSILVER
varMarcSTDA
varMarcSTDB
varStack_OffCuboVision
varMarcSTDB3
varStack_OffSochi
varTipoAbbonato
varTipoBusiness
varTipoPrepagato
var_ST3AM_IN2-0303_OUT-0
var_ST3AM_IN2-0502_OUT-1

Cluster privati
t1

ZLog Class Encoding

varTipoAbbonato(CLI) :- OR1(CLI).

OR1(CLI) :- AND1(CLI). OR1(CLI) :- AND2(CLI).

OR1(CLI) :- Abbonati_on_line1(CLI).

AND1(CLI) :- Clienti_Linee(CLI, ...), not Abbonati_on_line2(CLI).

AND2(CLI) :- Clienti_Linee1(CLI), not Abbonati_on_line2(CLI).

Abbonati_on_line1(CLI) :-

*Abbonati_on_line(CLI, ..., ESITO_OPSC, ESITO_TGDS, ...),
 ESITO_OPSC = "2", ESITO_TGDS = "0".*

Abbonati_on_line2(CLI) :-

*Abbonati_on_line(CLI, ..., ESITO_OPSC, ESITO_TGDS, ...),
 DatiOPSC(ESITO_OPSC).*

DatiOPSC(codifica : "11"). DatiOPSC(codifica : "12").

DatiOPSC(codifica : "13").

*Clienti_Linee1(CLI) :- Clienti_Linee(CLI, ..., TIPO_CLIENTE, STATO, ...),
 TIPO_CLIENTE = "ABB", STATO = "A".*

Deployment and Performance

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 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*
- The impossibility of arranging teams
→ contract violations → 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)

Requirements (2)

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)

-1-

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

(r) $assign(Em, Sh, Sk) \mid nAssign(Em, Sh, Sk) :- canBeAssigned(Em, Sh, Sk).$

% Workers potentially allocable on the given shift.

(r_{aux_1}) $canBeAssigned(Em, Sh, Sk) :- neededEmployees(Sh, Sk, _),$
 $hasSkill(Em, Sk), not exceedTimeLimit(Em, Sh),$
 $not absent(Em, Sh), not excluded(Em, Sh),$

% Workers not allocable due to contract constraints.

(r_{aux_2}) $exceedTimeLimit(Em, Sh) :- shift(Sh, _, Dur),$
 $workedWeeklyHours(Em, Wh), not Dur + Wh > 36.$

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

(r_{aux_3}) $exceedTimeLimit(Em, Sh) :- \dots\dots\dots$

(r_{aux_4}) $exceedTimeLimit(Em, Sh) :- \dots\dots\dots$

Team-Building Encoding (simplified)

-1-

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

(r) $assign(Em, Sh, Sk) \mid nAssign(Em, Sh, Sk) :- canBeAssigned(Em, Sh, Sk).$

% Workers potentially allocable on the given shift.

(r_{aux_1}) $canBeAssigned(Em, Sh, Sk) :- neededEmployees(Sh, Sk, _),$
 $hasSkill(Em, Sk), not exceedTimeLimit(Em, Sh),$
 $not absent(Em, Sh), not excluded(Em, Sh),$

% Workers not allocable due to contract constraints.

(r_{aux_2}) $exceedTimeLimit(Em, Sh) :- shift(Sh, _, Dur),$
 $workedWeeklyHours(Em, Wh), not Dur + Wh > 36.$

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

(r_{aux_3}) $exceedTimeLimit(Em, Sh) :- \dots\dots\dots$

(r_{aux_4}) $exceedTimeLimit(Em, Sh) :- \dots\dots\dots$

Team-Building Encoding (simplified)

-1-

% Discard assignments with a wrong number of employees in some skill.

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

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

$(c_2) :- assign(Em, Sh, Sk1), assign(Em, Sh, Sk2), Sk1 \neq Sk2.$



gennaio
febbraio
marzo
aprile
maggio
giugno
luglio
agosto
settembre

- Settimana - 36
- Settimana - 37
 - lunedì - 07/09/2009
 - martedì - 08/09/2009
 - mercoledì - 09/09/2009
 - Velasquez (18)
 - giovedì - 10/09/2009
 - venerdì - 11/09/2009
 - Autoroute (9)
 - sabato - 12/09/2009
 - domenica - 13/09/2009
- Settimana - 38
- Settimana - 39
- Settimana - 40

Calcolo Team (Finished at 11:00)

[Calcolo Team: Calcolo Team](#)

Calcolo Team (Finished at 11:05)

[Calcolo Team: Calcolo Team](#)

driver high_heavy lasher lasher_coord magazzinieri mobile_data_entry parker pd quality_checker service_person taxi_driver trucks yard_mde

| Data | giuseppe | oreste | antonio |
|------------|----------|--------|---------|
| 09/09/2009 | | 1 | 1 |
| 11/09/2009 | 1 | | |

Logistic

Tipologia Turno

▼ Turno

Descrizioni delle proprietà del turno

Nome Turno Data Orario

▼ Turno

Descrizioni delle proprietà del turno

☒ No PausaInizio pausa Durata turno Volumi Lavorazione

▼ Turno Raddoppio

▼ Mansioni Richieste

Totale mansioni richieste: 9

D HH L LC M MDE P Pd QC SP TD T Y/M

| Nome | Cognome | Mansione |
|----------|------------|-----------------|
| giuseppe | de germano | quality_checker |

| Nome | Cognome | Mansione |
|----------|------------|----------|
| giuseppe | de germano | |

| Nome | Cognome | Mansione |
|------------|--------------|-----------------|
| maurizio | lasher | lasher |
| giuseppe | de germano | quality_checker |
| rocco | barbano | driver |
| gianfranco | ella | driver |
| giuseppe | scattolaccia | driver |
| oreste | di franco | driver |
| francesco | piro | driver |
| valeriano | gilio | taxi_driver |
| fabio | curatò | lasher_coord |

| Nome | Cognome | Mansione |
|-----------|------------|----------|
| giuseppe | barbano | driver |
| rocco | barbano | driver |
| salvatore | barbano | driver |
| francesco | barbano | driver |
| gianluca | costantini | driver |
| natale | costantini | driver |
| antonio | costantini | driver |
| fabio | curatò | driver |
| giuseppe | de niro | driver |
| antonio | di franco | driver |
| oreste | di franco | driver |
| daniele | ella | driver |
| salvatore | gilio | driver |
| marco | gilio | driver |
| cinzia | lasher | driver |

ASP-Based Team-builder

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)

The need for programming tools

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.” In: Proc. of LPNMR 2011 (2011)

[DGM⁺15] C. Dodaro, P. Gasteiger, B. Musitsch, F. Ricca, K. Shchekotykhin

“Interactive debugging of non-ground ASP programs.” In: Proc. of LPNMR 2015 (2015)

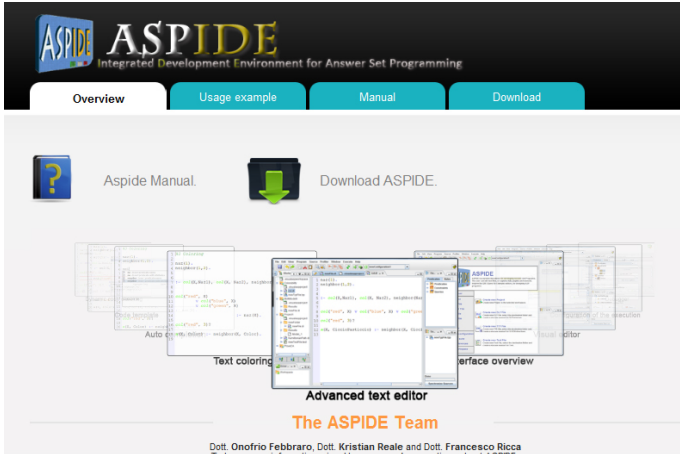
ASPIDE: Integrated Development Environment for Answer Set Programming

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

“ASPIDE: Integrated Development Environment for Answer Set Programming.” In: Proc. of LPNMR 2011 (2011)

[DGM⁺15] C. Dodaro, P. Gasteiger, B. Musitsch, F. Ricca, K. Shchekotykhin
“Interactive debugging of non-ground ASP programs.” In: Proc. of LPNMR 2015 (2015)

<http://www.mat.unical.it/ricca/asptide>



The image shows the ASPIDE website interface. At the top, the ASPIDE logo is displayed with the text "ASPIDe Integrated Development Environment for Answer Set Programming". Below the logo are four navigation buttons: "Overview", "Usage example", "Manual", and "Download". The "Overview" button is currently selected. Below the navigation bar, there are two main sections: "Asptide Manual." with a question mark icon, and "Download ASPIDE." with a green download arrow icon. Below these sections, there is a collage of images showing the ASPIDE IDE interface, including code editors, a debugger, and a visual editor. The text "Advanced text editor" is overlaid on the collage. At the bottom, the text "The ASPIDE Team" is displayed in orange, followed by the names "Dott. Onofrio Febraro, Dott. Kristian Reale and Dott. Francesco Ricca".

ASPIDe Integrated Development Environment for Answer Set Programming

Overview Usage example Manual Download

Asptide Manual. Download ASPIDE.

Advanced text editor

The ASPIDE Team

Dott. Onofrio Febraro, Dott. Kristian Reale and Dott. Francesco Ricca

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 an 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 an 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.”

In: Proc. of KR 2012 (2012)

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!**

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!**

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!!

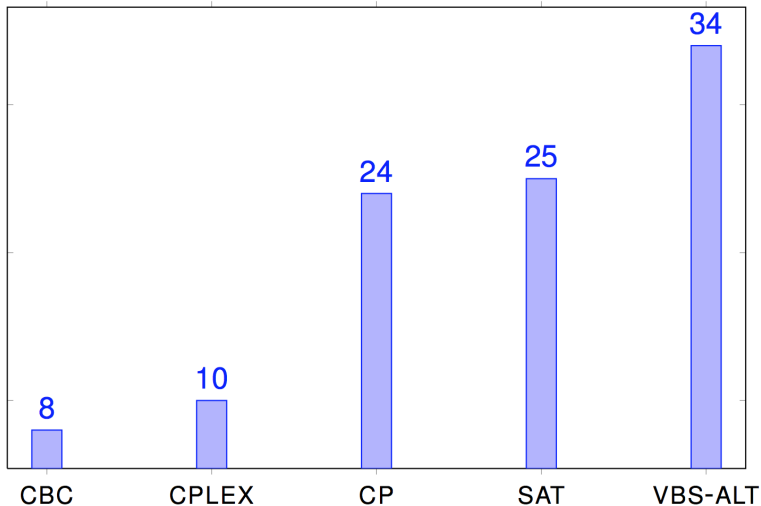
Hard Problems by Siemens

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!!

PUP: Previous attempts cfr. Aschinger et al. [ADF⁺11]



Improve Performance

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

Improve Performance

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  $P=\{1,\dots,n\}$ , defined by means of the predicate pigeon
# * a set of holes  $H=\{1,\dots,m\}$ , defined by means of the predicate hole
pigeon(1) ←      hole(1) ←
    ⋮
    ⋮
pigeon(n) ←      hole(m) ←

# Guess an assignment
inHole( $p,h$ ) ← not outHole( $p,h$ )  $\forall p \in P, \forall h \in H$ 
outHole( $p,h$ ) ← not inHole( $p,h$ )  $\forall p \in P, \forall h \in H$ 

# A hole contains at most one pigeon
← inHole( $p_i,h$ ), inHole( $p_j,h$ )  $\forall p_i,p_j \in P \mid i \neq j, \forall h \in H$ 

# A pigeon is assigned to at most one hole
← inHole( $p,h_i$ ), inHole( $p,h_j$ )  $\forall h_i,h_j \in H \mid i \neq j, \forall p \in P$ 

# A pigeon must be in some hole
inSomeHole( $p$ ) ← inHole( $p,h$ )  $\forall p \in P, \forall h \in H$ 
← not inSomeHole( $p$ )  $\forall p \in P$ 

```

Example of heuristics (2)

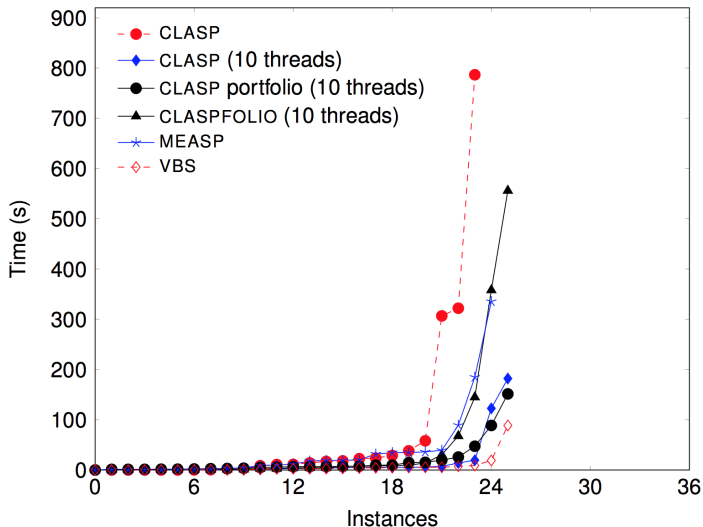
```
var = {1: 'false', 'false': 1}
P = [] # list of all pigeon-constants
H = [] # list of all hole-constants

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])

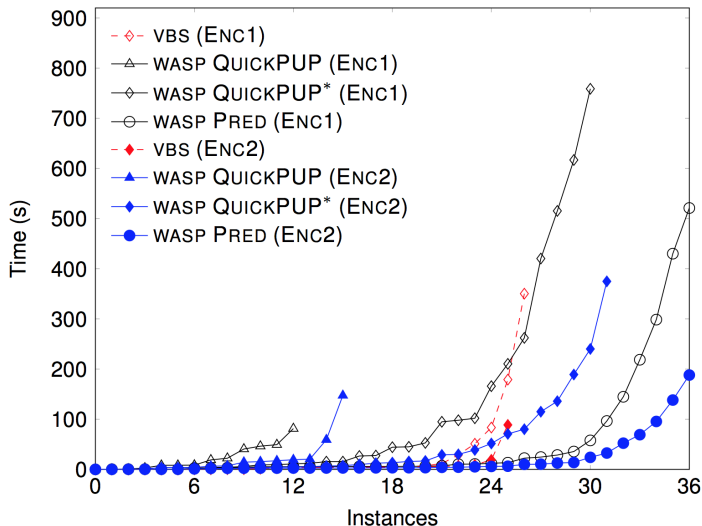
def onFinishedParsing():
    # disable simplifications of variables
    return [v for v in var.keys() if isinstance(v, int)]

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,%s)" % (i, i)] for i in range(1, len(P))]
```

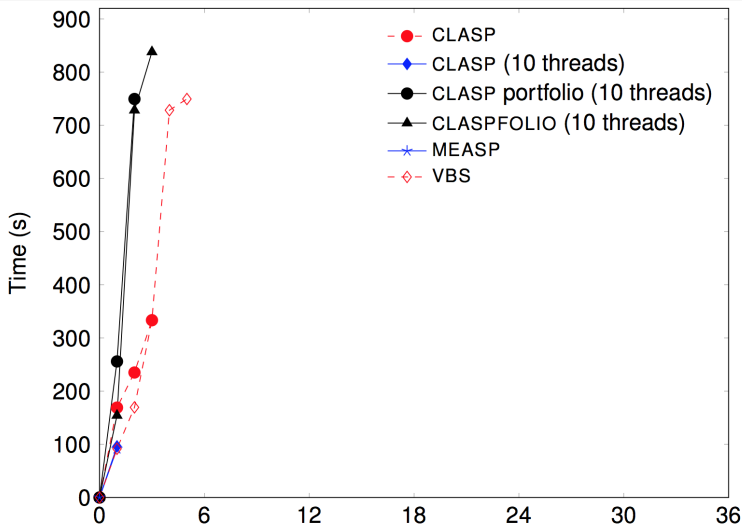
PUP: state of the art



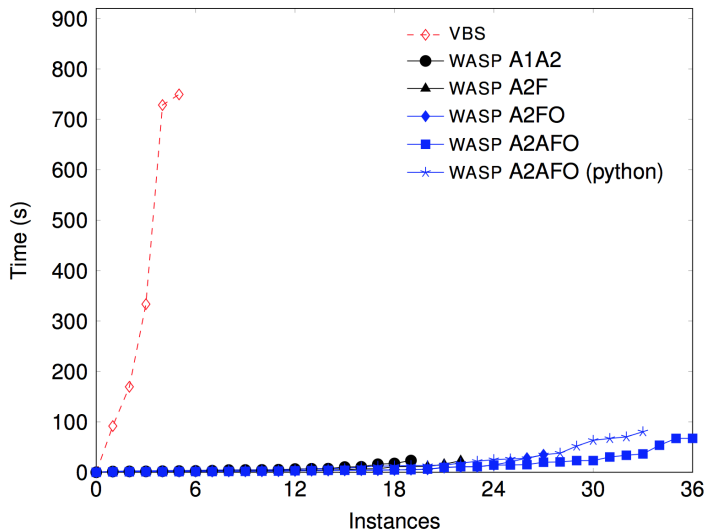
PUP: Wasp + domain heuristics



CCP: state of the art



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., Aciclicity 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

External Propagators in WASP

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 $\text{Propagate}(I)$

```
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()`

Case of study

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.”

Case of study

(2)

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

(2)

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

(2)

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

(2)

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).

| Obj. Func. | WASP | | WASP-LAZY | | WASP-EAGER | | WASP-POST | |
|------------|------|-------|-----------|-------|------------|-------|-----------|-------|
| | sol. | avg t | sol. | avg t | sol. | avg t | sol. | avg t |
| Card. | 43 | 39.7 | 50 | 2.3 | 50 | 4.3 | 50 | 3.3 |
| Coh. | 43 | 40.1 | 50 | 18.5 | 50 | 8.8 | 50 | 6.3 |
| W. Abd. | 43 | 49.3 | 50 | 26.6 | 49 | 66.1 | 50 | 62.6 |

- 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

Acknowledgments

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

- [ADF⁺11] Markus Aschinger, Conrad Drescher, Gerhard Friedrich, Georg Gottlob, Peter Jeavons, Anna Ryabokon, and Evgenij Thorstensen. Optimization methods for the partner units problem. In Tobias Achterberg and J. Christopher Beck, editors, Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - 8th International Conference, CPAIOR 2011, Berlin, Germany, May 23-27, 2011. Proceedings, volume 6697 of Lecture Notes in Computer Science, pages 4–19. Springer, 2011.
- [ADLR15] Mario Alviano, Carmine Dodaro, Nicola Leone, and Francesco Ricca. Advances in WASP. In LPNMR 2015, pages 40–54, 2015.
- [CDRS17] Bernardo Cuteri, Carmine Dodaro, Francesco Ricca, and Peter Schüller. Constraints, lazy constraints, or propagators in ASP solving: An empirical analysis. TPLP, 17(5-6):780–799, 2017.

References (cont.)

- [CHO⁺09] Ozan Caldiran, Kadir Haspalamutgil, Abdullah Ok, Can Palaz, Esra Erdem, and Volkan Patoglu. Bridging the gap between high-level reasoning and low-level control. In LPNMR, pages 342–354, 2009.
- [CIR⁺11] Francesco Calimeri, Giovambattista Ianni, Francesco Ricca, Mario Alviano, Annamaria Bria, Gelsomina Catalano, Susanna Cozza, Wolfgang Faber, Onofrio Febbraro, Nicola Leone, Marco Manna, Alessandra Martello, Claudio Panetta, Simona Perri, Kristian Reale, Maria Carmela Santoro, Marco Sirianni, Giorgio Terracina, and Pierfrancesco Veltri. The third answer set programming competition: Preliminary report of the system competition track. In LPNMR, volume 6645 of Lecture Notes in Computer Science, pages 388–403. Springer, 2011.
- [DEGV01] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and Expressive Power of Logic Programming. ACM Computing Surveys, 33(3):374–425, 2001.

References (cont.)

- [DGH09] James P. Delgrande, Torsten Grote, and Aaron Hunter. A general approach to the verification of cryptographic protocols using answer set programming. In LPNMR, pages 355–367, 2009.
- [DGL⁺16] Carmine Dodaro, Philip Gasteiger, Nicola Leone, Benjamin Musitsch, Francesco Ricca, and Konstantin Schekotihin. Combining answer set programming and domain heuristics for solving hard industrial problems (application paper). TPLP, 16(5-6):653–669, 2016.
- [DGM⁺15] Carmine Dodaro, Philip Gasteiger, Benjamin Musitsch, Francesco Ricca, and Kostyantyn Shchekotykhin. Interactive debugging of non-ground asp programs. In LPNMR, volume 9345 of LNCS, page (to appear), 2015.
- [DLNR15] Carmine Dodaro, Nicola Leone, Barbara Nardi, and Francesco Ricca. Allotment problem in travel industry: A solution based on ASP. In Balder ten Cate and Alessandra Mileo, editors, RR 2015, volume 9209 of LNCS, pages 77–92. Springer, 2015.

References (cont.)

- [EEB10] Halit Erdogan, Esra Erdem, and Olivier Bodenreider. Exploiting umls semantics for checking semantic consistency among umls concepts. In In Proc. of MedInfo, 2010.
- [EFLP99] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. The diagnosis frontend of the dl原因 system. AI Commun., 12(1-2):99–111, 1999.
- [EGM97] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. ACM Trans. Database Syst., 22(3):364–418, 1997.
- [EIST06] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In ESWC, pages 273–287, 2006.
- [FLGR12] Onofrio Febbraro, Nicola Leone, Giovanni Grasso, and Francesco Ricca. Jasp: A framework for integrating answer set programming with java. In KR. AAAI Press, 2012.

References (cont.)

- [FRR11] Onofrio Febbraro, Kristian Reale, and Francesco Ricca. Aspide: Integrated development environment for answer set programming. In LPNMR, volume 6645 of Lecture Notes in Computer Science, pages 317–330. Springer, 2011.
- [GKNS07] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. Conflict-driven answer set solving. In Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07), pages 386–392. Morgan Kaufmann Publishers, January 2007.
- [GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. New Generation Comput., 9(3/4):365–386, 1991.

References (cont.)

- [GLMR11] Giovanni Grasso, Nicola Leone, Marco Manna, and Francesco Ricca. ASP at work: Spin-off and applications of the DLV system. In Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning - Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday, volume 6565 of LNCS, pages 432–451. Springer, 2011.
- [GNA13] Terracina Giorgio, Leone Nicola, and Martello Alessandra. Logic-based techniques for data cleaning: an application to the italian national healthcare system. LPNMR, 2013.
- [Lif02] Vladimir Lifschitz. Answer set programming and plan generation. Artif. Intell., 138(1-2):39–54, 2002.
- [LM04] Yuliya Lierler and Marco Maratea. Cmodels-2: SAT-based Answer Set Solver Enhanced to Non-tight Programs. In Vladimir Lifschitz and Ilkka Niemelä, editors, Proceedings of the 7th International Conference on Logic Programming and volume 2923 of LNAI, pages 346–350. Springer, January 2004.

References (cont.)

- [LPF⁺06] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. TOCL, 7(3):499–562, July 2006.
- [LR15] Nicola Leone and Francesco Ricca. Answer set programming: A tour from the basics to advanced development tools and industrial applications. In Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures, volume 9203 of Lecture Notes in Computer Science, pages 308–326. Springer, 2015.
- [RDG⁺10] Francesco Ricca, Antonella Dimasi, Giovanni Grasso, Salvatore Maria Ielpa, Salvatore Iiritano, Marco Manna, and Nicola Leone. A logic-based system for e-tourism. Fundam. Inform., 105(1-2):35–55, 2010.

References (cont.)

- [RGA⁺12] Francesco Ricca, Giovanni Grasso, Mario Alviano, Marco Manna, Vincenzino Lio, Salvatore Iiritano, and Nicola Leone. Team-building with answer set programming in the gioia-tauro seaport. TPLP, 12(3):361–381, 2012.
- [RGS⁺09] Francesco Ricca, Lorenzo Gallucci, Roman Schindlauer, Tina Dell’Armi, Giovanni Grasso, and Nicola Leone. Ontodlv: An asp-based system for enterprise ontologies. J. Log. Comput., 19(4):643–670, 2009.
- [Ric03] Francesco Ricca. The dlv java wrapper. In 2003 Joint Conference on Declarative Programming, AGP-2003, Reggio Calabria, Italy, September 3-5, 2003, pages 263–274, 2003.
- [Sak11] Chiaki Sakama. Dishonest reasoning by abduction. In IJCAI, pages 1063–1064, 2011.

References (cont.)

- [SN99] Timo Soininen and Ilkka Niemelä. Developing a declarative rule language for applications in product configuration. In PADL, pages 305–319, 1999.
- [WMD08] Johan Wittocx, Maarten Mariën, and Marc Denecker. The IDP system: a model expansion system for an extension of classical logic. In Marc Denecker, editor, Logic and Search, Computation of Structures from Declarative Descriptions (LaSh 2008), pages 153–165, Leuven, Belgium, November 2008.