Answer Set Programming for the Semantic Web

Tutorial

Thomas Eiter, Roman Schindlauer (TU Wien)
Giovambattista Ianni (TU Wien, Univ. della Calabria)
Axel Polleres (Univ. Rey Juan Carlos, Madrid)

Supported by IST REWERSE, FWF Project P17212-N04, CICyT project TIC-2003-9001-C02.

Unit 3 – ASP: State of the Art and Applications

G. Ianni

Dipartimento di Matematica - Università della Calabria

European Semantic Web Conference 2006

Unit Outline

A very active field

Major Scientific Events have ASP as hot topic

- Intl. Workshop on ASP ('01, '03 and '05)
- LPNMR, NMR, JELIA
- Special Issue on Answer Set Programming (ASP) in AMAI
- Working group on Answer Set Programming (WASP, 15+ nodes)

Mature Solvers

- DLV [35], Smodels [68]
- ASSAT, Cmodels, dcs, DeRes, DisLog, DisLop, NoMoRe, aspps, SLG

A very active field

Major Scientific Events have ASP as hot topic

- Intl. Workshop on ASP ('01, '03 and '05)
- LPNMR, NMR, JELIA
- Special Issue on Answer Set Programming (ASP) in AMAI
- Working group on Answer Set Programming (WASP, 15+ nodes)

Mature Solvers

- DLV [35], Smodels [68]
- ASSAT, Cmodels, dcs, DeRes, DisLog, DisLop, NoMoRe, aspps, SLG

ASP Points of strength

Totally declarative

Order of rules and atoms do not matter. You can ignore how the solver operates

Decidable

Prototypes started from Datalog without function symbols. Extensions keep decidability.

Monotonic and nonmonotonic

Negation as failure, as well as classic ("with strong semantics") negation

Nondeterministic

You can specify a set of possible worlds ("guesses") you want. Dealing with uncertainty is thus enabled.

ASP Points of strength

Totally declarative

Order of rules and atoms do not matter. You can ignore how the solver operates

Decidable

Prototypes started from Datalog without function symbols. Extensions keep decidability.

Monotonic and nonmonotonic

Negation as failure, as well as classic (“with strong semantics”) negation

Nondeterministic

You can specify a set of possible worlds (“guesses”) you want. Dealing with uncertainty is thus enabled.

ASP Points of strength

Totally declarative

Order of rules and atoms do not matter. You can ignore how the solver operates

Decidable

Prototypes started from Datalog without function symbols. Extensions keep decidability.

Monotonic and nonmonotonic

Negation as failure, as well as classic ("with strong semantics") negation

Nondeterministic

You can specify a set of possible worlds ("guesses") you want. Dealing with uncertainty is thus enabled.

ASP Points of strength

Totally declarative

Order of rules and atoms do not matter. You can ignore how the solver operates

Decidable

Prototypes started from Datalog without function symbols. Extensions keep decidability.

Monotonic and nonmonotonic

Negation as failure, as well as classic ("with strong semantics") negation

Nondeterministic

You can specify a set of possible worlds ("guesses") you want. Dealing with uncertainty is thus enabled.

ASP Points of strength - 2

Versatile

Weak and Soft constraints, useful special constructs with well-defined formal semantics

Scalable

Can compete with top-down solvers now

Interoperable

- External built-ins, External predicates
- DLV Java API and ODBC Interface
- RuleML schema for program exchange

Note that the price of each achievement in terms of research work is high in the context of ASP (full declarativity is a big design constraint), but it pays off

ASP Points of strength - 2

Versatile

Weak and Soft constraints, useful special constructs with well-defined formal semantics

Scalable

Can compete with top-down solvers now

Interoperable

- External built-ins, External predicates
- DLV Java API and ODBC Interface
- RuleML schema for program exchange

Note that the price of each achievement in terms of research work is high in the context of ASP (full declarativity is a big design constraint), but it pays off

ASP Points of strength - 2

Versatile

Weak and Soft constraints, useful special constructs with well-defined formal semantics

Scalable

Can compete with top-down solvers now

Interoperable

- External built-ins, External predicates
- DLV Java API and ODBC Interface
- RuleML schema for program exchange

Note that the price of each achievement in terms of research work is high in the context of ASP (full declarativity is a big design constraint), but it pays off

ASP Points of strength - 2

Versatile

Weak and Soft constraints, useful special constructs with well-defined formal semantics

Scalable

Can compete with top-down solvers now

Interoperable

- External built-ins, External predicates
- DLV Java API and ODBC Interface
- RuleML schema for program exchange

Note that the price of each achievement in terms of research work is high in the context of ASP (full declarativity is a big design constraint), but it pays off

Current state-of-the-art

Semantics

- Introduction of Function Symbols [71, 14, 7]
- Introduction of various forms of *Generalized Quantifiers* (e.g. Aggregates [32, 55, 63, 26])
- Study of equivalence [25], and debuggers [30, 8]

Scalability

- Intelligent grounders, magic sets [51, 18]
- New heuristics for model generation [33, 31]
- Parallel execution [38]
- Intelligent reductions to SAT [37]

Applications

Hot Areas (non-complete list)

- Configuration/composition,
- Information integration,
- Security analysis,
- Agent systems,
- Semantic Web (see Units 4–6),
- Planning.

ASP is very well tailored at modelling problems that fits the G-C-O approach and need fast prototyping. In an increasing number of cases, ASP technologies can be kept in release versions of softwares they are embedded in.

Applications

Hot Areas (non-complete list)

- Configuration/composition,
- Information integration,
- Security analysis,
- Agent systems,
- Semantic Web (see Units 4–6),
- Planning.

ASP is very well tailored at modelling problems that fits the G-C-O approach and need fast prototyping. In an increasing number of cases, ASP technologies can be kept in release versions of softwares they are embedded in.

Applications

Hot Areas (non-complete list)

- Configuration/composition,
- Information integration,
- Security analysis,
- Agent systems,
- Semantic Web (see Units 4–6),
- Planning.

ASP is very well tailored at modelling problems that fits the G-C-O approach and need fast prototyping. In an increasing number of cases, ASP technologies can be kept in release versions of softwares they are embedded in.

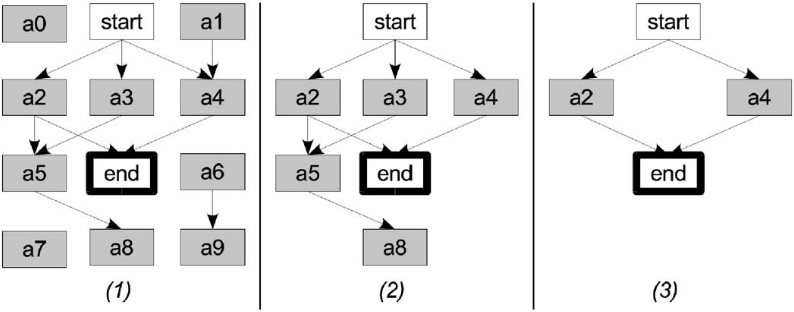
Applications

Hot Areas (non-complete list)

- Configuration/composition,
- Information integration,
- Security analysis,
- Agent systems,
- Semantic Web (see Units 4–6),
- Planning.

ASP is very well tailored at modelling problems that fits the G-C-O approach and need fast prototyping. In an increasing number of cases, ASP technologies can be kept in release versions of softwares they are embedded in.

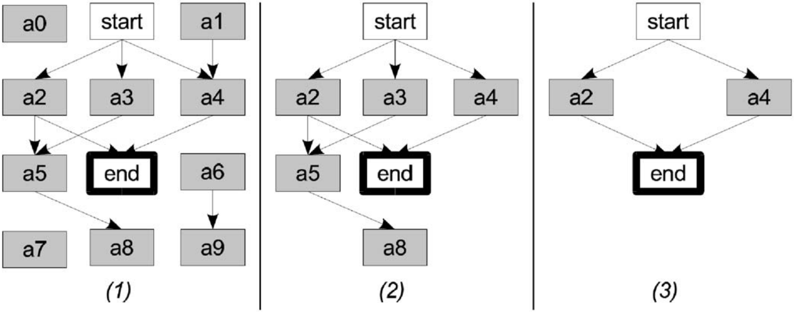
A Web Service Composition problem



Legenda

- Frame = Web Service
- Boxed White Frame = Final Goal
- $a \longrightarrow b$ = Output of a fulfills input preconditions for b

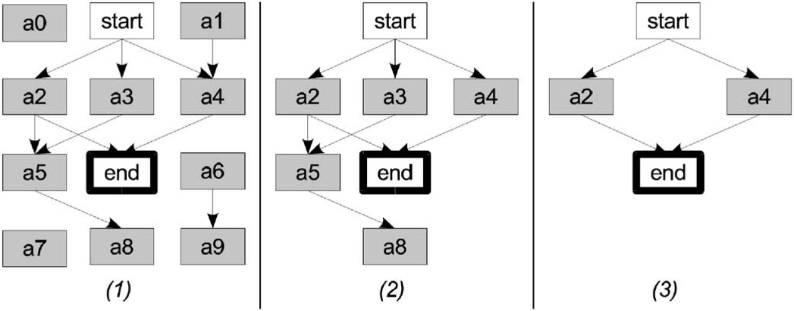
A Web Service Composition problem - 2



Some assumption

- Arrows are statically given.
- But they can come from any chosen semantic entailment.
- Also conjunctive conditions are possible (not shown).

A Web Service Composition problem - 3



ASP role

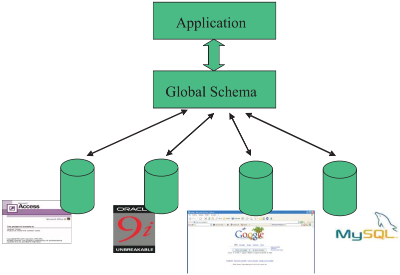
- To design whatever strategy for execution plan generation.
- One can use Guess, Check, and Optimize methodology.
- [64] won the EEE-Web'05 WS contest.

Data Integration Systems

- Offer uniform access to a set of heterogeneous sources
- The representation provided to the user is called global schema
- The user is freed from the knowledge about data location and format

When the user issues a query over the global schema, the system:

- determines which sources to query and how
- issues suitable queries to the sources
- assembles the results and provides the answer

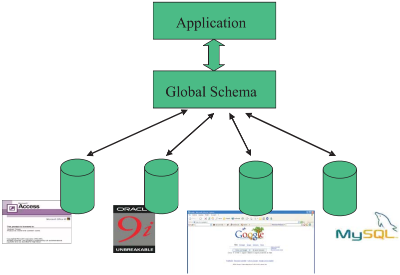


Data Integration Systems

- Offer uniform access to a set of heterogeneous sources
- The representation provided to the user is called global schema
- The user is freed from the knowledge about data location and format

When the user issues a query over the global schema, the system:

- determines which sources to query and how
- issues suitable queries to the sources
- assembles the results and provides the answer

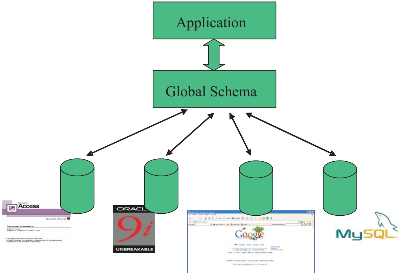


Data Integration Systems

- Offer uniform access to a set of heterogeneous sources
- The representation provided to the user is called global schema
- The user is freed from the knowledge about data location and format

When the user issues a query over the global schema, the system:

- determines which sources to query and how
- issues suitable queries to the sources
- assembles the results and provides the answer

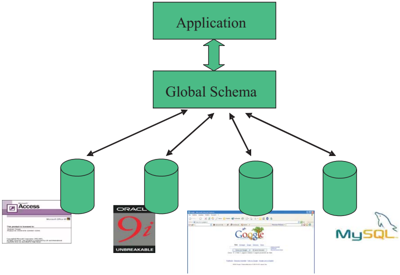


Data Integration Systems

- Offer uniform access to a set of heterogeneous sources
- The representation provided to the user is called global schema
- The user is freed from the knowledge about data location and format

When the user issues a query over the global schema, the system:

- determines which sources to query and how
- issues suitable queries to the sources
- assembles the results and provides the answer

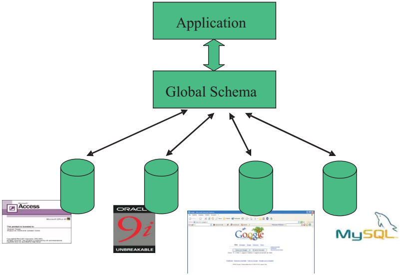


Data Integration Systems

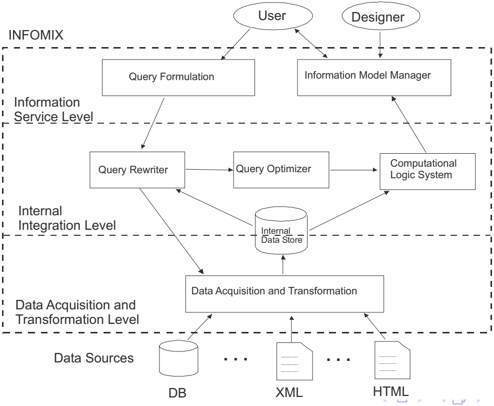
- Offer uniform access to a set of heterogeneous sources
- The representation provided to the user is called global schema
- The user is freed from the knowledge about data location and format

When the user issues a query over the global schema, the system:

- determines which sources to query and how
- issues suitable queries to the sources
- assembles the results and provides the answer

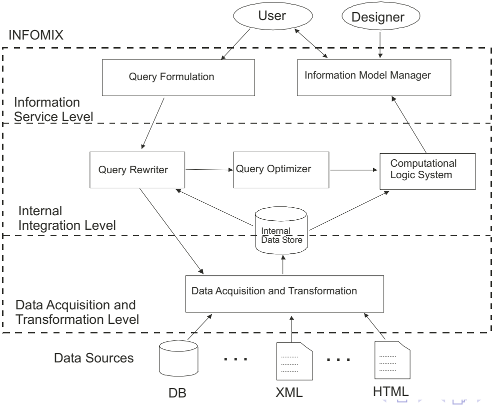


The INFOMIX architecture



The INFOMIX architecture

Three Layers:

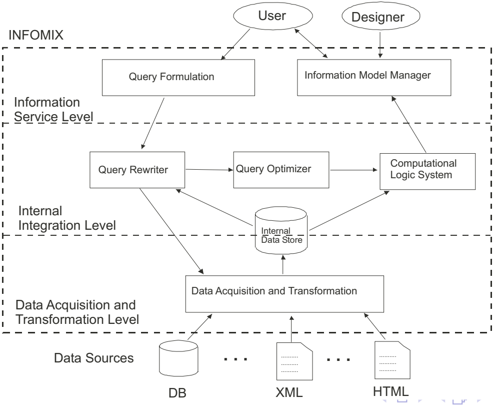


The INFOMIX architecture

Three Layers:

Extraction:

Data Acquisition and Transformation

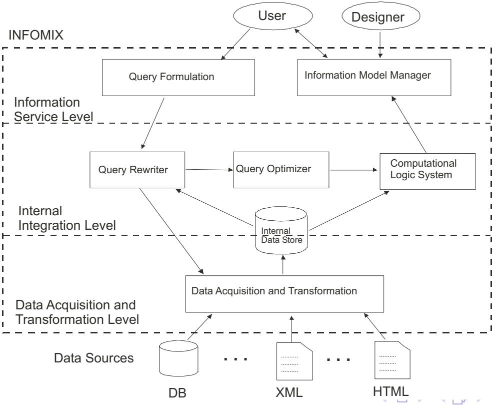


The INFOMIX architecture

Three Layers:

Processing:

Internal Integration Level

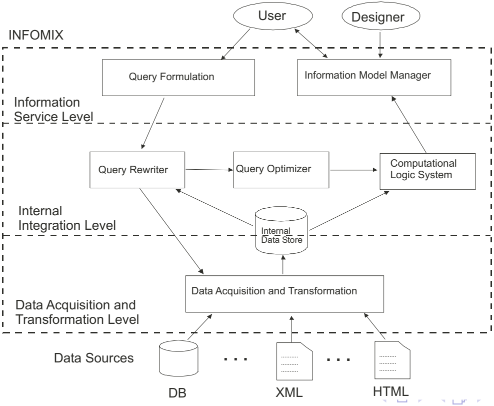


The INFOMIX architecture

Three Layers:

Frontend:

Information Service Level



Design Time

A designer specifies sources and mappings from sources to the global schema

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

$\Longrightarrow$

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

$\Longrightarrow$

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

GAV = Global as view

Design Time

A designer specifies sources and mappings from sources to the global schema

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

$\Rightarrow$

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

$\Rightarrow$

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

GAV = Global as view

Design Time

A designer specifies sources and mappings from sources to the global schema

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

$\Rightarrow$

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

$\Rightarrow$

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

GAV = Global as view

Design Time - 2

The designer can specify also constraints on the global schema, e.g. `KeyConstraint(g,1)`

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

Conflict!!

Design Time - 2

The designer can specify also constraints on the global schema, e.g. `KeyConstraint(g,1)`

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

$\Rightarrow$

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

$\Rightarrow$

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

Conflict!!

Design Time - 2

The designer can specify also constraints on the global schema, e.g.
`KeyConstraint(g,1)`

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

$\Longrightarrow$

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

$\Longrightarrow$

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

Conflict!!

Design Time - 2

The designer can specify also constraints on the global schema, e.g. `KeyConstraint(g,1)`

Data Sources

Ps1:

X	Y
1	2
2	3

Ps2:

X	Y
1	3
4	5

$\Rightarrow$

GAV Mapping

```
g(X,Y) :- ps1(X,Y).
g(X,Y) :- ps2(X,Y).
```

$\Rightarrow$

Global Schema

g:

X	Y
1	3
1	2
2	3
4	5

Conflict!!

Run Time

When a query is submitted, this has to be unfolded to the sources and a merging program has to be processed. But query answering under constraints is a *NP*-hard problem also in the simpler settings.

Two possible repairs

g:

X	Y
1	3
1	2
2	3
4	5

$\Longrightarrow$

Definite answers (cautious reasoning)

g:

X	Y
2	3
4	5

Idea: The query answering and conflict repair strategy can be programmed with ASP

Run Time

When a query is submitted, this has to be unfolded to the sources and a merging program has to be processed. But query answering under constraints is a *NP*-hard problem also in the simpler settings.

Two possible repairs

g:

X	Y
1	3
1	2
2	3
4	5

$\Longrightarrow$

Definite answers (cautious reasoning)

g:

X	Y
2	3
4	5

Idea: The query answering and conflict repair strategy can be programmed with ASP

Run Time

When a query is submitted, this has to be unfolded to the sources and a merging program has to be processed. But query answering under constraints is a *NP*-hard problem also in the simpler settings.

Two possible repairs

g:

X	Y
1	3
1	2
2	3
4	5

$\Longrightarrow$

Definite answers (cautious reasoning)

g:

X	Y
2	3
4	5

Idea: The query answering and conflict repair strategy can be programmed with ASP

Run Time

When a query is submitted, this has to be unfolded to the sources and a merging program has to be processed. But query answering under constraints is a *NP*-hard problem also in the simpler settings.

Two possible repairs

g:

X	Y
1	3
1	2
2	3
4	5

$\Rightarrow$

Definite answers (cautious reasoning)

g:

X	Y
2	3
4	5

Idea: The query answering and conflict repair strategy can be programmed with ASP

How ASP comes into play

Bad News:

- Sources contain a huge amount of data
- Evaluating a co-NP hard problem is unfeasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

A brute force approach would consider k rules and $n = 2^k$ minimal models

Ground program

```
p(1) v q(1) :- a(1).
...
p(k) v q(k) :- a(k).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1), p(2), \ldots, p(k-1), p(k)\} \cup D$
$M_2 : \{p(1), p(2), \ldots, p(k-1), q(k)\} \cup D$
...
$M_n : \{q(1), q(2), \ldots, q(k-1), q(k)\} \cup D$

How ASP comes into play

Bad News:

- Sources contain a huge amount of data
- Evaluating a co-NP hard problem is unfeasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

A brute force approach would consider k rules and $n = 2^k$ minimal models

Ground program

```
p(1) v q(1) :- a(1).
...
p(k) v q(k) :- a(k).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1), p(2), \ldots, p(k-1), p(k)\} \cup D$
$M_2 : \{p(1), p(2), \ldots, p(k-1), q(k)\} \cup D$
...
$M_n : \{q(1), q(2), \ldots, q(k-1), q(k)\} \cup D$

How ASP comes into play

Bad News:

- Sources contain a huge amount of data
- Evaluating a co-NP hard problem is unfeasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

A brute force approach would consider k rules and $n = 2^k$ minimal models

Ground program

```
p(1) v q(1) :- a(1).
...
p(k) v q(k) :- a(k).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1), p(2), \ldots, p(k-1), p(k)\} \cup D$
$M_2 : \{p(1), p(2), \ldots, p(k-1), q(k)\} \cup D$
$\ldots$
$M_n : \{q(1), q(2), \ldots, q(k-1), q(k)\} \cup D$

How ASP comes into play

Bad News:

- Sources contain a huge amount of data
- Evaluating a co-NP hard problem is unfeasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

A brute force approach would consider k rules and $n = 2^k$ minimal models

Ground program

```
p(1) v q(1) :- a(1).
...
p(k) v q(k) :- a(k).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1), p(2), \ldots, p(k-1), p(k)\} \cup D$
$M_2 : \{p(1), p(2), \ldots, p(k-1), q(k)\} \cup D$
...
$M_n : \{q(1), q(2), \ldots, q(k-1), q(k)\} \cup D$

Magic Sets

Given

Program P,
Query Q

$\Longrightarrow$ **Magic Sets**

Program rewriting

$\Longrightarrow$ **Rewritten Program**

$MS(Q, P)$

Magic sets:

- focus on the subset of P which is relevant for Q
- push down the query constants, to eliminate rule instances which cannot contribute to the derivation of Q
- simulate the top-down evaluation of Q

Applicability:

- Positive Programs (in the literature)
- Disjunctive programs (INFOMIX achievement)
- Programs with un-stratified negation (INFOMIX achievement)

Magic Sets

Given

Program P,
Query Q

$\Longrightarrow$

Magic Sets

Program rewriting

$\Longrightarrow$

Rewritten Program

$MS(Q, P)$

Magic sets:

- focus on the subset of P which is relevant for Q
- push down the query constants, to eliminate rule instances which cannot contribute to the derivation of Q
- simulate the top-down evaluation of Q

Applicability:

- Positive Programs (in the literature)
- Disjunctive programs (INFOMIX achievement)
- Programs with un-stratified negation (INFOMIX achievement)

Magic Sets

Given
Program P,
Query Q

$\Longrightarrow$

Magic Sets
Program rewriting

$\Longrightarrow$

Rewritten Program
$MS(Q, P)$

Magic sets:

- focus on the subset of P which is relevant for Q
- push down the query constants, to eliminate rule instances which cannot contribute to the derivation of Q
- simulate the top-down evaluation of Q

Applicability:

- Positive Programs (in the literature)
- Disjunctive programs (INFOMIX achievement)
- Programs with un-stratified negation (INFOMIX achievement)

Magic Sets

Given

Program P,
Query Q

$\Rightarrow$

Magic Sets

Program rewriting

$\Rightarrow$

Rewritten Program

$MS(Q, P)$

Magic sets:

- focus on the subset of P which is relevant for Q
- push down the query constants, to eliminate rule instances which cannot contribute to the derivation of Q
- simulate the top-down evaluation of Q

Applicability:

- Positive Programs (in the literature)
- Disjunctive programs (INFOMIX achievement)
- Programs with un-stratified negation (INFOMIX achievement)

Magic Sets

Given

Program P,
Query Q

$\Longrightarrow$

Magic Sets

Program rewriting

$\Longrightarrow$

Rewritten Program

$MS(Q, P)$

Magic sets:

- focus on the subset of P which is relevant for Q
- push down the query constants, to eliminate rule instances which cannot contribute to the derivation of Q
- simulate the top-down evaluation of Q

Applicability:

- Positive Programs (in the literature)
- Disjunctive programs (INFOMIX achievement)
- Programs with un-stratified negation (INFOMIX achievement)

Magic Sets

Given
Program P,
Query Q

$\Longrightarrow$

Magic Sets
Program rewriting

$\Longrightarrow$

Rewritten Program
$MS(Q, P)$

Magic sets:

- focus on the subset of P which is relevant for Q
- push down the query constants, to eliminate rule instances which cannot contribute to the derivation of Q
- simulate the top-down evaluation of Q

Applicability:

- Positive Programs (in the literature)
- Disjunctive programs (INFOMIX achievement)
- Programs with un-stratified negation (INFOMIX achievement)

Magic Sets approach has been extended to the ASP

Good News:

- Not all the data is necessary for answering user queries
- Magic sets can focus on relevant data
- Problems theoretically untractable become feasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

An intelligent approach could consider only one ground rule and 2 models

Ground program

```
p(1) v q(1) :- a(1).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1)\} \cup D$
$M_2 : \{q(1)\} \cup D$

Magic Sets approach has been extended to the ASP

Good News:

- Not all the data is necessary for answering user queries
- Magic sets can focus on relevant data
- Problems theoretically untractable become feasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

An intelligent approach could consider only one ground rule and 2 models

Ground program

```
p(1) v q(1) :- a(1).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1)\} \cup D$
$M_2 : \{q(1)\} \cup D$

Magic Sets approach has been extended to the ASP

Good News:

- Not all the data is necessary for answering user queries
- Magic sets can focus on relevant data
- Problems theoretically untractable become feasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:

```
D = { a(1),a(2), ...,a(k) }
```

Query: p(1)?

An intelligent approach could consider only one ground rule and 2 models

Ground program

```
p(1) v q(1) :- a(1).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1)\} \cup D$
$M_2 : \{q(1)\} \cup D$

Magic Sets approach has been extended to the ASP

Good News:

- Not all the data is necessary for answering user queries
- Magic sets can focus on relevant data
- Problems theoretically untractable become feasible

A simple program

```
p(X) v q(X) :- a(X).
```

Database and Query

Database:
`D = { a(1),a(2), ...,a(k) }`
Query: p(1)?

An intelligent approach could consider only one ground rule and 2 models

Ground program

```
p(1) v q(1) :- a(1).
```

$\Longrightarrow$

Stable Models

$M_1 : \{p(1)\} \cup D$
$M_2 : \{q(1)\} \cup D$

The lesson of INFOMIX

- The INFOMIX prototype is the most expressive current system for consistent query answering under incompleteness
- Expresses the full range of queries (not only fragments), with different sorts of constraints (KDs, IDs, EDs)
- Rich Data Acquisition and Transformation Layer
- Fruitful use of computational logic (proof of concept)
- Experimental results are encouraging, scalability feasible
- Further efforts for optimizing data access (cf. constant pushing)
- Tighter coupling between CL system and relational engine

Live Demo

Now, let's play with the Live Demo

http://www.mat.unical.it/infomix