External Computations and Interoperability in the new DLV Grounder
F. Calimeri1,2, D. Fuscà1, S. Perri1, J. Zangari1
1Department of Mathematics and Computer Science, University of Calabria, Rende (CS), Italy
2DLVSysterm Srl, Rende (CS), Italy
{calimeri, fusca, perri, zangari}@mat.unical.it

ANSWER SET PROGRAMMING

- Powerful declarative paradigm for KRR
- State-of-the-art systems perform two steps: Grounding & Solving
- Reliable and efficient ASP systems available:
  - Monolithic Systems: DLV, CLINGO
  - Standalone grounders and solvers:
    - LPARSE, GRINGO, I-DLV
    - SMODELS, CLASP, WASP
- Several ASP applications in academy and industry

Design and Implementation of the I-DLV Grounder: Optimizations, Customizability and Interoperability
DOCTORAL CONSORTIUM - J. Zangari

- Fully supports ASP-Core-2 syntax
- Able to interoperate with state-of-the-art solvers
- Generalizes and improves DLV grounding strategy
- Incorporates new optimization techniques
- Devoted to flexibility and customizability
- Full-fledged Deductive Database system
- Along with the solver WASP, I-DLV has been integrated in the new version of DLV recently released

I-DLV NOVEL FEATURES: MOTIVATIONS

- Ease the interoperability and integration with external systems
- Accommodate external sources of computation and value invention within ASP programs

I-DLV supports explicit native directives for connecting with DBs.

Relational Databases
Data can be imported/exported from/to relational DBs via SQL queries.

Graph Databases
Data can be imported via SPARQL queries, both from local DBs in RDF/XML files and remote SPARQL end-points.

INTEROPERABILITY MEANS

I-DLV supports explicit native directives for connecting with DBs.

Example: Reversing a String

reverseWord(Y) :- word(X), &reverse(X;Y).
def reverse(s):
    return :::-1
    ← oracle Python function for &reverse.

KRR WITH NOVEL FEATURES

Consider the problem of automatically assigning a score to students after an assessment test: given a list of students, a list of topics, and a set of questions regarding the given topics along with corresponding student answers, we want to determine the score of each student. It could be modelled as follows:

#correctAnswers(St, To, N) :- topic(To), student(St), N = #count(QID: question(QID, To, Ca), answer(St, QID, Ca)).

#wrongAnswers(St, To, N) :- topic(To), student(St), N = #count(QID: question(QID, To, Ca), answer(St, QID, Ans), Ans!=Ca).

topicScore(St, To, Sc) :- correctAnswers(St, To, Cn), &assignScore(To, Cn, &assignScore(To, Cn, Cn)).
testScore(St, Sc) :- student(St), Sc = #sum(Sc: topicScore(St, To, Sc)).

where the semantics of &assignScore could be defined as:

def assignScore(topic, numCorrectAns, numWrongAns):
    if(topic==`ComputerScience`` or topic==`Mathematics``):
        return numCorrectAns*2+numWrongAns*0.5
    return numCorrectAns*0.5

ADDITIONAL DETAILS

I-DLV is an open source project, further details are available at:...