Poster Session ACAI'09 Belfast

ASPARTIX: A System for Computing Different Argumentation Semantics in Answer-Set Programming

Sarah Alice Gaggl

Vienna University of Technology
Institute of Information Systems
Database and Artificial Intelligence Group
Supervisor: Stefan Woltran

FAKULTÄT FÜR !INFORMATIK

Wiener Wissenschafts-, Forschungs- und Technologiefonds

supported under grant ICT 08-028

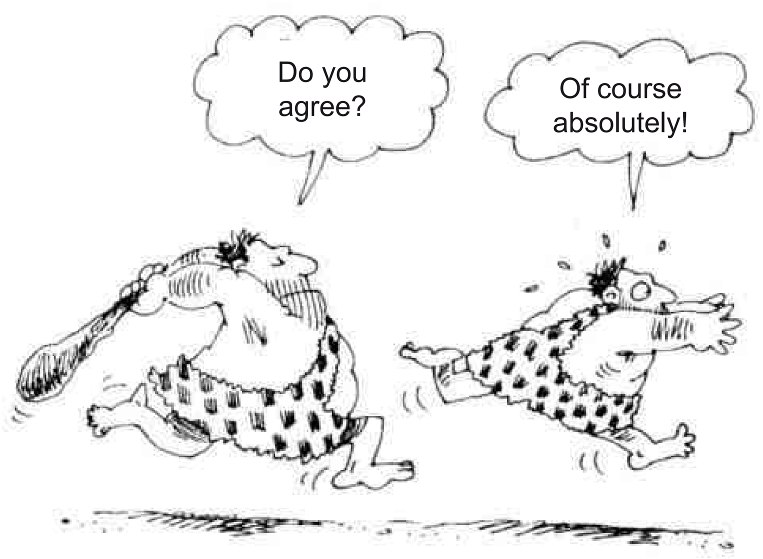
Motivation

- ***Argumentation*** has become one of the central issues in ***Artificial Intelligence (AI)***.
- ***Argumentation frameworks (AFs)*** formalize statements together with a relation for attack:
  - Selecting acceptable subsets of arguments allows to solve conflicts between statements.
  - A broad range of semantics exists.
  - Many problems associated to *AFs* are *intractable*.
  - Applications fields include Multi-Agent Systems and Law Research.
- ***General system required!***

Main Contributions

- ***ASPARTIX*** is capable to compute admissible, preferred, stable, semi-stable, ideal, complete, and grounded extensions for Dung´s original framework, PAFs, VAFs, and BAFs using ***ASP***.
- Can be used by researchers to compare different argumentation semantics on concrete examples within a uniform setting.
- We use ***DLV*** to compute the desired semantics via *fixed* datalog encodings.
- The input is the only part depending on the actual AF to process (in contrast to most previous work).
- The encodings are adequate from the complexity point of view.

Answer-Set Programing (ASP)



- Models of program represent solutions of problem.
- Separate problem specification and input data.
- Disjunctive logic programs with constraints: compact and easily maintainable representation.
- ***Guess&Check*** methodology: first generate the search space, then rule out wrong solutions.
- Efficient systems (DLV) exist.

Argumentation Frameworks

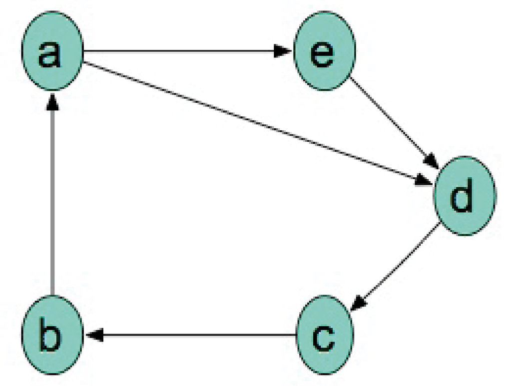
- An ***argumentation framework (AF)*** is a pair (A,R), where A is a set of arguments and R is a binary relation denoting attacks. The pair (a,b) *in* R means that a attacks b.
- A set S of arguments ***defeats*** b, if there is an a *in* S, s.t. (a,b) *in* R. An argument a *in* A is ***defended*** by a set S iff, for each b *in* A, it holds that, if (b,a) *in* R, then S defeats b.

Semantics

- A set S of arguments is ***conflict-free,*** if there are no arguments a and b *in* S, such that a attacks b. We denote the collection of conflict-free sets by $cf(AF)$.
- A conflict-free set S is ***admissible,*** if each argument a *in* S is defended by S. We denote the collection of admissible sets by $adm(AF)$.
- A set S of arguments is a ***stable extension***, if S *in* $cf(AF)$ and each a *in* $A \setminus S$ is defeated by S. We denote the collection of stable extensions by $stable(AF)$.
- An admissible set S is a ***preferred extension***, if it is maximal with respect to set inclusion.

Example

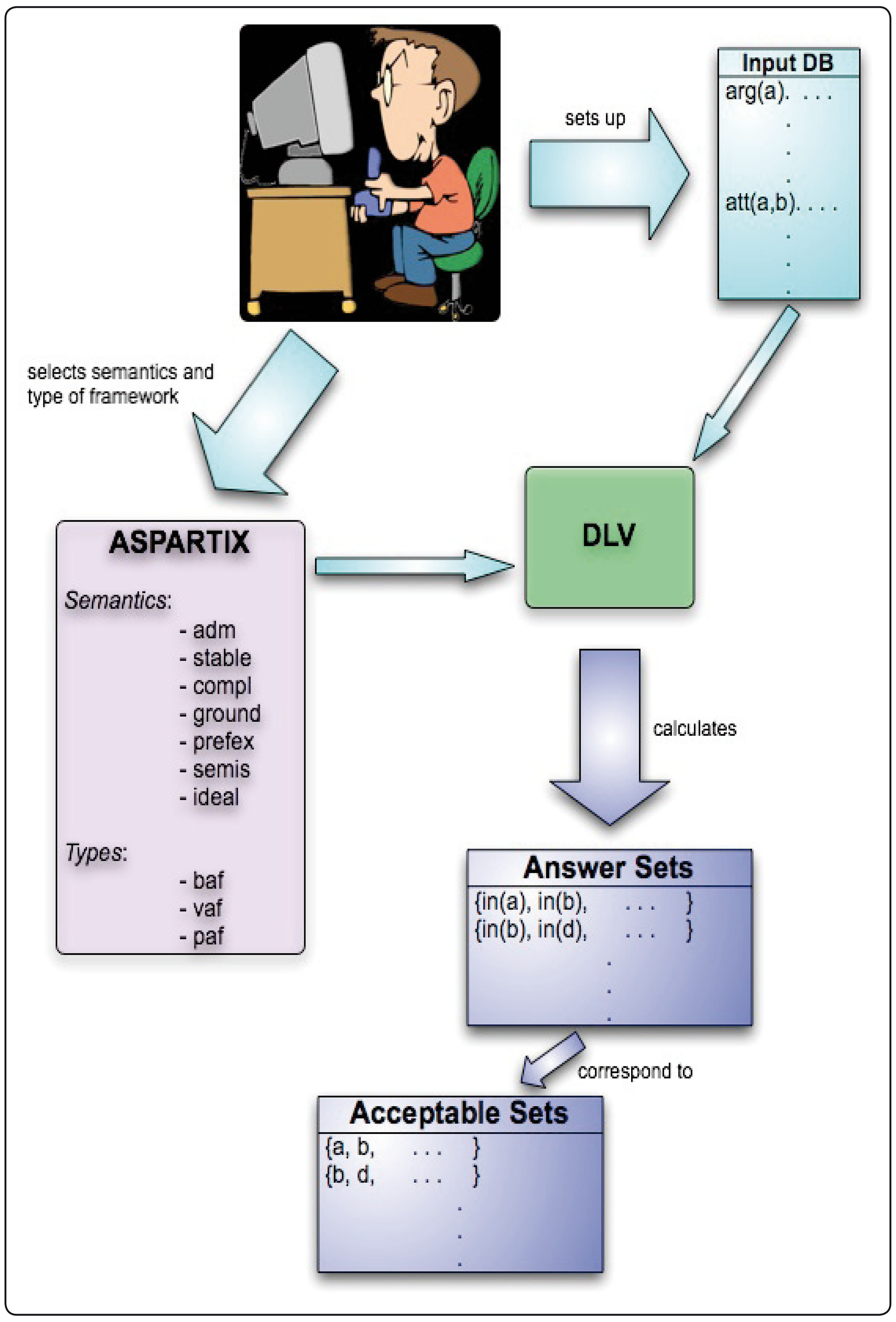
Let $AF=(A,R)$, be an AF with $A=\{a,b,c,d,e\}$ and $R=\{(a,e),(a,d),(b,a),(c,b),(d,c),(e,d)\}$. We obtain $adm(AF)=\{\{\},\{a,c\}\}$, $stable(AF)=prefex(AF)=\{a,c\}$.



Framework Types

- There exist several extensions of AFs like ***preference-based AFs (PAFs), value-based AFs (VAFs),*** and ***bipolar AFs (BAFs).***

System Architecture



Encodings

Input Database (DB) for $AF=(A,R)$

$DB = \{$`arg(a)`$\mid a \text{ in } A\} \cup \{$`defeat(a,b)`$\mid (a,b) \text{ in } R\}$.

Conflict-free Guess

$P_{cf} = \{$
```
in(X) :- not out(X), arg(X);
out(X) :- not in(X), arg(X);
:- in(X), in(Y), defeat(X,Y)}.
```

⇒ We guess all possible solutions via the predicates `in\1` and `out\1`. Solutions with conflicting arguments are ruled out.

Stable Extensions

$P_{stable} = P_{cf} \cup \{$
```
defeated(X) :-in(Y),defeat(Y,X);
:- out(X), not defeated(X)}.
```

Admissible Extensions

$P_{adm} = P_{cf} \cup \{$
```
defeated(X) :- in(Y), defeat(Y,X);
:- in(X), defeat(Y,X), not defeated(Y)}.
```

⇒ Guesses which are not stable (resp. admissible) are ruled out via constraints.

Preferred Extensions

$P_{satpref} = \{$
```
inN(X) v outN(X) :- out(X); inN(X) :- in(X);
sat :- eq;
sat :- inN(X), inN(Y), defeat(X,Y);
sat :- inN(X), outN(Y), defeat(Y,X), undefeated(Y);
inN(X) :- sat, arg(X); out(X) :- sat, arg(X);
:- not sat }.
```

$P_{prefex} = P_{adm} \cup P_{eq} \cup P_{undefeated} \cup P_{satpref}$

⇒ First, we compute the admissible extensions, then a second guess checks for maximality. We use a saturation technique to identify those solutions, where the second guess equals the first one, or is not admissible. (Predicates `eq\0` and `undefeated\1` are computed in additional modules P_{eq} and $P_{undefeated}$.)

Result: Answer sets of the encodings are in a *one-to-one correspondence* to the extensions of the resp. semantics.

Future Work

- Implementation of further semantics: e.g. CF2, resolution-based, meta-attacks, etc.
- Web application of ASPARTIX including a graphical representation of the output.
- Experimental evaluation of the system.

Contact: gaggl@dbai.tuwien.ac.at
System page: http://www.dbai.tuwien.ac.at/staff/gaggl/systempage/

Project page: http://www.dbai.tuwien.ac.at/research/project/argumentation
Joint work with Uwe Egly and Stefan Woltran