Towards a General Argumentation System Based on Answer-set Programming

Sarah Alice Gaggl

Institute of Information Systems, Vienna University of Technology

Edinburgh — July 21, 2010
Legal Reasoning

Sarah A. Gaggl, TU Vienna

General Argumentation System Based on ASP
Social Networks
Argumentation Frameworks (AFs)

- AFs provide a formalism for a compact representation and evaluation of such scenarios.
- More complex semantics, especially in combination with an increasing amount of data, requires an automated computation of such solutions.
- Most of these problems are intractable, so implementing dedicated systems from the scratch is not the best idea.
- Logic Programming (LP), in particular Answer-set Programming (ASP), turned out to be adequate to solve problems associated to AFs.
- We use ASP to design the system ASPARTIX for the evaluation of several approaches how to deal with AFs.
1. Introduction to Abstract Argumentation Frameworks
2. ASP Encoding
3. ASPARTIX - System Demonstration
Abstract Argumentation Frameworks

- First introduced by Phan Minh Dung in 1995.
- AFs provide a formal way of dealing with conflicting knowledge.
- Represent arguments together with a binary attack relation.
- Conflicts are solved via semantics (admissible, preferred, stable).
- They can be represented as directed graphs.

More formally

An argumentation framework (AF) is a pair \((A, R)\), where

- \(A\) is a set of arguments
- \(R \subseteq A \times A\) is a relation representing “attacks” (“defeats”)
Semantics

**Conflict-free**

Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is said to be conflict-free (in $F$), if there are no $a, b \in S$, such that $(a, b) \in R$. We denote the collection of sets which are conflict-free (in $F$) by $cf(F)$.

**Example**

\[
\begin{align*}
 cf(F) &= \{\{a, c\}\},
\end{align*}
\]
Conflict-free

Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is said to be conflict-free (in $F$), if there are no $a, b \in S$, such that $(a, b) \in R$. We denote the collection of sets which are conflict-free (in $F$) by $cf(F)$.

Example

$cf(F) = \{\{a, c\}, \{a, d\}\}.$
Conflict-free

Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is said to be conflict-free (in $F$), if there are no $a, b \in S$, such that $(a, b) \in R$. We denote the collection of sets which are conflict-free (in $F$) by $cf(F)$.

Example

\[
    cf(F) = \{ \{a, c\}, \{a, d\}, \{b, d\} \},
\]
Conflict-free

Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is said to be conflict-free (in $F$), if there are no $a, b \in S$, such that $(a, b) \in R$. We denote the collection of sets which are conflict-free (in $F$) by $cf(F)$.

Example

$$cf(F) = \{\{a, c\}, \{a, d\}, \{b, d\}, \{a\}, \{b\}, \{c\}, \{d\}, \emptyset\}$$
Stable Extension

Given an AF \((A, R)\). A set \(S \subseteq A\) is stable in \(F\), if
- \(S\) is conflict-free in \(F\)
- for each \(a \in A \setminus S\), there exists a \(b \in S\), such that \((b, a) \in R\).

Example

\[\text{stable}(F) = \{\{a, c\}\},\]
Semantics cont.

**Stable Extension**

Given an AF \((A, R)\). A set \(S \subseteq A\) is **stable** in \(F\), if

- \(S\) is conflict-free in \(F\)
- for each \(a \in A \setminus S\), there exists a \(b \in S\), such that \((b, a) \in R\).

**Example**

\[
\begin{align*}
&\text{stable}(F) = \{\{a, e\}, \{a, d\}\}, \\
\end{align*}
\]
Stable Extension

Given an AF \((A, R)\). A set \(S \subseteq A\) is **stable** in \(F\), if

- \(S\) is conflict-free in \(F\)
- for each \(a \in A \setminus S\), there exists a \(b \in S\), such that \((b, a) \in R\).

**Example**

\[ \text{stable}(F) = \{\{a, c\}, \{a, d\}, \{b, d\}\} \]
Stable Extension

Given an AF \((A, R)\). A set \(S \subseteq A\) is \(\text{stable}\) in \(F\), if

- \(S\) is conflict-free in \(F\)
- for each \(a \in A \setminus S\), there exists a \(b \in S\), such that \((b, a) \in R\).

Example

\[
\text{stable}(F) = \{ \{a, c\}, \{a, d\}, \{b, d\}, \{a\}, \{b\}, \{c\}, \{d\}, \emptyset \}
\]
Conflict-free Set

Given an AF \((A, R)\).
A set \(S \subseteq A\) is conflict-free in \(F\), if, for each \(a, b \in S\), \((a, b) \notin R\).

Encoding for \(F = (A, R)\)

\[
\hat{F} = \{\text{arg}(a) \mid a \in A\} \cup \{\text{att}(a, b) \mid (a, b) \in R\}
\]

\[
\pi_{cf} = \begin{cases} 
\text{in}(X) & \leftarrow \text{not} \ \text{out}(X), \ \text{arg}(X) \\
\text{out}(X) & \leftarrow \text{not} \ \text{in}(X), \ \text{arg}(X) \\
& \leftarrow \text{in}(X), \ \text{in}(Y), \ \text{att}(X, Y) 
\end{cases}
\]

Result: For each AF \(F\), \(cf(F) \equiv AS(\pi_{cf}(\hat{F}))\)
Stable Extension

Given an AF $(A, R)$. A set $S \subseteq A$ is stable in $F$, if
- $S$ is conflict-free in $F$
- each $a \in A \setminus S$, there exists a $b \in S$, such that $(b, a) \in R$.

Encoding

$$
\pi_{\text{stable}} = \begin{cases} 
\text{in}(X) & \leftarrow \not \text{out}(X), \text{arg}(X) \\
\text{out}(X) & \leftarrow \not \text{in}(X), \text{arg}(X) \\
& \leftarrow \text{in}(X), \text{in}(Y), \text{att}(X, Y) \\
\text{defeated}(X) & \leftarrow \text{in}(Y), \text{att}(Y, X) \\
& \leftarrow \text{out}(X), \not \text{defeated}(X)
\end{cases}
$$

Result: For each AF $F$, $\text{stable}(F) \equiv \mathcal{AS}(\pi_{\text{stable}}(\widehat{F}))$
Semantics and types of AFs incorporated in ASPARTIX:

- admissible, complete, stable, preferred, grounded, ideal, stage, semi-stable and cf2;
- Preference-based AFs, Value-based AFs, Bipolar AFs, Dynamic AFs and AFs with Recursive Attacks.
AFs became very important in Artificial Intelligence. They provide a popular tool for modeling and evaluating conflicting knowledge.

Problems associated to AFs are in general intractable, therefore we translate them to ASP.

Web front-end of ASPARTIX is freely available.

http://rull.dbai.tuwien.ac.at:8080/ASPARTIX
**Relation between Semantics**

```
stable → pref → compl → adm

stable → ground
```

**Complexity**

<table>
<thead>
<tr>
<th></th>
<th>stable</th>
<th>adm</th>
<th>pref</th>
<th>comp</th>
<th>ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cred</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
<td>in P</td>
</tr>
<tr>
<td>Skept</td>
<td>coNP-c</td>
<td>(trivial)</td>
<td>Π²_p</td>
<td>in P</td>
<td>in P</td>
</tr>
</tbody>
</table>

[Dimopoulos & Torres 96; Dunne & Bench-Capon 02; Coste-Marquis et al. 05]
Complexity of Argumentation

<table>
<thead>
<tr>
<th></th>
<th>stable</th>
<th>adm</th>
<th>pref</th>
<th>comp</th>
<th>ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cred</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
<td>NP-c</td>
<td>in P</td>
</tr>
<tr>
<td>Skept</td>
<td>coNP-c</td>
<td>(trivial)</td>
<td>Π²_p-c</td>
<td>in P</td>
<td>in P</td>
</tr>
</tbody>
</table>

Recall: Data-Complexity of Datalog

<table>
<thead>
<tr>
<th></th>
<th>stratified programs</th>
<th>with negation</th>
<th>with neg. and disjunction</th>
</tr>
</thead>
</table>
| \( 
\models_c \)   | P-c                 | NP-c          | \( \Sigma^p_2 \)-c        |
| \( 
\models_s \)   | P-c                 | coNP-c        | \( \Pi^p_2 \)-c           |

[Dantsin, Eiter, Gottlob, Voronkov 01]
Performance Tests

Tested Systems:
- **Grounders**: DLV, lparse, GrinGo
- **Solvers**: DLV, smodels, cmodels, clasp, claspD, gnt

Testing:
- Randomly generated AFs from 90 to 200 arguments with edge density from 10% to 30%.
- In total 21303 tests were performed.
Admissible Extensions:

- "dlv"
- "clasp-1parse"
- "clasp-gringo"
- "claspd-1parse"
- "claspd-gringo"
- "smodels-1parse"
- "smodels-gringo"
- "cmodels-1parse"
- "cmodels-gringo"
- "gnt2-1parse"
- "gnt2-gringo"

The diagram shows the time (in seconds) on the y-axis and the number of nodes on the x-axis for different admissible extensions.

Sarah A. Gaggl, TU Vienna
General Argumentation System Based on ASP
Preferred Extensions:

- "dlv"
- "claspd-lparse"
- "claspd-gringo"
Stable Extensions:

- "dlv"
- "clasp-lparse"
- "clasp-gringo"
- "clasps-lparse"
- "clasps-gringo"
- "smodels-lparse"
- "smodels-gringo"
- "cmodels-lparse"
- "cmodels-gringo"
- "gnt2-lparse"
- "gnt2-gringo"