Implementing Abstract Argumentation Frameworks

Sarah Gaggl

Dresden, 22th November 2013
Outline

- Direct- vs. Reduction-based Approach
- Propositional Logic
- Answer-Set Programming
- ASP Encodings of AF Semantics
- Students’ Topics
Motivation

- **Argumentation Frameworks** 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.
- Distinction between direct implementation and reduction-based approach.
- We focus on reductions to propositional logic and Answer-Set Programming (ASP).
The rapid implementation approach (RIA)

We know:
- Any complete problem can be translated into any other complete problem of the same complexity class
- Moreover, there exists poly-time translations (reductions)
- Complexity results (incl. completeness) for many reasoning tasks

We used already:
- e.g., the PTIME reduction from a CNF $\varphi$ to an AF $F(\varphi)$ such that $\varphi$ is satisfiable iff $F(\varphi)$ has an admissible set containing $\varphi$
- Can we “reverse” the reduction, i.e., from AFs to formulas?
- YES! Reduce to formalisms for which “good” solvers are available
  But we have to find the PTIME reduction!
The rapid implementation approach (2)

- Reduce reasoning tasks for AF, e.g., to SAT problems of (Q)BFs
- Reductions are “cheap” (wrt runtime and implementation effort!)
- Good SAT and QSAT solvers are available; simply use them

Benefits:

- Reductions are much easier to implement than full-fledged algorithms especially for “hard” reasoning tasks
- Basic reductions can be combined and reused
- Different formalisms can be reduced to same target formalism
  - beneficial for comparative studies
The rapid implementation approach (3)

Target formalisms are:

- The SAT problem for propositional formulas
- The SAT problem for quantified Boolean formulas
- Answer-set programs

Tools are available to solve all these three formalisms

Many developers are happy to give away their tool

They work hard to improve the tool’s performance (for you!)
Required properties of reductions: Faithfulness

- Let \( \Pi \) be a decision problem
- \( F_\Pi(\cdot) \) a reduction to a target formalism
- \( F_\Pi(\cdot) \) has to satisfy the following three conditions:
  1. \( F_\Pi(\cdot) \) is faithful, i.e., \( F_\Pi(K) \) is true iff \( K \) is a yes-instance of \( \Pi \)
  2. For each instance \( K \), \( F_\Pi(K) \) is poly-time computable wrt size of \( K \)
  3. Determining the truth of \( F_\Pi(K) \) is computationally not harder than deciding \( \Pi \)

Faithfulness guarantees a correct “simulation” of \( K \)
Reductions to Propositional Logic

Given an AF $F = (A, R)$, for each $a \in A$ a propositional variable $v_a$ is constructed.

- $S \subseteq A$ is a $\sigma$ extension of $F$ iff $\{v_a | a \in S\} \models \varphi$,
- with $\varphi$ a propositional formula that evaluates $F$ under semantics $\sigma$.

### Admissible Sets

$$adm_{A, R} := \bigwedge_{a \in A} (v_a \rightarrow \bigwedge_{(b, a) \in R} \neg v_b) \land (v_a \rightarrow \bigwedge_{(b, a) \in R} (\bigvee_{(c, b) \in R} v_c))$$

Models of $adm_{A, R}$ correspond to admissible sets of $F$ [Besnard & Doutre 04].
Admissible Sets

\[ adm_{A,R} := \land_{a \in A} ((v_a \to \land_{(b,a) \in R} \neg v_b) \land (v_a \to \land_{(b,a) \in R} (\lor_{(c,b) \in R} v_c))) \]

Example

\[ adm_{A,R} = ((v_a \to \top) \land (v_b \to (\neg v_a \land \neg v_c)) \land (v_c \to (\neg v_b \land \neg v_d)) \land (v_d \to \top) \land (v_e \to (\neg v_d \land \neg v_e))) \land ((v_a \to \top) \land (v_b \to (\bot \land (v_b \lor v_d))) \land (v_c \to ((v_a \lor v_c) \land \bot)) \land (v_d \to \top) \land (v_e \to (\bot \land d)))) \]
A rule $r$ is an expression of the form

$$a_1 \lor \cdots \lor a_n \leftarrow b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m,$$

with $n \geq 0$, $m \geq k \geq 0$, $n + m > 0$, where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms, and “not” stands for default negation.

We call

- $H(r) = \{a_1, \ldots, a_n\}$ the head of $r$;
- $B(r) = \{b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m\}$ the body of $r$;
- $B^+(r) = \{b_1, \ldots, b_k\}$ the positive body of $r$;
- $B^-(r) = \{b_{k+1}, \ldots, b_m\}$ the negative body of $r$. 
ASP Semantics

- An interpretation $I$ satisfies a ground rule $r$ iff $H(r) \cap I \neq \emptyset$ whenever
  - $B^+(r) \subseteq I$,
  - $B^-(r) \cap I = \emptyset$.
- $I$ satisfies a ground program $\pi$, if each $r \in \pi$ is satisfied by $I$.
- A non-ground rule $r$ (resp., a program $\pi$) is satisfied by an interpretation $I$ iff $I$ satisfies all groundings of $r$ (resp., $Gr(\pi)$).

Gelfond-Lifschitz reduct

An interpretation $I$ is an answer set of $\pi$ iff it is a subset-minimal set satisfying

$$\pi^I = \{H(r) \leftarrow B^+(r) \mid I \cap B^-(r) = \emptyset, r \in Gr(\pi)\}.$$
### Complexity of Argumentation

<table>
<thead>
<tr>
<th></th>
<th>adm</th>
<th>pref</th>
<th>semi</th>
<th>stage</th>
<th>grd*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cred</td>
<td>NP-c</td>
<td>NP-c</td>
<td>Σ²_p-c</td>
<td>Σ²_p-c</td>
<td>NP-c</td>
</tr>
<tr>
<td>Skept</td>
<td>(trivial)</td>
<td>Π²_p-c</td>
<td>Π²_p-c</td>
<td>Π²_p-c</td>
<td>co-NP-c</td>
</tr>
</tbody>
</table>

[Baroni et al. 11; Dimopoulos & Torres 96; Dunne & Bench-Capon 02; Dvořák & Woltran 10]

### Recall: Data-Complexity of Datalog

<table>
<thead>
<tr>
<th></th>
<th>normal programs</th>
<th>disjunctive program</th>
<th>optimization programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊨_c</td>
<td>NP</td>
<td>Σ²_p</td>
<td>Σ²_p</td>
</tr>
<tr>
<td>⊨_s</td>
<td>co-NP</td>
<td>Π²_p</td>
<td>Π²_p</td>
</tr>
</tbody>
</table>

[Dantsin, Eiter, Gottlob, Voronkov 01]
ASP Encodings

**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 \(F = (A, R)\)**

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

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

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

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

- $S$ is conflict-free in $F$
- each $a \in S$ is **defended** by $S$ in $F$
  - $a \in A$ is defended by $S$ in $F$, if for each $b \in A$ with $(b, a) \in R$, there exists a $c \in S$, such that $(c, b) \in R$.

Encoding

$$\pi_{adm} = \pi_{cf} \cup \{ \text{defeated}(X) \leftarrow \text{in}(Y), \text{att}(Y, X), \text{in}(X), \text{att}(Y, X), \text{not} \text{defeated}(Y) \}$$

Result: For each AF $F$, $adm(F) \equiv AS(\pi_{adm}(\hat{F}))$
### Stable Extensions

Given an AF $F = (A, R)$. A set $S \subseteq A$ is a **stable extension** of $F$, if

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

### Encoding

$$\pi_{stable} = \pi_{cf} \cup \left\{ \begin{array}{c}
\text{defeated}(X) \leftarrow \text{in}(Y), \text{att}(Y, X) \\
\text{out}(X), \text{not defeated}(X) \leftarrow \text{not defeated}(X)
\end{array} \right\}$$

**Result:** For each AF $F$, $\text{stable}(F) \equiv \text{AS}(\pi_{stable}(\hat{F}))$
Grounded Extension

Given an AF $F = (A, R)$. The characteristic function $F_F : 2^A \rightarrow 2^A$ of $F$ is defined as

$$F_F(E) = \{ x \in A \mid x \text{ is defended by } E \}.$$ 

The least fixed point of $F_F$ is the grounded extension.

Order over domain

$$\pi_\prec = \begin{cases} 
\text{lt}(X, Y) & \leftarrow \text{arg}(X), \text{arg}(Y), X < Y \\
\text{nsucc}(X, Z) & \leftarrow \text{lt}(X, Y), \text{lt}(Y, Z) \\
\text{succ}(X, Y) & \leftarrow \text{lt}(X, Y), \text{not } \text{nsucc}(X, Y) \\
n\text{inf}(X) & \leftarrow \text{lt}(Y, X) \\
n\text{sup}(X) & \leftarrow \text{lt}(X, Y) \\
\text{inf}(X) & \leftarrow \text{not } n\text{inf}(X), \text{arg}(X) \\
\text{sup}(X) & \leftarrow \text{not } n\text{sup}(X), \text{arg}(X) 
\end{cases}$$
Grounded Extension

Given an AF $F = (A, R)$. The characteristic function $\mathcal{F}_F : 2^A \to 2^A$ of $F$ is defined as

$$\mathcal{F}_F(E) = \{ x \in A \mid x \text{ is defended by } E \}.$$

The least fixed point of $\mathcal{F}_F$ is the grounded extension.

Encodings Grounded Extension

$$\pi_{\text{ground}} = \begin{cases} 
\text{def}_\text{upto}(X, Y) & \leftarrow \ \text{inf}(Y), \ \text{arg}(X), \ \text{not att}(Y, X) \\
\text{def}_\text{upto}(X, Y) & \leftarrow \ \text{inf}(Y), \ \text{in}(Z), \ \text{att}(Z, Y), \ \text{att}(Y, X) \\
\text{def}_\text{upto}(X, Y) & \leftarrow \ \text{succ}(Z, Y), \ \text{def}_\text{upto}(X, Z), \ \text{not att}(Y, X) \\
\text{def}_\text{upto}(X, Y) & \leftarrow \ \text{succ}(Z, Y), \ \text{def}_\text{upto}(X, Z), \ \text{in}(V), \ \text{att}(V, Y), \ \text{att}(Y, X) \\
\text{defended}(X) & \leftarrow \ \text{sup}(Y), \ \text{def}_\text{upto}(X, Y) \\
\text{in}(X) & \leftarrow \ \text{defended}(X)
\end{cases}$$

Result: For each AF $F$, $\text{ground}(F) \equiv \mathcal{AS}(\pi_{\text{ground}}(\widehat{F}))$
**Preferred Extensions**

Given an AF \( F = (A, R) \). A set \( S \subseteq A \) is a preferred extension of \( F \), if

- \( S \) is admissible in \( F \)
- for each \( T \subseteq A \) admissible in \( F \), \( S \not\subset T \)

**Encoding**

- Preferred semantics needs **subset maximization task**.
- Can be encoded in standard ASP but requires **insight** and **expertise**.
Saturation Encodings

Preferred Extension

Given an AF \((A, R)\). A set \(S \subseteq A\) is preferred in \(F\), if \(S\) is admissible in \(F\) and for each \(T \subseteq A\) admissible in \(T\), \(S \not\subset T\).

Encoding

\[
\pi_{\text{saturate}} = \begin{cases} 
\text{inN}(X) \lor \text{outN}(X) & \leftarrow \text{out}(X); \\
\text{inN}(X) & \leftarrow \text{in}(X) \\
\text{fail} & \leftarrow \text{eq} \\
\text{fail} & \leftarrow \text{inN}(X), \text{inN}(Y), \text{att}(X, Y) \\
\text{fail} & \leftarrow \text{inN}(X), \text{outN}(Y), \text{att}(Y, X), \text{undefeated}(Y) \\
\text{inN}(X) & \leftarrow \text{fail}, \text{arg}(X) \\
\text{outN}(X) & \leftarrow \text{fail}, \text{arg}(X) \\
& \leftarrow \text{not} \text{ fail} 
\end{cases}
\]

\[
\pi_{\text{pref}} = \pi_{\text{adm}} \cup \pi_{\text{helpers}} \cup \pi_{\text{saturate}}
\]

Result: For each AF \(F\), \(\text{pref}(F) \equiv \text{AS}(\pi_{\text{pref}}(\widehat{F}))\)
Metasp [Gebser et al., 2011]

- Recently proposed `metasp` front-end for the `gringo/claspD` package.
- The problem encoding is first grounded with the `reify` option, which outputs ground program as facts.
- Next the meta encodings mirror answer-set generation.
- Meta encodings also implement subset minimization for the `#minimize`-statement.
Metasp Encoding

- Together with the module admissibility, the remaining encoding for subset maximization reduces to

\[ \pi_{adm} \cup \{ \#\text{minimize}[\text{out}(X)] \}. \]

Preferred Extensions

- This relocates the optimization encoding to the meta-encodings.
- Enables simple encodings and performs surprisingly well.
Additional info on encodings and extensions

**ASPARTIX (ASP Argumentation Reasoning Tool)**

- Encodings are used together with an ASP-solver, like clasp or dvl
- Implements all prominent argumentation semantics
- Even for extended frameworks like PAFs, VAFs, BAPs, ...
- Easy to use
- Web-interface available:
  [http://rull.dbai.tuwien.ac.at:8080/ASPARTIX/](http://rull.dbai.tuwien.ac.at:8080/ASPARTIX/)

**Info and encodings are available under:**

[http://www.dbai.tuwien.ac.at/research/project/argumentation/](http://www.dbai.tuwien.ac.at/research/project/argumentation/)
Related work

**Other encodings**

- by [Nieves et al., 2008] and follow-up papers; mostly a new program has to be constructed for each instance
- DIAMOND (DIAlectical MOdels eNcoDing) is a software system to compute different ADF models (see https://isysrv.informatik.uni-leipzig.de/diamond)
- ConArg is a tool, based on Constraint Programming [Bistarelli and Santini, 2012] (see https://sites.google.com/site/santinifrancesco/tools)

**Other systems**

- Collection: http://wyner.info/LanguageLogicLawSoftware/index.php/software/
- System Demos at COMMA 2012: http://www.kr.tuwien.ac.at/events/comma2012/accepted.php#demos
Students’ Topics

- Abstract Dialectical Frameworks (ADFs),
- Instantiations,
- Dynamics in argumentation,
- Loops in argumentation frameworks,
- Argumentation and Answer-Set Programming (ASP),
- Equivalences in AFs,
- SAT-Procedures for AFs,
- Computational Complexity of AFs,
- Social Choice and Argumentation,
- Argumentation and the Semantic Web,
- Argumentation and Game Theory
Presentations and Invited Talk

**Presentations**
- 20 min presentation plus 10 min discussion
- send slides 2 weeks before presentation date to sarah.gaggl@tu-dresden.de
- check your paper after assignment!
- date for presentations: 17th and 18th January 2014

**Invited Talk**
- Stefan Woltran will give a talk probably on 15th January 2014
- participants of seminar should join the talk
Philippe Besnard and Sylvie Doutre.
Checking the acceptability of a set of arguments.

S. Bistarelli, F. Santini, Conarg: a tool to solve (weighted) abstract argumentation frameworks with (soft) constraints, CoRR abs/1212.2857.

On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.

Making use of advances in answer-set programming for abstract argumentation systems.

Uwe Egly and Stefan Woltran.
Reasoning in argumentation frameworks using quantified boolean formulas.

Uwe Egly, Sarah Gaggl, and Stefan Woltran.
Answer-set programming encodings for argumentation frameworks.

Complex optimization in answer set programming.

Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés.
Preferred extensions as stable models.