TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

SEMINAR ABSTRACT ARGUMENTATION

Implementing Abstract Argumentation Frameworks

Sarah Gaggl

Dresden, 7th November 2014

DRESDEN concept Exzellenz aus Wissenschaft und Kultur

Outline

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

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).

Laziness and Implementations

Alternative 1: The Japanese way

- Implement a separate algorithm for each reasoning task
- Implementation is complicated because most reasoning tasks are inherently intricate (☞ the complexity results given before)
- Implementation, testing, etc. require much effort and time

Laziness and Implementations

Alternative 1: The Japanese way

- Implement a separate algorithm for each reasoning task
- Implementation is complicated because most reasoning tasks are inherently intricate (☞ the complexity results given before)
- Implementation, testing, etc. require much effort and time

Alternative : The southern way

- Life is short; try to keep your effort as small as possible
- Let others work for you and use their results and software
- Be smart; apply what you have learned

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 φ to an AF $F(\varphi)$ such that

  φ is satisfiable iff $F(\varphi)$ has an admissible set containing φ
- 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 Π 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 Π
  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 Π

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 σ extension of F iff $\{v_a \mid a \in S\} \models \varphi$,
- with φ a propositional formula that evaluates F under semantics σ.

Admissible Sets

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

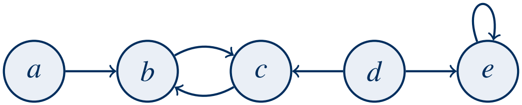
Models of $adm_{A,R}$ correspond to admissible sets of F [Besnard & Doutre 04].

Reductions to Propositional Logic ctd.

Admissible Sets

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

Example



$$\begin{aligned} adm_{A,R} =& ((v_a \rightarrow \top) \wedge (v_b \rightarrow (\neg v_a \wedge \neg v_c)) \wedge (v_c \rightarrow (\neg v_b \wedge \neg v_d)) \wedge (v_d \rightarrow \top) \wedge \\ & (v_e \rightarrow (\neg v_d \wedge \neg v_e))) \wedge ((v_a \rightarrow \top) \wedge (v_b \rightarrow (\bot \wedge (v_b \vee v_d))) \wedge \\ & (v_c \rightarrow ((v_a \vee v_c) \wedge \bot)) \wedge (v_d \rightarrow \top) \wedge (v_e \rightarrow (\bot \wedge v_d))) \end{aligned}$$

General Idea of Answer-Set Programming

Fundamental concept:

- Models = set of atoms
- Models, not proofs, represent solutions!
- Need techniques to compute models (not to compute proofs)

➥ Methodology to solve search problems

Solving search problems with ASP

- Given a problem Π and an instance K , reduce it to the problem of computing intended models of a logic program:
  1. Encode (Π, K) as a logic program P such that the solutions of Π for the instance K are represented by the intended models of P
  2. Compute one intended model M (an "answer set") of P
  3. Reconstruct a solution for K from M
- Variant: Compute all intended models to obtain all solutions

ASP Solvers

Efficient solvers available

- `gringo/clasp` (University of Potsdam)
- `dlv` (TU Wien, University of Calabria)
- `smodels, GnT` (Aalto University, Finland)
- `ASSAT` (Hong Kong University of Science and Technology)

Answer-Set Programming Syntax

- We assume a first-order vocabulary Σ comprised of nonempty finite sets of constants, variables, and predicate symbols, but no function symbols
- A term is either a variable or a constant
- An atom is an expression of form $p(t_1, \ldots, t_n)$, where
  - p is a predicate symbol of arity $n \geq 0$ from Σ, and
  - $t_1, \ldots, t_n$ are terms
- A literal is an atom p or a negated atom $\neg p$
  - ☞ $\neg$ is called strong negation, or classical negation
- A literal is ground if it contains no variable.

Answer-Set Programming Syntax ctd.

ASP Syntax

A rule r is an expression of the form

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

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;
- $B(r) = \{b_1, \ldots, b_k, not \ b_{k+1}, \ldots, not \ b_m\}$ 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.
- Intuitive meaning of r: if $b_1, \ldots, b_k$ are derivable, but $b_{k+1}, \ldots, b_m$ are not derivable, then one of $a_1, \ldots, a_n$ is asserted
- A program is a finite set of rules

Answer-Set Programming Syntax ctd.

A rule $a_1 \lor \cdots \lor a_n \leftarrow b_1, \ldots, b_k,\ \mathit{not}\ b_{k+1}, \ldots,\ \mathit{not}\ b_m$ is

- a fact if $m = 0$ and $n \geq 1$
- a constraint if $n = 0$ (i.e., the head is empty)
- basic if $m = k$ and $n \geq 1$
- non-disjunctive if $n = 1$
- normal if it is non-disjunctive and contains no strong negation $\neg$
- Horn if it is normal and basic
- ground if all its literals are ground

A program is basic, normal, etc., if all of its rules are

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 π, if each $r \in \pi$ is satisfied by I.
- A non-ground rule r (resp., a program π) 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 π iff it is a subset-minimal set satisfying

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

Programming methodology

Simplest technique: Guess and check

- Guess: Generate candidates for answer sets in the first step
- Check: Filter the answer sets and delete undesirable ones

Example (Graph coloring)

$$\begin{array}{ll}
\mathrm{node}(a).\mathrm{node}(b).\mathrm{node}(c).\mathrm{edge}(a,b).\mathrm{edge}(b,c). & \}\text{facts} \\
\mathrm{col}(red,X) \vee \mathrm{col}(green,X) \vee \mathrm{col}(blue,X) \leftarrow \mathrm{node}(X). & \}\text{guess} \\
\leftarrow \mathrm{edge}(X,Y), \mathrm{col}(C,X), \mathrm{col}(C,Y). & \}\text{check}
\end{array}$$

G: Generate all possible coloring candidates

C: Delete all candidates where adjacent nodes have same color

Corresponding Complexity Results

Complexity of Argumentation

	adm	*pref*	*semi*	*stage*	*grd*∗
Cred	NP-c	NP-c	Σ_2^p-c	Σ_2^p-c	NP-c
Skept	(trivial)	Π_2^p-c	Π_2^p-c	Π_2^p-c	co-NP-c

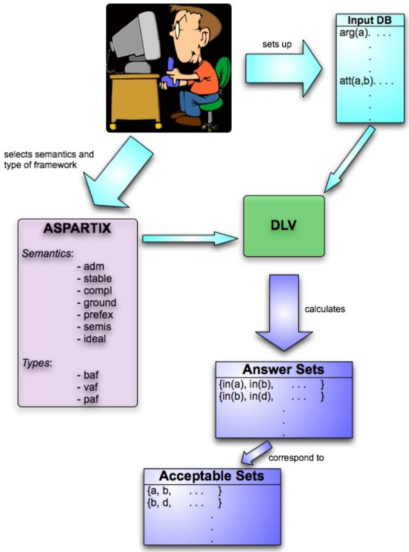
[Baroni et al. 11; Dimopoulos & Torres 96; Dunne & Bench-Capon 02; Dvořák & Woltran 10]

Recall: Data-Complexity of Datalog

	normal programs	disjunctive program	optimization programs
$\models_c$	NP	Σ_2^p	Σ_2^p
$\models_s$	co-NP	Π_2^p	Π_2^p

[Dantsin,Eiter,Gottlob,Voronkov 01]

ASPARTIX - System Description



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 $F = (A, R)$

$$\widehat{F} = \{\mathrm{arg}(a) \mid a \in A\} \cup \{\mathrm{att}(a, b) \mid (a, b) \in R\}$$

$$\pi_{cf} = \left\{ \begin{array}{lcl} \mathrm{in}(X) & \leftarrow & \mathit{not}\ \mathrm{out}(X), \mathrm{arg}(X) \\ \mathrm{out}(X) & \leftarrow & \mathit{not}\ \mathrm{in}(X), \mathrm{arg}(X) \\ & \leftarrow & \mathrm{in}(X), \mathrm{in}(Y), \mathrm{att}(X, Y) \end{array} \right\}$$

Result: For each AF F, $cf(F) \equiv \mathcal{AS}(\pi_{cf}(\widehat{F}))$

ASP Encodings cont.

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

$$\pi_{adm} = \pi_{cf} \cup \left\{ \begin{array}{lcl} \text{defeated}(X) & \leftarrow & \text{in}(Y), \text{att}(Y, X) \\ & \leftarrow & \text{in}(X), \text{att}(Y, X), \textit{not}\ \text{defeated}(Y) \end{array} \right\}$$

Result: For each AF F, $adm(F) \equiv \mathcal{AS}(\pi_{adm}(\widehat{F}))$

ASP Encodings ctd.

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

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

Result: For each AF F, $stable(F) \equiv \mathcal{AS}(\pi_{stable}(\widehat{F}))$

ASP Encodings ctd.

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.

Order over domain

$$\pi_< = \left\{ \begin{array}{lcl} \mathrm{lt}(X,Y) & \leftarrow & \mathrm{arg}(X), \mathrm{arg}(Y), X < Y \\ \mathrm{nsucc}(X,Z) & \leftarrow & \mathrm{lt}(X,Y), \mathrm{lt}(Y,Z) \\ \mathrm{succ}(X,Y) & \leftarrow & \mathrm{lt}(X,Y), \mathit{not}\ \mathrm{nsucc}(X,Y) \\ \mathrm{ninf}(X) & \leftarrow & \mathrm{lt}(Y,X) \\ \mathrm{nsup}(X) & \leftarrow & \mathrm{lt}(X,Y) \\ \mathrm{inf}(X) & \leftarrow & \mathit{not}\ \mathrm{ninf}(X), \mathrm{arg}(X) \\ \mathrm{sup}(X) & \leftarrow & \mathit{not}\ \mathrm{nsup}(X), \mathrm{arg}(X) \end{array} \right\}$$

ASP Encodings ctd.

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

$$\pi_{ground} = \left\{ \begin{array}{lcl} \text{def\_upto}(X, Y) & \leftarrow & \text{inf}(Y), \text{arg}(X), \textit{not } \text{att}(Y, X) \\ \text{def\_upto}(X, Y) & \leftarrow & \text{inf}(Y), \text{in}(Z), \text{att}(Z, Y), \text{att}(Y, X) \\ \text{def\_upto}(X, Y) & \leftarrow & \text{succ}(Z, Y), \text{def\_upto}(X, Z), \textit{not } \text{att}(Y, X) \\ \text{def\_upto}(X, Y) & \leftarrow & \text{succ}(Z, Y), \text{def\_upto}(X, Z), \text{in}(V), \\ & & \text{att}(V, Y), \text{att}(Y, X) \\ \text{defended}(X) & \leftarrow & \text{sup}(Y), \text{def\_upto}(X, Y) \\ \text{in}(X) & \leftarrow & \text{defended}(X) \end{array} \right\}$$

Result: For each AF F, $\textit{ground}(F) \equiv \mathcal{AS}(\pi_{ground}(\widehat{F}))$

ASP Encodings

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

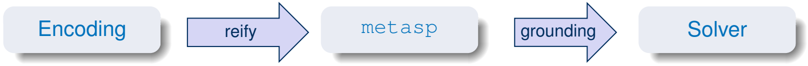
$$\pi_{saturate} = \left\{ \begin{array}{lcl} \mathrm{inN}(X) \vee \mathrm{outN}(X) & \leftarrow & \mathrm{out}(X); \\ \mathrm{inN}(X) & \leftarrow & \mathrm{in}(X) \\ \mathrm{fail} & \leftarrow & \mathrm{eq} \\ \mathrm{fail} & \leftarrow & \mathrm{inN}(X), \mathrm{inN}(Y), \mathrm{att}(X,Y) \\ \mathrm{fail} & \leftarrow & \mathrm{inN}(X), \mathrm{outN}(Y), \mathrm{att}(Y,X), \\ & & \mathrm{undefeated}(Y) \\ \mathrm{inN}(X) & \leftarrow & \mathrm{fail}, \mathrm{arg}(X) \\ \mathrm{outN}(X) & \leftarrow & \mathrm{fail}, \mathrm{arg}(X) \\ & \leftarrow & \mathit{not}\ \mathrm{fail} \end{array} \right\}$$

$$\pi_{pref} = \pi_{adm} \cup \pi_{helpers} \cup \pi_{saturate}$$

Result: For each AF F, $\mathit{pref}(F) \equiv \mathcal{AS}(\pi_{pref}(\widehat{F}))$

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

Preferred Extensions

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

- This relocates the optimization encoding to the meta-encodings.
- Enables simple encodings and performes 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/`

Info and encodings are available under:

`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
- Related approaches: reductions to SAT/QSAT [Besnard and Doutre, 2004, Egly and Woltran, 2006]
- 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 `http://www.dmi.unipg.it/conarg/`)

Other systems

- Collection: `http://wyner.info/LanguageLogicLawSoftware/index.php/software/`
- System Demos at COMMA 2014: `http://comma2014.arg.dundee.ac.uk/demoprogram`

Students' Topics so far

- Instantiations,
- Loops in argumentation frameworks,
- Evaluation criteria for argumentation semantics,
- Realizability,
- Translations,
- Equivalences in AFs,
- Argumentation and Answer-Set Programming (ASP),
- SAT-Procedures for AFs,
- Argumentation Systems,
- Applications

Invited Talk

Guest Research Talk

- Claudia Schulz from Imperial College London will give a talk on 28th November 2014 at 15:00 in INF 0005
- Title: Explaining Answer Sets using Argumentation Theory
- See `https://ddll.inf.tu-dresden.de/web/Explaining_Answer_Sets_using_Argumentation_Theory`
- Participants of seminar should join the talk!

Philippe Besnard and Sylvie Doutre.
Checking the acceptability of a set of arguments.
In Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR'02), pages 59–64, 2004.

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

Dung, P. M. (1995).
On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.
Artif. Intell., 77(2):321–358.

Dvořák, W., Gaggl, S. A., Wallner, J., and Woltran, S. (2011).
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.
In Proceedings of the 1st Conference on Computational Models of Argument (COMMA'06), pages 133–144. IOS Press, 2006.

Uwe Egly, Sarah Gaggl, and Stefan Woltran.
Answer-set programming encodings for argumentation frameworks.
In Argument and Computation, 1(2):147–177, 2010.

Gebser, M., Kaminski, R., and Schaub, T. (2011).
Complex optimization in answer set programming.
TPLP, 11(4-5):821–839.

Juan Carlos Nieves, Mauricio Osorio, and Ulises Cortés.
Preferred extensions as stable models.
Theory and Practice of Logic Programming, 8(4):527–543, July 2008.