FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Database and Artificial Intelligence Group

Utilizing ASP for Generating and Visualizing Argumentation Frameworks

Günther Charwat Johannes Peter Wallner Stefan Woltran

Database and Artificial Intelligence Group
Institute of Information Systems
Vienna University of Technology

ASPOCP'12 - September 4, 2012

Outline

Argumentation - Argumentation Frameworks

Steps

Starting point: knowledge-base

1) Form arguments
2) Identify conflicts
3) Abstract from internal structure
4) Resolve conflicts
5) Obtain conclusions

Argumentation - Argumentation Frameworks

Steps

Starting point: knowledge-base

1) Form arguments
2) Identify conflicts
3) Abstract from internal structure
4) Resolve conflicts
5) Obtain conclusions

- Input: A knowledge-base $\mathcal{K}$ and set $\mathcal{C}$ of claims

Example

$$\mathcal{K} = \{a, a \rightarrow b, \neg b\}$$
$$\mathcal{C} = \mathcal{K} \cup \{\neg a, b, a \wedge \neg b\}$$

Argumentation - Argumentation Frameworks

Steps

Starting point: knowledge-base

1) **Form arguments**
2) Identify conflicts
3) Abstract from internal structure
4) Resolve conflicts
5) Obtain conclusions

- Form arguments $A = (S, C)$ consisting of support $S \subseteq \mathcal{K}$ and claim $C \in \mathcal{C}$

Example

$$\mathcal{K} = \{a, a \rightarrow b, \neg b\}$$
$$\mathcal{C} = \mathcal{K} \cup \{\neg a, b, a \wedge \neg b\}$$

$(\{a\}, a)$

$(\{\neg b, a \rightarrow b\}, \neg a)$

$(\{a, a \rightarrow b\}, b)$ $(\{a, \neg b\}, a \wedge \neg b)$

$(\{\neg b\}, \neg b)$ $(\{a \rightarrow b\}, a \rightarrow b)$

Argumentation - Argumentation Frameworks

Steps

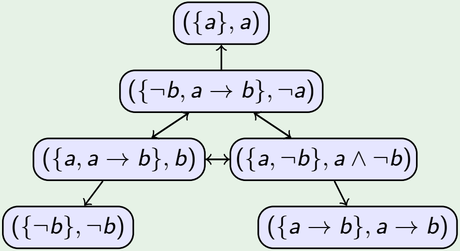
Starting point: knowledge-base

1) Form arguments
2) **Identify conflicts**
3) Abstract from internal structure
4) Resolve conflicts
5) Obtain conclusions

- Identify conflicts between arguments $A = (S, C)$ and $A' = (S', C')$

Example

$$\mathcal{K} = \{a, a \rightarrow b, \neg b\}$$
$$\mathcal{C} = \mathcal{K} \cup \{\neg a, b, a \wedge \neg b\}$$



Argumentation - Argumentation Frameworks

Steps

Starting point: knowledge-base

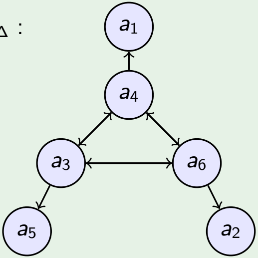
1) Form arguments
2) Identify conflicts
3) **Abstract from internal structure**
4) Resolve conflicts
5) Obtain conclusions

- Obtain an Abstract Argumentation Framework F_Δ (Dung [1995])

Example

$$\mathcal{K} = \{a, a \rightarrow b, \neg b\}$$
$$\mathcal{C} = \mathcal{K} \cup \{\neg a, b, a \wedge \neg b\}$$

F_Δ :



Argumentation - Argumentation Frameworks

Steps

Starting point: knowledge-base

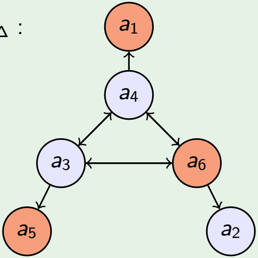
1) Form arguments
2) Identify conflicts
3) Abstract from internal structure
4) **Resolve conflicts**
5) Obtain conclusions

- Based on a semantics, select arguments that 'can stand together'

Example

$$\mathcal{K} = \{a, a \rightarrow b, \neg b\}$$
$$\mathcal{C} = \mathcal{K} \cup \{\neg a, b, a \wedge \neg b\}$$

F_Δ :



$$stb(F_\Delta) = \{\{a_1, a_5, a_6\}, \{a_1, a_2, a_3\}, \{a_2, a_4, a_5\}\}$$

Argumentation - Argumentation Frameworks

Steps

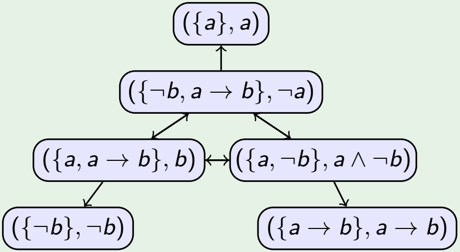
Starting point: knowledge-base

1) Form arguments
2) Identify conflicts
3) Abstract from internal structure
4) Resolve conflicts
5) **Obtain conclusions**

- Derive deductive closure of contents from accepted arguments

Example

$$\mathcal{K} = \{a, a \rightarrow b, \neg b\}$$
$$\mathcal{C} = \mathcal{K} \cup \{\neg a, b, a \wedge \neg b\}$$



$$Cn_{stb}(F_\Delta) = Cn((a \wedge \neg b) \vee (a \wedge b \wedge a \rightarrow b) \vee (a \rightarrow b \wedge \neg a \wedge \neg b))$$

Argumentation - Our Motivation

Steps

Starting point: knowledge-base

1) **Form arguments**
2) **Identify conflicts**
3) Abstract from internal structure
4) Resolve conflicts
5) Obtain conclusions

Current Status

- Not many systems for instantiation of AFs: e.g. CARNEADES, TOAST (for ASPIC+)
- Many different systems for evaluating AFs: e.g. ASPARTIX, Dungine, CASAPI, ArguMed, ConArg, reasoner by Visser (2008)
- Few tools for combining Computation and Visualization: e.g. CARNEADES, ASPARTIX (Web)

Argumentation - Our Motivation

Steps

Starting point: knowledge-base

1) Form arguments
2) Identify conflicts
3) Abstract from internal structure
4) **Resolve conflicts**
5) Obtain conclusions

Current Status

- Not many systems for instantiation of AFs: e.g. CARNEADES, TOAST (for ASPIC+)
- Many different systems for evaluating AFs: e.g. ASPARTIX, Dungine, CASAPI, ArguMed, ConArg, reasoner by Visser (2008)
- Few tools for combining Computation and Visualization: e.g. CARNEADES, ASPARTIX (Web)

Argumentation - Our Motivation

Steps

Starting point: knowledge-base

1) Form arguments
2) Identify conflicts
3) Abstract from internal structure
4) Resolve conflicts
5) Obtain conclusions

Current Status

- Not many systems for instantiation of AFs: e.g. CARNEADES, TOAST (for ASPIC+)
- Many different systems for evaluating AFs: e.g. ASPARTIX, Dungine, CASAPI, ArguMed, ConArg, reasoner by Visser (2008)
- Few tools for combining Computation and Visualization: e.g. CARNEADES, ASPARTIX (Web)

Argumentation - Our Motivation

Steps

Starting point: knowledge-base

1) **Form arguments**
2) **Identify conflicts**
3) **Abstract from internal structure**
4) Resolve conflicts
5) Obtain conclusions

Current Status

- Not many systems for instantiation of AFs: e.g. CARNEADES, TOAST (for ASPIC+)
- Many different systems for evaluating AFs: e.g. ASPARTIX, Dungine, CASAPI, ArguMed, ConArg, reasoner by Visser (2008)
- Few tools for combining Computation and Visualization: e.g. CARNEADES, ASPARTIX (Web)

Our Goals

→ Instantiation of AFs via ASP encodings

→ Tool for handling instantiation process and visualization of obtained Argumentation Frameworks

ASP Encodings

Overall Approach (Besnard and Hunter [2001]):

1. Form arguments $A = (S, C)$ such that for each argument:
   1. Input: $S \subseteq \mathcal{K}$ and $C \in \mathcal{C}$
   2. Consistency: S consistent
   3. Entailment: $S \models C$
   4. Subset Minimality: $\nexists S' \subset S$ s.t. $S' \models C$
2. Identify conflicts between arguments $A = (S, C)$ and $A' = (S', C')$:
   - Variety of different attack types, expressed by satisfiability of formulae, e.g.:
   - Defeat: $C \models \neg S'$
   - Direct Defeat: $C \models \neg\phi'_i$ for a $\phi'_i \in S'$
   - Rebuttal: $C \equiv \neg C'$
   - ...

$\rightarrow$ Model checking via ASP basis for construction of AFs

$\rightarrow$ Two-step approach: One encoding for arguments, another for attacks

Model Checking

- Basis for construction of arguments and attack relations
- Any propositional logic formula allowed (e.g. $\mathrm{imp}(a, \mathrm{neg}(b))$)

Model Checking Encoding ($\pi_{\mathrm{modelcheck}}$):

1) Input: Formula, specified by $\mathrm{formula}(F)$
2) Splitting of formula in subformulae and contained atoms
3) Guess interpretations, e.g. $\mathrm{true}(k, A) \vee \mathrm{false}(k, A) \leftarrow \mathrm{atom}(A).$
4) Obtain either $\mathrm{ismodel}(k, F)$ or $\mathrm{nomodel}(k, F)$ for formula F

Example

Input: $\mathrm{formula}(\mathrm{imp}(a, \mathrm{neg}(b)))$

Output (amongst others): $\{\mathrm{false}(k, a).\ \mathrm{true}(k, b).\ \mathrm{ismodel}(k, \mathrm{imp}(a, \mathrm{neg}(b))).\}$

Forming Arguments

(1) Input: $S \subseteq \mathcal{K}$ and $C \in \mathcal{C}$

- Knowledge-base $\mathcal{K}$, represented by the predicate $\mathrm{kb}(\cdot)$
- A set $\mathcal{C}$ of claims, represented by the predicate $\mathrm{cl}(\cdot)$

Guess arguments:

$$\begin{aligned}\pi_{\mathsf{arg}} \quad = \{ \quad & 1\{\ \mathrm{claim}(C) : \mathrm{cl}(C)\ \}1; \\ & 1\{\ \mathrm{fs}(FS) : \mathrm{kb}(FS)\ \}; \\ & \mathrm{formula}(C) \leftarrow \mathrm{claim}(C); \\ & \mathrm{formula}(FS) \leftarrow \mathrm{fs}(FS).\ \}\end{aligned}$$

(2) Consistency: S consistent

$$\begin{aligned}\pi_{\mathrm{consistent}} \quad = \{ \quad & 1\{\ \mathrm{true}(\mathrm{consistent}, A), \mathrm{false}(\mathrm{consistent}, A)\ \}1 \leftarrow \mathrm{atom}(A). \\ & \leftarrow \mathrm{nomodel}(\mathrm{consistent}, FS), \mathrm{fs}(FS).\ \}\end{aligned}$$

Forming Arguments

(3) Entailment: $S \models C$

- Expressible by unsatisfiability of $\neg(S \rightarrow C) \equiv \neg(\neg S \vee C) \equiv S \wedge \neg C$
- We apply the saturation technique (Eiter and Gottlob [1995])

$$
\begin{aligned}
\pi_{\text{entailment}} \quad = \{ \quad & \text{true}(\mathit{entail}, A) \vee \text{false}(\mathit{entail}, A) \leftarrow \text{atom}(A); \\
& \text{entails\_claim} \leftarrow \text{nomodel}(\mathit{entail}, \mathit{neg}(C)), \text{claim}(C); \\
& \text{entails\_claim} \leftarrow \text{nomodel}(\mathit{entail}, \mathit{FS}), \text{fs}(\mathit{FS}); \\
& \text{true}(\mathit{entail}, A) \leftarrow \text{entails\_claim}, \text{atom}(A); \\
& \text{false}(\mathit{entail}, A) \leftarrow \text{entails\_claim}, \text{atom}(A) : \\
& \leftarrow \mathit{not}\ \text{entails\_claim}. \ \}
\end{aligned}
$$

- If S or C not satisfied for interpretation, we obtain entails_claim and saturate
- Otherwise, constraint $\leftarrow \mathit{not}\ \text{entails\_claim}$ removes answer set
- Due to stable model semantics, answer set is only returned in case $S \wedge \neg C$ is unsatisfiable

Forming Arguments

(4) Subset Minimality: $\nexists S' \subset S$ s.t. $S' \models C$

- We apply concept of loop (see e.g. Eiter et al. [2009])
- All $S' \subset S$ considered where exactly one formula $\alpha \in S$ but $\alpha \notin S'$
- Sufficient due to monotonicity of classical logic

Minimality Encoding (π_{minimize}):

- For each S', check if $S' \models C$ valid
- Only keep answer sets, where guessed interpretation is no model for $S' \models C$
- If $S' \models C$ valid, S is not subset minimal

$\rightarrow$ All answer sets containing S as support are removed

Forming Arguments

Arguments obtained by:

$$\pi_{\text{arguments}} = \pi_{\text{modelcheck}} \cup \pi_{\mathsf{arg}} \cup \pi_{\text{consistent}} \cup \pi_{\text{entailment}} \cup \pi_{\text{minimize}}$$

Example

Input:

$$\{\text{kb}(a).\ \text{kb}(\text{imp}(a,b)).\ \text{kb}(\text{neg}(b)).$$
$$\text{cl}(a).\ \text{cl}(\text{imp}(a,b)).\ \text{cl}(\text{neg}(b)).\ \text{cl}(\text{neg}(a)).\ \text{cl}(b).\ \text{cl}(\text{and}(a,\text{neg}(b))).\}$$

Output:

$$a_1 : \quad \{\text{fs}(a).\ \text{claim}(a).\}$$
$$a_2 : \quad \{\text{fs}(\text{imp}(a,b)).\ \text{claim}(\text{imp}(a,b)).\}$$
$$a_3 : \quad \{\text{fs}(a).\ \text{fs}(\text{imp}(a,b)).\ \text{claim}(b).\}$$
$$a_4 : \quad \{\text{fs}(\text{neg}(b)).\ \text{fs}(\text{imp}(a,b)).\ \text{claim}(\text{neg}(a)).\}$$
$$a_5 : \quad \{\text{fs}(\text{neg}(b)).\ \text{claim}(\text{neg}(b)).\}$$
$$a_6 : \quad \{\text{fs}(a).\ \text{fs}(\text{neg}(b)).\ \text{claim}(and(a,\text{neg}(b))).\}$$

Identifying Conflicts between Arguments

(1) Input: A set of arguments

- Consists of 'flattened' arguments obtained by $\pi_{\text{arguments}}$
- Each argument specified by a set of predicates $\text{as}(A, \mathit{fs}, \cdot)$ and the predicate $\text{as}(A, \mathit{claim}, \cdot)$

Example

$$\{\text{as}(1, \text{fs}, a).\ \text{as}(1, \text{claim}, a).\ \text{as}(2, \text{fs}, \text{imp}(a, b)).\ \text{as}(2, \text{claim}, \text{imp}(a, b)).\ \text{as}(3, \text{fs}, a).\ \text{as}(3, \text{fs}, \text{imp}(a, b)).\ \text{as}(3, \text{claim}, b).\}$$

(2) Guess exactly two arguments (π_{att})

(3) Build single support formula (π_{support})

- Define an ordering over support formulae $\text{as}(A, \mathit{fs}, \cdot)$
- 'Iterate' over ordering and combine by conjunction

Identifying Conflicts between Arguments

(4) Define Attack Types (e.g. π_{defeat})

- Construct formula based on attack type
  e.g. Defeat: $C \models \neg S'$ specified as $\text{imp}(C, \text{neg}(S'))$
- Apply model checking to formula
- Derive predicate entails in case formula is satisfied

(5) Saturate ($\pi_{\text{att\_sat}}$)

- Apply coNP check (similar to entailment for arguments)
- If entails not derived, answer set is removed
- Otherwise, we saturate
- If formula of some attack type valid, we saturate all attack types

Attacks (defeats) obtained by:

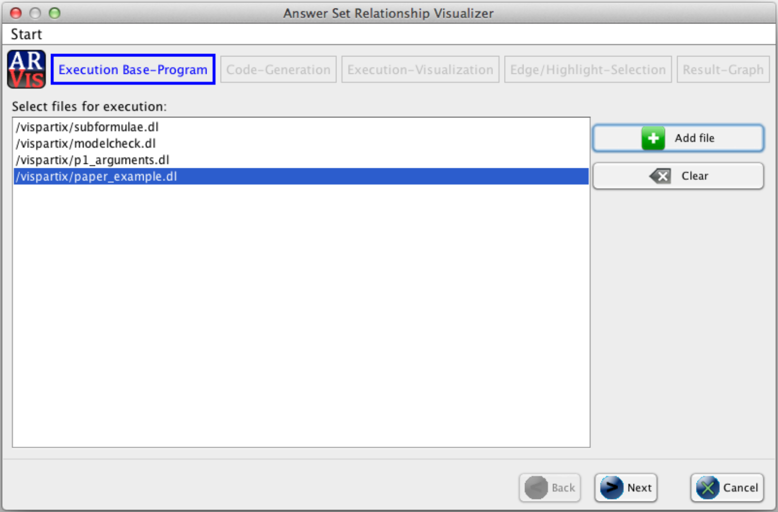
$$\pi_{\text{attacks}} = \pi_{\text{modelcheck}} \cup \pi_{\text{att}} \cup \pi_{\text{support}} \cup \pi_{\text{defeat}} \cup \pi_{\text{att\_sat}}$$

Visualization of Argumentation Frameworks

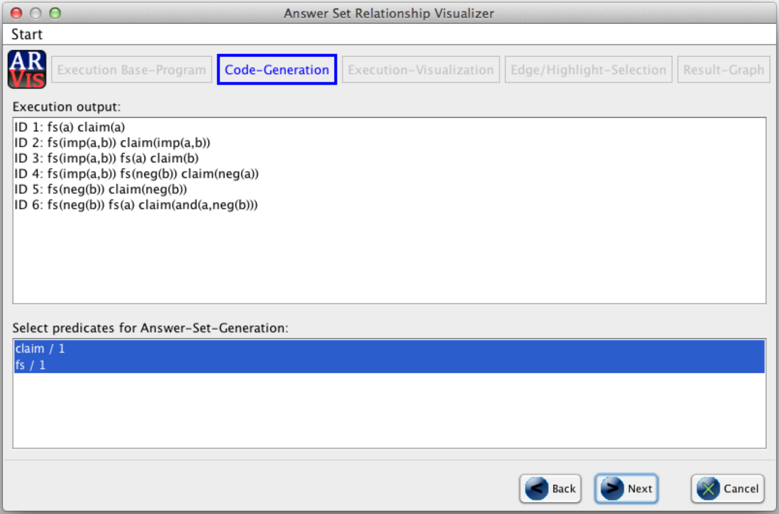
We utilise the purpose built tool ARVis (Answer set Relationship Visualizer):

1. Obtain arguments: Provide $\pi_{\text{arguments}}$ and a problem instance, gringo and claspD compute arguments
2. Flatten arguments: Generate argument facts
3. Obtain attacks: Provide π_{attacks} and any attack type programs, attacks are computed
4. Argumentation Framework: ARVis provides graph visualization consisting of arguments (vertices) and attacks (edges)
5. Export: Graph may be exported for further processing

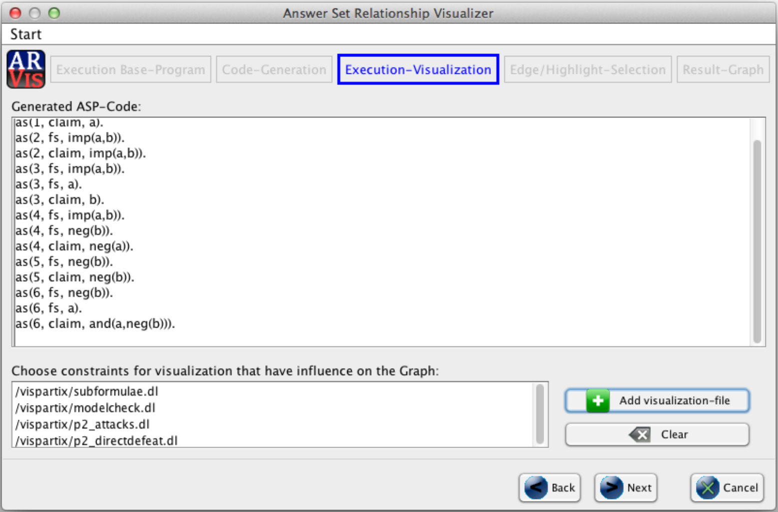
Visualization of Argumentation Frameworks



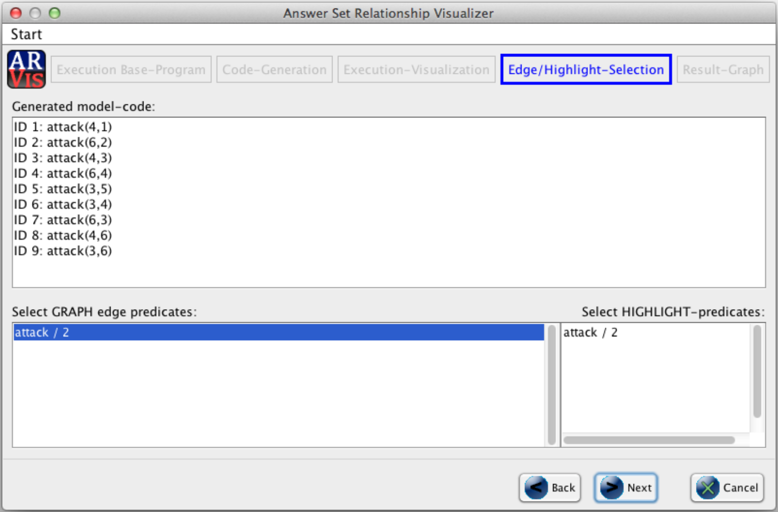
Visualization of Argumentation Frameworks



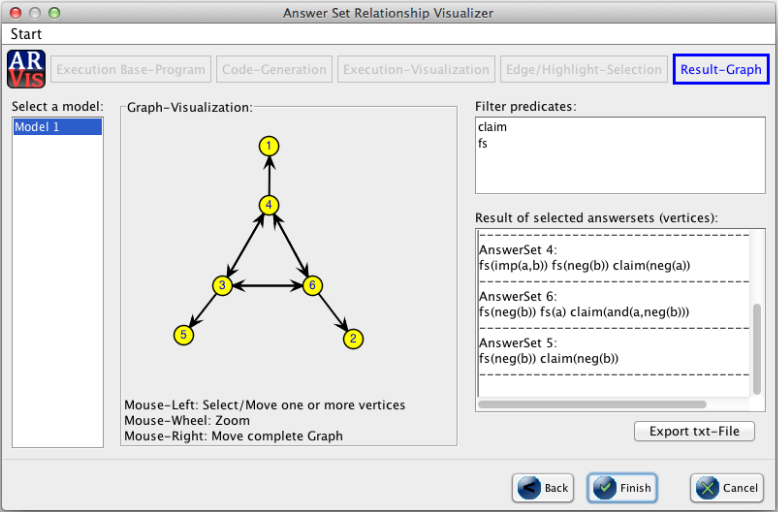
Visualization of Argumentation Frameworks



Visualization of Argumentation Frameworks



Visualization of Argumentation Frameworks



Summary and Future Work

Summary:

- ASP encodings for instantiation of Argumentation Frameworks (two steps)
  - → Declaratively descibed
  - → Easily adaptable to other notions
- New tool for visualising relations between answer sets (ARVis)
  - → Not restricted to Argumentation domain
  - → Versatile, e.g. export of graph for post-processing
- Encodings and tool publicly available:

  `http://dbai.tuwien.ac.at/proj/argumentation/vispartix`

Future Work:

- Implement further instantiation approaches
- Performance evaluation
- Analyze further application areas for answer set relationship visualization

References

Besnard, P. and Hunter, A. (2001).
A logic-based theory of deductive arguments.
Artif. Intell., 128(1-2):203–235.

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):321–357.

Eiter, T. and Gottlob, G. (1995).
On the Computational Cost of Disjunctive Logic Programming: Propositional Case.
Ann. Math. Artif. Intell., 15(3-4):289–323.

Eiter, T., Ianni, G., and Krennwallner, T. (2009).
Answer Set Programming: A Primer.
In *Reasoning Web*, volume 5689 of *LNCS*, pages 40–110.