Motivation: AFs
State of the art in abstract argumentation

Abstract Argumentation Frameworks (AFs)
- syntactically: directed graphs
- conceptually: nodes are arguments, edges denote attacks between arguments
- semantically: extensions are sets of “acceptable” arguments
- immensely popular in the argumentation community
- drawback: can only express attack

Motivation: ADFs
Recent improvements

Abstract Dialectical Frameworks (ADFs)
- generalise AFs, arguments are now called statements
- can also (although less directly) be visualised as graphs
- edges express that there is some relationship between the two statements
- relationship need not be “attack”, precise nature specified by acceptance condition for each statement
- acceptance condition specifies status of node given status of direct predecessors
Motivation: Argumentation in practice

Recent Projects and Ideas

- lawsuit analysis with argumentation
- argumentation for discussion analysis on social media (e.g. facebook)
- argumentation for blog-discussions

Abstract Argumentation

is for determining acceptance of abstract arguments

- argumentation framework $F = (A, R)$
- $A$ : set of arguments
- $R \subseteq A \times A$ : attack relation
- argument $a \in A$ is **defended** by a set $S \subseteq A$ if all attackers of $a$ are attacked by $S$:
  \[(\forall b \in A) (b Ra) \implies (\exists c \in S) c R b)\]

AF Semantics

are defined via extensions

for $AF = (A, R)$, a set $S \subseteq A$ of arguments is

- conflict-free iff for all $a, b \in S$, $(a, b) \notin R$
- a conflict-free set $S$ is
  - **admissible** iff it defends all arguments it contains
  - **preferred** iff it is $\subseteq$-maximal admissible
  - **complete** iff it contains exactly the arguments it defends
  - **grounded** iff it is $\subseteq$-minimal complete
  - **stable** iff it attacks all arguments not in $S$

AF Semantics

an example framework

- $AF = (A, R)$ with $A = \{a, b, c, d\}$ and $R = \{(a, b), (c, d), (d, c)\}$:
  \[
  \begin{array}{ccc}
  a & \rightarrow & b \\
  \rightarrow & c & \rightarrow d \\
  \end{array}
  \]
  - grounded extension: $G = \{a\}$
  - stable extensions: $E_1 = \{a, c\}$ and $E_2 = \{a, d\}$
  - preferred extensions: $E_1, E_2$
  - complete extensions: $G, E_1, E_2$
Abstract Dialectical Frameworks

Syntax

Definition: Abstract Dialectical Framework

An abstract dialectical framework (ADF) is a triple \( D = (S, L, C) \),

- \( S \) ... set of statements (correspond to AF arguments)
- \( L \subseteq S \times S \) ... links \((par(s) = L^{-1}(s))\)
- \( C = \{C_s\}_{s \in S} \) ... acceptance conditions

- links denote some kind of dependency relation
- acceptance condition: Boolean function \( C_s : 2^{par(s)} \to \{t, f\} \)
- here: \( C_s \) often specified by propositional formula \( \varphi_s \)

Semantics

Truth values, interpretations

- truth values: true \( t \), false \( f \), unknown \( u \)
- interpretation: \( \nu : S \to \{t, f, u\} \)
- interpretations can be represented as consistent sets of literals

Information ordering

- \( u <_i t \) and \( u <_i f \) (as usual \( x <_i y \) iff \( x < y \) or \( x = y \))
- consensus \( \sqcap \) is greatest lower bound w.r.t. \( \leq_i \):
  \( t \sqcap t = t \) and \( f \sqcap f = f \), otherwise \( x \sqcap y = u \)
- information ordering generalised to interpretations:
  \( \nu_1 <_i \nu_2 \) iff \( \nu_1(s) <_i \nu_2(s) \) for all \( s \in S \)

Characteristic Operator

- for a valuation \( \nu \), we define \( [\nu]_2 = \{w \leq_i w \mid w \text{ two-valued}\} \)
- for ADF \( D \), we define an operator \( \Gamma_D \) on interpretations
- for interpretation \( \nu : S \to \{t, f, u\} \), the operator yields a new interpretation (the consensus over \([\nu]_2\))

\[
\Gamma_D(\nu) : S \to \{t, f, u\} \quad s \mapsto \bigcap \{w(\varphi_s) \mid w \in [\nu]_2\}
\]

Semantics

- two-valued \( \nu \) is a model of \( D \) iff \( \nu(s) = \nu(\varphi_s) \) for all \( s \in S \)
- \( \nu \) is the grounded model of \( D \) iff it is the least fixpoint of \( \Gamma_D \)
Abstract Dialectical Frameworks
Semantics: Example

\[ \varphi_a = t \]
\[ \varphi_b = b \]
\[ \varphi_c = a \land b \]
\[ \varphi_d = \neg b \]

- models:
  - \( v_1 = \{ a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto f \} \equiv \{ a, b, c, \neg d \} \)
  - \( v_2 = \{ a \mapsto t, b \mapsto f, c \mapsto f, d \mapsto t \} \equiv \{ a, \neg b, \neg c, d \} \)
- grounded model: \( v_3 = \{ a \mapsto t, b \mapsto u, c \mapsto u, d \mapsto u \} \equiv \{ a \} \)

Definition: Admissible
Interpretation \( v \) is admissible for ADF \( D \) iff \( v \leq \Gamma_D(v) \).

- intuitively: does not contain too much information
- example ADF has 16 admissible interpretations:

\[ \{ a, b, c, \neg d \} \]
\[ \{ a, b, \neg d \} \]
\[ \{ a, \neg b, d \} \]
\[ \{ a, \neg b, \neg c \} \]
\[ \{ b, \neg d \} \]
\[ \{ b, \neg c \} \]
\[ \{ b \} \]
\[ \{ \} \]

S. Ellmauthaler, H. Strass
The DIAMOND System for Argumentation
ASPOCP 2013, Istanbul 15 / 28

Abstract Dialectical Frameworks
Complete Semantics

Definition: Complete
Interpretation \( v \) is complete for ADF \( D \) iff \( v = \Gamma_D(v) \).

- complete interpretations are stationary w.r.t. revision operator

\[ \{ a, b, c, \neg d \} \]
\[ \{ a, b, \neg d \} \]
\[ \{ a, \neg b, d \} \]
\[ \{ a, \neg b, \neg c \} \]
\[ \{ b, \neg d \} \]
\[ \{ b, \neg c \} \]
\[ \{ b \} \]
\[ \{ \} \]

S. Ellmauthaler, H. Strass
The DIAMOND System for Argumentation
ASPOCP 2013, Istanbul 16 / 28

DIAMOND
DIAleetical MOdels eNcoDing

- Set of encodings utilizing the Potsdam Answer Set Solving Collection\(^6\)
- Python script for command-line-interface
- ECL\textsuperscript{PS}\textsuperscript{e}-Prolog program for fast input-format conversions

\(^6\)M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider.
“Potassco: The Potsdam Answer Set Solving Collection”. In: AI Communications 24.2 (2011),
pages 105–124.
Propositional Formula Representation

- Statements are defined via the unary predicate \texttt{statement}(a).
- Their acceptance condition is a propositional formula, represented by \texttt{ac}(s, \phi), where \phi is a prop. formula in prefix notation.
- Links between statements are implicitly defined by variable occurrence in \phi.
- Allowed constants in \phi are c(v), c(f).
- Allowed operators are neg, and, or, imp, and iff.

Example

\begin{align*}
\text{statement}(a). \quad \text{statement}(b). \quad \text{statement}(c). \quad \text{statement}(d).
\text{ac}(a, \text{c(v)}). \quad \text{ac}(b, b).
\text{ac}(c, \text{and}(a, b)). \quad \text{ac}(d, \text{neg}(b)).
\end{align*}

Implementation of \( \Gamma_D \)

- Semantics are based upon fixpoint computation of \( \Gamma_D \).
- Binary predicates in(X,I) and out(X,I) represent an interpretation at step(I).
- Operator applied if step(I) exists.

Predicates

- Unary predicate \texttt{s}(a) to represent statements.
- Binary predicate \texttt{l}(b,a) to represent links.
- Unary and ternary predicates \texttt{ci} and \texttt{co} to model the Boolean acceptance function.

Example

\begin{align*}
\text{s}(a). \quad \text{s}(b). \quad \text{s}(c). \quad \text{s}(d).
\text{l}(a,c). \quad \text{l}(b,b). \quad \text{l}(b,c). \quad \text{l}(b,d).
\text{ci}(a).
\text{co}(b). \quad \text{ci}(b,1,b).
\text{co}(c). \quad \text{co}(c,1,a). \quad \text{co}(c,2,b). \quad \text{ci}(c,3,a). \quad \text{ci}(c,3,b).
\text{ci}(d). \quad \text{co}(d,1,b).
\end{align*}

“Interesting” links at step I

\begin{align*}
\text{ciui}(S,J,I) & : \text{lin}(X,S,I), \text{not} \text{ci}(S,J,X), \text{ci}(S,J).
\text{ciui}(S,J,I) & : \text{lout}(X,S,I), \text{ci}(S,J,X).
\text{cii}(S,J,I) & : \text{not} \text{ciui}(S,J,I), \text{ci}(S,J), \text{step}(I).
\end{align*}

Validity check

\begin{align*}
\text{pmodel}(S,I) & : \text{cii}(S,J,I). \quad \text{pmodel}(S,I) : \text{verum}(S), \text{step}(I).
\text{pmodel}(S,I) & : \text{not} \text{lin}(S,I), \text{ci}(S), \text{step}(I).
\text{valid}(S,I) & : \text{pmodel}(S,I), \text{not} \text{imodel}(S,I).
\end{align*}

Fixpoint check

\begin{align*}
\text{nofp}(I) & : \text{in}(X,I), \text{not} \text{valid}(X,I), \text{step}(I).
\text{nofp}(I) & : \text{valid}(X,I), \text{not} \text{in}(X,I), \text{step}(I).
\text{nofp}(I) & : \text{out}(X,I), \text{not} \text{unsat}(X,I), \text{step}(I).
\text{nofp}(I) & : \text{unsat}(X,I), \text{not} \text{out}(X,I), \text{step}(I).
\text{fp}(I) & : \text{not} \text{nofp}(I), \text{step}(I).
\end{align*}
Conclusion

- DIAMOND utilizes ASP for implementing abstract argumentation (like ASPARTIX\textsuperscript{7} does for AFs)
- All currently known representations for ADFs are understood by DIAMOND
- More complex semantics (e.g. preferred) can be implemented with Meta-ASP\textsuperscript{8}
- ADFsys\textsuperscript{9} is a similar system, but it
  - uses different semantics, and
  - it already needs disjunctive programs for the computation of $\Gamma_D$

\textsuperscript{7}Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. “Answer-set programming encodings for argumentation frameworks”. In: Argument and Computation 1.2 (2010), pages 147–177.

Thank you!
AF Semantics: Alternative Formulation via operators

**Definition**

For AF \( F = (A, R) \), define two operators

- \( U_F(S) = \{ a \in A \mid S \text{ does not attack } a \} \) (Pollock)
- \( V_F(S) = \{ a \in A \mid S \text{ defends } a \} \) (Dung)

- \( U_F \) is \( \subseteq \)-antimonotone (\( S_1 \subseteq S_2 \) implies \( U_F(S_2) \subseteq U_F(S_1) \))
- \( V_F \) is \( \subseteq \)-monotone (\( S_1 \subseteq S_2 \) implies \( V_F(S_1) \subseteq V_F(S_2) \))

**Lemma (Dung)**

For any AF \( F \), we have \( U_F^2 = V_F \).

---

**Abstract Dialectical Frameworks**

**Semantics: Example**

\[
\begin{align*}
\varphi_a &= t \\
\varphi_b &= b \\
\varphi_c &= a \land b \\
\varphi_d &= \neg b
\end{align*}
\]

- \( v_1 = \{ a, b, c, \neg d \} \): reduct \( D^{v_1} \) with grounded extension \( \{ a \} \), \( \{ a \} \neq E_{v_1} \), thus \( v_1 \) not stable (statements \( b \) and \( c \) unjustified)
- \( v_2 = \{ a, \neg b, \neg c, d \} \): reduct \( D^{v_2} \) with grounded extension \( \{ a, d \} = E_{v_2} \), thus \( v_2 \) stable
Thank you!