The DIAMOND System for Argumentation
Preliminary Report

Stefan Ellmauthaler    Hannes Strass

Computer Science Institute
Leipzig University
Germany

ASPOCP 2013
August 25th 2013
Outline

1 Motivation

2 Background
   - Abstract Argumentation Frameworks
   - Abstract Dialectical Frameworks

3 DIAMOND

4 Conclusion & Future Work
Outline

1 Motivation

2 Background
   - Abstract Argumentation Frameworks
   - Abstract Dialectical Frameworks

3 DIAMOND

4 Conclusion & Future Work
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

Abstract Dialectical Frameworks (ADFs)

- generalise AFs, arguments are now called \textit{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 \textit{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

---


Outline

1 Motivation

2 Background
   - Abstract Argumentation Frameworks
   - Abstract Dialectical Frameworks

3 DIAMOND

4 Conclusion & Future Work
Abstract Argumentation

is for determining acceptance of abstract arguments

- argumentation framework\(^4\) \( F = (A, R) \)
- \( A \) \( \ldots \) set of arguments
- \( R \subseteq A \times A \) \( \ldots \) attack relation
- argument \( a \in A \) is \textit{defended} by a set \( S \subseteq A \) if all attackers of \( a \) are attacked by \( S \):

\[
(\forall b \in A)(bRa \implies (\exists c \in S)cRb)
\]

AF Semantics
are defined via extensions

for AF $F = (A, R)$, 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** $F = (A, R)$ with $A = \{a, b, c, d\}$ and $R = \{(a, b), (c, d), (d, c)\}$:

  ![Graph](attachment:image.png)

- 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 Frameworks⁵

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
- $C = \{ C_s \}_{s \in S}$ ... acceptance conditions

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

---

$\varphi_a = t$

$\varphi_b = b$

$\varphi_c = a \land b$

$\varphi_d = \neg b$
Abstract Dialectical Frameworks

Semantics

Truth values, interpretations

- truth values: true $t$, false $f$, unknown $u$
- interpretation: $v : S \rightarrow \{t, f, u\}$
- interpretations can be represented as consistent sets of literals

Information ordering

- $u <_i t$ and $u <_i f$ (as usual $x \leq_i y$ iff $x <_i y$ 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:
  - $v_1 \leq_i v_2$ iff $v_1(s) \leq_i v_2(s)$ for all $s \in S$
Abstract Dialectical Frameworks

Semantics

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

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

### Semantics
- two-valued \( \nu \) is a model of \( D \) iff \( \nu(s) = \nu(\phi_s) \) 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

\[ \varphi_a = t \]  \quad \varphi_b = b 

\[ \varphi_c = a \land b \]  \quad \varphi_d = \neg b 

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

- grounded model: \( v_3 = \{ a \mapsto t, b \mapsto u, c \mapsto u, d \mapsto u \} \models \{ a \} \)
Abstract Dialectical Frameworks

Semantics: Example

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

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

Semantics: Example

\( \varphi_a = \top \)
\( \varphi_b = b \)
\( \varphi_c = a \land b \)
\( \varphi_d = \neg b \)

- models:
  - \( v_1 = \{ a \mapsto \top, b \mapsto \top, c \mapsto \top, d \mapsto \bot \} \models \{ a, b, c, \neg d \} \)
  - \( v_2 = \{ a \mapsto \top, b \mapsto \bot, c \mapsto \bot, d \mapsto \top \} \models \{ a, \neg b, \neg c, d \} \)
- grounded model: \( v_3 = \{ a \mapsto \top, b \mapsto \bot, c \mapsto \bot, d \mapsto \bot \} \models \{ a \} \)
Abstract Dialectical Frameworks

Semantics: Example

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

• models:
  
  ▶ \( v_1 = \{ a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f} \} \downarrow \{ a, b, c, \neg d \} \)
  
  ▶ \( v_2 = \{ a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t} \} \downarrow \{ a, \neg b, \neg c, d \} \)

• grounded model: \( v_3 = \{ a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u} \} \downarrow \{ a \} \)
Abstract Dialectical Frameworks
Admissible Semantics

**Definition: Admissible**

Interpretation $\nu$ is *admissible* for ADF $D$ iff $\nu \leq_{i} \Gamma_{D}(\nu)$.

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

1. Motivation

2. Background
   - Abstract Argumentation Frameworks
   - Abstract Dialectical Frameworks

3. DIAMOND

4. Conclusion & Future Work
Set of encodings utilizing the Potsdam Answer Set Solving Collection
Python script for command-line-interface
ECL\textsuperscript{i}PS\textsuperscript{e}-Prolog program for fast input-format conversions

Propositional Formula Representation

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

Example

```
ac(a,c(v)). ac(b,b).
ac(c, and(a,b)). ac(d, neg(b)).
```
DIAMOND

Instance Representation

Predicates

- unary predicate \( s(a) \). to represent statements
- binary predicate \( l(b,a) \). to represent links
- unary and ternary predicates \( ci \) and \( co \) to model the Boolean acceptance function

Example

\[
\begin{align*}
  s(a) & . \quad s(b) & . \quad s(c) & . \quad s(d) & . \\
  l(a,c) & . \quad l(b,b) & . \quad l(b,c) & . \quad l(b,d) & . \\
  ci(a) & . \\
  co(b) & . \quad ci(b,1,b) & . \\
  co(c) & . \quad co(c,1,a) & . \quad co(c,2,b) & . \quad ci(c,3,a) & . \quad ci(c,3,b) & . \\
  ci(d) & . \quad co(d,1,b) & .
\end{align*}
\]
### Predicates

- **unary predicate** `s(a)`. to represent statements
- **binary predicate** `l(b,a)`. to represent links
- **unary and ternary predicates** `ci` and `co` to model the Boolean acceptance function

### Example

```
s(a).  s(b).  s(c).  s(d).
l(a,c).  l(b,b).  l(b,c).  l(b,d).
ci(a).
co(b).  ci(b,1,b).
co(c).  co(c,1,a).  co(c,2,b).  ci(c,3,a).  ci(c,3,b).
ci(d).  co(d,1,b).
```
Implementation of $\Gamma_D$

- semantics are based upon fixpoint computation of $\Gamma_D$
- binary predicates $\text{in}(X,I)$ and $\text{out}(X,I)$ represent an interpretation at $\text{step}(I)$
- operator applied if $\text{step}(I)$ exists
DIAMOND
Implementation of $\Gamma_D$

“Interesting” links at step I

\begin{align*}
\text{ciui}(S,J,I) & : \text{lin}(X,S,I), \neg \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) & : \neg \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). \\
\text{pmodel}(S,I) & : \text{verum}(S), \text{step}(I). \\
\text{pmodel}(S,I) & : \neg \text{lin}(S,I), \text{ci}(S), \text{step}(I). \\
\text{valid}(S,I) & : \text{pmodel}(S,I), \neg \text{imodel}(S,I).
\end{align*}

Fixpoint check

\begin{align*}
\text{nofp}(I) & : \text{in}(X,I), \neg \text{valid}(X,I), \text{step}(I). \\
\text{nofp}(I) & : \neg \text{valid}(X,I), \text{not in}(X,I), \text{step}(I). \\
\text{nofp}(I) & : \text{out}(X,I), \neg \text{unsat}(X,I), \text{step}(I). \\
\text{nofp}(I) & : \text{unsat}(X,I), \text{not out}(X,I), \text{step}(I). \\
\text{fp}(I) & : \neg \text{nofp}(I), \text{step}(I).
\end{align*}
### “Interesting” links at step I

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ciui(S,J,I) :- lin(X,S,I), not ci(S,J,X), ci(S,J).</code></td>
<td>Link between <code>S</code> and <code>I</code>, not between <code>S</code> and <code>J</code>, and <code>ci(J,S)</code></td>
</tr>
<tr>
<td><code>ciui(S,J,I) :- lout(X,S,I), ci(S,J,X).</code></td>
<td>Link out from <code>S</code> to <code>I</code>, and <code>ci(J,S)</code></td>
</tr>
<tr>
<td><code>cii(S,J,I) :- not ciui(S,J,I), ci(S,J), step(I).</code></td>
<td>Not link <code>ciui</code>, <code>ci(S,J)</code>, step <code>I</code></td>
</tr>
</tbody>
</table>

### Validity check

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pmodel(S,I) :- cii(S,J,I).</code></td>
<td><code>pmodel</code> if <code>cii(S,J,I)</code></td>
</tr>
<tr>
<td><code>pmodel(S,I) :- verum(S), step(I).</code></td>
<td><code>pmodel</code> if <code>verum(S)</code> and step <code>I</code></td>
</tr>
<tr>
<td><code>pmodel(S,I) :- not lin(S,I), ci(S), step(I).</code></td>
<td>Not link <code>lin</code>, <code>ci(S)</code> and step <code>I</code></td>
</tr>
<tr>
<td><code>valid(S,I) :- pmodel(S,I), not imodel(S,I).</code></td>
<td><code>valid</code> if <code>pmodel(S,I)</code> and not <code>imodel(S,I)</code></td>
</tr>
</tbody>
</table>

### Fixpoint check

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nofp(I) :- in(X,I), not valid(X,I), step(I).</code></td>
<td>Not in <code>I</code>, not valid <code>X</code>, step <code>I</code></td>
</tr>
<tr>
<td><code>nofp(I) :- valid(X,I), not in(X,I), step(I).</code></td>
<td>Valid <code>X</code>, not in <code>I</code>, step <code>I</code></td>
</tr>
<tr>
<td><code>nofp(I) :- out(X,I), not unsat(X,I), step(I).</code></td>
<td>Not out <code>X</code>, not unsat <code>X</code>, step <code>I</code></td>
</tr>
<tr>
<td><code>nofp(I) :- unsat(X,I), not out(X,I), step(I).</code></td>
<td>Not unsat <code>X</code>, not out <code>X</code>, step <code>I</code></td>
</tr>
<tr>
<td><code>fp(I) :- not nofp(I), step(I).</code></td>
<td><code>fp</code> if not <code>nofp</code> and step <code>I</code></td>
</tr>
</tbody>
</table>
### “Interesting” links at step I

\[
\text{ciui}(S,J,I) \leftarrow \text{lin}(X,S,I), \text{not ci}(S,J,X), \text{ci}(S,J).
\]
\[
\text{ciui}(S,J,I) \leftarrow \text{lout}(X,S,I), \text{ci}(S,J,X).
\]
\[
\text{cii}(S,J,I) \leftarrow \text{not ciui}(S,J,I), \text{ci}(S,J), \text{step}(I).
\]

### Validity check

\[
\text{pmodel}(S,I) \leftarrow \text{cii}(S,J,I).
\]
\[
\text{pmodel}(S,I) \leftarrow \text{verum}(S), \text{step}(I).
\]
\[
\text{pmodel}(S,I) \leftarrow \text{not lin}(S,I), \text{ci}(S), \text{step}(I).
\]
\[
\text{valid}(S,I) \leftarrow \text{pmodel}(S,I), \text{not imodel}(S,I).
\]

### Fixpoint check

\[
\text{nAFP}(I) \leftarrow \text{in}(X,I), \text{not valid}(X,I), \text{step}(I).
\]
\[
\text{nAFP}(I) \leftarrow \text{valid}(X,I), \text{not in}(X,I), \text{step}(I).
\]
\[
\text{nAFP}(I) \leftarrow \text{out}(X,I), \text{not unsat}(X,I), \text{step}(I).
\]
\[
\text{nAFP}(I) \leftarrow \text{unsat}(X,I), \text{not out}(X,I), \text{step}(I).
\]
\[
\text{fp}(I) \leftarrow \text{not nAFP}(I), \text{step}(I).
\]
Admissible/Complete

\begin{verbatim}
step(0).
in(S,0):s(S).
out(S,0):s(S).
:- in(S,0), out(S,0).
:- in(S), not valid(S,0).
:- out(S), not unsat(S,0).
:- nofp(0).
\end{verbatim}

Grounded

\begin{verbatim}
maxit(I) :- I:=s(S). step(0).
in(S,I+1) :- valid(S,I). out(S,I+1) :- unsat(S,I).
step(I+1) :- step(I), not maxit(I).
in(S) :- fp(I), in(S,I).
out(S) :- fp(I), out(S,I).
udec(S) :- fp(I), s(S), not in(S), not out(S).
\end{verbatim}
### Admissible/Complete

```prolog
step(0).
in(S,0):s(S).
out(S,0):s(S).
:- in(S,0), out(S,0).
:- in(S), not valid(S,0).
:- out(S), not unsat(S,0).

:- nofp(0).
```

### Grounded

```prolog
maxit(I) :- I:=s(S). step(0).
in(S,I+1) :- valid(S,I). out(S,I+1) :- unsat(S,I).
step(I+1) :- step(I), not maxit(I).
in(S) :- fp(I), in(S,I).
out(S) :- fp(I), out(S,I).
udec(S) :- fp(I), s(S), not in(S), not out(S).
```
Outline

1 Motivation

2 Background
   ◆ Abstract Argumentation Frameworks
   ◆ Abstract Dialectical Frameworks

3 DIAMOND

4 Conclusion & Future Work
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

\begin{itemize}
  \item uses different semantics, and
  \item it already needs disjunctive programs for the computation of $\Gamma_D$
\end{itemize}

\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.


Future Work

- Implementation of further semantics for ADFs
- Add more usability to DIAMOND
- Use ADFs and DIAMOND for discussion analysis in the web
- ...
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 $U_F^2 = V_F$. 
AF Semantics: Alternative Formulation
are defined via extensions

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

- \textit{conflict-free} iff $S \subseteq U_F(S)$
- \textit{admissible} iff $S \subseteq U_F(S)$ and $S \subseteq V_F(S)$
- \textit{preferred} iff it is $\subseteq$-maximal admissible
- \textit{complete} iff $S \subseteq U_F(S)$ and $S = V_F(S)$
- \textit{grounded} iff it is the $\subseteq$-least fixpoint of $V_F$
- \textit{stable} iff $S = U_F(S)$
Stable Models
for ADFs

Definition: Associated Extension

For an interpretation $\nu$, the set $E_\nu = \{ s \in S \mid \nu(s) = t \}$ defines the unique extension associated with $\nu$.

Definition: Stable Model

Let $D = (S, L, C)$ be an ADF with $C = \{ \varphi_s \}_{s \in S}$. A two-valued model $\nu$ of $D$ is a stable model of $D$ iff $E_\nu$ equals the grounded extension of the reduced ADF $D^\nu = (E_\nu, L^\nu, C^\nu)$, where

- $L^\nu = L \cap (E_\nu \times E_\nu)$ and
- for $s \in E_\nu$ we set $\varphi_s^\nu = \varphi_s[b/f : \nu(b) = f]$. 
Abstract Dialectical Frameworks

Semantics: Example

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

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

Semantics: Example

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

- \(v_1 \hat{=} \{a, b, c, \neg d\}\): reduct \(D^{v_1}\) with grounded extension \(\{a\}\), \(\{a\} \neq E_{v_1}\), thus \(v_1\) not stable (statements \(b\) and \(c\) unjustified)
- \(v_2 \hat{=} \{a, \neg b, \neg c, d\}\): reduct \(D^{v_2}\) with grounded extension \(\{a, d\} = E_{v_2}\), thus \(v_2\) stable
Abstract Dialectical Frameworks
Semantics: Example

\( \varphi_a = t \)

\( \varphi_b = b \)

\( \varphi_c = a \land b \)

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

- \( v_2 \models \{ a, \neg b, \neg c, d \} \): reduct \( D^{v_2} \) with grounded extension \( \{ a, d \} = E_{v_2} \), thus \( v_2 \) stable
Abstract Dialectical Frameworks
Semantics: Example

\[ \varphi_a = t \quad \varphi_b = b \]

\[ \varphi_c = a \land b \]

\[ v_1 \hat{=} \{ a, b, c, \neg d \}: \text{reduct } D^{v_1} \text{ with grounded extension } \{ a \}, \]
\[ \{ a \} \neq E_{v_1}, \text{thus } v_1 \text{ not stable (statements } b \text{ and } c \text{ unjustified)} \]

\[ v_2 \hat{=} \{ a, \neg b, \neg c, d \}: \text{reduct } D^{v_2} \text{ with grounded extension } \{ a, d \} = E_{v_2}, \]
\[ \text{thus } v_2 \text{ stable} \]
Abstract Dialectical Frameworks

Semantics: Example

\( \varphi_a = \top \quad \varphi_b = b \)

\( \varphi_c = a \land b \quad \varphi_d = \neg b \)

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

Semantics: Example

\[ \varphi_a = \top \quad \varphi_b = b \]

\[ \varphi_c = a \land b \quad \varphi_d = \neg b \]

- \( v_1 \models \{a, b, c, \neg d\} \): reduct \( D^{v_1} \) with grounded extension \( \{a\} \), \( \{a\} \not\models E_{v_1} \), thus \( v_1 \) not stable (statements \( b \) and \( c \) unjustified)

- \( v_2 \models \{a, \neg b, \neg c, d\} \): reduct \( D^{v_2} \) with grounded extension \( \{a, d\} = E_{v_2} \), thus \( v_2 \) stable
Abstract Dialectical Frameworks
Semantics: Example

\( \varphi_a = t \)

\( a \)

\( \varphi_d = \neg f \)

\( d \)

- \( v_1 \upmodels \{a, b, c, \neg d\} \): reduct \( D^{v_1} \) with grounded extension \( \{a\} \), \( \{a\} \neq E_{v_1} \), thus \( v_1 \) not stable (statements \( b \) and \( c \) unjustified)
- \( v_2 \upmodels \{a, \neg b, \neg c, d\} \): reduct \( D^{v_2} \) with grounded extension \( \{a, d\} = E_{v_2} \), thus \( v_2 \) stable
Abstract Dialectical Frameworks
Semantics: Example

\[ \varphi_a = t \]

\[ \varphi_d = \neg f \]

\[ v_1 \models \{ a, b, c, \neg d \} \]: reduct \( D^{v_1} \) with grounded extension \( \{ a \} \), \( \{ a \} \neq E_{v_1} \), thus \( v_1 \) not stable (statements \( b \) and \( c \) unjustified)

\[ v_2 \models \{ a, \neg b, \neg c, d \} \]: reduct \( D^{v_2} \) with grounded extension \( \{ a, d \} = E_{v_2} \), thus \( v_2 \) stable
Abstract Dialectical Frameworks
Semantics: Example

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

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