The DIAMOND System for Argumentation

Preliminary Report

Stefan Ellmauthaler Hannes Strass

Computer Science Institute
Leipzig University
Germany

ASPOCP 2013
August 25^{th} 2013

UNIVERSITÄT LEIPZIG

Outline

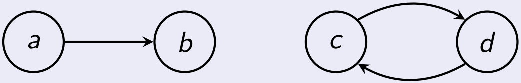
Outline

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[1]
- argumentation for discussion analysis on social media (e.g. facebook)[2]
- argumentation for blog-discussions[3]

[1] Marcello Ceci and Thomas Gordon. "Browsing Case-law: an Application of the Carneades Argumentation System". In: *Proceedings of the RuleML2012@ECAI Challenge, at the 6th International Symposium on Rules, Montpellier, France, August 27th-29th, 2012*. Edited by Hassan Aït-Kaci, Yuh-Jong Hu, Grzegorz J. Nalepa, Monica Palmirani, and Dumitru Roman. Volume 874. CEUR Workshop Proceedings. CEUR-WS.org, 2012.

[2] Francesca Toni and Paolo Torroni. "Bottom-Up Argumentation". In: *TAFA*. edited by Sanjay Modgil, Nir Oren, and Francesca Toni. Volume 7132. Lecture Notes in Computer Science. Springer, 2011, pages 249–262. ISBN: 978-3-642-29183-8.

[3] Mark Snaith, Floris Bex, John Lawrence, and Chris Reed. "Implementing ArguBlogging". In: *COMMA*. edited by Bart Verheij, Stefan Szeider, and Stefan Woltran. Volume 245. Frontiers in Artificial Intelligence and Applications. IOS Press, 2012, pages 511–512. ISBN: 978-1-61499-110-6.

Outline

Abstract Argumentation

is for determining acceptance of abstract arguments

- argumentation framework[4] $F = (A, R)$
- 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:

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

[4] Phan Minh Dung. "On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games". In: *Artificial Intelligence* 77 (1995), pages 321–358.

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 $R = \{(a, b), (c, d), (d, c)\}$:

- 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[5]

Syntax

Definition: Abstract Dialectical Framework

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

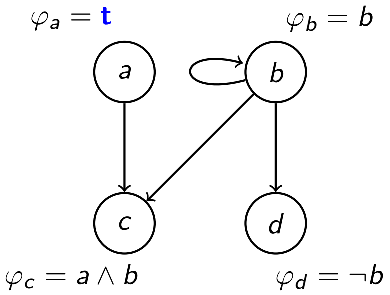
- S ... set of statements (correspond to AF arguments)
- $L \subseteq S \times S$ ... links $(par(s) = L^{-1}(s))$
- $C = \{C_s\}_{s \in S}$ ... acceptance conditions

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

[5] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. "Abstract Dialectical Frameworks Revisited". In: *IJCAI*. to appear. Beijing, China: AAAI Press, Aug. 2013.

Abstract Dialectical Frameworks

Example



Truth values, interpretations

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

Information ordering

- $\mathbf{u} <_i \mathbf{t}$ and $\mathbf{u} <_i \mathbf{f}$ (as usual $x \leq_i y$ iff $x <_i y$ or $x = y$)
- *consensus* $\sqcap$ is greatest lower bound w.r.t. $\leq_i$:
  $\mathbf{t} \sqcap \mathbf{t} = \mathbf{t}$ and $\mathbf{f} \sqcap \mathbf{f} = \mathbf{f}$, otherwise $x \sqcap y = \mathbf{u}$
- information ordering generalised to interpretations:
  $v_1 \leq_i v_2$ iff $v_1(s) \leq_i v_2(s)$ for all $s \in S$

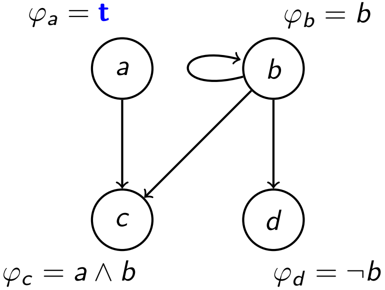
Characteristic Operator

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

$$\Gamma_D(v) : S \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \qquad s \mapsto \bigsqcap \{w(\varphi_s) \mid w \in [v]_2\}$$

Semantics

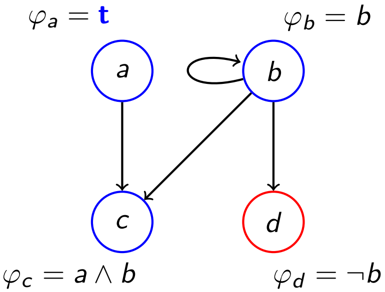
- two-valued v is a model of D iff $v(s) = v(\varphi_s)$ for all $s \in S$
- v is the grounded model of D iff it is the least fixpoint of Γ_D



- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \hat{=} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \hat{=} \{a, \neg b, \neg c, d\}$
- grounded model: $v_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\} \hat{=} \{a\}$

Abstract Dialectical Frameworks

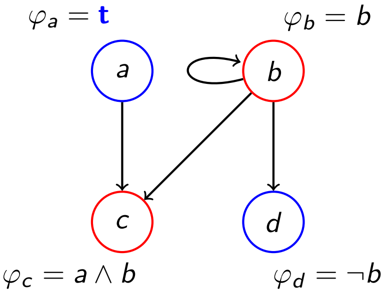
Semantics: Example



- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \mathrel{\hat{=}} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \mathrel{\hat{=}} \{a, \neg b, \neg c, d\}$
- grounded model: $v_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\} \mathrel{\hat{=}} \{a\}$

Abstract Dialectical Frameworks

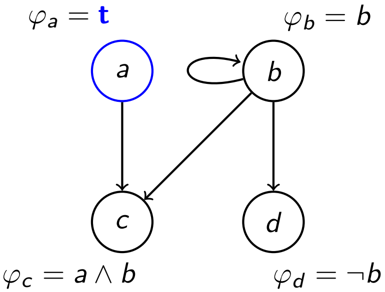
Semantics: Example



- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \mathrel{\hat{=}} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \mathrel{\hat{=}} \{a, \neg b, \neg c, d\}$
- grounded model: $v_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\} \mathrel{\hat{=}} \{a\}$

Abstract Dialectical Frameworks

Semantics: Example

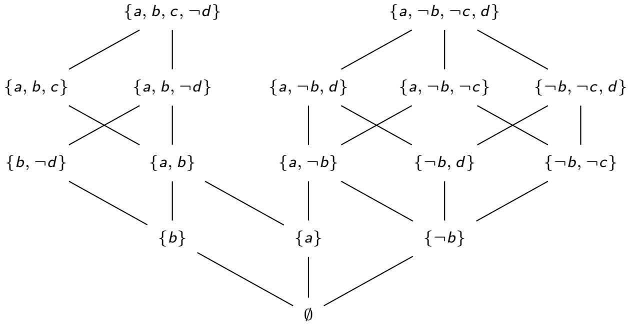


- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \mathrel{\hat{=}} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \mathrel{\hat{=}} \{a, \neg b, \neg c, d\}$
- grounded model: $v_3 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\} \mathrel{\hat{=}} \{a\}$

Definition: Admissible

Interpretation v is *admissible* for ADF D iff $v \leq_i \Gamma_D(v)$.

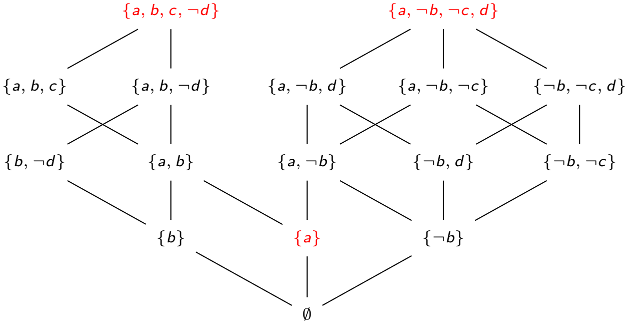
- intuitively: does not contain too much information
- example ADF has 16 admissible interpretations:



Definition: Complete

Interpretation v is *complete* for ADF D iff $v = \Gamma_D(v)$.

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



Outline

DIAMOND

DIAlectical MOdels eNcoDing

- Set of encodings utilizing the Potsdam Answer Set Solving Collection[6]
- Python script for command-line-interface
- ECL^{i}PS^{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 `statement(a).`
- Their acceptance condition is a propositional formula, represented by `ac(s,`ϕ`)`, where ϕ is a prop. 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

```
statement(a).  statement(b).  statement(c).  statement(d).
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

```
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).
```

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

```
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).
```

DIAMOND

Implementation of Γ_D

- semantics are based upon fixpoint computation of Γ_D
- binary predicates `in(X,I)` and `out(X,I)` represent an interpretation at `step(I)`
- operator applied if `step(I)` exists

DIAMOND

Implementation of Γ_D

"Interesting" links at step I

```
ciui(S,J,I) :- lin(X,S,I), not ci(S,J,X), ci(S,J).
ciui(S,J,I) :- lout(X,S,I), ci(S,J,X).
cii(S,J,I) :- not ciui(S,J,I), ci(S,J), step(I).
```

Validity check

```
pmodel(S,I) :- cii(S,J,I). pmodel(S,I) :- verum(S), step(I).
pmodel(S,I) :- not lin(S,I), ci(S), step(I).
valid(S,I) :- pmodel(S,I), not imodel(S,I).
```

Fixpoint check

```
nofp(I) :- in(X,I), not valid(X,I), step(I).
nofp(I) :- valid(X,I), not in(X,I), step(I).
nofp(I) :- out(X,I), not unsat(X,I), step(I).
nofp(I) :- unsat(X,I), not out(X,I), step(I).
fp(I) :- not nofp(I), step(I).
```

"Interesting" links at step I

```
ciui(S,J,I) :- lin(X,S,I), not ci(S,J,X), ci(S,J).
ciui(S,J,I) :- lout(X,S,I), ci(S,J,X).
cii(S,J,I) :- not ciui(S,J,I), ci(S,J), step(I).
```

Validity check

```
pmodel(S,I) :- cii(S,J,I). pmodel(S,I) :- verum(S), step(I).
pmodel(S,I) :- not lin(S,I), ci(S), step(I).
valid(S,I) :- pmodel(S,I), not imodel(S,I).
```

Fixpoint check

```
nofp(I) :- in(X,I), not valid(X,I), step(I).
nofp(I) :- valid(X,I), not in(X,I), step(I).
nofp(I) :- out(X,I), not unsat(X,I), step(I).
nofp(I) :- unsat(X,I), not out(X,I), step(I).
fp(I) :- not nofp(I), step(I).
```

DIAMOND

Implementation of Γ_D

"Interesting" links at step I

```
ciui(S,J,I) :- lin(X,S,I), not ci(S,J,X), ci(S,J).
ciui(S,J,I) :- lout(X,S,I), ci(S,J,X).
cii(S,J,I) :- not ciui(S,J,I), ci(S,J), step(I).
```

Validity check

```
pmodel(S,I) :- cii(S,J,I). pmodel(S,I) :- verum(S), step(I).
pmodel(S,I) :- not lin(S,I), ci(S), step(I).
valid(S,I) :- pmodel(S,I), not imodel(S,I).
```

Fixpoint check

```
nofp(I) :- in(X,I), not valid(X,I), step(I).
nofp(I) :- valid(X,I), not in(X,I), step(I).
nofp(I) :- out(X,I), not unsat(X,I), step(I).
nofp(I) :- unsat(X,I), not out(X,I), step(I).
fp(I) :- not nofp(I), step(I).
```

Admissible/Complete

```
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

```
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).
```

DIAMOND

Semantics

Admissible/Complete

```
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

```
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

Conclusion

- DIAMOND utilizes ASP for implementing abstract argumentation (like ASPARTIX[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[8]
- ADFsys[9] is a similar system, but it
  - uses different semantics, and
  - it already needs disjunctive programs for the computation of Γ_D

---

[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.

[8] M. Gebser, R. Kaminski, and T. Schaub. "Complex Optimization in Answer Set Programming". In: *Theory and Practice of Logic Programming* 11.4–5 (2011), pages 821–839.

[9] Stefan Ellmauthaler and Johannes Peter Wallner. "Evaluating Abstract Dialectical Frameworks with ASP". . In: *COMMA*. edited by Bart Verheij, Stefan Szeider, and Stefan Woltran. Volume 245. Frontiers in Artificial Intelligence and Applications. IOS Press, 2012, pages 505–506. ISBN: 978-1-61499-110-6.

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

- *conflict-free* iff $S \subseteq U_F(S)$
- *admissible* iff $S \subseteq U_F(S)$ and $S \subseteq V_F(S)$
- *preferred* iff it is $\subseteq$-maximal admissible
- *complete* iff $S \subseteq U_F(S)$ and $S = V_F(S)$
- *grounded* iff it is the $\subseteq$-least fixpoint of V_F
- *stable* iff $S = U_F(S)$

Stable Models

for ADFs

Definition: Associated Extension

- For an interpretation v, the set $E_v = \{s \in S \mid v(s) = \mathbf{t}\}$ defines the unique *extension* associated with v.

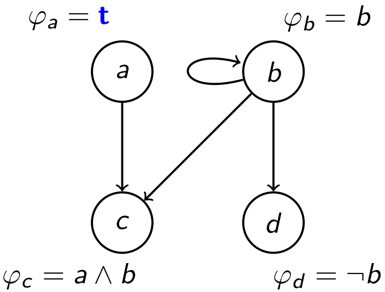
Definition: Stable Model

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

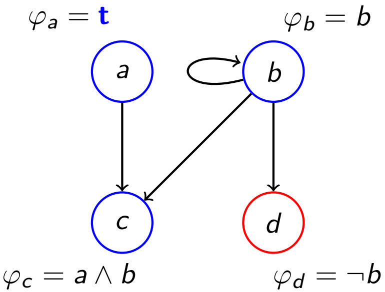
- $L^v = L \cap (E_v \times E_v)$ and
- for $s \in E_v$ we set $\varphi_s^v = \varphi_s[b/\mathbf{f} : v(b) = \mathbf{f}]$.

Abstract Dialectical Frameworks

Semantics: Example



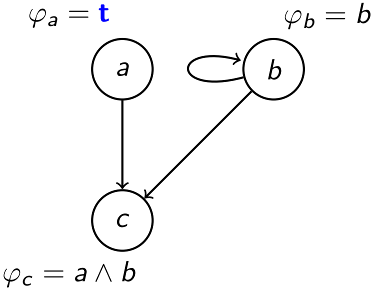
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, $\{a\} \neq E_{v_1}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\} = E_{v_2}$, thus v_2 stable



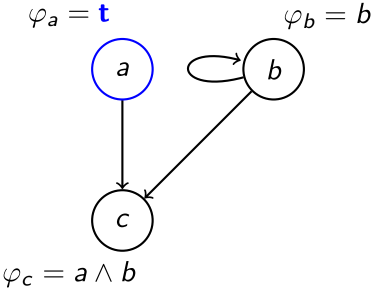
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, $\{a\} \neq E_{v_1}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\} = E_{v_2}$, thus v_2 stable

Abstract Dialectical Frameworks

Semantics: Example



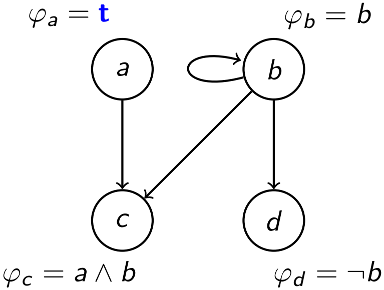
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, $\{a\} \neq E_{v_1}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\} = E_{v_2}$, thus v_2 stable



- $v_1 \hat{=} \{a, b, c, \neg d\}$: 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

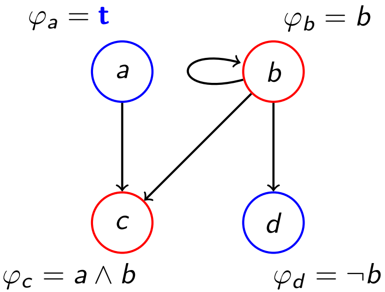
Semantics: Example



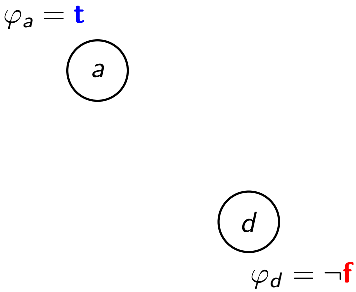
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, $\{a\} \neq E_{v_1}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\} = E_{v_2}$, thus v_2 stable

Abstract Dialectical Frameworks

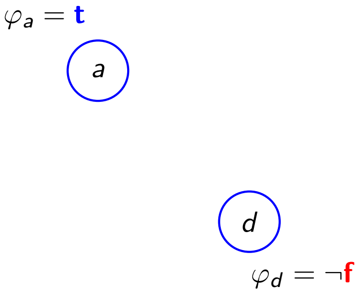
Semantics: Example



- $v_1 \hat{=} \{a, b, c, \neg d\}$: 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



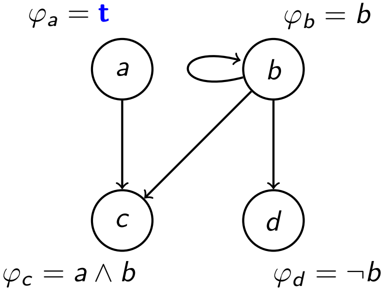
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, $\{a\} \neq E_{v_1}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\} = E_{v_2}$, thus v_2 stable



- $v_1 \hat{=} \{a, b, c, \neg d\}$: 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



- $v_1 \hat{=} \{a, b, c, \neg d\}$: 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

Thank you!