Abstract Dialectical Frameworks and Their Potential for (Legal) Argumentation

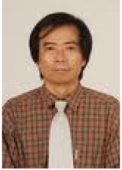
Gerhard Brewka

Computer Science Institute
University of Leipzig
brewka@informatik.uni-leipzig.de

joint work with S. Ellmauthaler, H. Strass, J. Wallner, S. Woltran

Where we are

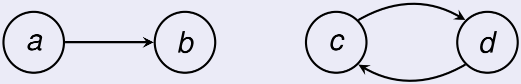


- Argumentation a highly active area in Artificial Intelligence.
- Dung's abstract argumentation frameworks (AFs) a *gold standard* in argumentation.
- Provide account of how to select acceptable arguments given arguments and attacks among them.
- Abstract away from everything but attacks: calculus of opposition
- Can be instantiated in many different ways.
- Useful semantical tool and target system for translations.

More on AFs

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

AF Semantics

Let $F = (A, R)$ be an argumentation framework, $S \subseteq A$.

- S is *conflict-free* iff no element of S attacks an element in S.
- $a \in A$ is *defended* by S iff all attackers of a are attacked by an element of S.
- 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.

Remark: iterative construction of grounded extension: start with $\emptyset$, include arguments not attacked at all, remove arguments attacked by included ones, continue until nothing changes.

AF Semantics

- 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

Common Use of AFs in Argumentation

- Prototypical example: Prakken's *ASPIC*$^+$
- KB consisting of strict/defeasible rules, preferences, proof standards etc.
- Compiled into adequate arguments and attacks
- Resulting AF provides system with choice of semantics

- Goal: more general target system; compilation made easy/trivial

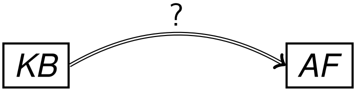
Common Use of AFs in Argumentation

- Prototypical example: Prakken's *ASPIC*$^{+}$
- KB consisting of strict/defeasible rules, preferences, proof standards etc.
- Compiled into adequate arguments and attacks
- Resulting AF provides system with choice of semantics

- Goal: more general target system; compilation made easy/trivial

Common Use of AFs in Argumentation

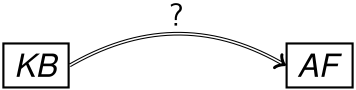
- Prototypical example: Prakken's $ASPIC^{+}$
- KB consisting of strict/defeasible rules, preferences, proof standards etc.
- Compiled into adequate arguments and attacks
- Resulting AF provides system with choice of semantics



- Goal: more general target system; compilation made easy/trivial

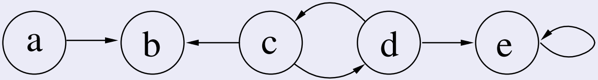
Common Use of AFs in Argumentation

- Prototypical example: Prakken's *ASPIC*$^+$
- KB consisting of strict/defeasible rules, preferences, proof standards etc.
- Compiled into adequate arguments and attacks
- Resulting AF provides system with choice of semantics



- Goal: more general target system; compilation made easy/trivial

Restrictions of AFs



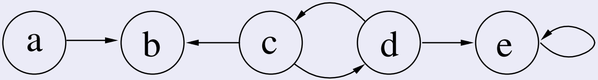
- fixed meaning of links: attack
- fixed acceptance condition for args: no parent accepted
- want more flexibility:
  1. links supporting arguments/positions
  2. nodes not accepted unless supported
  3. flexible means of combining attack and support
- from *calculus of opposition* to *calculus of support and opposition*

Abstract Dialectical Framework

=

Dependency Graph + Acceptance Conditions

Restrictions of AFs



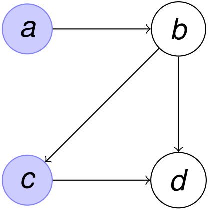
- fixed meaning of links: attack
- fixed acceptance condition for args: no parent accepted
- want more flexibility:
  1. links supporting arguments/positions
  2. nodes not accepted unless supported
  3. flexible means of combining attack and support
- from *calculus of opposition* to *calculus of support and opposition*

Abstract Dialectical Framework

=

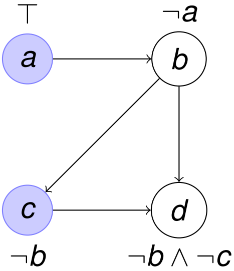
Dependency Graph + Acceptance Conditions

Basic idea



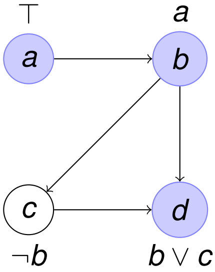
An Argumentation Framework

Basic idea



An Argumentation Framework
with explicit acceptance conditions

Basic idea



A Dialectical Framework
with flexible acceptance conditions

Outline

1. Background and Motivation (done)
2. What are ADFs?
3. What are They Good For? Semantical Tool for Graphical Argumentation Models
   - Pro/Con Arguments
   - Priorities
   - Weights
   - Proof Standards
4. A Case Study: Reconstructing Carneades
5. Conclusions

2. Abstract Dialectical Frameworks

Syntax

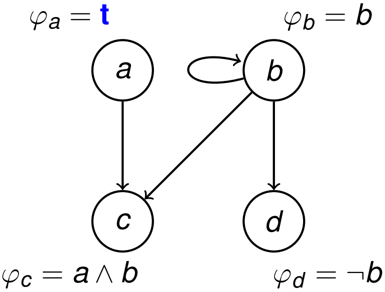
Definition: Abstract Dialectical Framework

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

- S ... set of statements, arguments; anything one might accept
- $L \subseteq S \times S$ ... links
- $C = \{\varphi_s\}_{s \in S}$ ... acceptance conditions

- links denote a dependency
- acceptance condition: defines truth value for s based on truth values of its parents
- specified as propositional formula φ_s

Abstract Dialectical Frameworks: Example



Abstract Dialectical Frameworks: Semantics

Based on 3-valued analysis; gives a handle on what is yet unknown.

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 $s \in S$

The Characteristic Operator

- Takes interpretation v and produces a new (revised) one v'.
- v' makes a node s
  - **t** iff acceptance condition true under any 2-valued completion of v,
  - **f** iff acceptance condition false under any 2-valued completion of v,
  - **u** otherwise.
- Operator thus checks what can be justified based on v.
  - Can information in v be justified?
  - Can further information be justified?

Characteristic Operator Γ_D

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

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

Semantics via Fixed Points

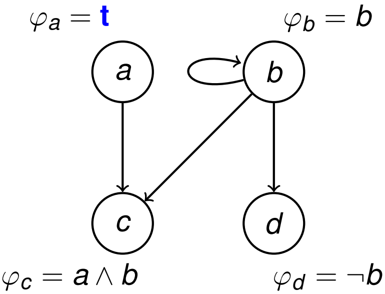
2 valued models

- A two-valued interpretation v is a model of D iff $\Gamma_D(v) = v$.
- Intuition: a statement is **t** iff its acceptance condition says so.

Grounded model

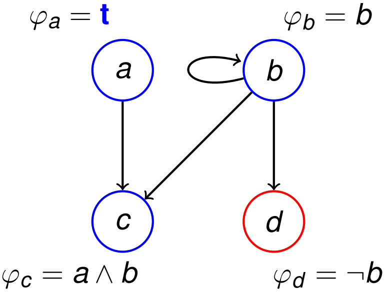
- v is the grounded model of D iff it is the least fixpoint of Γ_D.
- Intuition: obtained by iterating Γ_D on completely uninformed interpretation; collects information beyond doubt.

Abstract Dialectical Frameworks: Example



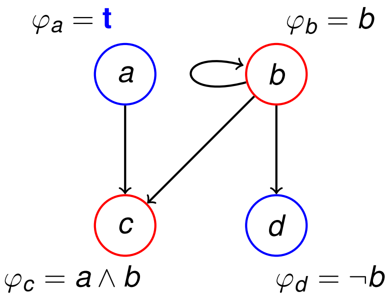
- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \hat{=} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \hat{=} \{a, \neg b, \neg c, d\}$
- grounded model: $v_2 =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

Abstract Dialectical Frameworks: Example



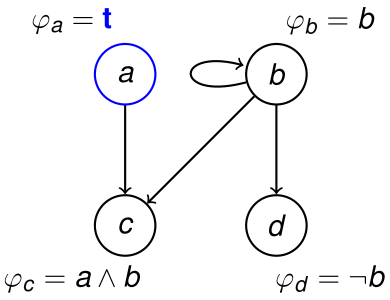
- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \hat{=} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \hat{=} \{a, \neg b, \neg c, d\}$
- grounded model: $v_2 =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

Abstract Dialectical Frameworks: Example



- 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: $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}, c \mapsto \mathbf{u}, d \mapsto \mathbf{u}\} \hat{=} \{a\}$

Abstract Dialectical Frameworks: Example



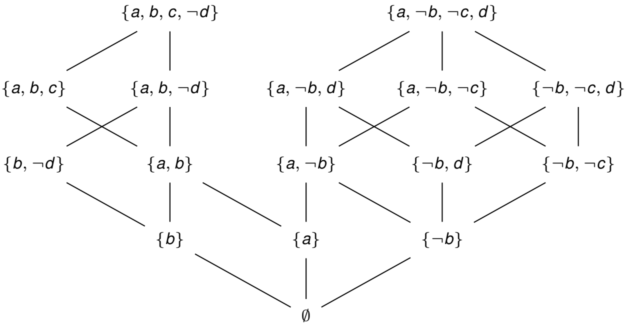
- models:
  - $v_1 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{t}, c \mapsto \mathbf{t}, d \mapsto \mathbf{f}\} \hat{=} \{a, b, c, \neg d\}$
  - $v_2 = \{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}, c \mapsto \mathbf{f}, d \mapsto \mathbf{t}\} \hat{=} \{a, \neg b, \neg c, d\}$
- grounded model: $v_2 =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

Further Semantics: Admissible

Definition: Admissible

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

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

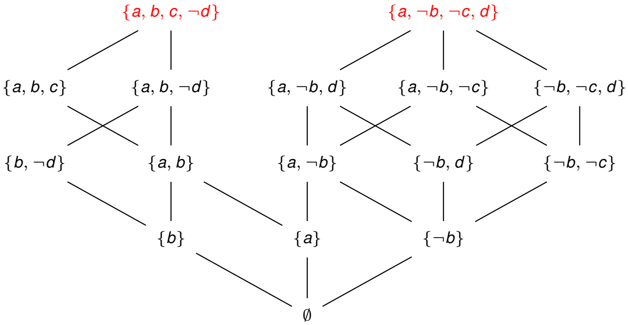


Preferred

Definition: Preferred

- Interpretation v is *preferred* for ADF D iff it is $\leq_i$-maximal admissible for D.

- intuitively: want maximal information content

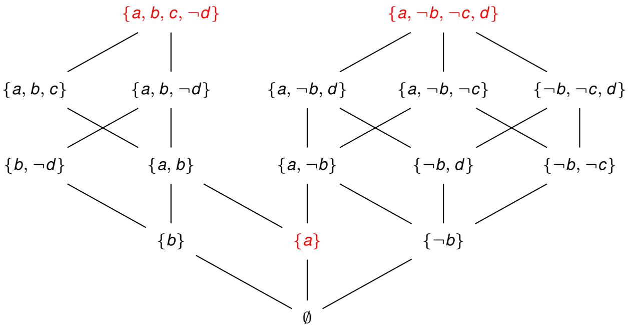


Complete

Definition: Complete

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

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



Stable Models for ADFs

Based on ideas from Logic Programming:

- no self-justifying cycles,
- achieved by reduct-based check.

To check whether a two-valued model v of D is *stable* do the following:

- eliminate in D all nodes with value **f** and corresponding links,
- replace eliminated nodes in acceptance conditions by **f**,
- check whether nodes **t** in v coincide with grounded model of reduced ADF.

Stable Models for ADFs

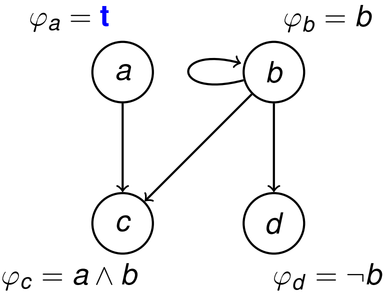
Based on ideas from Logic Programming:

- no self-justifying cycles,
- achieved by reduct-based check.

To check whether a two-valued model v of D is *stable* do the following:

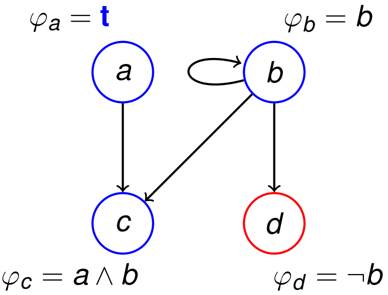
- eliminate in D all nodes with value **f** and corresponding links,
- replace eliminated nodes in acceptance conditions by **f**,
- check whether nodes **t** in v coincide with grounded model of reduced ADF.

Stable Models: Example



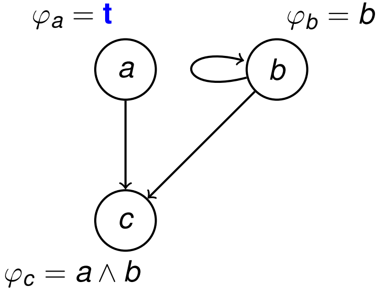
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



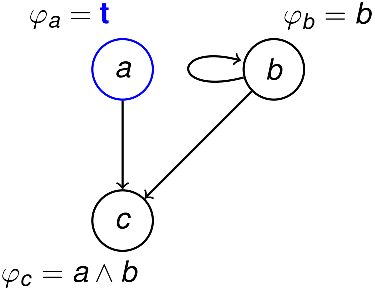
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



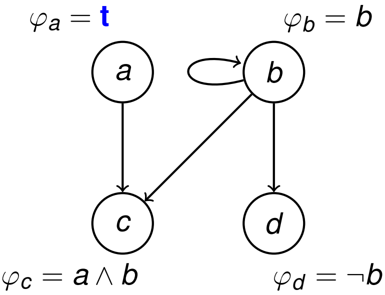
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



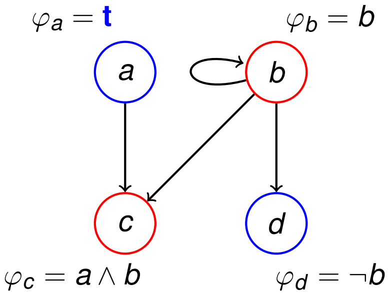
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



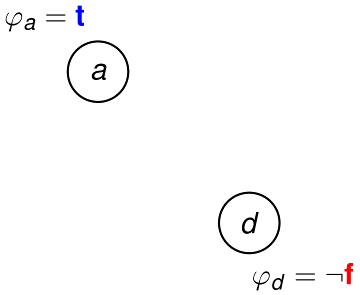
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



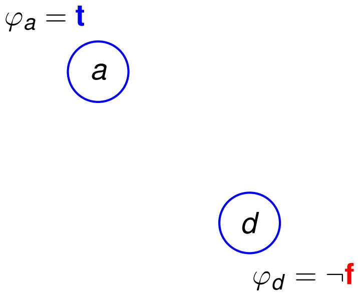
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



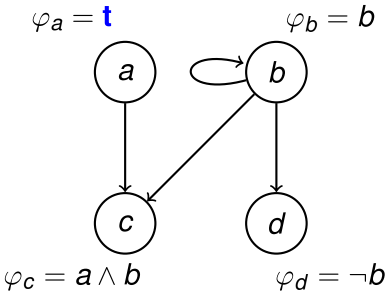
- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, thus v_1 not stable (statements b and c unjustified)
- $v_2 \hat{=} \{a, \neg b, \neg c, d\}$: reduct D^{v_2} with grounded extension $\{a, d\}$, thus v_2 stable

Stable Models: Example



- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, 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 $\{a, d\}$, thus v_2 stable

Stable Models: Example



- $v_1 \hat{=} \{a, b, c, \neg d\}$: reduct D^{v_1} with grounded extension $\{a\}$, 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 $\{a, d\}$, thus v_2 stable

Results

- ADFs properly generalize AFs.
- All major semantics available.
- Many results carry over, eg. the following inclusions hold:

$$sta(D) \subseteq val_2(D) \subseteq pref(D) \subseteq com(D) \subseteq adm(D).$$

- for ADFs corresponding to AFs models and stable models coincide (as AFs cannot express support).

Results

- ADFs properly generalize AFs.
- All major semantics available.
- Many results carry over, eg. the following inclusions hold:

$$sta(D) \subseteq val_2(D) \subseteq pref(D) \subseteq com(D) \subseteq adm(D).$$

- for ADFs corresponding to AFs models and stable models coincide (as AFs cannot express support).

Results

- ADFs properly generalize AFs.
- All major semantics available.
- Many results carry over, eg. the following inclusions hold:

$$sta(D) \subseteq val_2(D) \subseteq pref(D) \subseteq com(D) \subseteq adm(D).$$

- for ADFs corresponding to AFs models and stable models coincide (as AFs cannot express support).

Results

- ADFs properly generalize AFs.
- All major semantics available.
- Many results carry over, eg. the following inclusions hold:

$$sta(D) \subseteq val_2(D) \subseteq pref(D) \subseteq com(D) \subseteq adm(D).$$

- for ADFs corresponding to AFs models and stable models coincide (as AFs cannot express support).

3. What are ADFs Good For?

- Are ADFs needed?
  - Depends on meaning of “needed".
  - Can compile many (not all) things down to AFs.
  - Are high level programming languages needed?
- Are ADFs abstract locution frameworks (ALFs)?
  - Modgil: extensions of AFs best viewed as representing locutions.
  - ADFs explicitly not viewed as ALFs: do not represent information the way people would.
- Our view: semantical tool for graphical argumentation models
  - Not a direct representation formalism: argumentation middleware.
  - Need interface to be used by lay users.
  - Highly useful for argumentation models based on annotated graphs.

3. What are ADFs Good For?

- Are ADFs needed?
  - Depends on meaning of “needed".
  - Can compile many (not all) things down to AFs.
  - Are high level programming languages needed?
- Are ADFs abstract locution frameworks (ALFs)?
  - Modgil: extensions of AFs best viewed as representing locutions.
  - ADFs explicitly not viewed as ALFs: do not represent information the way people would.
- Our view: semantical tool for graphical argumentation models
  - Not a direct representation formalism: argumentation middleware.
  - Need interface to be used by lay users.
  - Highly useful for argumentation models based on annotated graphs.

3. What are ADFs Good For?

- Are ADFs needed?
  - Depends on meaning of “needed".
  - Can compile many (not all) things down to AFs.
  - Are high level programming languages needed?
- Are ADFs abstract locution frameworks (ALFs)?
  - Modgil: extensions of AFs best viewed as representing locutions.
  - ADFs explicitly not viewed as ALFs: do not represent information the way people would.
- Our view: semantical tool for graphical argumentation models
  - Not a direct representation formalism: argumentation middleware.
  - Need interface to be used by lay users.
  - Highly useful for argumentation models based on annotated graphs.

3. What are ADFs Good For?

- Are ADFs needed?
  - Depends on meaning of “needed".
  - Can compile many (not all) things down to AFs.
  - Are high level programming languages needed?
- Are ADFs abstract locution frameworks (ALFs)?
  - Modgil: extensions of AFs best viewed as representing locutions.
  - ADFs explicitly not viewed as ALFs: do not represent information the way people would.
- Our view: semantical tool for graphical argumentation models
  - Not a direct representation formalism: argumentation middleware.
  - Need interface to be used by lay users.
  - Highly useful for argumentation models based on annotated graphs.

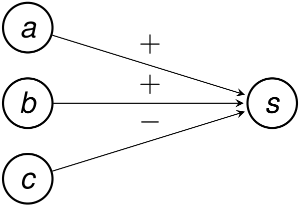
3. What are ADFs Good For?

- Are ADFs needed?
  - Depends on meaning of “needed".
  - Can compile many (not all) things down to AFs.
  - Are high level programming languages needed?
- Are ADFs abstract locution frameworks (ALFs)?
  - Modgil: extensions of AFs best viewed as representing locutions.
  - ADFs explicitly not viewed as ALFs: do not represent information the way people would.
- Our view: semantical tool for graphical argumentation models
  - Not a direct representation formalism: argumentation middleware.
  - Need interface to be used by lay users.
  - Highly useful for argumentation models based on annotated graphs.

3. What are ADFs Good For?

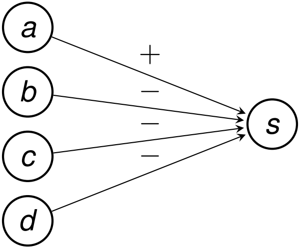
- Are ADFs needed?
  - Depends on meaning of “needed".
  - Can compile many (not all) things down to AFs.
  - Are high level programming languages needed?
- Are ADFs abstract locution frameworks (ALFs)?
  - Modgil: extensions of AFs best viewed as representing locutions.
  - ADFs explicitly not viewed as ALFs: do not represent information the way people would.
- Our view: semantical tool for graphical argumentation models
  - Not a direct representation formalism: argumentation middleware.
  - Need interface to be used by lay users.
  - Highly useful for argumentation models based on annotated graphs.

Specifying ADFs: Link Types, Acceptance Patterns



- Positive and negative links
- Pattern for deriving acceptance condition of s:
  - no negative and all positive links active: $\neg c \wedge (a \wedge b)$
  - no negative and at least one positive link active: $\neg c \wedge (a \vee b)$
  - more positive than negative links active: $(\neg c \wedge (a \vee b)) \vee (a \wedge b)$
- Pattern can be defined individually for each node

Specifying ADFs 2: Qualitative Preferences

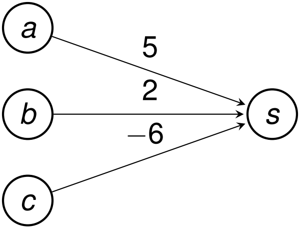


- Assume $a > c$ and $s > d$.
- Intuition: attacker does not succeed if attacked node more preferred or there is a more preferred supporting node.
- ϕ_s: conjunction of implications, one for each attacker not less preferred than s. Left side attacker, right side disjunction of more preferred supporting nodes.
- Here: $(b \rightarrow \mathbf{f}) \wedge (c \rightarrow a)$ or, equivalently $\neg b \wedge (c \rightarrow a)$.

Dynamic Preferences

- Often preferences not given in advance, but a matter of debate.
- Established dynamically in the course of argumentation.
- Assume some nodes represent (possibly conflicting) preference information.
- Handle dynamic preferences as follows:
  - Guess a (stable, preferred, grounded) model M.
  - Extract preference information from respective nodes in M.
  - Check whether M can be reconstructed under this (now static) preference information.
- Verifies preferences represented in the model itself were taken into account adequately.

Specifying ADFs: Weights



- Positive and negative weights
- Pattern for deriving acceptance condition:
  - sum of weights of active links positive: $(\neg c \wedge (a \vee b)) \vee (a \wedge b)$
  - maximal positive weight higher than maximal negative weight: $(\neg c \wedge (a \vee b))$
  - ...

Legal Proof Standards: Farley and Freeman

Introduced (1995) model of legal argumentation which distinguishes 4 types of arguments:

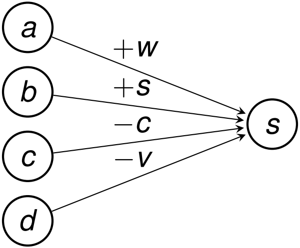
- *valid* arguments based on deductive inference,
- *strong* arguments based on inference with defeasible rules,
- *credible* arguments where premises give some evidence,
- *weak* arguments based on abductive reasoning.

By using values $V = \{+v, +s, +c, +w, -v, -s, -c, -w\}$ we can distinguish pro and con links of corresponding types.

Farley and Freeman's proof standards

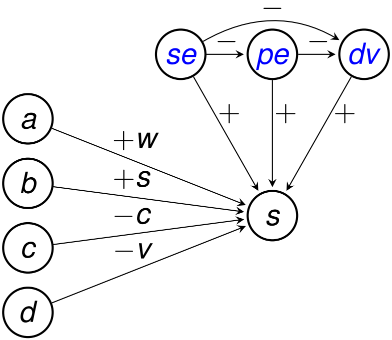
- *Scintilla of Evidence*: at least one active pro-argument.
- *Preponderance of Evidence*: at least one active pro-argument, the other side's arguments outweighed.
- *Dialectical Validity*: at least one credible, active pro-argument, the other side's arguments all inactive.
- *Beyond Reasonable Doubt*: at least one strong, active pro-argument, the other side's arguments all inactive.
- *Beyond Doubt*: at least one valid active pro-argument, the other side's arguments all inactive.

Proof Standards: Example



- Scintilla of evidence: $a \vee b$
- Preponderance of evidence: $(a \vee b) \wedge (c \rightarrow b) \wedge \neg d$
- Dialectical validity: $b \wedge \neg c \wedge \neg d$
- Beyond Reasonable Doubt: $b \wedge \neg c \wedge \neg d$
- Beyond Doubt: **f**

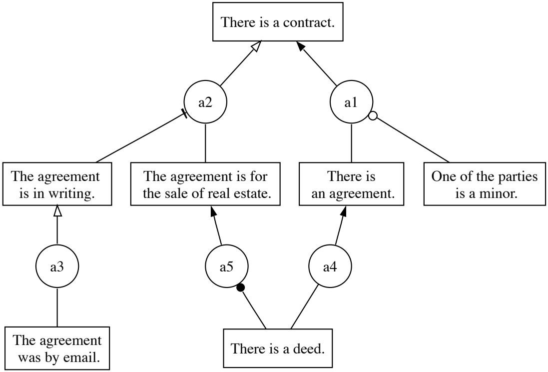
Dynamic Proof Standards



C_s: $(se \wedge (a \vee b)) \vee (pe \wedge (a \vee b) \wedge (c \rightarrow b) \wedge \neg d) \vee (dv \wedge b \wedge \neg c \wedge \neg d)$

4. Case Study: Reconstructing Carneades

- Advanced model of argumentation (Gordon, Prakken, Walton 07)
- Proof standards: scintilla of evid., preponderance of evid., clear and convincing evid., beyond reas. doubt and dial. validity
- Some paraconsistency at work
- Major restriction: no cycles (case where Dung semantics coincide)



Carneades: Basic Definitions

- An **argument** is a tuple $\langle P, E, c\rangle$ with premises P, exceptions E ($P \cap E = \emptyset$) and conclusion c. c and elements of P, E are literals.
- An **argument evaluation structure** (CAES) is a tuple $\mathcal{S} = \langle \mathit{args}, \mathit{ass}, \mathit{weight}, \mathit{standard}\rangle$, where
  - *args* is an acyclic set of arguments,
  - *ass* is a consistent set of literals,
  - *weight* assigns a real number to each argument, and
  - *standard* maps propositions to a proof standard.
- $\langle P, E, c\rangle \in \mathit{args}$ is **applicable** in $\mathcal{S}$ iff
  - $p \in P$ implies $p \in \mathit{ass}$ or [$\overline{p} \notin \mathit{ass}$ and p acceptable in $\mathcal{S}$], and
  - $p \in E$ implies $p \notin \mathit{ass}$ and [$\overline{p} \in \mathit{ass}$ or p is not acceptable in $\mathcal{S}$].

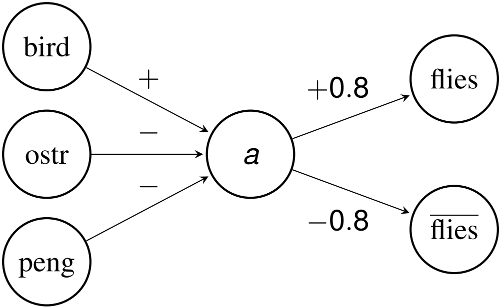
Carneades: Acceptability

A proposition *p* is **acceptable** in $\mathcal{S}$ iff:

- *standard*(*p*) $=$ *se* and there is an applicable argument for *p*,
- *standard*(*p*) $=$ *pe*, *p* satisfies *se*, and max weight assigned to applicable argument pro *p* greater than the max weight of applicable argument con *p*,
- *standard*(*p*) $=$ *ce*, *p* satisfies *pe*, and max weight of applicable pro argument exceeds threshold α, and difference between max weight of applicable pro arguments and max weight of applicable con arguments exceeds threshold β,
- *standard*(*p*) $=$ *bd*, *p* satisfies *ce*, and max weight of the applicable con arguments less than threshold γ,
- *standard*(*p*) $=$ *dv*, and there is an applicable argument pro *p* and no applicable argument con *p*.

Translation: Example

$a = \langle\{bird\}, \{peng, ostr\}, flies\rangle$ with $weight(a) = 0.8$ translates to:

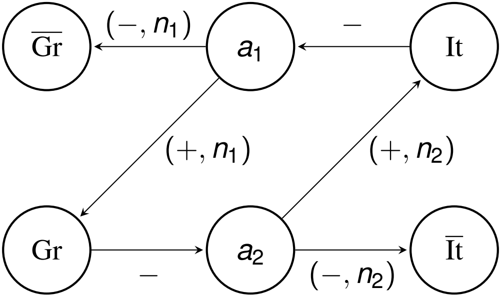


- ADF graph very similar to Carneades graph.
- Can properly define acceptance conditions such that arg node **t** iff argument applicable, prop node **t** iff proposition acceptable.

Why a Reconstruction?

- shows generality of ADFs: Dung and Carneades special cases
- puts Carneades on safe formal ground
- allows us to lift restriction of Carneades to acyclic graphs

$$a_1 = \langle \emptyset, \{It\}, Gr \rangle, a_2 = \langle \emptyset, \{Gr\}, It \rangle.$$



Slide for Henry

- In the meantime van Gijzel/Prakken gave a similar reconstruction based on AFs (via $ASPIC^{+}$).
- ADF approach criticized: "obscuring the direct relation with AFs".
- Who cares? Goal was to provide Carneades with clean semantics and ability to handle cycles.
- Would anybody criticize the use of an electric drill as obscuring the use of a gimlet?
- AFs are a tool, not the holy grail (nor a fetish).

Slide for Henry

- In the meantime van Gijzel/Prakken gave a similar reconstruction based on AFs (via $ASPIC^+$).
- ADF approach criticized: "obscuring the direct relation with AFs".
- Who cares? Goal was to provide Carneades with clean semantics and ability to handle cycles.
- Would anybody criticize the use of an electric drill as obscuring the use of a gimlet?
- AFs are a tool, not the holy grail (nor a fetish).

Slide for Henry

- In the meantime van Gijzel/Prakken gave a similar reconstruction based on AFs (via $ASPIC^+$).
- ADF approach criticized: "obscuring the direct relation with AFs".
- Who cares? Goal was to provide Carneades with clean semantics and ability to handle cycles.
- Would anybody criticize the use of an electric drill as obscuring the use of a gimlet?
- AFs are a tool, not the holy grail (nor a fetish).

Slide for Henry

- In the meantime van Gijzel/Prakken gave a similar reconstruction based on AFs (via $ASPIC^+$).
- ADF approach criticized: "obscuring the direct relation with AFs".
- Who cares? Goal was to provide Carneades with clean semantics and ability to handle cycles.
- Would anybody criticize the use of an electric drill as obscuring the use of a gimlet?
- AFs are a tool, not the holy grail (nor a fetish).

Slide for Henry

- In the meantime van Gijzel/Prakken gave a similar reconstruction based on AFs (via $ASPIC^{+}$).
- ADF approach criticized: "obscuring the direct relation with AFs".
- Who cares? Goal was to provide Carneades with clean semantics and ability to handle cycles.
- Would anybody criticize the use of an electric drill as obscuring the use of a gimlet?
- AFs are a tool, not the holy grail (nor a fetish).

5. Conclusions

- Presented ADFs, an expressive generalization of Dung AFs.
- Expressiveness due to flexible acceptance conditions for nodes.
- Standard Dung semantics were generalized adequately.
- Not meant for direct use by “normal" people; not an abstract locution framework ...
- but a useful semantical tool for graphical argumentation models.
- Complexity analysis (Strass, Wallner, submitted) shows
  - ADFs CANNOT in general be translated to AFs in polynomial time.
  - Same complexity in case of bipolar ADFs.
  - Shows that in this case additional expressiveness comes for free.

5. Conclusions

- Presented ADFs, an expressive generalization of Dung AFs.
- Expressiveness due to flexible acceptance conditions for nodes.
- Standard Dung semantics were generalized adequately.
- Not meant for direct use by “normal" people; not an abstract locution framework ...
- but a useful semantical tool for graphical argumentation models.
- Complexity analysis (Strass, Wallner, submitted) shows
  - ADFs CANNOT in general be translated to AFs in polynomial time.
  - Same complexity in case of bipolar ADFs.
  - Shows that in this case additional expressiveness comes for free.

5. Conclusions

- Presented ADFs, an expressive generalization of Dung AFs.
- Expressiveness due to flexible acceptance conditions for nodes.
- Standard Dung semantics were generalized adequately.
- Not meant for direct use by "normal" people; not an abstract locution framework ...
- but a useful semantical tool for graphical argumentation models.
- Complexity analysis (Strass, Wallner, submitted) shows
  - ADFs CANNOT in general be translated to AFs in polynomial time.
  - Same complexity in case of bipolar ADFs.
  - Shows that in this case additional expressiveness comes for free.

5. Conclusions

- Presented ADFs, an expressive generalization of Dung AFs.
- Expressiveness due to flexible acceptance conditions for nodes.
- Standard Dung semantics were generalized adequately.
- Not meant for direct use by “normal" people; not an abstract locution framework ...
- but a useful semantical tool for graphical argumentation models.
- Complexity analysis (Strass, Wallner, submitted) shows
  - ADFs CANNOT in general be translated to AFs in polynomial time.
  - Same complexity in case of bipolar ADFs.
  - Shows that in this case additional expressiveness comes for free.

5. Conclusions

- Presented ADFs, an expressive generalization of Dung AFs.
- Expressiveness due to flexible acceptance conditions for nodes.
- Standard Dung semantics were generalized adequately.
- Not meant for direct use by "normal" people; not an abstract locution framework ...
- but a useful semantical tool for graphical argumentation models.
- Complexity analysis (Strass, Wallner, submitted) shows
    - ADFs CANNOT in general be translated to AFs in polynomial time.
    - Same complexity in case of bipolar ADFs.
    - Shows that in this case additional expressiveness comes for free.

5. Conclusions

- Presented ADFs, an expressive generalization of Dung AFs.
- Expressiveness due to flexible acceptance conditions for nodes.
- Standard Dung semantics were generalized adequately.
- Not meant for direct use by “normal" people; not an abstract locution framework ...
- but a useful semantical tool for graphical argumentation models.
- Complexity analysis (Strass, Wallner, submitted) shows
  - ADFs CANNOT in general be translated to AFs in polynomial time.
  - Same complexity in case of bipolar ADFs.
  - Shows that in this case additional expressiveness comes for free.

Current Work

- Generalization of further semantics
- Suitability of ADFs as semantical tools in argumentation
- System development
  - DIAMOND: system based on encoding in ASP
  - GRAPPA: graphical user interface based on acceptance patterns

A joke I'm not going to make:
After a lot of Aspic have some Grappa!

Current Work

- Generalization of further semantics
- Suitability of ADFs as semantical tools in argumentation
- System development
  - DIAMOND: system based on encoding in ASP
  - GRAPPA: graphical user interface based on acceptance patterns

A joke I'm not going to make:
After a lot of Aspic have some Grappa!

Thank You and Merry Christmas