Computational Models of Argumentation: A New Perspective on Persisting KR Problems

Gerhard Brewka

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

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

Computational Models of Argumentation: New Keys That Open Old Doors

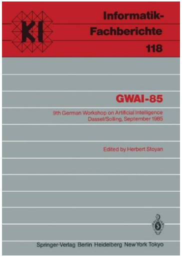
Gerhard Brewka

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

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




A Bit of Nostalgia



My very first conference: 9th KI 1985
(formerly known as GWAI)

KR session:

- John McCarthy: *What is Common Sense and How to Formalize it?*
- Gerhard Brewka: *Über normale Vögel, anwendbare Regeln und einen Default-Beweiser.*
- Peter Schefe: *Zur Rekonstruktion von Wissen in neueren Repräsentationssprachen der Künstlichen Intelligenz.*
- Kai von Luck, Bernhard Nebel, Christof Peltason, Albrecht Schmiedel: *BACK to Consistency and Incompleteness.*

Good Times for AI - Bad Times for Logicians?

- Major breakthroughs: AI in the media more than ever
- Major increase in submissions: IJCAI-17 2500; AAAI-18 3800; IJCAI-18 3400
- Excellent opportunities: digital association Bitkom demands 4 billion Euros + 40 additional professorships for AI in Germany
- Huge investments planned, e.g. in France (1.4 billion Euros)
- AI = deep learning? ... a closer look often reveals intricate combination of learning and "classical" AI methods

A Prominent Example: Google Deep Mind's AlphaGo

- widely perceived as neural network; but (Darwiche/Etzioni):
- "*AlphaGo is not a neural network since its architecture is based on a collection of AI techniques ... in the works for at least fifty years*."
- minimax technique for two-player games, stochastic search, learning from self play, evaluation functions to cut off minimax search trees, reinforcement learning, in addition to neural nets.

Further Witnesses

- Sandholm's Libratus, beating a team of four top pros in poker, powered by new, domain-independent algorithms for
  - computing approximate Nash equilibrium strategies beforehand,
  - endgame solving during play, and
  - fixing its own strategy to play even closer to equilibrium based on what holes the opponents have been able to identify and exploit.
- Dan Roth (IJCAI 2017 John McCarthy Award): success in NLP will be limited unless reasoning gets involved
- Wolfgang Wahlster, KI 2017: "Without good planning techniques the vision of Industrie 4.0 will not come true"
- Andrew Ng: "Anything typical person can do with $\leq$ 1 sec of thought we can probably now or soon automate". And the rest?
- "There is life in AI outside deep learning" R. Lopez de Mantaras

Explainable AI

- To gain user confidence, AI systems must be able to explain their recommendations and actions
- Black box often unsuitable; not understanding brain no excuse
- Explanation: *a reason or justification given for an action or belief* (online dictionary)
- *Reasoning* the main object of study of a logician, so why worry?
- But: AI logicians need to be open to deviate from classical techniques

As a Modern AI Logician

Be prepared to work

- with inference based on some specific (preferred) rather than all models
- with inference relations that are nonmonotonic as what is preferred may change with new information
- with partial rather than complete interpretations as sometimes there is no reasonable way to assign a truth value
- with modern, operator-based techniques to single out the preferred semantic objects, e.g. as fixpoints of these operators
- with multiple semantics, as different situations may require different inferences
- with representations users want, which may look very different from classical logic syntax, e.g. labelled graphs

How Does Argumentation Come In?

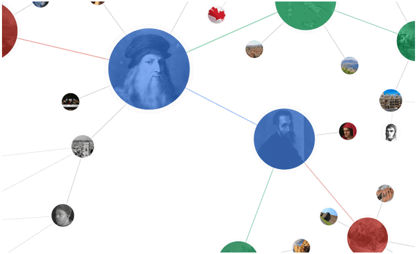
Modgil/Prakken AIJ 2013:

Form of reasoning that *makes explicit the reasons for the conclusions* drawn and *how conflicts between reasons are resolved*.

Provides *natural mechanism to handle inconsistent and uncertain information* and to resolve conflicts of opinion.

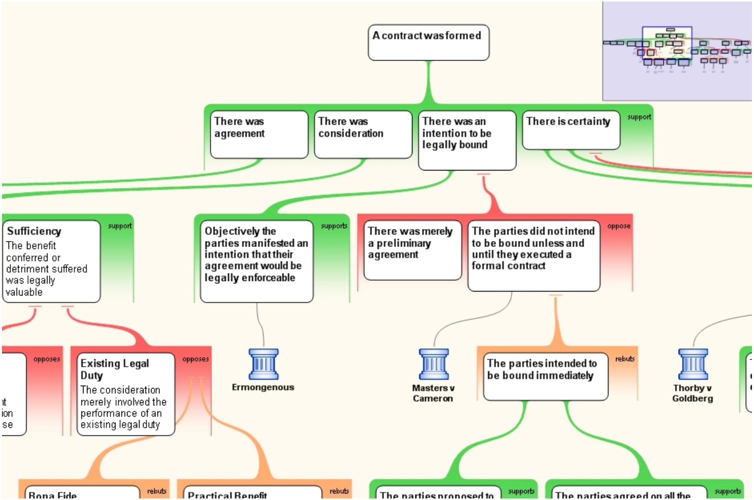
Argumentation approach bridges gap [between logic and human reasoning] by providing *logical formalisms rigid enough to be formally studied* ..., while being *close enough to informal reasoning* ...

Graphs as Knowledge Representation Languages



- Graphical representations extremely popular: semantic nets, rdf graphs, knowledge graphs, argument graphs
- Easy to construct, easy to read by humans, easy to maintain
- Links often represent 2-place predicates, nodes their arguments
- Focus here on acceptance graphs:
  nodes represent statements, positions, arguments ...,
  links relationships between the former, e.g. support, attack ...
- Main goal: identify nodes that can reasonably be accepted

Motivation: Argument Mapping



Argument Graphs

T. Gordon: "*It's graphs what you want to present to the audience ...*"

- Labelled graphs conveniently visualize argumentation scenarios
- Nodes propositions, statements, arguments ... whatever can be accepted or not
- Links represent relationships, labels the type of the relationship
- But what do the links really mean?
- Want to use maps not only for visualization, but for *evaluation*
- Requires a framework for specifying semantics!

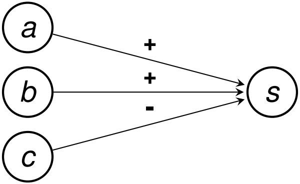
Argument Graphs

T. Gordon: "*It's graphs what you want to present to the audience ...*"

- Labelled graphs conveniently visualize argumentation scenarios
- Nodes propositions, statements, arguments ... whatever can be accepted or not
- Links represent relationships, labels the type of the relationship

- But what do the links really mean?
- Want to use maps not only for visualization, but for *evaluation*
- Requires a framework for specifying semantics!

Another Example



Should s be accepted? Various options, e.g.

- no negative and all positive links are active, or
- no negative and at least one positive link is active, or
- more positive than negative links are active.

Bottom line: need an acceptance condition for each of the nodes.

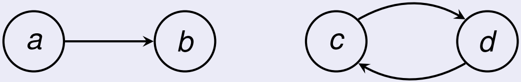
Outline

1. Setting the Stage (done)
2. A Precedent: Dung's Argumentation Frameworks
3. A Step Forward: Abstract Dialectical Frameworks
4. From ADFs to GRaph-based Argument Processing (GRAPPA)
5. Conclusions

2. A Precedent: Dung Frameworks

Abstract Argumentation Frameworks (AFs)

- immensely popular in the argumentation community
- syntactically: directed graphs



- conceptually: nodes arguments, edges attacks between arguments
- semantically: *extensions* are sets of “acceptable” arguments
- a simple special case of labelled graphs:
  single label (left implicit), fixed acceptance condition

AF Semantics

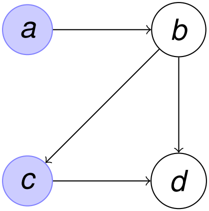
$F = (A, R)$ an argumentation framework, $S \subseteq A$.

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

Main goal:

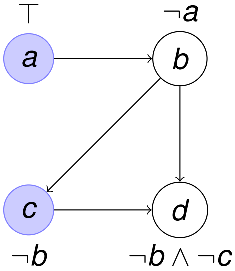
Generalize what Dung did for simple AFs to arbitrary labelled graphs.

3. ADFs: Basic Idea



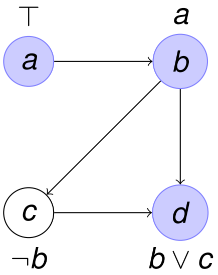
An Argumentation Framework

Basic Idea



An Argumentation Framework
with explicit acceptance conditions

Basic Idea



A Dialectical Framework
with flexible acceptance conditions

Background on ADFs

- Directed graph, each node has explicit acceptance condition expressed as propositional formula.
- ADFs properly generalize AFs under major semantics.
- Semantics based on operator Γ_D over partial (3-valued) interpretations (here represented as consistent sets of literals).
- Takes interpretation v and produces a new (revised) one v'.
- $v' = \Gamma_D(v)$ makes a node s
  - **t** iff acceptance condition true under all 2-valued completions of v,
  - **f** iff acceptance condition false under all 2-valued completions of v,
  - undefined otherwise.
- Operator thus checks what can be justified based on v.

Background on ADFs

- Directed graph, each node has explicit acceptance condition expressed as propositional formula.
- ADFs properly generalize AFs under major semantics.
- Semantics based on operator Γ_D over partial (3-valued) interpretations (here represented as consistent sets of literals).
- Takes interpretation v and produces a new (revised) one v'.
- $v' = \Gamma_D(v)$ makes a node s
  - **t** iff acceptance condition true under all 2-valued completions of v,
  - **f** iff acceptance condition false under all 2-valued completions of v,
  - undefined otherwise.
- Operator thus checks what can be justified based on v.

Background on ADFs

- Directed graph, each node has explicit acceptance condition expressed as propositional formula.
- ADFs properly generalize AFs under major semantics.
- Semantics based on operator Γ_D over partial (3-valued) interpretations (here represented as consistent sets of literals).
- Takes interpretation v and produces a new (revised) one v'.
- $v' = \Gamma_D(v)$ makes a node s
  - **t** iff acceptance condition true under all 2-valued completions of v,
  - **f** iff acceptance condition false under all 2-valued completions of v,
  - undefined otherwise.
- Operator thus checks what can be justified based on v.

Semantics via Fixed Points

An interpretation v of ADF D is

- a *model* of D iff v is total and $\Gamma_D(v) = v$.
  Intuition: statement is **t** iff its acceptance condition says so.
- *grounded* for D iff it is the least fixpoint of Γ_D.
  Intuition: collects information beyond doubt.
- *admissible* for D iff $v \subseteq \Gamma_D(v)$
  Intuition: does not contain unjustifiable information
- *preferred* for D iff it is $\subseteq$-maximal admissible for D
  Intuition: want maximal information content.
- *complete* for D iff $v = \Gamma_D(v)$.
  Intuition: contains exactly the justifiable information.

Stable semantics: reduct-based check as in logic programming.

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.

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).
- various results regarding realizability, complexity, ...
  - 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.

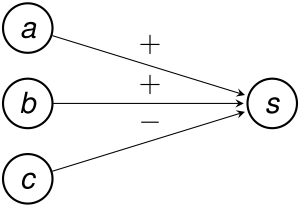
Results

- 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).
- various results regarding realizability, complexity, ...
  - 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.

Using ADFs: Earlier Example



- Positive and negative links
- 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: $(\neg c \wedge (a \vee b)) \vee (a \wedge b)$
- Acceptance condition defined individually for each node

4. From ADFs to GRAPPA

- Compiling argumentation graphs to ADFs tedious in general
- Can we define ADF-like semantics directly for any labelled graph?
- YES, requires
  - to define acceptance conditions in terms of labels of active links
  - and adequate modification of characteristic operator
- The rest basically falls into place
- Main advantages:
  - Closer to graphical models people use
  - Same intuition has same representation for all nodes, e.g. $\#+ > \#-$ rather than node specific prop. formula

From ADFs to GRAPPA, ctd.

- Acceptance conditions based on multisets of labels of active links
- New characteristic operator taking these into account

An acceptance function over labels L is a function $c : (L \to \mathbb{N}) \to \{t, f\}$.

A labelled argument graph (LAG) is a tuple $G = (S, E, L, \lambda, \alpha)$ where

- S is a set of nodes (statements),
- E is a set of edges (dependencies),
- L is a set of labels,
- $\lambda : E \to L$ assigns labels to edges,
- $\alpha : S \to F^L$ assigns L-acceptance-functions to nodes.

The Characteristic Operator Γ_G

- Operator revises partial interpretation v, produces new one v'.
- Checks which truth values of nodes in S can be justified by v.
- Done by considering all possible completions of v and their induced multisets of active labels:
  - if acceptance function of s yields t under all such multisets, then v' assigns **t** to s.
  - if acceptance function of s yields f under all such multisets, then v' assigns **f** to s.
  - otherwise the value remains open.
- Basically the same as for ADFs, except for acceptance functions involved.

Semantics

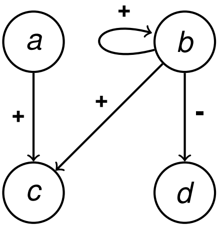
As for ADFs

Let $G = (S, E, L, \lambda, \alpha)$ be an LAG, v a partial interpretation of S.

- v is a model of G iff v is total and $v = \Gamma_G(v)$,
- v is grounded in G iff v is the least fixed point of Γ_G,
- v is admissible in G iff $v \subseteq \Gamma_G(v)$,
- v is preferred in G iff v is $\subseteq$-maximal admissible in G,
- v is complete in G iff $v = \Gamma_G(v)$.

Stable models: no self-justifying cycles. Checked by LP-style reduct.

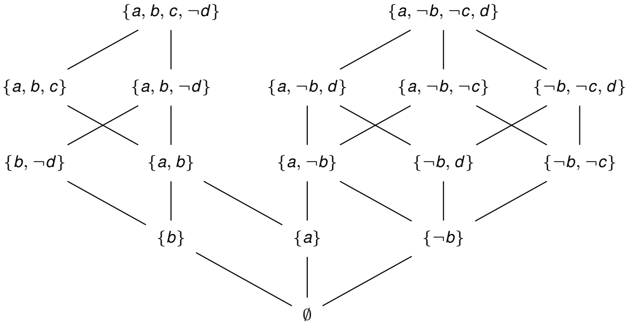
Example



Acceptance condition for all nodes: all positive links active, no negative link active.

Example, ctd.

16 admissible interpretations:



Models: $\{a, b, c, \neg d\}$, $\{a, \neg b, \neg c, d\}$.
Grounded: $\{a\}$.
Preferred: $\{a, b, c, \neg d\}$, $\{a, \neg b, \neg c, d\}$.
Complete: $\{a, b, c, \neg d\}$, $\{a, \neg b, \neg c, d\}$, $\{a\}$.

The GRAPPA Pattern Language

- How to express acceptance conditions?
- Developed pattern language for this purpose.
- Can refer to number of total and active labels of specific types; to minimal/maximal elements; simple arithmetics and relations.
- Won't define language completely, illustrate it with examples.
  - Let $L = \{\mathbf{++}, \mathbf{+}, \mathbf{-}, \mathbf{--}\}$
  - Assume node accepted if support stronger than attack, measure strength by counting respective links; multiply strong support/attack with a factor of 2.
  - Describe this using pattern:

$$2(\#\mathbf{++}) + (\#\mathbf{+}) - 2(\#\mathbf{--}) - (\#\mathbf{-}) > 0.$$

Use Cases

Dung AFs

Single label **-** left implicit. Single pattern for all nodes:

- no negative active link: $(\#\textbf{-}) = 0$

ADFs

ADF acceptance conditions propositional formulas. GRAPPA: label each link with its source node. Pattern:

- replace each occurrence of atom a in ADF acceptance condition by the basic pattern $\#a = 1$.

Use Cases, ctd.

Bipolar argument graphs

Labels for support (**+**) and attack (**-**). Possible acceptance conditions:

- all positive, no negative link active: $(\#_t\mathbf{+}) - (\#\mathbf{+}) = 0 \wedge (\#\mathbf{-}) = 0$,
- at least one positive, no negative active link: $(\#\mathbf{+}) > 0 \wedge (\#\mathbf{-}) = 0$,
- more positive than negative active links: $(\#\mathbf{+}) - (\#\mathbf{-}) > 0$.

Weighted argument graphs

Labels positive or negative numbers. Various possible patterns:

- the sum of weights of active links is greater than 0: $sum > 0$.
- the highest active support is stronger than the strongest (lowest) attack: $max + min > 0$
- the difference among strongest active support and the strongest active attack is above some threshold b: $max + min > b$.

Use Cases, ctd.

Proof standards (Farley and Freeman)

Framework for expressing proof standards based on 4 types of arguments: valid, strong, credible and weak.
Need 8 labels $v, s, c, w, -v, -s, -c, -w$. Patterns of some of the proof standards:

- scintilla of evidence: $\#\{v, s, c, w\} > 0$
- dialectical validity: $\#\{v, s, c\} > 0$, $\#\{-v, -s, -c, -w\} = 0$
- beyond reasonable doubt: $\#\{v, s\} > 0$, $\#\{-v, -s, -c, -w\} = 0$
- beyond doubt: $\#v > 0$, $\#\{-v, -s, -c, -w\} = 0$

Use Cases, ctd.

Carneades (Gordon, Prakken, Walton)

Argument graphs with 2 types of nodes. Pattern for argument nodes:

- $(\#_t\textbf{+}) - (\#\textbf{+}) = 0 \wedge (\#\textbf{-}) = 0$,

Patterns for proposition nodes (α, β and γ numerical parameters):

- scintilla of evidence: $\mathit{max} > 0$
- preponderance of evidence: $\mathit{max} + \mathit{min} > 0$
- clear and convincing evidence: $\mathit{max} > \alpha \wedge \mathit{max} + \mathit{min} > \beta$
- beyond reasonable doubt: $\mathit{max} > \alpha \wedge \mathit{max} + \mathit{min} > \beta \wedge -\mathit{min} < \gamma$
- dialectical validity: $\mathit{max} > 0 \wedge \mathit{min} > 0$

5. Conclusions

- Presented a semantical framework for labelled argument graphs
  - based on ideas from ADFs, yet domain of acceptance conditions multisets of labels,
  - pattern language for expressing acceptance conditions,
  - demonstrated generality by reconstructing various systems,
  - implementations by compilation to ADFs.
- What does GRAPPA buy you?
  - pick your favourite graphical representation of argumentation scenarios
  - turn it into a well-founded formalism with full range of Dung semantics
  - by specifying patterns in a convenient language.

5. Conclusions

- Presented a semantical framework for labelled argument graphs
  - based on ideas from ADFs, yet domain of acceptance conditions multisets of labels,
  - pattern language for expressing acceptance conditions,
  - demonstrated generality by reconstructing various systems,
  - implementations by compilation to ADFs.
- What does GRAPPA buy you?
  - pick your favourite graphical representation of argumentation scenarios
  - turn it into a well-founded formalism with full range of Dung semantics
  - by specifying patterns in a convenient language.

Current Work

- System development, based on DIAMOND, our ADF solver
- Mobile App *ArgueApply*, LPNMR-17 best system description
- Extension to weighted case: partial multi-valued interpretations
- Application to interesting argument graphs