Formalizing Practical Argumentation Lecture 2: Logics for Defeasible Argumentation I. Semantics

Henry Prakken
Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands

Bahia Blanca, June 1998

Introduction

- Practical reasoning is defeasible: jumping to conclusions which hold ordinarily, presumptively, unless there is evidence to the contrary, . . .
- Logics for defeasible reasoning lack monotonicity: it is not the case that
  $X \subseteq Y \longrightarrow Cn(X) \subseteq Cn(Y)$
- Systems for defeasible argumentation: constructing and comparing conflicting arguments
  - Arguments *constructed* monotonically
  - Nonmonotonicity arises from possibility of defeat by counterarguments

Relevant names:

Pollock, Loui, Simari & Loui, Lin & Shoham, Vreeswijk, Dung & Kowalski & Toni & Bondarenko, Verheij, Nute, Prakken & Sartor ...

Popular domain of application: the law

Loui & Norman, Verheij, Prakken & Sartor, Gordon, Nitta ...

Topic of today: Argumentation systems

- General ideas
- Semantics: status assignments
- Some semantics of Dung (1995)
- Resource-bounded reasoning
- Reasoning about strength of arguments

General Ideas

An argumentation system has five elements:

- Logical language
- Arguments (monotonic derivation!)
- Conflict between arguments
- Comparing conflicting arguments (relation of 'defeat')
- Status assignment to arguments

Logic: Assesses validity of arguments

$$\frac{Premises}{Conclusion}$$

Premises	Evaluation standards	Assessment

Arg1 Arg2 Justified Args

Arg3 Overruled Args

Arg4 Arg5 Defensible Args

Conflict between arguments

- Attack on conclusion (rebutting)
- Attack on assumption, or on inference step (undercutting)

Rebutting:

Nixon was a pacifist because he was a quaker
Nixon was no pacifist because he was a republican

Undercutting (of assumption):

Assuming that the buyer is not a minor, a sale is valid
The buyer is a minor

Undercutting (of inference step):

All observed ravens were black, so the next raven will be black
Yesterday I saw a white raven

Comparing conflicting arguments

- On the basis of some 'input'-ordering of premisses or arguments (specificity, reliability, authority ...)
- Result: a binary relation of 'defeat' among arguments ($\rightarrow$)

Assigning a status to arguments

- 'Defeat' is insufficient: suppose

  $A =$ 'Tweety can fly because Tweety is a bird'
  $B =$ 'Tweety cannot fly because Tweety is a penguin'
  $C =$ 'The observation that Tweety is a penguin, is unreliable'

- $A \leftarrow B \leftarrow C$.

  A is 'reinstated' by C.

Semantics for argumentation systems

- INPUT: a pair $\langle Args, Defeat \rangle$
  - $Args$ is a set of arguments
  - $Defeat$ is een binaire relatie op $Args$
- OUTPUT: an assignment of the status 'in' or 'out' to all arguments in $Args$

Or: find an appropriate labelling of the 'defeat graph'.

Conditions on labellings: **(I)** Every argument is either 'in' or 'out'.

1. an argument is 'in' if all arguments defeating it are 'out'.
2. an argument is 'out' if it is defeated by an argument dat is 'in'.

Works OK with:

A = 'Tweety can fly because Tweety is a bird'
B = 'Tweety cannot fly because Tweety is a penguin'
C = 'The observation that Tweety is a penguin, is unreliable'

$A \leftarrow B \leftarrow C$

Conditions on labellings: **(I)** Every argument is either 'in' or 'out'.

1. an argument is 'in' if all arguments defeating it are 'out'.
2. an argument is 'out' if it is defeated by an argument dat is 'in'.

But goes wrong with:

A = 'Nixon was a pacifist because he was a quaker'
B = 'Nixon was not a pacifist because he was a republican

$A \leftrightarrow B$

Two possible solutions:

1. Change the conditions in such a way that always a unique status assignment results
2. Make use of multiple status assignments

	Unique s.a.	Multiple s.a.
$A \leftarrow B \leftarrow C$:	s.a. 1: $\{A, C\}$	s.a. 1: $\{A, C\}$
$A \leftrightarrow B$:	s.a. 1: $\emptyset$	s.a. 1: $\{A\}$ s.a. 2: $\{B\}$

Status definition with unique s.a.:

- *Justified argument:* 'In'
- *Overruled argument:* 'Out' and attacked by an argument that is 'in'
- *Defensible argument:* 'Out' but not attacked by an argument that is 'in'.

Status definition with multiple s.a.:

- *Justified argument:* 'In' in all status assignments
- *Overruled argument:* 'Out' in all status assignments
- *Defensible argument:* 'In' in some but not all status assignments

Makes a difference with:

- 'Floating argument statuses': $A \leftrightarrow B \rightarrow C \rightarrow D$

- 'Floating conclusions':

A^-:	Brygt is Dutch because he was born in Holland
B^-:	Brygt is Norwegian because he has Norwegian parents
A:	Brygt likes ice skating because he is Dutch
B:	Brygt likes ice skating because he is Norwegian

Some semantics of Dung (1995)

Basic notions:

- An argument A is *acceptable* with respect to a set S of arguments iff each argument defeating A is defeated by an argument in S.

- A conflict-free set of arguments S is *admissible* iff each argument in S is acceptable with respect to S.

Unique status assignments: 'grounded extensions'

- $F : Pow(Args) \longrightarrow Pow(Args)$

- $F(S) = \{A \in Args | A \text{ is acceptable with respect to } S\}$

The *grounded extension* of a pair $\langle Args, Defeat \rangle$ is the least fixed point of F.

Constructive part of grounded extension

Consider the following sequence of arguments.

- $F^0 = \emptyset$
- $F^{i+1} = \{A \in Args \mid A$ is acceptable with respect to $F^i\}$.

The following observations hold.

1. $\cup_{i=0}^{\infty}(F^i) \subseteq$ grounded extension.
2. If each argument is defeated by at most a finite number of arguments, then $\cup_{i=0}^{\infty}(F^i) =$ grounded extension.

Multiple status assignments

- A *stable extension* of a pair $\langle Args, Defeat\rangle$ is a conflict-free set S such that all $A \in Args$ outside S are defeated by a member of S

  Problem: pairs $\langle Args, Defeat\rangle$ with odd defeat cycles have no stable extensions.

  Solution: make assignments partial, by omitting odd defeat cycles
- A *preferred extension* of a pair $\langle Args, Defeat\rangle$ is a $\subset$-maximal admissible subset of $Args$

Resource-bounded reasoning

(Pollock, Loui, Vreeswijk)

- Suppose: arguments are computed on the basis of premises.
- Two forms of defeasibility:
  - If more premises, then more arguments
  - If more computation, then more arguments
- Semantic idea: status assignments relative to computed arguments only
- Proof-theoretic idea: give a reasoner 'resources'

Reasoning about the strength of arguments

- In law priorities are
  - domain dependent;
  - subject to debate.

r_1: $Protected(Villa) \Rightarrow \neg Modifiable(Villa)$

r_2: $Needs\_restructuring(Villa) \Rightarrow Modifiable(Villa)$

r_3: $Town\_planning\_rule(r_2) \wedge Artistic\_buildings\_rule(r_1) \Rightarrow r_2 \prec r_1$

r_4: $Later(r_2, r_1) \Rightarrow r_1 \prec r_2$

- Language extended with a special predicate symbol $\prec$;
- $<$ is now determined by the arguments with conclusions on $\prec$

- *A strictly defeats* B iff A defeats B and B does not defeat A.
- *Notation:* For any set S of arguments
  - $<_S = \{r < r' \mid r \prec r'$ is a conclusion of some $A \in S\}$
  - A (strictly) S-defeats B iff, assuming the ordering $<_S$ on Δ, A (strictly) defeats B.
- Add the following axioms to the 'strict' rules:

$$\begin{array}{ll} \textit{transitivity}: & \forall x, y, z\ (x \prec y \wedge y \prec z \supset x \prec z) \\ \textit{asymmetry}: & \forall x, y\ (x \prec y \supset \neg\ y \prec x) \end{array}$$

- An argument A is *acceptable* with respect to a set S of arguments iff all arguments S–defeating A are strictly S–defeated by some argument in S.

Example

$$\begin{array}{ll} r_1: & Protected(x) \Rightarrow \neg Modifiable(x) \\ r_2: & Needs\_restructuring(x) \Rightarrow Modifiable(x) \\ r_3: & Town\_planning\_rule(y) \wedge Artistic\_buildings\_rule(x) \Rightarrow y \prec x \\ r_4: & Later(x, y) \Rightarrow x \prec y \\ r_5: & \Rightarrow Artistic\_buildings\_rule(r_1) \\ r_6: & \Rightarrow Town\_planning\_rule(r_2) \\ r_7: & \Rightarrow Later(r_1, r_2) \\ r_8: & \Rightarrow Protected(Villa_0) \\ r_9: & \Rightarrow Needs\_restructuring(Villa_0) \\ r_{10}: & \Rightarrow r_4 \prec r_3 \end{array}$$

$$
\begin{aligned}
F^0 &= \emptyset \\
<_0 &= \emptyset \\
\\
F^1 &= F^0 \cup \{(r_5) - (r_{10})\} \\
<_1 &= \{r_4 < r_3\} \\
\\
F^2 &= F^1 \cup \{(r_4, r_5, r_3)\} \\
<_2 &= \{r_4 < r_3, r_2 < r_1\} \\
\\
F^3 &= F^2 \cup \{(r_6, r_1)\} \\
<_3 &= <_2 \\
\\
F^4 &= F^3
\end{aligned}
$$