An Overview of Labelling-Based Justification Status

Martin Caminada
Yining Wu

Preliminaries

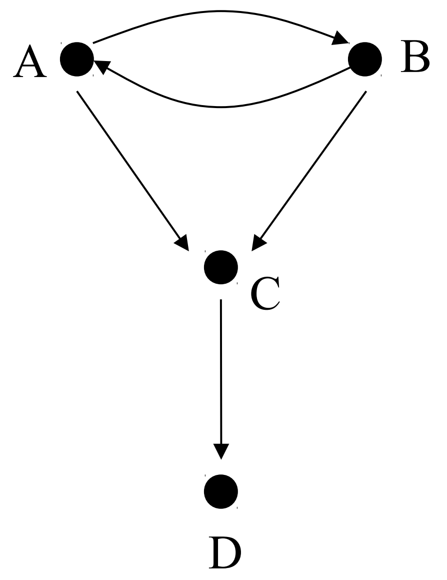
- *Argumentation framework*: graph (Ar, att) in which
  - the nodes (Ar) represent a given set of *arguments,*
  - the arrows (att) represent the *attack* relation.
- A *labelling* is a function *L*ab: Ar → {in, out, undec}.
- A *complete labelling* is a labelling s.t. For each argument A,
  - A is labelled in iff all its attackers are labelling out.
  - A is labelled out iff it has an attacker that is labelled in.

eg. in a gun fight

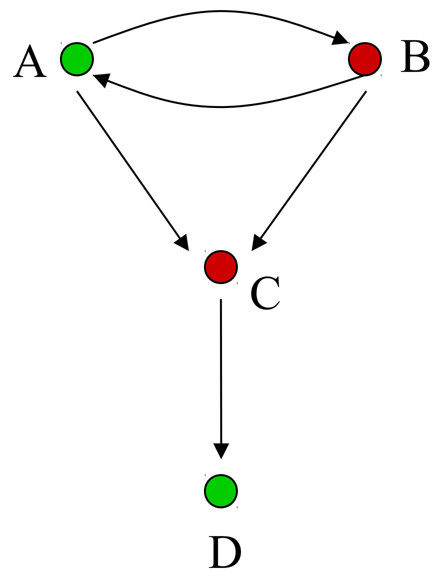
You survive iff al your attackers are killed.

You get killed iff at least one attacker remains alive.

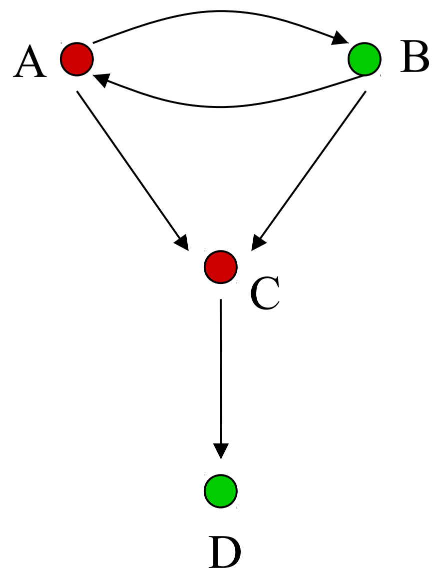
An Example



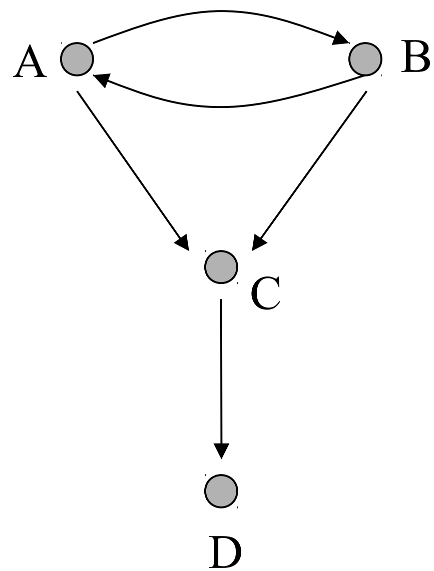
An Example



An Example

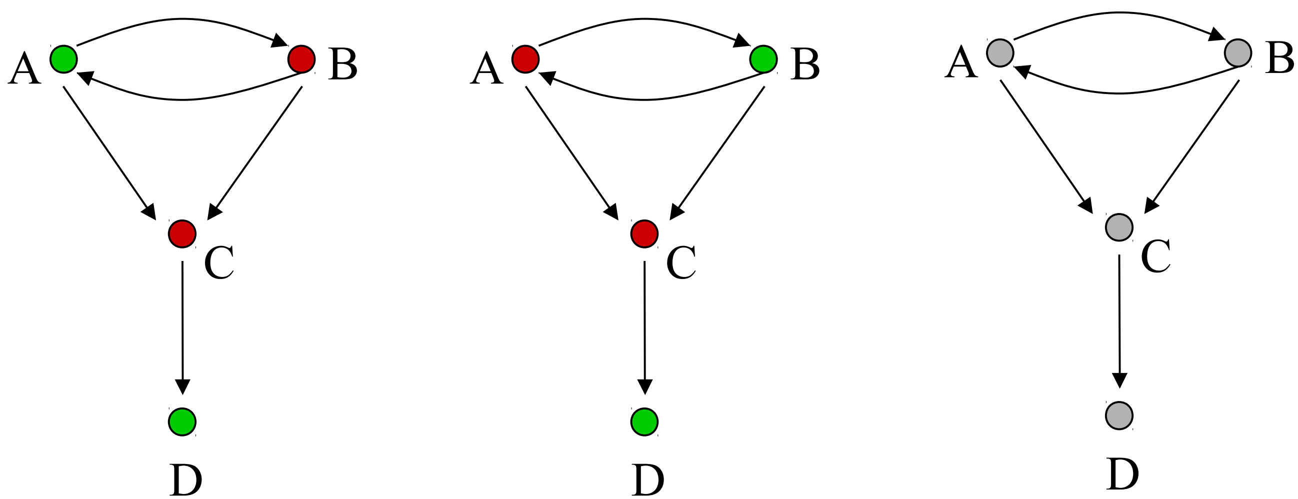


An Example



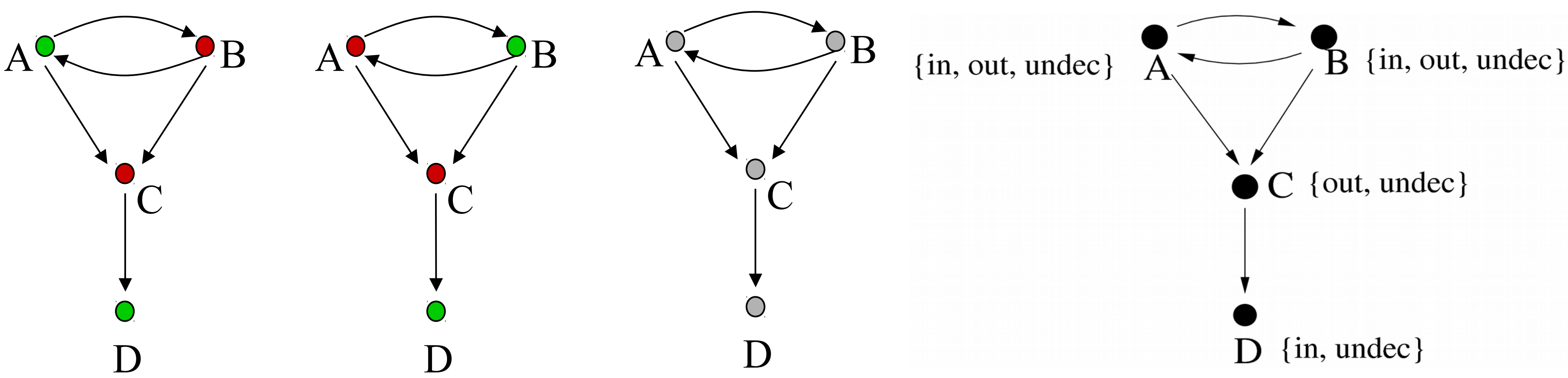
Justification Status

Justification status: the set of labels that can be assigned to an argument by the complete labellings.



Justification Status

Justification status: the set of labels that can be assigned to an argument by the complete labellings.



Justification status

$\mathcal{JS}(A) =$

$\{\texttt{in}\}$ iff A is in every complete extension.

$\{\texttt{out}\}$ iff A is attacked by every complete extension.

$\{\texttt{undec}\}$ iff A is not in any complete extension and
A is not attacked by any complete extension.

$\{\texttt{in}, \texttt{undec}\}$ iff A is not in a complete extension and
A is in a complete extension and
no complete extension attacks A.

$\{\texttt{out}, \texttt{undec}\}$ iff A is not in any complete extension and
A is attacked by a complete extension and
there is a complete extension does not attack A.

$\{\texttt{in}, \texttt{out}, \texttt{undec}\}$ iff A is in a complete extension and
A is attacked by a complete extension.

$\{\texttt{in}, \texttt{out}\}$: impossible

$\emptyset$: impossible

Justification status

$\mathcal{JS}(A) =$

$\{\mathtt{in}\}$ iff A is in the grounded extension.

$\{\mathtt{out}\}$ iff A is attacked by the grounded extension.

$\{\mathtt{undec}\}$ iff A is not in any admissible extension and
A is not attacked by any admissible extension.

$\{\mathtt{in}, \mathtt{undec}\}$ iff A is not in the grounded extension and
A is in an admissible extension and
A is not attacked by any admissible extension.

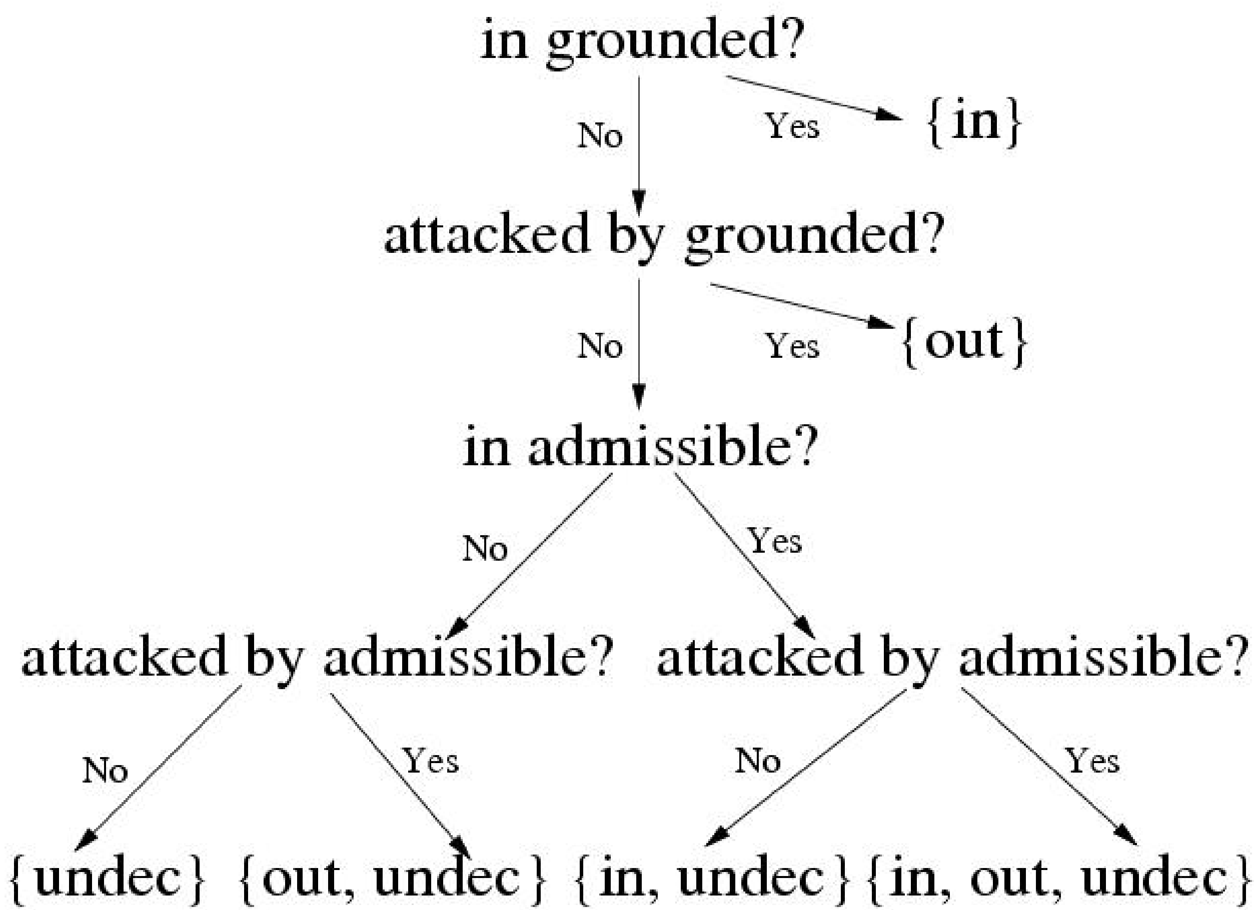
$\{\mathtt{out}, \mathtt{undec}\}$ iff A is not in any admissible extension and
A is attacked by an admissible extension and
A is not attacked by the grounded extension.

$\{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ iff A is in an admissible extension and
A is attacked by an admissible extension.

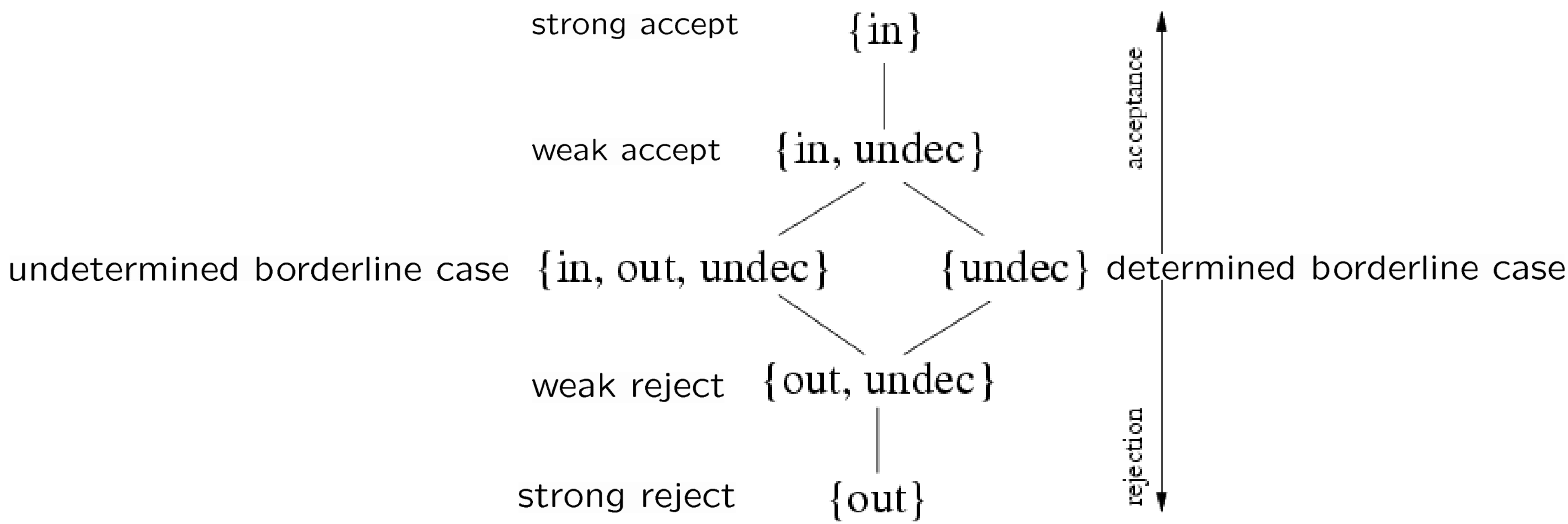
$\{\mathtt{in}, \mathtt{out}\}$: impossible

$\emptyset$: impossible

Justification status



Degrees of Justification status



Justification status

Proposition

- A is in the grounded extension iff it is strongly accepted.
- if A is in every preferred extension then A is strongly or weakly accepted.
- A is in at least one preferred extension iff A is strongly accepted , weakly accepted, or undetermined borderline.
- if A is strongly accepted then A is in every semi-stable extension. if A is weakly accepted then A is in at least one semi-stable extension.
- A is in an ideal set iff A is member of an admissible set consisting only of strongly or weakly accepted arguments.

Justification Status of Conclusions

- each argument A has a conclusion Conc(A) $\in \mathcal{L}$
- a conclusion labelling is a function
  ConcLab: $\mathcal{L} \rightarrow$ {in, out, undec}
- Given an argument labelling ArgLab, we define
  the associated conclusion labelling ConcLab s.t.
  ConcLab(c) is the label of the “best” argument for c
  (or out, if no argument for c exists)

Justification Status of Conclusions

- each argument A has a conclusion Conc(A) $\in \mathcal{L}$
- a conclusion labelling is a function
  ConcLab: $\mathcal{L} \rightarrow$ {in, out, undec}
- Given an argument labelling ArgLab, we define
  the associated conclusion labelling ConcLab s.t.
  ConcLab(c) = max({ArgLab(A) | Conc(A)=c} $\cup$ {out})

Justification Status of Conclusions

- each argument A has a conclusion Conc(A) $\in \mathcal{L}$
- a conclusion labelling is a function
  ConcLab: $\mathcal{L} \rightarrow$ {in, out, undec}
- Given a complete argument labelling ArgLab, we define the associated complete conclusion labelling ConcLab s.t.
  ConcLab(c) = max({ArgLab(A) | Conc(A)=c} $\cup$ {out})
- JS(c) = {ConcLab(c) | ConcLab is a complete conclusion labelling}

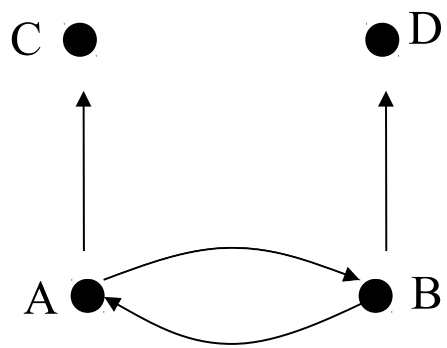
Example:Dealing with Floating Conclusions

- Brygt Rykkje is Dutch because he was born in Holland
- Brygt Rykkje is Norwegian because he has a Norwegian name
- Brygt Rykkje likes ice skating
  because he is Norwegian
- Brygt Rykkje likes ice skating
  because he is Dutch

Example:Dealing with Floating Conclusions

- John says the suspect killed the victim by stabbing him
- Bob says the suspect killed the victim by shooting him
- The suspect killed the victim,
  because Bob says the suspect killed the victim by shooting him
- The suspect killed the victim,
  because John says the suspect killed the victim by stabbing him

Example:Dealing with Floating Conclusions



Conc(A) = a
Conc(B) = b
Conc(C) = e
Conc(D) = e

$ArgLab_1$ = ({A, D}, {B,C}, ∅)
$ConcLab_1$ = ({a, e}, {b}, ∅)

$ArgLab_2$ = ({B,C}, {A,D}, ∅)
$ConcLab_2$ = ({b, e}, {a}, ∅)

$ArgLab_3$ = (∅, ∅, {A,B,C,D})
$ConcLab_3$ = (∅, ∅, {a,b,e})

JS(e) = {in, undec} *(weak accept)*

Labelling-Based JS:

- provides *levels* of justification based on *standard* AFs (so no probabilities or other numerical add-ons)
- provides a more *refined* status
  than the usual extension based approached
  (e.g. grounded or credulous preferred)
- can easily be *computed* (based on existing proof procedures for grounded and preferred)
- can be applied to *arguments* as well as to *conclusions* (floating conclusions become weakly accepted)

Literature

- Yining Wu and Martin Caminada
  A Labelling-Based Justification Status of Arguments
  Studies in Logic 3(4):12-29 (2010)

- Wolfgang Dvořák
  On the Complexity of Computing
  the Justification Status of an Argument
  TAFA post proceedings, pages 32-49 (2012)