UNIVERSITÀ DEGLI STUDI
DI PERUGIA

G S
S I
GRAN SASSO
SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

A Concurrent Language for Argumentation: Preliminary Notes

Stefano Bistarelli and Carlo Taticchi

DIP2020

Overview

- General notions on Argumentation
- Four-state labelling
- Concurrent Argumentation Language (CA)
  - syntax + operational semantics
- Expansion, contraction and revision

Extension-based Semantics for AFs[1]

Conflict-Free

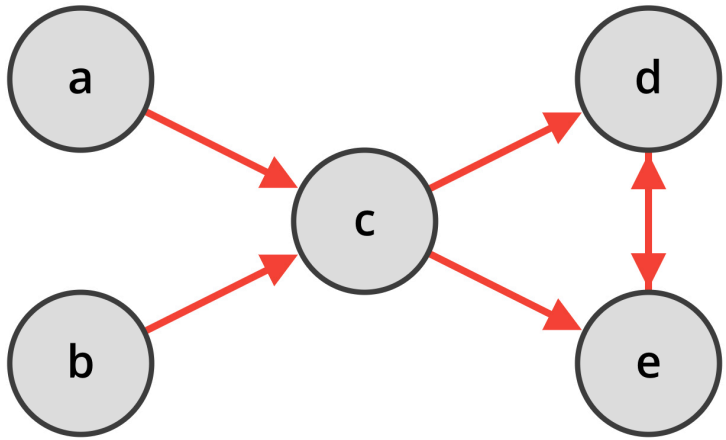
{a}, {c}, {a,d}, ...

Admissible

{a}, {a, d}, {a, b, d}, ...

Complete

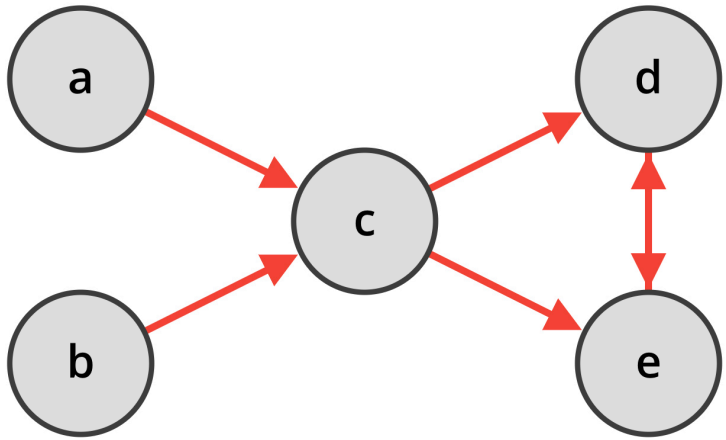
{a, b}, {a, b, d}, ...



[1]Phan Minh Dung. **On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.** Artificial Intelligence, 77(2):321–357.

Credulous/Skeptical Acceptability

- Two approaches to check the acceptance status:
  - (Credulous) Is argument **a** accepted in some extension?
  - (Skeptical) Is argument **a** accepted in all extensions?



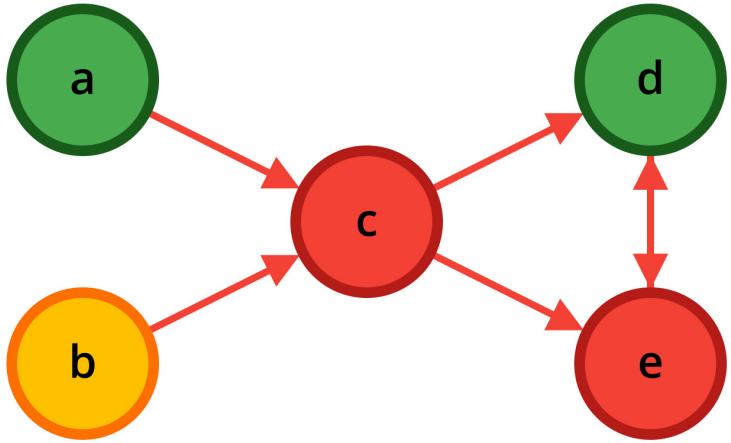
Complete: {{a, b}, {a, b, d}, {a, b, e}}

Reinstatement Labelling[2]

IN if it is attacked only by **OUT** arguments

OUT if it is attacked by at least an **IN** argument

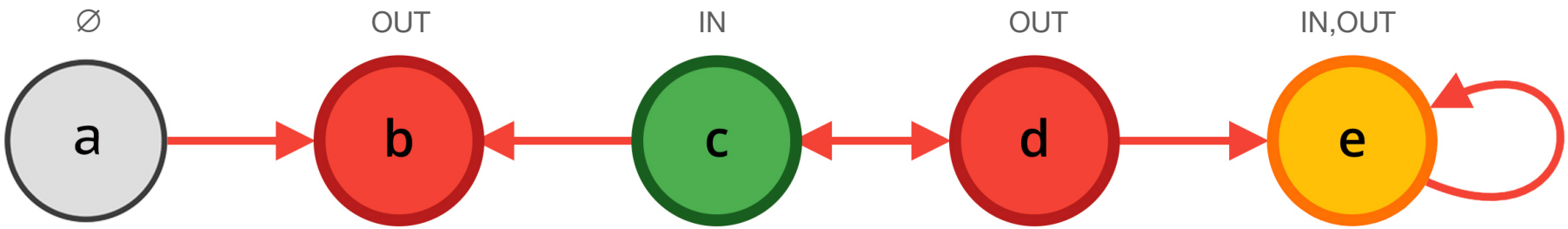
UNDEC otherwise



[2]Martin Caminada. **On the Issue of Reinstatement in Argumentation.** JELIA 2006: 111-123.

Four-State Labelling[3]

- Introduces **don't know** and **don't care** labels
- Each argument has a label $l \subseteq \{in, out\}$
- Identifies extensions of the various semantics

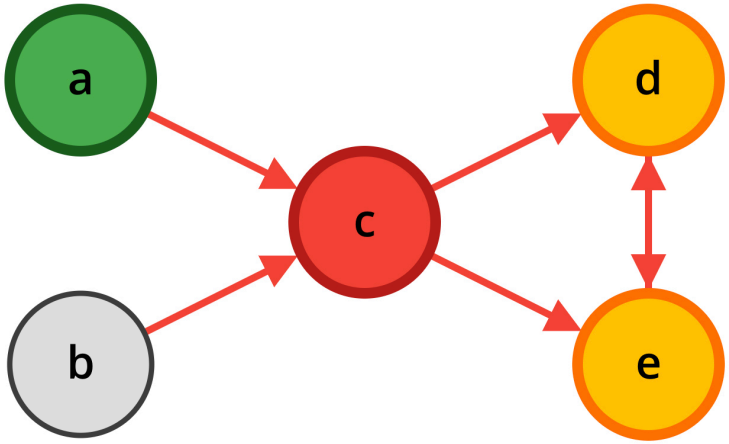


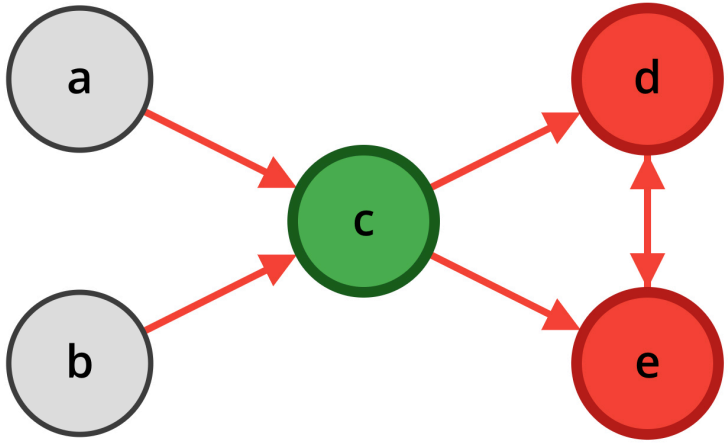
[3]Hadassa Jakobovits and Dirk Vermeir. **Robust Semantics for Argumentation Frameworks**. J. Log. Comput. 9(2): 215-261 (1999)

Labelling-Based Semantics

CF

a is **IN** ⇒ a is not attacked by any **IN**
a is **OUT** ⇒ a is attacked by some **IN**

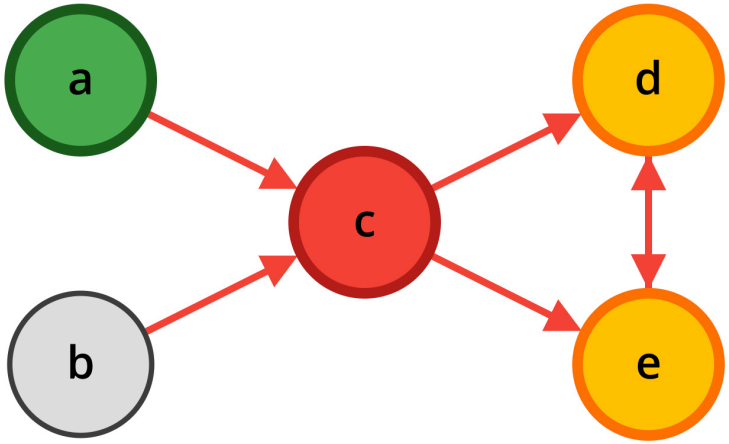


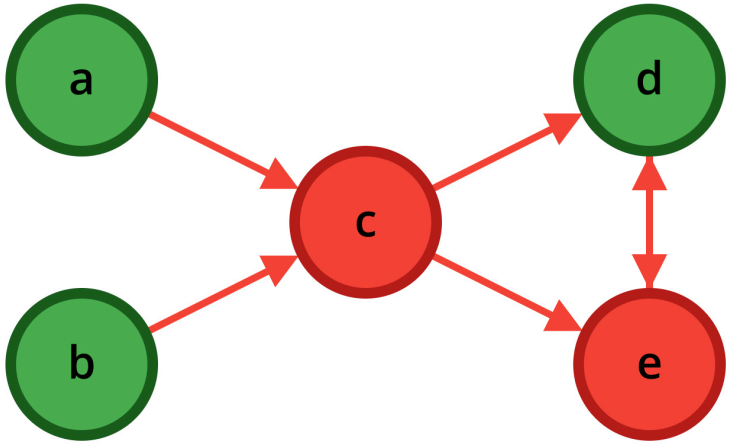


Labelling-Based Semantics

ADM

a is **IN** ⇒ a is only attacked by **OUT**
a is **OUT** ⇒ a is attacked by some **IN**



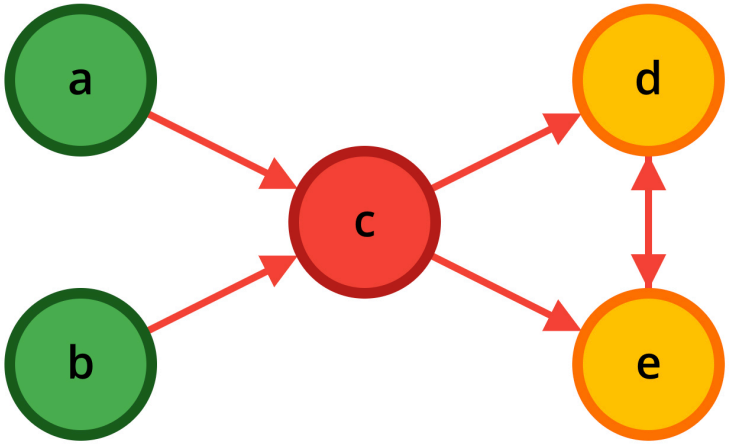


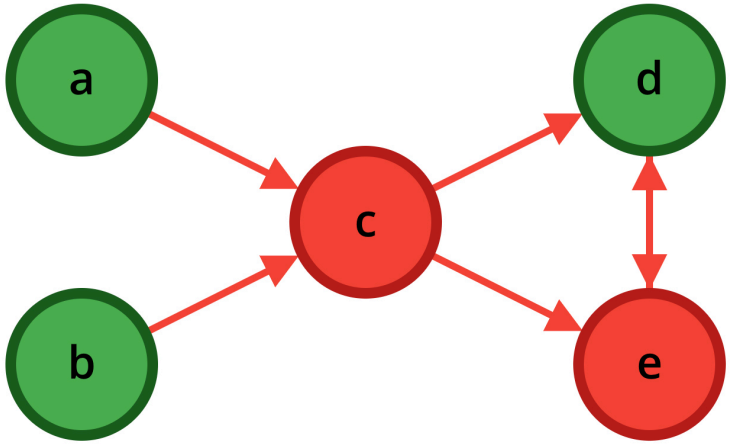
Labelling-Based Semantics

COM

a is **IN** ⟺ a is only attacked by **OUT**

a is **OUT** ⟺ a is attacked by some **IN**

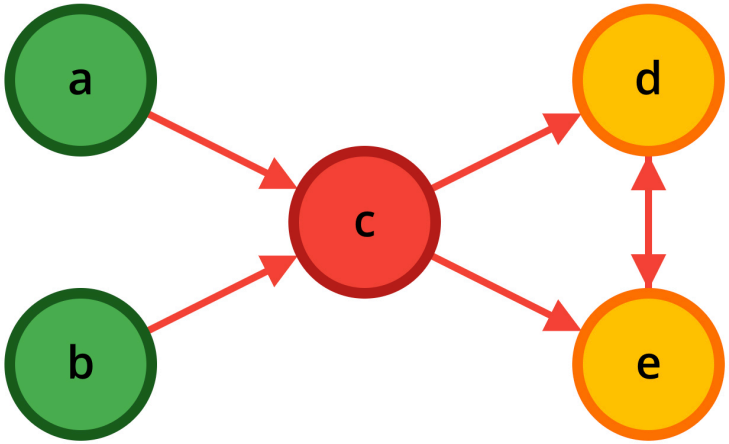




Labelling-Based Semantics

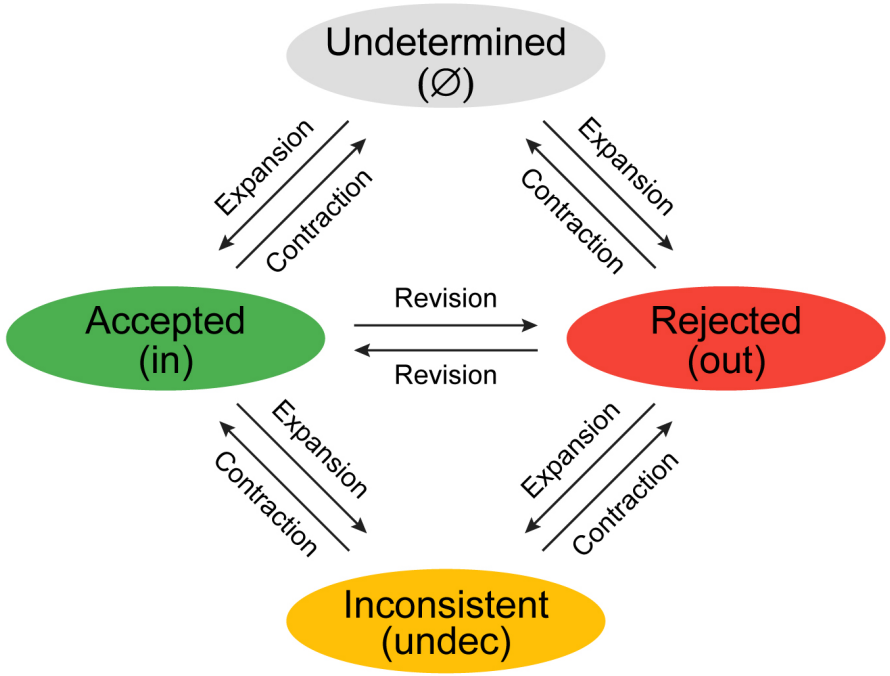
GDE

L is a complete labelling
L is minimal w.r.t. set inclusion



Connections with the AGM Framework

- **expansion**: increases the number of labels
- **contraction**: decreases the number of labels
- **revision**: changes a label from **in** to **out** (and vice versa)
- These operations can be implemented in our language



Concurrent Argumentation Language (CA)

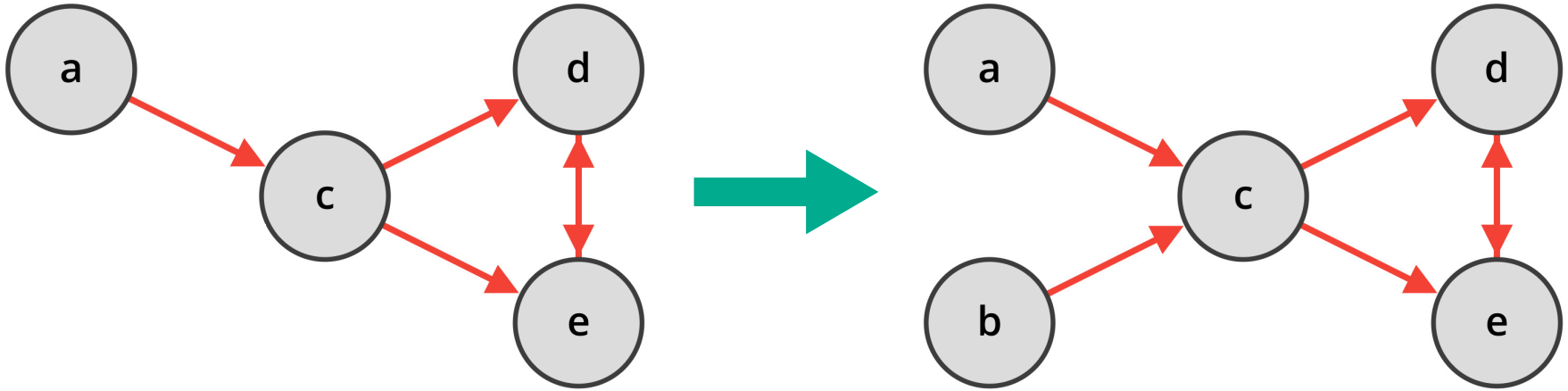
Syntax

$$
\begin{aligned}
A ::= \; & success \\
& \mid insert(Arg, R) \rightarrow A \\
& \mid rmv(Arg, R) \rightarrow A \\
& \mid A \| A \\
& \mid \exists_x A \\
& \mid E
\end{aligned}
\qquad
\begin{aligned}
E ::= \; & test_c(a, l, \sigma) \rightarrow A \\
& \mid test_s(a, l, \sigma) \rightarrow A \\
& \mid check(Arg, R) \rightarrow A \\
& \mid E + E \\
& \mid E +_P E \\
& \mid E \|_G E
\end{aligned}
$$

Concurrent Argumentation Language (CA)

Operational Semantics: insertion

$$\langle insert(Arg', R') \rightarrow A, \langle Arg, R \rangle \rangle \rightarrow \langle A, \langle Arg \cup Arg', R \cup R' \rangle \rangle \qquad \text{Insertion}$$



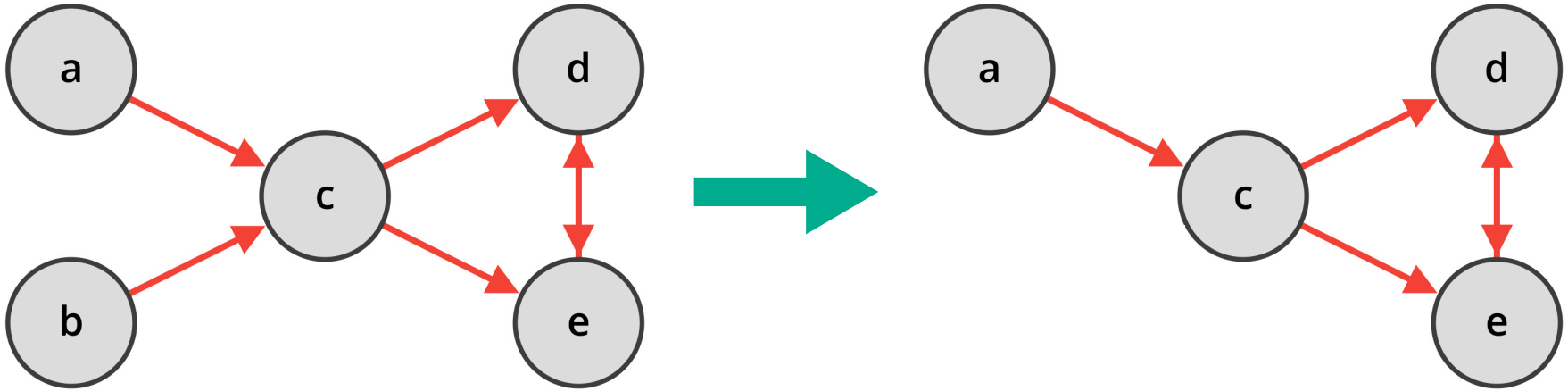
insert($\{b\}, \{(b, c)\}$)

Concurrent Argumentation Language (CA)

Operational Semantics: removal

$$\langle rmv(Arg', R') \rightarrow A, \langle Arg, R \rangle \rangle \rightarrow \langle A, \langle Arg \setminus Arg', R \setminus \{R' \cup R''\} \rangle \rangle$$
$$\text{where } R'' = \{(a, b) \in R \mid a \in Arg' \vee b \in Arg'\}$$

Removal



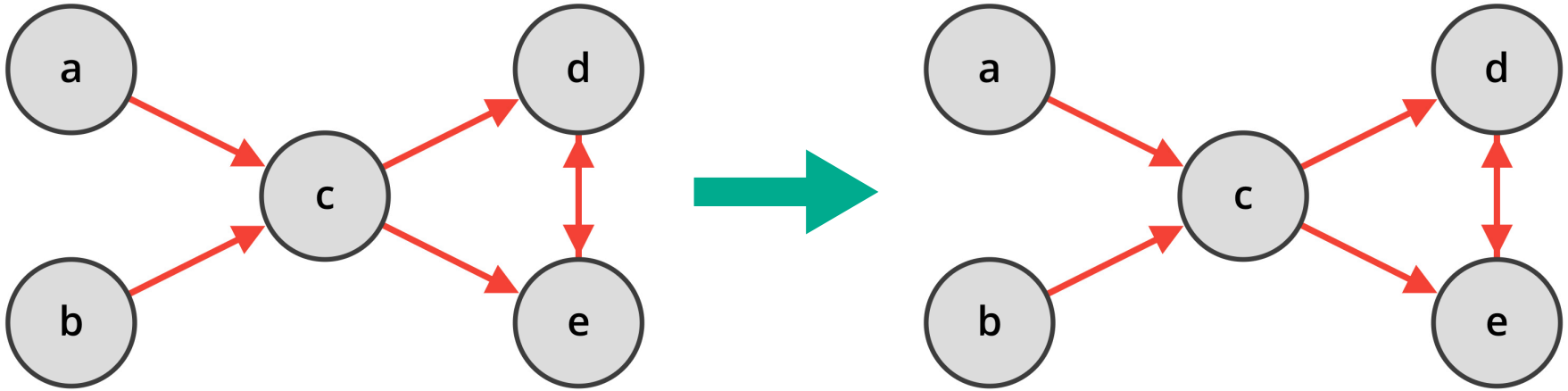
rmv($\{b\}$)

(b, c) is also removed

Concurrent Argumentation Language (CA)

Operational Semantics: check

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R}{\langle check(Arg', R') \rightarrow A, \langle Arg, R \rangle \rangle \rightarrow \langle A, \langle Arg, R \rangle \rangle} \quad \text{Check}$$



check($\{a\}, \{(a, c)\}$)

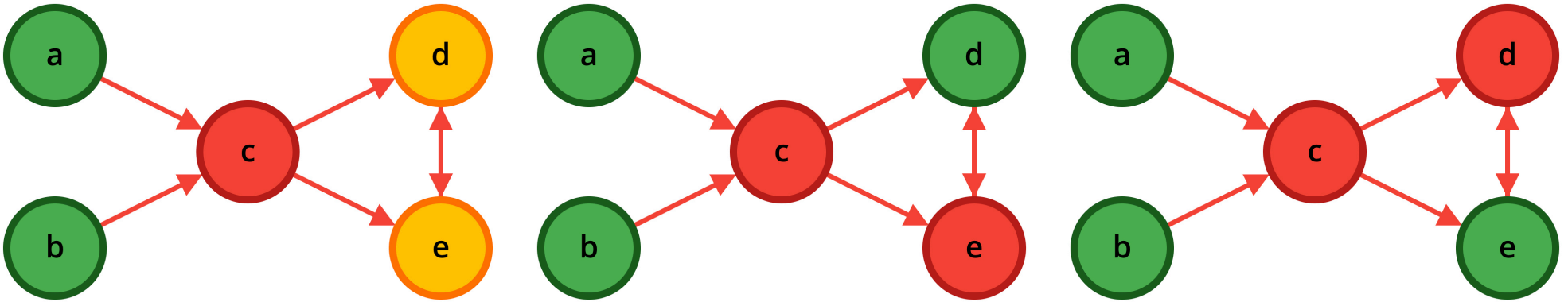
Concurrent Argumentation Language (CA)

Operational Semantics: c/s test

$$\frac{\exists L \in S_\sigma(F) \mid l \in L(a)}{\langle test_c(a, l, \sigma) \rightarrow A, F\rangle \rightarrow \langle A, F\rangle}$$

$$\frac{\forall L \in S_\sigma(F).l \in L(a)}{\langle test_s(a, l, \sigma) \rightarrow A, F\rangle \rightarrow \langle A, F\rangle}$$

Credulous and Sceptical Test



$\text{test}_C(\{a\}, \text{IN}, \text{complete})$

Concurrent Argumentation Language (CA)

Operational Semantics: constructs

$$\frac{\langle A_1, F\rangle \rightarrow \langle A_1', F'\rangle}{\begin{array}{c}\langle A_1 \| A_2, F\rangle \rightarrow \langle A_1' \| A_2, F'\rangle \\ \langle A_2 \| A_1, F\rangle \rightarrow \langle A_2 \| A_1', F'\rangle\end{array}} \qquad \frac{\langle A_1, F\rangle \rightarrow \langle success, F'\rangle}{\begin{array}{c}\langle A_1 \| A_2, F\rangle \rightarrow \langle A_2, F'\rangle \\ \langle A_2 \| A_1, F\rangle \rightarrow \langle A_2, F'\rangle\end{array}} \qquad \text{Parallelism}$$

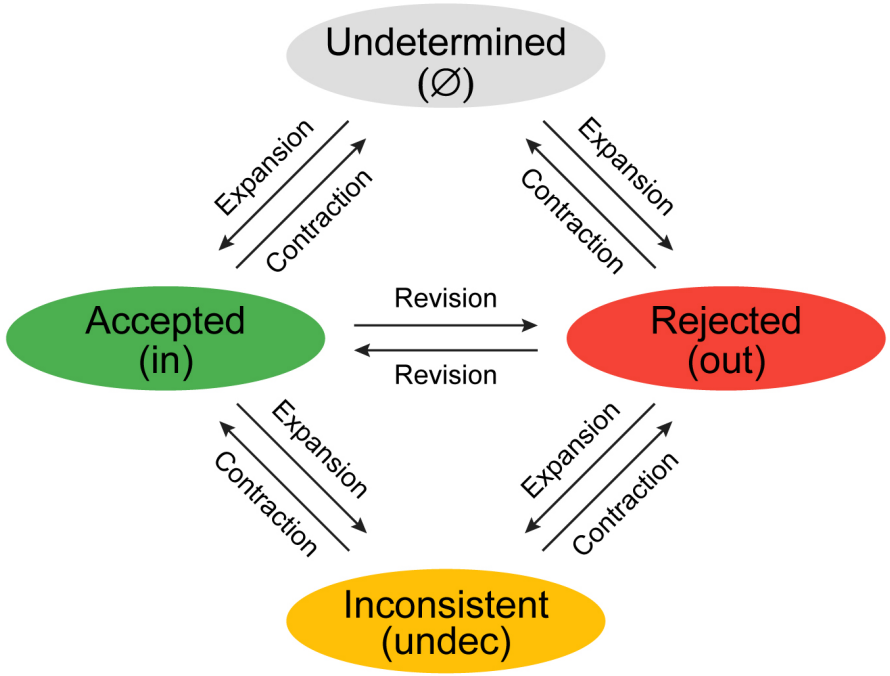
$$\frac{\langle E_1, F\rangle \rightarrow \langle A_1, F\rangle, \langle E_2, F\rangle \not\rightarrow}{\begin{array}{c}\langle E_1 \|_G E_2, F\rangle \rightarrow \langle A_1, F\rangle \\ \langle E_2 \|_G E_1, F\rangle \rightarrow \langle A_1, F\rangle\end{array}} \qquad \frac{\langle E_1, F\rangle \rightarrow \langle A_1, F\rangle, \langle E_2, F\rangle \rightarrow \langle A_2, F\rangle}{\langle E_1 \|_G E_2, F\rangle \rightarrow \langle A_1 \| A_2, F\rangle} \qquad \begin{array}{c}\text{Guarded} \\ \text{Parallelism}\end{array}$$

$$\frac{\langle E_1, F\rangle \rightarrow \langle A_1, F\rangle}{\begin{array}{c}\langle E_1 + E_2, F\rangle \rightarrow \langle A_1, F\rangle \\ \langle E_2 + E_1, F\rangle \rightarrow \langle A_1, F\rangle\end{array}} \qquad \text{Nondeterminism}$$

$$\frac{\langle E_1, F\rangle \rightarrow \langle A_1, F\rangle}{\langle E_1 +_P E_2, F\rangle \rightarrow \langle E_1, F\rangle} \qquad \frac{\langle E_1, F\rangle \not\rightarrow, \langle E_2, F\rangle \rightarrow \langle A_2, F\rangle}{\langle E_1 +_P E_2, F\rangle \rightarrow \langle E_2, F\rangle} \qquad \text{If Then Else}$$

Connections with the AGM Framework

- **expansion**: increases the number of labels
- **contraction**: decreases the number of labels
- **revision**: changes a label from **in** to **out** (and vice versa)
- These operations can be implemented in our language



Example of Expansion Operator

$$\oplus^{gde,in}_{a,L\ (L(a)=\emptyset)}(F) : add(\{a\},\{\}) \rightarrow success$$

$$\oplus^{gde,out}_{a,L\ (L(a)=\emptyset)}(F) : \sum_{b \in Arg} (test_c(b,in,gde) \rightarrow add(\{a\},\{(b,a)\})) \rightarrow success$$

$$+_P$$

$$add(\{a,u\},\{(u,a)\}) \rightarrow success$$

$$\oplus^{gde,undec}_{a,L\ (L(a)=in)}(F) : \sum_{b \in Arg} (test_c(b,undec,gde) \rightarrow add(\{\},\{(b,a)\})) \rightarrow success$$

$$+_P$$

$$add(\{\},\{(a,a)\}) \rightarrow success$$

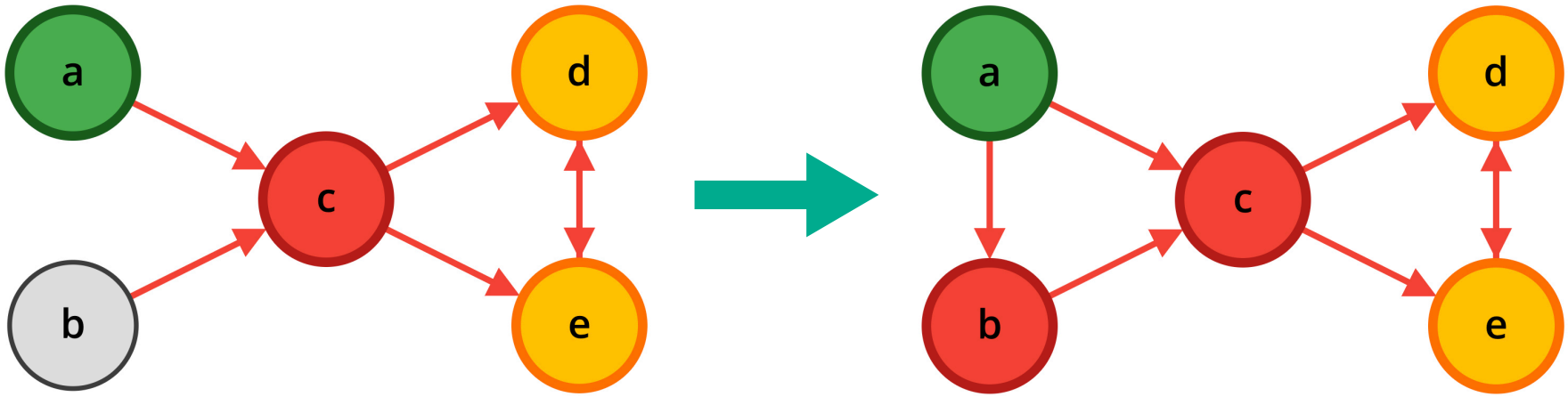
$$\oplus^{gde,undec}_{a,L\ (L(a)=out)}(F) : \|_{G\ b \in Arg} (test_c(b,in,gde) \wedge check(\{\},\{(b,a)\})$$

$$\rightarrow add(\{\},\{(a,b)\})) \rightarrow success$$

Example of Expansion Operator

Expanding **b** from ∅ to **out**:

- if there exists an **in** argument **a**, let **a** attack **b**



Example of Contraction Operator

$$\oslash^{gde,in}_{a,L}(F) : \underset{b \in Arg}{\|_G} (test_c(b, undec, gde) \rightarrow rmv(\{\}, \{(b,a)\})) \rightarrow success$$
$$(L(a)=undec)$$

$$\oslash^{gde,out}_{a,L}(F) : \sum_{b \in Arg} (test_c(b, in, gde) \rightarrow add(\{\}, \{b,a\})) \rightarrow success$$
$$(L(a)=undec)$$

$$+_P$$

$$add(\{u\}, \{u,a\}) \rightarrow success$$

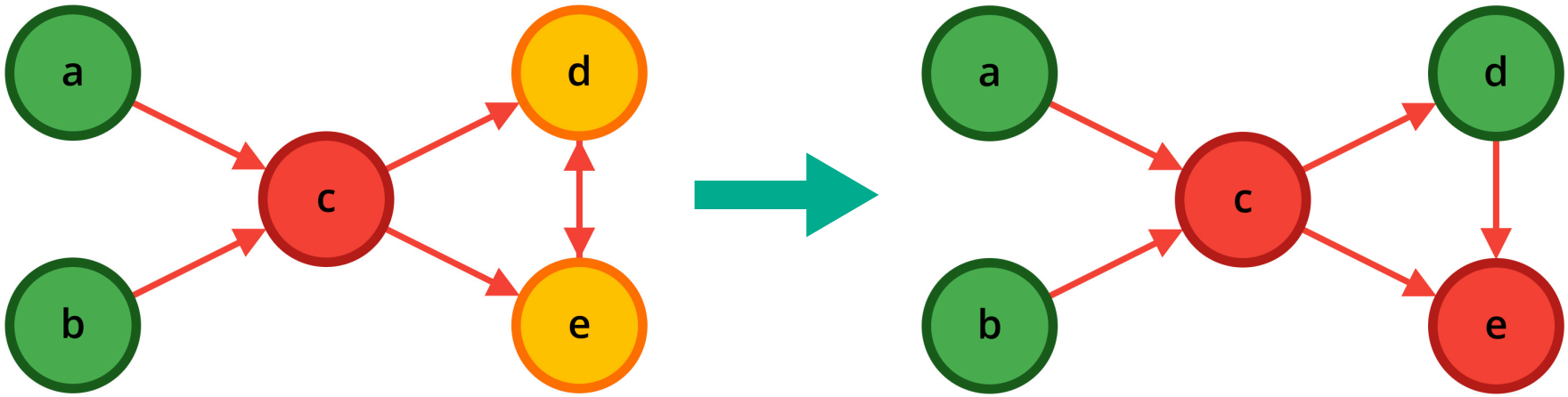
$$\oslash^{gde,\emptyset}_{a,L}(F) : rmv(\{a\}, \{\}) \rightarrow success$$
$$(L(a)=in)$$

$$\oslash^{gde,\emptyset}_{a,L}(F) : rmv(\{a\}, \{\}) \rightarrow success$$
$$(L(a)=out)$$

Example of Contraction Operator

Contracting **d** from **undec** to **in**:

- remove all attacks from **undec** arguments towards **d**



Example of Revision Operator

$$\circledast^{gde,out}_{\substack{a,L \\ (L(a)=in)}}(F): \sum_{b \in Arg} (test_c(b, in, gde) \rightarrow add(\{\}, \{(b,a)\})$$
$$\rightarrow \|_{\substack{G \\ c \in Arg}} (check(\{c\}, \{a,c\}) \rightarrow add(\{\}, \{(b,c)\})) \; \|_G \; true \rightarrow success$$

$$+_P$$

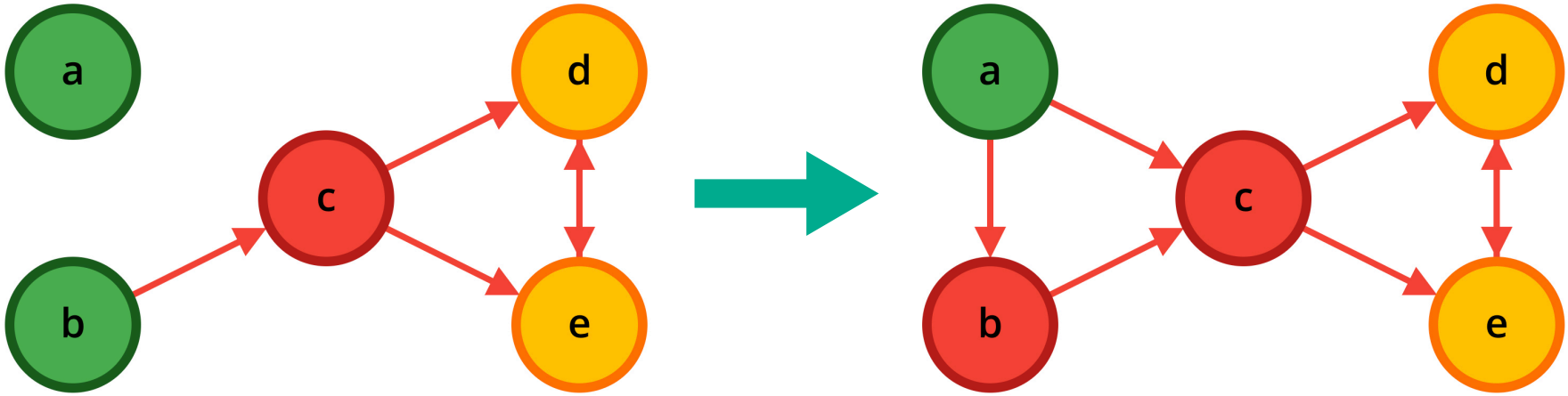
$$add(\{b\}, \{(b,a)\})$$
$$\rightarrow \|_{\substack{G \\ c \in Arg}} (check(\{c\}, \{a,c\}) \rightarrow add(\{\}, \{(b,c)\})) \; \|_G \; true) \rightarrow success$$

$$\circledast^{gde,in}_{\substack{a,L \\ (L(a)=out)}}(F): \|_{\substack{G \\ b \in Arg}} (test_c(b, \{in, undec\}, gde) \rightarrow rmv(\{\}, \{(b,a)\}))$$
$$\rightarrow \|_{\substack{G \\ c \in Arg}} ($$
$$test_c(c, \{in, undec\}, gde) \wedge check(\{c\}, \{a,c\})$$
$$\rightarrow rmv(\{\}, \{(a,c)\}) \; \|_G \; true$$
$$) \rightarrow success$$

Example of Revision Operator

Revising **b** from **in** to **out**:

- if there exists an **in** argument **a**, let **a** attack **b**
- and let **a** attack all arguments attacked by **b**



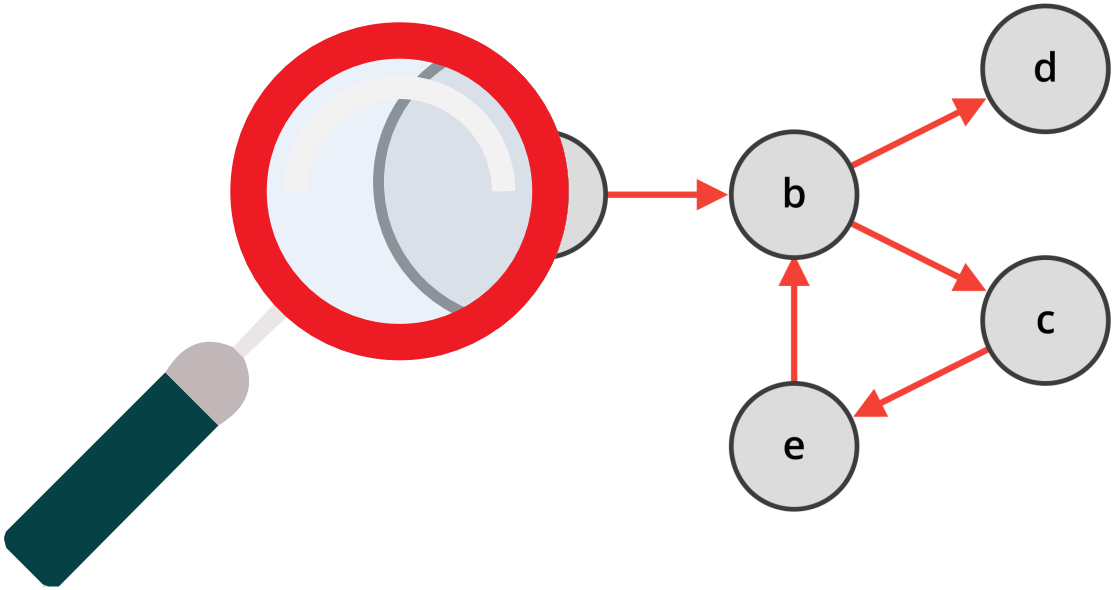
Conclusion

- Mapping between four-state labelling and classical semantics
- Definition of a concurrent language for argumentation
  - syntax + operational semantics
- Expansion, contraction and revision operators for AFs

- Introduced the semantics of failure to ensure termination
- Provided a working implementation of the language

Future Work

- Consider structured AFs instead of abstract
- Handle processes involving time-critical aspects
- Introduce a notion of “ownership” for arguments
  - Local store for agents



Thanks for your attention!

UNIVERSITÀ DEGLI STUDI
DI PERUGIA

G S
S I
GRAN SASSO
SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

A Concurrent Language for Argumentation: Preliminary Notes

Stefano Bistarelli and Carlo Taticchi

DIP2020