CILC 2024

Preserving Privacy in a (Timed) Concurrent Language for Argumentation

Stefano Bistarelli, Maria Chiara Meo and Carlo Taticchi

A.D. 1308
unipg
DIPARTIMENTO
DI MATEMATICA E INFORMATICA

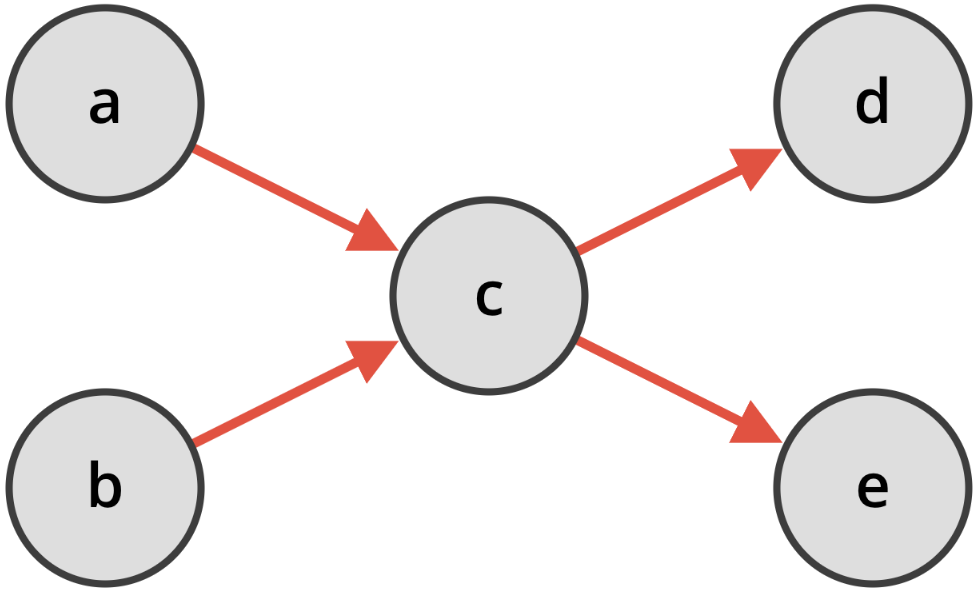
Ud'A
Università degli Studi "G. d'Annunzio"

Overview

- Abstract Argumentation Frameworks + Labelling
- Timed Concurrent Language for Argumentation
- Locality semantics
- Preserving Privacy in Multi-Agent Decision
- Conclusion & Future Work

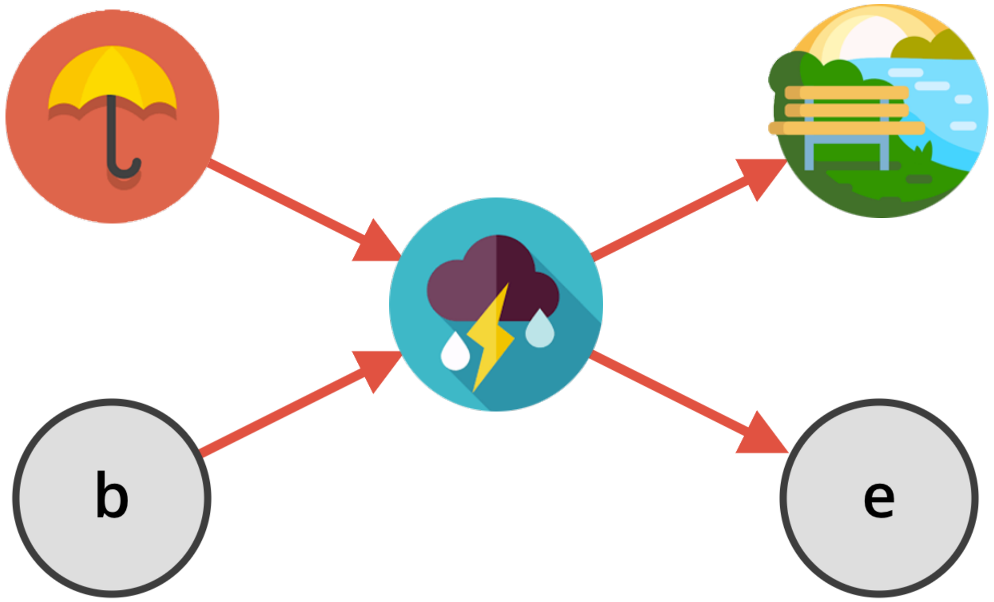
Abstract Argumentation

- Represent and evaluate arguments
- Abstract Argumentation Framework $F = \langle Arg, R \rangle$



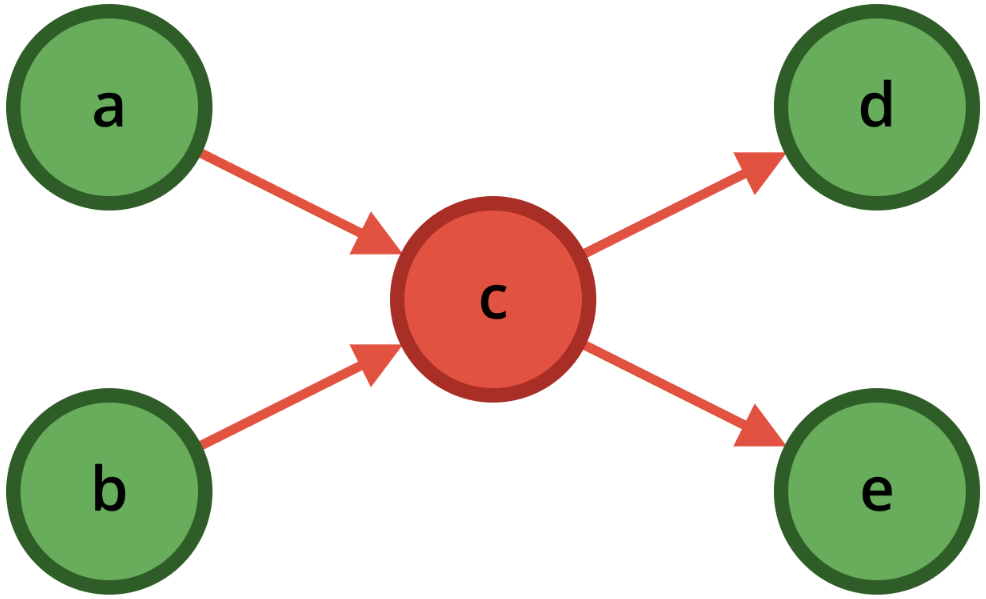
Abstract Argumentation

- Represent and evaluate arguments
- Abstract Argumentation Framework $F = \langle Arg, R \rangle$



Abstract Argumentation

- Represent and evaluate arguments
- Abstract Argumentation Framework $F = \langle Arg, R \rangle$
- Argumentation Semantics (e.g. Labelling)



An argument is:

- IN if it only attacked by OUT
- OUT if it is attacked by at least one IN
- UNDEC otherwise

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store
- Syntax:

$$
\begin{aligned}
P &::= \textit{let}\, C \textit{ in } A \\
C &::= p(x) :: A \mid C, C \\
A &::= \textit{success} \mid \textit{failure} \mid \textit{add}(Arg, R) \rightarrow A \mid \textit{rmv}(Arg, R) \rightarrow A \mid E \mid A \,\|\, A \mid \textit{new}\, S \textit{ in } A \mid p(x) \\
E &::= \textit{check}_t(Arg, R) \rightarrow A \mid \textit{c-test}_t(a, l, \sigma) \rightarrow A \mid \textit{s-test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store
- Syntax:

$$\begin{aligned} P &::= \textit{let}\ C\ \textit{in}\ A \\ C &::= p(x) :: A \mid C, C \\ A &::= \textit{success} \mid \textit{failure} \mid \textit{add}(Arg, R) \rightarrow A \mid \textit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \textit{new}\ S\ \textit{in}\ A \mid p(x) \\ E &::= \textit{check}_t(Arg, R) \rightarrow A \mid \textit{c-test}_t(a, l, \sigma) \rightarrow A \mid \textit{s-test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E \end{aligned}$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store
- Syntax:

$$
\begin{aligned}
P &::= \mathit{let}\ C\ \mathit{in}\ A \\
C &::= p(x) :: A \mid C, C \\
A &::= \mathit{success} \mid \mathit{failure} \mid \mathit{add}(Arg, R) \rightarrow A \mid \mathit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \mathit{new}\ S\ \mathit{in}\ A \mid p(x) \\
E &::= \mathit{check}_t(Arg, R) \rightarrow A \mid \mathit{c\text{-}test}_t(a, l, \sigma) \rightarrow A \mid \mathit{s\text{-}test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store

- Syntax:

$$
\begin{aligned}
P &::= \mathit{let}\ C\ \mathit{in}\ A\\
C &::= p(x) :: A \mid C, C\\
A &::= \mathit{success} \mid \mathit{failure} \mid \mathit{add}(Arg, R) \rightarrow A \mid \mathit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \mathit{new}\ S\ \mathit{in}\ A \mid p(x)\\
E &::= \mathit{check}_t(Arg, R) \rightarrow A \mid \mathit{c\text{-}test}_t(a, l, \sigma) \rightarrow A \mid \mathit{s\text{-}test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store
- Syntax:

$$
\begin{aligned}
P &::= \mathit{let}\ C\ \mathit{in}\ A \\
C &::= p(x) :: A \mid C, C \\
A &::= \mathit{success} \mid \mathit{failure} \mid \mathit{add}(Arg, R) \rightarrow A \mid \mathit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \mathit{new}\ S\ \mathit{in}\ A \mid p(x) \\
E &::= \mathit{check}_t(Arg, R) \rightarrow A \mid \mathit{c\text{-}test}_t(a, l, \sigma) \rightarrow A \mid \mathit{s\text{-}test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store

- Syntax:

$$
\begin{aligned}
P &::= \textit{let}\ C\ \textit{in}\ A\\
C &::= p(x) :: A \mid C, C\\
A &::= \textit{success} \mid \textit{failure} \mid \textit{add}(Arg, R) \rightarrow A \mid \textit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \textit{new}\ S\ \textit{in}\ A \mid p(x)\\
E &::= \textit{check}_t(Arg, R) \rightarrow A \mid \textit{c-test}_t(a, l, \sigma) \rightarrow A \mid \textit{s-test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store

- Syntax:

$$
\begin{aligned}
P &::= \textit{let}\ C\ \textit{in}\ A \\
C &::= p(x) :: A \mid C, C \\
A &::= \textit{success} \mid \textit{failure} \mid \textit{add}(Arg, R) \rightarrow A \mid \textit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \textit{new}\ S\ \textit{in}\ A \mid p(x) \\
E &::= \textit{check}_t(Arg, R) \rightarrow A \mid \textit{c-test}_t(a, l, \sigma) \rightarrow A \mid \textit{s-test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Timed Concurrent Language for Argumentation (TCLA)

- Argumentation-based communication between concurrent agents sharing a common store

- Syntax:

$$
\begin{aligned}
P &::= \textit{let}\, C \textit{ in } A\\
C &::= p(x) :: A \mid C, C\\
A &::= \textit{success} \mid \textit{failure} \mid \textit{add}(Arg, R) \rightarrow A \mid \textit{rmv}(Arg, R) \rightarrow A \mid E \mid A \parallel A \mid \textit{new}\, S \textit{ in } A \mid p(x)\\
E &::= \textit{check}_t(Arg, R) \rightarrow A \mid \textit{c-test}_t(a, l, \sigma) \rightarrow A \mid \textit{s-test}_t(a, l, \sigma) \rightarrow A \mid E + E \mid E +_P E
\end{aligned}
$$

Parallel executions

- **True concurrency**: we assume infinite processors

Parallel executions

- **True concurrency**: we assume infinite processors
- **Global clock** for the the passing of time

Parallel executions

- **True concurrency**: we assume infinite processors
- **Global clock** for the the passing of time
- We decrement the **timeout environment** $T : \mathscr{I} \rightharpoonup \mathbb{N} \cup \{\infty\}$

$$T[\bar{I} : \bar{t}](I) = \begin{cases} T(I) & \text{if } I \neq \bar{I} \\ \bar{t} & \text{otherwise} \end{cases}$$

$$dec(T)(I) = \begin{cases} T(I) - 1 & \text{if } 0 < T(I) \in \mathbb{N} \\ T(I) & \text{if } T(I) = 0 \text{ or } T(I) = \infty \end{cases}$$

I	T(I)
Agent A	4
Agent B	1
Agent C	2

True concurrency

$$\frac{\langle A_1, F, T\rangle \longrightarrow \langle A_1', F', T_1\rangle, \quad \langle A_2, F, T\rangle \longrightarrow \langle A_2', F'', T_2\rangle}{\langle A_1 \parallel A_2, F, T\rangle \longrightarrow \langle A_1' \parallel A_2', *(F, F', F''), T_1 \cup T_2\rangle}$$

- With $(T_1 \cup T_2)(I) = \begin{cases} T_1(I) & \text{if } I \in dom(T_1) \\ T_2(I) & \text{otherwise} \end{cases}$

True concurrency

$$\frac{\langle A_1, F, T\rangle \longrightarrow \langle A_1', F', T_1\rangle, \quad \langle A_2, F, T\rangle \longrightarrow \langle A_2', F'', T_2\rangle}{\langle A_1 \parallel A_2, F, T\rangle \longrightarrow \langle A_1' \parallel A_2', *(F, F', F''), T_1 \cup T_2\rangle}$$

- With $(T_1 \cup T_2)(I) = \begin{cases} T_1(I) & \text{if } I \in dom(T_1) \\ T_2(I) & \text{otherwise} \end{cases}$
- Possible implementation
  $*(F, F', F'') := (F' \cap F'') \cup ((F' \cup F'') \setminus F)$

True concurrency

$$\frac{\langle A_1, F, T\rangle \longrightarrow \langle A_1', F', T_1\rangle, \quad \langle A_2, F, T\rangle \longrightarrow \langle A_2', F'', T_2\rangle}{\langle A_1 \parallel A_2, F, T\rangle \longrightarrow \langle A_1' \parallel A_2', *(F, F', F''), T_1 \cup T_2\rangle}$$

- With $(T_1 \cup T_2)(I) = \begin{cases} T_1(I) & \text{if } I \in dom(T_1) \\ T_2(I) & \text{otherwise} \end{cases}$
- Possible implementation
  $*\,(F, F', F'') := (F' \cap F'') \cup ((F' \cup F'') \backslash F)$
- Alternative approach: interleaving

Addition & removal

$$\langle \mathit{add}(Arg', R') \rightarrow A, \langle Arg, R\rangle, T\rangle \longrightarrow \langle A, \langle Arg \cup Arg', (R \cup R')_{\|(Arg \cup Arg')}\rangle, \mathit{dec}(T)\rangle$$

$$\langle \mathit{rmv}(Arg', R') \rightarrow A, \langle Arg, R\rangle, T\rangle \longrightarrow \langle A, \langle Arg \setminus Arg', (R \setminus R')_{\|(Arg \setminus Arg')}\rangle, \mathit{dec}(T)\rangle$$

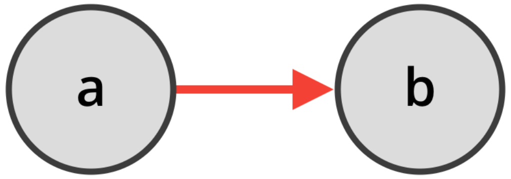
- Example: add({a,b},{(a,b)}) -> rmv({a},{}) -> success;

Addition & removal

$$\langle \textbf{\textit{add}}(Arg', R') \rightarrow A, \langle Arg, R\rangle, T\rangle \longrightarrow \langle A, \langle Arg \cup Arg', (R \cup R')_{\|(Arg \cup Arg')}\rangle, \textbf{\textit{dec}}(T)\rangle$$

$$\langle \textbf{\textit{rmv}}(Arg', R') \rightarrow A, \langle Arg, R\rangle, T\rangle \longrightarrow \langle A, \langle Arg \setminus Arg', (R \setminus R')_{\|(Arg \setminus Arg')}\rangle, \textbf{\textit{dec}}(T)\rangle$$

- Example: add({a,b},{(a,b)}) -> rmv({a},{}) -> success;

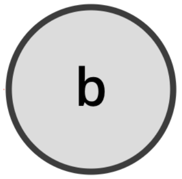


Addition & removal

$$\langle \textit{add}(Arg', R') \rightarrow A, \langle Arg, R\rangle, T\rangle \longrightarrow \langle A, \langle Arg \cup Arg', (R \cup R')_{\|(Arg \cup Arg')}\rangle, \textit{dec}(T)\rangle$$

$$\langle \textit{rmv}(Arg', R') \rightarrow A, \langle Arg, R\rangle, T\rangle \longrightarrow \langle A, \langle Arg \setminus Arg', (R \setminus R')_{\|(Arg \setminus Arg')}\rangle, \textit{dec}(T)\rangle$$

- Example: add({a,b},{(a,b)}) -> rmv({a},{}) -> success;



Check

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ \ t > 0}{\langle \mathit{check}_t(Arg', R') \to A, F, T \rangle \longrightarrow \langle A, F, \mathit{dec}(T) \rangle}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ \ t > 0}{\langle \mathit{check}_t(Arg', R') \to A, F, T \rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \to A, F, \mathit{dec}(T[I : t]) \rangle}$$

where I is a fresh timeout identifier

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ \ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \to A, F, T \rangle \longrightarrow \langle A, F, \mathit{dec}(T) \rangle}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ \ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \to A, F, T \rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \to A, F, \mathit{dec}(T) \rangle}$$

$$\langle \mathit{check}_0(Arg', R') \to A, F, T \rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T) \rangle$$

$$\frac{T(I) = 0}{\langle \mathit{check}_I(Arg', R') \to A, F, T \rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T) \rangle}$$

Check

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ \ t > 0}{\langle \mathit{check}_t(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle A, F, \mathit{dec}(T)\rangle}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ \ t > 0}{\langle \mathit{check}_t(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \rightarrow A, F, \mathit{dec}(T[I : t])\rangle}$$

where I is a fresh timeout identifier

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ \ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle A, F, \mathit{dec}(T)\rangle}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ \ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \rightarrow A, F, \mathit{dec}(T)\rangle}$$

$$\langle \mathit{check}_0(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T)\rangle$$

$$\frac{T(I) = 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T)\rangle}$$

Check

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ \ t > 0}{\langle \mathit{check}_t(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle A, F, \mathit{dec}(T)\rangle}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ \ t > 0}{\langle \mathit{check}_t(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \rightarrow A, F, \mathit{dec}(T[I : t])\rangle}$$

where I is a fresh timeout identifier

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ \ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle A, F, \mathit{dec}(T)\rangle}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ \ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \rightarrow A, F, \mathit{dec}(T)\rangle}$$

$$\langle \mathit{check}_0(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T)\rangle$$

$$\frac{T(I) = 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T\rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T)\rangle}$$

Test

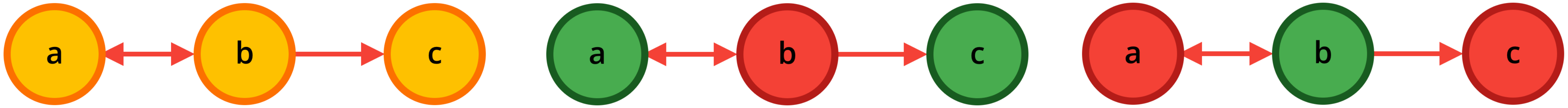
- Test semantics are similar to the check one but for the conditions to satisfy

- **Credulous test:** $\exists L \in \mathscr{L}_{\sigma}^{F} \mid L(a) = l$

- **Sceptical test:** $\forall L \in \mathscr{L}_{\sigma}^{F} \mid L(a) = l$

Test

- Test semantics are similar to the check one but for the conditions to satisfy
- **Credulous test:** $\exists L \in \mathscr{L}_{\sigma}^{F} \mid L(a) = l$
- **Sceptical test:** $\forall L \in \mathscr{L}_{\sigma}^{F} \mid L(a) = l$
- Example: $\mathrm{ctest}(2,\{\mathrm{b}\},\mathrm{IN},\mathrm{complete})$ / $\mathrm{stest}(2,\{\mathrm{s}\},\mathrm{IN},\mathrm{complete})$



Locality semantics

$$\frac{\langle A, (AF \uparrow S) \cup AF_{loc}, T\rangle \longrightarrow \langle \mathsf{B}, AF', T'\rangle}{\langle \textit{new}\ S\ \textit{in}\ A^{AF_{loc}}, AF, T\rangle \longrightarrow \langle \textit{new}\ S\ \textit{in}\ B^{AF'\downarrow S}, (AF' \uparrow S) \cup (AF'' \downarrow S), T'\rangle}$$

$$\text{where } AF = \langle Arg, R\rangle,\ AF' = \langle Arg', R'\rangle \text{ and } AF'' = \langle Arg, R_{\|Arg' \cup S}\rangle$$

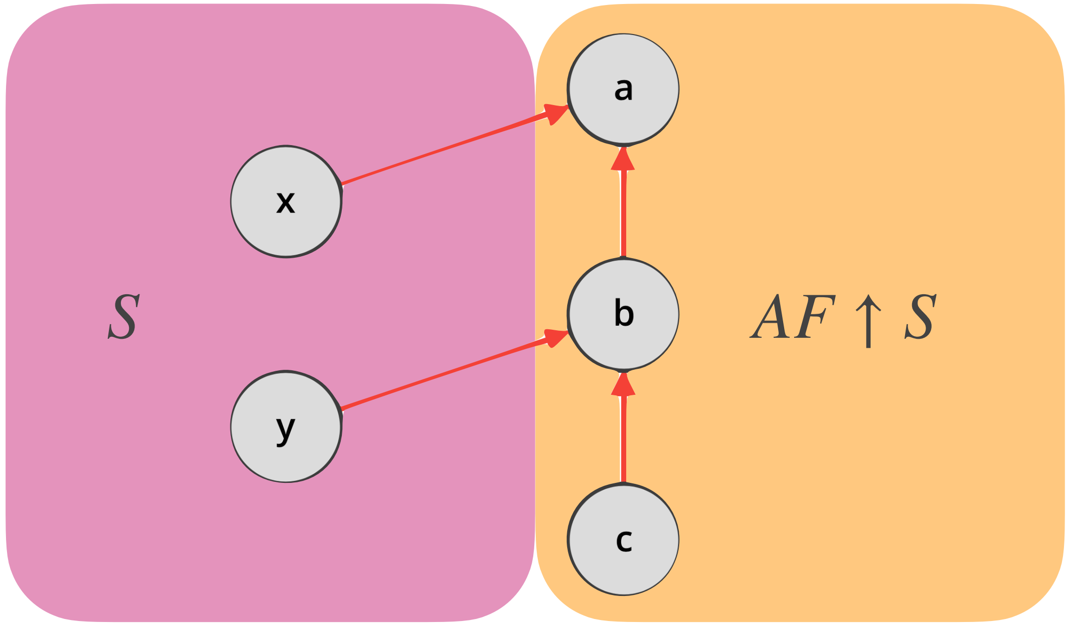
- $new\ S\ in\ A$ behaves like A where arguments in S are local to A
- AF_{loc} contains information on S which is **hidden** from the external AF

Locality semantics

$$\frac{\langle A, (AF \uparrow S) \cup AF_{loc}, T\rangle \longrightarrow \langle \mathsf{B}, AF', T'\rangle}{\langle \textit{new}\ S\ \textit{in}\ A^{AF_{loc}}, AF, T\rangle \longrightarrow \langle \textit{new}\ S\ \textit{in}\ B^{AF'\downarrow S}, (AF' \uparrow S) \cup (AF'' \downarrow S), T'\rangle}$$

$$\text{where } AF = \langle Arg, R\rangle,\ AF' = \langle Arg', R'\rangle \text{ and } AF'' = \langle Arg, R_{\|Arg' \cup S}\rangle$$

- $new\ S\ in\ A$ behaves like A where arguments in S are local to A
- AF_{loc} contains information on S which is **hidden** from the external AF



Example: Multi-Agent Decision Making with Privacy Preserved problem

- Charlie's, Alice's and Bob's beliefs[1]

$$\begin{array}{lllll}
\mathit{Charlie}: & \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob} \rangle} & \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali} \rangle} \longleftarrow \mathit{Wrong} & \mathit{Suez} & \mathit{Urgency} \\
\mathit{Alice}: & \underline{\mathrm{Sw}} & \underline{\mathrm{Hw}} \longleftarrow \mathbf{NoStock} \longleftarrow \mathit{SpecDel} & & \\
\mathit{Bob}: & \underline{\mathrm{Sw}} \longleftarrow \mathbf{NoTec} \longleftarrow \mathit{NewTwo} & \underline{\mathrm{Hw}} & \mathit{NewOffice} & \\
\mathit{Public\ Attacks}: & \mathit{SpecDel} \longleftarrow \mathit{Suez} & \mathit{NewTwo} \longleftarrow \mathit{Urgency} & \mathit{Wrong} \longleftarrow \mathit{NewOffice} &
\end{array}$$

[1] Yang Gao, Francesca Toni, Hao Wang, Fanjiang Xu: Argumentation-Based Multi-Agent Decision Making with Privacy Preserved. AAMAS 2016: 1153-1161

Example: Multi-Agent Decision Making with Privacy Preserved problem

- Charlie's, Alice's and Bob's beliefs[1]

$$\mathit{Charlie}: \quad \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob} \rangle} \qquad \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali} \rangle} \longleftarrow \mathit{Wrong} \qquad \mathit{Suez} \qquad \mathit{Urgency}$$

$$\mathit{Alice}: \quad \underline{\mathrm{Sw}} \qquad \underline{\mathrm{Hw}} \longleftarrow \mathbf{NoStock} \longleftarrow \mathit{SpecDel}$$

$$\mathit{Bob}: \quad \underline{\mathrm{Sw}} \longleftarrow \mathbf{NoTec} \longleftarrow \mathit{NewTwo} \qquad \underline{\mathrm{Hw}} \qquad \mathit{NewOffice}$$

$$\mathit{Public\ Attacks}: \quad \mathit{SpecDel} \longleftarrow \mathit{Suez} \qquad \mathit{NewTwo} \longleftarrow \mathit{Urgency} \qquad \mathit{Wrong} \longleftarrow \mathit{NewOffice}$$

- Acceptable solutions:

$$Sol = \{ \langle C : \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob} \rangle}, \mathrm{A}{:}\underline{\mathrm{Hw}}, \mathrm{B}{:}\underline{\mathrm{Sw}} \rangle, \ \langle C : \underline{\langle \mathrm{Sw}_{Ali}, \mathrm{Hw}_{Bob} \rangle}, \mathrm{A}{:}\underline{\mathrm{Sw}}, \mathrm{B}{:}\underline{\mathrm{Hw}} \rangle \ \}$$

---

[1] Yang Gao, Francesca Toni, Hao Wang, Fanjiang Xu: Argumentation-Based Multi-Agent Decision Making with Privacy Preserved. AAMAS 2016: 1153-1161

tcla for DMPP

- We can write a tcla program emulating a DMPP problem using N tcla agents in parallel
  - Each agent builds its local framework by using an *add* step
  - The computation starts in the initial public argumentation framework

$$
\begin{aligned}
AFd = \langle \quad & \{\ \mathit{SpecDel}, \mathit{Suez}, \mathit{NewTwo}, \mathit{Urgency}, \mathit{Wrong}, \mathit{NewOffice}, \\
& \quad \langle C : \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob} \rangle}, \mathrm{A}{:}\underline{\mathrm{Hw}}, \mathrm{B}{:}\underline{\mathrm{Sw}} \rangle,\ \langle C : \underline{\langle \mathrm{Sw}_{Ali}, \mathrm{Hw}_{Bob} \rangle}, \mathrm{A}{:}\underline{\mathrm{Sw}}, \mathrm{B}{:}\underline{\mathrm{Hw}} \rangle\ \} \\
& \{\ (\mathit{Suez}, \mathit{SpecDel}), (\mathit{Urgency}, \mathit{NewTwo}), (\mathit{NewOffice}, \mathit{Wrong})\ \}\ \rangle.
\end{aligned}
$$

Example

$$
\begin{aligned}
&\mathcal{T}(Charlie) = \\
&\quad \textit{new}\,(\{\underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}, \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle}\})\ \textit{in}\ T_C(\underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}, \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle})^{AFp_{Charlie}}
\end{aligned}
$$

$$
\begin{aligned}
&T_C(\underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}, \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle}) = \\
&\quad \textit{c-test}_1(\underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}, \textit{in}, \textit{adm}) \rightarrow \\
&\qquad (\ \textit{add}(\{C : \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}\}, \\
&\qquad\qquad \{(C : \underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle}, \langle C : \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}, A : \underline{\mathrm{Sw}}, B : \underline{\mathrm{Hw}}\rangle)\}) \rightarrow \\
&\qquad\qquad (\ (\ \textit{c-test}_1(\langle C : \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}, A : \underline{\mathrm{Sw}}, B : \underline{\mathrm{Hw}}\rangle, \textit{in}, \textit{adm}) \vee \\
&\qquad\qquad\quad \textit{c-test}_1(\langle C : \underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle}, A : \underline{\mathrm{Hw}}, B : \underline{\mathrm{Sw}}\rangle, \textit{in}, \textit{adm})\ ) \rightarrow \\
&\qquad\qquad\qquad (\ \textit{add}(\{tok_A\}, \emptyset) \rightarrow \\
&\qquad\qquad\qquad\quad (\ \textit{check}_\infty(\{gd\}, \emptyset) \rightarrow success\ + \\
&\qquad\qquad\qquad\quad\ \ \textit{check}_\infty(\{ngd_C\}, \emptyset) \rightarrow \textit{rmv}(\{ngd_C,\ C : \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}\}, \emptyset) \rightarrow \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad T_C(\underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle})\ )\ )\ ) \\
&\qquad\qquad +_P \\
&\qquad\qquad \ldots
\end{aligned}
$$

Translation

- $Agent_1$ checks whether its preferred action choice a is globally feasible $\mathrm{ctest}(1,\{a\},IN,admissible)$
- If this is the case, $Agent_1$ adds the $Agent_1$:a to the public AF and checks its partial consistency, namely $\exists s \in Sol \mid \mathrm{ctest}(1,\{s\},IN,admissible)$
  - If $Agent_i$:a is consistent, either continues with other agents or terminate with success
  - If $Agent_i$:a is not consistent, $Agent_i$ removes it from the public AF
- If no action is found which can be extended to find a solution, the computation terminates with failure

Conclusion

- Functionalities of the Timed Concurrent Language for Argumentation which can be used to implement decision-making processes

Conclusion

- Functionalities of the Timed Concurrent Language for Argumentation which can be used to implement decision-making processes
- Local stores for enforcing privacy between agents

Conclusion

- Functionalities of the Timed Concurrent Language for Argumentation which can be used to implement decision-making processes
- Local stores for enforcing privacy between agents
- Illustrative example demonstrating how the Timed Concurrent Language for Argumentation can be used for modelling DMPP problems

Conclusion

- Functionalities of the Timed Concurrent Language for Argumentation which can be used to implement decision-making processes
- Local stores for enforcing privacy between agents
- Illustrative example demonstrating how the Timed Concurrent Language for Argumentation can be used for modelling DMPP problems
- Automatic translation from a DMPP problem to a tcla program

Future Perspectives

- Explore other features of tcla to simplify the construction of the models and achieve more natural interactions with the native constructs

Future Perspectives

- Explore other features of tcla to simplify the construction of the models and achieve more natural interactions with the native constructs
- Further develop illustrative examples to showcase the system’s effectiveness and highlight limitations across various scenarios

Future Perspectives

- Explore other features of tcla to simplify the construction of the models and achieve more natural interactions with the native constructs
- Further develop illustrative examples to showcase the system's effectiveness and highlight limitations across various scenarios
- Extend tcla to model real-world applications where agents can coordinate autonomously and concurrently without being bound to a fixed agent ordering

Future Perspectives

- Explore other features of tcla to simplify the construction of the models and achieve more natural interactions with the native constructs
- Further develop illustrative examples to showcase the system's effectiveness and highlight limitations across various scenarios
- Extend tcla to model real-world applications where agents can coordinate autonomously and concurrently without being bound to a fixed agent ordering
- Endow the agents with a notion of ownership to establish which actions can be performed on the shared arguments

Preserving Privacy in a (Timed) Concurrent Language for Argumentation

Stefano Bistarelli, Maria Chiara Meo and Carlo Taticchi

Thank you for your attention!