Preserving Privacy in a (Timed) Concurrent Language for Argumentation

Stefano Bistarelli[1], Maria Chiara Meo[2] and Carlo Taticchi[1,*]

[1]Department of Mathematics and Computer Science, University of Perugia, Italy
[2]Department of Economics, University "G. d'Annunzio" of Chieti-Pescara, Italy

Abstract

Modelling the behaviour of multiple agents concurrently interacting and reasoning in a dynamic environment is a challenging task. It necessitates tools capable of effectively capturing various forms of interaction, such as persuasion and deliberation while aiding agents in decision-making or consensus-building. In [1], we extended a language for modelling concurrent interactions between agents (*tcla*), which allowed us to specify agents equipped with a local argument memory and to reason with private knowledge. Furthermore, an initial formalisation of a multi-agent decision problem that preserves privacy has been provided. To illustrate the language's capabilities, in this paper, we give a complete formalisation of a privacy-preserving multi-agent decision problem, and we demonstrate how it can be employed to define a general (correct and complete) translation function that generates a *tcla* program from a multi-agent decision-making process. Additionally, we present an application example that models a privacy-preserving multi-agent decision-making process to showcase an instance of our general translation function.

Keywords

Computational Argumentation, Concurrency, Locality

1. Introduction

Intelligent agents can exploit argumentation techniques to accomplish complex interactions like, for instance, negotiation [2, 3] and persuasion [4, 5]. The Timed Concurrent Language for Argumentation (TCLA) [6, 7] offers constructs to implement such interactions. Agents involved in the process share an argumentation store that serves as a knowledge base and where arguments and attacks represent the agreed beliefs. The framework can be changed via a set of primitives that allow the addition and removal of arguments and attacks. The language makes use of two kinds of expressions: a syntactic check that verifies if a given set of arguments and attacks is contained in the knowledge base, and semantic test operations that retrieve information about the acceptability of arguments in the AF.

The presented study extends the work in [1], where we introduced the formalism of local stores and provided operational semantics for the locality operator within TCLA. The primary focus of [1] was to establish the theoretical foundation and mechanisms for autonomous agents to reason with private information, allowing them to hide data they are unwilling to disclose and only reveal the necessary information to reach desired outcomes. In contrast, this paper advances that foundational work by defining a general translation function that generates a TCLA program from a multi-agent decision-making process. This implements the theoretical constructs introduced earlier and bridges the gap between theory and practical applications. Additionally, we include practical examples to demonstrate the effectiveness of TCLA, with particular emphasis on using local stores. These examples showcase how the enhanced framework can be applied to complex, real-world scenarios, providing a more comprehensive evaluation of its capabilities and practical benefits.

CILC 2024: 39th Italian Conference on Computational Logic, June 26-28, 2024, Rome, Italy
*Corresponding author.
stefano.bistarelli@unipg.it (S. Bistarelli); mariachiara.meo@unich.it (M. C. Meo); carlo.taticchi@unipg.it (C. Taticchi)
0000-0001-7411-9678 (S. Bistarelli); 0000-0002-3700-3788 (M. C. Meo); 0000-0003-1260-4672 (C. Taticchi)

2. Background

In this section, we recall some fundamental notions concerning Computational Argumentation. The first definition we need is that of Abstract Argumentation Framework (AF) [8].

Definition 1. *Let U be the set of all possible arguments, which we refer to as the* universe.[1] *An Abstract Argumentation Framework is a pair $\langle Arg, R\rangle$ where $Arg \subseteq U$ is a set of* adopted *arguments and R is a binary relation on Arg (representing attacks among adopted arguments).*

Given an AF, we aim to identify subsets of *acceptable* arguments using argumentation semantics. Different semantics, such as admissible, complete, stable, semi-stable, preferred, and grounded (denoted as *adm*, *com*, *stb*, *sst*, *prf*, and *gde*), have been introduced to reflect desirable qualities for argument sets [9, 8]. Labelling-based semantics offer an approach to identify acceptable arguments by associating an AF with a subset of possible labellings [10].

Definition 2. *A labelling of an AF $F = \langle Arg, R\rangle$ is a total function $L : Arg \to \{in, out, undec\}$. Moreover, L is an admissible labelling for F when $\forall a \in Arg$*

- $L(a) = in \implies \forall b \in Arg \mid (b, a) \in R. L(b) = out$;
- $L(a) = out \iff \exists b \in Arg \mid (b, a) \in R \wedge L(b) = in$.

In particular, *in* arguments are acceptable, while the others will be rejected. Similar criteria to that shown in Definition 2 can be used to capture other semantics [10]. In the following, we will write $\mathcal{L}^F_\sigma$ to identify the set of all possible labellings of F with respect to the semantics σ. Besides computing the possible labellings with respect to a certain semantics σ, one of the most common tasks performed on AFs is to decide whether an argument a is accepted (labelled as *in*) in some labelling of $\mathcal{L}^F_\sigma$ or in all labellings. In the former case, we say that a is *credulously* accepted with respect to σ; in the latter, a is instead *sceptically* accepted with respect to σ. In the following, we say that an argument a is admissible (in AF) if and only if a is *credulously* accepted with respect to *adm*.

In the following, we summarise the main features of TCLA, including its syntax and operational semantics. For more details, we direct the reader to the work in [1, 6]. Communication between TCLA agents is implemented via a shared memory consisting of an AF which agents can access and modify. All agents are synchronised via a shared global clock, tracking the simultaneous execution of concurrent agents. We present the syntax of TCLA in Table 1, where P denotes a generic process, C a sequence of procedure declarations (or clauses), A a generic agent and E a generic guarded agent.

Table 1
TCLA syntax.

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

where Arg and S are supposed to be sets of arguments, R a set of attacks, a an argument, $l \in \{in, out, undec\}$, $\sigma \in \{adm, com, stb, prf, gde\}$.

In a process $P = let\ C\ in\ A$, A is the initial agent to be executed in the context of the set of declarations C. A clause defined with C, C corresponds to the concatenation of more procedure declarations. To specify how many times the execution can be repeated, guarded agents (i.e., the *check*, *c-test* and *s-test* operations) are endowed with a timeout $t \in \mathbb{N} \cup \{\infty\}$ stating after how many cycles of the global clock the operation will expire and terminate with failure. Time passes between all concurrent

[1] The set U is not present in the original definition by Dung, and we introduce it for our convenience to distinguish all possible arguments from the adopted ones.

agents on the same global clock via a timeout environment T, as specified below. Let $\mathcal{I}$ be a set of (timeout) identifiers. A timeout environment is a partial mapping $T : \mathcal{I} \rightharpoonup \mathbb{N} \cup \{\infty\}$ such that the set $dom(T) = \{I \in \mathcal{I} \mid T(I) \in \mathbb{N} \cup \{\infty\}\}$ (domain of T) is finite. T_0 is the empty timeout environment ($dom(T_0) = \emptyset$).

We introduce some utility functions as we need to manipulate T, for instance, to insert new timeouts or update a timer. If T_1 and T_2 are timeout environments such that for each $I \in dom(T_1) \cap dom(T_2)$ we have that $T_1(I) = T_2(I)$, then $T_1 \cup T_2$ is the timeout environment such that $dom(T_1 \cup T_2) = dom(T_1) \cup dom(T_2)$ and for each $I \in dom(T_1 \cup T_2)$

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

We denote by $T[\bar{I} : \bar{t}]$ an update of T, with a possibly enlarged domain, namely

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

The passing of time is marked by decreasing timers in a timeout environment T. At each step of the execution, all timers are decreased according to the function $dec(T)$ such that $dom(dec(T)) = dom(T)$ and

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

Below, we outline the functioning of all TCLA operators, focusing in particular on the management of the timeouts and the composition of concurrent agents.

Notation 1. *Let $F = \langle Arg, R \rangle$ be an AF and S a set of arguments. We define:*

- $R_{|S} = \{(a, b) \in R \mid a \in S \text{ or } b \in S\}$;
- $R_{\|S} = \{(a, b) \in R \mid a \in S \text{ and } b \in S\}$;
- $F \downarrow S = \langle Arg \cap S', R_{|S} \rangle$ *where* $S' = S \cup \{b \mid (b, c) \in R_{|S} \text{ or } (c, b) \in R_{|S}\}$;
- $F \uparrow S = \langle Arg \setminus S, R \setminus R_{|S} \rangle$.

The operational model of TCLA processes can be formally described by a transition system $T = (Conf, \longrightarrow)$, where we assume that each transition step takes exactly one time unit. Configurations (in) *Conf* are triples, each triple consisting in a process P, an abstract argumentation framework AF representing a knowledge base and a timeout environment T. The transition relation $\longrightarrow \subseteq Conf \times Conf$ is the least relation satisfying the rules in Tables 2 - 8, and it characterises the (temporal) evolution of the system. So, $\langle A, F, T \rangle \longrightarrow \langle A', F', T' \rangle$ means that, if at time t we have the process A, the framework F and the time environment T, then at time $t+1$ we have the process A', the framework F' and the time environment T'. In the following, we will usually write a *tcla* process $P = C.A$ as the corresponding agent A, omitting C when not required by the context.

Rules **Ad** and **Re** of Table 2 modify the store by adding and removing, respectively, arguments and attacks. Attacks can only be added between arguments that will be in the store after the execution of the add operation. On the other hand, when an argument is removed, also the attacks involving that argument are removed.

Table 2
Add and remove semantics.

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

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

Table 3 presents the semantics for the check operation. When a rule is executed, all timers in the timeout environment T decrease after the possible introduction of new timeouts. Initially, rules **Ch1**

and **Ch2** require inserting a new timeout in T. If not successful, the check repeats using the existing timeout. Rule **Ch2** adds a new timeout to T if conditions fail and the timer has not expired, executing a subsequent check. Rule **Ch3** progresses the execution if the guard succeeds and the existing timer has not expired. Rule **Ch4** repeats a failed check without adding new timeouts. If a timer hits 0, the agent fails per rules **Ch5** and **Ch6**.

Table 3
Check semantics. In the rules, $F = \langle Arg, R \rangle$.

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ t > 0}{\langle \mathit{check}_t(Arg', R') \rightarrow A, F, T \rangle \longrightarrow \langle A, F, \mathit{dec}(T) \rangle} \quad \textbf{Ch1}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ t > 0}{\langle \mathit{check}_t(Arg', R') \rightarrow A, F, T \rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \rightarrow A, F, \mathit{dec}(T[I:t]) \rangle} \quad \textbf{Ch2}$$

where I is a fresh timeout identifier

$$\frac{Arg' \subseteq Arg \wedge R' \subseteq R,\ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T \rangle \longrightarrow \langle A, F, \mathit{dec}(T) \rangle} \quad \textbf{Ch3}$$

$$\frac{\neg(Arg' \subseteq Arg \wedge R' \subseteq R),\ T(I) > 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T \rangle \longrightarrow \langle \mathit{check}_I(Arg', R') \rightarrow A, F, \mathit{dec}(T) \rangle} \quad \textbf{Ch4}$$

$$\langle \mathit{check}_0(Arg', R') \rightarrow A, F, T \rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T) \rangle \quad \textbf{Ch5}$$

$$\frac{T(I) = 0}{\langle \mathit{check}_I(Arg', R') \rightarrow A, F, T \rangle \longrightarrow \langle \mathit{failure}, F, \mathit{dec}(T) \rangle} \quad \textbf{Ch6}$$

Credulous and sceptical test operators validate the acceptability of arguments in the store. Their rules follow the same idea as those for the check operation, with the only exception of the condition to satisfy. The credulous test checks if at least one labelling L in $\mathcal{L}^F_\sigma$ assigns label l to argument a ($\exists L \in \mathcal{L}^F_\sigma \mid L(a) = l$ for $\mathit{c\text{-}test}_I(a, l, \sigma)$). The sceptical test requires all labellings to assign the same label l to a ($\forall L \in \mathcal{L}^F_\sigma . L(a) = l$ for $\mathit{s\text{-}test}_I(a, l, \sigma)$).

Tables 4 and 5 detail rules for nondeterminism and if-then-else constructs. $\mathcal{E}$ represents guarded agents, with $\mathcal{E}_f$ comprising those with timers set to zero, ensuring failure. Rule **ND1** enables two non-failing expressions E_1 and E_2 to simultaneously transition into guarded expressions E_1' and E_2', producing $E_1' + E_2'$. Rule **ND2** progresses with E_1 if it becomes non-guarded. Rule **ND3** discards any failing agent, continuing with the other.

Table 4
Nondeterminism semantics.

$$\frac{\langle E_1, F, T \rangle \longrightarrow \langle E_1', F, T_1 \rangle,\ \langle E_2, F, T \rangle \longrightarrow \langle E_2', F, T_2 \rangle,\ E_1, E_2 \notin \mathcal{E}_f,\ E_1', E_2' \in \mathcal{E}}{\langle E_1 + E_2, F, T \rangle \longrightarrow \langle E_1' + E_2', F, T_1 \cup T_2 \rangle} \quad \textbf{ND1}$$

$$\frac{\langle E_1, F, T \rangle \longrightarrow \langle A_1, F, T' \rangle,\ E_1 \notin \mathcal{E}_f,\ A_1 \notin \mathcal{E}}{\langle E_1 + E_2, F, T \rangle \longrightarrow \langle A_1, F, T' \rangle} \quad \textbf{ND2}$$

$$\frac{E_1 \in \mathcal{E}_f,\ \langle E_2, F, T \rangle \longrightarrow \langle A_2, F, T' \rangle}{\langle E_1 + E_2, F, T \rangle \longrightarrow \langle A_2, F, T' \rangle} \quad \textbf{ND3}$$

The operator in Table 5 implements a non-commutative if-then-else: if $E_1 +_P E_2$ sees E_1 transition to E_1', execution moves to $E_1' +_P E_2$ (rule **If1**). If E_1 becomes a non-guarded agent, E_2 is discarded (rule **If2**). If E_1 fails, execution shifts to E_2 (rule **If3**).

Table 5
If-then-else semantics.

$$\frac{\langle E_1, F, T\rangle \longrightarrow \langle E_1', F, T'\rangle,\ E_1 \notin \mathcal{E}_f,\ E_1' \in \mathcal{E}}{\langle E_1 +_P E_2, F, T\rangle \longrightarrow \langle E_1' +_P E_2, F, T'\rangle} \quad \textbf{If1}$$

$$\frac{\langle E_1, F, T\rangle \longrightarrow \langle A_1, F, T'\rangle,\ E_1 \notin \mathcal{E}_f,\ A_1 \notin \mathcal{E}}{\langle E_1 +_P E_2, F, T\rangle \longrightarrow \langle A_1, F, T'\rangle} \quad \textbf{If2}$$

$$\frac{E_1 \in \mathcal{E}_f, \langle E_2, F, T\rangle \longrightarrow \langle A_2, F, T'\rangle}{\langle E_1 +_P E_2, F, T\rangle \longrightarrow \langle A_2, F, T'\rangle} \quad \textbf{If3}$$

The procedure call in Table 6 takes a single parameter x that can be an argument, a label (*in*, *out*, *undec*), a semantics σ, or a time instant t. This setup can be modified to include more parameters or to handle parameterless procedures. Execution involves replacing instances of p with agent A as defined in the procedure declaration within context C.

Table 6
Procedure call semantics.

$$\langle p(y), F, T\rangle \longrightarrow \langle A[y/x], F, \mathit{dec}(T)\rangle \text{ with } p(x) :: A \text{ and } x \in \{a, l, \sigma, t\} \quad \textbf{PC}$$

Rule **TC** in Table 7 depicts the true concurrency operator, where concurrent agents succeed only if all components succeed. We use $*(F, F', F'') := (F' \cap F'') \cup ((F' \cup F'') \setminus F)$ to handle concurrent additions and removals of arguments: if an argument a is simultaneously added and removed, its final status depends on its initial presence: added if absent, removed if present.

Table 7
True concurrency semantics.

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

In Tables 4 and 7, we have omitted the symmetric rules for the choice operator $+$ and the parallel composition $\parallel$. Indeed, $+$ is commutative, so $E_1 + E_2$ produces the same result as $E_2 + E_1$. The same is true for $\parallel$. Moreover, $success$ and $failure$ are the identity and the absorbing elements, respectively, under the parallel composition $\parallel$.

The rule in Table 8 defines the agent *new* S *in* A such that $S \subseteq Arg$ is local to A, concealing S from the external argumentation framework (AF) and vice versa. This results in agent A operating within $(AF \uparrow S) \cup AF_{loc}$, where AF_{loc} holds local information on S. The syntax extends to *new* S *in* $A^{AF_{loc}}$, initiating with an empty local AF_{loc}. In this setup, *new* S *in* $success$ and *new* S *in* $failure$ return the agents $success$ and $failure$ respectively.

Table 8
Locality semantics.

$$\frac{\langle A, (AF \uparrow S) \cup AF_{loc}, T\rangle \longrightarrow \langle \mathsf{B}, AF', T'\rangle}{\langle \textit{new } S \textit{ in } A^{AF_{loc}}, AF, T\rangle \longrightarrow \langle \textit{new } S \textit{ in } B^{AF' \downarrow S}, (AF' \uparrow S) \cup (AF'' \downarrow S), T'\rangle} \quad \textbf{Loc}$$
$$\text{where } AF = \langle Arg, R\rangle,\ AF' = \langle Arg', R'\rangle \text{ and } AF'' = \langle Arg, R_{\parallel Arg' \cup S}\rangle$$

In the next section, given $S = \{a_1, \ldots, a_n\} \subseteq Arg$, in order to simplify the presentation of the results, we use $\exists x \in S \mid c\text{-}test_t(x, l, \sigma) \to A$ as a shorthand of $(c\text{-}test_t(a_1, l, \sigma) \to A) + (c\text{-}test_t(a_2, l, \sigma) \to A) + \cdots + (c\text{-}test_t(a_n, l, \sigma) \to A)$.

A possible use case for TCLA can be identified in modelling Multi-Agent Decision Making with Privacy Preserved (DMPP) in which agents need to communicate with other agents to make socially

optimal decisions but, at the same time, have some private information that they do not want to share. This problem can be instantiated as done in other works like [11] as a debate in a multi-agent environment where argumentation techniques are exploited for arriving at socially optimal outcomes by only disclosing the "necessary" and "disclosable (public)" information. A DMPP problem is formalised as follows.

Definition 3. *[1] A Multi-Agent Decision Making with Privacy Preserved problem (DMPP) is a triple* $\langle Ag, Act, Sol \rangle$*, where*

- $Ag = \{Agent_1 \ldots, Agent_N\}$ *is a finite set of agents;*
- $Act = \{a_1, \ldots, a_M\}$ *is a finite set of available actions for the agents;*
- $SP = \{\langle Agent_1 : a^1, \ldots, Agent_N : a^N \rangle \mid \{a^1, \ldots, a^N\} \subseteq Act\}$ *is the set of all strategic profiles, namely the set of all the possible tuples of actions (one for each agent);*
- $Sol \subseteq SP$ *is the set of acceptable solutions to the problem.*

3. Preserving Privacy in Multi-Agent Decision with TCLA

In this section, we focus on applying TCLA to model DMPP problems. This involves scenarios where agents must communicate and make decisions while safeguarding private information. We formalise the DMPP problem and demonstrate how tcla can facilitate socially optimal decisions by disclosing only necessary and public information.

Example 1. To demonstrate our formalisation and translation, we analyze an e-procurement scenario involving a buyer's acquisition of combined services and products from two hardware and software suppliers. This particular e-procurement scenario was selected for evaluation due to its suitability, including the use of distributed agents representing buyers and suppliers. Each agent represents a user in the scenario.

We consider a specific case where a buyer (Charlie) looks for an e-ordering system which consists of hardware systems (Hw) and software packages (Sw). The agents providing Hw and Sw are Alice and Bob. Charlie needs to purchase products or services from both operators, Alice and Bob, for business reasons. Furthermore, Charlie will ask her first since he is Alice's friend. If possible, Charlie would prefer to buy the hardware from Alice (we represent this statement with C:$\underline{\langle \text{Hw}_{Ali},\ \text{Sw}_{Bob} \rangle}$) because typically, Alice's selling price for hardware is lower than Bob's. Moreover, Charlie does not want to buy hardware from Bob because, in the last sale, Bob delivered the wrong goods (denoted by *Wrong*).

Alice and Bob would prefer to provide the software (we represent this statement with A:$\underline{\text{Sw}}$ and B:$\underline{\text{Sw}}$). Alice would not want to provide the hardware because she does not have the hardware components in stock at the moment, but this is confidential information (denoted by **NoStock**). However, she can make a deal with the suppliers to make a special delivery (*SpecDel*). On the other hand, Bob is having trouble providing the software quickly because, at the moment, his company lacks skilled technicians for installation; this is private information (denoted by **NoTec**). However, two excellent technicians will be taking over in the next few days (*NewTwo*).

Charlie is aware that newspapers have recently reported that, due to the blockage of the Suez Canal, it is not economically feasible to obtain one-off deliveries from suppliers (*Suez*). Moreover, Charlie needs to purchase the service urgently and might not be willing to wait for the arrival of the two new technicians hired by Bob (*Urgency*). Finally, Bob has a new, very well-organized office that handles hardware sales (*NewOffice*). All these beliefs can be modelled as in the AFs of Figure 1.

Example 2. In Example 1, $N = 3$, since there are three agents: C (Charlie), A (Alice) and B (Bob), $M = 4$ since there are four actions $\underline{\text{Sw}}$ and $\underline{\text{Hw}}$ ("provide the software" and "provide the hardware", respectively), and $\underline{\langle \text{Hw}_{Ali}, \text{Sw}_{Bob} \rangle}$ and $\underline{\langle \text{Sw}_{Ali}, \text{Hw}_{Bob} \rangle}$ ("buy the hardware from Alice and the software from Bob" or vice versa, respectively), and two acceptable solutions to the problem, namely $Sol = \{\langle C : \underline{\langle \text{Hw}_{Ali}, \text{Sw}_{Bob} \rangle}, \text{A:}\underline{\text{Hw}}, \text{B:}\underline{\text{Sw}} \rangle,\ \langle C : \underline{\langle \text{Sw}_{Ali}, \text{Hw}_{Bob} \rangle}, \text{A:}\underline{\text{Sw}}, \text{B:}\underline{\text{Hw}} \rangle, \}$. Note that for the problem specification, the remaining strategic profiles are not acceptable solutions.

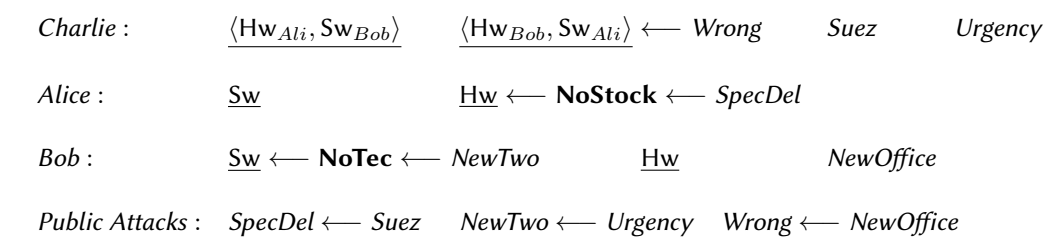

Figure 1: AFs representing Charlie's, Alice's and Bob's beliefs and observation [11].

In the following, as in [11], we distinguish two categories of arguments: practical and non-practical (epistemic). Practical arguments recommend actions to agents. For simplicity, for each action and agent, we allow for one and only one practical argument to recommend this action to the agent. Let a^i_j denote the (one and only) practical argument recommending action a_j to $Agent_i$,[2] and denote the set of all practical arguments belonging to $Agent_i$ by $pPra_i$ (p is short for 'private', and Pra stands for 'Practical'), i.e. $pPra_i = \{a^i_{i_1}, \ldots, a^i_{i_{M_i}}\}$, where $\{a_{i_1}, \ldots, a_{i_{M_i}}\} \subseteq Act$. Practical arguments are private since agents directly inform other agents about their action choices in the protocol of [11].

We distinguish two subcategories for non-practical arguments: public (disclosable) and private arguments, which an agent is willing and unwilling to disclose to other agents, respectively. We denote $Agent_i$'s public and private non-practical argument sets by $PArg_i$ and $pArg_i$, respectively and by $PArg = \cup_{i\in\{1,\ldots,n\}} PArg_i$ the set of all public arguments. Moreover, analogously to [11], we assume that each agent has a complete preorder $\leq_i$ expressing a preference on all available actions, and we model the $Agent_i$'s preference as a sequence of actions $L_i = a'_1 \cdots a'_{M_i}$ such that $pPra_i = \{a'_1, \ldots, a'_{M_i}\}$ and if $k \leq h$ then $a'_h \leq_i a'_k$ (the action with the smaller index is the preferred one). Finally, we can define a preorder $\leq$ on Sol as follows: given $s = \langle Agent_1 : a^1, \ldots, Agent_N : a^N\rangle$ and $s' = \langle Agent_1 : b^1, \ldots, Agent_N : b^N\rangle$ we say that $s \leq s'$ if and only if for each $i \in \{1, \ldots, N\}$ $a^i \leq_i b^i$.

We denote $Agent_i$'s private arguments set by $Arg_i = pPra_i \cup pArg_i$. We impose that $pPra_i$ and $pArg_i$ are disjoint and that Arg_i is finite. In Figure 1, practical arguments are underlined, private arguments are in boldface, and public arguments are in italics.

Since a practical argument concerns the actions that the agents can perform, while a non-practical argument justifies beliefs and observations on which practical arguments are built, in line with [12], practical arguments are not allowed to attack non-practical arguments. As for attacks between non-practical arguments, since private arguments usually represent agents' private beliefs/concerns, whereas public arguments usually represent observations/facts (see Example 1), we assume that private arguments do not attack public arguments. Finally, each private non-practical argument is attacked by some public arguments. This guarantees that, when negotiating with other agents, other agents can 'indirectly' defend or attack each private argument by (directly) attacking or defending some public arguments attacking it.

Now, we discuss attacks between arguments of different agents and attacks between agents' actions and acceptable solutions. Following [11], since an agent cannot know another agent's private arguments, we restrict attacks between agents' arguments only to the public arguments, i.e. given a practical or private argument of $Agent_i$, this argument cannot attack or be attacked by another agent's practical or private arguments. In addition, we assume that agents have a consensus on the attack relation between (public) arguments. Note that with this assumption, we do not mean that an agent knows another agent's public (disclosable) arguments a priori; instead, we mean that if two public arguments are presented in front of an agent, this agent can decide the attack relation between them, and all other agents agree on this attack relation.

Moreover, differently from [11], we model agents' actions and acceptable solutions, which are common and visible to all agents, as public arguments. Each agent $Agent_i$ can decide on its own actions and

[2] In the examples, by an abuse of notation, we use the same name both for the actions and for the arguments that recommend them.

not on those of the other agents. Therefore, the actions of one agent are disjointed from those of the others and are of the form $Agent_i : a_j$, where $a_j \in Act$. We denote by $Actions = \{Agent_i : a \mid i \in \{1, \ldots, N\}, a \in Act_i\}$.

Finally we define the attacks between agents' actions and acceptable solutions as follows

$$NoG = \{(Agent_i : a, \langle Agent_1 : b_1, \ldots, Agent_N : b_N\rangle \mid Agent_i : a \in Actions \\ \langle Agent_1 : b_1, \ldots, Agent_N : b_N\rangle \in Sol, \text{ and } a \neq b_i\}$$

Definition 4. *The attack relation between public arguments is denoted by* $Att^* \subseteq (PArg \times PArg) \cup NoG$.

Example 3. In Example 1 we have that

$$\begin{aligned} NoG = \{ \quad & (C : \underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle}, \langle C : \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}, A : \underline{\mathrm{Sw}}, B : \underline{\mathrm{Hw}}\rangle), \\ & (C : \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}, \langle C : \underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle}, A : \underline{\mathrm{Hw}}, B : \underline{\mathrm{Sw}}\rangle), \\ & (A : \underline{\mathrm{Hw}}, \langle C : \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}, A : \underline{\mathrm{Sw}}, B : \underline{\mathrm{Hw}}\rangle), \\ & (A : \underline{\mathrm{Sw}}, \langle C : \underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle}, A : \underline{\mathrm{Hw}}, B : \underline{\mathrm{Sw}}\rangle), \\ & (B : \underline{\mathrm{Hw}}, \langle C : \underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle}, A : \underline{\mathrm{Hw}}, B : \underline{\mathrm{Sw}}\rangle), \\ & (B : \underline{\mathrm{Sw}}, \langle C : \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}, A : \underline{\mathrm{Sw}}, B : \underline{\mathrm{Hw}}\rangle)\ \}. \end{aligned}$$

In this formalisation, $\underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}$ and $\underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle}$ are private argument, allowing Charlie to internally decide his actions without revealing his reasoning. In contrast, $C : \underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}$ and $C : \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle}$, are public arguments that communicate Charlie's decisions to Alice and Bob. Alice and Bob analogously use private and public arguments to manage their decisions and communications.

Given a DMPP, as detailed in [11], we define two specific argumentation frameworks for each agent: the perfect-view AFv, representing each agent's ideal perspective, and the private internal AFp, containing confidential information specific to that agent. Additionally, we also define the global public initial AFd, which is common and visible to all agents in the DMPP.

Definition 5. *Let* $\langle Ag, Act, Sol\rangle$ *a DMPP problem. We define*

- $Agent_i$*'s perfect-view internal AF as* $AFv_i = \langle Arg_i \cup PArg,\ Att_i \cup Att^*\rangle$*;*
- $Agent_i$*'s private internal AF as* $AFp_i = AFv_i \downarrow (pPra_i \cup pArg_i)$ *and*
- *the public argumentation framework* $AFd \subseteq \langle PArg \cup Sol, Att^*\rangle$

Note that if $AFp_i = \langle Arg_i'; Att_i\rangle$, then $Arg_i' \subseteq Arg_i \cup PArg$ and $Att_i \subseteq (pArg_i \cup PArg) \times (pPra_i \times pArg_i)$.

Example 4. Charlie, Alice and Bob's private internal AFs are the following.

$$\begin{aligned} AFp_{Charlie} = & \langle \{\underline{\langle \mathrm{Hw}_{Ali}, \mathrm{Sw}_{Bob}\rangle}, \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle}, \mathit{Wrong}\},\ \{(\mathit{Wrong}, \underline{\langle \mathrm{Hw}_{Bob}, \mathrm{Sw}_{Ali}\rangle})\}\rangle \\ AFp_{Alice} = & \langle \{\underline{\mathrm{SwHw}}, \mathbf{NoStock}, \mathit{SpecDel}\},\ \{(\mathbf{NoStock}, \underline{\mathrm{Hw}})(\mathit{SpecDel}, \mathbf{NoStock})\}\rangle \\ AFp_{Bob} = & \langle \{\underline{\mathrm{Sw}}, \mathbf{NoTec}, \mathit{NewTwo}\},\ \{(\mathbf{NoTec}, \underline{\mathrm{Sw}}), (\mathit{NewTwo}, \mathbf{NoTec})\}\rangle \end{aligned}$$

Moreover, we also know that Charlie's preference is the sequence of actions $\underline{\langle \mathrm{Hw}_{Ali},\ \mathrm{Sw}_{Bob}\rangle} \cdot \underline{\langle \mathrm{Hw}_{Bob},\ \mathrm{Sw}_{Ali}\rangle}$, while Alice and Bob's preference is the sequence of actions $\underline{\mathrm{Sw}} \cdot \underline{\mathrm{Hw}}$.

We want the solutions of the form $s = \langle Agent_1 : a^1, \ldots, Agent_N : a^N\rangle$ obtained by our protocol to meet the following requirements: i) feasible, i.e. if a strategy profile assigns action a_j to $Agent_i$, then a_j should be 'doable' for $Agent_i$; ii) acceptable (i.e. is a solution of the problem) and socially optimal with respect to agents' preferences.

Definition 6. *Let* $\langle Ag, Act, Sol\rangle$ *a DMPP problem and let* $s = \langle Agent_1 : a^1, \ldots, Agent_N : a^N\rangle \in Sol$.

- $a_j \in Act$ *is (globally) feasible for the agent* $Agent_i$ *if argument* $a_j^i \in pPra_i$ *(the practical argument that recommends* a_j *to* $Agent_i$*) is admissible in the* $Agent_i$*'s perfect view,* AFv_i*;*
- s *is feasible if and only if, for each* $i \in \{1, \ldots, N\}$ *the action* a^i *is globally feasible for* $Agent_i$*;*

- *s is socially optimal if and only if s is feasible and there is no $s' \in Sol$ such that s' is feasible and $s' < s$.*

Now, let us consider a communication protocol inspired by Chronological synchronous backtracking (SBT), one of the simplest while most fundamental algorithms for distributed constraint satisfaction problems, which requires a static ordering of agents. Following this, agents try to make a social decision. Agents pass a token among them; the agent holding the token checks whether it can extend the current partial solution by making a move to obtain a new partial acceptable solution. In this case, he sends the token to the next agent; otherwise, he sends a message ngd (short for "not good") to the previous agent. The communication terminates either because all the agents agree on a solution or because every (partial) solution has been discarded. In the former case, the agents found a solution to the problem, while in the latter case, the problem is unsatisfiable. SBT is guaranteed to terminate and is sound and complete (i.e., it terminates only with correct answers and for all problems).

We can write a TCLA program emulating a DMPP $\langle Ag, Act, Sol\rangle$, where $Ag = \{Agent_1 \ldots, Agent_N\}$, using N TCLA agents in parallel. Without loss of generality, we assume that agents with smaller index numbers are higher in the static agent ordering required by SBT; therefore, $Agent_1$ activates the protocol.

Definition 7 (Translation). *Let $\langle Ag, Act, Sol\rangle$ a DMPP problem, such that $Ag = \{Agent_1 \ldots, Agent_N\}$ and for $i \in \{1, \ldots, n\}$, L_i is the $Agent_i$'s preference. The translation of $\langle Ag, Act, Sol\rangle$ is $\mathcal{T}(\langle Ag, Act, Sol\rangle) = \mathcal{T}(Agent_1) \| \cdots \| \mathcal{T}(Agent_N)$, where*

- *$\mathcal{T}(Agent_1) =$ new $(pPra_1 \cup pArg_1)$ in $T_1(L_1)^{AFp_1}$ and*
- *for $1 < i \leq N$, $\mathcal{T}(Agent_i) =$ new $(pPra_i \cup pArg_i)$ in $p_i^{AFp_i}$ such that p_i is the (without parameters) procedure $p_i ::$ check$_\infty(\{tok_i\}, \emptyset) \rightarrow$ rmv$(\{tok_i\}, \emptyset) \rightarrow T_i(L_i) \in P$*

such that

$$T_i(\varepsilon) = \begin{cases} failure & \text{if } i = 1 \\ \text{add}(\{ngd_{i-1}\}, \emptyset) \rightarrow p_i & \text{if } 1 < i \leq N \end{cases}$$

- $T_i(a \cdot L') =$ c-test$_1(a,$ in$, adm) \rightarrow$ (add$(\{Agent_i : a\}, NoG_{|\{Agent_i:a\}}) \rightarrow$
  ($\exists s \in Sol \mid$ c-test$_1(s,$ in$, adm) \rightarrow R_i(a \cdot L')$
  $+_P$
  $true \rightarrow$ rmv$(\{Agent_i : a\}, \emptyset) \rightarrow T_i(L')$)
  $+_P$
  $true \rightarrow T_i(L')$

  and

$$R_i(a \cdot L') = \begin{cases} \text{add}(\{tok_{i+1}\}, \emptyset) \rightarrow & \text{if } 1 \leq i < N \\ \quad (\ \text{check}_\infty(\{gd\}, \emptyset) \rightarrow success & \\ \quad\ + & \\ \quad\ \text{check}_\infty(\{ngd_i\}, \emptyset) \rightarrow & \\ \quad\quad \text{rmv}(\{ngd_i, Agent_i : a\}, \emptyset) \rightarrow T_i(L'))\ ) & \\ \\ \text{add}(\{gd\}, \emptyset) \rightarrow success & \text{if } i = N \end{cases}$$

The translation of a DMPP problem is presented in Definition 7. First of all, note that the agents in the translation have a non-empty local argumentation framework. We can assume that initially, an agent builds its local framework by using an *add* step. The computation starts in the initial public argumentation framework AFd (Definition 5) and the empty timeout T_0. The computation is started by $Agent_1$, which checks for a (globally) feasible action a. When an $Agent_i$ receives the token tok_i, it iterates over all actions starting from the agent's most preferred one(s). The step add$(\{Agent_i : a\}, NoG_{|\{Agent_i:a\}})$ stores $Agent_i$'s action choice in the public shared argumentation framework. For

the current action a, the agent checks whether it is (globally) feasible ($c\text{-}test_i(a, \textit{in}, adm)$). In this case, it adds the argument $Agent_i : a$ to the global framework and checks the consistency of $Agent_i : a$ wrt. the current partial solution and the acceptable solutions, namely the partial consistency of $Agent_i : a$ ($\exists s \in Sol \mid c\text{-}test_1(s, \textit{in}, adm)$). If the argument $Agent_i : a$ is not consistent, $Agent_i$ removes it from the global framework. After obtaining a partially consistent and feasible action a, $Agent_i$ either adds tok_{i+1} to the public framework if it is not the last ($1 \leq i < N$), or (if $i = N$) it adds gd to the public framework to communicate to other agents that an acceptable solution has been computed and then terminates with success.

However, when $i \neq 1$ (i.e. the current agent is not the first agent), if $Agent_i$ fails to find any partially consistent and feasible action, it adds the argument ngd_{i-1} to the public framework (for the previous agent) to backtrack. Finally, if $Agent_1$ cannot find any action which can be extended to find a solution, the computation terminates with failure.

Example 5. Let us consider the DMPP problem of the Example 1. For the agent $Charlie$ we have $\mathcal{T}(\textit{Charlie}) =$
new $(\{\underline{\langle \text{Hw}_{Ali}, \text{Sw}_{Bob}\rangle}, \underline{\langle \text{Hw}_{Bob}, \text{Sw}_{Ali}\rangle}\})$ *in* $T_C(\underline{\langle \text{Hw}_{Ali}, \text{Sw}_{Bob}\rangle}, \underline{\langle \text{Hw}_{Bob}, \text{Sw}_{Ali}\rangle})^{AFp_{\textit{Charlie}}}$, where $T_C(\underline{\langle \text{Hw}_{Ali}, \text{Sw}_{Bob}\rangle}, \underline{\langle \text{Hw}_{Bob}, \text{Sw}_{Ali}\rangle})$ is defined in Table 9. Analogously for the agents $Alice$ and Bob. The computation of $\mathcal{T}(\textit{Charlie}) \parallel \mathcal{T}(\textit{Alice}) \parallel \mathcal{T}(\textit{Bob})$ starts in an initial public shared argumentation framework

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

In Example 5, we translate the agent Charlie of Example 1 to coordinate the three agents' actions. *Charlie* is ranked higher in the static agent ordering, so he obtains the token first.

Charlie checks whether the action $\underline{\langle \text{Hw}_{Ali}, \text{Sw}_{Bob}\rangle}$ ('buy hardware from Alice and software from Bob') is (globally) feasible, and in this case, he stores his most preferred action choice and the corresponding attack in NoG in the shared AF. Then he checks the consistency of $C : \underline{\langle \text{Hw}_{Ali},\ \text{Sw}_{Bob}\rangle}$ wrt. the acceptable solutions, namely he checks if $\exists s \in Sol \mid c\text{-}test_1(s, \textit{in}, adm)$. If the argument is not consistent, Charlie removes it from the global framework. If vice versa, the argument is consistent, Charlie adds tok_A (for Alice) to the global framework and then waits until either argument gd or argument ngd_C is checked for presence in the global framework. If the gd token is present in the global framework, Charlie terminates successfully.

In the other case, Charlie removes the argument $C : \underline{\langle \text{Hw}_{Ali},\ \text{Sw}_{Bob}\rangle}$ from the global framework and he tries again to coordinate with Alice and Bob, adding the argument $\langle C : \underline{\langle \text{Hw}_{Bob},\ \text{Sw}_{Ali}\rangle}$ ('buy the hardware from Bob and the software from Alice'), with its corresponding attack in NoG, to the global framework. Finally, if neither $\langle C : \underline{\langle \text{Hw}_{Ali},\ \text{Sw}_{Bob}\rangle}$ nor $\langle C : \underline{\langle \text{Hw}_{Bob},\ \text{Sw}_{Ali}\rangle}$ can be extended to find a solution, Charlie terminates with failure and causes the termination with failure of the entire translation of the initial DMPP problem.

Since all moves other than the first in our disputes are the same as in standard TPI-disputes, and there are finitely many arguments presented in the first move, we have:

Proposition 1. Let $\mathcal{T}$ be a translation of a DMPP $\langle Ag, Act, Sol\rangle$, AFd is as in Definition 5 and T_0 is the empty timeout environment. Then the computation starting from $\langle \mathcal{T}, AFd, T_0\rangle$ is guaranteed to terminate. Moreover there exist an AF and a timer T such that either $\langle \mathcal{T}, AFd, T_0\rangle \longrightarrow \langle failure, AF, T\rangle$ or $\langle \mathcal{T}, AFd, T_0\rangle \longrightarrow \langle success, AF, T\rangle$.

Theorem 1 (Correctness and Completness). Let $\mathcal{T}$ be a translation of a DMPP $\langle Ag, Act, Sol\rangle$, AFd as specified in Definition 5 and T_0 the empty timeout environment. The following holds:

1. $\langle \mathcal{T}, AFd, T_0\rangle \longrightarrow^* \langle success, AF, T\rangle$ if and only if $\exists s \in Sol$ such that s is feasible for $\langle Ag, Act, Sol\rangle$. Moreover if $\langle \mathcal{T}, AFd, T_0\rangle \longrightarrow^* \langle success, AF, T\rangle$ then there is a unique $s' \in Sol$ such that s' is feasible in AF and so s' is socially optimal for $\langle Ag, Act, Sol\rangle$.

Table 9
Definition of $T_C(\underline{\langle \mathsf{Hw}_{Ali}, \mathsf{Sw}_{Bob}\rangle}, \underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle})$ for agent $Charlie$ used in Example 5.

$$
\begin{array}{l}
T_C(\underline{\langle \mathsf{Hw}_{Ali}, \mathsf{Sw}_{Bob}\rangle}, \underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle}) = \\
\quad c\text{-}test_1(\underline{\langle \mathsf{Hw}_{Ali}, \mathsf{Sw}_{Bob}\rangle}, \mathit{in}, adm) \rightarrow \\
\qquad (\ \mathit{add}(\{C : \underline{\langle \mathsf{Hw}_{Ali}, \mathsf{Sw}_{Bob}\rangle}\}, \\
\qquad\quad \{(C : \underline{\langle \mathsf{Hw}_{Ali},\ \mathsf{Sw}_{Bob}\rangle}, \langle C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}, A : \underline{\mathsf{Sw}}, B : \underline{\mathsf{Hw}}\rangle)\}) \rightarrow \\
\qquad\quad (\ (\ c\text{-}test_1(\langle C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}, A : \underline{\mathsf{Sw}}, B : \underline{\mathsf{Hw}}\rangle, \mathit{in}, adm) \vee \\
\qquad\qquad c\text{-}test_1(\langle C : \underline{\langle \mathsf{Hw}_{Ali},\ \mathsf{Sw}_{Bob}\rangle}, A : \underline{\mathsf{Hw}}, B : \underline{\mathsf{Sw}}\rangle, \mathit{in}, adm)\ ) \rightarrow \\
\qquad\qquad\quad (\ \mathit{add}(\{tok_A\}, \emptyset) \rightarrow \\
\qquad\qquad\qquad (\ \mathit{check}_\infty(\{gd\}, \emptyset) \rightarrow success\ + \\
\qquad\qquad\qquad\ \mathit{check}_\infty(\{ngd_C\}, \emptyset) \rightarrow \mathit{rmv}(\{ngd_C,\ C : \underline{\langle \mathsf{Hw}_{Ali}, \mathsf{Sw}_{Bob}\rangle}\}, \emptyset) \rightarrow \\
\qquad\qquad\qquad\qquad\qquad\qquad T_C(\underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle})\ )\ )\ ) \\
\qquad\qquad +_P \\
\qquad\qquad true \rightarrow \mathit{rmv}(\{C : \underline{\langle \mathsf{Hw}_{Ali}, \mathsf{Sw}_{Bob}\rangle}\}, \emptyset) \rightarrow T_C(\underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle}))))\ ) \\
\quad +_P \\
\quad true \rightarrow T_C(\underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle}) \\
\qquad\qquad\qquad\qquad\qquad \text{and} \\
T_C(\underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle}) = \\
\quad c\text{-}test_1(\underline{\langle \mathsf{Hw}_{Bob}, \mathsf{Sw}_{Ali}\rangle}, \mathit{in}, adm) \rightarrow \\
\qquad (\ \mathit{add}(\{C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}\}, \\
\qquad\quad \{(C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}, \langle C : \underline{\langle \mathsf{Hw}_{Ali},\ \mathsf{Sw}_{Bob}\rangle}, A : \underline{\mathsf{Hw}}, B : \underline{\mathsf{Sw}}\rangle)\}) \rightarrow \\
\qquad\quad (\ (\ c\text{-}test_1(\langle C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}, A : \underline{\mathsf{Sw}}, B : \underline{\mathsf{Hw}}\rangle, \mathit{in}, adm) \vee \\
\qquad\qquad c\text{-}test_1(\langle C : \underline{\langle \mathsf{Hw}_{Ali},\ \mathsf{Sw}_{Bob}\rangle}, A : \underline{\mathsf{Hw}}, B : \underline{\mathsf{Sw}}\rangle, \mathit{in}, adm)\ ) \rightarrow \\
\qquad\qquad\quad (\ \mathit{add}(\{tok_A\}, \emptyset) \rightarrow \\
\qquad\qquad\qquad (\ \mathit{check}_\infty(\{gd\}, \emptyset) \rightarrow success\ + \\
\qquad\qquad\qquad\ \mathit{check}_\infty(\{ngd_C\}, \emptyset) \rightarrow \mathit{rmv}(\{ngd_C,\ C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}\}, \emptyset) \rightarrow \\
\qquad\qquad\qquad\qquad\qquad\qquad failure\ )\ )\ ) \\
\qquad\qquad +_P \\
\qquad\qquad true \rightarrow \mathit{rmv}(\{C : \underline{\langle \mathsf{Hw}_{Bob},\ \mathsf{Sw}_{Ali}\rangle}\}, \emptyset) \rightarrow failure\ ) \\
\quad +_P \\
\quad true \rightarrow failure
\end{array}
$$

2. $\langle \mathcal{T}, AFd, T_0\rangle \longrightarrow^* \langle failure, AF, T\rangle$ if and only if $/\exists s \in Sol$ such that s is feasible for $\langle Ag, Act, Sol\rangle$.

4. Conclusion

We have discussed a new version of TCLA (introduced in [1]) featuring local stores, enabling agents to keep important data private while integrating it into their decision-making processes. Consequently, the language is particularly effective at modelling real-life scenarios, as evidenced by our explicit provision of a translation of a general DMPP in TCLA with local stores, which is both correct and complete. Additionally, we have provided a working example demonstrating its use in a privacy-preserving setting for decision-making.

In this paper, we performed a direct translation of the reasoning process from [11] to demonstrate how a decision-making process can be implemented by considering local knowledge. This study, therefore, mainly illustrates how the decision-making process can be implemented in TCLA using the local store. However, it is important to note that this language has several other features that could be used to simplify the construction of such models. Exploring these features to achieve more natural interactions with the native TCLA constructs will be part of our future work. Besides, we plan to develop further illustrative examples to showcase the framework's effectiveness and highlight limitations across various scenarios.

Another goal is to extend TCLA to cover additional aspects that could impact agents' behaviour during argument exchange interactions. For instance, we would like to model real-world applications where agents can coordinate autonomously and concurrently without being bound to a fixed agent ordering. Moreover, we want to endow the agents with a notion of ownership to establish which actions can be performed on the shared arguments. In the current implementation, an autonomous agent could remove arguments added to the shared store by its opponents in order to win a debate. However, this is not an effective solution in practical cases and is also considered an illegal move in dialogue games. Finally, we plan to use the language to model chatbot interactions [13] and explanation activities [14].

Acknowledgments

The authors are member of the INdAM Research group GNCS and of Consorzio CINI. This work has been partially supported by: GNCS-INdAM, CUP_E53C23001670001; GNCS-INdAM, CUP_E53C22001930001; EU PNRR MUR PRIN project EPICA: "Empowering Public Interest Communication with Argumentation" - Next Generation (J53D23007220006); EU PNRR MUR project SERICS - Next Generation (PE00000014); PNRR MIUR project FAIR - Future AI Research (PE00000013), Spoke 9 - Green aware AI; University of Perugia - Fondo Ricerca di Ateneo (2020, 2021, 2022) - Projects BLOCKCHAIN4FOODCHAIN, FICO, AIDMIX, "Civil Safety and Security for Society"; European Union - Next Generation EU NRRP-MUR - Project J97G22000170005 VITALITY: "Innovation, digitalisation and sustainability for the diffused economy in Central Italy"; Piano di Sviluppo e Coesione del Ministero della Salute 2014-2020 - Project I83C22001350001 LIFE: "the itaLian system Wide Frailty nEtwork" (Linea di azione 2.1 "Creazione di una rete nazionale per le malattie ad alto impatto" - Traiettoria 2 "E-Health, diagnostica avanzata, medical devices e mini invasività").

References

[1] S. Bistarelli, M. C. Meo, C. Taticchi, On the role of local arguments in the (timed) concurrent language for argumentation, in: Proceedings of the 7th Workshop on Advances in Argumentation in Artificial Intelligence (AIˆ3 2023) co-located with the 22nd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2023), Rome, Italy, November 9, 2023, volume 3546 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.

[2] A. C. Kakas, P. Moraitis, Adaptive agent negotiation via argumentation, in: 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), ACM, 2006, pp. 384–391.

[3] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, L. Sonenberg, Argumentation-based negotiation, Knowl. Eng. Rev. 18 (2003) 343–375.

[4] K. Atkinson, T. J. M. Bench-Capon, P. McBurney, Multi-agent argumentation for edemocracy, in: Proceedings of the Third European Workshop on Multi-Agent Systems, Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten, 2005, pp. 35–46.

[5] A. Rosenfeld, S. Kraus, Strategical argumentative agent for human persuasion, in: ECAI 2016 - 22nd European Conference on Artificial Intelligence - Including Prestigious Applications of Artificial Intelligence (PAIS 2016), volume 285 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2016, pp. 320–328.

[6] S. Bistarelli, M. C. Meo, C. Taticchi, Timed concurrent language for argumentation with maximum parallelism, J. Log. Comput. 33 (2023) 712–737.

[7] S. Bistarelli, M. C. Meo, C. Taticchi, Timed concurrent language for argumentation: An interleaving approach, in: Practical Aspects of Declarative Languages - 24th International Symposium, PADL 2022, Proceedings, volume 13165 of *LNCS*, Springer, 2022, pp. 101–116.

[8] P. M. Dung, On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games, Artif. Intell. 77 (1995) 321–358.

[9] P. Baroni, M. Caminada, M. Giacomin, An introduction to argumentation semantics, Knowl. Eng. Rev. 26 (2011) 365–410.
[10] M. Caminada, On the issue of reinstatement in argumentation, in: Proc. of JELIA 2006 - 10th European Conference on Logics in Artificial Intelligence, volume 4160 of *LNCS*, Springer, 2006, pp. 111–123.
[11] Y. Gao, F. Toni, H. Wang, F. Xu, Argumentation-based multi-agent decision making with privacy preserved, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, ACM, 2016, pp. 1153–1161.
[12] L. Amgoud, H. Prade, Using arguments for making and explaining decisions, Artif. Intell. 173 (2009) 413–436.
[13] S. Bistarelli, C. Taticchi, F. Santini, A chatbot extended with argumentation, in: Proceedings of the 5th Workshop on Advances in Argumentation in Artificial Intelligence 2021 co-located with the 20th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2021), Milan, Italy, November 29th, 2021, volume 3086 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021.
[14] S. Bistarelli, A. Mancinelli, F. Santini, C. Taticchi, Arg-xai: a tool for explaining machine learning results, in: 34th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2022, IEEE, 2022, pp. 205–212.