Argumentation and Cooperation for Distributed Extended Logic Programs*

Iara de Almeida Móra[†], José Júlio Alferes[‡] and Michael Schroeder [§]

Abstract

Argumentation semantics in extended logic programming has been defined in [3, 10] for a single agent, determining the agents' beliefs by an internal argumentation process. In this paper we extend the argumentation framework to a multi-agent setting including cooperation among agents. We present an algorithm for inference in the framework and an example to show the process of cooperation and argumentation.

Keywords: Argumentation, Extended Logic Programming, Multi-agent system

*Thanks to Luis Moniz Pereira, Wolfgang Nejdl, Daniela Plewe, and Gerd Wagner. The work is financially supported by JNICT project ACROPOLE PBIC/TIT/2519/95, the Europeu project BMFB/JNICT, and the Brazilian CAPES.

[†]DI/FCT, Univ. Nova de Lisboa, 2825 Monte de Caparica, Portugal, idm@di.fct.unl.pt

[‡]DI/FCT and DM, Univ. Évora, Largo dos Colegiais, 7000 Évora, Portugal, jja@dmat.uevora.pt

[§]Institut für Rechnergestützte Wissensverarbeitung, University of Hannover, Lange Laube 3, 30159 Hannover, Germany, schroede@kbs.uni-hannover.de

Argumentation and Cooperation for Distributed Extended Logic Programs

Abstract

Argumentation semantics in extended logic programming has been defined in [3, 10] for a single agent, determining the agents' beliefs by an internal argumentation process. In this paper we extend the argumentation framework to a multi-agent setting including cooperation among agents. We present an algorithm for inference in the framework and an example to show the process of cooperation and argumentation.

Keywords: Argumentation, Extended Logic Programming, Multi-agent system

1 Introduction

In the last 5-10 years researchers have been devoting much work to the semantics of logic programs. Among the different approaches, that usually either emphasize an operational or a declarative view, argumentation semantics turned out to be a very intuitive approach. Rather than defining a semantics in technical terms, argumentation semantics uses the metaphor of argumentation as used in politics, law, discourse, etc, and formalizes a part of it sufficient to give a meaning to extended logic programs. Intuitively, argumentation semantics treats the evaluation of a logic program as an argumentation process, where a goal G holds if all arguments supporting G can no longer be attacked. Thus, logic programming is seen as a discourse involving attacking and counter-attacking arguments.

Argumentation semantics in extended logic programming has been defined in [3, 10] for a single agent, determining an agent's beliefs by an internal argumentation process. While Dung [3] uses argumentation to define a declarative semantics for extended logic programs (ELP), i.e. logic programming with both default negation (not) and explicit or "classical" negation ($\neg$), Prakken and Sartor's work [10] is driven by their application in legal reasoning. Logic programming can be described in terms of two figures: Reductio ad Absurdum- and ground-attack [3] or, equivalently, rebut and undercut [10]. The former classifies an argument that leads to a contradiction under the current believes and arguments, and the latter an argument that falsifies the premise of one of the current arguments.

In the next section we present basic argumentation definitions and a proof proposal for an argument that coincides with the extended well founded semantics ($WFSX$), a semantics for extended logic programs. Then we extend the argumentation framework for a multi-agent setting [11] including cooperation. Finally, we present an algorithm for inference in the framework and an example to show the process of cooperation and argumentation.

2 Argumentation

The following definitions for argumentation are based on Prakken and Sartor's proposal [10]. They propose an argument as a sequence of grounded rules [1]:

Definition 2.1 (Argument) *Let P be an extended logic program. An* argument *for a conclusion L is a finite sequence $A = [r_n; \ldots; r_m]$ of rules $r_i \in P$ such that*

[1] In contrast to the latter we do not distinguish between strict and defensible rules. However, our results can be extended in this direction.

1. *for every i ($n \leq i \leq m$), and for every objective literal L_j in the body of r_i there is a $k < i$ such that L_j is the head of r_k;*
2. *L is the head of some rule of A;*
3. *No two distinct rules in the sequence have the same head.*

An argument A' (for some conclusion L') is a sub-argument *of the argument A (possibly for some other conclusion L) iff A' is a prefix of A.*

Definition 2.2 (Undercut, Rebut, Attack) *Let A_1 and A_2 be two arguments, then (1) A_1* undercuts *A_2 iff A_1 is an argument for L and A_2 is an argument with assumption not L, i.e. there is an $r : L_0 \leftarrow L_1, \ldots, L_l, not\ L_{l+1}, \ldots, not\ L_m \in A_2$ and a j ($l+1 \leq j \leq m$) such that $L = L_j$; (2) A_1* rebuts *A_2 iff A_1 is an argument for L and A_2 is an argument for $\neg L$; and (3) A_1* attacks *A_2 iff A_1 undercuts or rebuts A_2.*

Definition 2.3 (Coherent, Conflict-free) *An argument is* coherent *if it does not contain sub-arguments attacking each other. A set of arguments Args is called* conflict-free *if no two arguments in Args attack each other.*

Defeat of an argument can be direct, or indirect, by defeating one of its sub-arguments. In particular, any incoherent argument is defeated by the empty argument.

Definition 2.4 (Defeat, Strictly Defeat) *Let A_1 and A_2 be two arguments, then (1) A_1* defeats *A_2 iff A_1 is empty and A_2 incoherent; or A_1 undercuts A_2 or A_1 rebuts A_2 and A_2 does not undercut A_1; and (2) A_1* strictly defeats *A_2 iff A_1 defeats A_2 but not vice versa*

An argument A is acceptable if it can be defended against all attacks, i.e. all arguments that attack A can be counter attacked by *undercuting*.

Definition 2.5 (Acceptable) *An argument A is* acceptable *wrt. a set of arguments Args iff each argument undercuting A is strictly defeated by an argument in Args.*

Our notion of acceptability differs from Prakken and Sartor's definition. In theirs, an argument A is acceptable wrt. *Args* iff each *defeating* argument is strictly defeated by *Args*. In general, their proposal is more sceptical than ours:

Example 2.1 *Consider the program $P = \{\neg a; a \leftarrow not\ b; b \leftarrow not\ a\}$ and $Args = \{[\neg a], [a \leftarrow not\ b], [b \leftarrow not\ a]\}$. Note that arguments $[\neg a]$ and $[a \leftarrow not\ b]$ defeat each other. So, for Prakken and Sartor's definition there is no acceptable argument wrt. Args: e.g, $[\neg a]$ is not acceptable because $[a \leftarrow not\ b]$ defeats it and is not strictly defeated by Args. This conclusion contradicts our intuition: since $\neg a$ is a fact, it should have some kind of "priority" over the rule $a \leftarrow not\ b$ whose premise is a* default *assumption. Based on our proposal, $[\neg a]$ and $[b \leftarrow not\ a]$ are acceptable wrt. Args whereas $[a \leftarrow not\ b]$ is not. In fact: $[\neg a]$ is acceptable because no argument undercuts it; $[b \leftarrow not\ a]$ is acceptable because the only argument undercutting it ($[a \leftarrow not\ b]$) is defeated by $[\neg a]$; $[a \leftarrow not\ b]$ is not acceptable because the only argument undercutting it ($[b \leftarrow not\ a]$) is not strictly defeated by any other argument in Args.*

The fixpoint operator below captures the set of acceptable arguments:

Definition 2.6 (Characteristic Function) *Let P be an ELP and S be a subset of arguments of P. The* characteristic function *of P and S is*

$$F_P(S) = \{A \in S \mid A \textit{ is acceptable wrt. } S\}$$

Proposition 2.1 *F_P is monotonic.*

The status of arguments is determined based on the ways in which they interact. The definition respect the "weakest link" principle that an argument cannot be justified unless all of its sub-arguments are justified. It takes as input the set of all possible arguments and their mutual relations of defeat, and produces as output a split of arguments into three classes: arguments with which a dispute can be 'won', 'lost', and arguments which leave the dispute 'undecided'.

Definition 2.7 (Justified, Overruled, Defeasible) *Let P be an ELP and F_P be the* characteristic function *of P then (1) A is* justified *iff A is in the least fixpoint of F_P*[2] *(called $JustArgs$); (2) A is* overruled *iff A is attacked by a justified argument; and (3) A is* defeasible *iff A is neither justified nor overruled.*

Proposition 2.2 *F_P is finitary iff each argument in $JustArgs$ is attacked by at most a finite numbers of arguments in $JustArgs$.*

Proposition 2.3 *Let P be a non-contradictory extended logic program and L an objective literal then (1) $L \in WFSX(P)$ (L is true in P) iff an argument for L is justified; (2) $not\ L \in WFSX(P)$ (L is false in P) iff all arguments for L are overruled; and (3) $\{L, not\ L\} \cap WFSX(P) = \emptyset$ (L is undefined in P) iff all arguments for L are defensible*

Prakken and Sartor propose a proof for a justified argument for L as a dialogue tree where the root of the tree is an argument for L, and each branch of the tree is a dialogue between a proponent P and an opponent O. A *move* in a dialogue consists of an argument attacking the last move of the other player. The required strength of a move depends on who states it. Since the proponent wants a conclusion to be justified, a proponent's argument has to be strictly defeating. The opponent simply wants to prevent the conclusion from being justified. Thus there is no need for its move to be strictly defeating: it is enough for it to be defeating. The dialogue definition below is a modification of the original one [10], so that it can capture our modified notion of acceptability, where attacks to an argument cannot be rebuts.

Definition 2.8 (Dialogue) *A* dialogue *is a finite nonempty sequence of moves $move_i = (Player_i, A_i)(i > 0)$, such that*

1. *$Player_i = P$ iff i is* odd; *and $Player_i = O$ iff i is* even
2. *If $Player_i = Player_j = P$ and $i \neq j$, then $A_i \neq A_j$*
3. *If $Player_i = P(i > 1)$, then A_i is a minimal (wrt. set inclusion) argument strictly defeating A_{i-1}*
4. *If $Player_i = O$, then A_i undercuts A_{i-1}*

[2] Since F_P is monotonic, justified arguments can be obtained interactively, starting from $\emptyset$.

The first condition says that P begins and the players take turns. The second condition prevents the proponent from repeating its attacks. The remaining two conditions form the heart of the definition: they state the burdens of proof for P and O. The minimally condition on P's move makes it impossible to make arguments trivially different by combining them with some other, irrelevant argument. Finally, a dialogue tree considers all possible ways in which an opponent can attack an argument:

Definition 2.9 (Dialogue Tree) *A* dialogue tree *is a finite tree of moves* $move_i = (Player_i, A_i)$, $Player_i \in \{P, O\}$ *such that*

1. *Each branch is a dialogue*
2. *If* $Player_i = P$ *then the children of* $move_i$ *are all defeaters of* A_i

A player wins a dialogue tree iff it wins all branches (i.e. dialogues) of the tree and a player wins a dialogue if the other player cannot move (i.e. counter-argue).

Proposition 2.4 *An argument A is* justified *iff the proponent wins the dialogue tree starting with A.*

3 Multi-Agent Argumentation

To extend the single-agent to a multi-agent approach we define start by defining what a multi-agent system is: a set of several extended logic programs, each one corresponding to an individual agent with an individual view of the world.

Definition 3.1 (Multi-agent System) *Let* Ag_i *be extended logic programs, where* $1 \leq i \leq n$. *Then the set* $\mathcal{A} = \{Ag_1, \ldots, Ag_n\}$ *is called a* multi-agent system *(MAS).*

We want to exchange parts of several independent or overlapping programs and to obtain a consensus among the programs concerning inference of literals. Our multi-agent semantics is closely related to the single agent one. In particular we want the behavior of a MAS with only one agent to coincide with the single agent approach. But the semantics of the MAS, when more than one agent is involved, might not necessary coincide with the semantics of the union of all agents.

Definition 3.2 (Multi-agent argumentation) *Let* $\mathcal{A}$ *be a MAS, Args be the set of arguments of* $\bigcup_i (Ag_i \in \mathcal{A})$. *Agent* $Ag \in \mathcal{A}$ *successfully argues wrt. an argument A of Ag iff A* justified *wrt. Args; A is* overruled *in Ag iff A is overruled wrt Args; otherwise, A is concluded as* defeasible *in Ag.*

Since previous approaches did not tackle multi-agent argumentation there was no need for cooperation. With two or more agents involved in the process it may be the case that an agent needs the support of another one in order to counter-argue.

Example 3.1 *Consider a trial with a judge, a defender, a prosecutor and a witness. The judge has no knowledge, while the defender knows that the accused is not guilty by default:* $\neg guilty \leftarrow not\ guilty$. *The prosecutor knows that the accused is guilty if there is evidence:* $guilty \leftarrow seen$. *Finally the witness saw the accused committing the crime and therefore knows seen. The* $\mathcal{A} = \{\{\}, \{\neg guilty \leftarrow not\ guilty\}, \{guilty \leftarrow seen\}, \{seen\}\}$ *is a MAS* *that models the trial knowledge, the agents being judge, defender, prosecutor and witness, respectively. The defender has an argument for* $\neg guilty$: $A_1 = [\neg guilty \leftarrow not\ guilty]$, *and the witness has* $A_2 = [seen]$ *for* $seen$. *The prosecutor cannot support* $guilty$ *because it has no knowledge about* $seen$. *If the judge asks the defender to defend the accused, the prosecutor needs the help of a witness to defeat the arguments of the defender.*

As shown by the example above, the prosecutor cannot continue to argue because it has just a part of an argument, i.e. a partial argument:

Definition 3.3 (Partial Argument) *Let* Ag *be an ELP. A* partial argument *for a conclusion* L *is a finite sequence* $PA = [r_n; \ldots; r_m]$ *of rules* $r_i \in Ag$ *such that*

1. *for every* i $(n < i \leq m)$, *and for every objective literal* L_j *in the body of* r_i *there is a* $k < i$ *such that* L_j *is the head of* r_k.
2. L *is the head of some rule of* A
3. *No two distinct rules in the sequence have the same head.*

Note that, contrary to the definition of argument, an objective literal L_j in the body of r_n need not necessarily have a rule with head L in the argument. Intuitively this simply means that the argument is incomplete with respect to that L_j.

Example 3.2 *Continuing the example 3.1, the prosecutor has a partial argument for* $guilty$, $PA = [guilty \leftarrow seen]$. *This partial argument is incomplete wrt* $seen$.

To continue an argumentation process, a partial argument is not enough. If an agent has only a partial argument then it should be completed by a multi-agents cooperation:

Definition 3.4 (Multi-agent cooperation) *Let* $\mathcal{A}$ *be a MAS. Agent* $Ag_i \in \mathcal{A}$ *successfully cooperates with a set of agents* $Ags \subseteq \mathcal{A}$ *wrt. a partial argument* PA *(for some literal* L*) of* Ag_i *iff for all objective literals* $L_k \in PA$ *there is an agent* $Ag_j \in Ags$ *such that* Ag_j *has an argument for* L_k.

To obtain the results of multi-agent cooperation we define cooperative dialogue. Intuitively, in cooperative dialogues other agents provide arguments for the still incomplete objective literals. The process stop either if no agents can provide the needed arguments or if there are no more incomplete literals:

Definition 3.5 (Cooperative Dialogue) *Let* $\mathcal{A}$ *be a MAS,* Ag *be an agent in* $\mathcal{A}$, $A = [r_n; \ldots; r_m]$ *be a partial argument (for some literal* L*) of* Ag, *and* Inc *be the set of objective literals* L_j *in the body of* r_n. *A* cooperative dialogue *for* A *in* Ag *is a finite nonempty sequence of moves,* $move_i^c = (Ag_k, A_i)$ $(i > 0)$ *such that*

1. $move_1^c = (Ag, A)$.
2. *if* $i \neq j$ *then* $A_i \neq A_j$.
3. *Let* $move_i^c = (Ag_i, [r_n; \ldots; r_m])$. *Then if there exists an* $Ag_k \in \mathcal{A}$ *with an argument for some* $L \in Inc$, $move_{i+1}^c = (Ag_k, [r_{n_k}; \ldots; r_{m_k}; r_n; \ldots; r_m])$; *otherwise,* $move_i^c$ *is the last move in the dialogue.*

A cooperative dialogue of an agent Ag *for a partial argument* A *is successful if its last move* (Ag_f, A_f) *is a (complete) argument. Then we call* A_f *its* result.

The first condition says that an agent Ag starts a cooperation process for a partial argument A. The second prevents agents from repeating arguments. The third defines that the next movement is an argument for some incomplete literal L in the previous partial argument. The process stops either if there is no argument for such L, meaning that the cooperation failed, or if there is no more incomplete literal, meaning that the partial argument is completed.

Proposition 3.1 *Let $\mathcal{A}$ be a MAS, and Ag_k be an agent in $\mathcal{A}$. A cooperation for a partial argument PA (for some L) of Ag_k is successful if a cooperative dialogue for PA is successful.*

Finally, we extended the dialogue process of Def. 2.8 to include cooperation and argumentation between various agents.

Definition 3.6 (Extended Dialogue) *Let $\mathcal{A}$ be a MAS and Ag be an agent in $\mathcal{A}$. An* extended dialogue *for an argument A of an agent Ag is a finite nonempty sequence of moves $move_i^a = (Role_i, Ag_i, A_i)$ $(i > 0)$, where $Role_i \in \{P, O\}$ and $Ag_i \in \mathcal{A}$, such that*

1. *$move_1^a = (P, Ag, A)$.*
2. *$Role_i = P$ iff i is* odd*; and $Role_i = O$ iff i is* even
3. *If $Role_i = Role_j = P$ and $i \neq j$, then $A_i \neq A_j$*
4. *A_i is either an argument of Ag_i or the* result *of a successful cooperation dialogue for a partial argument of Ag_i.*
5. *If $Role_i = P$ $(i > 1)$ then A_i is a minimal (wrt. set inclusion) argument strictly defeating A_{i-1}*
6. *If $Role_i = O$ then A_i undercuts A_{i-1}*

The first condition says that an agent Ag starts an argumentation process for an argument A as a proponent. The second one says that the agent's argument can either be used to attack the argument in the first move (in which case $Role = O$), i.e. the agent is an opponent; or to counterattack (in which case $Role = P$), i.e. the agent is a proponent. The third prevents the proponent agents from repeating arguments. The fourth states that a dialogue just continues with arguments of individual agents, or the results of successful cooperations for a partial arguments of individual agents. The remaining two conditions state the burdens of proof for agents having the role P and O. The minimally condition on P's move makes it impossible to make arguments trivially different by combining them with some other, irrelevant argument.

Finally, an extended dialogue tree considers all possible ways in which an agent can attack an argument:

Definition 3.7 (Extended Dialogue Tree) *Let $\mathcal{A}$ be a MAS, Ag be an agent in $\mathcal{A}$ and A be an argument of Ag. An* extended dialogue tree *for A is a finite tree of moves $move_i^a = (Role, Ag_k, A_i)$, such that*

1. *Each branch is an* extended dialogue *for argument A of Ag*
2. *If $Role = P$ then the children of $move_i^a$ are all the defeaters of an argument A_i*

An agent wins an extended dialogue tree iff it wins all branches (i.e. dialogues) of the tree and an agent wins a dialogue if another agent cannot move (i.e. counter-argue).

Proposition 3.2 *Let Ag and Ag_k be agents, and A be an argument (for some literal L) of Ag. An argument A is* justified *iff Ag wins the dialogue tree starting with A; otherwise, A is overruled or defeasible. An argument A is overruled when an opponent Ag_k moves at least one justified argument; otherwise, A is defeasible.*

In the remainder of the paper we only use *extended dialogue tree* (resp. *extended dialogue*). So, for simplicity, we refer to them simply as *dialogue tree* (resp. *dialogue*).

4 Argumentation Protocol

To implement the MAS's inference operation, the agents evolve an argumentation process. In the light of speech act theory and subsequent agent communication languages such as ACL, KIF, or KQML [5] we can identify five relevant speech acts of the dialogues. By assuming that L (resp. L') is an objective literal and A (resp. A') is an argument that supports L (resp. L') then an agent requests inference of a literal L by sending a token $ask(L)$ and replies to a request by $reply(L, A)$ if the agent won the argumentation with argument A; otherwise, it replies $reply(not\ L, none)$. An agent proposes a conclusion L to other involved agents by sending the token $propose(L, A)$. An agent opposes an argument by $oppose(L', A', L, A)$ where $L' \leftarrow A'$ is an argument attacking the previously received $L \leftarrow A$. If the attack is by rebut then $L' = \neg L$; if it is by undercut then $L' = L''$ and $A = [\ldots, not\ L'', \ldots]$. In case the received argument A is acceptable the agent acknowledges this by sending $agree(L, A)$.

Using the above speech acts we define an argumentation protocol. We assume that a proposing agent sends its conclusion to all involved agents, but the opposition against a conclusion occurs only between the proponent and the opposer. In the following we sketch an algorithm for multi-agents argumentation (as defined in section 3).

Algorithm:

- Proponent and Opponent, Cooperation for L: If the agent has just a partial argument for L (conf.Def.3.3) then *ask* all involved agents, i.e. agents that are able to cooperate (conf. Def.3.4) to complete the partial argument. On receipt of a *query*, from an agent that is not external[3], to infer L try to built an argument A for L, by itself or by cooperation, and reply it. If there is no such A then reply negatively. On receipt of a *reply*. If it is a positive answer then update the cooperation process as a cooperative dialogue (conf.Def.3.5) and continue to argue; otherwise, give up to built the previous (counter-)argument and try to built other argument (see argumentation).
- Proponent, Argumentation: On receipt of a *query*, from an external agent, to infer L try to built an argument, by itself or by asking other agents to complete a partial argument PA (see cooperation). If there is one argument A then create a dialogue tree (conf.Def.3.7) for A and propose it to all involved agents, i.e. agents that are able to argue; otherwise, reply that L is a false conclusion. On receipt of an *agreement*, check the state of the dialogue tree: if some dialogue (conf.Def.2.8) has not finished yet then wait for other answers, else check dialogue tree's leaves: if all leaves received are wining leaves then reply the agreement positively; otherwise, reply negatively.

[3]An external agent is an agent not belonging to the MAS. In our trial example, the judge is an external agent.

On receipt of an *oppose*, put the counter-argument CA on the involved dialogue (conf.Def.2.8). Try to built a counter-argument CCA as a proponent, itself or by asking other agents to complete a partial argument PA (see cooperation): if there is such CCA then put it on the involved dialogue (conf.Def.2.8) and propose CCA to all involved agents. If there is no such CCA then reply the initial request negatively.

- Opponent, Argumentation: On receipt of a *proposal*, try to built a counter-argument CA as an opponent, itself or by asking for cooperation to complete a partial argument PA, and then oppose. If there is no such CA then agree with proposal.

Finally, we present an example to show the process of cooperation and argumentation. We assume a trial with a judge, a defensor, a prosecutor, a defense witness and a prosecution witness. The judge, the prosecutor and the defensor have the same knowledge as shown in example 3.1. If the judge asks the prosecutor to prove that the accused is guilty, the prosecutor needs the help of a witness. The prosecution witness saw the accused committing the crime and therefore knows seen. As the defensor cannot rebut this testimony, then it agrees and the prosecutor wins the argumentation dialogue. On the other side, if the judge asks the defender to prove that the accused is not guilty, the defender needs the help of a witness to destroy the prosecutor testimony. The defense witness knows that the prosecution witness needs to use glasses, and that if he is not using them then he cannot see. The accusation wins because the prosecution witness affirms he was using the glasses and the defense cannot rebut this argument. Then the trial knowledge is model by $\mathcal{A} = \{defender, prosecutor, witness_d, witness_p\}$. As shown in the example, it is not necessary for an agent to cooperate and argue with all other agents in the multi-agent system. Accordingly, in the implementation, we restrict argumentation and cooperation of an agent to a predefined (provided by the user) set of other agents. These predefined set are declared in the program of an agent Ag_i with predicates $argue\_with(ListOfAgentsA)$ and $cooperate\_with(ListOfAgentsC)$ meaning that Ag_i can only argue (resp. ask by cooperation) with the agents in $ListOfAgentsA$ (resp. $ListOfAgentsC$).

$$\begin{array}{rcl}
defender & = & \{argue_with([prosecuter]);\ cooperate_with([witness_d]);\\
 & & \neg guilty \leftarrow not\ guilty\}\\
prosecutor & = & \{argue_with([defender]);\ cooperate_with([witness_p]);\\
 & & guilty \leftarrow seen\}\\
witness_p & = & \{argue_with([]);\ cooperate_with([]);\\
 & & using_glasses;\ seen\}\\
witness_d & = & \{argue_with([]);\ cooperate_with([]);\\
 & & need_glasses;\ \neg seen \leftarrow need_glasses, not\ using_glasses\}
\end{array}$$

The argumentation dialogue is shown in the figure 1. Note that the judge can conclude the accused is guilty in both cases. This is so because the testimonies of the prosecution witness cannot be defeated. Nevertheless if the prosecutor witness could not remember that he was using glasses and the testimonies of both witnesses could not be defeated and the judge would then be able to conclude both *guilty* and $\neg guilty$ (i.e. a consensus could not be reached). To solve this problem priorities should be used in order to always obtain a consensual result.

1	*judge*	$\longrightarrow$	*prosecutor*	$ask(guilty)$
2	*prosecutor*			*cooperation for a partial argument* PA_1
3	*prosecutor*	$\longrightarrow$	$witness_p$	$ask(seen)$
4	*prosecutor*	$\longleftarrow$	$witness_p$	$reply(seen, A_3)$
5	*prosecutor*			*assimilates* $A_5 = A_3 + PA_1$
6	*prosecutor*	$\longrightarrow$	*defender*	$propose(guilty, A_5)$
7	*prosecutor*	$\longleftarrow$	*defender*	$agree(guilty, A_5)$
8	*prosecutor*			*checks the dialogue*
9	*judge*	$\longleftarrow$	*prosecutor*	$reply(guilty, A_5)$

1	*judge*	$\longrightarrow$	*defender*	$ask(\neg guilty)$
2	*defender*	$\longrightarrow$	*prosecutor*	$propose(\neg guilty, A_1)$
3	*prosecutor*			*cooperation for a partial argument* PA_1
4	*prosecutor*	$\longrightarrow$	$witness_p$	$ask(seen)$
5	*prosecutor*	$\longleftarrow$	$witness_p$	$reply(seen, A_3)$
6	*prosecutor*			*assimilates* $A_5 = A_3 + PA_1$
7	*defender*	$\longleftarrow$	*prosecutor*	$oppose(guilty, A_5, \neg guilty, A_1)$
8	*defender*			*cooperation to prove the golo* $\neg seen$
9	*defender*	$\longrightarrow$	$witness_d$	$ask(\neg seen)$
10	*defender*	$\longleftarrow$	$witness_d$	$reply(\neg seen, A_2)$
11	*defender*			*assimilates* $A_6 = A_2 + \emptyset$
12	*defender*	$\longrightarrow$	*prosecutor*	$propose(\neg seen, A_6)$
13	*prosecutor*			*cooperation to prove the golo* $using\_glasses$
14	*prosecutor*	$\longrightarrow$	$witness_p$	$ask(using\_glasses)$
15	*prosecutor*	$\longleftarrow$	$witness_p$	$reply(using\_glasses, A_4)$
16	*prosecutor*			*assimilates* $A_7 = A_4 + \emptyset$
17	*defender*	$\longleftarrow$	*prosecutor*	$oppose(using\_glasses, A_7, \neg seen, A_6)$
18	*defender*			*cooperation to prove the golo* $\neg using\_glasses$
19	*defender*	$\longrightarrow$	$witness_d$	$ask(\neg using\_glasses)$
20	$witness_d$			*cooperation to prove the golo* $\neg using\_glasses$
21	*defender*	$\longleftarrow$	$witness_d$	$reply(not\ \neg using\_glasses, none)$
22	*judge*	$\longleftarrow$	*defender*	$reply(not\ \neg guilty, none)$

$$
\begin{aligned}
A_1 &= [\neg guilty \leftarrow not\ guilty] \\
A_2 = A_6 &= [need\_glasses; \neg seen \leftarrow need\_glasses, not\ using\_glasses] \\
A_3 &= [seen] \\
A_7 = A_4 &= [using\_glasses] \\
PA_1 &= [guilty \leftarrow seen] \\
A_5 &= [seen; guilty \leftarrow seen]
\end{aligned}
$$

Figure 1: Trace of the consistent trial example.

5 Comparison, Conclusion and Future work

The work presented in this paper is based on work by Dung [3, 4] and Prakken and Sartor [10] on argumentation. Dung defines a declarative semantics for extended logic programs using the metaphor of argumentation. Our work continues this line of research in that we extend the single-agent approach to a multi-agent one. Prakken and Sartor are motivated by legal reasoning and define a rigorous and concise framework similar to Dung's, and they also deal only with a single agent and therefore do not tackle the issue of cooperation. The Prakken and Sartor's argumentation semantics defines a set of priorities to solve conflicts on the argumentation dialogue (for details see [10]) and it can be easily added to our argumentation proposal. When there is no determined priority, literals involved in some conflict are simple considered as defeasible. Since our argumentation semantic is based on the Coherence principle it is more credulous and leads to more intuitive results than their proposal. Our previous work on argumentation [11] has defined multi-agent argumentation. However it proposed a simplified cooperation process: a partial argument is defined as a rule. We extend the definition of partial argument for also incomplete sequence of rules. We improve the previous algorithm to have cooperation and simulate it for a multi-agent inference. We show the connection between *WFSX* semantics and argumentation. Finally, Prakken and Sartor [10] and Dung [4] propose different methods to get sceptical or credulous argumentation. As future work, we intend to define just one (flexible) argumentation framework to obtain conclusions in more credulous or sceptical way [7], depending on the application's goal.

References

[1] J. J. Alferes, C. V. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.

[2] J. J. Alferes and L. M. Pereira. *Reasoning with Logic Programming.* (LNAI 1111), Springer-Verlag, 1996.

[3] P. M. Dung. An argumentation semantics for logic programming with explicit negation. In *Proc. of the 10th International Conference on Logic Programming*, pages 616–630. MIT Press, 1993.

[4] P. M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[5] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, R. Pelavin, S. Shapiro, and C. Beck. Specification of the KQML agent communication language. Technical report, The DARPA Knowledge Sharing Initiative, External Interfaces Working Group, Baltimore, USA, 1993.

[6] Allen Van Gelder, Kenneth Ross, and John S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceeding of the 7th ACM Symposium on Principles of Databse Systems*, pages 221–230. Austin, Texas, 1988.

[7] I. A. Móra and J. J. Alferes. Credulus and sceptical argumentation for extended logic programs. Submited to ECAI'98.

[8] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Non–monotonic reasoning with logic programming. *Journal of Logic Programming. Special issue on Nonmonotonic reasoning*, 17(2, 3 & 4), 1993.

[9] Luís Moniz Pereira and José Júlio Alferes. Well founded semantics for logic programs with explicit negation. In *B. Neumann (Ed.), European Conference on Artificial Intelligence*, pages 102–106. John Wiley & Sons, 1992.

[10] Henry Prakken and Giovanni Sartor. Argument-based extended logig programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 1997.

[11] M. Schroeder, I. A. Móra, and J. J. Alferes. Vivid agents arguing about distributed extended logic programs. In Ernesto Costa and Amilcar Cardoso, editors, *Progress in Artificial Intelligence, 8th Portuguese Conference on Artificial Intelligence EPIA97*, volume LNAI1323, pages 217–228. Springer–Verlag, Coimbra, Portugal, 1997.