Reasoning about Trust and Belief in Possibilistic Answer Set Programming

Gabriel Maia
Department of Computer Science
Federal University of Ceará
Fortaleza, Ceará 60455-760
Email: gabrielrocha@lia.ufc.br

João Alcântara
Department of Computer Science
Federal University of Ceará
Fortaleza, Ceará 60455-760
Email: jnando@lia.ufc.br

***Abstract*—The Possibilistic Answer Set Framework was conceived to deal with not only non monotonic reasoning, but also with uncertainty by associating a certainty level to each piece of knowledge. Here we extend this formalism to a multiagent approach robust enough to manage both the uncertainty about autonomous agents expressed in terms of degrees of trust and the possibilistic uncertainty about their knowledge bases expressed as possibilistic answer set programs. As result, we have a decentralized system able to reason about trust and beliefs in an integrated way. Then we motivate its behavior on an example and highlight how our proposal can be employed to make decisions when the information is distributed, uncertain, potentially contradictory and not necessarily reliable.**

I. Introduction

In the development of computer systems to tackle problems issued from Artificial Intelligence, we note these systems have become increasingly distributed and with a decentralized control. In such a scenario, when representing knowledge and inference, it urges to resort to mechanisms to manage unreliable sources of information as well as the uncertainty associated with the information itself. With this motivation in mind, our main goal here is to define an extension of the Answer Set Programming paradigm (ASP) [1], [2], [3], [4] for distributed incomplete and uncertain knowledge bases whose reliability is described in terms of degrees of trust.

As we know, in the last decades, many works have been proposed to deal with non classical forms of reasoning such as non monotonic reasoning. Such efforts culminate with the development of the aforementioned declarative programming paradigm of Answer Set Programming. Despite gloomy prognostications about the future of non monotonic reasoning, owing to its highly efficient solvers [5], [6], nowadays ASP is regarded as a primary candidate for an effective knowledge representation tool [7]. In [8] this formalism is extended to encompass also uncertainty related to Possibility Theory [9]. The resulting formalism is called Possibilistic Answer Set Program ($PASP$). The distinguishing aspect of these programs is to embed two aspects of common sense reasoning: non monotonicity and uncertainty.

Notwithstanding, $PASP$ was tailored to consider only a unique knowledge base. This means that it cannot manage the uncertainty related to the existence of distributed sources of information and decentralized mechanisms of inference as it typically occurs in multiagent systems. The situation is still worse if we could not assume that the knowledge represented by every $PASP$ is reliable!

In this work, in order to fill this gap, we will assess the reliability of each $PASP$ in terms of a degree of trust. As pointed out in [10], although in the multiagent systems community, there is not a consensual definition of trust, they are in accord that there is a degree of uncertainty associated with trust, and that trust is tied up with the relationships between individuals.

Still in [10], the authors showed a general system of argumentation grounded on a trust network $\mathcal{T}$ to capture relationships between agents. In $\mathcal{T}$ each agent maintains ratings of how much its acquaintances are trusted, and how much its acquaintances trust their acquaintances, and so on. These ratings are employed to infer reasonably how much an agent trusts each other agent. The system was characterized in such a way that an agent can construct arguments in which belief in the conclusions is related to the degree of trust in the agents who supplied useful information for these arguments.

Our idea is to take this proposal for general systems of argumentation and adapt it to represent reasoning about trust and belief in possibilistic answer set programming. As consequence, in order to obtain a conclusion, an agent will resort to its own knowledge base as well as its acquaintances' knowledge bases, which will be weighted by the degree of trust inferred from a trust network. Our approach is designed to offer a unifying view of the uncertainty related to trust and that related to possibility theory by integrating degrees of trust with the certainty weights associated with each rule of a possibilistic answer set program.

In the next section, we provide the reader with some important notions related to possibilistic answer set programming, in which weights refer to the certainty with which the conclusions of rules can be derived, given that their body is known to hold. In the sequel, we show how to reason about trust by resorting to a trust network. The main contributions of this work can be found in Section IV, where we extend possibilistic answer set to couple with a multiagent approach based on a trust network. Then, in Section V, we illustrate the intuition of our proposal by resorting to a running example based on a multiagent scenario. Finally, in Section VI, we conclude our

978-1-5090-3566-3/16 $31.00 © 2016 IEEE
DOI 10.1109/BRACIS.2016.38

work and motivate further directions.

II. Possibilistic Answer Set Programming

Now we will present the definition of Possibilistic Answer Set Programs ($PASP$) [8], in which uncertainty is taken into account by using possibility theory. Then we will study the behavior of such programs under a computational point of view based on fixpoint operators.

We start by defining a finite set of *atoms* $\mathcal{A}$. A *literal* is simply an atom $a \in \mathcal{A}$ or its explicit negation $\neg a$. For a set of literals L, by $\neg L$ we mean the set $\{\neg l \mid l \in L\}$, in which by definition $\neg\neg a = a$. A set of literals is said to be *consistent* when $L \cap \neg L = \emptyset$, i.e. L does not contain two contradictory literals. The set of all literals is written as $L = \mathcal{A} \cup \neg\mathcal{A}$. A naf-literal is either a literal 'l' or an expression of the form '$not\ l$', in which 'not' denotes negation-as-failure. Intuitively, we have that '$not\ l$' is true when we have no proof for 'l'. A rule is an expression of the form

$$l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n,$$

in which l_i is a literal for every $0 \leq i \leq n$.

Definition 1 (Possibilistic Answer Set Programs)**.** *A possibilistic answer set program (PASP) is a set of pairs $p = (r, c)$, in which r is a rule and $c \in]0, 1]$ a certainty associated with r. We will also write a pair $p = (r, c)$ as*

$$\mathbf{c} : l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n,$$

in which l_i is a literal for every $0 \leq i \leq n$.

Note the weight associated with a rule is interpreted as the certainty with which we can deduce the head when the body is known to hold [11]. When the rule has an empty body, it is called a *fact* rule and is denoted by $\mathbf{c} : l_0 \leftarrow \top$, in which $\top$ is a special language construct representing a tautology. In addition, we say a *simple rule* is a rule without negation-as-failure, i.e., when $m = n$. We also will call a *possibilistic simple program* a set of simple rules.

Consider $\mathcal{B}_P$ as the set of literals appearing in a $PASP$ P. The set of relevant literals of P is given by $Lit_P = (\mathcal{B}_P \cup \neg\mathcal{B}_P)$. Now we can define Herbrand valuations:

Definition 2 (Herbrand Valuation)**.** *A Herbrand valuation can be seen as a function $V : Lit_P \rightarrow [0, 1]$. The underlying intuition of $V(l) = c$ is that the literal 'l' is true with certainty 'c'.*

For notational convenience, we often also use the set notation $V = \{l^c \mid V(l) = c\}$. In accordance with this set notation, we write $V = \emptyset$ to denote the valuation in which each literal is mapped to 0. For a certainty $c \in [0, 1]$ and a valuation V, we use V^c to denote the classical projection $\{l \mid l \in Lit_P, V(l) \geq c\}$. We also use $V^{\underline{c}} = \{l | l \in Lit_P, V(l) > c\}$, i.e. those literals that can be derived to be true with certainty strictly greater than c.

Definition 3 (Reduct)**.** *Let P be a PASP and $V : Lit_P \rightarrow [0, 1]$ a Herbrand valuation. We define the reduct of P w.r.t V as*

$$\frac{P}{V} = \{\mathbf{c} : l_0 \leftarrow l_1, \ldots, l_m \mid \mathbf{c} : l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n \in P \text{ and } V(l_{m+1}) = \cdots = V(l_n) = 0\}.$$

The reduct of P w.r.t V is obtained by evaluating the naf-literals in each rule of P according to V. As it is clear, the resulting $\frac{P}{V}$ is a possibilistic simple program.

Let the c-cut P_c of a $PASP$ P, with $c \in [0, 1]$, be defined as

$$P_c = \{r \mid (r, c') \in P \text{ and } c' \geq c\}.$$

In other words, P_c is the subprogram of P whose rules have an associated certainty higher than or equal to 'c'.

Definition 4 (Immediate Consequences Operator)**.** *Let P be a possibilistic simple program and V be a Herbrand valuation. The immediate consequence operator T_P is defined as*

$$T_P(V)(l_0) = max\{c \in [0, 1] \mid ((l_0 \leftarrow l_1, \ldots, l_m), c') \in P_c \text{ and } l_i \in V^c \text{ for all } 1 \leq i \leq m\}.$$

We proceed by introducing the function $lfp(P)$ to denote the fixpoint obtained by repeatedly applying T_P starting from the minimal valuation $V = \emptyset$:

Definition 5 (Fixpoint operator)**.** *Let P be a possibilistic simple program. We define the least fixpoint of T_P, denoted by $lfp(P)$, as*

$$T_P^{\uparrow 0} = \emptyset$$
$$T_P^{\uparrow i+1} = T_P(T_P^{\uparrow i})$$
$$lfp(P) = \bigcup_{i \geq 0} T_P^{\uparrow i}.$$

In [8], the authors prove that every possibilistic simple program has a (unique) least fixpoint. Now we can define possibilitics answer sets:

Definition 6 (Possibilistic Answer Set)**.** *Let P be a PASP. We say a valuation V is a possibilistic answer set of P iff*

$$V = lfp(\frac{P}{V}).$$

Example 1. *[8] Let P be the following PASP:*

$$P = \left\{ \begin{array}{ll} \mathbf{1} : dr_1 \leftarrow di_1, not\ dr_2 & \mathbf{1} : dr_2 \leftarrow di_2, not\ dr_1 \\ \mathbf{0.7} : c_1 \leftarrow dr_1, di_1 & \mathbf{0.3} : c_2 \leftarrow dr_2, di_2 \\ \mathbf{0.9} : di_1 \leftarrow \top & \mathbf{0.7} : di_2 \leftarrow \top \end{array} \right\}.$$

The two first rules (appropriateness and incompatibility of the drugs) are considered absolutely sure. The third (resp. fourth) rule expresses that we are almost (resp. little) certain of the efficiency of the drug 1 (resp. drug 2). The two last rules indicate that the diagnosis of disease 1 (resp. disease 2) is

quasi (resp. almost) certain. According to Definition 6, P has the following possibilistic answer sets:

$$S_1 = \{(di_1, 0.9), (di_2, 0.7), (dr_1, 0.9), (c_1, 0.7)\}.$$
$$S_2 = \{(di_1, 0.9), (di_2, 0.7), (dr_2, 0.7), (c_2, 0.3)\}.$$

So, the doctor can observe that he has an alternative. On one hand, he can quasi certainly give the drug dr_1 and be almost certain that the patient will be cured of the disease di_1. On the other hand, he can give almost certainly the drug dr_2 and the patient will be cured of the disease di_2 but that is only little certain. However, if the doctor considers that disease di_2 is very serious, maybe he will choose the drug dr_2 even if the degree is lower. That is why it is interesting to obtain and keep the possibilistic answer sets in which every conclusion is weighted with a certainty degree.

III. Trust Network

Inspired by [12], we will show in this section how to extend possibilistic answer sets programs to deal with a multiagent scenario based on a trust network. In this scenario, agents maintain a trust network of their acquaintances, which includes ratings of how much those acquaintances are trusted, and how much those acquaintances trust their acquaintances, and so on. Then an agent will employ the information obtained from his trust network to extend and review his own point of view and to infer new information in a reasonable way.

Definition 7 (Trust Network)**.** *A trust network T is a tuple*

$$\mathcal{T} = \langle Ags, \tau, tr \rangle,$$

in which Ags is a finite set of agents, $\tau \subseteq Ags \times Ags$ is a direct trust relation and $tr : Ags \times Ags \to [0, 1]$ is a trust value function such that

$$tr(Ag_i, Ag_j) \neq 0 \Leftrightarrow (Ag_i, Ag_j) \in \tau.$$

A trust network is a weighted directed graph $\mathcal{T}$, whose vertices are the agents in Ags; the trust relation τ between pairs of agents characterises the directed edges of $\mathcal{T}$, and tr is a function that assigns a degree of trust that one agent has in another, which is also the weight of the edge such trust represents.

Note that in a trust network, the existence of (Ag_1, Ag_2) in τ is intended to mean that agent Ag_1 trusts Ag_2, and its degree of trust is given by $tr(Ag_1, Ag_2)$. If $tr(Ag_1, Ag_2) = 0$, then Ag_1 definitely does not trust Ag_2; if it is 1, then Ag_1 totally trusts Ag_2. However, we emphasize we cannot take as granted that the trust relation τ is symmetric. In other words, from (Ag_1, Ag_2) in τ, we will not assume (Ag_2, Ag_1) in τ.

The relation in τ encodes direct trust. As pointed in [12], [13], [14], it is usual to assume trust relations are transitive. Such an indirect trust is captured in a trust network in terms of trust paths:

Definition 8 (Generalized Trust Relation)**.** *Let $\mathcal{T} = \langle Ags, \tau, tr \rangle$ be a trust network. We define τ^* as the transitive closure of τ, i.e., for all $Ag_i, Ag_j, Ag_k \in Ags$, if $(Ag_i, Ag_j) \in \tau$ and $(Ag_j, Ag_k) \in \tau$, then $(Ag_i, Ag_k) \in \tau$. We say Ag_i trusts Ag_j in $\mathcal{T}$ if $(Ag_i, Ag_j) \in \tau^*$.*

Although in this work we assume trust is a transitive relation, as pointed in [15], [16], we acknowledge that this is not always the case. Indeed, in [12], some alternatives based on metareasoning to model indirect trust are discussed. Besides transitivity, other rules to propagate trust can be considered as, for instance, *co-citation*[14]:

Agent Ag_{i_1} trusts Ag_{j_1} and Ag_{j_2}, and Ag_{i_2} trusts Ag_{j_2}. Under co-citation, we would conclude that Ag_{i_2} should also trust Ag_{j_1}.

Anyway, our purpose here is to model at least those situations in which it is reasonable to consider trust to be transitive. This means that if Ag_i trusts Ag_j in $\mathcal{T}$, there exists a directed path $\langle Ag_i = Ag_{i_1}, Ag_{i_2}, \ldots, Ag_{i_n} = Ag_j \rangle$ from Ag_i to Ag_j in $\mathcal{T}$. Then we have

$$\{(Ag_{i_1}, Ag_{i_2}), (Ag_{i_2}, Ag_{i_3}), \ldots, (Ag_{i_{n-1}}, Ag_{i_n})\} \subseteq \tau.$$

Such path provides the means to compute the degree of trust that Ag_1 has in Ag_2 according to $\mathcal{T}$.

Definition 9 (Generalized Trust Degree)**.** *Let $\mathcal{T} = \langle Ags, \tau, tr \rangle$ be a trust network. We define $tr^* : Ags \times Ags \to [0, 1]$ as follows*

$$tr^*(Ag_i, Ag_j) = max\,[min\,[tr(Ag_{i_1}, Ag_{i_2}), tr(Ag_{i_2}, Ag_{i_3}), \ldots, tr(Ag_{i_{n-1}}, Ag_{i_n})] \mid \{(Ag_{i_1}, Ag_{i_2}), (Ag_{i_2}, Ag_{i_3}), \ldots, (Ag_{i_{n-1}}, Ag_{i_n})\} \subseteq \tau \text{ and } Ag_i = Ag_{i_1} \text{ and } Ag_j = Ag_{i_n}].$$

The function $tr^*(Ag_i, Ag_j)$ determines the the trust degree of Ag_i in Ag_j according to $\mathcal{T}$ by taking into account not only the (direct) trust τ and its corresponding trust degree expressed by tr, but also the indirect trust determined by τ^*. Now we will display an example where we show how to calculate $tr^*(Ag_i, Ag_j)$:

Example 2. *Let $\mathcal{T} = \langle Ags, \tau, tr \rangle$ be the trust network depicted below:*

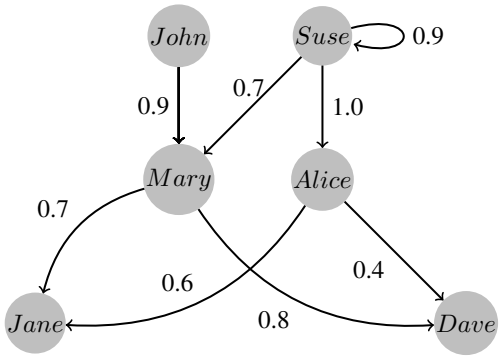

Fig. 1: Trust Network

It is clear that in $\mathcal{T}$ *Alice* trusts *Dave* with degree 0.4, i.e., $tr^*(Alice, Dave) = tr(Alice, Dave) = 0.4$. But how about

$tr^*(Suse, Jane)$? Firstly, observe that because of the loop in $Suse$, there is an infinite number of paths in $\mathcal{T}$ from $Suse$ to $Jane$:

$$\langle Suse, Mary, Jane\rangle \quad \langle Suse, Alice, Jane\rangle$$
$$\langle Suse, Suse, Mary, Jane\rangle, \quad \langle Suse, Suse, Alice, Jane\rangle, \ldots$$

We show, however, that the minimal paths suffice to determine $tr^*(Ag_i, Ag_j)$:

Proposition 1. *Let $\mathcal{T} = \langle Ags, \tau, tr\rangle$ be a trust network, Ag_i and Ag_j agents in Ags, and $\mathcal{P}_1 = \langle Ag_i, Ag_{i_1}, \ldots, Ag_{i_{m-1}}, Ag_j\rangle$ and $\mathcal{P}_2 = \langle Ag_i, Ag_{j_1}, \ldots, Ag_{j_{n-1}}, Ag_j\rangle$ two paths in $\mathcal{T}$ from Ag_i to Ag_j such that*

$$\{Ag_i, Ag_{i_1}, \ldots, Ag_{i_{m-1}}, Ag_j\} \subseteq \{Ag_i, Ag_{j_1}, \ldots, Ag_{j_{n-1}}, Ag_j\}$$

Then

$$min\left[tr(Ag_i, Ag_{j_1}), \ldots, tr(Ag_{j_{n-1}}, Ag_j)\right] \leq min\left[tr(Ag_i, Ag_{i_1}), \ldots, tr(Ag_{i_{m-1}}, Ag_j)\right].$$

Proof. Let $min\left[tr(Ag_i, Ag_{j_1}), \ldots, tr(Ag_{j_{n-1}}, Ag_j)\right] = k$. This means that there exists $tr(Ag_p, Ag_q)$ in $\left[tr(Ag_i, Ag_{j_1}), \ldots, tr(Ag_{j_{n-1}}, Ag_j)\right]$ such that $tr(Ag_p, Ag_q) = i$. As

$$\{Ag_i, Ag_{i_1}, \ldots, Ag_{i_{m-1}}, Ag_j\} \subseteq \{Ag_i, Ag_{j_1}, \ldots, Ag_{j_{n-1}}, Ag_j\}$$

if $tr(Ag_p, Ag_q)$ is in $\left[tr(Ag_i, Ag_{i_1}), \ldots, tr(Ag_{i_{m-1}}, Ag_j)\right]$, then $min\left[tr(Ag_i, Ag_{i_1}), \ldots, tr(Ag_{i_{m-1}}, Ag_j)\right] = k$; otherwise, $min\left[tr(Ag_i, Ag_{i_1}), \ldots, tr(Ag_{i_{m-1}}, Ag_j)\right] \geq k$. □

Thus, as $tr^*(Ag_i, Ag_j)$ is defined in terms of a maximization, the smaller is the path from Ag_i to Ag_j, the greater is the value of $tr^*(Ag_i, Ag_j)$. Consequently, only the minimal paths from Ag_i to Ag_j are enough to calculate such a value. Then, we have that in Example 2, we may consider only the minimal paths $\langle Suse, Mary, Jane\rangle$ and $\langle Suse, Alice, Jane\rangle$ to obtain $tr^*(Suse, Jane)$:

$$min[tr(Suse, Mary), tr(Mary, Jane)] = min[0.7, 0.7] = 0.7$$
$$min[tr(Suse, Alice), tr(Alice, Jane)] = min[1.0, 0.6] = 0.6$$

Hence $tr^*(Suse, Jane) = max[0, 7, 0, 6] = 0, 7$.

The generalized trust degree is based on a $max - min$ operation as portrayed above. We note instead of min operation, any t-norm (see [17]) could be employed; similarly, instead of max operation, any t-conorm (see [17]) could be employed [14], [18], [19], [20].

A generalized trust network can be defined as follows:

Definition 10 (Generalized Trust Network)**.** *Let $\mathcal{T} = \langle Ags, \tau, tr\rangle$ be a trust network. The generalized trust network $\mathcal{T}^*$ from $\mathcal{T}$ is defined as*

$$\mathcal{T}^* = \langle Ags, \tau^*, tr^*\rangle.$$

In Fig. 2 we show a trust network borrowed from [10]. The solid lines are direct trust relationships, the dashed lines are indirect links derived from the direct links. Thus John trusts Jane with degree $0.7 = min[0.9, 0.7]$ and Dave with degree $0.8 = min[0.9, 0.8]$ because he trusts Mary with degree 0.9 and Mary trusts Jane with degree 0.7 and Dave with degree 0.8. However, John does not, even indirectly, trust Alice.

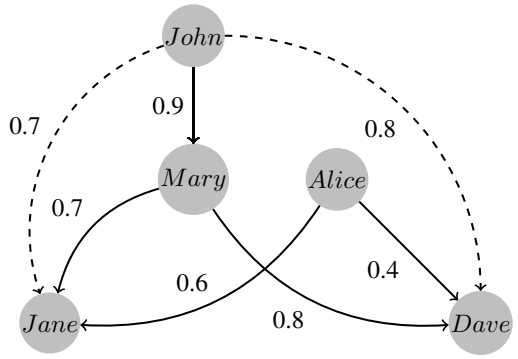

Fig. 2: (Generalized) Trust Network

IV. A Trust Network for Possibilistic Answer Set Programs

In the previous section, we have developed an approach to model how an agent Ag_i can reason about the trust in other agents according to a trust network. Essentially we have assumed trust is a transitive relation and employed a kind of $max - min$ operator to assess the degree of trust assigned by an agent to another. Nonetheless, this approach is general enough to consider other properties to propagate trust (besides transitivity) and other operations on degree of trust (besides $max - min$). Having defined how an agent Ag_i can reason *about* the trust-worthiness of its acquaintances, now we are interested in how an agent Ag_i can reason *with* the trust-worthiness of its acquaintances. In order to attain such an objective, we will show how an agent Ag_i can take profit of its trust information to empower its own knowledge base with its acquaintances's knowledge bases.

We will formalize this approach by assuming that the knowledge base of each agent Ag_i in a trust network $\mathcal{T}$ is represented by a Possibilistic Answer Set Program under the answer set semantics disclosed in Section II. Then we will resort to the trust information of Ag_i to incorporate its acquaintances's Possibilistic Answer Set Programs in a proper way:

Definition 11 (Knowledge Base)**.** *Let $\mathcal{T} = \langle Ags, \tau, tr\rangle$ be a trust network. We define the knowledge base of an agent Ag_i simply as a Possibilistic Answer Set Program, denoted by P_i.*

An agent in $\mathcal{T}$ can also resort to its acquaintances's knowledge bases:

Definition 12 (Acquaintances' Knowledge Base)**.** *Let $\mathcal{T} = \langle Ags, \tau, tr\rangle$ be a trust network, and P_i and P_j be respectively the knowledge bases of agents Ag_i and Ag_j. We define the knowledge base P_i^j in the following manner:*

$P_i^j = \{\mathbf{d} : l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n \mid$
$\mathbf{c} : l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n \in P_j$ *and* $\mathbf{d} = min\{\mathbf{c}, tr^*(Ag_i, Ag_j)\}\}$.

The agent i's acquaintances' knowledge base is given by

$$P_i^{\mathcal{T}} = \bigcup_{j \in Ags, j \neq i} P_i^j.$$

The knowledge base of each acquaintance Ag_j of Ag_i is subjected to $tr^*(Ag_i, Ag_j)$. If Ag_i trusts Ag_j with a low degree, it will reflect on little confidence on Ag_j's knowledge base. Besides the min operator employed in Definition 12, more sophisticated operators can be exploited as well as the relations of this definition and the properties of the network. Now we will combine Ag_i's knowledge base with its acquaintances' knowledge bases to characterize the Ag_i's profile:

Definition 13 (Profile)**.** *Let $\mathcal{T} = \langle Ags, \tau, tr \rangle$ be a trust network. The agent's profile Δ_i of $Ag_i \in Ags$ w.r.t. $\mathcal{T}$ is*

$$\Delta_i^{\mathcal{T}} = P_i \cup P_i^{\mathcal{T}}.$$

An agent's profile is a Possibilistic Answer Set Program and as such it can be interpreted under the Possibilistic Answer Set Semantics. Next we will see how to employ this semantics in an example to reason with trust network.

V. A Running Example

In the sequel, we will show a simple, albeit comprehensive, example portraying these reasoning and calculations settled on our approach. Suppose we have a machine, or agent, that is running a decision-making software based on the trust network $\mathcal{T}$ displayed in Fig. 3 for Possibilistic Answer Set Programs:

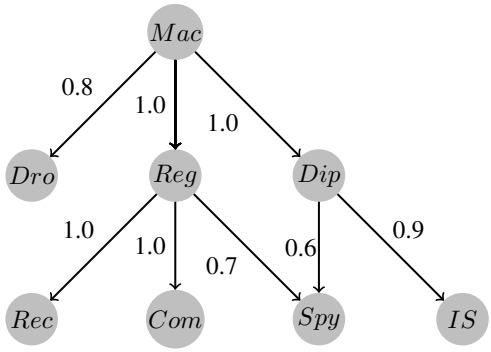

Fig. 3: Military Trust Network

The machine (Mac) belongs to the military forces of a nation and the decision is to send a missile, or not, to an enemy base location. Interpreting the machine as an agent in a trust network, we suppose it has the knowledge base

$$P_{Mac} = \begin{cases} \mathbf{1.0} : Safe \leftarrow ManyEnemies, not\ \neg Safe \\ \mathbf{1.0} : Shoot \leftarrow Safe, Ammo \\ \mathbf{1.0} : \neg Shoot \leftarrow \neg Safe \\ \mathbf{1.0} : Ammo \leftarrow \top \end{cases}$$

In other words, the mission is safe if there are many enemies in the area and there is no evidence it is unsafe; if it is both safe and ammunition is available, a shot can be taken; if it is unsafe, no shot should be taken. The machine also knows it has ammunition. Clearly, the machine needs more information to reach any conclusion. Indeed, as portrayed in the trust network $\mathcal{T}$, we know Mac has a direct relation with agents Drone (Dro), Regiment (Reg) and Diplomatic Mission (Dip), whose degrees of trust are respectively 0.8 (due to malfunctioning probability), 1.0 and 1.0. The Drone was conceived to check the presence of many enemies and if the area is safe for an attack. It believes that if enemies are confirmed and cars are found, there are many enemies, and that if there are hostages in the area and no evidence it is safe, then it is not safe to shoot. Besides, the Drone spots objects that may be cars (with degree 0.6). It has as knowledge base

$$P_{Dro} = \begin{cases} \mathbf{1.0} : \neg Safe \leftarrow Hostage, not\ Safe \\ \mathbf{1.0} : ManyEnemies \leftarrow Cars, Enemy \\ \mathbf{0.6} : Cars \leftarrow \top \end{cases}$$

The Regiment and the Diplomatic Mission obtain all the information they need from their acquaintances and have $P_{Reg} = P_{Dip} = \emptyset$. Agent Reg has a direct relation with a Reconnaissance Team (Rec), a Combat Team (Com) and Spies (Spy) infiltrated into enemy-held territory. The respective degrees of trust are 1.0, 1.0 and 0.7. Concerning their knowledge bases, we have

$$P_{Rec} = \begin{cases} \mathbf{1.0} : Enemy \leftarrow Campfire \\ \mathbf{0.8} : Hostage \leftarrow Screams, Enemy, not\ Deal \\ \mathbf{0.9} : Campfire \leftarrow \top \end{cases}$$

$$P_{Com} = \{\ \mathbf{1.0} : Combat \leftarrow Enemy, not\ Hostage$$

$$P_{Spy} = \begin{cases} \mathbf{0.9} : \neg Safe \leftarrow Combat, not\ Safe \\ \mathbf{1.0} : Enemy \leftarrow \top \\ \mathbf{0.2} : Screams \leftarrow \top \end{cases}$$

The Reconnaissance Team believes that the presence of campfires means presence of enemies, and that if you have enemies around, hear screams, and there is no evidence of a Peace Deal, then there is a great possibility of hostages in the area. This team also has found traces of campfires with belief degree 0.9. The combat team simply reacts with combat if there are enemies and no evidence of hostages. The Spies have reported there is great evidences that it is not safe if there is a combat in the area. They have also reported the presence of enemies and a vague feeling of hearing screams.

Agent Dip has a direct relation with the group of Spies (Spy) and the Intelligence Service (IS), whose degrees of trust are respectively 0.6 and 0.9. The Intelligence Service has wired that if the enemies have no evidence their base location would be attacked and there is no evidence of hostages, then the enemies will seal a peace deal with belief degree 0.9.

$$P_{IS} = \{\ \mathbf{0.9} : Deal \leftarrow not\ Shoot, not\ Hostage$$

The Machine then starts trying to come up with a decision by building up its profile $\Delta_{Mac} =$

$$P_{Mac} \cup P_{Mac}^{Dro} \cup P_{Mac}^{Reg} \cup P_{Mac}^{Dip} \cup P_{Mac}^{Rec} \cup P_{Mac}^{Com} \cup P_{Mac}^{Spy} \cup P_{Mac}^{IS}.$$

based on its trust network $\mathcal{T}$. We obtain Δ_{Mac} has the following Possibilistic Answer Sets:

$$S_1 = \mathcal{C} \cup \{(Safe, 0.6), (Shoot, 0.6), (Hostage, 0.2)\}.$$
$$S_2 = \mathcal{C} \cup \{(\neg Safe, 0.7), (\neg Shoot, 0.7), (Deal, 0.9), (Combat, 0.9)\}.$$
$$S_3 = \mathcal{C} \cup \{(\neg Safe, 0.2), (\neg Shoot, 0.2), (Hostage, 0.2)\}.$$

in which $\mathcal{C} = \{(Ammo, 1.0), (Cars, 0.6), (Campfire, 0.9), (Enemy, 0.9), (ManyEnemies, 0.6), (Screams, 0.2)\}$. Hence, the machine has two alternatives: on one hand, it is quite certain that it can shoot with safety; on the other hand, the available information has ground enough to justify that it is not safe and it would not shoot. In this situation, depending on the algorithm's approach, the machine will or will not take the shot. For instance, geopolitical concerns can justify to shoot even though its degree 0.6 is lower than not to shoot's highest (degree 0.7). In a cautious approach, one can compare the lowest calculable values of both propositions as in a worst case scenario of each; in our example, it could be an interesting choice, meaning the shot would still be made. You can go more lenient by choosing the one with the best average, or less, as in a high-risk situation with lives on the line, and picking $Shoot$ only if no $\neg Shoot$ can be inferred with any degree of belief. Anyway, no matter the approach we take to reason with uncertainty and to translate the output as "yes" or "no", it is clear why we should obtain and keep every possibilistic answer set.

VI. Conclusion and Future Works

It is clear the important role trust can play in multiagent systems where both the knowledge bases are distributed and the control in those systems are decentralized. With that motivation in mind, in this paper, we have extended the framework for Possibilistic Answer Set Programs to reason with (and about) trust in a multiagent scenario. Hence, the resulting framework is general enought to deal with incomplete (non monotonic), uncertain knowledge representation and reasoning based on a trust network.

In order to achieve this objective, we have adapted an approach for general systems of argumentation introduced in [10] to represent reasoning about trust and belief in possibilistic answer set programming. One of its distinguishing aspects is the unified way it relates the uncertainty associated with trust to that associated with possibilistic theory by integrating degrees of trust with the certainty weights connected to each rule of a possibilistic answer set program.

A notable feature of our proposal is inherited from [10]: flexibility. In this paper, we have defined trust as a kind of transitive relation and have resorted to this property to propagate trust through the trust network. However, new properties to propagating trust can be added (or even removed) by changing the notion of Generalized Trust Relation (Definition 8). Besides, as we illustrated in Section V, in order to reason with both uncertainty and trust, we keep every possibilistic answer set. Thus, the agent will have at hand all the available information to take its decision, and depending on the context, it can follow a more (or less) cautious approach.

Regarding future works, we envisage to exploit two paths: on one hand, we will implement a software system that employs this approach and provide a complexity analysis; on the other hand, we plan to consider alternative views on interpreting the uncertainty attached to rules of a $PASP$; a natural development is to resort to probability to model uncertainty instead of possibility theory. Besides considering rules with uncertain conclusions, we can also attach an uncertainty to the rule itself, resulting in a more powerful and flexible $PASP$.

Acknowledgment

The authors would like to thank CNPq for the financial support.

References

[1] M. Gelfond and V. Lifschitz, "Classical negation in logic programs and disjunctive databases," *New generation computing*, vol. 9, no. 3-4, pp. 365–385, 1991.

[2] V. W. Marek and M. Truszczyński, "Stable models and an alternative logic programming paradigm," in *The Logic Programming Paradigm*. Springer, 1999, pp. 375–398.

[3] I. Niemelä, "Logic programs with stable model semantics as a constraint programming paradigm," *Annals of Mathematics and Artificial Intelligence*, vol. 25, no. 3-4, pp. 241–273, 1999.

[4] V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence*, vol. 138, no. 1-2, pp. 39–54, 2002.

[5] `http://potassco.sourceforge.net`, [Online; accessed 02-June-2016].

[6] `http://www.dbai.tuwien.ac.at/proj/dlv`, [Online; accessed 02-June-2016].

[7] C. Anger, K. Konczak, T. Linke, and T. Schaub, "A glimpse of answer set programming." *Künstliche Intelligenz*, vol. 19, no. 1, p. 12, 2005.

[8] P. Nicolas, L. Garcia, I. Stéphan, and C. Lefèvre, "Possibilistic uncertainty handling for answer set programming," *Annals of Mathematics and Artificial Intelligence*, vol. 47, no. 1-2, pp. 139–181, 2006.

[9] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy sets and systems*, vol. 1, no. 1, pp. 3–28, 1978.

[10] Y. Tang, K. Cai, P. McBurney, E. Sklar, and S. Parsons, "Using argumentation to reason about trust and belief," *Journal of Logic and Computation*, vol. 22, no. 5, pp. 979–1018, 2012.

[11] K. Bauters, S. Schockaert, M. De Cock, and D. Vermeir, "Semantics for possibilistic answer set programs: uncertain rules versus rules with uncertain conclusions," *International Journal of Approximate Reasoning*, vol. 55, no. 2, pp. 739–761, 2014.

[12] S. Parsons, E. Sklar, and P. McBurney, "Using argumentation to reason with and about trust," in *Argumentation in Multi-Agent Systems*. Springer, 2011, pp. 194–212.

[13] A. Jøsang, C. Keser, and T. Dimitrakos, "Can we manage trust?" in *Trust management*. Springer, 2005, pp. 93–107.

[14] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 403–412.

[15] B. Christianson and W. S. Harbison, "Why isn't trust transitive?" in *Security protocols*. Springer, 1996, pp. 171–176.

[16] S. Parsonsa, K. Atkinsonb, K. Haighc, K. Levittd, P. M. J. Rowed, M. P. Singhf, and E. Sklara, "Argument schemes for reasoning about trust," *Computational Models of Argument: Proceedings of COMMA 2012*, vol. 245, p. 430, 2012.

[17] P. Hájek, *Metamathematics of fuzzy logic*. Springer Science & Business Media, 1998, vol. 4.

[18] Y. Katz and J. Golbeck, "Social network-based trust in prioritized default logic," in *AAAI*, vol. 6, 2006, pp. 1345–1350.

[19] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *The Semantic Web-ISWC 2003*. Springer, 2003, pp. 351–368.

[20] Y. Wang and M. P. Singh, "Trust representation and aggregation in a distributed agent system," in *AAAI*, vol. 6, 2006, pp. 1425–1430.