Most-Probable: A New Argumentation Semantics through Optimization

Davide DI PIERRO[1]

davide.dipierro@uniba.it

Stefano FERILLI[1]

stefano.ferilli@uniba.it

[1]Università degli Studi di Bari

Outline

Introduction

- Argumentation has origin from philosophy, but has gathered interests in many fields (e.g. law).
- We refer to the Dung's formalization [2,3].
- Extensions to represent uncertainty are available.
- Representation techniques are often borrowed from the KRR field.

[2] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial intelligence 77 (1995) 321–357.
[3] A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, Artificial intelligence 93 (1997) 63–101.

Answer Set Programming (ASP)

An Answer Set Program is a program made up of fact(s) like

$$p(t_1, \dots, t_p).$$

and rule(s) like

$$H :- A_1, \dots, A_n, \text{not } B_1, \dots, \text{not } B_{m.}$$

Special rule(s) have no head (constraints).

$$:- A_1, \dots, A_n, \neg B_1, \dots, \neg B_{m.}$$

ASP supports aggregates ($\text{sum}, \text{count}, \dots$) and optimizations ($\text{optimize}$).

Objectives

- We propose a strategy for updating beliefs in an argumentation network based on attack relationships.

- We propose a new semantics (*most-probable*) that deals with degrees of belief.

- This semantics focuses on a *target* argument, trying to collect all the arguments that justify (or defend) the target maximizing degrees of belief.

- We integrated the belief updating and the semantics computation with an existing Prolog-based system for argumentation.

Background

- **Definition** An Argumentation Framework (AF) is a pair $\langle Args, Att \rangle$ where $Args$ is a (finite) set of arguments and $Att \subseteq Args \times Args$ the attack relationship. The concept of attack can also be extended to a set of arguments. Given $S \subseteq Args$, $A \in Args$ attacks S iff $exists$ $B \in S$ such that A attacks B.

Background

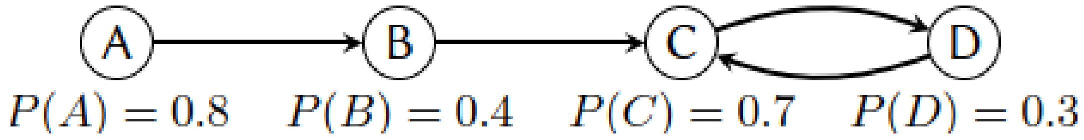
- **Definition** Let an AF $G = \langle Args, Att \rangle$ be an AF, a set of arguments $S \subseteq Args$ is **conflict-free** iff $\langle A, B \rangle \in S$ such that $\langle A, B \rangle \in Att$. The set of all conflict-free extensions is indicated as $\mathrm{cf}(G)$.

Background

- **Definition** Let an $AF\ G = \langle Args, Att \rangle$ be an AF, a set of arguments $S \subseteq Args$ is **admissible** iff S is conflict-free and $\forall \gamma \in Args\ \gamma$ attacks $S \Rightarrow S$ attacks γ. In other words, S defends every $A \in S$.

Simplified Probabilistic Argumentation Framework

- Probabilities come under the assumption of non-additivity, meaning that it is not impossible that $\Delta \nvDash \alpha$ and $\Delta \nvDash \neg\alpha$.
- Attacks always trigger with the same intensity.
- Probabilities of arguments cannot be compared with Kolmogorov Axioms.
- **Definition** Let an $AF\ G = \langle Args, Att \rangle$, a Simplified Probabilistic Argumentation Framework (SPAF) $\langle Args, Att, PArgs \rangle$ is a triple in which $Args$ and Att are defined as above, $PArgs : Args \rightarrow]0, 1]$, a function indicating the likelihood of arguments.



Probability Update

- Probabilities of arguments are "blind" with respect to other arguments and attacks.
- We update the probabilities of arguments, based on attacks and the initial probability of arguments.
- The proposed formula is

$$P'(A) = P(A) \cdot \prod_{\substack{\gamma \in Args \\ \langle \gamma, A \rangle \in Att}} 1 - \alpha \cdot P(\gamma)$$

where $P'(A)$ represents the probability of A after being attacked and $\alpha \in]0, 1]$.

Probability Update

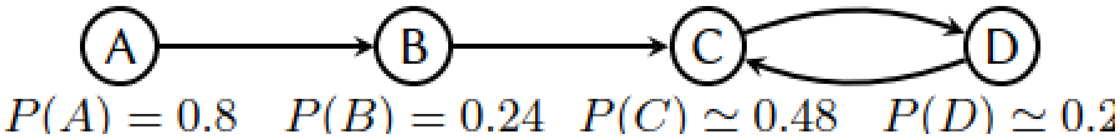
- The nice properties are:
  - multiple attacks influence much the updated belief.
  - the product $1 - \alpha \cdot P(\gamma)$ is always between 0 and 1.
  - the initial belief of an argument is an upper bound for the update.
  - many weak (low $P(\gamma)$) attackers are less influential than few strong (high $P(\gamma)$) attackers.

$$P'(A) = P(A) \cdot \prod_{\substack{\gamma \in Args \\ \langle \gamma, A \rangle \in Att}} 1 - \alpha \cdot P(\gamma)$$

Probability Update

- The main limitations are:
  - if an attacker has probability 1, the attacked is not vanished.
  - the function is nonlinear, meaning that small changes have large effects.
  - it is assumed attacks are independent.

$$P'(A) = P(A) \cdot \prod_{\substack{\gamma \in Args \\ \langle \gamma, A \rangle \in Att}} 1 - \alpha \cdot P(\gamma)$$



Probability Update

- We may also consider an iterative updating probability step.

$$P^n(A) = \begin{cases} P(A) & \text{if } n = 0, \\ P^{n-1}(A) \cdot U^n(A) & otherwise \end{cases}$$

where

$$U^i(A) = P^{i-1}(A) \cdot \prod_{\substack{\gamma \in Args \\ \langle \gamma, A \rangle \in Att}} 1 - \alpha \cdot P^{i-1}(\gamma).$$

The proposed formula resembles one of those proposed by Gabbay et al. [11]. However, the starting probabilities of the arguments were not taken into account.

[11] D. M. Gabbay, Equational approach to argumentation networks, Argument & Computation 3 (2012) 87–142.

Most-Probable Semantics

- **Definition** Let $\langle Args, Att, P\ Args \rangle$ G a Simplified Probabilistic Argumentation Framework and $t \in Args$, a set of arguments $S \subseteq Args$ is a most-probable extension for t, indicated as $S \in \text{most-prob}_t(G)$ iff $t \in S$, S is admissible and $\nexists Y \subset S$ such that:
  - $t \in Y$
  - Y is admissible
  - $P(Y \setminus \{t\}) > P(S \setminus \{t\})$.

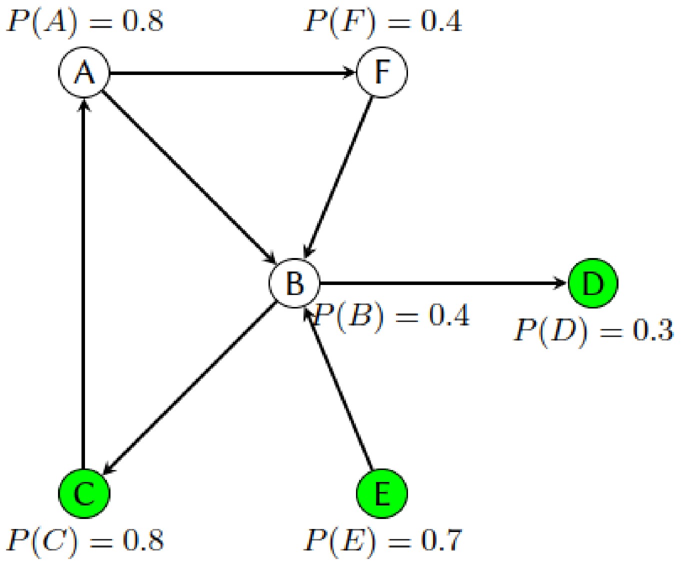
  and $\nexists Z \subseteq Args$ such that:
  - $S \subset Z$
  - Z is admissible
  - $P(Z \setminus \{t\}) = P(S \setminus \{t\})$.

$$P(\{ a_1, \dots, a_n \}) = \min \{ P(a_1), \dots, P(a_n) \}$$

Most-Probable Semantics

- **Lemma** There is not a unique solution for the most-probable extension.

  Proof. By construction, suppose G a SPAF with $Args = \{a_1, a_2, ..., a_n, s_0, s_1, t\}$ where $\{a_1, a_2, ..., a_n, t\}$ are not attacked by anyone, and s_0, s_1 attack each other. Then, both $\{a_1, a_2, ..., a_n, s_0, t\}$ and $\{a_1, a_2, ..., a_n, s_1, t\}$ are $\text{most-prob}_t(G)$.



δ-most-Probable Semantics

- **Definition** Let $\langle Args, Att, P\ Args\rangle$ G a Simplified Probabilistic Argumentation Framework, $t \in Args$ and $\delta \in \mathrm{R}_{>}$, a set of arguments $S \subseteq Args$ is a δ-most-probable extension with respect to t iff S is most-probable with respect to t and $P(S \setminus \{t\}) >= \delta$.

  $\delta = 0$ would reduce threshold most-probable to most-probable.

Implementation

➢ We developed this probabilistic argumentation framework in a platform for argument reasoning called ARGuing Using Enhanced Reasoning (Arguer).

```
  __ _ _ __ __ _ _   _  ___ _ __
 / _` | '__/ _` | | | |/ _ \ '__|
| (_| | | | (_| | |_| |  __/ |
 \__,_|_|  \__, |\__,_|\___|_|
           |___/
ARGuing Using Enhanced Reasoning v2.0
(C) 2018, Universita' degli studi di Bari Aldo Moro
Department of Computer Science - Dr. Andrea Pazienza, PhD

       1 - Abstract Argumentation Framework (AF)
       2 - Value-Based AF (VAF)
       3 - Bipolar AF (BAF)
       4 - Weighted AF (WAF)
       5 - Bipolar Weighted AF (BWAF) Strength Propagation Ranking Semantics
       6 - Simplified Probabilistic AF (SPAF)
       0 - Quit

Enter your choice (0-6):
```

Implementation

- Arguer provides the following argumentation frameworks:
  - Abstract Argumentation Framework (AF)
  - Value-Based AF
  - Bipolar AF
  - Weighted AF
  - Bipolar Weighted AF
  - Simplified Probabilistic AF

Implementation (Probability Update)

```
update(Arguments, Attacks, Likelihoods) :-
    adjust_likelihoods(Arguments, Attacks, Likelihoods), normalize_probabilities.

adjust_likelihoods([], _, _).
adjust_likelihoods([Arg | Rest], Attacks, Likelihoods) :- adjust_likelihood(Arg, Attacks, Likelihoods),
    adjust_likelihoods(Rest, Attacks, Likelihoods).

adjust_likelihood(Arg, Attacks, Likelihoods) :-
    findall(X, lists:member([X,Arg], Attacks), Attackers),
    findall(X, (lists:member(A, Attackers),  lists:member([A,X], Likelihoods)), ProbabilitiesAttackers),
    lists:member([Arg, OriginalProbability], Likelihoods), Alfa is 0.2,
    compute_product(Alfa, ProbabilitiesAttackers, Product), AdjustedProbability is Product * OriginalProbability,
    assertz(not_normalized_probability(Arg, AdjustedProbability)).

compute_product(Alfa, [], 1) :-
    !.
compute_product(Alfa, [P | Rest], Product) :-
    TermProduct is 1 - Alfa * P, compute_product(Alfa, Rest, OldProduct), Product is TermProduct * OldProduct.

find_max([X], X).
find_max([Number | Rest], Max) :-
    find_max(Rest, MaxRest), Max is max(Number, MaxRest).

normalize_probabilities :-
    findall(X, not_normalized_probability(_,X), Probabilities), find_max(Probabilities, Max),
    Factor is 100 / Max, findall((Arg, Probability),
    not_normalized_probability(Arg, Probability), ArgProbabilities), assert_probabilities(ArgProbabilities, Factor).

assert_probabilities([], Factor) :-
    !.
assert_probabilities([ArgProbability | Rest], Factor) :-
    ArgProbability = (Argument, Probability), NormalizedProbability is round(Factor * Probability),
    assertz(probability(Argument, NormalizedProbability)), assert_probabilities(Rest, Factor).
```

Implementation (Most-Probable)

```
argument(a;b;c;d;e;f).
target(d).
attack(a, b). attack(a, b).
attack(b, c). attack(b, d).
attack(c, a). attack(e, b).
attack(f, b).
likelihood(a, 80).
likelihood(b, 40).
likelihood(c, 80).
likelihood(d, 30).
likelihood(e, 70).
likelihood(f, 40).

% the target is always in the most-prob
most_prob(N) :- target(N).
%all the others are candidate
{most_prob(N) : argument(N)} :- not target(N).
%arguments must be defended (or attacks free) to be good candidates
:- most_prob(A), attack(B, A), not most_prob(C) : attack(C, B).
%take minimum likelihood
min_likelihood(Min) :- Min = #min { L : most_prob(A), likelihood(A, L), not target(A) }.
%take cardinality
size_most_prob(C) :- C = #count { A : most_prob(A) }.
%take set with the maximum minimum likelihood
#maximize { L : min_likelihood(L) }.
%add as many arguments as possible without affecting the minimum likelihood
#maximize { C : size_most_prob(C) }.
#show most_prob/1.
```

Conclusions

- We proposed:
  - a new representation for probabilistic argumentation frameworks.
  - a new semantics, with implementation and interpretation of it.

- Much research can be pursued in:
  - mitigating the current limitations of probability updates.
  - proposing new interpretations for initial probabilities.
  - suitable applications.