Computational Logic

Argumentation

Martin Baláž

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

2011

Defeasible Logic Program:

$$
\begin{aligned}
penguin(X) &\rightarrow bird(X) \\
supernatural\_penguin(X) &\rightarrow penguin(X) \\
bird(X) &\Rightarrow fly(X) \\
penguin(X) &\Rightarrow \neg fly(X) \\
supernatural\_penguin(X) &\Rightarrow fly(X)
\end{aligned}
$$

$$
\begin{aligned}
&\rightarrow bird(tweety) \\
&\rightarrow penguin(skippy) \\
&\rightarrow supernatural\_penguin(rocky)
\end{aligned}
$$

Argumentation Process

1. Constructing arguments
2. Conflicts between arguments
3. Comparing arguments
4. The status of arguments

Defeasible Logic Program

A literal is either an atom or a negated atom.

A *strict rule* is a formula of the form

$$L_1, \ldots, L_n \rightarrow L_0$$

where $n \geq 0$ and L_i, $0 \leq i \leq n$, are literals.

A *defeasible rule* is a formula of the form

$$L_1, \ldots, L_n \Rightarrow L_0$$

where $n \geq 0$ and L_i, $0 \leq i \leq n$, are literals.

A *defeasible logic program* is a set of strict and defeasible rules.

Argument

Let P be a defeasible logic program. An *argument* is

- $[A_1, \ldots, A_n \rightarrow L]$ if $A_1, \ldots, A_n$ are arguments and there exists a strict rule $r\colon Conc(A_1), \ldots, Conc(A_n) \rightarrow L$ in $Ground(P)$.

$$\begin{aligned} Conc(A) &= L \\ Concs(A) &= Concs(A_1) \cup \cdots \cup Concs(A_n) \cup \{L\} \\ SubArgs(A) &= SubArgs(A_1) \cup \cdots \cup SubArgs(A_n) \cup \{A\} \\ DefRules(A) &= DefRules(A_1) \cup \cdots \cup DefRules(A_n) \end{aligned}$$

- $[A_1, \ldots, A_n \Rightarrow L]$ if $A_1, \ldots, A_n$ are arguments and there exists a defeasible rule $r\colon Conc(A_1), \ldots, Conc(A_n) \Rightarrow L$ in $Ground(P)$.

$$\begin{aligned} Conc(A) &= L \\ Concs(A) &= Concs(A_1) \cup \cdots \cup Concs(A_n) \cup \{L\} \\ SubArgs(A) &= SubArgs(A_1) \cup \cdots \cup SubArgs(A_n) \cup \{A\} \\ DefRules(A) &= DefRules(A_1) \cup \cdots \cup DefRules(A_n) \cup \{r\} \end{aligned}$$

Conflicts

An argument A *attacks* an argument B iff $Conc(A) = \neg\, Conc(B)$.

An argument A *defeats* an argument B iff there exist $A' \in SubArgs(A)$ and $B' \in SubArgs(B)$ such that A' attacks B' and $B' \not\prec A'$.

An argument A *strictly* defeats an argument B iff A defeats B and B does not defeat A.

Comparing Arguments

Preferences on rules

- Strict rules preferred over defeasible rules.
- Informations from more reliable source preferred over information from less reliable source.
- Newer information preferred over older information.
- ...

Preferences on arguments

- Arguments containing only strict rules are preferred over arguments containing a defeasible rule.
- Specific arguments preferred over general arguments.
- Arguments are compared with respect to the last used defeasible rules.
- Arguments are compared with respect to the weakest used defeasible rule.

Characteristic Function

An argument A is *acceptable with respect to* a set of arguments S iff each argument defeating A is strictly defeated by an argument from S.

Let P be a defeasible logic program. The characteristic function F_P is defined as follows:

$$F_P(S) = \{A \in Args_P \mid A \text{ is acceptable with respect to } S\}$$

The iteration of a characteristic function is defined as follows:

$$\begin{aligned} F_P \uparrow 0 &= \emptyset \\ F_P \uparrow (n+1) &= F_P(F_P \uparrow n) \\ F_P \uparrow \omega &= \bigcup_{n<\omega} F_P \uparrow n \end{aligned}$$

An argument is *justified* if it is in the least fixpoint of F_P.

Properties

A defeasible logic program P is *finitary* iff each argument in $Args_P$ is attacked by at most finite number of arguments in $Args_P$.

Let $JustArgs_P$ be the set of all justified arguments of a defeasible logic program P. Then $F_P \uparrow \omega \subseteq JustArgs_P$. If P is finitary, then $JustArgs_P \subseteq F_P \uparrow \omega$.

Dialog

A *move* is a pair $\mu = (Player, Argument)$ where $Player \in \{Proponent, Oponent\}$ and *Argument* is an argument. We will denote $player(\mu) = Player$ and $argument(\mu) = Argument$.

A *dialog* is a finite non-empty sequence of moves $\mu_0, \mu_1, \ldots, \mu_n$, $n > 0$, where

- $player(\mu_0) = Proponent$ and $player(\mu_{i+1}) \neq player(\mu_i)$
- if $player(\mu_i) = player(\mu_j)$ for $i \neq j$, then $argument(\mu_i) \neq argument(\mu_j)$
- if $player(\mu_{i+1}) = Proponent$, then $argument(\mu_{i+1})$ strictly defeats $argument(\mu_i)$
- if $player(\mu_{i+1}) = Oponent$, then $argument(\mu_{i+1})$ defeats $argument(\mu_i)$

Dialog Tree

A *dialog tree* is a finite tree such that

- nodes are moves
- each branch is a dialog
- if $\mathit{player}(\mu) = \mathit{Proponent}$ for a node μ, then for all defears A of $\mathit{argument}(\mu)$ holds $(\mathit{Oponent}, A)$ is a child of μ.

A player *wins a dialog* iff the other player cannot move.
A player *wins a dialog tree* iff it wins all branches of the tree.

Properties

An argument A is *provably justified* if there exists a dialog tree with root $(Proponent, A)$ won by $Proponent$.
A literal L is *provably justified* if it is a *conclusion* of a provably justified argument.

All provably justified arguments are justified.

For finitary argumentation framework, justified arguments are provably justified.