Representing Argumentation Frameworks in Answer Set Programming

Chiaki Sakama (Wakayama Univ. Japan)

Tjitze Rienstra (Univ. of Luxembourg)

ArgLP, August 2015, Cork

Transformation from AF to LP
representational viewpoint

- **meta-interpretative representation**

  individual AFs are given as input to a single metalogic program which produces canonical (or selected) models characterizing input AF semantics.

- **object-level representation**

  individual AFs are transformed to corresponding logic programs whose canonical models characterize input AF semantics.

! Encoding AF semantics in meta-interpretative LP often results in complicated programs.

Transformation from AF to LP
semantical viewpoint

- **extension-based semantics**

  extensions of an AF are characterized by canonical models of a transformed logic program.

- **labelling-based semantics**

  labellings of an AF are characterized by canonical models of a transformed logic program.

! Extension-based semantics does not distinguish rejected arguments and undecided arguments.

Transformation from AF to LP
transformational viewpoint

- **one-to-one mapping**

  different semantics of an AF are characterized by different semantics of a transformed LP.

- **many-to-one mapping**

  different semantics of an AF are characterized by a single semantics of a transformed LP.

! Many-to-one mapping enables one to use a single LP solver for computing different semantics of AF.

Transformation from AF to LP

Existing Studies

Studies	representation	semantics	transformation
Dung (1995)	meta-interpretative	extension	stable ext. → stable model grounded ext. → well-founded model
Nieves, et al. (2008)	object level	extension	preferred ext. → stable model
Wu, et al. (2009)	object level	labelling	complete labelling → 3-valued stable model
Wakaki, et al. (2009)	meta-interpretative	labelling	complete/stable/grounded/preferred/ semi-stable labelling → ASP
Egly, et al. (2010)	meta-interpretative	extension	complete/stable/grounded/preferred/ ext. → ASP
Caminada, et al. (2015)	object level	labelling	stable/grounded/preferred/semi-stable labelling → stable/well-founded/ regular/L-stable model
Our current study	object level	labelling	complete/stable/grounded/preferred/ labelling → ASP

Preliminaries

- An **argumentation framework**: $AF=(Ar,att)$.
- For $x \in Ar$, $x^{-}=\{ y \mid (y, x) \in att \}$.
- **Labelling** $L: Ar \rightarrow \{ in, out, und \}$
- When $L(a)=in$ (resp. out or und) for $a \in Ar$, it is written as $in(a)$ (resp. $out(a)$ or $und(a)$) (called **labelled arguments**).
- **Complete labelling**, **stable labelling**, **grounded labelling**, and **preferred labelling** are defined as usual.
- A **logic program** consists of rules:

  $$a_1 \vee \cdots \vee a_l \leftarrow a_{l+1},\ldots,a_m, \textbf{not } a_{m+1},\ldots,\textbf{not } a_n$$

  where a_i :ground atom, **not**:negation as failure
- The semantics of a program is given by **stable models** (or **answer sets**).

LP Rules for AF

- Given $AF=(Ar,att)$, the **Herbrand base** B is defined as

  $$B=\{\ in(x),\ out(x),\ und(x) \mid x \in Ar\ \}.$$

- The set Γ_{AF} of rules is defined as follows:

$$\Gamma_{AF} = \{\ in(x) \leftarrow out(y_1),\ldots,out(y_k) \mid x \in Ar \text{ and } x^- = \{y_1,\ldots,y_k\}\ (k \geqq 0)\ \}$$

$$\cup\ \{\ out(x) \leftarrow in(y) \mid (y,x) \in att\ \}$$

$$\cup\ \{\ \leftarrow in(x), \textbf{not}\ out(y) \mid (y,x) \in att\ \}$$

$$\cup\ \{\ \leftarrow out(x), \textbf{not}\ in(y_1),\ldots,\textbf{not}\ in(y_k) \mid x \in Ar \text{ and } x^- = \{y_1,\ldots,y_k\}\ (k \geqq 0)\ \}$$

AF program under complete semantics

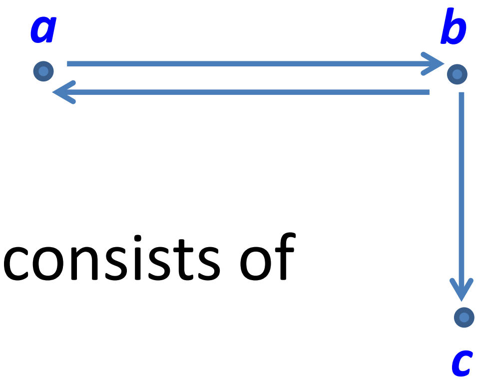
Given $AF=(Ar,att)$, an **AF-program under the complete semantics** Π^{C}_{AF} is defined as follows:

$$\Pi^{C}_{AF} = \Gamma_{AF} \cup \{\, in(x) \vee out(x) \vee und(x) \leftarrow \mid x \in Ar \,\}$$
$$\cup \{\, \leftarrow in(x), out(x) \mid x \in Ar \,\}$$
$$\cup \{\, \leftarrow in(x), und(x) \mid x \in Ar \,\}$$
$$\cup \{\, \leftarrow out(x), und(x) \mid x \in Ar \,\}$$

Theorem

The sets of labelled arguments under the complete semantics of *AF* coincide with the stable models of Π^{C}_{AF}.

Example



- Given *AF*=({*a*,*b*,*c*},{(*a*,*b*),(*b*,*a*),(*b*,*c*)}), Π^{C}_{AF} consists of

  in(*a*)←*out*(*b*), *in*(*b*)←*out*(*a*), *in*(*c*)←*out*(*b*),

  out(*a*)←*in*(*b*), *out*(*b*)←*in*(*a*), *out*(*c*)←*in*(*b*),

  ←*in*(*a*), **not** *out*(*b*), ←*in*(*b*), **not** *out*(*a*), ←*in*(*c*), **not** *out*(*b*),

  ←*out*(*a*), **not** *in*(*b*), ←*out*(*b*), **not** *in*(*a*), ←*out*(*c*), **not** *in*(*b*),

  in(*x*) v *out*(*x*) v *und*(*x*) ← where *x*∈{*a*,*b*,*c*}

  ←*in*(*x*),*out*(*x*), ←*in*(*x*),*und*(*x*), ←*out*(*x*),*und*(*x*) where *x*∈{*a*,*b*,*c*}

- Π^{C}_{AF} has 3 stable models:

  {*in*(*a*),*out*(*b*),*in*(*c*)}, {*out*(*a*),*in*(*b*),*out*(*c*)}, {*und*(*a*),*und*(*b*),*und*(*c*)}

which are equivalent to 3 sets of labelled arguments under the complete semantics of *AF*.

AF program under stable semantics

Given $AF=(Ar,att)$, an **AF-program under the stable semantics** Π^{S}_{AF} is defined as follows:

$$\Pi^{S}_{AF} = \Gamma_{AF} \;\cup\; \{\, in(x) \vee out(x) \leftarrow \;|\; x \in Ar \,\}$$
$$\cup\; \{\, \leftarrow in(x), out(x) \;|\; x \in Ar \,\}$$

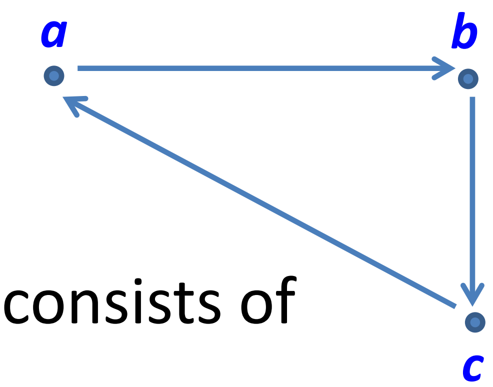
<u>Theorem</u>

The sets of labelled arguments under the stable semantics of *AF* coincide with the stable models of Π^{S}_{AF}.

<u>Corollary</u>

AF has no stable labelling iff Π^{S}_{AF} has no stable model.

Example



- Given $AF=(\{a,b,c\},\{(a,b),(b,c),(c,a)\})$, Π^{S}_{AF} consists of

$in(a) \leftarrow out(c)$, $in(b) \leftarrow out(a)$, $in(c) \leftarrow out(b)$,

$out(a) \leftarrow in(c)$, $out(b) \leftarrow in(a)$, $out(c) \leftarrow in(b)$,

$\leftarrow in(a)$, **not** $out(c)$, $\leftarrow in(b)$, **not** $out(a)$, $\leftarrow in(c)$, **not** $out(b)$,

$\leftarrow out(a)$, **not** $in(c)$, $\leftarrow out(b)$, **not** $in(a)$, $\leftarrow out(c)$, **not** $in(b)$,

$in(x) \vee out(x) \leftarrow$ where $x \in \{a,b,c\}$

$\leftarrow in(x), out(x)$ where $x \in \{a,b,c\}$

- Π^{S}_{AF} has no stable model, so AF has no stable labelling.

AF program under grounded semantics

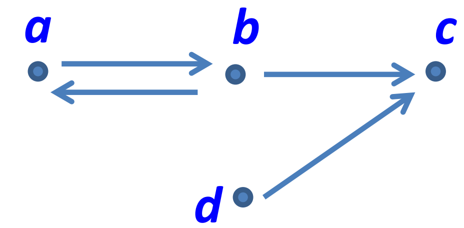
Given $AF=(Ar,att)$, an **AF-program under the grounded semantics** Π^{G}_{AF} is defined as follows:

$$\Pi^{G}_{AF} = \Gamma_{AF} \cup \{ und(x) \leftarrow \textbf{not } in(x), \textbf{not } out(x) \mid x \in Ar \}$$

Theorem

The set of labelled arguments under the grounded semantics of *AF* coincides with the stable model of Π^{G}_{AF}.

Example



- Given *AF*=({*a*,*b*,*c*,*d*},{(*a*,*b*),(*b*,*a*),(*b*,*c*),(*d*,*c*)}), Π^{G}_{AF} consists of

in(*a*)←*out*(*b*), *in*(*b*)←*out*(*a*), *in*(*c*)←*out*(*b*),*out*(*d*), *in*(*d*)←,

out(*a*)←*in*(*b*), *out*(*b*)←*in*(*a*), *out*(*c*)←*in*(*b*), *out*(*c*)←*in*(*d*),

←*in*(*a*), **not** *out*(*b*), ←*in*(*b*), **not** *out*(*a*), ←*in*(*c*), **not** *out*(*b*),

←*in*(*c*), **not** *out*(*d*), ←*out*(*a*), **not** *in*(*b*), ←*out*(*b*), **not** *in*(*a*),

←*out*(*c*), **not** *in*(*b*), **not** *in*(*d*), ←*out*(*d*),

und(*x*) ← **not** *in*(*x*), **not** *out*(*x*) where *x*∈{*a*,*b*,*c*,*d*}

- Π^{G}_{AF} has the unique stable model {*und*(*a*),*und*(*b*),*out*(*c*),*in*(*d*)} which is equivalent to the set of labelled arguments under the grounded semantics of *AF*.

AF program under preferred semantics

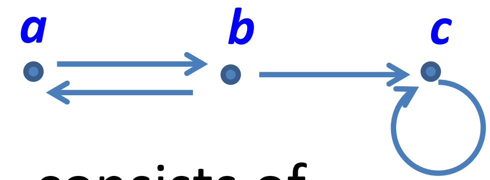
- Given $AF=(Ar,att)$, the **Herbrand base** B' is defined as
  $B'=\{ in(x), out(x), IN(x), OUT(x), UND(x) \mid x \in Ar \}$.
- Given $AF=(Ar,att)$, an **AF-program under the preferred semantics** Π^{P}_{AF} is defined as follows:

$$
\begin{aligned}
\Pi^{P}_{AF} = \Gamma_{AF} & \cup \{ in(x) \vee out(x) \leftarrow \mid x \in Ar \} \\
& \cup \{ IN(x) \leftarrow in(x), \textbf{not } out(x) \mid x \in Ar \} \\
& \cup \{ OUT(x) \leftarrow \textbf{not } in(x), out(x) \mid x \in Ar \} \\
& \cup \{ UND(x) \leftarrow in(x), out(x) \mid x \in Ar \}
\end{aligned}
$$

Theorem

There is a 1-1 correspondence between the sets of labelled arguments under the preferred semantics of *AF* and the stable models of Π^{P}_{AF}.

Example



- Given *AF*=({*a*,*b*,*c*},{(*a*,*b*),(*b*,*a*),(*b*,*c*),(*c*,*c*)}), Π^{P}_{AF} consists of

in(*a*)←*out*(*b*), *in*(*b*)←*out*(*a*), *in*(*c*)←*out*(*b*),*out*(*c*),
out(*a*)←*in*(*b*), *out*(*b*)←*in*(*a*), *out*(*c*)←*in*(*b*), *out*(*c*)←*in*(*c*),
←*in*(*a*), **not** *out*(*b*), ←*in*(*b*), **not** *out*(*a*), ←*in*(*c*), **not** *out*(*b*),
←*in*(*c*), **not** *out*(*c*), ←*out*(*a*), **not** *in*(*b*), ←*out*(*b*), **not** *in*(*a*),
←*out*(*c*), **not** *in*(*b*), **not** *in*(*c*), *in*(*x*) v *out*(*x*) ← where *x*∈{*a*,*b*,*c*}
IN(x)←*in*(*x*), **not** *out*(*x*), *OUT*(x)←**not** *in*(*x*), *out*(*x*),
UND(x)←*in*(*x*), *out*(*x*) where *x*∈{*a*,*b*,*c*}

- Π^{P}_{AF} has 2 stable models
  { *out*(*a*), *in*(*b*), *out*(*c*), *OUT*(*a*), *IN*(*b*), *OUT*(*c*) }
  { *in*(*a*), *out*(*b*), *in*(*c*), *out*(*c*), *IN*(*a*), *OUT*(*b*), *UND*(*c*) }

Then 2 sets {*OUT*(*a*), *IN*(*b*), *OUT*(*c*)} and {*IN*(*a*), *OUT*(*b*), *UND*(*c*)} correspond to 2 sets of labelled arguments under the preferred semantics of *AF* (of which the 1st one represents stable labelling).

Application: Query Answering

Theorem Let $AF=(Ar,att)$. For any $x \in Ar$,

1. x is labelled *in* in some complete labelling of AF
   iff $\Pi^{C}_{AF} \cup \{ \leftarrow \textbf{not } in(x) \}$ has a stable model.
2. x is labelled *in* in every complete labelling of AF
   iff $\Pi^{C}_{AF} \cup \{ \leftarrow in(x) \}$ has no stable model.

The result also holds by replacing *in* with *out* or *und*.

Similar results hold for Π^{S}_{AF}, Π^{G}_{AF}, and Π^{P}_{AF}.

Application: Enforcement

- The **universal argumentation framework (UAF)** is (U, att_U) where U is the set of all arguments in the language and $att_U \subseteq U \times U$.
- $AF=(Ar, att)$ is defined as a sub-AF of the UAF s.t. $Ar \subseteq U$ and $att = att_U \cap (Ar \times Ar)$.
- B_U is defined as $B_U = \{ in(x), out(x), und(x) \mid x \in U \}$.
- Given an **enforcement set** $E \subset B_U$, if one can construct a new AF' such that (i) $AF'=(Ar', att')$ where $Ar \subseteq Ar' \subseteq U$ and $att' = att_U \cap (Ar' \times Ar')$, and (ii) AF' has a complete labelling L s.t. $L(x)=\ell$ for any $\ell(x) \in E$, then AF **satisfies** the enforcement E (under the complete semantics).

Application: Enforcement

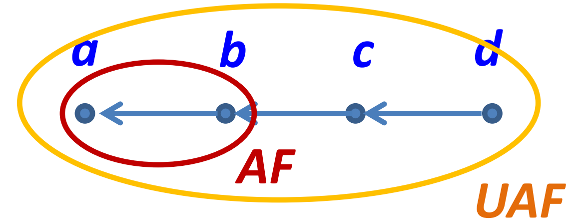
- An **AF-program under the complete semantics for the enforcement** $\varepsilon\Pi^{C}_{AF}$ is defined as

  $\varepsilon\Pi^{C}_{AF} = \Pi^{C}_{UAF} \setminus \{\, in(x)\leftarrow,\ \leftarrow out(x) \mid x \in U \setminus Ar \,\}$

Theorem Given an enforcement set $E \subset B_U$, $AF = (Ar, att)$ satisfies the enforcement E iff
$\varepsilon\Pi^{C}_{AF} \cup \{ \leftarrow \textbf{not}\ \ell(x) \mid \ell(x) \in E \text{ where } \ell \in \{in, out, und\} \}$
has a stable model.

Similar results hold for Π^{S}_{AF}, Π^{G}_{AF}, and Π^{P}_{AF}.

Example



- Let *UAF*=({*a*,*b*,*c*,*d*},{(*d*,*c*),(*c*,*b*),(*b*,*a*)}) and *AF*=({*a*,*b*},{(*b*,*a*)}). Then *AF* has the complete labelling {*out*(*a*),*in*(*b*)}. $\varepsilon\Pi^{C}_{AF}$ consists of

in(*a*)←*out*(*b*), *in*(*b*)←*out*(*c*), *in*(*c*)←*out*(*c*), ~~*in*(*d*)←~~

out(*a*)←*in*(*b*), *out*(*b*)←*in*(*c*), *out*(*c*)←*in*(*d*),

←*in*(*a*), **not** *out*(*b*), ←*in*(*b*), **not** *out*(*c*), ←*in*(*c*), **not** *out*(*d*),

←*out*(*a*), **not** *in*(*b*), ←*out*(*b*), **not** *in*(*c*), ←*out*(*c*), **not** *in*(d), ~~←*out*(*d*),~~

in(*x*) v *out*(*x*) v *und*(*x*)← where *x*∈{*a*,*b*,*c*,*d*}

←*in*(*x*), *out*(*x*), ← *in*(*x*), *und*(*x*), ←*out*(*x*), *und*(*x*) where *x*∈{*a*,*b*,*c*,*d*}

- Given *E*={*in*(*a*)}, $\varepsilon\Pi^{C}_{AF}$ ∪ { ← **not** *in*(*a*) } has the stable model {*in*(*a*), *out*(*b*), *in*(*c*), *out*(*d*)}. Then *AF* satisfies the enforcement *E*, i.e., to enforce *in*(*a*), *AF* is modified by introducing the new argument *c* and the attack relation (*c*,*b*).

Application: Agreement

- Let AF_1 and AF_2 be two sub-AFs of the *UAF*. If AF_1 (resp. AF_2) has a set S (resp. T) of labelled arguments under a complete labelling such that $S \cap T \neq \{\}$, then AF_1 and AF_2 can reach an **agreement**.
- Let $©\Pi^{C}_{AF}$ be a program in which predicates *in*, *out* and *und* in Π^{C}_{AF} are renamed by *in'*, *out'* and *und'*, respectively. Define

$$\Phi = \{\, agree(x) \leftarrow in(x), in'(x) \mid x \in U \,\}$$
$$\cup \{\, agree(x) \leftarrow out(x), out'(x) \mid x \in U \,\}$$
$$\cup \{\, agree(x) \leftarrow und(x), und'(x) \mid x \in U \,\}$$
$$\cup \{\, ok \leftarrow agree(x) \mid x \in U \,\} \cup \{\, \leftarrow \textbf{not}\ ok \,\}$$

<u>Theorem</u> AF_1 and AF_2 can reach an agreement iff $\Pi^{C}_{AF_1} \cup ©\Pi^{C}_{AF_2} \cup \Phi$ has a stable model M. In this case, AF_1 and AF_2 agree on each argument x s.t. $agree(x) \in M$.

† Similar results hold for Π^{S}_{AF}, Π^{G}_{AF}, and Π^{P}_{AF}.

†† The result is extended to agreement among more than 2 agents.

Final Remark

- Π^{C}_{AF} and Π^{S}_{AF} can be represented by semantically equivalent **normal logic programs**, while Π^{P}_{AF} cannot.
- Thus, Π^{P}_{AF} is in the class of LPs that are more expressive and computationally expensive than others.
- The proposed method is **simple** and **uniform** for different AF semantics.
- Several techniques developed in LP (e.g. equivalence issue, optimisation, update, etc) are directly applied to transformed AF-programs.
- The result of this study implicates potential use of rich LP techniques in AF (via AF-programs).