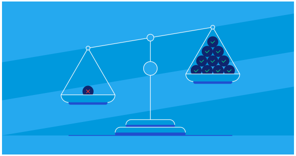
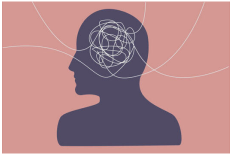
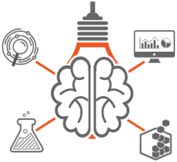
Module 5

Relations to Other Approaches for Non-Monotonic Reasoning

Relations to Formalisms for NMR

Argumentation theory has been related to a number of formalisms for NMR, among which are:

- Reasoning with (maximal) consistent subsets of premises (MCS)
- Semantics of logic programs; Answer-set programming (ASP)
- Adaptive logics
- Default logics
- Autoepistemic logics
- Input/Output logic
- . . .

Relations to Formalisms for NMR

Argumentation theory has been related to a number of formalisms for NMR, among which are:

- Reasoning with (maximal) consistent subsets of premises (MCS)
- Semantics of logic programs; Answer-set programming (ASP)
- Adaptive logics
- Default logics
- Autoepistemic logics
- Input/Output logic
- $\ldots$

Some relations to NMR have already been considered in this course:

- General patterns of NMR (e.g., the KLM postulates)
- Reasoning with MCS

Some References to Relevant Papers

Form Default Logic to ASPIC

C. Straßer and P. Pardo. *Prioritized defaults and formal argumentation.*
Proceedings of DEON'21, pp.427–446, 2021.

Form Default Logic to ABA

A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni. *An abstract argumentation-theoretic approach to default reasoning.*
Artificial Intelligence 93, pp.63–101, 1997.

Form ASPIC to Default Logic

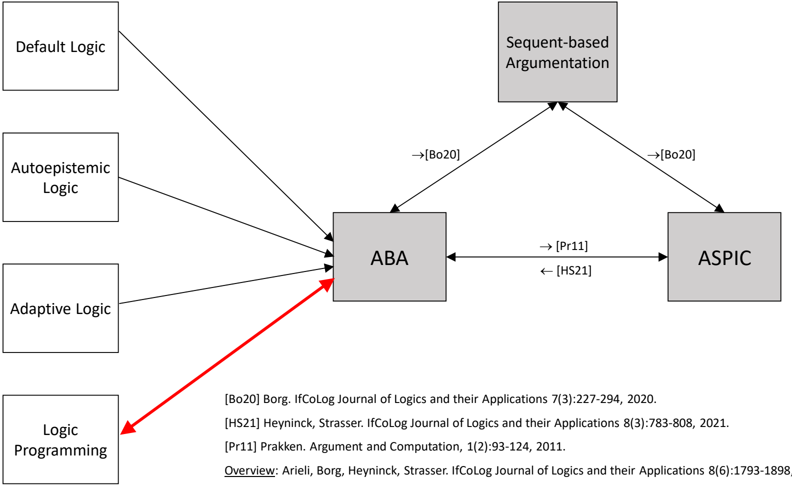
J. Heyninck and C. Straßer. *Rationality and maximal consistent sets for a fragment of ASPIC+ without undercut.*
Argument & Computation 12(1), pp.3–47, 2021.

Form Input/Output Logic to Sequent-Based Argumentation

C. Straßer, O. Arieli. *Normative reasoning by sequent-based argumentation.*
Journal of Logic and Computation 29(3), pp.387–415, 2015.

Case Study: Argumentation and Logic Programming

Recall From Module II:



[Bo20] Borg. IfCoLog Journal of Logics and their Applications 7(3):227-294, 2020.

[HS21] Heyninck, Strasser. IfCoLog Journal of Logics and their Applications 8(3):783-808, 2021.

[Pr11] Prakken. Argument and Computation, 1(2):93-124, 2011.

Overview: Arieli, Borg, Heyninck, Strasser. IfCoLog Journal of Logics and their Applications 8(6):1793-1898, 2021.

Logic Programming – A (very) Brief Overview[1]

Logic programming is a programming paradigm, based on formal logic. Major logic programming language families include Prolog, answer set programming (ASP) and Datalog.

[1] Source: Wikipedia.

Logic Programming – A (very) Brief Overview[1]

Logic programming is a programming paradigm, based on formal logic. Major logic programming language families include Prolog, answer set programming (ASP) and Datalog.

In all of these languages, rules are written in the form of *clauses*:

$$\phi \text{ :- } \psi_1, \ldots, \psi_n \quad (\textit{alternatively}, \phi \leftarrow \psi_1, \ldots, \psi_n)$$

and are read as logical implications: "ϕ if ψ_1 and $\ldots$ and ψ_n".

[1] Source: Wikipedia.

Logic Programming – A (very) Brief Overview[1]

Logic programming is a programming paradigm, based on formal logic. Major logic programming language families include Prolog, answer set programming (ASP) and Datalog.

In all of these languages, rules are written in the form of *clauses*:

$$\phi \text{ :- } \psi_1, \ldots, \psi_n \quad (\textit{alternatively}, \phi \leftarrow \psi_1, \ldots, \psi_n)$$

and are read as logical implications: "ϕ if ψ_1 and $\ldots$ and ψ_n".

- ϕ is called the head of the rule, and $\psi_1, \ldots, \psi_n$ is called the body.
- Facts are rules that have no body (ϕ).

[1] Source: Wikipedia.

Types of Logic Programs

- The simplest case:
  - Positive logic programs:

    $$p \leftarrow q_1, \ldots, q_n$$

    The heads and the bodies consist only of atomic formulas.
    The rules are called definite clauses or Horn clauses.

Types of Logic Programs

- The simplest case:
  - Positive logic programs:

    $$p \leftarrow q_1, \ldots, q_n$$

    The heads and the bodies consist only of atomic formulas.
    The rules are called definite clauses or Horn clauses.

- Adding capabilities of non-monotonic reasoning:

  Conditions in the body of a rule can also be negations (as failure) of atomic formulas:

  - Normal logic programs:

    $$p \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$$

  - Disjunctive logic programs:

    $$p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$$

  - Extended [normal/disjunctive] logic programs:
    Literals ($l \in \{p, \neg p\}$) instead of atoms.

Negation As Failure

$p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$ may be read as follows:

"If q_i holds (for every $i = 1, \ldots n$) and r_i fails to hold (for every $i = 1, \ldots m$), then at least one of p_i's must hold (for some $1 \leq i \leq k$)."

Negation As Failure

$p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$ may be read as follows:
"If q_i holds (for every $i = 1, \ldots n$) and r_i fails to hold (for every $i = 1, \ldots m$), then at least one of p_i's must hold (for some $1 \leq i \leq k$)."

Example

Consider the following normal logic program:

$$\begin{array}{l} canfly(X) \leftarrow bird(X), not\ abnormal(X) \\ abnormal(X) \leftarrow wounded(X) \\ bird(John) \\ bird(Mary) \end{array}$$

We would like to infer in this case that both John and Mary can fly.

Negation As Failure

$p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$ may be read as follows:
"If q_i holds (for every $i = 1, \ldots n$) and r_i fails to hold (for every $i = 1, \ldots m$), then at least one of p_i's must hold (for some $1 \leq i \leq k$)."

Example

Consider the following normal logic program:

$$\begin{aligned} & canfly(X) \leftarrow bird(X), not\ abnormal(X) \\ & abnormal(X) \leftarrow wounded(X) \\ & bird(John) \\ & bird(Mary) \end{aligned}$$

We would like to infer in this case that both John and Mary can fly.

But if we are informed that:

$$wounded(John)$$

then now we should conclude that only Mary can fly.

Semantics of Logic Programs

$\mathcal{P}$ – a disjunctive logic program (DLP),
Atoms($\mathcal{P}$) – the set of atomic formulas that appear in $\mathcal{P}$.

1

Semantics of Logic Programs

$\mathcal{P}$ – a disjunctive logic program (DLP),
Atoms($\mathcal{P}$) – the set of atomic formulas that appear in $\mathcal{P}$.

- $M \subseteq \text{Atoms}(\mathcal{P})$ *satisfies* a rule
  $p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$ in $\mathcal{P}$ iff either:
  $\exists 1 \leq i \leq n\ q_i \notin M$, or $\exists 1 \leq i \leq m\ r_i \in M$, or $\exists 1 \leq i \leq k\ p_i \in M$.
- M is a *model* of $\mathcal{P}$ if it satisfies every rule in $\mathcal{P}$.

1

Semantics of Logic Programs

$\mathcal{P}$ – a disjunctive logic program (DLP),
Atoms($\mathcal{P}$) – the set of atomic formulas that appear in $\mathcal{P}$.

- $M \subseteq$ Atoms($\mathcal{P}$) *satisfies* a rule
  $p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m$ in $\mathcal{P}$ iff either:
  $\exists 1 \leq i \leq n \; q_i \notin M$, or $\exists 1 \leq i \leq m \; r_i \in M$, or $\exists 1 \leq i \leq k \; p_i \in M$.
- M is a *model* of $\mathcal{P}$ if it satisfies every rule in $\mathcal{P}$.

- The *Gelfond-Lifschitz reduct* [1] of $\mathcal{P}$ with respect to M is the disjunctive (positive) logic program $\mathcal{P}^M$, where
  $p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n \in \mathcal{P}^M$ iff there is a rule
  $p_1 \vee \ldots \vee p_k \leftarrow q_1, \ldots, q_n, \text{not } r_1, \ldots, \text{not } r_m \in \mathcal{P}$
  and $r_i \notin M$ for every $1 \leq i \leq m$.
- M is a *stable model* of $\mathcal{P}$ iff it is a $\subseteq$-minimal model of $\mathcal{P}^M$.

[1] M. Gelfond and V. Lifschitz. *The Stable Model Semantics for Logic Programming*, Proc. ICLP'88, pp.1070–1080, 1988.

Example, Continued

$$P_1 = \left\{ \begin{array}{l} \mathit{canfly}(\mathit{John}) \leftarrow \mathit{bird}(\mathit{John}), \mathit{not}\ \mathit{abnormal}(\mathit{John}) \\ \mathit{abnormal}(\mathit{John}) \leftarrow \mathit{wounded}(\mathit{John}) \\ \mathit{bird}(\mathit{John}) \end{array} \right\}$$

$M_1 = \{\mathit{bird}(\mathit{John}), \mathit{canfly}(\mathit{John})\}$ is the stable model of $\mathcal{P}_1$, since it is a $\subseteq$-minimal model of the GL-reduct

$$P_1^{M_1} = \left\{ \begin{array}{l} \mathit{canfly}(\mathit{John}) \leftarrow \mathit{bird}(\mathit{John}) \\ \mathit{abnormal}(\mathit{John}) \leftarrow \mathit{wounded}(\mathit{John}) \\ \mathit{bird}(\mathit{John}) \end{array} \right\}$$

Example, Continued

$$P_1 = \left\{ \begin{array}{l} \mathit{canfly}(\mathit{John}) \leftarrow \mathit{bird}(\mathit{John}), \mathit{not\ abnormal}(\mathit{John}) \\ \mathit{abnormal}(\mathit{John}) \leftarrow \mathit{wounded}(\mathit{John}) \\ \mathit{bird}(\mathit{John}) \end{array} \right\}$$

$M_1 = \{\mathit{bird}(\mathit{John}), \mathit{canfly}(\mathit{John})\}$ is the stable model of $\mathcal{P}_1$, since it is a $\subseteq$-minimal model of the GL-reduct

$$P_1^{M_1} = \left\{ \begin{array}{l} \mathit{canfly}(\mathit{John}) \leftarrow \mathit{bird}(\mathit{John}) \\ \mathit{abnormal}(\mathit{John}) \leftarrow \mathit{wounded}(\mathit{John}) \\ \mathit{bird}(\mathit{John}) \end{array} \right\}$$

$$P_2 = P_1 \cup \mathit{wounded}(\mathit{John})$$

$M_2 = \{\mathit{bird}(\mathit{John}), \mathit{wounded}(\mathit{John}), \mathit{abnormal}(\mathit{John})\}$ is the stable model of $\mathcal{P}_2$, since it is a $\subseteq$-minimal model of the GL-reduct

$$P_2^{M_2} = \left\{ \begin{array}{l} \mathit{abnormal}(\mathit{John}) \leftarrow \mathit{wounded}(\mathit{John}) \\ \mathit{bird}(\mathit{John}) \\ \mathit{wounded}(\mathit{John}) \end{array} \right\}$$

Representation of DLPs by ABFs

Goal: Given a disjunctive logic program (DLP) $\mathcal{P}$, we construct an assumption-based argumentation framework (ABF) $\mathcal{ABF}$, such that there is a one-to-one correspondence between the stable models of $\mathcal{P}$ and the stable extensions of $\mathcal{AF}$.

Representation of DLPs by ABFs

Goal: Given a disjunctive logic program (DLP) $\mathcal{P}$, we construct an assumption-based argumentation framework (ABF) $\mathcal{ABF}$, such that there is a one-to-one correspondence between the stable models of $\mathcal{P}$ and the stable extensions of $\mathcal{AF}$.

Recall:

An *assumption-based framework* is a tuple $\mathcal{ABF} = \langle \mathfrak{L}, \Gamma, \Delta, \sim \rangle$, s.t.:

- $\mathfrak{L} = \langle \mathcal{L}, \vdash \rangle$ is a (propositional) logic,
- Γ is a set of $\mathcal{L}$-formulas, called the *strict rules*,
- Δ is a set of $\mathcal{L}$-formulas, called the *defeasible assumptions*,
- $\sim : \Delta \to 2^{\mathcal{L}}$ is a *contrariness operator*.

$\Theta \subseteq \Delta$ *attacks* ψ if there are $\Theta' \subseteq \Theta$ and $\phi \in {\sim}\psi$ such that $\Gamma, \Theta' \vdash \phi$.

Θ_1 attacks Θ_2 if Θ_1 attacks some $\psi \in \Theta_2$.

The Underlying Logic

All the ABA frameworks that are induced from disjunctive logic programs are based on the same core logic $\mathfrak{L}_{\mathsf{DLP}} = \langle \mathcal{L}, \vdash \rangle$, where:

- $\mathcal{L} = \{\leftarrow, \vee, \mathit{not}\}$ consists of disjunctions of atoms ($p_1 \vee \ldots \vee p_n$ for $n \geq 1$), negated atoms (*not p*), or DLP rules.
- $\mathcal{S} \vdash \psi$ iff $\psi \in \mathcal{S}$ or ψ is derived from $\mathcal{S}$ using Modus Ponens (MP), Resolution (Res) and Reasoning by Cases (RBC).

The Underlying Logic

All the ABA frameworks that are induced from disjunctive logic programs are based on the same core logic $\mathfrak{L}_{\mathrm{DLP}} = \langle \mathcal{L}, \vdash \rangle$, where:

- $\mathcal{L} = \{\leftarrow, \vee, \mathit{not}\}$ consists of disjunctions of atoms ($p_1 \vee \ldots \vee p_n$ for $n \geq 1$), negated atoms ($\mathit{not}\ p$), or DLP rules.
- $\mathcal{S} \vdash \psi$ iff $\psi \in \mathcal{S}$ or ψ is derived from $\mathcal{S}$ using Modus Ponens (MP), Resolution (Res) and Reasoning by Cases (RBC).

[MP] $$\frac{p1 \vee \ldots \vee p_n \leftarrow l_1, \ldots, l_n \quad l_1 \quad l_2 \quad \cdots \quad l_n}{p1 \vee \ldots \vee p_n} \quad (\ l_i \in \{p_i, \mathit{not}\ p_i\}\ )$$

[Res] $$\frac{p'_1 \vee \ldots \vee p'_m \vee q_1 \vee \ldots \vee q_n \vee p''_1 \vee \ldots \vee p''_k \quad \mathit{not}\ q_1 \quad \cdots \quad \mathit{not}\ q_n}{p'_1 \vee \ldots \vee p'_m \vee \ldots \vee p''_1 \vee \ldots \vee p''_k}$$

[RBC] $$\frac{\begin{array}{c} q_1 \\ \vdots \\ p_1 \vee \ldots \vee p_n \end{array} \quad \begin{array}{c} q_2 \\ \vdots \\ p_1 \vee \ldots \vee p_n \end{array} \quad \cdots \quad \begin{array}{c} q_m \\ \vdots \\ p_1 \vee \ldots \vee p_n \end{array} \quad q_1 \vee \ldots \vee q_m}{p_1 \vee \ldots \vee p_n}$$

Representation of DLPs by ABFs

Definition

The assumption-based argumentation framework that is *induced* by a disjunctive logic program $\mathcal{P}$ is $\mathcal{ABF}(\mathcal{P}) = \langle \mathfrak{L}_{\mathsf{DLP}}, \Gamma, \Delta, \sim \rangle$, where: $\Gamma = \mathcal{P}$, $\Delta = \{not\ p \mid p \in \mathsf{Atom}(\mathcal{P})\}$, and $\forall p \in \mathsf{Atom}(\mathcal{P})$ $\sim(not\ p) = \{p\}$.

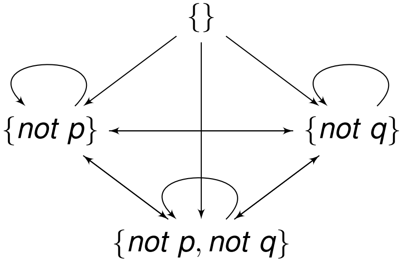
Representation of DLPs by ABFs

Definition

The assumption-based argumentation framework that is *induced* by a disjunctive logic program $\mathcal{P}$ is $\mathcal{ABF}(\mathcal{P}) = \langle \mathfrak{L}_{\mathsf{DLP}}, \Gamma, \Delta, \sim \rangle$, where: $\Gamma = \mathcal{P}$, $\Delta = \{not\ p \mid p \in \mathsf{Atom}(\mathcal{P})\}$, and $\forall p \in \mathsf{Atom}(\mathcal{P})\ \sim(not\ p) = \{p\}$.

Example: $\mathcal{P} = \{p \vee q \leftarrow,\ p \leftarrow q,\ q \leftarrow p\}$

Attack diagram for the induced ABF:



For instance, *not q* attacks itself since $\mathcal{P}$, *not q* $\vdash \sim(not\ q)$, i.e., $\mathcal{P}$, *not q* $\vdash q$.
Indeed, $p \vee q \leftarrow\ \in \mathcal{P}$. By [MP], $p \vee q$. [Res] with *not q* gives p, and [*MP*] with $q \leftarrow p$ gives q.

Representation of DLPs by ABFs

Let $\mathcal{P}$ be a disjunctive logic program and $\Theta \subseteq \text{Atoms}(\mathcal{P})$.

We denote:

- $\textit{not}\ \Theta = \{\textit{not}\ \theta \mid \theta \in \Theta\}$.
- $\lfloor \textit{not}\ \Theta \rfloor = \Theta$.
- If $\Delta \subseteq \textit{not}\ \text{Atoms}(\mathcal{P})$ then $\underline{\Delta} = \text{Atoms}(\mathcal{P}) \setminus \lfloor \Delta \rfloor$.
- If $\Delta \subseteq \text{Atoms}(\mathcal{P})$ then $\overline{\Delta} = \textit{not}(\text{Atoms}(\mathcal{P}) \setminus \Delta)$.

Thus, $\lfloor \Theta \rfloor$ eliminates the leading *not* from formulas in Θ.
$\underline{\Delta}$ (respectively, $\overline{\Delta}$) takes the complementary set of Δ and removes (respectively, adds) the negation-as-failure operator from (respectively, to) the prefix of its formulas.

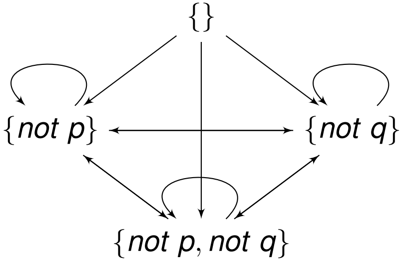
Theorem

- *If Δ is a stable extension of $\mathcal{ABF}(\mathcal{P})$, $\underline{\Delta}$ is a stable model of $\mathcal{P}$,*
- *If Δ is a stable model of $\mathcal{P}$, $\overline{\Delta}$ is a stable extension of $\mathcal{ABF}(\mathcal{P})$.*

Example, Revisited

$$\mathcal{P} = \{p \vee q \leftarrow,\ p \leftarrow q,\ q \leftarrow p\}$$

Attack diagram for the induced ABF:



Atoms($\mathcal{P}$) $= \{p, q\}$. The stable model of $\mathcal{P}$ is {p,q}.
The stable extension of $\mathcal{ABF}(\mathcal{P})$ is $\overline{\{p, q\}} = \{\}$.

Further Notes

Theorem

- *If Δ is a stable extension of $\mathcal{ABF}(\mathcal{P})$, $\underline{\Delta}$ is a stable model of $\mathcal{P}$,*
- *If Δ is a stable model of $\mathcal{P}$, $\overline{\Delta}$ is a stable extension of $\mathcal{ABF}(\mathcal{P})$.*

- The theorem was shown for disjunctive normal programs in [1].
- Caminada and Schulz [2] have shown that for normal logic programs the base logic of the ABF may contain only [MP].
- Wakaki [3] has proven similar results for extended normal logic programs.

[1] J. Heyninck , O. Arieli. *An argumentative characterization of disjunctive logic programming*. Proceedings of EPIA'21, LNCS 11805, pp.52–538, Springer, 2019.

[2] M. Caminada, C. Schulz. *On the equivalence between assumption-based argumentation and logic programming*. Artif. Intell. Research 60:779–825, 2017.

[3] T. Wakaki. *Consistency in assumption-based argumentation*. Proceedings of COMMA'20, pp.371–382, IOS Press, 2020.