TECHNISCHE UNIVERSITÄT DRESDEN

Jesse Heyninck, Hannes Strass

Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

An Introduction to Approximation Fixpoint Theory

Lecture 1, 3rd Nov 2024 // Tutorial, KR 2024, Hanoi

Motivation: Objective

Goal: Define semantics for (rule-based) KR formalisms in the presence of:

Recursion

- transitive closure
- indirect effects of actions

Motivation: Objective

Goal: Define semantics for (rule-based) KR formalisms in the presence of:

Recursion

- transitive closure
- indirect effects of actions

Negation

- shorter and more intuitive descriptions
- defaults and assumptions (e.g. closed world, non-effects of actions)

Motivation: Objective

Goal: Define semantics for (rule-based) KR formalisms in the presence of:

Recursion

- transitive closure
- indirect effects of actions

Negation

- shorter and more intuitive descriptions
- defaults and assumptions (e.g. closed world, non-effects of actions)

Recursion Through Negation

- mutually exclusive alternatives
- non-deterministic effects of actions

Motivation: Basic Idea

Approximation Fixpoint Theory

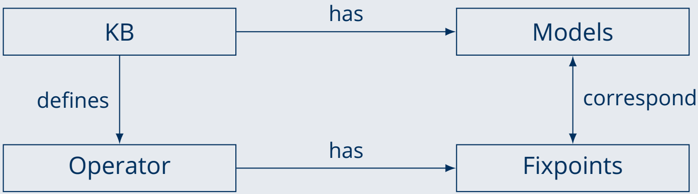
- Framework for studying semantics of (non-monotonic) KR formalisms
- Due to Denecker, Marek, and Truszczyński [2000, 2003, 2004]
- Based on lattice theory and fixpoint theory:

$$\boxed{\text{KB}} \xrightarrow{\text{has}} \boxed{\text{Models}}$$

Motivation: Basic Idea

Approximation Fixpoint Theory

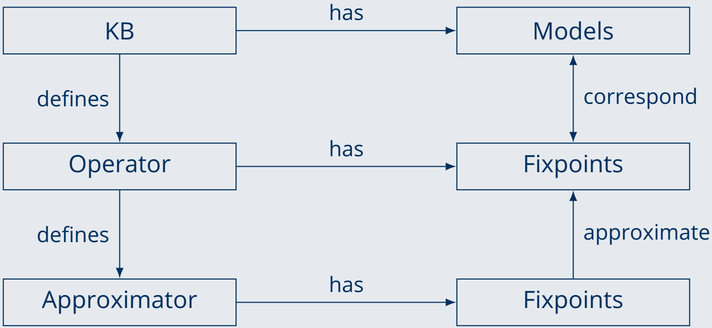
- Framework for studying semantics of (non-monotonic) KR formalisms
- Due to Denecker, Marek, and Truszczyński [2000, 2003, 2004]
- Based on lattice theory and fixpoint theory:



Motivation: Basic Idea

Approximation Fixpoint Theory

- Framework for studying semantics of (non-monotonic) KR formalisms
- Due to Denecker, Marek, and Truszczyński [2000, 2003, 2004]
- Based on lattice theory and fixpoint theory:



Motivation: History and Context

Approximation Fixpoint Theory

... emerged from similarities in the semantics of

- Default Logic
- Autoepistemic Logic
- Logic Programs, in particular Stable Models

... and has since been applied to define/reconstruct semantics of ...

- Abstract Argumentation Frameworks
- Abstract Dialectical Frameworks
- Active Integrity Constraints
- Recursive SHACL

... and develop language-independent theory about ...

- Complexity
- Stratification and independence
- Groundedness

Learning Outcomes

- Understand the role of operators and fixpoints in KRR.
- Understand the concept of an approximation in KRR, and its connection with three- and four-valued logics.
- Understand the idea of an approximation of a potentially non-monotonic operator, and how this allows to approximate fixpoints of the original operator.
- Understand how the stable approximator is constructed, and how this allows to define the well-founded fixpoint.
- Realize the benefit of the algebraic approach to KRR underlying AFT, and how this allows to give a language-independent account of important concepts occurring in different sub-fields of KRR.

Agenda

Partially Ordered Sets

Definition

A **partially ordered set** is a pair $(L, \leqslant)$ with

- L a set, and (carrier set)
- $\leqslant \subseteq L \times L$ a partial order. (reflexive, antisymmetric, transitive)

Partially Ordered Sets

Definition

A **partially ordered set** is a pair $(L, \leqslant)$ with

- L a set, and (carrier set)
- $\leqslant \subseteq L \times L$ a partial order. (reflexive, antisymmetric, transitive)

A partially ordered set $(L, \leqslant)$ has a

- **bottom element** $\bot \in L$ iff $\bot \leqslant x$ for all $x \in L$,
- **top element** $\top \in L$ iff $x \leqslant \top$ for all $x \in L$.

Partially Ordered Sets

Definition

A **partially ordered set** is a pair $(L, \leqslant)$ with

- L a set, and (carrier set)
- $\leqslant \subseteq L \times L$ a partial order. (reflexive, antisymmetric, transitive)

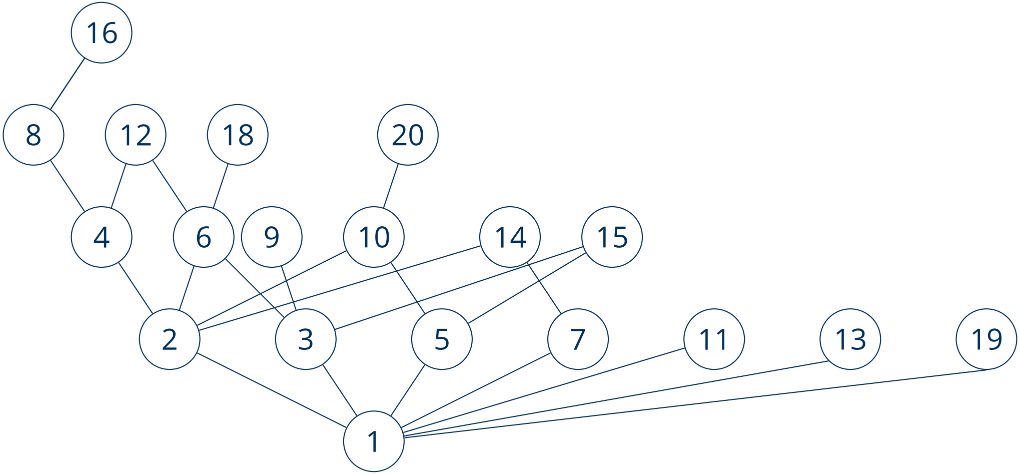
A partially ordered set $(L, \leqslant)$ has a

- **bottom element** $\bot \in L$ iff $\bot \leqslant x$ for all $x \in L$,
- **top element** $\top \in L$ iff $x \leqslant \top$ for all $x \in L$.

Examples

- $(\mathbb{N}, \leq)$: natural numbers with "usual" ordering, $\bot = 0$, no $\top$
- $(2^S, \subseteq)$: any powerset with subset relation, $\bot = \emptyset$, $\top = S$
- $(\mathbb{N}, |)$: natural numbers with divisibility relation, $\bot = 1$, $\top = 0$

Graphic Intuition for $(\{1, \ldots, 20\}, |)$



Minimal, Maximal, Least, Greatest

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in S$. We say that:

- x is a **minimal element** of S iff for each $y \in S$, $y \leqslant x$ implies $y = x$, dually,
- x is a **maximal element** of S iff for each $y \in S$, $x \leqslant y$ implies $y = x$;

Minimal, Maximal, Least, Greatest

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in S$. We say that:

- x is a **minimal element** of S iff for each $y \in S$, $y \leqslant x$ implies $y = x$, dually,
- x is a **maximal element** of S iff for each $y \in S$, $x \leqslant y$ implies $y = x$;
- x is the **least element** of S iff for each $y \in S$, we have $x \leqslant y$, dually,
- x is the **greatest element** of S iff for each $y \in S$, we have $y \leqslant x$.

Minimal, Maximal, Least, Greatest

Definition

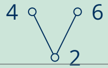
Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in S$. We say that:

- x is a **minimal element** of S iff for each $y \in S$, $y \leqslant x$ implies $y = x$, dually,
- x is a **maximal element** of S iff for each $y \in S$, $x \leqslant y$ implies $y = x$;
- x is the **least element** of S iff for each $y \in S$, we have $x \leqslant y$, dually,
- x is the **greatest element** of S iff for each $y \in S$, we have $y \leqslant x$.

Example

In $(\mathbb{N}, |)$ (natural numbers with divisibility $a \mid b \iff (\exists k \in \mathbb{N}) a \cdot k = b$), ...

- the set $\{2, 3, 6\}$ has minimal elements 2 and 3, greatest element 6,
- the set $\{2, 4, 6\}$ has least element 2, and maximal elements 4 and 6.



Least Upper and Greatest Lower Bounds

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in L$.

- x is an **upper bound** of S iff for each $s \in S$, we have $s \leqslant x$, dually,
- x is a **lower bound** of S iff for each $s \in S$, we have $x \leqslant s$.

Least Upper and Greatest Lower Bounds

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in L$.

- x is an **upper bound** of S iff for each $s \in S$, we have $s \leqslant x$, dually,
- x is a **lower bound** of S iff for each $s \in S$, we have $x \leqslant s$.

The set of all upper bounds of S is denoted by S^u, its lower bounds by S^ℓ.

Least Upper and Greatest Lower Bounds

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in L$.

- x is an **upper bound** of S iff for each $s \in S$, we have $s \leqslant x$, dually,
- x is a **lower bound** of S iff for each $s \in S$, we have $x \leqslant s$.

The set of all upper bounds of S is denoted by S^u, its lower bounds by S^ℓ.

- If S^u has a least element $z \in S$, z is the **least upper bound** of S, dually,
- if S^ℓ has a greatest element $z \in S$, z is the **greatest lower bound** of S.

Least Upper and Greatest Lower Bounds

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in L$.

- x is an **upper bound** of S iff for each $s \in S$, we have $s \leqslant x$, dually,
- x is a **lower bound** of S iff for each $s \in S$, we have $x \leqslant s$.

The set of all upper bounds of S is denoted by S^u, its lower bounds by S^ℓ.

- If S^u has a least element $z \in S$, z is the **least upper bound** of S, dually,
- if S^ℓ has a greatest element $z \in S$, z is the **greatest lower bound** of S.

We denote the **glb** of $\{x, y\}$ by $x \wedge y$, and the **lub** of $\{x, y\}$ by $x \vee y$.

Least Upper and Greatest Lower Bounds

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in L$.

- x is an **upper bound** of S iff for each $s \in S$, we have $s \leqslant x$, dually,
- x is a **lower bound** of S iff for each $s \in S$, we have $x \leqslant s$.

The set of all upper bounds of S is denoted by S^u, its lower bounds by S^ℓ.

- If S^u has a least element $z \in S$, z is the **least upper bound** of S, dually,
- if S^ℓ has a greatest element $z \in S$, z is the **greatest lower bound** of S.

We denote the **glb** of $\{x, y\}$ by $x \wedge y$, and the **lub** of $\{x, y\}$ by $x \vee y$.
We denote the glb of S by $\bigwedge S$, and the lub of S by $\bigvee S$.

Least Upper and Greatest Lower Bounds

Definition

Let $(L, \leqslant)$ be a partially ordered set with $S \subseteq L$ and $x \in L$.

- x is an **upper bound** of S iff for each $s \in S$, we have $s \leqslant x$, dually,
- x is a **lower bound** of S iff for each $s \in S$, we have $x \leqslant s$.

The set of all upper bounds of S is denoted by S^u, its lower bounds by S^ℓ.

- If S^u has a least element $z \in S$, z is the **least upper bound** of S, dually,
- if S^ℓ has a greatest element $z \in S$, z is the **greatest lower bound** of S.

We denote the **glb** of $\{x, y\}$ by $x \wedge y$, and the **lub** of $\{x, y\}$ by $x \vee y$.
We denote the glb of S by $\bigwedge S$, and the lub of S by $\bigvee S$.

Examples

- In $(2^S, \subseteq)$, $\wedge = \cap$ and $\vee = \cup$;
- in $(\mathbb{N}, |)$, $\wedge = \gcd$ and $\vee = \mathrm{lcm}$, e.g. $4 \vee 6 = 12$ and $23 \wedge 42 = 1$.

(Complete) Lattices

Definition

Let $(L, \leqslant)$ be a partially ordered set.

1. $(L, \leqslant)$ is a **lattice** if and only if for all $x, y \in L$, both $x \wedge y$ and $x \vee y$ exist;

(Complete) Lattices

Definition

Let $(L, \leqslant)$ be a partially ordered set.

1. $(L, \leqslant)$ is a **lattice** if and only if for all $x, y \in L$, both $x \wedge y$ and $x \vee y$ exist;
2. $(L, \leqslant)$ is a **complete lattice** iff for all $S \subseteq L$, both $\bigwedge S$ and $\bigvee S$ exist.

(Complete) Lattices

Definition

Let $(L, \leqslant)$ be a partially ordered set.

1. $(L, \leqslant)$ is a **lattice** if and only if for all $x, y \in L$, both $x \wedge y$ and $x \vee y$ exist;
2. $(L, \leqslant)$ is a **complete lattice** iff for all $S \subseteq L$, both $\bigwedge S$ and $\bigvee S$ exist.

In particular, a complete lattice has $\bigvee \emptyset = \bigwedge L = \bot$ and $\bigwedge \emptyset = \bigvee L = \top$.

(Complete) Lattices

Definition

Let $(L, \leqslant)$ be a partially ordered set.

1. $(L, \leqslant)$ is a **lattice** if and only if for all $x, y \in L$, both $x \wedge y$ and $x \vee y$ exist;
2. $(L, \leqslant)$ is a **complete lattice** iff for all $S \subseteq L$, both $\bigwedge S$ and $\bigvee S$ exist.

In particular, a complete lattice has $\bigvee \emptyset = \bigwedge L = \bot$ and $\bigwedge \emptyset = \bigvee L = \top$.

Examples

- $(2^S, \subseteq)$ is a complete lattice for every set S.
- $(\mathbb{N}, \mid)$ is a complete lattice.
- $(\{M \subseteq \mathbb{N} \mid M \text{ is finite}\}, \subseteq)$ is a lattice.
- Every lattice $(L, \leqslant)$ with L finite is a complete lattice. (induction on $|S|$)

Further reading: B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Second Edition. Cambridge University Press, 2002

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O \colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O \colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

Example

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O \colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O\colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

Example

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.

- $O(\{2,3\}) = \{1,2,3,6\}$ and $O(\{2,3,5\}) = \{1,2,3,5,6,10,15,30\}$.

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O\colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

Example

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.

- $O(\{2,3\}) = \{1,2,3,6\}$ and $O(\{2,3,5\}) = \{1,2,3,5,6,10,15,30\}$.
- O is $\subseteq$-monotone:

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O \colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

Example

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O \colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.

- $O(\{2,3\}) = \{1,2,3,6\}$ and $O(\{2,3,5\}) = \{1,2,3,5,6,10,15,30\}$.
- O is $\subseteq$-monotone:
  - Let $M_1 \subseteq M_2 \subseteq \mathbb{N}$ and consider $k \in O(M_1)$.

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O \colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

Example

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O \colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.

- $O(\{2,3\}) = \{1,2,3,6\}$ and $O(\{2,3,5\}) = \{1,2,3,5,6,10,15,30\}$.
- O is $\subseteq$-monotone:
  - Let $M_1 \subseteq M_2 \subseteq \mathbb{N}$ and consider $k \in O(M_1)$.
  - Then there is a $K \subseteq M_1$ with $k = \prod K$.

Operators and Their Properties

Definition

Let $(L, \leqslant)$ be a partially ordered set.
An operator $O\colon L \to L$ is $\leqslant$-**monotone** if and only if for all $x, y \in L$,

$$x \leqslant y \quad \text{implies} \quad O(x) \leqslant O(y)$$

Intuition: Operator application preserves ordering.

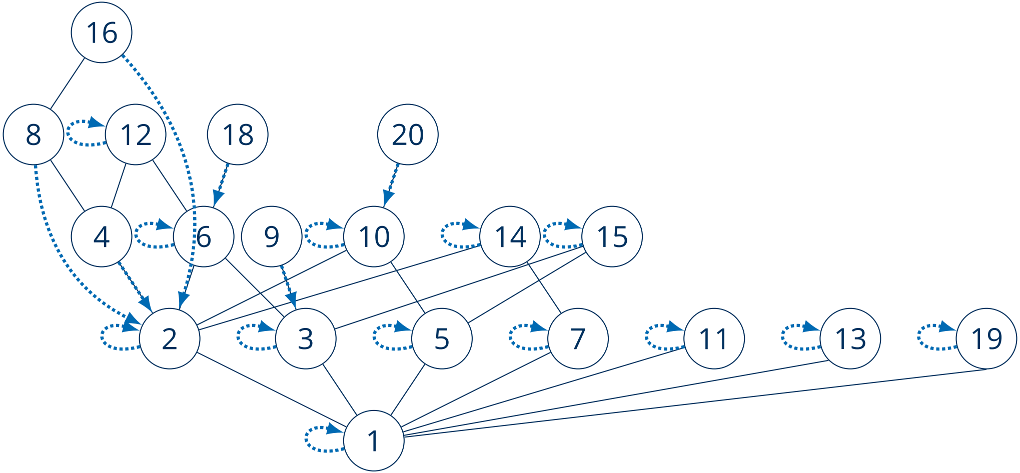
Example

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.

- $O(\{2,3\}) = \{1,2,3,6\}$ and $O(\{2,3,5\}) = \{1,2,3,5,6,10,15,30\}$.
- O is $\subseteq$-monotone:
  - Let $M_1 \subseteq M_2 \subseteq \mathbb{N}$ and consider $k \in O(M_1)$.
  - Then there is a $K \subseteq M_1$ with $k = \prod K$.
  - By $K \subseteq M_1 \subseteq M_2$, we get $k \in O(M_2)$.

Operators and Their Properties: Example

Consider $(\mathbb{N}, |)$ with operator $O\colon \mathbb{N} \to \mathbb{N}$, $\quad n \mapsto \prod\{m \mid m \text{ is a prime factor of } n\}$:



Is this operator monotone?

Fixpoints of Operators

Definition

Let $(L, \leqslant)$ be a partially ordered set and $O\colon L \to L$ be an operator.

- $x \in L$ is a **fixpoint** of O iff $O(x) = x$;
- $x \in L$ is a **prefixpoint** of O iff $O(x) \leqslant x$;
- $x \in L$ is a **postfixpoint** of O iff $x \leqslant O(x)$.

Fixpoints of Operators

Definition

Let $(L, \leqslant)$ be a partially ordered set and $O \colon L \to L$ be an operator.

- $x \in L$ is a **fixpoint** of O iff $O(x) = x$;
- $x \in L$ is a **prefixpoint** of O iff $O(x) \leqslant x$;
- $x \in L$ is a **postfixpoint** of O iff $x \leqslant O(x)$.

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Fixpoints of Operators

Definition

Let $(L, \leqslant)$ be a partially ordered set and $O \colon L \to L$ be an operator.

- $x \in L$ is a **fixpoint** of O iff $O(x) = x$;
- $x \in L$ is a **prefixpoint** of O iff $O(x) \leqslant x$;
- $x \in L$ is a **postfixpoint** of O iff $x \leqslant O(x)$.

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Order-preserving operators on complete lattices have a fixpoint.

Fixpoints of Operators

Definition

Let $(L, \leqslant)$ be a partially ordered set and $O\colon L \to L$ be an operator.

- $x \in L$ is a **fixpoint** of O iff $O(x) = x$;
- $x \in L$ is a **prefixpoint** of O iff $O(x) \leqslant x$;
- $x \in L$ is a **postfixpoint** of O iff $x \leqslant O(x)$.

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Order-preserving operators on complete lattices have a fixpoint.

Example (Continued.)

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.
O has least and greatest fixpoints: $O(\{1\}) = \{1\}$ and $O(\mathbb{N}) = \mathbb{N}$.

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $a = \bigwedge A$. $(A \neq \emptyset$ as $\top \in A.)$

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $\alpha = \bigwedge A$. ($A \neq \emptyset$ as $\top \in A$.)

- For every $x \in A$, we have $\alpha \leqslant x$ and by monotonicity $O(\alpha) \leqslant O(x) \leqslant x$.

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $a = \bigwedge A$. $(A \neq \emptyset \text{ as } \top \in A.)$

- For every $x \in A$, we have $a \leqslant x$ and by monotonicity $O(a) \leqslant O(x) \leqslant x$.
- Thus $O(a)$ is a lower bound of A.

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $a = \bigwedge A$. $(A \neq \emptyset$ as $\top \in A.)$

- For every $x \in A$, we have $a \leqslant x$ and by monotonicity $O(a) \leqslant O(x) \leqslant x$.
- Thus $O(a)$ is a lower bound of A.
- Since a is the greatest lower bound of A, we get $O(a) \leqslant a$, that is, $a \in A$.

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $a = \bigwedge A$. $(A \neq \emptyset$ as $\top \in A.)$

- For every $x \in A$, we have $a \leqslant x$ and by monotonicity $O(a) \leqslant O(x) \leqslant x$.
- Thus $O(a)$ is a lower bound of A.
- Since a is the greatest lower bound of A, we get $O(a) \leqslant a$, that is, $a \in A$.
- Furthermore, monotonicity yields $O(O(a)) \leqslant O(a)$, whence $O(a) \in A$.

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $a = \bigwedge A$. $(A \neq \emptyset$ as $\top \in A.)$

- For every $x \in A$, we have $a \leqslant x$ and by monotonicity $O(a) \leqslant O(x) \leqslant x$.
- Thus $O(a)$ is a lower bound of A.
- Since a is the greatest lower bound of A, we get $O(a) \leqslant a$, that is, $a \in A$.
- Furthermore, monotonicity yields $O(O(a)) \leqslant O(a)$, whence $O(a) \in A$.
- Since a is a lower bound of A, we get $a \leqslant O(a)$, thus $O(a) = a$.

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $\alpha = \bigwedge A$. $(A \neq \emptyset$ as $\top \in A.)$

- For every $x \in A$, we have $\alpha \leqslant x$ and by monotonicity $O(\alpha) \leqslant O(x) \leqslant x$.
- Thus $O(\alpha)$ is a lower bound of A.
- Since α is the greatest lower bound of A, we get $O(\alpha) \leqslant \alpha$, that is, $\alpha \in A$.
- Furthermore, monotonicity yields $O(O(\alpha)) \leqslant O(\alpha)$, whence $O(\alpha) \in A$.
- Since α is a lower bound of A, we get $\alpha \leqslant O(\alpha)$, thus $O(\alpha) = \alpha$.
- Greatest fixpoint β is obtained dually: $B = \{x \in L \mid x \leqslant O(x)\}$, $\beta = \bigvee B$. $\square$

Fixpoints of Operators (2)

Theorem (Knaster/Tarski)

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be a monotone operator.
Then the set F of fixpoints of O has a least element and a greatest element.

Proof.

Define $A = \{x \in L \mid O(x) \leqslant x\}$ and $\alpha = \bigwedge A$. $(A \neq \emptyset$ as $\top \in A.)$

- For every $x \in A$, we have $\alpha \leqslant x$ and by monotonicity $O(\alpha) \leqslant O(x) \leqslant x$.
- Thus $O(\alpha)$ is a lower bound of A.
- Since α is the greatest lower bound of A, we get $O(\alpha) \leqslant \alpha$, that is, $\alpha \in A$.
- Furthermore, monotonicity yields $O(O(\alpha)) \leqslant O(\alpha)$, whence $O(\alpha) \in A$.
- Since α is a lower bound of A, we get $\alpha \leqslant O(\alpha)$, thus $O(\alpha) = \alpha$.
- Greatest fixpoint β is obtained dually: $B = \{x \in L \mid x \leqslant O(x)\}$, $\beta = \bigvee B$. $\square$

$(F, \leqslant)$ is a complete lattice: for $G \subseteq F$, take $([\bigvee G, \bigvee L], \leqslant)$ and $([\bigwedge L, \bigwedge G], \leqslant)$.

Fixpoints of Operators (3)

Nice to know there is one, but how do we get there?

Theorem

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be a $\leqslant$-monotone operator. For ordinals α, β, define

$$\begin{aligned} O^0(\bot) &= \bot \\ O^{\alpha+1}(\bot) &= O(O^\alpha(\bot)) && \text{for successor ordinals} \\ O^\beta(\bot) &= \bigvee \left\{ O^\alpha(\bot) \;\middle|\; \alpha < \beta \right\} && \text{for limit ordinals} \end{aligned}$$

Then for some ordinal α, the element $O^\alpha(\bot)$ is a fixpoint of O.

Example (Continued.)

Consider $(2^{\mathbb{N}}, \subseteq)$ with operator $O\colon 2^{\mathbb{N}} \to 2^{\mathbb{N}}, \quad M \mapsto \{\prod K \mid K \subseteq M,\ K \text{ finite}\}$.
We obtain the chain $O^0(\emptyset) = \emptyset \rightsquigarrow O^1(\emptyset) = \{1\} \rightsquigarrow O^2(\emptyset) = O(\{1\}) = \{1\}$.

Answer Set Programming: Motivation

- Specific, powerful family of languages for knowledge representation (problems up to second level of polynomial hierarchy).
- Efficient, user-friendly solvers (clingo[1], DLV) and tools.[2]
- Hallmark of the declarative programming approach: describe a problem (without having to describe how to find solutions).

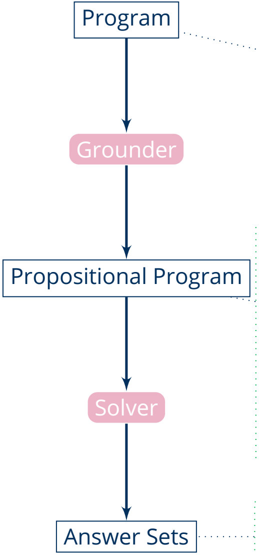
```
node(1..6).
edge(1,2;1,3;1,4;2,4;2,5;2,6;3,1;3,4;3,5;4,1).
col(r). col(g). col(b).

{ color(X,C) : col(C) } = 1 :- node(X).
:- edge(X,Y), color(X,C), color(Y,C).
```

[1] https://potassco.org/clingo/run/

[2] https://potassco.org/related/ and their weekly seminar.

The ASP Workflow



penguin(tweety).
bird(zazu).
bird(X)← penguin(X).
flies(X)← bird(X), ¬ ab(X).
ab(X)← penguin(X).

penguin(tweety).
bird(zazu).
bird(tweety)← penguin(tweety).
flies(tweety)← bird(tweety), ¬ ab(tweety).
ab(tweety)← penguin(tweety).
bird(zazu)← penguin(zazu).
flies(zazu)← bird(zazu), ¬ ab(zazu).
ab(zazu)← penguin(zazu).

$$\left\{\begin{array}{ccc} \textit{penguin}(\textit{tweety}) & \textit{bird}(\textit{tweety}) & \textit{ab}(\textit{tweety}) \\ \textit{bird}(\textit{zazu}) & \textit{bird}(\textit{zazu}) & \textit{flies}(\textit{zazu}) \end{array}\right\}$$

The ASP Workflow: Today's Focus

Propositional Program

$\downarrow$

Solver

$\downarrow$

Answer Sets

penguin(tweety).
bird(zazu).
bird(tweety)$\leftarrow$ penguin(tweety).
flies(tweety)$\leftarrow$ bird(tweety), $\neg$ ab(tweety).
ab(tweety)$\leftarrow$ penguin(tweety).
bird(zazu)$\leftarrow$ penguin(zazu).
flies(zazu)$\leftarrow$ bird(zazu), $\neg$ ab(zazu).
ab(zazu)$\leftarrow$ penguin(zazu).

$$\left\{ \begin{array}{ccc} \mathit{penguin}(\mathit{tweety}) & \mathit{bird}(\mathit{tweety}) & \mathit{ab}(\mathit{tweety}) \\ \mathit{bird}(\mathit{zazu}) & \mathit{bird}(\mathit{zazu}) & \mathit{flies}(\mathit{zazu}) \end{array} \right\}$$

What are answer sets, and are there other semantics?

The ASP Workflow: Today's Focus

Propositional Program

Solver

Answer Sets

penguin(tweety).
bird(zazu).
bird(tweety)$\leftarrow$ penguin(tweety).
flies(tweety)$\leftarrow$ bird(tweety), $\neg$ ab(tweety).
ab(tweety)$\leftarrow$ penguin(tweety).
bird(zazu)$\leftarrow$ penguin(zazu).
flies(zazu)$\leftarrow$ bird(zazu), $\neg$ ab(zazu).
ab(zazu)$\leftarrow$ penguin(zazu).

$$\left\{ \begin{matrix} \mathit{penguin}(\mathit{tweety}) & \mathit{bird}(\mathit{tweety}) & \mathit{ab}(\mathit{tweety}) \\ \mathit{bird}(\mathit{zazu}) & \mathit{bird}(\mathit{zazu}) & \mathit{flies}(\mathit{zazu}) \end{matrix} \right\}$$

What are answer sets, and are there other semantics?†

†Interested in other aspects of logic programming? Take a look at https://teaching.potassco.org/.

Definite Logic Programs

Consider a set $\mathcal{A}$ of propositional atoms.

Definition

A **definite logic program** over $\mathcal{A}$ is a set P of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_m$$

for $a_0, \ldots, a_m \in \mathcal{A}$ with $0 \leq m$.

A set of definite Horn clauses (exactly one positive literal).

Definite Logic Programs

Consider a set $\mathcal{A}$ of propositional atoms.

Definition

A **definite logic program** over $\mathcal{A}$ is a set P of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_m$$

for $a_0, \ldots, a_m \in \mathcal{A}$ with $0 \leq m$.

A set of definite Horn clauses (exactly one positive literal).

Definition

- A set $S \subseteq \mathcal{A}$ is **closed** under a rule $a \leftarrow a_1, \ldots, a_m$ if and only if $\{a_1, \ldots, a_m\} \subseteq S$ implies $a \in S$.

Definite Logic Programs

Consider a set $\mathcal{A}$ of propositional atoms.

Definition

A **definite logic program** over $\mathcal{A}$ is a set P of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_m$$

for $a_0, \ldots, a_m \in \mathcal{A}$ with $0 \leq m$.

A set of definite Horn clauses (exactly one positive literal).

Definition

- A set $S \subseteq \mathcal{A}$ is **closed** under a rule $a \leftarrow a_1, \ldots, a_m$ if and only if $\{a_1, \ldots, a_m\} \subseteq S$ implies $a \in S$.
- The **least model** of P is the $\subseteq$-least set that is closed under all rules in P.

Definite Logic Programs

Consider a set $\mathcal{A}$ of propositional atoms.

Definition

A **definite logic program** over $\mathcal{A}$ is a set P of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_m$$

for $a_0, \ldots, a_m \in \mathcal{A}$ with $0 \leq m$.

A set of definite Horn clauses (exactly one positive literal).

Definition

- A set $S \subseteq \mathcal{A}$ is **closed** under a rule $a \leftarrow a_1, \ldots, a_m$ if and only if $\{a_1, \ldots, a_m\} \subseteq S$ implies $a \in S$.
- The **least model** of P is the $\subseteq$-least set that is closed under all rules in P.

Does such a least model always exist?

Semantics via Operators

Definition

Let P be a definite logic program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,\ \{a_1, \ldots, a_m\} \subseteq S\}$$

The T_P operator maps an interpretation S to a revised interpretation $T_P(S)$.

Semantics via Operators

Definition

Let P be a definite logic program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P, \ \{a_1, \ldots, a_m\} \subseteq S\}$$

The T_P operator maps an interpretation S to a revised interpretation $T_P(S)$.

Proposition

For any definite logic program P, the operator T_P is $\subseteq$-monotone.

Semantics via Operators

Definition

Let P be a definite logic program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,\ \{a_1, \ldots, a_m\} \subseteq S\}$$

The T_P operator maps an interpretation S to a revised interpretation $T_P(S)$.

Proposition

For any definite logic program P, the operator T_P is $\subseteq$-monotone.

Proof.

Let $S_1 \subseteq S_2 \subseteq \mathcal{A}$ and $a \in T_P(S_1)$.

Semantics via Operators

Definition

Let P be a definite logic program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P, \ \{a_1, \ldots, a_m\} \subseteq S\}$$

The T_P operator maps an interpretation S to a revised interpretation $T_P(S)$.

Proposition

For any definite logic program P, the operator T_P is $\subseteq$-monotone.

Proof.

Let $S_1 \subseteq S_2 \subseteq \mathcal{A}$ and $a \in T_P(S_1)$.
Then there is a rule $a \leftarrow a_1, \ldots, a_m \in P$ with $\{a_1, \ldots, a_m\} \subseteq S_1$.

Semantics via Operators

Definition

Let P be a definite logic program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P\colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,\ \{a_1, \ldots, a_m\} \subseteq S\}$$

The T_P operator maps an interpretation S to a revised interpretation $T_P(S)$.

Proposition

For any definite logic program P, the operator T_P is $\subseteq$-monotone.

Proof.

Let $S_1 \subseteq S_2 \subseteq \mathcal{A}$ and $a \in T_P(S_1)$.
Then there is a rule $a \leftarrow a_1, \ldots, a_m \in P$ with $\{a_1, \ldots, a_m\} \subseteq S_1$.
But then $\{a_1, \ldots, a_m\} \subseteq S_1 \subseteq S_2$, thus $a \in T_P(S_2)$. $\square$

Semantics via Operators

Definition

Let P be a definite logic program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,\ \{a_1, \ldots, a_m\} \subseteq S\}$$

The T_P operator maps an interpretation S to a revised interpretation $T_P(S)$.

Proposition

For any definite logic program P, the operator T_P is $\subseteq$-monotone.

Theorem

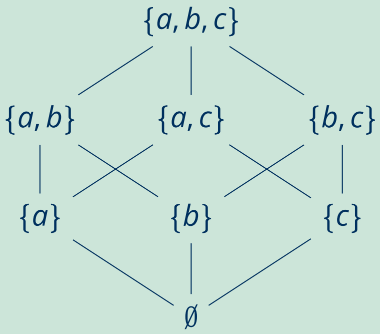
Every definite logic program P has a least model, given by the least fixpoint of T_P in $(2^{\mathcal{A}}, \subseteq)$.

The least model of P captures its intended meaning.

Definite Logic Programs: Example

Example

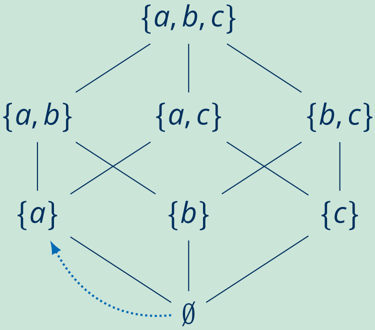
Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:



Definite Logic Programs: Example

Example

Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:

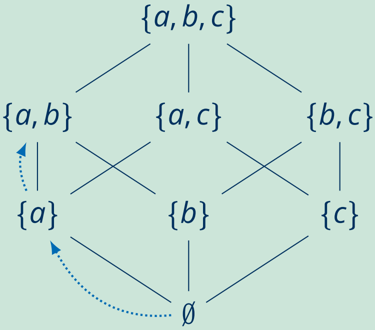


Definite Logic Programs: Example

Example

Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.

The operator T_P maps as follows:

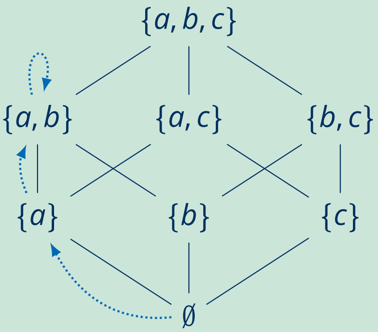


Definite Logic Programs: Example

Example

Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.

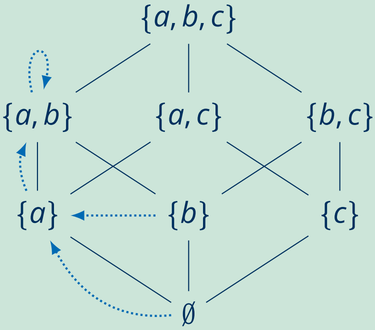
The operator T_P maps as follows:



Definite Logic Programs: Example

Example

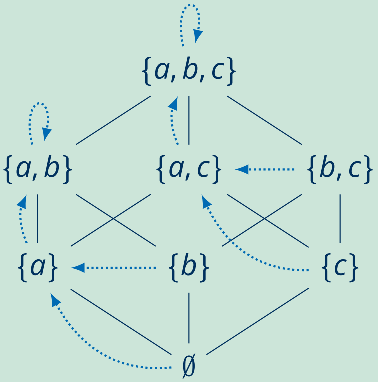
Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:



Definite Logic Programs: Example

Example

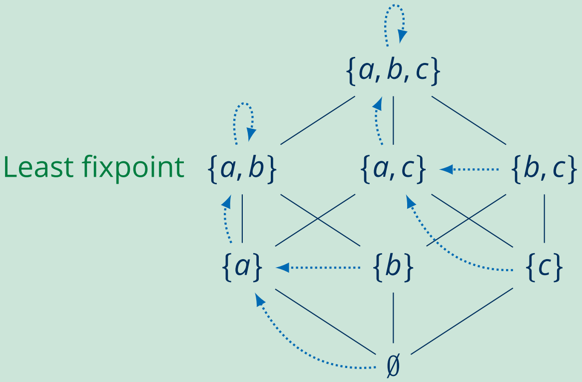
Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:



Definite Logic Programs: Example

Example

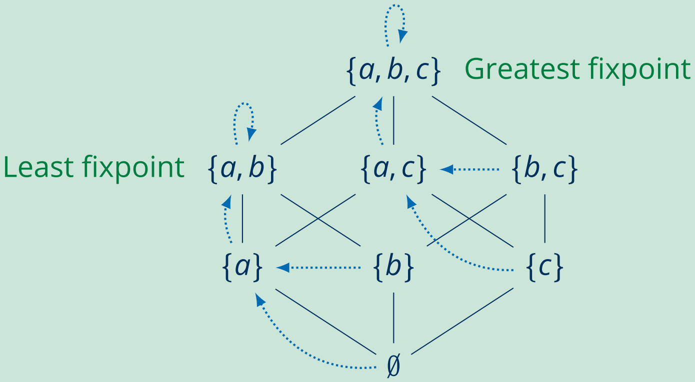
Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:



Definite Logic Programs: Example

Example

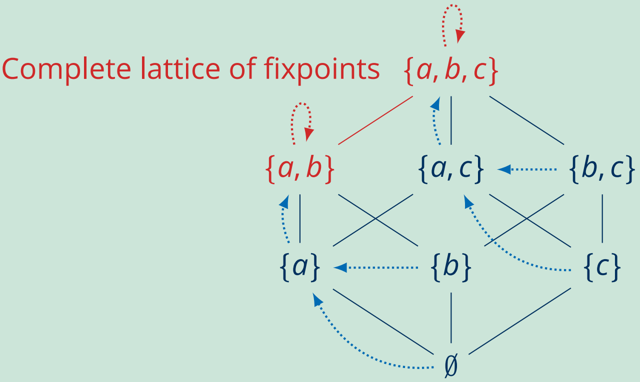
Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:



Definite Logic Programs: Example

Example

Consider $\mathcal{A} = \{a, b, c\}$ and the logic program $P = \{a \leftarrow, \quad b \leftarrow a, \quad c \leftarrow c\}$.
The operator T_P maps as follows:



Normal Logic Programs

Definition

A **normal logic program** over $\mathcal{A}$ is a set P of rules of the form
$a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n$ for $a_0, \ldots, a_n \in \mathcal{A}$ with $0 \leq m \leq n$.

Allow negated atoms ${\sim}a$ in rule bodies.

Normal Logic Programs

Definition

A **normal logic program** over $\mathcal{A}$ is a set P of rules of the form
$a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n$ for $a_0, \ldots, a_n \in \mathcal{A}$ with $0 \leq m \leq n$.

Allow negated atoms ${\sim}a$ in rule bodies.

Definition

Let P be a normal logic program. The operator T_P on $(2^{\mathcal{A}}, \subseteq)$ assigns thus:

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq S, \ \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A set $S \subseteq \mathcal{A}$ is a **supported model** of P iff it is a fixpoint of T_P.

Allow to derive the rule head from S whenever the rule body is satisfied in S.

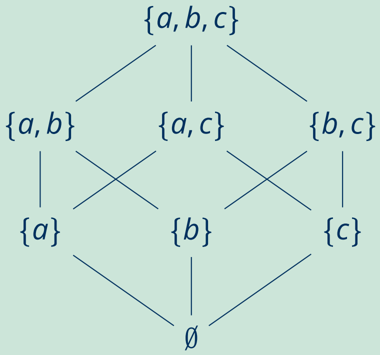
Alternative definition of supported models via Clark completion.

Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by

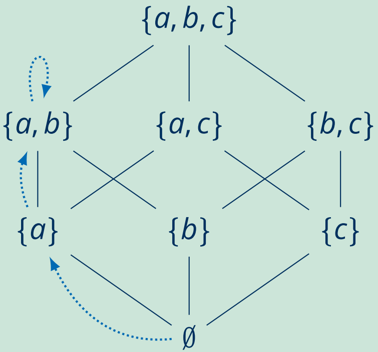


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by $\dashrightarrow$

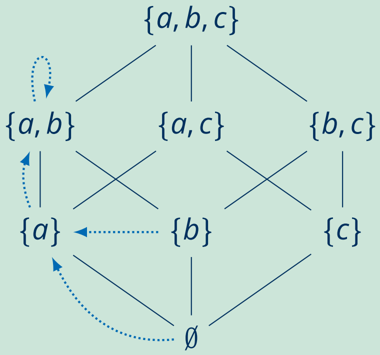


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by

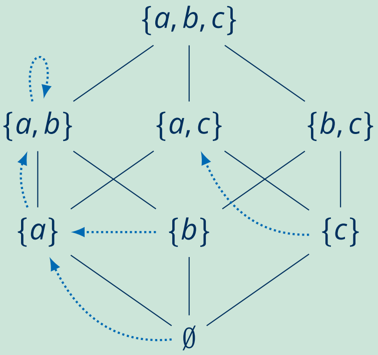


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by



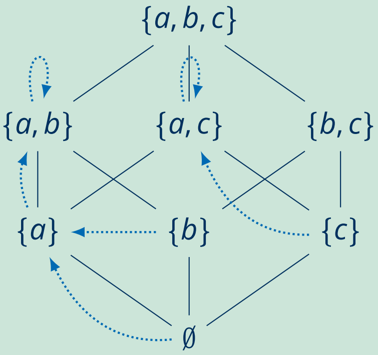
Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program

$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by $\dashrightarrow$

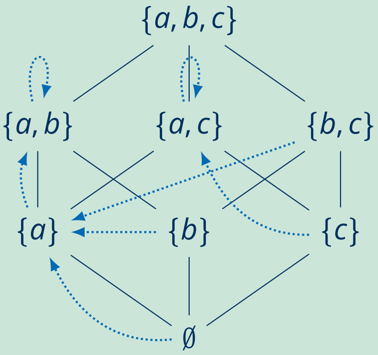


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by $\dashrightarrow$



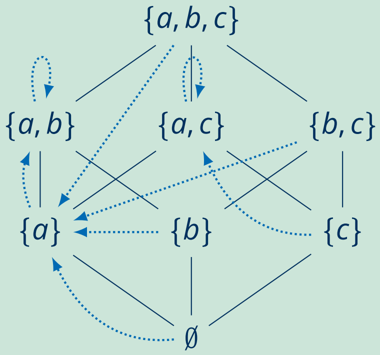
Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program

$$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}.$$

Operator T_P visualised by

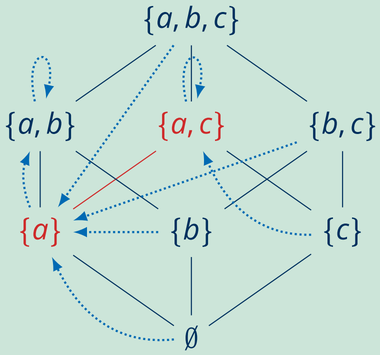


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by $\dashrightarrow$

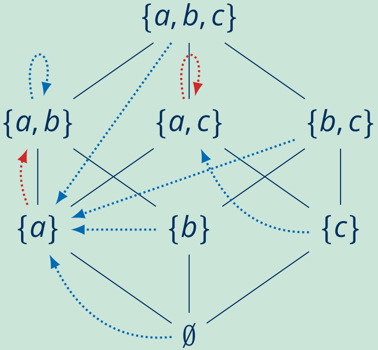


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by

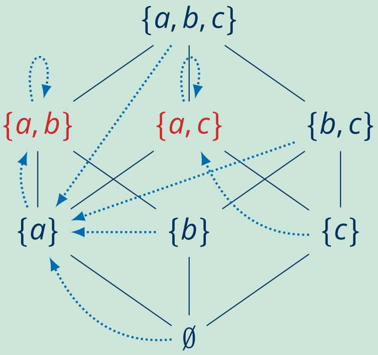


Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by ⇢



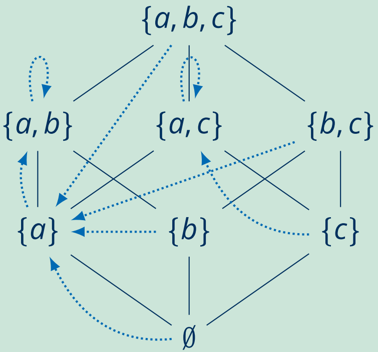
Normal Logic Programs: Example

Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by

T_P is not $\subseteq$-monotone.



Normal Logic Programs: Example

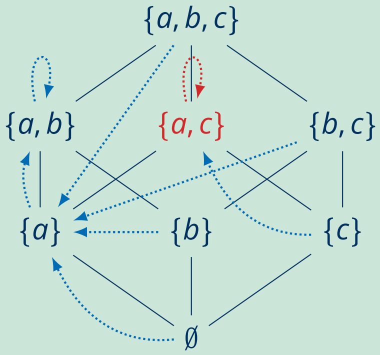
Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, {\sim}c, \quad c \leftarrow c, {\sim}b\}$.

Operator T_P visualised by

T_P is not $\subseteq$-monotone.

In $\{a, c\}$, atom c justifies itself.



Normal Logic Programs: Example

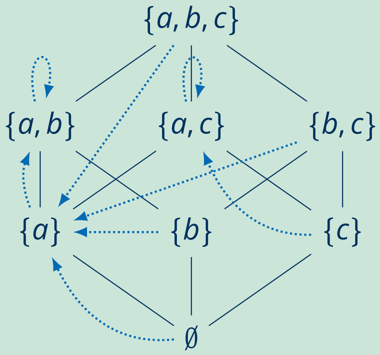
Example

Let $\mathcal{A} = \{a, b, c\}$.
Consider the normal logic program
$P = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c, \sim b\}$.

Operator T_P visualised by ⇢

T_P is not $\subseteq$-monotone.

In $\{a, c\}$, atom c justifies itself.



- How to avoid self-justification?
- How to obtain interpretation operators with “nice” properties?

Stable Model Semantics

Definition

Let P be a normal logic program and $S \subseteq \mathcal{A}$ be a set of atoms.
The **reduct of** P **with** S is the definite logic program P^S given by:

$$\{a \leftarrow a_1, \ldots, a_m \mid a \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A set $S \subseteq \mathcal{A}$ is a **stable model of** P iff S is the $\subseteq$-least model of P^S.

Stable Model Semantics

Definition

Let P be a normal logic program and $S \subseteq \mathcal{A}$ be a set of atoms.
The **reduct of** P **with** S is the definite logic program P^S given by:

$$\{a \leftarrow a_1, \ldots, a_m \mid a \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A set $S \subseteq \mathcal{A}$ is a **stable model of** P iff S is the $\subseteq$-least model of P^S.

In other words, P^S is obtained from P by:

- removing all rules containing $\sim a$ for some $a \in S$;
- removing all $\sim a$ from the remaining rules.

Stable Model Semantics

Definition

Let P be a normal logic program and $S \subseteq \mathcal{A}$ be a set of atoms.
The **reduct of** P **with** S is the definite logic program P^S given by:

$$\{a \leftarrow a_1, \ldots, a_m \mid a \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A set $S \subseteq \mathcal{A}$ is a **stable model of** P iff S is the $\subseteq$-least model of P^S.

In other words, P^S is obtained from P by:

- removing all rules containing ${\sim}a$ for some $a \in S$;
- removing all ${\sim}a$ from the remaining rules.

Example (Continued.)

Reconsider logic program $P = \{a \leftarrow, \quad b \leftarrow a, {\sim}c, \quad c \leftarrow c, {\sim}b\}$ with supported models $\{a, b\}$ and $\{a, c\}$. Are they stable models?

Stable Model Semantics

Definition

Let P be a normal logic program and $S \subseteq \mathcal{A}$ be a set of atoms.
The **reduct of** P **with** S is the definite logic program P^S given by:

$$\{a \leftarrow a_1, \ldots, a_m \mid a \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A set $S \subseteq \mathcal{A}$ is a **stable model of** P iff S is the $\subseteq$-least model of P^S.

In other words, P^S is obtained from P by:

- removing all rules containing ${\sim}a$ for some $a \in S$;
- removing all ${\sim}a$ from the remaining rules.

Example (Continued.)

Reconsider logic program $P = \{a \leftarrow, \quad b \leftarrow a, {\sim}c, \quad c \leftarrow c, {\sim}b\}$ with supported models $\{a, b\}$ and $\{a, c\}$. Are they stable models?

- $P^{\{a,b\}} = \{a \leftarrow, \quad b \leftarrow a\}$ with least model $\{a, b\}$, so $\{a, b\}$ is a stable model.

Stable Model Semantics

Definition

Let P be a normal logic program and $S \subseteq \mathcal{A}$ be a set of atoms.
The **reduct of** P **with** S is the definite logic program P^S given by:

$$\{a \leftarrow a_1, \ldots, a_m \mid a \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A set $S \subseteq \mathcal{A}$ is a **stable model of** P iff S is the $\subseteq$-least model of P^S.

In other words, P^S is obtained from P by:

- removing all rules containing ${\sim}a$ for some $a \in S$;
- removing all ${\sim}a$ from the remaining rules.

Example (Continued.)

Reconsider logic program $P = \{a \leftarrow, \quad b \leftarrow a, {\sim}c, \quad c \leftarrow c, {\sim}b\}$ with supported models $\{a, b\}$ and $\{a, c\}$. Are they stable models?

- $P^{\{a,b\}} = \{a \leftarrow, \quad b \leftarrow a\}$ with least model $\{a, b\}$, so $\{a, b\}$ is a stable model.
- $P^{\{a,c\}} = \{a \leftarrow, \quad c \leftarrow c\}$ with least model $\{a\}$, so $\{a, c\}$ is not stable.

Stocktaking

- Monotone operators in complete lattices have (least and greatest) fixpoints.
- Operators can be associated with knowledge bases such that their fixpoints correspond to models.
- Definite logic programs lead to an operator that is monotone on $(2^{\mathcal{A}}, \subseteq)$, and thus have unique least models.
- Normal logic programs lead to a non-monotone operator; model existence and uniqueness cannot be guaranteed.
- Stable model semantics deals with self-justification.
- Can we find an operator-based version of stable model semantics?

Approximating Operators

Approximating Operators

Main Idea

Use a more fine-grained structure to keep track of (partial) truth values.

Desiderata

- Preserve “interpretation revision” character of operators
- Preserve correspondence of fixpoints with models
- Obtain useful properties of operators

Approach

- Approximate sets of models by intervals.
- Use an information ordering on these approximations.
- Approximate operators by approximators – operators on intervals.
- Guarantee that fixpoints of approximators contain original fixpoints.

From Lattices to Bilattices

Definition

Let $(L, \leqslant)$ be a partially ordered set.
Its associated **information bilattice** is $(L^2, \leq_i)$ with $L^2 = L \times L$ and

$$(u, v) \leq_i (x, y) \quad \text{iff} \quad u \leqslant x \text{ and } y \leqslant v$$

- A pair (x, y) approximates all $z \in L$ with $x \leqslant z \leqslant y$.
- Information ordering $\hat{=}$ interval inclusion: $(u, v) \leq_i (x, y)$ iff $[x, y] \subseteq [u, v]$

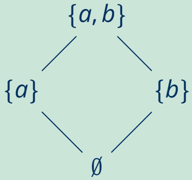
Proposition

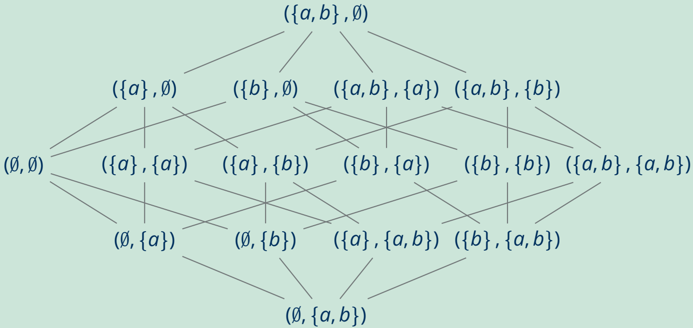
If $(L, \leqslant)$ is a complete lattice, then $(L^2, \leq_i)$ is a complete lattice. For $S \subseteq L^2$:

$$\bigwedge_i S = \left(\bigwedge S_1, \bigvee S_2\right) \quad \text{and} \quad \bigvee_i S = \left(\bigvee S_1, \bigwedge S_2\right) \qquad \begin{array}{l} S_1 = \{x \mid (x,y) \in S\} \\ S_2 = \{y \mid (x,y) \in S\} \end{array}$$

From Lattice to Bilattice: Example

Example



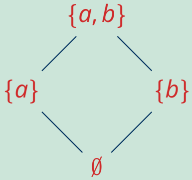


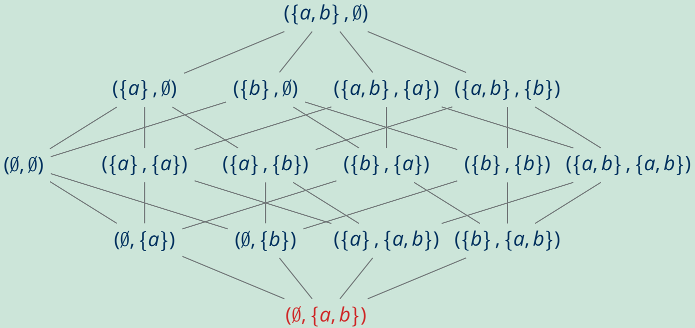
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

From Lattice to Bilattice: Example

Example



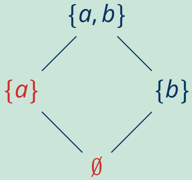


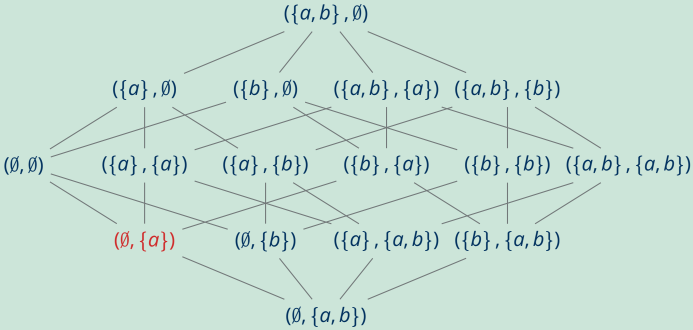
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

From Lattice to Bilattice: Example

Example



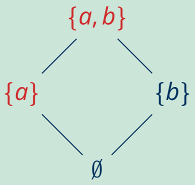


Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

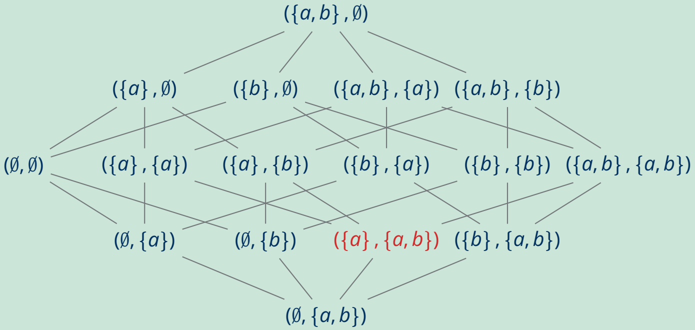
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

From Lattice to Bilattice: Example

Example



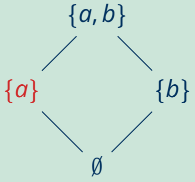
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$



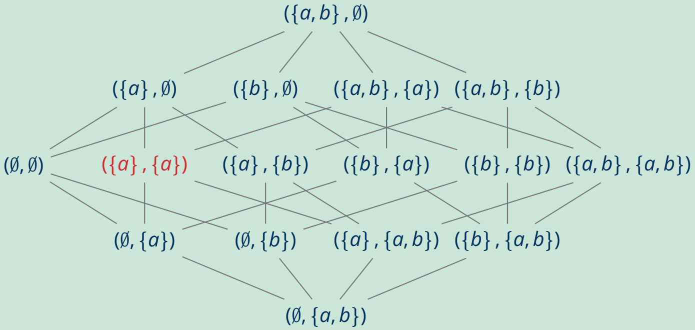
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

From Lattice to Bilattice: Example

Example



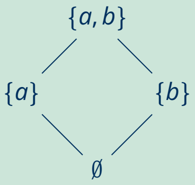
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$



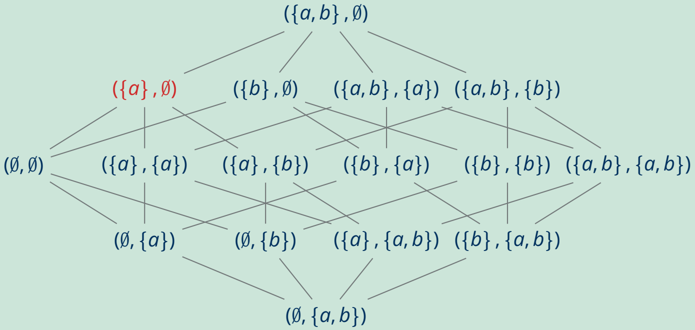
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

From Lattice to Bilattice: Example

Example



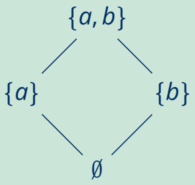
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

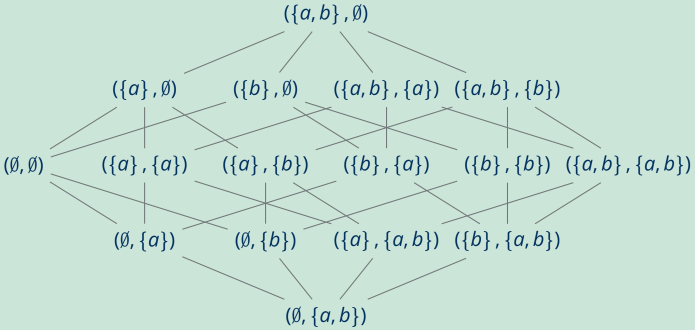


Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

From Lattice to Bilattice: Example

Example



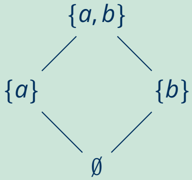


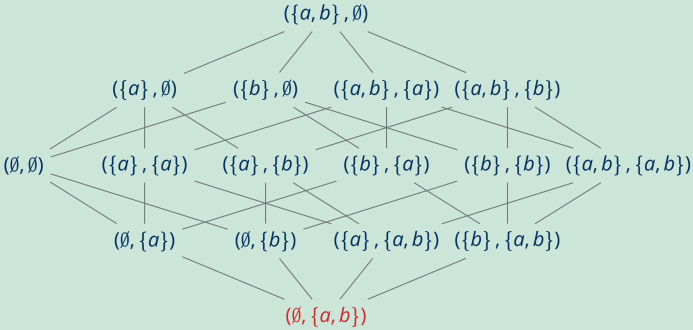
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$ Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Pairs in the bilattice correspond to four-valued interpretations $v \colon \{a, b\} \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$.

From Lattice to Bilattice: Example

Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

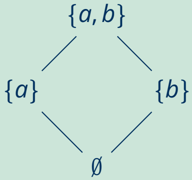
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

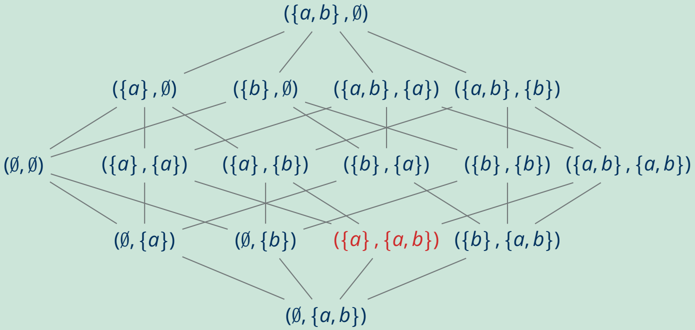
Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a,b\} \to \{\mathbf{t},\mathbf{f},\mathbf{u},\mathbf{i}\}$.

$$\{a \mapsto \mathbf{u}, b \mapsto \mathbf{u}\}$$

From Lattice to Bilattice: Example

Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

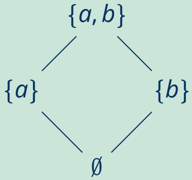
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

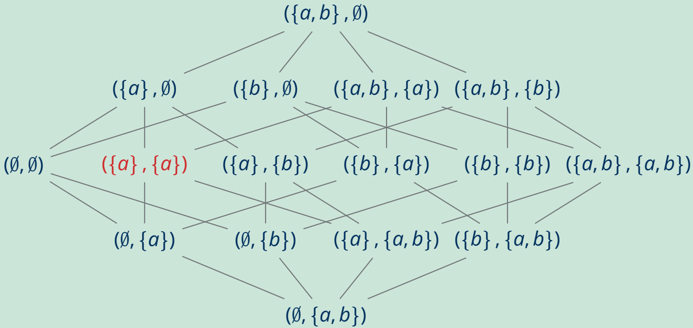
Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a, b\} \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$.

$$\{a \mapsto \mathbf{t}, b \mapsto \mathbf{u}\}$$

From Lattice to Bilattice: Example

Example





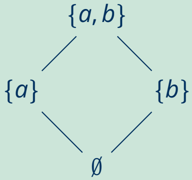
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$ Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a,b\} \to \{\mathbf{t},\mathbf{f},\mathbf{u},\mathbf{i}\}$.

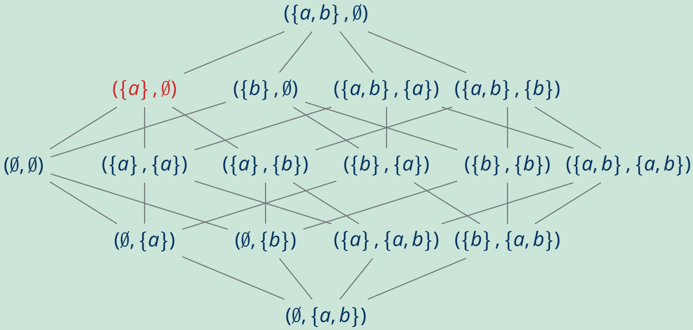
$$\{a \mapsto \mathbf{t}, b \mapsto \mathbf{f}\}$$

From Lattice to Bilattice: Example

Example



Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$



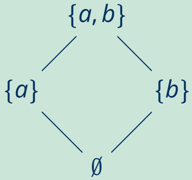
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

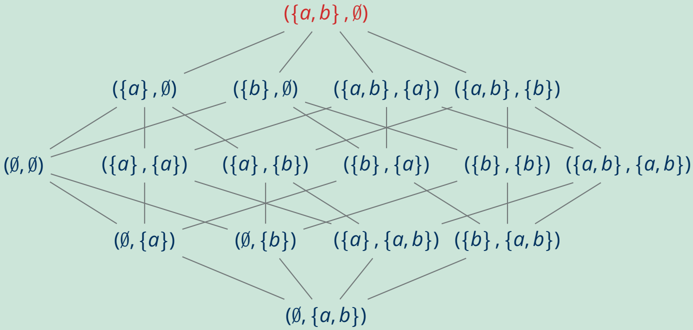
Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a, b\} \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$.

$$\{a \mapsto \mathbf{i}, b \mapsto \mathbf{f}\}$$

From Lattice to Bilattice: Example

Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

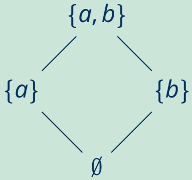
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

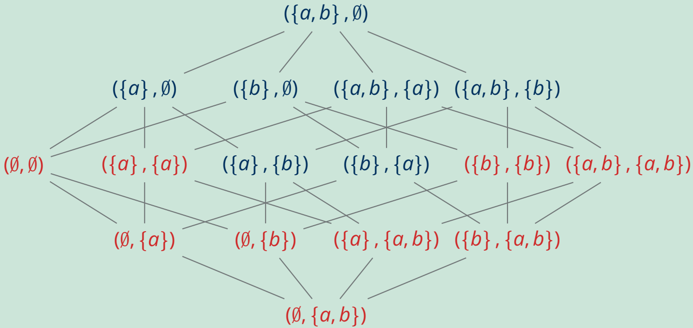
Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a,b\} \to \{\mathbf{t},\mathbf{f},\mathbf{u},\mathbf{i}\}$.

$$\{a \mapsto \mathbf{i}, b \mapsto \mathbf{i}\}$$

From Lattice to Bilattice: Example

Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

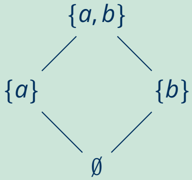
Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

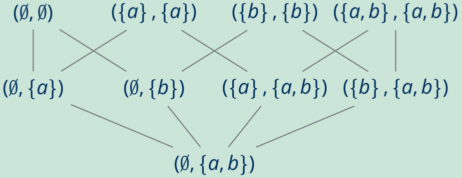
Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a,b\} \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$.

We will mostly be concerned with the consistent pairs.

From Lattice to Bilattice: Example

Example





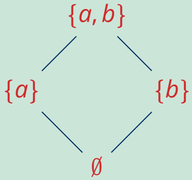
Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$ Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

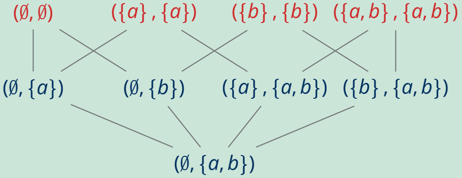
Pairs in the bilattice correspond to four-valued interpretations $v \colon \{a, b\} \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$.

We will mostly be concerned with the consistent pairs.

From Lattice to Bilattice: Example

Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Pairs in the bilattice correspond to four-valued interpretations $v\colon \{a, b\} \to \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\}$.

Elements of the original lattice correspond to exact pairs.

Approximator

Recall approach: Approximate lattice operators on a richer structure.

Definition

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be an operator.
An operator $\mathcal{A} \colon L^2 \to L^2$ **approximates** O iff for all $x \in L$, we have

$$\mathcal{A}(x, x) = (O(x), O(x))$$

$\mathcal{A}$ is an **approximator** iff $\mathcal{A}$ approximates some O and $\mathcal{A}$ is $\leq_i$-monotone.

Approximator coincides with the operator on exact pairs.

Approximator

Recall approach: Approximate lattice operators on a richer structure.

Definition

Let $(L, \leqslant)$ be a complete lattice and $O \colon L \to L$ be an operator.
An operator $\mathcal{A} \colon L^2 \to L^2$ **approximates** O iff for all $x \in L$, we have

$$\mathcal{A}(x, x) = (O(x), O(x))$$

$\mathcal{A}$ is an **approximator** iff $\mathcal{A}$ approximates some O and $\mathcal{A}$ is $\leq_i$-monotone.

Approximator coincides with the operator on exact pairs.

$\mathcal{A} \colon L^2 \to L^2$ induces $\mathcal{A}', \mathcal{A}'' \colon L^2 \to L$ with $\mathcal{A}(x, y) = (\mathcal{A}'(x, y), \mathcal{A}''(x, y))$.

Approximator

Recall approach: Approximate lattice operators on a richer structure.

Definition

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be an operator.
An operator $\mathcal{A}\colon L^2 \to L^2$ **approximates** O iff for all $x \in L$, we have

$$\mathcal{A}(x, x) = (O(x), O(x))$$

$\mathcal{A}$ is an **approximator** iff $\mathcal{A}$ approximates some O and $\mathcal{A}$ is $\leq_i$-monotone.

Approximator coincides with the operator on exact pairs.

$\mathcal{A}\colon L^2 \to L^2$ induces $\mathcal{A}', \mathcal{A}''\colon L^2 \to L$ with $\mathcal{A}(x, y) = (\mathcal{A}'(x, y), \mathcal{A}''(x, y))$.

Definition

An approximator is **symmetric** iff $\mathcal{A}'(x, y) = \mathcal{A}''(y, x)$.

Approximator

Recall approach: Approximate lattice operators on a richer structure.

Definition

Let $(L, \leqslant)$ be a complete lattice and $O\colon L \to L$ be an operator.
An operator $\mathcal{A}\colon L^2 \to L^2$ **approximates** O iff for all $x \in L$, we have

$$\mathcal{A}(x, x) = (O(x), O(x))$$

$\mathcal{A}$ is an **approximator** iff $\mathcal{A}$ approximates some O and $\mathcal{A}$ is $\leq_i$-monotone.

Approximator coincides with the operator on exact pairs.

$\mathcal{A}\colon L^2 \to L^2$ induces $\mathcal{A}', \mathcal{A}''\colon L^2 \to L$ with $\mathcal{A}(x, y) = (\mathcal{A}'(x, y), \mathcal{A}''(x, y))$.

Definition

An approximator is **symmetric** iff $\mathcal{A}'(x, y) = \mathcal{A}''(y, x)$.

If $\mathcal{A}$ is symmetric, then $\mathcal{A}(x, y) = (\mathcal{A}'(x, y), \mathcal{A}'(y, x))$, so $\mathcal{A}'$ fully specifies $\mathcal{A}$.

Approximator: Example

Example

Let P be a normal logic program.
Recall its one-step consequence operator T_P, defined by

$$T_P(S) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq S, \ \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

Approximator: Example

Example

Let P be a normal logic program.
Recall its one-step consequence operator T_P, defined by

$$T_P(S) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq S, \ \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A symmetric approximator for T_P is given by $\mathcal{T}_P$ with

$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L, \ \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

Approximator: Example

Example

Let P be a normal logic program.
Recall its one-step consequence operator T_P, defined by

$$T_P(S) = \{a_0 \in \mathcal{A} \mid \; a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq S, \; \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

A symmetric approximator for T_P is given by $\mathcal{T}_P$ with

$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid \; a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L, \; \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

That is, $\mathcal{T}_P(L, U) = \big(\mathcal{T}_P{}'(L, U), \mathcal{T}_P{}'(U, L)\big)$.

Approximator: Example

Example

Let P be a normal logic program.
Recall its one-step consequence operator T_P, defined by

$$T_P(S) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq S, \; \{a_{m+1}, \ldots, a_n\} \cap S = \emptyset\}$$

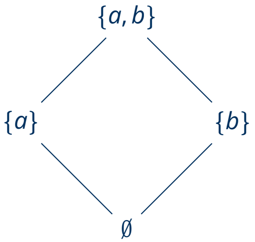
A symmetric approximator for T_P is given by $\mathcal{T}_P$ with

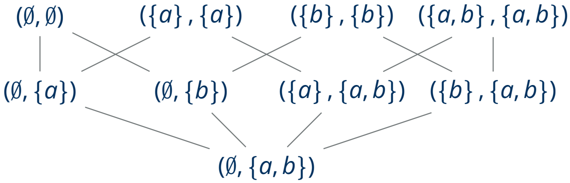
$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L, \; \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

That is, $\mathcal{T}_P(L, U) = \big(\mathcal{T}_P{}'(L, U), \mathcal{T}_P{}'(U, L)\big)$.

For new lower bound: check truth against lower, falsity against upper bound.

Approximator $\mathcal{T}_P$: Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

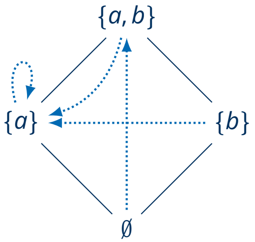
Normal logic program
$P = \{a \leftarrow, \quad b \leftarrow \sim a, \sim b\}$

T_P: ········►

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{T}_P$ for T_P: - - ►

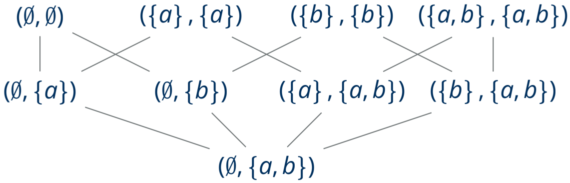
Approximator $\mathcal{T}_P$: Example



Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Normal logic program
$P = \{a \leftarrow, \quad b \leftarrow \sim a, \sim b\}$

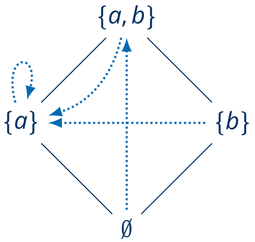
T_P:

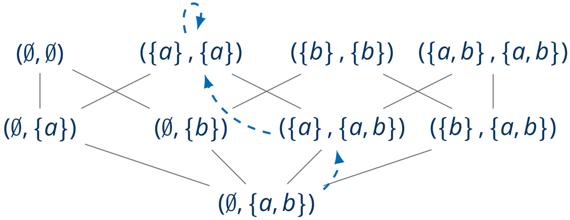


Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{T}_P$ for T_P:

Approximator $\mathcal{T}_P$: Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

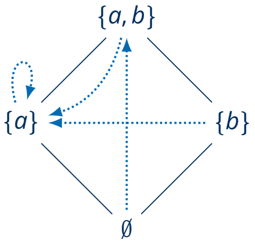
Normal logic program
$P = \{a \leftarrow, \quad b \leftarrow \sim a, \sim b\}$

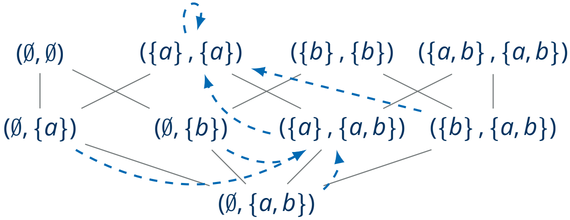
T_P: $\cdots\cdots\blacktriangleright$

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{T}_P$ for T_P: $- - \blacktriangleright$

Approximator $\mathcal{T}_P$: Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

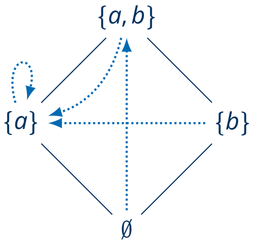
Normal logic program
$P = \{a \leftarrow, \quad b \leftarrow \sim a, \sim b\}$

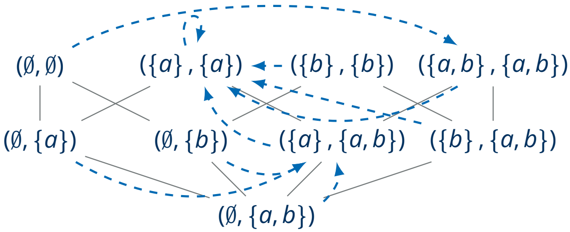
T_P: ·········▶

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{T}_P$ for T_P: - - -▶

Approximator $\mathcal{T}_P$: Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Normal logic program
$P = \{a \leftarrow, \quad b \leftarrow \sim a, \sim b\}$

T_P:

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{T}_P$ for T_P:

Quiz: Approximator $\mathcal{T}_P$

Recall that for $L, U \subseteq \mathcal{A}$ we defined $\mathcal{T}_P(L, U) = \left(\mathcal{T}_P{}'(L, U), \mathcal{T}_P{}'(U, L)\right)$ with

$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L, \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

Quiz

Consider the normal logic program P:

$$a \leftarrow \qquad b \leftarrow a, \sim c \qquad c \leftarrow c$$

What is the result of applying $\mathcal{T}_P$ to $(\{a\}, \{a, b\})$?

1. $(\emptyset, \{a, b\})$
2. $(\{a\}, \{a, b\})$
3. $(\{a, b\}, \{a, b\})$
4. $(\{a, b, c\}, \{a, b, c\})$

Quiz: Approximator $\mathcal{T}_P$

Recall that for $L, U \subseteq \mathcal{A}$ we defined $\mathcal{T}_P(L, U) = \big(\mathcal{T}_P{}'(L, U), \mathcal{T}_P{}'(U, L)\big)$ with

$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P,$$
$$\{a_1, \ldots, a_m\} \subseteq L, \; \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

Quiz

Consider the normal logic program P:

$$a \leftarrow \qquad\qquad b \leftarrow a, \sim c \qquad\qquad c \leftarrow c$$

What is the result of applying $\mathcal{T}_P$ to $(\{a\}, \{a, b\})$?

1. $(\emptyset, \{a, b\})$ ✗	2. $(\{a\}, \{a, b\})$
3. $(\{a, b\}, \{a, b\})$	4. $(\{a, b, c\}, \{a, b, c\})$

Quiz: Approximator $\mathcal{T}_P$

Recall that for $L, U \subseteq \mathcal{A}$ we defined $\mathcal{T}_P(L, U) = \big(\mathcal{T}_P'(L, U), \mathcal{T}_P'(U, L)\big)$ with

$$\mathcal{T}_P'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L, \ \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

Quiz

Consider the normal logic program P:

$$a \leftarrow \qquad b \leftarrow a, \sim c \qquad c \leftarrow c$$

What is the result of applying $\mathcal{T}_P$ to $(\{a\}, \{a, b\})$?

1. $(\emptyset, \{a, b\})$ ✗
2. $(\{a\}, \{a, b\})$
3. $(\{a, b\}, \{a, b\})$
4. $(\{a, b, c\}, \{a, b, c\})$ ✗

Quiz: Approximator $\mathcal{T}_P$

Recall that for $L, U \subseteq \mathcal{A}$ we defined $\mathcal{T}_P(L, U) = \big(\mathcal{T}_P{}'(L, U), \mathcal{T}_P{}'(U, L)\big)$ with

$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L, \ \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

Quiz

Consider the normal logic program P:

$$a \leftarrow \qquad b \leftarrow a, \sim c \qquad c \leftarrow c$$

What is the result of applying $\mathcal{T}_P$ to $(\{a\}, \{a, b\})$?

1. $(\emptyset, \{a, b\})$ ✗
2. $(\{a\}, \{a, b\})$ ✗
3. $(\{a, b\}, \{a, b\})$
4. $(\{a, b, c\}, \{a, b, c\})$ ✗

Quiz: Approximator $\mathcal{T}_P$

Recall that for $L, U \subseteq \mathcal{A}$ we defined $\mathcal{T}_P(L, U) = \big(\mathcal{T}_P{}'(L, U), \mathcal{T}_P{}'(U, L)\big)$ with

$$\mathcal{T}_P{}'(L, U) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n \in P, \\ \{a_1, \ldots, a_m\} \subseteq L,\ \{a_{m+1}, \ldots, a_n\} \cap U = \emptyset\}$$

Quiz

Consider the normal logic program P:

$$a \leftarrow \qquad b \leftarrow a, {\sim}c \qquad c \leftarrow c$$

What is the result of applying $\mathcal{T}_P$ to $(\{a\}, \{a, b\})$?

1. $(\emptyset, \{a, b\})$ ✗
2. $(\{a\}, \{a, b\})$ ✗
3. $(\{a, b\}, \{a, b\})$ ✓
4. $(\{a, b, c\}, \{a, b, c\})$ ✗

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Proof.

1. Let $a, b \in C$. Since C is a chain, $a \leq_i b$ (then $a' \leqslant b' \leqslant b''$) or $b \leq_i a$ (then $a' \leqslant a'' \leqslant b''$).

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Proof.

1. Let $a, b \in C$. Since C is a chain, $a \leq_i b$ (then $a' \leqslant b' \leqslant b''$) or $b \leq_i a$ (then $a' \leqslant a'' \leqslant b''$). In any case, $a' \leqslant b''$. So every $c'' \in C''$ is an upper bound of C', and $\bigvee C' \leqslant c''$.

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Proof.

1. Let $a, b \in C$. Since C is a chain, $a \leq_i b$ (then $a' \leqslant b' \leqslant b''$) or $b \leq_i a$ (then $a' \leqslant a'' \leqslant b''$). In any case, $a' \leqslant b''$. So every $c'' \in C''$ is an upper bound of C', and $\bigvee C' \leqslant c''$. Hence $\bigvee C'$ is a lower bound of C'' and $\bigvee C' \leqslant \bigwedge C''$.

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Proof.

1. Let $a, b \in C$. Since C is a chain, $a \leq_i b$ (then $a' \leqslant b' \leqslant b''$) or $b \leq_i a$ (then $a' \leqslant a'' \leqslant b''$). In any case, $a' \leqslant b''$. So every $c'' \in C''$ is an upper bound of C', and $\bigvee C' \leqslant c''$. Hence $\bigvee C'$ is a lower bound of C'' and $\bigvee C' \leqslant \bigwedge C''$.
2. If $x \leqslant y$, then for z with $x \leqslant z \leqslant y$ we have $(x, y) \leq_i (z, z)$. $\mathcal{A}$ is $\leq_i$-monotone, thus $\mathcal{A}(x, y) \leq_i \mathcal{A}(z, z)$.

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Proof.

1. Let $a, b \in C$. Since C is a chain, $a \leq_i b$ (then $a' \leqslant b' \leqslant b''$) or $b \leq_i a$ (then $a' \leqslant a'' \leqslant b''$). In any case, $a' \leqslant b''$. So every $c'' \in C''$ is an upper bound of C', and $\bigvee C' \leqslant c''$. Hence $\bigvee C'$ is a lower bound of C'' and $\bigvee C' \leqslant \bigwedge C''$.
2. If $x \leqslant y$, then for z with $x \leqslant z \leqslant y$ we have $(x, y) \leq_i (z, z)$. $\mathcal{A}$ is $\leq_i$-monotone, thus $\mathcal{A}(x, y) \leq_i \mathcal{A}(z, z)$. $\mathcal{A}$ approximates some O, thus $\mathcal{A}(z, z) = (O(z), O(z))$.

Approximator: Observations (1)

Lemma

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ an approximator on $(L^2, \leq_i)$.

1. If C is a non-empty chain of consistent pairs, then $\bigvee_i C$ is consistent.
2. If (x, y) is consistent, then $\mathcal{A}(x, y)$ is consistent.

Approximators map consistent pairs to consistent pairs.

Proof.

1. Let $a, b \in C$. Since C is a chain, $a \leq_i b$ (then $a' \leqslant b' \leqslant b''$) or $b \leq_i a$ (then $a' \leqslant a'' \leqslant b''$). In any case, $a' \leqslant b''$. So every $c'' \in C''$ is an upper bound of C', and $\bigvee C' \leqslant c''$. Hence $\bigvee C'$ is a lower bound of C'' and $\bigvee C' \leqslant \bigwedge C''$.
2. If $x \leqslant y$, then for z with $x \leqslant z \leqslant y$ we have $(x, y) \leq_i (z, z)$. $\mathcal{A}$ is $\leq_i$-monotone, thus $\mathcal{A}(x, y) \leq_i \mathcal{A}(z, z)$. $\mathcal{A}$ approximates some O, thus $\mathcal{A}(z, z) = (O(z), O(z))$. In combination $\mathcal{A}'(x, y) \leqslant O(z) \leqslant \mathcal{A}''(x, y)$. $\square$

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O \colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O \colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*).

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O\colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent:

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O \colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent: Define $Q = \left\{(x, y) \in L^2 \;\middle|\; x \leqslant y \;\;\&\;\; (x, y) \leq_i \mathcal{A}(x, y) \;\;\&\;\; (x, y) \leq_i (x^*, y^*)\right\}$.

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O\colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent:
   Define $Q = \left\{(x,y) \in L^2 \;\middle|\; x \leqslant y \;\;\&\;\; (x,y) \leq_i \mathcal{A}(x,y) \;\;\&\;\; (x,y) \leq_i (x^*, y^*)\right\}$.
   Q is non-empty as $(\bot, \top) \in Q$.

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O\colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent: Define $Q = \left\{(x,y) \in L^2 \;\middle|\; x \leqslant y \;\;\&\;\; (x,y) \leq_i \mathcal{A}(x,y) \;\;\&\;\; (x,y) \leq_i (x^*,y^*)\right\}$. Q is non-empty as $(\bot, \top) \in Q$. Each non-empty chain in Q has an upper bound in Q,

Let $\emptyset \neq C \subseteq Q$ be a chain. Define $d = \bigvee_i C$. (1) By the previous lemma, d is consistent. (2) For every $c \in C$ we have $c \leq_i d$ and thus $c \leq_i \mathcal{A}(c) \leq_i \mathcal{A}(d)$; thus $\mathcal{A}(d)$ is an upper bound of C, whence $d \leq_i \mathcal{A}(d)$. (3) We know that $C \subseteq Q$ whence (x^*, y^*) is an upper bound of C, thus $d \leq_i (x^*, y^*)$.

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O\colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent: Define $Q = \left\{(x, y) \in L^2 \;\middle|\; x \leqslant y \;\;\&\;\; (x, y) \leq_i \mathcal{A}(x, y) \;\;\&\;\; (x, y) \leq_i (x^*, y^*)\right\}$. Q is non-empty as $(\bot, \top) \in Q$. Each non-empty chain in Q has an upper bound in Q, therefore by Zorn's Lemma, Q has a maximal element, p.

Let $\emptyset \neq C \subseteq Q$ be a chain. Define $d = \bigvee_i C$. (1) By the previous lemma, d is consistent. (2) For every $c \in C$ we have $c \leq_i d$ and thus $c \leq_i \mathcal{A}(c) \leq_i \mathcal{A}(d)$; thus $\mathcal{A}(d)$ is an upper bound of C, whence $d \leq_i \mathcal{A}(d)$. (3) We know that $C \subseteq Q$ whence (x^*, y^*) is an upper bound of C, thus $d \leq_i (x^*, y^*)$.

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O\colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent: Define $Q = \left\{(x, y) \in L^2 \,\middle|\, x \leqslant y \;\;\&\;\; (x, y) \leq_i \mathcal{A}(x, y) \;\;\&\;\; (x, y) \leq_i (x^*, y^*)\right\}$. Q is non-empty as $(\bot, \top) \in Q$. Each non-empty chain in Q has an upper bound in Q, therefore by Zorn's Lemma, Q has a maximal element, ρ. Since ρ is maximal, $\rho \leq_i \mathcal{A}(\rho)$ directly yields $\mathcal{A}(\rho) = \rho = (x^*, y^*)$.

Let $\emptyset \neq C \subseteq Q$ be a chain. Define $d = \bigvee_i C$. (1) By the previous lemma, d is consistent. (2) For every $c \in C$ we have $c \leq_i d$ and thus $c \leq_i \mathcal{A}(c) \leq_i \mathcal{A}(d)$; thus $\mathcal{A}(d)$ is an upper bound of C, whence $d \leq_i \mathcal{A}(d)$. (3) We know that $C \subseteq Q$ whence (x^*, y^*) is an upper bound of C, thus $d \leq_i (x^*, y^*)$.

Approximator: Observations (2)

Theorem

Let $(L, \leqslant)$ be a complete lattice with $O \colon L \to L$, and $\mathcal{A}$ an approximator for O.

1. $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*) with $x^* \leqslant y^*$.
2. Every fixpoint z of O satisfies $x^* \leqslant z \leqslant y^*$.

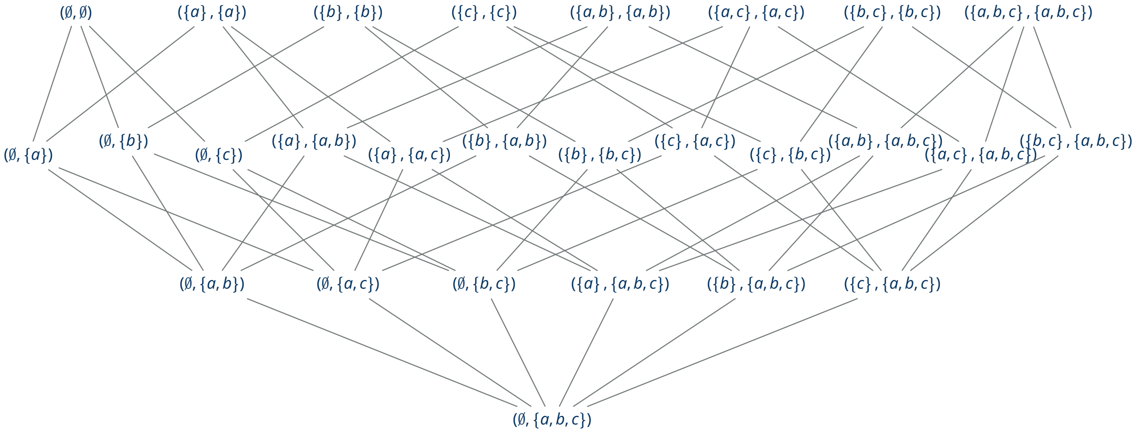
The least fixpoint of $\mathcal{A}$ is consistent and approximates all fixpoints of O.

Proof.

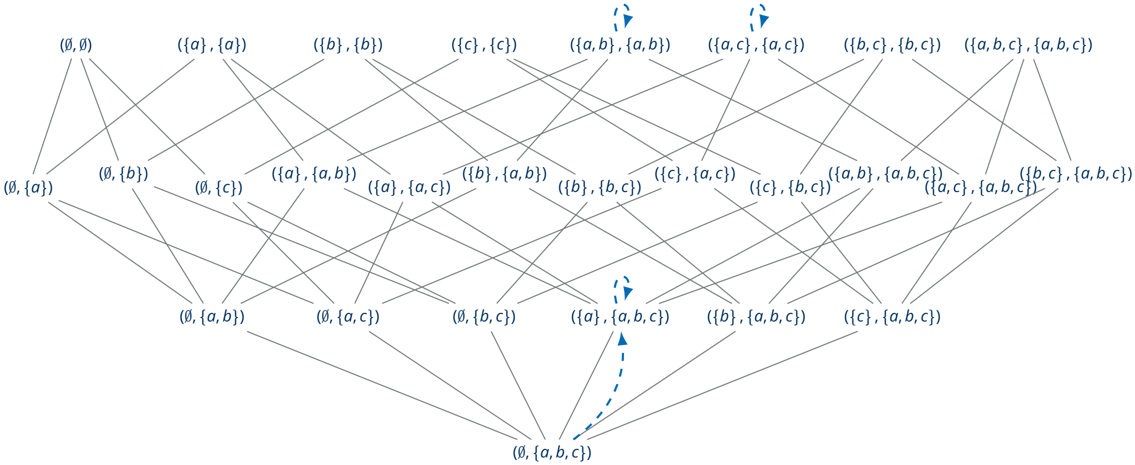
1. By Knaster/Tarski, $\mathcal{A}$ has a $\leq_i$-least fixpoint (x^*, y^*). It is also consistent: Define $Q = \left\{(x, y) \in L^2 \;\middle|\; x \leqslant y \;\;\&\;\; (x, y) \leq_i \mathcal{A}(x, y) \;\;\&\;\; (x, y) \leq_i (x^*, y^*)\right\}$. Q is non-empty as $(\bot, \top) \in Q$. Each non-empty chain in Q has an upper bound in Q, therefore by Zorn's Lemma, Q has a maximal element, ρ. Since ρ is maximal, $\rho \leq_i \mathcal{A}(\rho)$ directly yields $\mathcal{A}(\rho) = \rho = (x^*, y^*)$.
2. If $O(z) = z$ then $\mathcal{A}(z, z) = (O(z), O(z)) = (z, z)$ and $(x^*, y^*) \leq_i (z, z)$. $\square$

Let $\emptyset \neq C \subseteq Q$ be a chain. Define $d = \bigvee_i C$. (1) By the previous lemma, d is consistent. (2) For every $c \in C$ we have $c \leq_i d$ and thus $c \leq_i \mathcal{A}(c) \leq_i \mathcal{A}(d)$; thus $\mathcal{A}(d)$ is an upper bound of C, whence $d \leq_i \mathcal{A}(d)$. (3) We know that $C \subseteq Q$ whence (x^*, y^*) is an upper bound of C, thus $d \leq_i (x^*, y^*)$.

Approximator $\mathcal{T}_P$: Examples

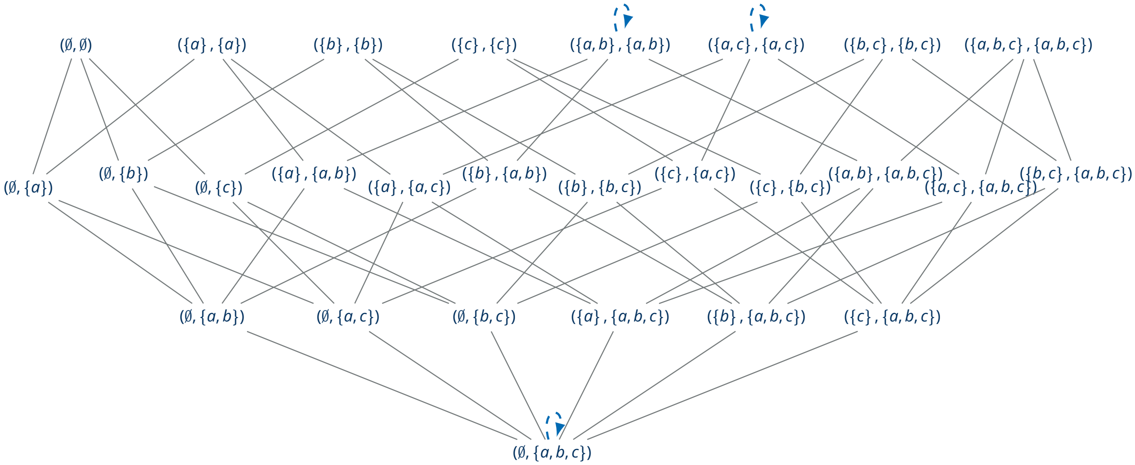


Approximator $\mathcal{T}_P$: Examples



$$P_1 = \{a \leftarrow, \quad b \leftarrow a, {\sim}c, \quad c \leftarrow c\}$$

Approximator $\mathcal{T}_P$: Examples



$$P_2 = \{a \leftarrow b, \quad a \leftarrow c, \quad b \leftarrow {\sim}c, \quad c \leftarrow {\sim}b\}$$

Recovering Semantics

Approximator fixpoints give rise to several semantics.

Proposition

Let P be a normal logic program over $\mathcal{A}$ with approximator $\mathcal{T}_P$, $X \subseteq Y \subseteq \mathcal{A}$.

- X is a supported model of P iff $\mathcal{T}_P(X, X) = (X, X)$.
- (X, Y) is a three-valued supported model of P iff $\mathcal{T}_P(X, Y) = (X, Y)$.
- (X, Y) is the Kripke-Kleene semantics of P iff $(X, Y) = \mathrm{lfp}(\mathcal{T}_P)$.

But what about stable model semantics?

Stable Operators

Stable Operator: Intuition

The Gelfond-Lifschitz Reduct of P ...

- ...starts out with a two-valued interpretation $S \subseteq \mathcal{A}$;
- ...removes all rules requiring some $a \in S$ to be false;
- ...assumes all $a \in \mathcal{A} \setminus S$ to be false in the remaining rules.

Stable Operator: Intuition

The Gelfond-Lifschitz Reduct of P …

- …starts out with a two-valued interpretation $S \subseteq \mathcal{A}$;
- …removes all rules requiring some $a \in S$ to be false;
- …assumes all $a \in \mathcal{A} \setminus S$ to be false in the remaining rules.

- To obtain reduct P^S, assume all and only atoms $a \in \mathcal{A} \setminus S$ to be false.
- Using P^S, try to constructively prove all and only atoms $a \in S$ to be true.
- P^S is a definite logic program, so T_{P^S} is a $\subseteq$-monotone operator on $(2^{\mathcal{A}}, \subseteq)$.

Stable Operator: Intuition

The Gelfond-Lifschitz Reduct of P …

- …starts out with a two-valued interpretation $S \subseteq \mathcal{A}$;
- …removes all rules requiring some $a \in S$ to be false;
- …assumes all $a \in \mathcal{A} \setminus S$ to be false in the remaining rules.

- To obtain reduct P^S, assume all and only atoms $a \in \mathcal{A} \setminus S$ to be false.
- Using P^S, try to constructively prove all and only atoms $a \in S$ to be true.
- P^S is a definite logic program, so T_{P^S} is a $\subseteq$-monotone operator on $(2^{\mathcal{A}}, \subseteq)$.

Expressing the Reduct via an Operator

- For pair (X, Y), an $a \in \mathcal{A}$ is true iff $a \in X$; atom a is false iff $a \notin Y$.
- Use $\mathcal{T}_P{}'$ to reconstruct what is true, fixing the upper bound to S:

$$\mathcal{T}_P{}'(\cdot, S) \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}, \quad X \mapsto \mathcal{T}_P{}'(X, S)$$

Stable Operator: Preparation

Proposition

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.
For every pair $(x, y) \in L^2$, the following operators are $\leqslant$-monotone:

$$\mathcal{A}'(\cdot, y)\colon L \to L, \quad z \mapsto \mathcal{A}'(z, y) \qquad \text{and} \qquad \mathcal{A}''(x, \cdot)\colon L \to L, \quad z \mapsto \mathcal{A}''(x, z)$$

Stable Operator: Preparation

Proposition

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.
For every pair $(x, y) \in L^2$, the following operators are $\leqslant$-monotone:

$$\mathcal{A}'(\cdot, y) \colon L \to L, \quad z \mapsto \mathcal{A}'(z, y) \qquad \text{and} \qquad \mathcal{A}''(x, \cdot) \colon L \to L, \quad z \mapsto \mathcal{A}''(x, z)$$

Proof.

1. Let $x_1 \leqslant x_2$ and $y \in L$.

Stable Operator: Preparation

Proposition

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.
For every pair $(x, y) \in L^2$, the following operators are $\leqslant$-monotone:

$$\mathcal{A}'(\cdot, y) \colon L \to L, \quad z \mapsto \mathcal{A}'(z, y) \qquad \text{and} \qquad \mathcal{A}''(x, \cdot) \colon L \to L, \quad z \mapsto \mathcal{A}''(x, z)$$

Proof.

1. Let $x_1 \leqslant x_2$ and $y \in L$.
   Then $(x_1, y) \leq_i (x_2, y)$ and $\mathcal{A}(x_1, y) \leq_i \mathcal{A}(x_2, y)$, thus $\mathcal{A}'(x_1, y) \leqslant \mathcal{A}'(x_2, y)$.

Stable Operator: Preparation

Proposition

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.
For every pair $(x, y) \in L^2$, the following operators are $\leqslant$-monotone:

$$\mathcal{A}'(\cdot, y)\colon L \to L, \quad z \mapsto \mathcal{A}'(z, y) \qquad \text{and} \qquad \mathcal{A}''(x, \cdot)\colon L \to L, \quad z \mapsto \mathcal{A}''(x, z)$$

Proof.

1. Let $x_1 \leqslant x_2$ and $y \in L$.
   Then $(x_1, y) \leq_i (x_2, y)$ and $\mathcal{A}(x_1, y) \leq_i \mathcal{A}(x_2, y)$, thus $\mathcal{A}'(x_1, y) \leqslant \mathcal{A}'(x_2, y)$.
2. Let $x \in L$ and $y_1 \leqslant y_2$.

Stable Operator: Preparation

Proposition

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.
For every pair $(x, y) \in L^2$, the following operators are $\leqslant$-monotone:

$$\mathcal{A}'(\cdot, y) \colon L \to L, \quad z \mapsto \mathcal{A}'(z, y) \qquad \text{and} \qquad \mathcal{A}''(x, \cdot) \colon L \to L, \quad z \mapsto \mathcal{A}''(x, z)$$

Proof.

1. Let $x_1 \leqslant x_2$ and $y \in L$.
   Then $(x_1, y) \leq_i (x_2, y)$ and $\mathcal{A}(x_1, y) \leq_i \mathcal{A}(x_2, y)$, thus $\mathcal{A}'(x_1, y) \leqslant \mathcal{A}'(x_2, y)$.
2. Let $x \in L$ and $y_1 \leqslant y_2$.
   Then $(x, y_2) \leq_i (x, y_1)$ and $\mathcal{A}(x, y_2) \leq_i \mathcal{A}(x, y_1)$, thus $\mathcal{A}''(x, y_1) \leqslant \mathcal{A}''(x, y_2)$. □

- $\mathcal{A}'(\cdot, y)$ has a $\leqslant$-least fixpoint, denoted $\mathrm{lfp}(\mathcal{A}'(\cdot, y))$;
- $\mathcal{A}''(x, \cdot)$ has a $\leqslant$-least fixpoint, denoted $\mathrm{lfp}(\mathcal{A}''(x, \cdot))$.

Stable Operator: Definition

Definition

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.
The **stable approximator** for $\mathcal{A}$ is given by $\mathcal{S}\mathcal{A} : L^2 \to L^2$ with

$$\mathcal{S}\mathcal{A}' : L^2 \to L, \qquad (x, y) \mapsto \operatorname{lfp}(\mathcal{A}'(\cdot, y))$$
$$\mathcal{S}\mathcal{A}'' : L^2 \to L, \qquad (x, y) \mapsto \operatorname{lfp}(\mathcal{A}''(x, \cdot))$$

- $\mathcal{S}\mathcal{A}'$: improve lower bound for all fixpoints of O at or below upper bound;
- $\mathcal{S}\mathcal{A}''$: obtain tightmost new upper bound (eliminate non-minimal fixpoints).

Proposition

Let (x, y) be a postfixpoint of approximator $\mathcal{A}$. Then

$$a \in [\bot, y] \text{ implies } \mathcal{A}'(a, y) \in [\bot, y] \quad \text{and} \quad b \in [x, \top] \text{ implies } \mathcal{A}''(x, b) \in [x, \top].$$

In particular, $\operatorname{lfp}(\mathcal{A}'(\cdot, y)) \leqslant y$ and $x \leqslant \operatorname{lfp}(\mathcal{A}''(x, \cdot))$.

Stable Operator: Observations

Theorem

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{SA}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{SA}(x, y)$ is consistent.

Proof.

1. Let $(u, v) \leq_i (x, y)$.

Stable Operator: Observations

Theorem

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{S}\mathcal{A}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{S}\mathcal{A}(x, y)$ is consistent.

Proof.

1. Let $(u, v) \leq_i (x, y)$. Now $y \leqslant v$ implies $\mathcal{A}'(z, v) \leqslant \mathcal{A}'(z, y)$ for all $z \in L$ since $\mathcal{A}$ is $\leq_i$-monotone.

Stable Operator: Observations

Theorem

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{SA}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{SA}(x, y)$ is consistent.

Proof.

1. Let $(u, v) \leq_i (x, y)$. Now $y \leqslant v$ implies $\mathcal{A}'(z, v) \leqslant \mathcal{A}'(z, y)$ for all $z \in L$ since $\mathcal{A}$ is $\leq_i$-monotone. In particular, for $z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$, $\mathcal{A}'(z^*, v) \leqslant \mathcal{A}'(z^*, y) = z^*$ whence z^* is a prefixpoint of $\mathcal{A}'(\cdot, v)$.

Stable Operator: Observations

Theorem

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{S}\mathcal{A}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{S}\mathcal{A}(x, y)$ is consistent.

Proof.

1. Let $(u, v) \leq_i (x, y)$. Now $y \leqslant v$ implies $\mathcal{A}'(z, v) \leqslant \mathcal{A}'(z, y)$ for all $z \in L$ since $\mathcal{A}$ is $\leq_i$-monotone. In particular, for $z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$, $\mathcal{A}'(z^*, v) \leqslant \mathcal{A}'(z^*, y) = z^*$ whence z^* is a prefixpoint of $\mathcal{A}'(\cdot, v)$. Thus $\mathrm{lfp}(\mathcal{A}'(\cdot, v)) \leqslant z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$.

Stable Operator: Observations

Theorem

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{SA}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{SA}(x, y)$ is consistent.

Proof.

1. Let $(u, v) \leq_i (x, y)$. Now $y \leqslant v$ implies $\mathcal{A}'(z, v) \leqslant \mathcal{A}'(z, y)$ for all $z \in L$ since $\mathcal{A}$ is $\leq_i$-monotone. In particular, for $z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$, $\mathcal{A}'(z^*, v) \leqslant \mathcal{A}'(z^*, y) = z^*$ whence z^* is a prefixpoint of $\mathcal{A}'(\cdot, v)$. Thus $\mathrm{lfp}(\mathcal{A}'(\cdot, v)) \leqslant z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$. In combination, $\mathcal{SA}'(u, v) = \mathrm{lfp}(\mathcal{A}'(\cdot, v)) \leqslant \mathrm{lfp}(\mathcal{A}'(\cdot, y)) = \mathcal{SA}'(x, y)$.

Stable Operator: Observations

Theorem

Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{SA}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{SA}(x, y)$ is consistent.

Proof.

1. Let $(u, v) \leq_i (x, y)$. Now $y \leqslant v$ implies $\mathcal{A}'(z, v) \leqslant \mathcal{A}'(z, y)$ for all $z \in L$ since $\mathcal{A}$ is $\leq_i$-monotone. In particular, for $z^* = \text{lfp}(\mathcal{A}'(\cdot, y))$, $\mathcal{A}'(z^*, v) \leqslant \mathcal{A}'(z^*, y) = z^*$ whence z^* is a prefixpoint of $\mathcal{A}'(\cdot, v)$. Thus $\text{lfp}(\mathcal{A}'(\cdot, v)) \leqslant z^* = \text{lfp}(\mathcal{A}'(\cdot, y))$. In combination, $\mathcal{SA}'(u, v) = \text{lfp}(\mathcal{A}'(\cdot, v)) \leqslant \text{lfp}(\mathcal{A}'(\cdot, y)) = \mathcal{SA}'(x, y)$. $\mathcal{SA}''$: dual.

Stable Operator: Observations

Theorem

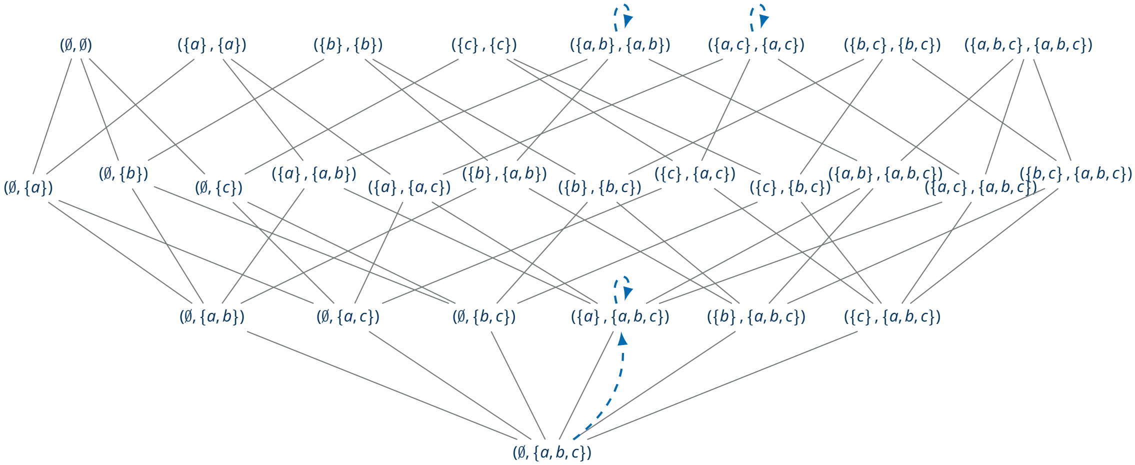
Let $(L, \leqslant)$ be a complete lattice and $\mathcal{A}$ be an approximator on $(L^2, \leq_i)$.

1. $\mathcal{SA}$ is $\leq_i$-monotone.
2. If (x, y) is a consistent postfixpoint of $\mathcal{A}$, then $\mathcal{SA}(x, y)$ is consistent.

Proof.

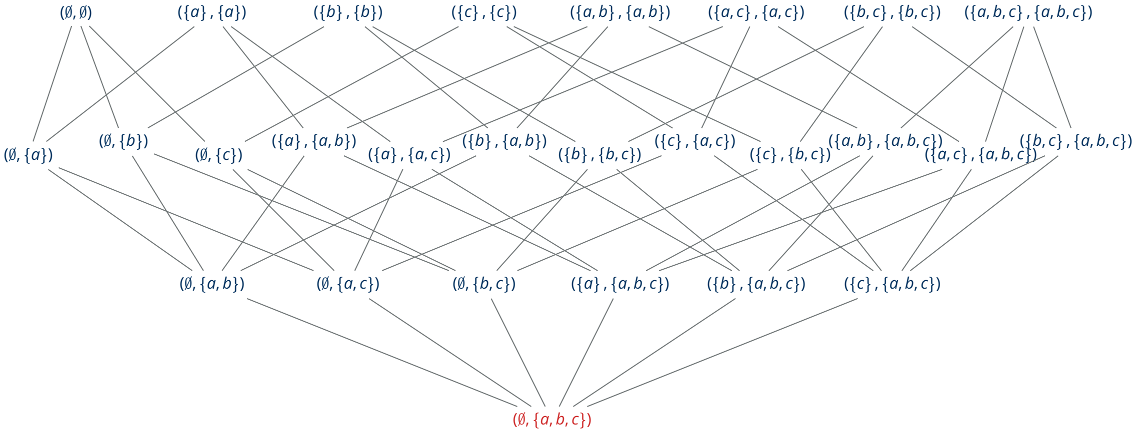
1. Let $(u, v) \leq_i (x, y)$. Now $y \leqslant v$ implies $\mathcal{A}'(z, v) \leqslant \mathcal{A}'(z, y)$ for all $z \in L$ since $\mathcal{A}$ is $\leq_i$-monotone. In particular, for $z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$, $\mathcal{A}'(z^*, v) \leqslant \mathcal{A}'(z^*, y) = z^*$ whence z^* is a prefixpoint of $\mathcal{A}'(\cdot, v)$. Thus $\mathrm{lfp}(\mathcal{A}'(\cdot, v)) \leqslant z^* = \mathrm{lfp}(\mathcal{A}'(\cdot, y))$. In combination, $\mathcal{SA}'(u, v) = \mathrm{lfp}(\mathcal{A}'(\cdot, v)) \leqslant \mathrm{lfp}(\mathcal{A}'(\cdot, y)) = \mathcal{SA}'(x, y)$. $\mathcal{SA}''$: dual.
2. Let $x \leqslant y$ with $(x, y) \leq_i \mathcal{A}(x, y)$. For every $z \in L$ with $x \leqslant z \leqslant y$, we have $\mathcal{SA}'(x, y) \leqslant \mathcal{SA}'(z, z) = \mathrm{lfp}(\mathcal{A}'(\cdot, z)) \leqslant z \leqslant \mathrm{lfp}(\mathcal{A}''(z, \cdot)) = \mathcal{SA}''(z, z) \leqslant \mathcal{SA}''(x, y)$. $\square$

Stable Operator $\mathcal{ST}_P$: Example



$$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$$

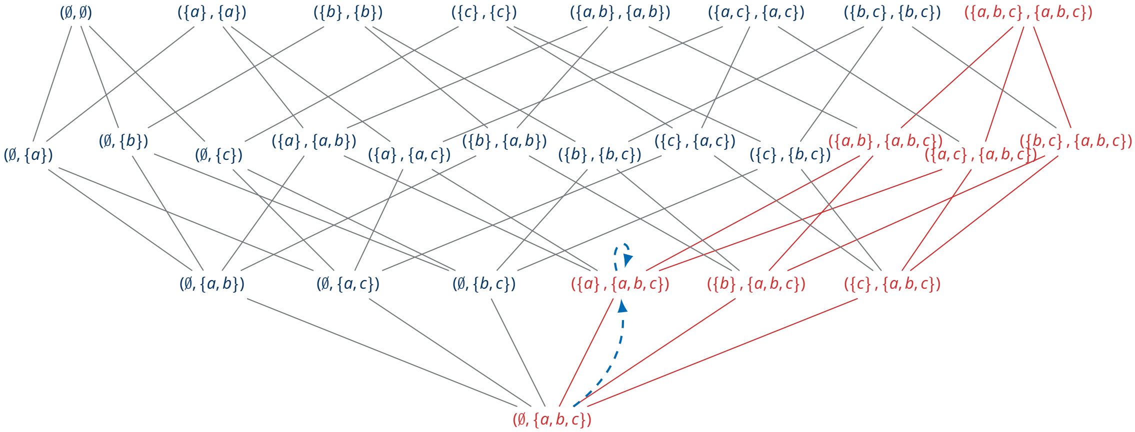
Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$

$\mathcal{ST}_P(\emptyset, \{a, b, c\}) = (\mathrm{lfp}(\mathcal{T}_P{}'(\cdot, \{a, b, c\})), \mathrm{lfp}(\mathcal{T}_P{}''(\emptyset, \cdot)))$

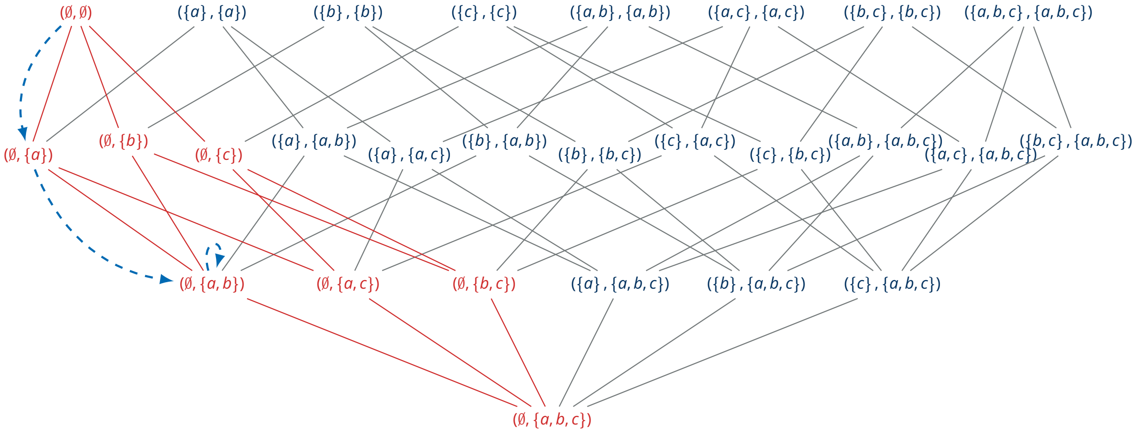
Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$

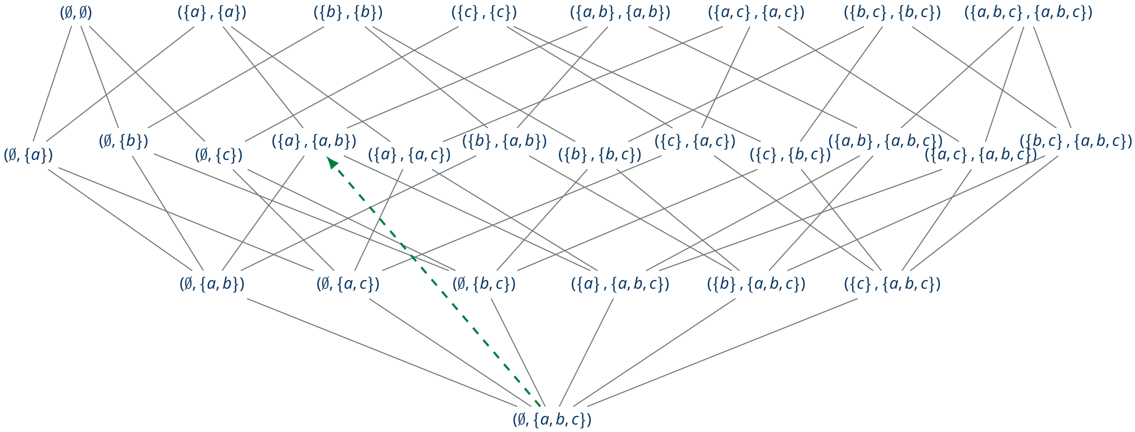
$\mathcal{ST}_P(\emptyset, \{a,b,c\}) = (\mathrm{lfp}(\mathcal{T}_P{}'(\cdot, \{a,b,c\})), \mathrm{lfp}(\mathcal{T}_P{}''(\emptyset, \cdot)))$

Stable Operator $\mathcal{ST}_P$: Example



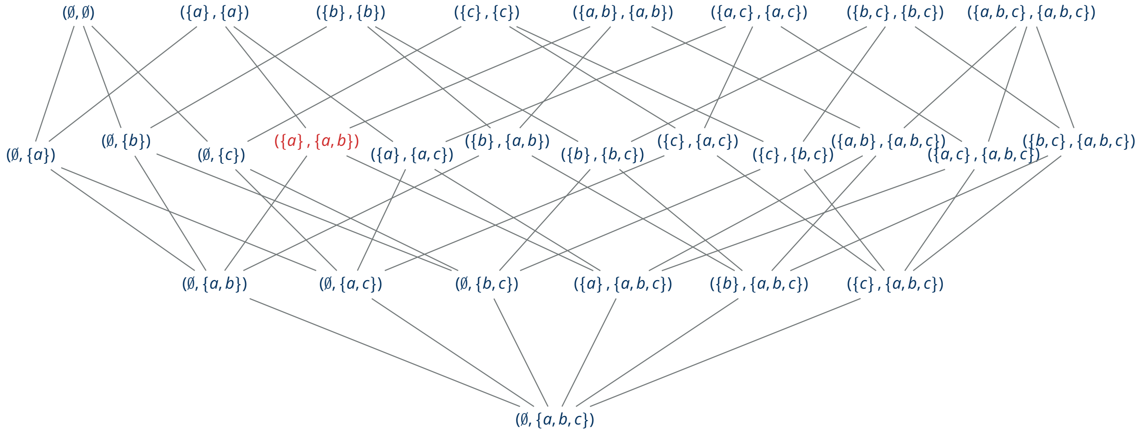
$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$ $\qquad$ $\mathcal{ST}_P(\emptyset, \{a, b, c\}) = (\{a\}, \mathrm{lfp}(\mathcal{T}_P''(\emptyset, \cdot)))$

Stable Operator $\mathcal{ST}_P$: Example



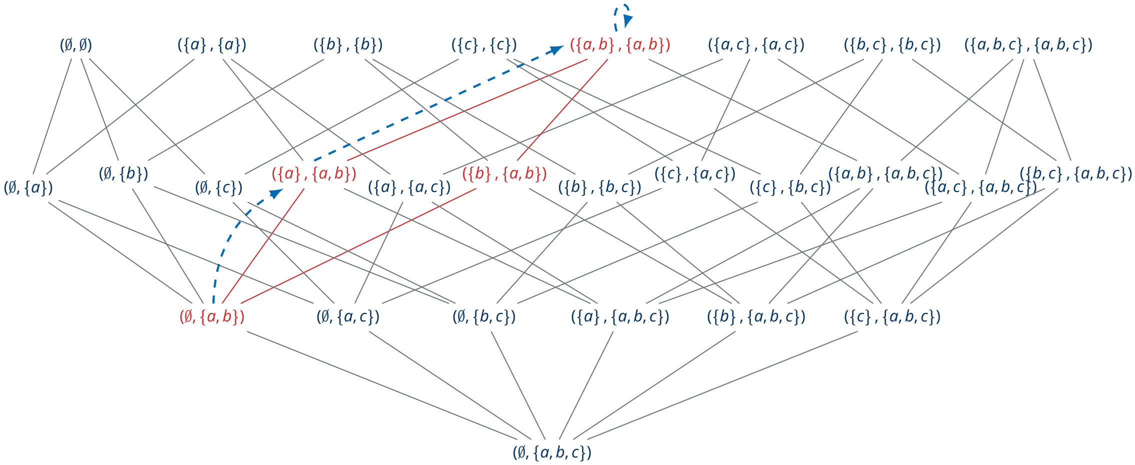
$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$ $\qquad$ $\mathcal{ST}_P(\emptyset, \{a, b, c\}) = (\{a\}, \{a, b\})$

Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$ $\qquad$ $\mathcal{ST}_P(\{a\}, \{a, b\}) = (\text{lfp}(\mathcal{T}_P{}'(\cdot, \{a, b\})), \text{lfp}(\mathcal{T}_P{}''(\{a\}, \cdot)))$

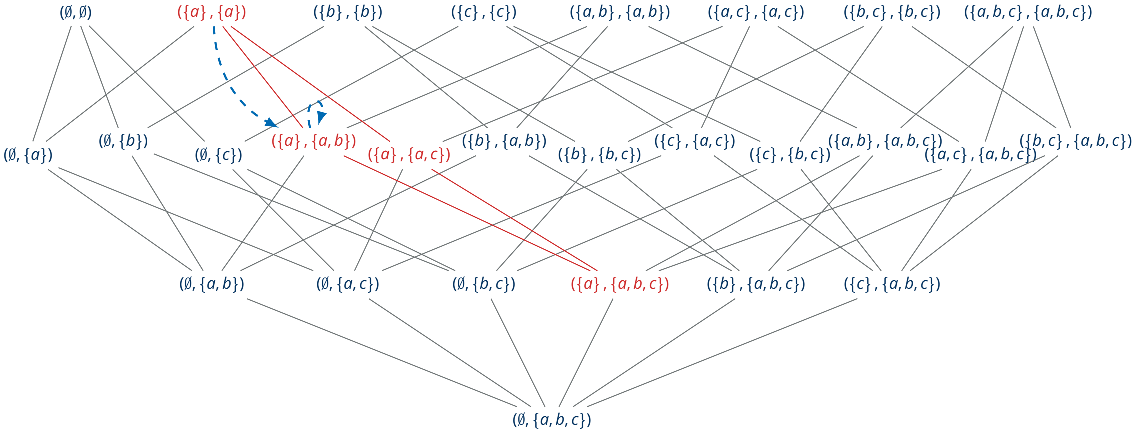
Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$

$\mathcal{ST}_P(\{a\}, \{a, b\}) = (\mathrm{lfp}(\mathcal{T}_P{}'(\cdot, \{a, b\})), \mathrm{lfp}(\mathcal{T}_P{}''(\{a\}, \cdot)))$

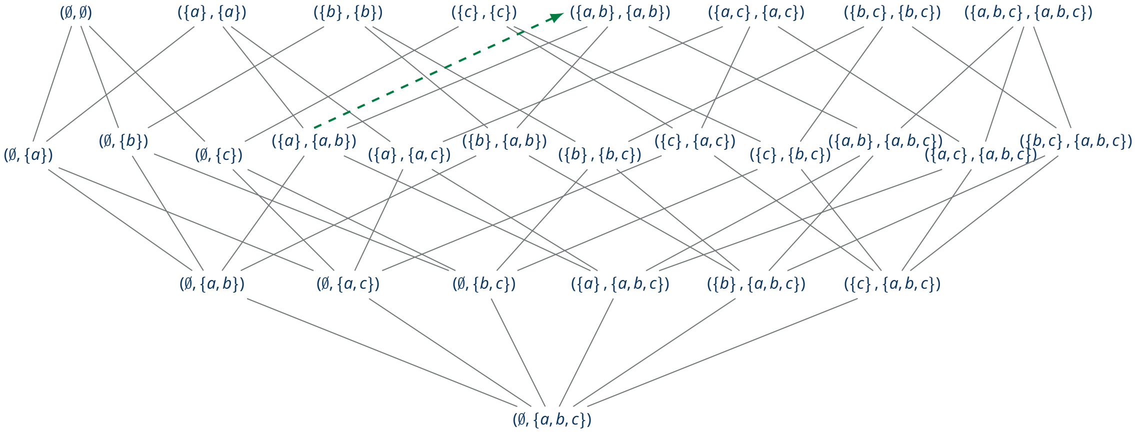
Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$

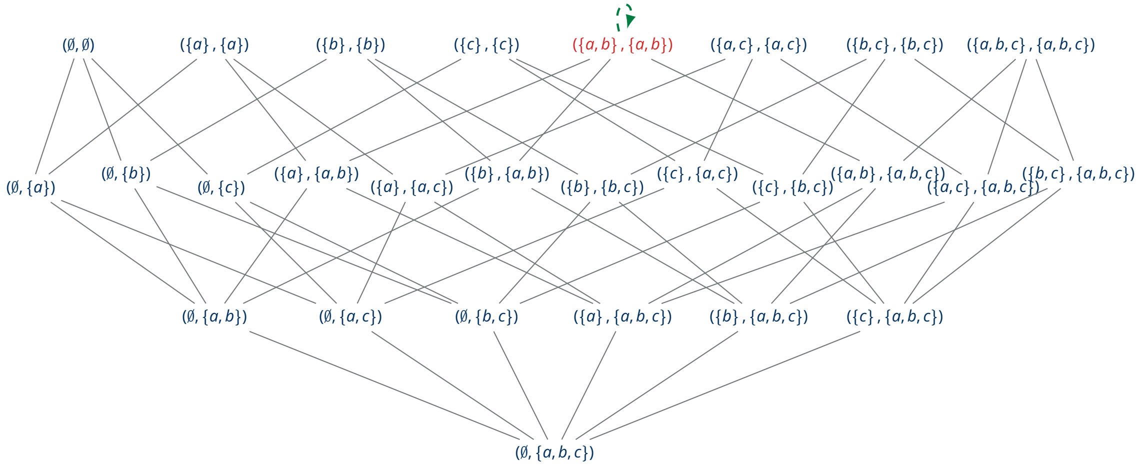
$\mathcal{ST}_P(\{a\}, \{a, b\}) = (\{a, b\}, \mathrm{lfp}(\mathcal{T}_P''(\{a\}, \cdot)))$

Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, {\sim}c, \quad c \leftarrow c\}$ $\qquad$ $\mathcal{ST}_P(\{a\}, \{a,b\}) = (\{a,b\}, \{a,b\})$

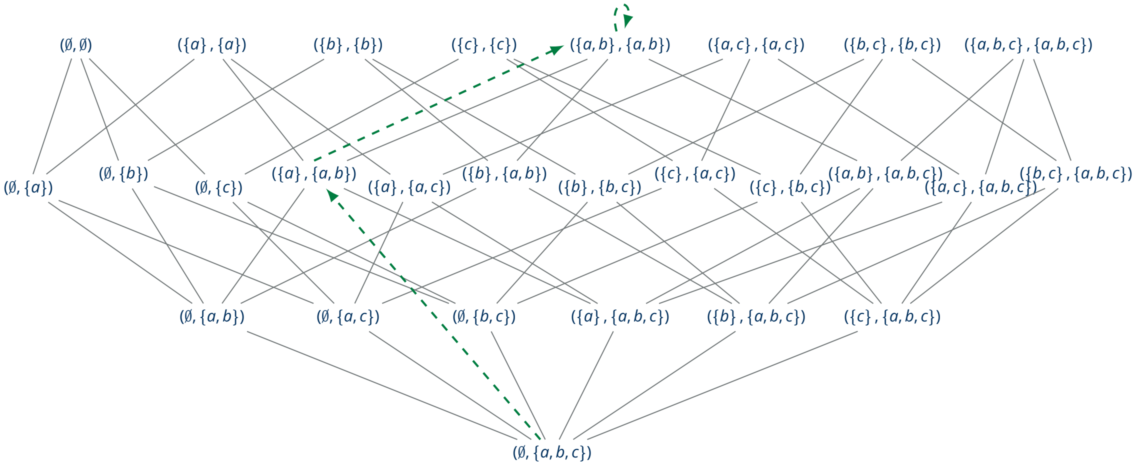
Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$

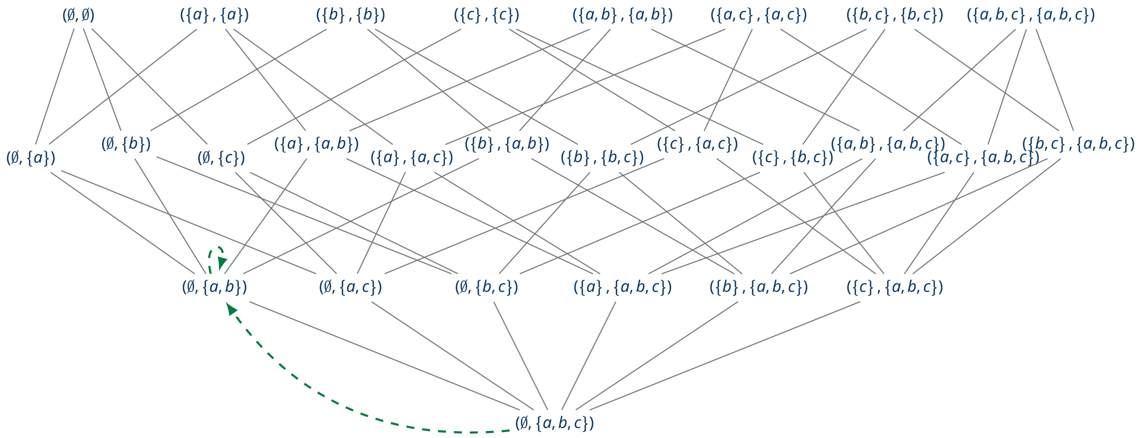
$\mathcal{ST}_P(\{a,b\}, \{a,b\}) = (T_P(\{a,b\}), T_P(\{a,b\})) = (\{a,b\}, \{a,b\})$

Stable Operator $\mathcal{ST}_P$: Example



$P_1 = \{a \leftarrow, \quad b \leftarrow a, \sim c, \quad c \leftarrow c\}$ $\qquad$ $\mathrm{lfp}(\mathcal{ST}_P) = (\{a, b\}, \{a, b\})$: well-founded semantics of P_1

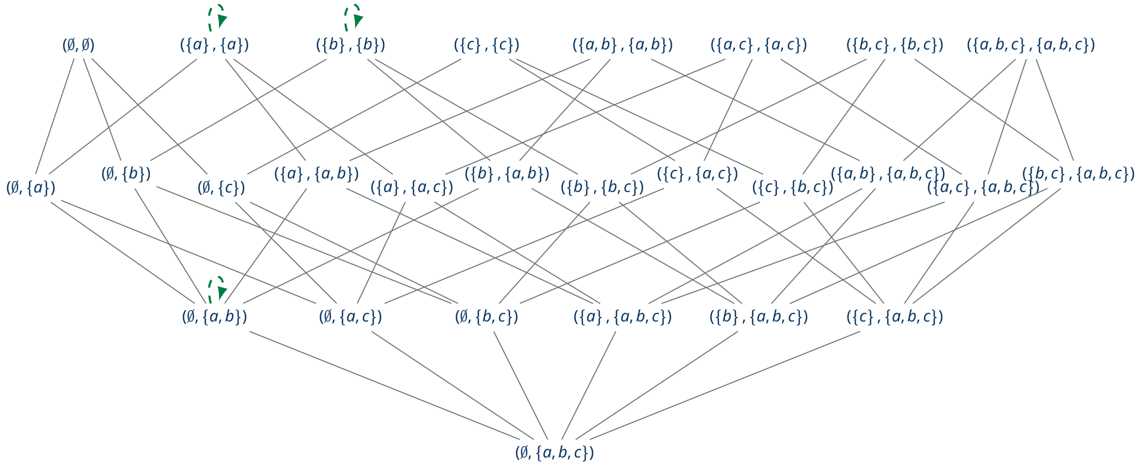
Stable Operator $\mathcal{ST}_P$: Example



$P_2 = \{a \leftarrow {\sim}b, \quad b \leftarrow {\sim}a, \quad c \leftarrow c\}$

$\mathrm{lfp}(\mathcal{ST}_P)$: well-founded semantics of P_2

Stable Operator $\mathcal{ST}_P$: Example



$P_2 = \{a \leftarrow \sim b, \quad b \leftarrow \sim a, \quad c \leftarrow c\}$ three-valued stable models of P_2

Stable Semantics: Definition via Operators

Definition

Let $(L, \leqslant)$ be a complete lattice, $O : L \to L$ be an operator.
Let $\mathcal{A} : L^2 \to L^2$ be an approximator of O in $(L^2, \leq_i)$. A pair $(x, y) \in L^2$ is

- a **two-valued stable model of** $\mathcal{A}$ iff $x = y$ and $\mathcal{SA}(x, y) = (x, y)$;
- a **three-valued stable model of** $\mathcal{A}$ iff $x \leqslant y$ and $\mathcal{SA}(x, y) = (x, y)$;
- the **well-founded model of** $\mathcal{A}$ iff it is the least fixpoint of $\mathcal{SA}$.

Names inspired by notions from logic programming.

Theorem

1. $\mathrm{lfp}(\mathcal{A}) \leq_i \mathrm{lfp}(\mathcal{SA})$;
2. $\mathcal{SA}(x, y) = (x, y)$ implies $\mathcal{A}(x, y) = (x, y)$;
3. if $\mathcal{SA}(x, x) = (x, x)$ then x is a $\leqslant$-minimal fixpoint of O;

Reprise: How to Find an Approximator?

Definition

Let $O : L \to L$ be an operator in a complete lattice $(L, \leqslant)$.
Define the **ultimate approximator of** O as follows:

$$\mathcal{X}_O : L^2 \to L^2, \qquad (x,y) \mapsto \left(\bigwedge \{O(z) \mid x \leqslant z \leqslant y\} , \bigvee \{O(z) \mid x \leqslant z \leqslant y\} \right)$$

Intuition: Consider glb and lub of applying O pointwise to given interval.

Theorem

For every approximator $\mathcal{A}$ of O and consistent pair $(x,y) \in L^2$, we find

$$\mathcal{A}(x,y) \leq_i \mathcal{X}_O(x,y)$$

Ultimate approximator is most precise approximator possible.

Used e.g. for (PSP-)semantics of aggregates in logic programming.

Ultimate Approximator: Example

$P = \{a \leftarrow \sim a, \quad a \leftarrow a\}$

x	$\emptyset$	$\{a\}$
$T_P(x)$	a	a

$\mathcal{X}_{T_P}(\emptyset, \{p\}) = (\{p\}, \{p\})$.

Compare this with $\mathcal{T}_P(\emptyset, \{p\}) = (\emptyset, \{p\})$.

Conclusion

Conclusion

Summary

- Operators in complete lattices can be used to define semantics of KR formalisms.
- Approximation fixpoint theory provides a general account of operator-based semantics.
- Stable approximator reconstructs well-founded and stable model semantics of logic programming.

Outlook

AFT can be used to show correspondence of …

- … extensions of default theories with stable models of logic programs;
- … expansions of autoepistemic theories with supported models of LPs;
- … semantics of argumentation frameworks with semantics of LPs.

Conclusion

Summary

- Operators in complete lattices can be used to define semantics of KR formalisms.

Conclusion

Summary

- Operators in complete lattices can be used to define semantics of KR formalisms.
- Approximation fixpoint theory provides a general account of operator-based semantics.

Conclusion

Summary

- Operators in complete lattices can be used to define semantics of KR formalisms.
- Approximation fixpoint theory provides a general account of operator-based semantics.
- Stable approximator reconstructs well-founded and stable model semantics of logic programming.

Conclusion

Summary

- Operators in complete lattices can be used to define semantics of KR formalisms.
- Approximation fixpoint theory provides a general account of operator-based semantics.
- Stable approximator reconstructs well-founded and stable model semantics of logic programming.
- To define semantics for new formalisms, only an approximator needs to be defined, AFT does the rest.

Conclusion

Summary

- Operators in complete lattices can be used to define semantics of KR formalisms.
- Approximation fixpoint theory provides a general account of operator-based semantics.
- Stable approximator reconstructs well-founded and stable model semantics of logic programming.
- To define semantics for new formalisms, only an approximator needs to be defined, AFT does the rest.
- With ultimate approximation, only a consequence operator needs to be defined.

Outlook

What else can Approximation Fixpoint Theory do for KR?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

AFT could be used to analyse/define/compare semantics of . . .

- . . . epistemic logic programs?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

AFT could be used to analyse/define/compare semantics of ...

- ... epistemic logic programs?
- ... (first-order) conditionals?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

AFT could be used to analyse/define/compare semantics of . . .

- . . . epistemic logic programs?
- . . . (first-order) conditionals?
- . . . (non-monotonic) existential rules?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

AFT could be used to analyse/define/compare semantics of ...

- ... epistemic logic programs?
- ... (first-order) conditionals?
- ... (non-monotonic) existential rules?
- ... description logics with defeasible subsumption?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

AFT could be used to analyse/define/compare semantics of ...

- ... epistemic logic programs?
- ... (first-order) conditionals?
- ... (non-monotonic) existential rules?
- ... description logics with defeasible subsumption?
- ... assumption-based argumentation?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

AFT could be used to analyse/define/compare semantics of ...

- ... epistemic logic programs?
- ... (first-order) conditionals?
- ... (non-monotonic) existential rules?
- ... description logics with defeasible subsumption?
- ... assumption-based argumentation?
- ... non-monotonic causal theories?

Outlook

What else can Approximation Fixpoint Theory do for KR?

Open Topics

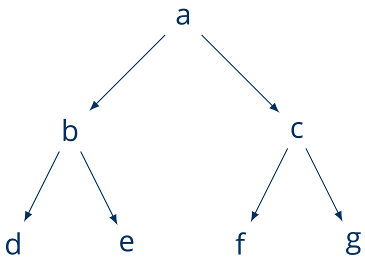
AFT could be used to analyse/define/compare semantics of . . .

- . . . epistemic logic programs?
- . . . (first-order) conditionals?
- . . . (non-monotonic) existential rules?
- . . . description logics with defeasible subsumption?
- . . . assumption-based argumentation?
- . . . non-monotonic causal theories?
- . . . the formalism you are interested in?

Aggregates

Aggregates: Basic Idea Alviano, Faber and Gebser, ‘Aggregate semantics for propositional answer set programs’

```
tree(a).
tree(b). tree(c).
tree(d). tree(e).
tree(f). tree(g).
child(a,b). child(a,c).
child(b,d). child(b,e).
child(c,f). child(c,g).
```




```
children(X,N):- tree(X), #count{Y: child(X,Y)}=N.
```

Choice Atoms

Definition

A **choice atom** is an expression $C = (\mathrm{dom}, \mathrm{sat})$ where $\mathrm{dom} \subseteq \mathcal{A}$ and $\mathrm{sat} \subseteq 2^{\mathrm{dom}}$.
A set of atoms $X \subseteq \mathcal{A}$ satisfies $(\mathrm{dom}, \mathrm{sat})$ if $X \cap \mathrm{dom} \in \mathrm{sat}$.

Example

$\texttt{\#count}\{p, q, r\} > 0$ corresponds to the choice atom
$C_1 = (\{p, q, r\}, \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\})$.

- $\{p, q, s\}$ satisfies C_1 as
  $\{p, q, s\} \cap \{p, q, r\} = \{p, q\} \in \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\}$.
- $\{p, q, r\}$ does not satisfy C_1 as
  $\{p, q, r\} \cap \{p, q, r\} = \{p, q, r\} \notin \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\}$.

Aggregate Programs: Syntax

Definition

A **definite aggregate program** over $\mathcal{A}$ is a set P of rules of the form

$$a_0 \leftarrow a_1, \ldots, a_m$$

for $a_0 \in \mathcal{A}$ and $a_1 \ldots, a_m$ choice atoms (with $0 \leq m$).

Example

```
tree(a). tree(b). tree(c). child(a,b). child(a,c).
children(a,2):- tree(a), #count{b:child(a,b); c:child(a,c)}=2.
```

where `#count{b: child(a,b);c: child(a,c) }=2` is an “abbreviation” for the choice atom:

$$(\{\texttt{child(a,b)},\texttt{child(a,c)}\},\{\{\texttt{child(a,b)},\texttt{child(a,c)}\}\})$$

Extending the T_P-operator

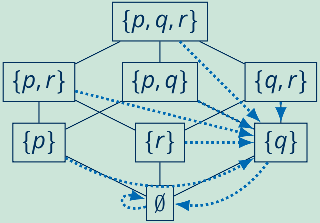
Definition

Let P be a definite aggregate program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,\ \mathrm{dom}(a_i) \cap x \in \mathrm{sat}(a_i) \text{ for every } i = 1 \ldots m\}$$

Example

$$P = \{q \leftarrow \#\mathtt{count}\{p, r\} > 1\}.$$



Aggregates introduce non-monotonicity

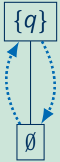
Definition

Let P be a definite aggregate program over atoms $\mathcal{A}$.
The **one-step consequence operator** of P is given by $T_P \colon 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ with

$$S \mapsto \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,\ \mathrm{dom}(a_i) \cap x \in \mathrm{sat}(a_i) \text{ for every } i = 1 \ldots m\}$$

Example

$$P = \{q \leftarrow \#\mathrm{count}\{q\} < 1\}.$$



Semantics for aggregate programs: An arduous task

- Historically, aggregates have a long tradition in database query languages, including Datalog.
- Already in Datalog, aggregates caused trouble, e.g. violating the unique-model property.[3]
- First attempt at stable semantics[4] sanctioned non-minimal stable models.
- Some highlights of subsequent attempts are listed below.[5]

---

[3] Inderpal Singh Mumick, Hamid Pirahesh and Raghu Ramakrishnan. 'The magic of duplicates and aggregates'. In: *Proceedings of the 16th International Conference on Very Large Data Bases*. 1990, pp. 264–277; Kenneth A Ross. 'Modular stratification and magic sets for DATALOG programs with negation'. In: *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1990, pp. 161–171.

[4] David B Kemp and Peter J Stuckey. 'Semantics of Logic Programs with Aggregates.'. In: *ISLP*. vol. 91. Citeseer. 1991, pp. 387–401.

[5] Wolfgang Faber, Gerald Pfeifer and Nicola Leone. 'Semantics and complexity of recursive aggregates in answer set programming'. In: *Artificial Intelligence* 175.1 (2011), pp. 278–298; Nikolay Pelov, Marc Denecker and Maurice Bruynooghe. 'Well-founded and stable semantics of logic programs with aggregates'. In: *Theory and Practice of Logic Programming* 7.3 (2007), pp. 301–353; Lengning Liu, Enrico Pontelli, Tran Cao Son and Miroslaw Truszczyński. 'Logic programs with abstract constraint atoms: The role of computations'. In: *Artificial Intelligence* 174.3-4 (2010), pp. 295–315; Michael Gelfond and Yuanlin Zhang. 'Vicious circle principle and logic programs with aggregates'. In: *Theory and Practice of Logic Programming* 14.4-5 (2014), pp. 587–601.

Ultimate Approximator for T_P

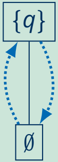
Definition

$\mathcal{X}_{T_P}(x, y) =$

$$\left(\bigcap \{T_P(z) \mid x \subseteq z \subseteq y\}, \bigcup \{T_P(z) \mid x \subseteq z \subseteq y\} \right)$$

Example

$P = \{q \leftarrow \#\mathtt{count}\{q\} < 1\}.$



$$\mathcal{X}_{T_P}(\emptyset, \{q\}) = (\emptyset, \{q\}).$$
$$\mathcal{X}_{T_P}(\{q\}, \{q\}) = (\emptyset, \emptyset).$$
$$\mathcal{X}_{T_P}(\emptyset, \emptyset) = (\{q\}, \{q\}).$$

Ultimate Approximator for T_P

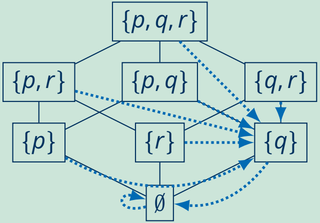
Definition

$\mathcal{X}_{T_P}(x, y) =$

$$\left(\bigcap \{ T_P(z) \mid x \subseteq z \subseteq y \}, \bigcup \{ T_P(z) \mid x \subseteq z \subseteq y \} \right)$$

Example

$P = \{ q \leftarrow \#\mathtt{count}\{p, r\} > 1 \}.$



$\mathcal{X}_{T_P}(\emptyset, \{p, q, r\}) = (\emptyset, \{q\}).$
$\mathcal{X}_{T_P}(\{p\}, \{p, q, r\}) = (\{q\}, \{q\}).$

Other Approximators for Aggregate Programs

Trivial Approximator

$$\mathcal{T}_P^{GZ,l}(x,y) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P, \\ \forall i = 1 \ldots m, \mathrm{dom}(a_i) \cap x \in \mathrm{sat}(a), \text{and} \\ \mathrm{dom}(a_i) \cap x = \mathrm{dom}(a_i) \cap y\}$$

MR-Approximator

$$\mathcal{T}_P^{MR,l}(x,y) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P, \\ \forall i = 1 \ldots m, \exists x' \subseteq x : \mathrm{dom}(a_i) \cap x \in \mathrm{sat}(a_i), \text{and} \\ \forall i = 1 \ldots m \text{ for every } i = 1 \ldots m : \mathrm{dom}(a_i) \cap y \in \mathrm{sat}(a_i)\}$$

$$\mathcal{T}_P^{MR}(x,y) = \left(\mathcal{T}_P^{MR,l}(x,y), \bigcup \{T_P(z) \mid x \subseteq z \subseteq y\}\right).$$

Approximators for Aggregate Programs

Example

$P = \{q \leftarrow \#\mathrm{count}\{p, r\} > 1\}.$

$$\mathcal{T}_P^{GZ} T_P(\{p\}, \{p,q,r\}) = (\emptyset, \{p,q,r\}) \qquad \mathcal{T}_P^{MR} T_P(\{p\}, \{p,q,r\}) = (\emptyset, \{p,q,r\})$$

$$\mathcal{T}_P^{GZ} T_P(\{p\}, \{p\}) = (\{q\}, \{q\}) \qquad \mathcal{T}_P^{MR} T_P(\{p\}, \{p\}) = (\{q\}, \{q\})$$

$$\mathcal{T}_P^{GZ} T_P(\emptyset, \emptyset) = (\emptyset, \emptyset) \qquad \mathcal{T}_P^{MR} T_P(\emptyset, \emptyset) = (\emptyset, \emptyset)$$

Example

$P = \{q \leftarrow \#\mathrm{count}\{q\} < 1\}.$

$$\mathcal{T}_P^{GZ}(\emptyset, \{q\}) = (\emptyset, \{q\}) \qquad \mathcal{T}_P^{MR}(\emptyset, \{q\}) = (\{q\}, \{q\})$$

$$\mathcal{T}_P^{GZ}(\{q\}, \{q\}) = (\emptyset, \emptyset) \qquad \mathcal{T}_P^{MR}(\{q\}, \{q\}) = (\emptyset, \emptyset)$$

$$\mathcal{T}_P^{GZ}(\emptyset, \emptyset) = (\{q\}, \{q\}) \qquad \mathcal{T}_P^{MR}(\emptyset, \emptyset) = (\{q\}, \{q\})$$

MR-Operator is not $\leq_i$-monotone

MR-Approximator

$$\mathcal{T}_P^{\mathrm{MR},l}(x,y) = \{a_0 \in \mathcal{A} \mid a_0 \leftarrow a_1, \ldots, a_m \in P,$$
$$\forall i = 1 \ldots m, \exists x' \subseteq x : \mathrm{dom}(a_i) \cap x \in \mathrm{sat}(a_i), \text{and}$$
$$\forall i = 1 \ldots m \text{ for every } i = 1 \ldots m : \mathrm{dom}(a_i) \cap y \in \mathrm{sat}(a_i)\}$$

$$\mathcal{T}_P^{\mathrm{MR}}(x,y) = \left(\mathcal{T}_P^{\mathrm{MR},l}(x,y), \bigcup \{T_P(z) \mid x \subseteq z \subseteq y\}\right).$$

Example

$P = \{q \leftarrow \#\mathtt{sum}\{1 : p, -1 : q\} \geq 0\}.$
$\mathcal{T}_P^{\mathrm{MR},l}(\emptyset, \{p,q\}) = \{q\}.$
$\mathcal{T}_P^{\mathrm{MR},l}(\{p\}, \{q\}) = \emptyset.$

Yet $\mathcal{T}_P^{\mathrm{MR},l}(\cdot, y)$ is $\subseteq$-monotonic for any $y \subseteq \mathcal{A}$, and thus, we can still use the stable construction.

Comparison between different semantics I

Example

$$P = \begin{Bmatrix} p \leftarrow \#\mathtt{sum}\{1 : p\} > 0. \\ p \leftarrow \#\mathtt{sum}\{1 : p\} < 1. \end{Bmatrix}$$

- $\{p\}$ is a stable fixpoint of $\mathcal{X}_{T_P}$:
  - $\mathcal{X}_{T_P}(\emptyset, \{p\}) = (\{p\}, \{p\})$
    (as $\emptyset$ satisfies $\#\mathtt{sum}\{1 : p\} < 1$, and $\{p\}$ satisfies $\#\mathtt{sum}\{1 : p\} > 0$).
  - $\mathcal{X}_{T_P}(\{p\}, \{p\}) = (\{p\}, \{p\})$ as $\{p\}$ satisfies $\#\mathtt{sum}\{1 : p\} > 0$.
- $\{p\}$ is not a stable fixpoint of $\mathcal{T}_P^{GZ}$ or $\mathcal{T}_P^{MR}$:
  - $\mathcal{T}_P^{MR,l}(\emptyset, \{p\}) = \emptyset$
    (as on the one hand $\emptyset$ does not satisfy $\#\mathtt{sum}\{1 : p\} > 0$ and on the other hand $\{p\}$ does not satisfy $\#\mathtt{sum}\{1 : p\} < 1$). (Likewise, $\mathcal{T}_P^{GZ}(\emptyset, \{p\}) = \emptyset$ as $dom(\#\mathtt{sum}\{1 : p\} > 0) \cap \emptyset \neq dom(\#\mathtt{sum}\{1 : p\} > 0) \cap \{p\}$ and $dom(\#\mathtt{sum}\{1 : p\} < 1) \cap \emptyset \neq dom(\#\mathtt{sum}\{1 : p\} < 10) \cap \{p\}$).

Comparison between different semantics II

Example

$$P = \begin{Bmatrix} b \leftarrow \#\mathtt{count}\{a, b\} > 0. \\ a \leftarrow . \end{Bmatrix}$$

- $\{a, b\}$ is a stable fixpoint of $\mathcal{X}_{T_P}$ and $\mathcal{T}_P^{\mathrm{MR},I}$.
  $\mathcal{T}_P^{\mathrm{MR},I}(\emptyset, \{a, b\}) = \{a\}$ as expected.
  $\mathcal{T}_P^{\mathrm{MR},I}(\{a\}, \{a, b\}) = \{a\}$
  (as $\{a\}$ and $\{a, b\}$ satisfy $\#\mathtt{count}\{a, b\} > 0$).
- $\{a, b\}$ is *not* a stable fixpoint of $\mathcal{T}_P^{\mathrm{GZ}}$.
  $\mathcal{T}_P^{\mathrm{GZ}}(\emptyset, \{a, b\}) = \{a\}$ as expected.
  $\mathcal{T}_P^{\mathrm{GZ}}(\{a\}, \{a, b\}) = \{a\}$
  (as $\mathrm{dom}(\#\mathtt{count}\{a, b\} > 0) \cap \{a\} \neq \mathrm{dom}(\#\mathtt{count}\{a, b\} > 0) \cap \{a, b\}$).

Example due to Alviano, Faber and Gebser, 'Aggregate semantics for propositional answer set programs'.

AFT-based semantics for aggregate programs and their relation with normal logic programs

Definition

Given a normal logic program P, we can rewrite it to a choice program:

- $\pi(a) = (\{a\}, \{\{a\}\})$ for any $a \in \mathcal{A}$,
- $\pi(\sim a) = (\{a\}, \{\emptyset\})$ for any $a \in \mathcal{A}$,

$\pi(P) = \{a_0 \leftarrow \pi(a_1), \ldots, \pi(a_n) \mid a_0 \leftarrow a_1, \ldots, a_n \in P\}$.

Theorem

For any normal logic program P,

1. $T_P = T_{\pi(P)}$,
2. $\mathcal{T}^{GZ}_{\pi(P)} = \mathcal{T}^{MR}_{\pi(P)} = \mathcal{T}_P$,
3. $\mathcal{X}_{T_{\pi(P)}} = \mathcal{X}_{T_P}$.

Operator-Based Semantics for Dialects of Logic Programming

- ∨ Aggregates in the body: $p \leftarrow \#\texttt{sum}\{2 : p; q : 1; r : 1\} \geq 2.$
- ∨ Propositional formulas in the body: $p \leftarrow q \wedge (r \vee (s \wedge \neg t)).$
- ∨ Disjunctions in the head: $p \vee q \leftarrow q \wedge (r \vee (s \wedge \neg t)).$
- ∨ Choice constructs in the head: $\#\texttt{count}\{p; q; r\} = 2 \leftarrow \neg r.$
- ∨ DL-based logic programs: $\mathbf{K}C(x) \leftarrow \neg p(X); C \sqsubseteq D.$
- ∨ Higher-order logic programs: $S(P, Q) \leftarrow; P(X) \leftarrow \neg Q(X).$
- ? Fuzzy logic programs: $p(X) \leftarrow 0.5 \cdot (q(x) + r(X)).$
- ? Probabilistic logic programs: $0.3 :: p(X).$
- ? Hex-programs: $tr(S, P, O) \leftarrow \&RDF[uri](S, P, O).$

Argumentation

Abstract Argumentation Frameworks

We assume some background reservoir of (abstract) arguments.

Definition (Dung, 1995)

An **argumentation framework** is a pair $F = (A, R)$ with $R \subseteq A \times A$.

A pair $(a, b) \in R$ expresses that a attacks b.

Abstract Argumentation Frameworks

We assume some background reservoir of (abstract) arguments.

Definition (Dung, 1995)

An **argumentation framework** is a pair $F = (A, R)$ with $R \subseteq A \times A$.

A pair $(a, b) \in R$ expresses that a attacks b.

Definition (Dung, 1995)

For an AF $F = (A, R)$, its **characteristic operator** is given by

$$\Gamma_F : 2^A \to 2^A, \qquad S \mapsto \{a \in A \mid S \text{ defends } a\}$$

S **defends** a iff S attacks all attackers of a.

Abstract Argumentation Frameworks

We assume some background reservoir of (abstract) arguments.

Definition (Dung, 1995)

An **argumentation framework** is a pair $F = (A, R)$ with $R \subseteq A \times A$.

A pair $(a, b) \in R$ expresses that a attacks b.

Definition (Dung, 1995)

For an AF $F = (A, R)$, its **characteristic operator** is given by

$$\Gamma_F : 2^A \to 2^A, \qquad S \mapsto \{a \in A \mid S \text{ defends } a\}$$

S **defends** a iff S attacks all attackers of a.

Example

In $F_1 = a \longrightarrow b$, we have $\Gamma_{F_1}(\emptyset) = \{a\}$ and $\Gamma_{F_1}(\{a\}) = \{a\}$.

Semantics via Operators

Observation

- For any AF F, the operator Γ_F is monotone in the complete lattice $(2^A, \subseteq)$.
- Therefore, Γ_F always has a least fixpoint.

Semantics via Operators

Observation

- For any AF F, the operator Γ_F is monotone in the complete lattice $(2^A, \subseteq)$.
- Therefore, Γ_F always has a least fixpoint.

Proposition

Let F be an argumentation framework.

- The $\subseteq$-least fixpoint of Γ_F corresponds to the grounded extension of F.

Semantics via Operators

Observation

- For any AF F, the operator Γ_F is monotone in the complete lattice $\left(2^A, \subseteq\right)$.
- Therefore, Γ_F always has a least fixpoint.

Proposition

Let F be an argumentation framework.

- The $\subseteq$-least fixpoint of Γ_F corresponds to the grounded extension of F.
- The conflict-free fixpoints of Γ_F correspond to complete extensions of F.

Semantics via Operators

Observation

- For any AF F, the operator Γ_F is monotone in the complete lattice $\left(2^A, \subseteq\right)$.
- Therefore, Γ_F always has a least fixpoint.

Proposition

Let F be an argumentation framework.

- The $\subseteq$-least fixpoint of Γ_F corresponds to the grounded extension of F.
- The conflict-free fixpoints of Γ_F correspond to complete extensions of F.

Open Questions

- Can other semantics also be recast in terms of operators?

Semantics via Operators

Observation

- For any AF F, the operator Γ_F is monotone in the complete lattice $\left(2^A, \subseteq\right)$.
- Therefore, Γ_F always has a least fixpoint.

Proposition

Let F be an argumentation framework.

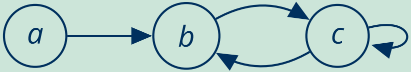
- The $\subseteq$-least fixpoint of Γ_F corresponds to the grounded extension of F.
- The conflict-free fixpoints of Γ_F correspond to complete extensions of F.

Open Questions

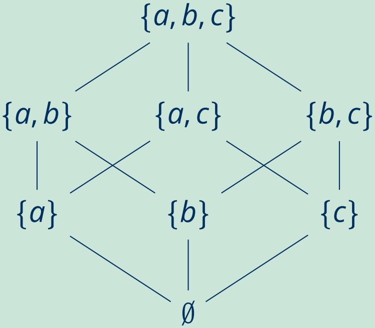
- Can other semantics also be recast in terms of operators?
- Can the extra condition of conflict-freeness be eliminated?

Characteristic Operator: Example

Example

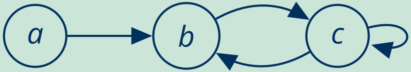
Consider $A = \{a, b, c\}$ and the AF $F_2 =$  .

The operator Γ_{F_2} maps as follows:

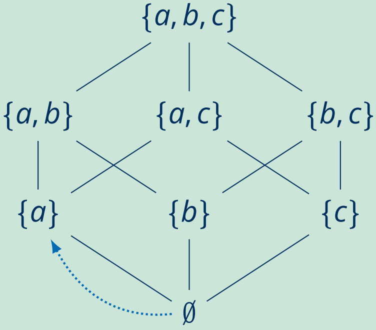


Characteristic Operator: Example

Example

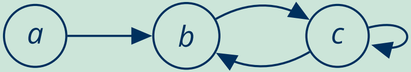
Consider $A = \{a, b, c\}$ and the AF $F_2 =$  .

The operator Γ_{F_2} maps as follows:



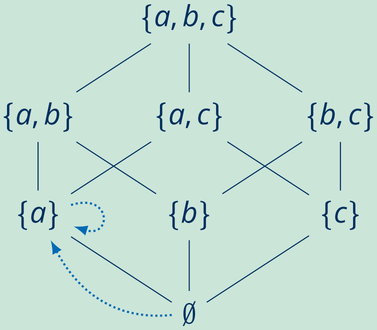
Characteristic Operator: Example

Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$ 

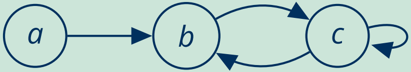
.

The operator Γ_{F_2} maps as follows:



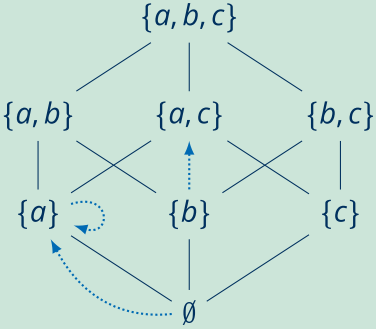
Characteristic Operator: Example

Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$ 

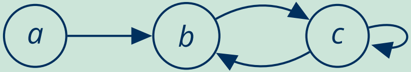
.

The operator Γ_{F_2} maps as follows:



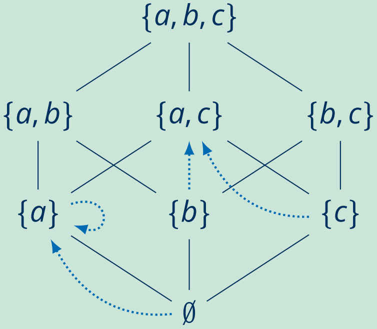
Characteristic Operator: Example

Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$ 

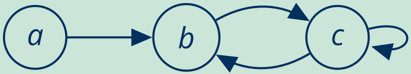
.

The operator Γ_{F_2} maps as follows:



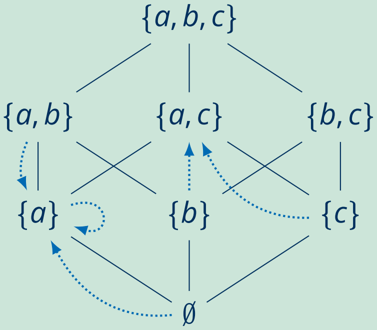
Characteristic Operator: Example

Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$ 

.

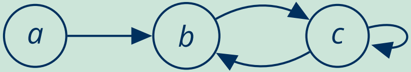
The operator Γ_{F_2} maps as follows:



Characteristic Operator: Example

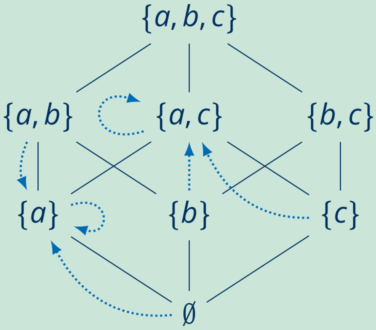
Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$



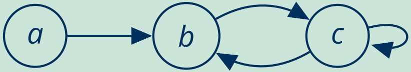
.

The operator Γ_{F_2} maps as follows:



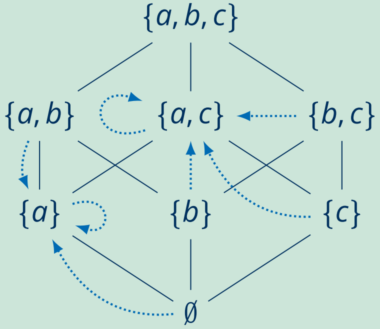
Characteristic Operator: Example

Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$ 

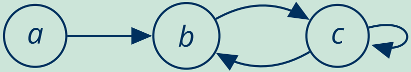
.

The operator Γ_{F_2} maps as follows:

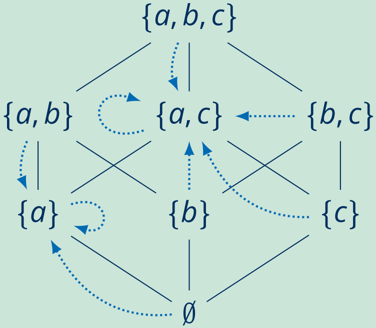


Characteristic Operator: Example

Example

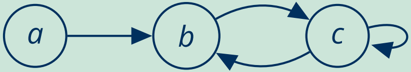
Consider $A = \{a, b, c\}$ and the AF $F_2 =$  .

The operator Γ_{F_2} maps as follows:

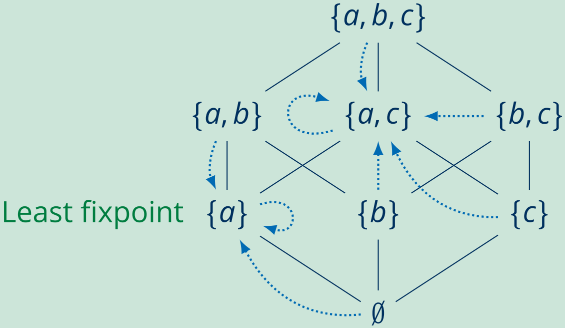


Characteristic Operator: Example

Example

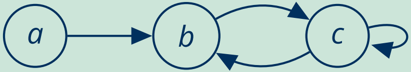
Consider $A = \{a, b, c\}$ and the AF $F_2 =$  .

The operator Γ_{F_2} maps as follows:

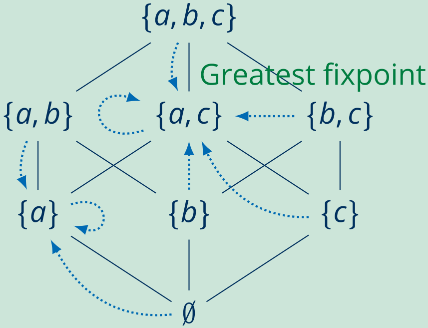


Characteristic Operator: Example

Example

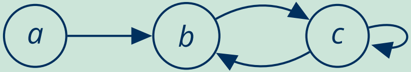
Consider $A = \{a, b, c\}$ and the AF $F_2 =$  .

The operator Γ_{F_2} maps as follows:



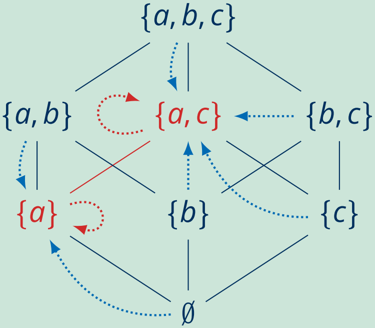
Characteristic Operator: Example

Example

Consider $A = \{a, b, c\}$ and the AF $F_2 =$ 

.

The operator Γ_{F_2} maps as follows:



Complete lattice of fixpoints

Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F \colon 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

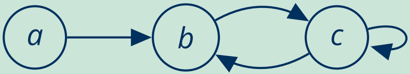
Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

Example

$F_2 =$ 

:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$		$\{a, b\}$	
$\{a\}$		$\{a, c\}$	
$\{b\}$		$\{b, c\}$	
$\{c\}$		$\{a, b, c\}$	

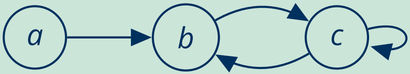
Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \text{ with } R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\} .$$

Example

$F_2 =$ 

:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$		$\{a, b\}$	
$\{a\}$		$\{a, c\}$	
$\{b\}$		$\{b, c\}$	
$\{c\}$		$\{a, b, c\}$	

Pollock's Operator

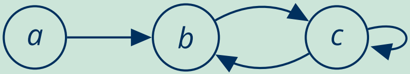
Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}.$$

Example

$F_2 =$



:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	
$\{a\}$		$\{a, c\}$	
$\{b\}$		$\{b, c\}$	
$\{c\}$		$\{a, b, c\}$	

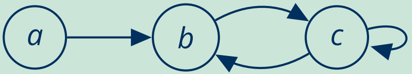
Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}.$$

Example

$F_2 =$  :

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	
$\{a\}$	$\{a, c\}$	$\{a, c\}$	
$\{b\}$		$\{b, c\}$	
$\{c\}$		$\{a, b, c\}$	

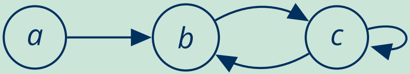
Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

Example

$F_2 =$ 

:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	
$\{a\}$	$\{a, c\}$	$\{a, c\}$	
$\{b\}$	$\{a, b\}$	$\{b, c\}$	
$\{c\}$		$\{a, b, c\}$	

Pollock’s Operator

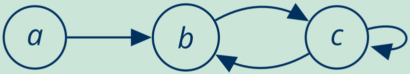
Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F\colon 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \ \text{ with } \ R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

Example

$F_2 =$



:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	
$\{a\}$	$\{a, c\}$	$\{a, c\}$	
$\{b\}$	$\{a, b\}$	$\{b, c\}$	
$\{c\}$	$\{a\}$	$\{a, b, c\}$	

Pollock's Operator

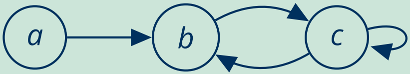
Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

Example

$F_2 =$



:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	$\{a\}$
$\{a\}$	$\{a, c\}$	$\{a, c\}$	
$\{b\}$	$\{a, b\}$	$\{b, c\}$	
$\{c\}$	$\{a\}$	$\{a, b, c\}$	

Pollock's Operator

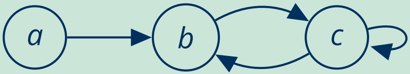
Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \ \text{ with } \ R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\} .$$

Example

$F_2 =$



:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	$\{a\}$
$\{a\}$	$\{a, c\}$	$\{a, c\}$	$\{a\}$
$\{b\}$	$\{a, b\}$	$\{b, c\}$	
$\{c\}$	$\{a\}$	$\{a, b, c\}$	

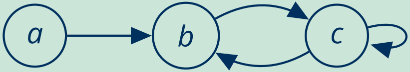
Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F \colon 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\text{ with }\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

Example

$F_2 =$ 

:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	$\{a\}$
$\{a\}$	$\{a, c\}$	$\{a, c\}$	$\{a\}$
$\{b\}$	$\{a, b\}$	$\{b, c\}$	$\{a\}$
$\{c\}$	$\{a\}$	$\{a, b, c\}$	

Pollock's Operator

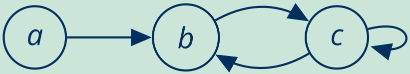
Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F \colon 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \;\; \text{with} \;\; R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}\,.$$

Example

$F_2 =$



:

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	$\{a\}$
$\{a\}$	$\{a, c\}$	$\{a, c\}$	$\{a\}$
$\{b\}$	$\{a, b\}$	$\{b, c\}$	$\{a\}$
$\{c\}$	$\{a\}$	$\{a, b, c\}$	$\{a\}$

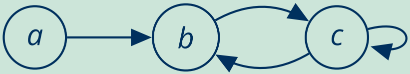
Pollock's Operator

Definition (Pollock, 1987)

For an AF $F = (A, R)$, its **unattacked operator** is given by

$$U_F : 2^A \to 2^A, \quad S \mapsto A \setminus R(S) \text{ with } R(S) := \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}.$$

Example

$F_2 =$  :

S	$U_{F_2}(S)$	S	$U_{F_2}(S)$
$\emptyset$	$\{a, b, c\}$	$\{a, b\}$	$\{a\}$
$\{a\}$	$\{a, c\}$	$\{a, c\}$	$\{a\}$
$\{b\}$	$\{a, b\}$	$\{b, c\}$	$\{a\}$
$\{c\}$	$\{a\}$	$\{a, b, c\}$	$\{a\}$

Proposition

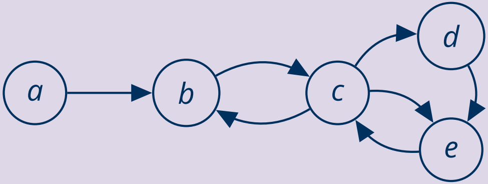
For any AF $F = (A, R)$ and $S, T \subseteq A$, we have: $\quad S \subseteq T \implies U_F(T) \subseteq U_F(S)$.

Quiz: `https://tud.link/jamqpw`

Recall: $U_F(S) = A \setminus \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}$.

Quiz

Consider the argumentation framework $F_3 = (A, R)$:



Which of the following propositions are true?

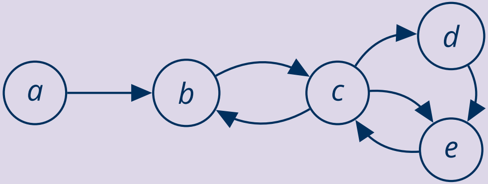
1. $U_F(A) = \{a\}$
2. $U_F(\{c, d, e\}) = \{c, d, e\}$
3. $U_F(\{a, c\}) = \{a, c\}$
4. $U_F(U_F(\{a\})) = \{a, c\}$

Quiz: `https://tud.link/jamqpw`

Recall: $U_F(S) = A \setminus \{a \in A \mid (b,a) \in R \text{ for some } b \in S\}$.

Quiz

Consider the argumentation framework $F_3 = (A, R)$:



Which of the following propositions are true?

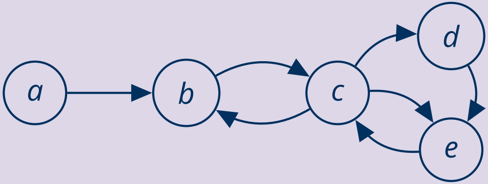
1. $U_F(A) = \{a\}$ ✓
2. $U_F(\{c,d,e\}) = \{c,d,e\}$
3. $U_F(\{a,c\}) = \{a,c\}$
4. $U_F(U_F(\{a\})) = \{a,c\}$

Quiz: `https://tud.link/jamqpw`

Recall: $U_F(S) = A \setminus \{a \in A \mid (b, a) \in R \text{ for some } b \in S\}$.

Quiz

Consider the argumentation framework $F_3 = (A, R)$:



Which of the following propositions are true?

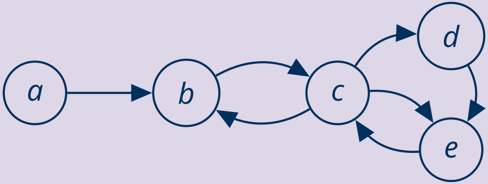
1. $U_F(A) = \{a\}$ ✓
2. $U_F(\{c, d, e\}) = \{c, d, e\}$ ✗
3. $U_F(\{a, c\}) = \{a, c\}$
4. $U_F(U_F(\{a\})) = \{a, c\}$

Quiz: `https://tud.link/jamqpw`

Recall: $U_F(S) = A \setminus \{a \in A \mid (b,a) \in R \text{ for some } b \in S\}$.

Quiz

Consider the argumentation framework $F_3 = (A, R)$:



Which of the following propositions are true?

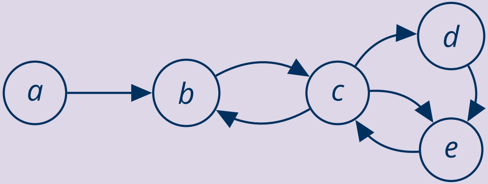
1. $U_F(A) = \{a\}$ ✓
2. $U_F(\{c,d,e\}) = \{c,d,e\}$ ✗
3. $U_F(\{a,c\}) = \{a,c\}$ ✓
4. $U_F(U_F(\{a\})) = \{a,c\}$

Quiz: `https://tud.link/jamqpw`

Recall: $U_F(S) = A \setminus \{a \in A \mid (b,a) \in R \text{ for some } b \in S\}$.

Quiz

Consider the argumentation framework $F_3 = (A, R)$:



Which of the following propositions are true?

1. $U_F(A) = \{a\}$ ✓
2. $U_F(\{c,d,e\}) = \{c,d,e\}$ ✗
3. $U_F(\{a,c\}) = \{a,c\}$ ✓
4. $U_F(U_F(\{a\})) = \{a,c\}$ ✗

Pollock's Operator: Properties

Lemma 45 (Dung, 1995)

For any argumentation framework $F = (A, R)$ and $S \subseteq A$, $\Gamma_F(S) = U_F(U_F(S))$.

Pollock's Operator: Properties

Lemma 45 (Dung, 1995)

For any argumentation framework $F = (A, R)$ and $S \subseteq A$, $\Gamma_F(S) = U_F(U_F(S))$.

Proof.

$$
\begin{aligned}
a \notin \Gamma_F(S) &\iff \text{there is a } b \in U_F(S) \text{ with } (b, a) \in R \\
&\iff a \in R(U_F(S)) \\
&\iff a \notin A \setminus R(U_F(S)) \\
&\iff a \notin U_F(U_F(S))
\end{aligned}
$$

$\square$

Pollock's Operator: Properties

Lemma 45 (Dung, 1995)

For any argumentation framework $F = (A, R)$ and $S \subseteq A$, $\Gamma_F(S) = U_F(U_F(S))$.

Proof.

$$\begin{aligned} a \notin \Gamma_F(S) &\iff \text{there is a } b \in U_F(S) \text{ with } (b, a) \in R \\ &\iff a \in R(U_F(S)) \\ &\iff a \notin A \setminus R(U_F(S)) \\ &\iff a \notin U_F(U_F(S)) \end{aligned}$$

□

Proposition

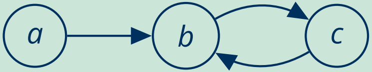
For any AF $F = (A, R)$ and $S \subseteq A$,

$$S \text{ is conflict-free} \iff S \subseteq U_F(S)$$

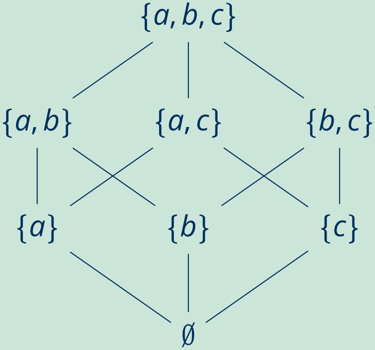
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



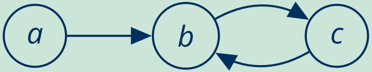
Operator U_F visualised by



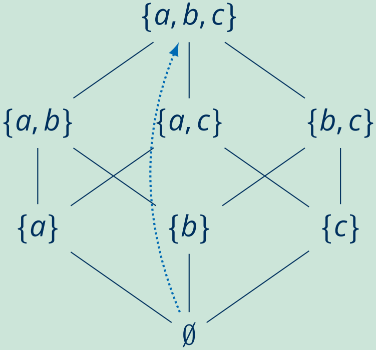
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



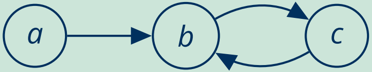
Operator U_F visualised by ⋯⋯▸



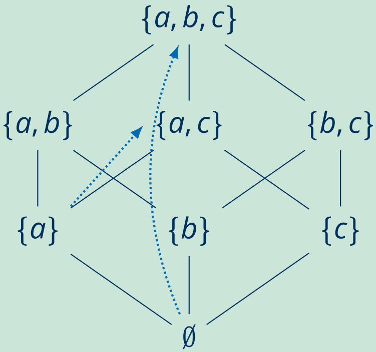
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



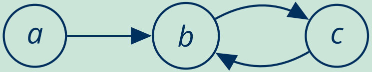
Operator U_F visualised by



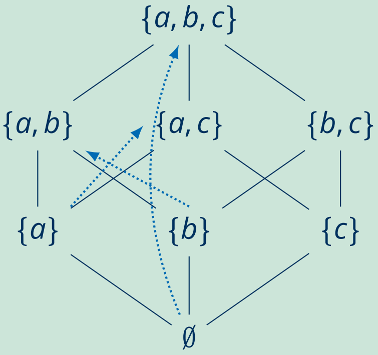
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



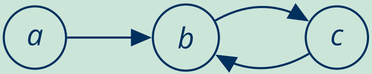
Operator U_F visualised by



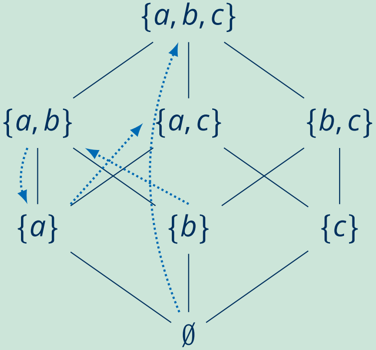
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



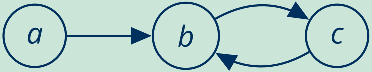
Operator U_F visualised by



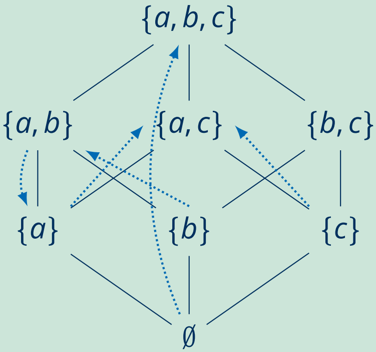
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



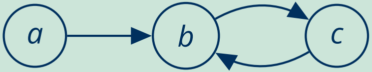
Operator U_F visualised by



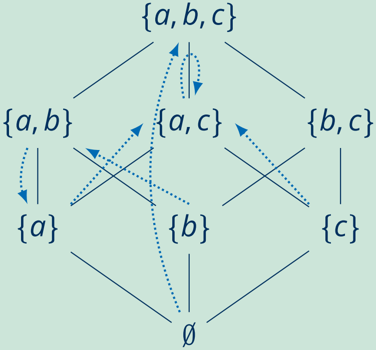
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



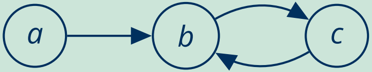
Operator U_F visualised by



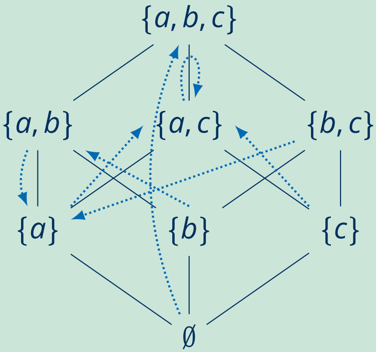
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



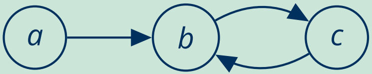
Operator U_F visualised by



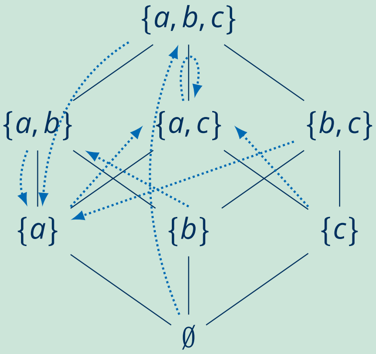
Pollock's Operator: Example

Example

Consider $F_4 = (A, R)$:



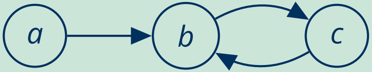
Operator U_F visualised by



Pollock's Operator: Example

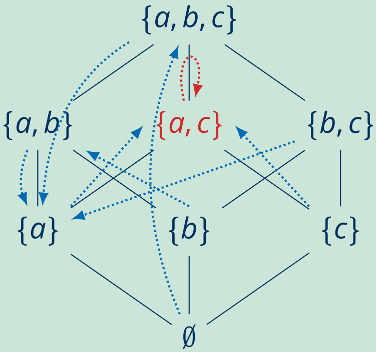
Example

Consider $F_4 = (A, R)$:



Operator U_F visualised by

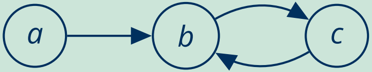
U_F has a fixpoint:
The stable extension of F_4.



Pollock's Operator: Example

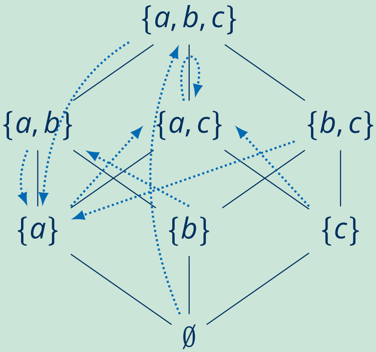
Example

Consider $F_4 = (A, R)$:



Operator U_F visualised by

U_F has a fixpoint:
The stable extension of F_4.

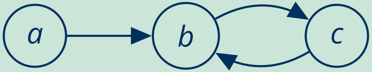


- Does the correspondence fixpoints/stable extensions generalise?

Pollock's Operator: Example

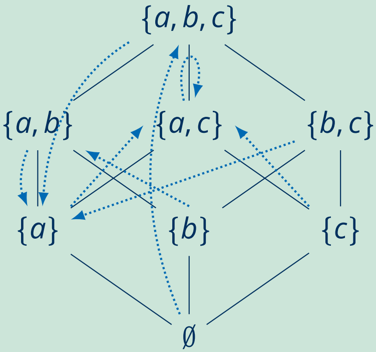
Example

Consider $F_4 = (A, R)$:



Operator U_F visualised by

U_F has a fixpoint:
The stable extension of F_4.



- Does the correspondence fixpoints/stable extensions generalise?
- How to capture more semantics?

Characterising Semantics via Operators

Theorem

Let $F = (A, R)$ be an argumentation framework. A set $S \subseteq A$ is ...

1. conflict-free iff $S \subseteq U_F(S)$;
2. admissible iff $S \subseteq U_F(S)$ and $S \subseteq \Gamma_F(S)$;
3. complete iff $S \subseteq U_F(S)$ and $S = \Gamma_F(S)$;
4. stable iff $S = U_F(S)$;
5. grounded iff it is the least fixpoint of Γ_F.

Characterising Semantics via Operators

Theorem

Let $F = (A, R)$ be an argumentation framework. A set $S \subseteq A$ is …

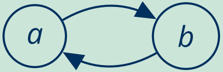
1. conflict-free iff $S \subseteq U_F(S)$;
2. admissible iff $S \subseteq U_F(S)$ and $S \subseteq \Gamma_F(S)$;
3. complete iff $S \subseteq U_F(S)$ and $S = \Gamma_F(S)$;
4. stable iff $S = U_F(S)$;
5. grounded iff it is the least fixpoint of Γ_F.

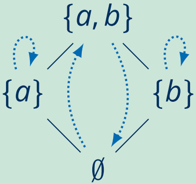
Proof.

4. S is stable iff S is conflict-free and S attacks all arguments in $A \setminus S$
iff S is conflict-free and $R(S) \supseteq A \setminus S$
iff $S \subseteq U_F(S)$ and $A \setminus R(S) \subseteq A \setminus (A \setminus S)$
iff $S \subseteq U_F(S)$ and $U_F(S) \subseteq S$ $\square$

Why Is This Not Enough?

Example

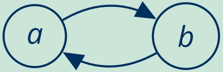
- $F_5 =$ 

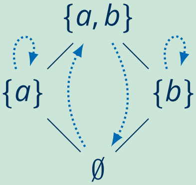
$\rightsquigarrow$ 

(no least fixpoint)

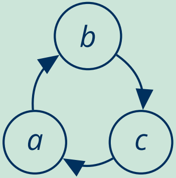
Why Is This Not Enough?

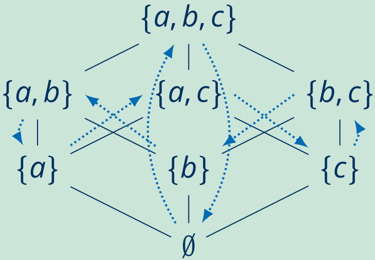
Example

- $F_5 =$ 

$\rightsquigarrow$ 

(no least fixpoint)

- $F_6 =$ 

$\rightsquigarrow$ 

(no fixpoint at all)

Stocktaking

- Monotone operators in complete lattices have (least and greatest) fixpoints.
- Operators can be associated with knowledge bases such that their fixpoints correspond to models.
- An AF F induces its characteristic operator Γ_F, whose least fixpoint is exactly the grounded extension of F.
- An AF F also induces its unattacked operator U_F, which characterises conflict-freeness and stable semantics.
- The unattacked operator U_F can emulate the characteristic operator Γ_F.
- Can semantics be formulated only in terms of U_F, and in a more uniform manner?

Canonical approximator for argumentation frameworks

Example

An argumentation framework $F = (A, R)$ induces U_F with $U_F(S) = A \setminus R(S)$.
The **canonical approximator** of U_F is

$$\mathcal{U}_F \colon 2^A \times 2^A \to 2^A \times 2^A, \qquad (X, Y) \mapsto (U_F(Y), U_F(X))$$

In other words, $\mathcal{U}_F$ is symmetric with $\mathcal{A}'(X, Y) = U_F(Y)$.

Canonical approximator for argumentation frameworks

Example

An argumentation framework $F = (A, R)$ induces U_F with $U_F(S) = A \setminus R(S)$.
The **canonical approximator** of U_F is

$$\mathcal{U}_F \colon 2^A \times 2^A \to 2^A \times 2^A, \qquad (X, Y) \mapsto (U_F(Y), U_F(X))$$

In other words, $\mathcal{U}_F$ is symmetric with $\mathcal{A}'(X, Y) = U_F(Y)$.

- $\mathcal{U}_F$ approximates U_F, as $\mathcal{U}_F(X, X) = (U_F(X), U_F(X))$.

Canonical approximator for argumentation frameworks

Example

An argumentation framework $F = (A, R)$ induces U_F with $U_F(S) = A \setminus R(S)$.
The **canonical approximator** of U_F is

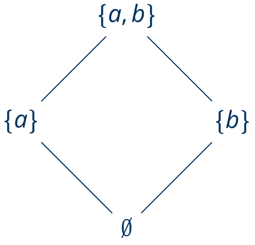
$$\mathcal{U}_F \colon 2^A \times 2^A \to 2^A \times 2^A, \qquad (X, Y) \mapsto (U_F(Y), U_F(X))$$

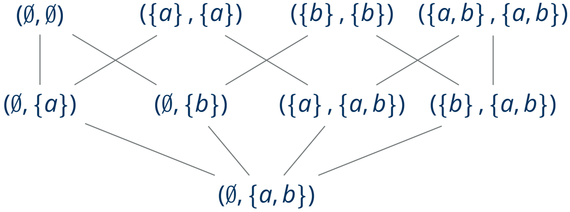
In other words, $\mathcal{U}_F$ is symmetric with $\mathcal{A}'(X, Y) = U_F(Y)$.

- $\mathcal{U}_F$ approximates U_F, as $\mathcal{U}_F(X, X) = (U_F(X), U_F(X))$.
- $\mathcal{U}_F$ is $\leq_i$-monotone:

$$\begin{aligned}
(X_1, Y_1) \leq_i (X_2, Y_2) &\iff X_1 \subseteq X_2 \ \ \& \ \ Y_2 \subseteq Y_1 \\
&\implies U_F(X_2) \subseteq U_F(X_1) \ \ \& \ \ U_F(Y_1) \subseteq U_F(Y_2) \\
&\iff (U_F(Y_1), U_F(X_1)) \leq_i (U_F(Y_2), U_F(X_2)) \\
&\iff \mathcal{U}_F(X_1, Y_1) \leq_i \mathcal{U}_F(X_2, Y_2)
\end{aligned}$$

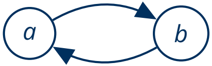
Approximator $\mathcal{U}_F$: Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Argumentation Framework

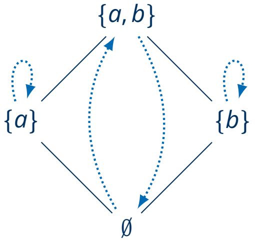
$F_5 =$ 

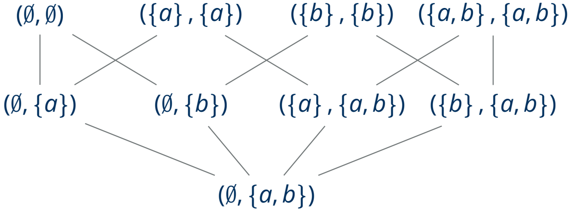
Operator U_F:

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{U}_F$ for U_F:

Approximator $\mathcal{U}_F$: Example

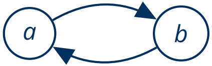




Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

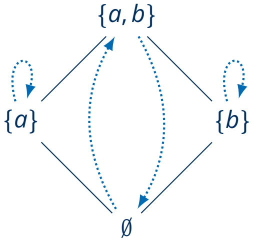
Argumentation Framework

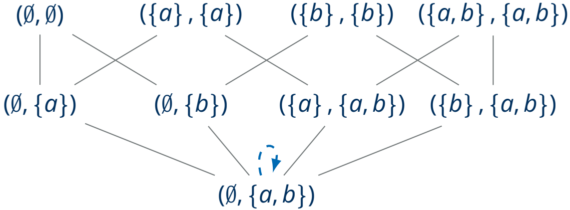
$F_5 =$ 

Approximator $\mathcal{U}_F$ for U_F:

Operator U_F:

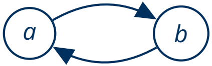
Approximator $\mathcal{U}_F$: Example





Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Argumentation Framework

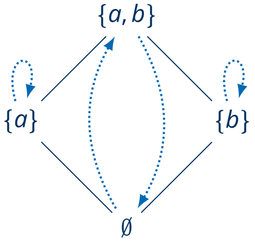
$F_5 =$ 

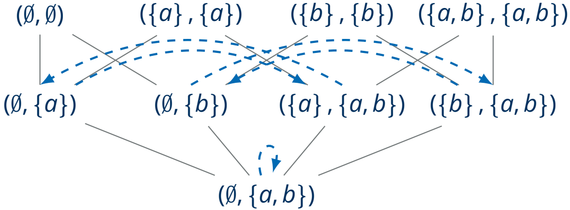
Operator U_F:

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{U}_F$ for U_F:

Approximator $\mathcal{U}_F$: Example

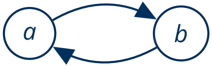




Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

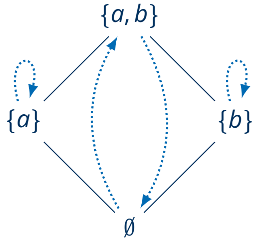
Argumentation Framework

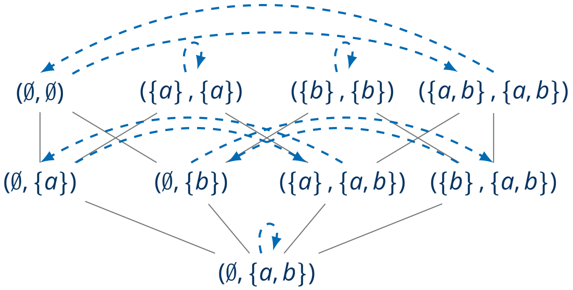
$F_5 =$ 

Approximator $\mathcal{U}_F$ for U_F:

Operator U_F:

Approximator $\mathcal{U}_F$: Example

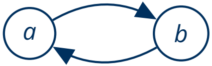




Original lattice $\left(2^{\{a,b\}}, \subseteq\right)$

Argumentation Framework

$F_5 =$



Operator U_F:

Bilattice $\left(2^{\{a,b\}} \times 2^{\{a,b\}}, \leq_i\right)$

Approximator $\mathcal{U}_F$ for U_F:

Quiz: Approximator $\mathcal{U}_F$

`https://tud.link/8jn6f9`

Recall: $\mathcal{U}_F(X, Y) = (U_F(Y), U_F(X))$, with $U_F(S) = A \setminus R(S)$.

Quiz

Consider the following argumentation framework:

$$F_7 = \begin{tikzcd} a \arrow[r, bend left] & b \arrow[l, bend left] \arrow[r, bend left] & c \arrow[l, bend left] \end{tikzcd}$$

What is the result of applying $\mathcal{U}_F$ to $(\{b\}, \{a, b, c\})$?

1. $(\emptyset, \{a, b, c\})$
2. $(\{b\}, \{a, b, c\})$
3. $(\emptyset, \{b\})$
4. $(\{a, b, c\}, \{b\})$

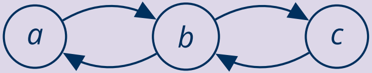
Quiz: Approximator $\mathcal{U}_F$

`https://tud.link/8jn6f9`

Recall: $\mathcal{U}_F(X, Y) = (U_F(Y), U_F(X))$, with $U_F(S) = A \setminus R(S)$.

Quiz

Consider the following argumentation framework:

$F_7 =$ 

What is the result of applying $\mathcal{U}_F$ to $(\{b\}, \{a, b, c\})$?

1. $(\emptyset, \{a, b, c\})$ ✗
2. $(\{b\}, \{a, b, c\})$
3. $(\emptyset, \{b\})$
4. $(\{a, b, c\}, \{b\})$

Quiz: Approximator $\mathcal{U}_F$

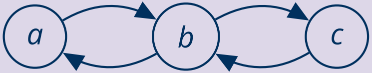
`https://tud.link/8jn6f9`

Recall: $\mathcal{U}_F(X, Y) = (U_F(Y), U_F(X))$, with $U_F(S) = A \setminus R(S)$.

Quiz

Consider the following argumentation framework:

$F_7 =$



What is the result of applying $\mathcal{U}_F$ to $(\{b\}, \{a, b, c\})$?

1. $(\emptyset, \{a, b, c\})$ ✗
2. $(\{b\}, \{a, b, c\})$
3. $(\emptyset, \{b\})$
4. $(\{a, b, c\}, \{b\})$ ✗

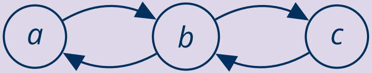
Quiz: Approximator $\mathcal{U}_F$

`https://tud.link/8jn6f9`

Recall: $\mathcal{U}_F(X, Y) = (U_F(Y), U_F(X))$, with $U_F(S) = A \setminus R(S)$.

Quiz

Consider the following argumentation framework:

$F_7 =$ 

What is the result of applying $\mathcal{U}_F$ to $(\{b\}, \{a, b, c\})$?

1. $(\emptyset, \{a, b, c\})$ ✗
2. $(\{b\}, \{a, b, c\})$ ✗
3. $(\emptyset, \{b\})$
4. $(\{a, b, c\}, \{b\})$ ✗

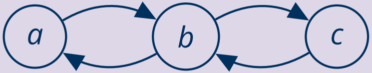
Quiz: Approximator $\mathcal{U}_F$

`https://tud.link/8jn6f9`

Recall: $\mathcal{U}_F(X, Y) = (U_F(Y), U_F(X))$, with $U_F(S) = A \setminus R(S)$.

Quiz

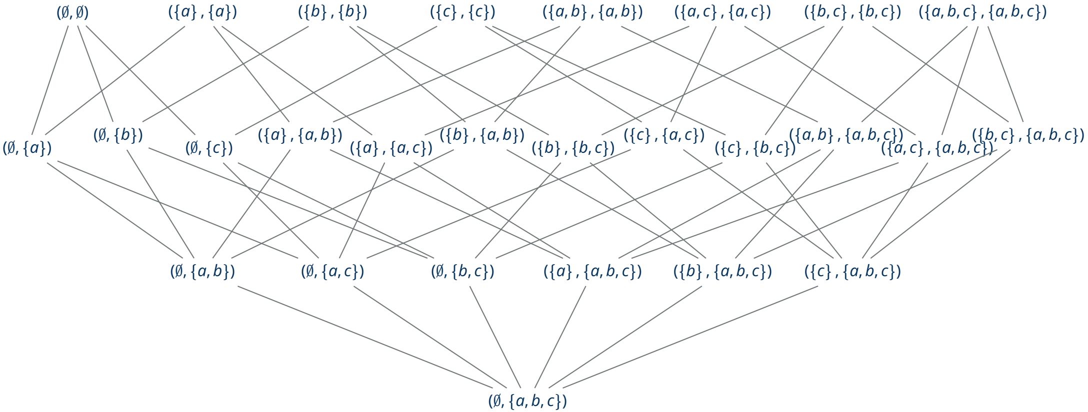
Consider the following argumentation framework:

$F_7 =$ 

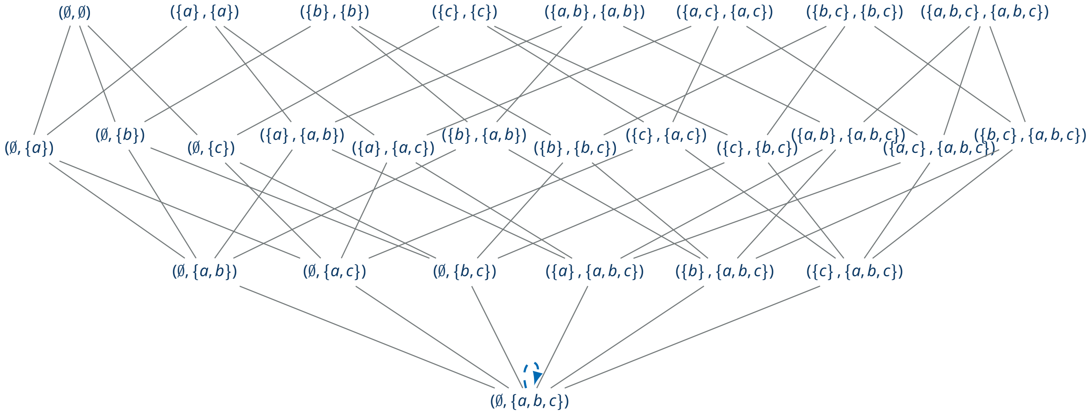
What is the result of applying $\mathcal{U}_F$ to $(\{b\}, \{a, b, c\})$?

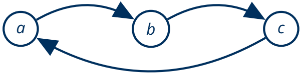
1. $(\emptyset, \{a, b, c\})$ ✗
2. $(\{b\}, \{a, b, c\})$ ✗
3. $(\emptyset, \{b\})$ ✓
4. $(\{a, b, c\}, \{b\})$ ✗

Approximator $\mathcal{U}_F$: Examples

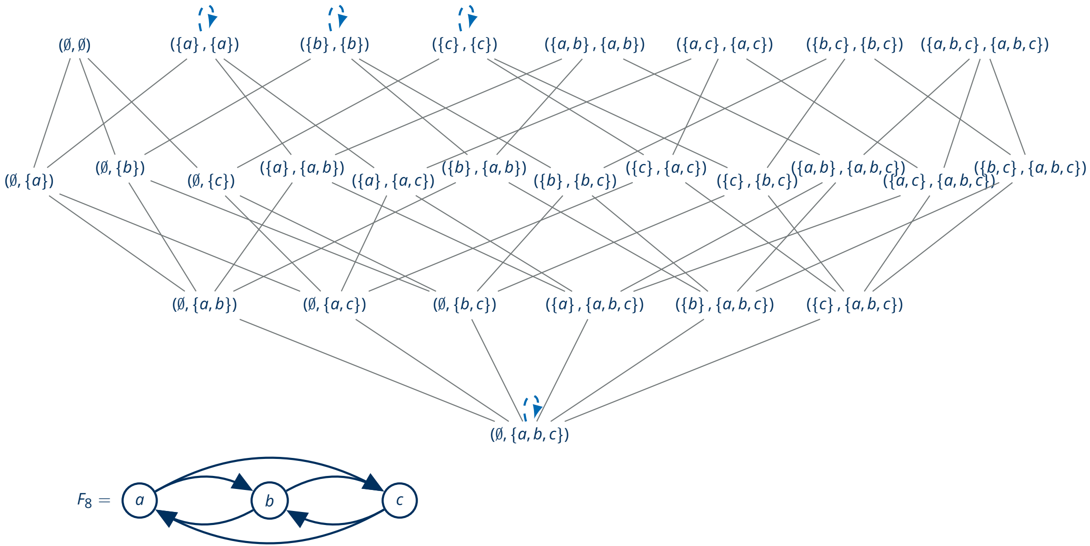


Approximator $\mathcal{U}_F$: Examples



$F_6 =$ 

Approximator $\mathcal{U}_F$: Examples



Recovering Semantics

Approximator fixpoints give rise to several semantics.

Theorem

Let $F = (A, R)$ be an argumentation framework and $X \subseteq Y \subseteq A$.

- X is stable for F iff $\mathcal{U}_F(X, X) = (X, X)$.
- (X, Y) is complete for F iff $\mathcal{U}_F(X, Y) = (X, Y)$.
- (X, Y) is grounded for F iff $(X, Y) = \mathrm{lfp}(\mathcal{U}_F)$.
- (X, Y) is admissible for F iff $(X, Y) \leq_i \mathcal{U}_F(X, Y)$.

Further semantics (e.g. preferred, ideal) via maximisation/intersection/...

Recovering Semantics

Approximator fixpoints give rise to several semantics.

Theorem

Let $F = (A, R)$ be an argumentation framework and $X \subseteq Y \subseteq A$.

- X is stable for F iff $\mathcal{U}_F(X, X) = (X, X)$.
- (X, Y) is complete for F iff $\mathcal{U}_F(X, Y) = (X, Y)$.
- (X, Y) is grounded for F iff $(X, Y) = \text{lfp}(\mathcal{U}_F)$.
- (X, Y) is admissible for F iff $(X, Y) \leq_i \mathcal{U}_F(X, Y)$.

Further semantics (e.g. preferred, ideal) via maximisation/intersection/...

So what does it buy us?

Recovering Semantics

Approximator fixpoints give rise to several semantics.

Theorem

Let $F = (A, R)$ be an argumentation framework and $X \subseteq Y \subseteq A$.

- X is stable for F iff $\mathcal{U}_F(X, X) = (X, X)$.
- (X, Y) is complete for F iff $\mathcal{U}_F(X, Y) = (X, Y)$.
- (X, Y) is grounded for F iff $(X, Y) = \text{lfp}(\mathcal{U}_F)$.
- (X, Y) is admissible for F iff $(X, Y) \leq_i \mathcal{U}_F(X, Y)$.

Further semantics (e.g. preferred, ideal) via maximisation/intersection/...

So what does it buy us?

For a new formalism, we only have to define an approximator!

Abstract Dialectical Frameworks: Syntax

Main Idea: Allow for more flexible specification of argument relationships.

Definition (Brewka and Woltran, 2010)

An **abstract dialectical framework** (ADF) is a triple $D = (S, L, C)$ with

- a finite set S of statements (arguments),
- a set $L \subseteq S \times S$ of links, $(par(s) = \{r \in S \mid (r, s) \in L\})$
- a family $C = \{C_s\}_{s \in S}$ of acceptance conditions $C_s \colon 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$.

Abstract Dialectical Frameworks: Syntax

Main Idea: Allow for more flexible specification of argument relationships.

Definition (Brewka and Woltran, 2010)

An **abstract dialectical framework** (ADF) is a triple $D = (S, L, C)$ with

- a finite set S of statements (arguments),
- a set $L \subseteq S \times S$ of links, $(par(s) = \{r \in S \mid (r, s) \in L\})$
- a family $C = \{C_s\}_{s \in S}$ of acceptance conditions $C_s \colon 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$.

- For $M \subseteq par(s)$, $C_s(M) = \mathbf{t}$ expresses that s can be accepted if all statements in M are accepted (and all statements in $par(s) \setminus M$ are not accepted).

Abstract Dialectical Frameworks: Syntax

Main Idea: Allow for more flexible specification of argument relationships.

Definition (Brewka and Woltran, 2010)

An **abstract dialectical framework** (ADF) is a triple $D = (S, L, C)$ with

- a finite set S of statements (arguments),
- a set $L \subseteq S \times S$ of links, $\quad (par(s) = \{r \in S \mid (r, s) \in L\})$
- a family $C = \{C_s\}_{s \in S}$ of acceptance conditions $C_s : 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$.

- For $M \subseteq par(s)$, $C_s(M) = \mathbf{t}$ expresses that s can be accepted if all statements in M are accepted (and all statements in $par(s) \setminus M$ are not accepted).
- An acceptance condition C_s is typically represented by a propositional formula φ_s over $par(s)$, with all $M \subseteq par(s)$ satisfying $C_s(M) = \mathbf{t}$ iff $M \models \varphi_s$.

ADFs: Syntax and Semantics

Main Idea: Allow for more flexible specification of argument relationships.

Definition (Brewka and Woltran, 2010)

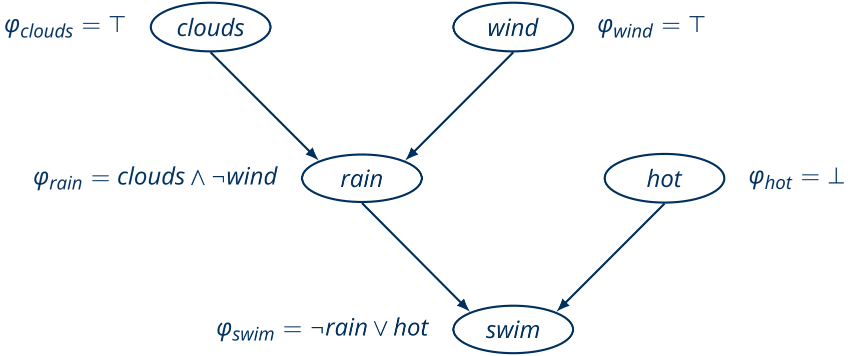
An **abstract dialectical framework** (ADF) is a triple $D = (S, L, C)$ with

- a finite set S of statements (arguments),
- a set $L \subseteq S \times S$ of links, $\quad (par(s) = \{r \in S \mid (r, s) \in L\})$
- a family $C = \{C_s\}_{s \in S}$ of acceptance conditions $C_s \colon 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$.

A set $M \subseteq S$ is a **model** for D iff
for all $s \in S$, we have $s \in M$ iff $C_s(M \cap par(s)) = \mathbf{t}$.

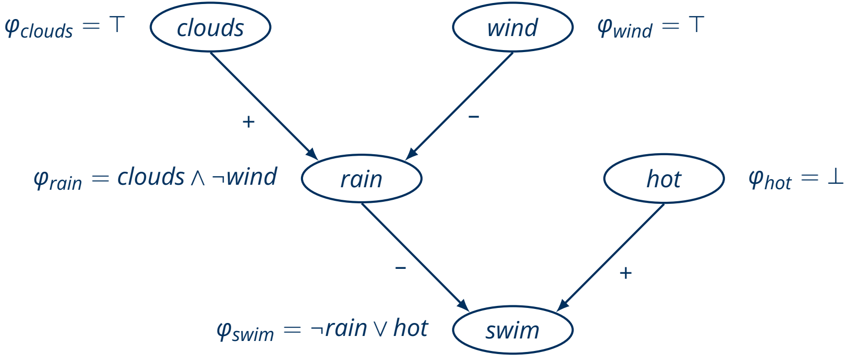
- For $M \subseteq par(s)$, $C_s(M) = \mathbf{t}$ expresses that s can be accepted if all statements in M are accepted (and all statements in $par(s) \setminus M$ are not accepted).
- An acceptance condition C_s is typically represented by a propositional formula φ_s over $par(s)$, with all $M \subseteq par(s)$ satisfying $C_s(M) = \mathbf{t}$ iff $M \models \varphi_s$.

Abstract Dialectical Frameworks: Example



Single model: $M = \{clouds, wind, swim\}$

Bipolar ADFs: Example



Single model: $M = \{clouds, wind, swim\}$

Bipolar: All links are attacking (–) or supporting (+).

Link (r, s) is **attacking** iff for all $M \subseteq par(s)$, if $C_s(M) = \mathbf{f}$ then $C_s(M \cup \{r\}) = \mathbf{f}$;
link (r, s) is **supporting** iff for all $M \subseteq par(s)$, if $C_s(M) = \mathbf{t}$ then $C_s(M \cup \{r\}) = \mathbf{t}$.

From AFs to ADFs: Translation

Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s \colon 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

From AFs to ADFs: Translation

Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s \colon 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

From AFs to ADFs: Translation

Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s \colon 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

From AFs to ADFs: Translation

Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s \colon 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

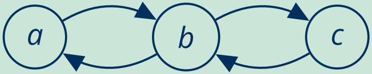
From AFs to ADFs: Translation

Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s \colon 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Example

$F_7 =$ 

$\rightsquigarrow \quad D_{F_7} =$

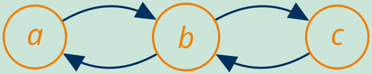
From AFs to ADFs: Translation

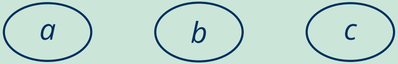
Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s \colon 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Example

$F_7 =$ 

$\rightsquigarrow$ $D_{F_7} =$ 

From AFs to ADFs: Translation

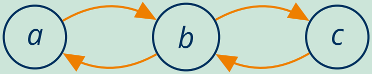
Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

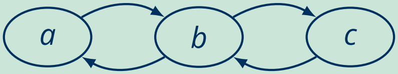
$$C_s : 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Example

$F_7 =$



$\rightsquigarrow$ $D_{F_7} =$



From AFs to ADFs: Translation

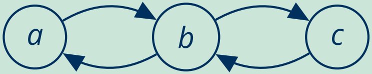
Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

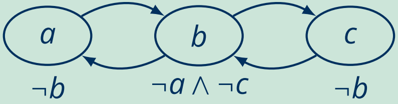
$$C_s : 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Example

$F_7 =$



$\rightsquigarrow$ $D_{F_7} =$



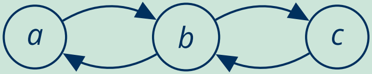
From AFs to ADFs: Translation

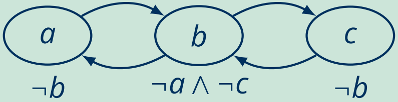
Definition

Let $F = (A, R)$ be an argumentation framework. Define its corresponding ADF $D_F = (S, L, C)$ by setting $S = A$, $L = R$, and for every $s \in S$:

$$C_s : 2^{\mathbf{par}(s)} \to \{\mathbf{t}, \mathbf{f}\}, \qquad M \mapsto \begin{cases} \mathbf{t} & \text{if } M = \emptyset, \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

Example

$F_7 =$ 

$\rightsquigarrow$ $D_{F_7} =$ 

Proposition

For any $F = (A, R)$: $M \subseteq A$ is stable for F iff M is a model of D_F.

ADFs: Operator

Definition

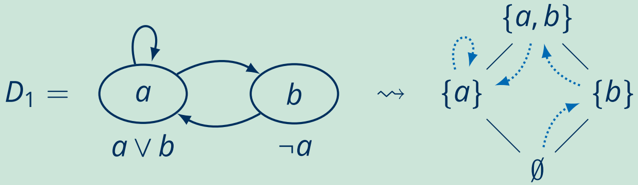
Let $D = (S, L, C)$ be an abstract dialectical framework. A consequence operator is given by $G_D \colon 2^S \to 2^S$ with $M \mapsto \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$.

ADFs: Operator

Definition

Let $D = (S, L, C)$ be an abstract dialectical framework. A consequence operator is given by $G_D \colon 2^S \to 2^S$ with $M \mapsto \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$.

Example



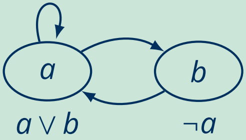
ADFs: Operator

Definition

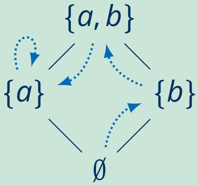
Let $D = (S, L, C)$ be an abstract dialectical framework. A consequence operator is given by $G_D \colon 2^S \to 2^S$ with $M \mapsto \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$.

Example

$D_1 =$



$\rightsquigarrow$



Proposition

Let $D = (S, L, C)$ be an abstract dialectical framework. For any $M \subseteq S$:

$$G_D(M) = M \text{ if and only if } M \text{ is a model for } D.$$

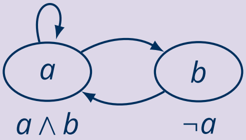
Quiz: `https://tud.link/djuy7e`

Recall: $G_D(M) = \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$

Quiz

Consider the following ADF:

$D_5 =$



Which of the following equations hold?

1. $G_D(\emptyset) = \{b\}$
2. $G_D(\{a\}) = \{b\}$
3. $G_D(\{b\}) = \{a, b\}$
4. $G_D(\{a, b\}) = \{b\}$

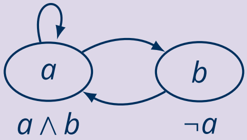
Quiz: `https://tud.link/djuy7e`

Recall: $G_D(M) = \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$

Quiz

Consider the following ADF:

$D_5 =$



Which of the following equations hold?

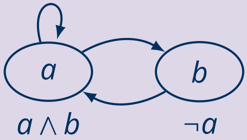
1. $G_D(\emptyset) = \{b\}$ ✓
2. $G_D(\{a\}) = \{b\}$
3. $G_D(\{b\}) = \{a, b\}$
4. $G_D(\{a, b\}) = \{b\}$

Quiz: `https://tud.link/djuy7e`

Recall: $G_D(M) = \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$

Quiz

Consider the following ADF:

$D_5 =$ 

Which of the following equations hold?

1. $G_D(\emptyset) = \{b\}$ ✓
2. $G_D(\{a\}) = \{b\}$ ✓
3. $G_D(\{b\}) = \{a, b\}$
4. $G_D(\{a, b\}) = \{b\}$

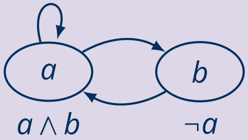
Quiz: `https://tud.link/djuy7e`

Recall: $G_D(M) = \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$

Quiz

Consider the following ADF:

$D_5 =$



Which of the following equations hold?

1. $G_D(\emptyset) = \{b\}$ ✓
2. $G_D(\{a\}) = \{b\}$ ✓
3. $G_D(\{b\}) = \{a, b\}$ ✗
4. $G_D(\{a, b\}) = \{b\}$

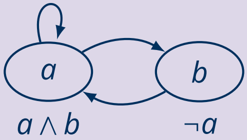
Quiz: `https://tud.link/djuy7e`

Recall: $G_D(M) = \{s \in S \mid C_s(M \cap par(s)) = \mathbf{t}\}$

Quiz

Consider the following ADF:

$D_5 =$



Which of the following equations hold?

1. $G_D(\emptyset) = \{b\}$ ✓
2. $G_D(\{a\}) = \{b\}$ ✓
3. $G_D(\{b\}) = \{a, b\}$ ✗
4. $G_D(\{a, b\}) = \{b\}$ ✗

ADFs: Approximator

Main Benefit of Approximation Fixpoint Theory

To obtain semantics for ADFs, we only need to define an approximator.

ADFs: Approximator

Main Benefit of Approximation Fixpoint Theory

To obtain semantics for ADFs, we only need to define an approximator.

Definition

Let $D = (S, L, C)$ be an ADF. Define approximator $\mathcal{G}_D : (2^S \times 2^S) \to (2^S \times 2^S)$ via

$$(X, Y) \mapsto \left(\bigcap_{X \subseteq Z \subseteq Y} G_D(Z), \bigcup_{X \subseteq Z \subseteq Y} G_D(Z) \right)$$

ADFs: Approximator

Main Benefit of Approximation Fixpoint Theory

To obtain semantics for ADFs, we only need to define an approximator.

Definition

Let $D = (S, L, C)$ be an ADF. Define approximator $\mathcal{G}_D \colon (2^S \times 2^S) \to (2^S \times 2^S)$ via

$$(X, Y) \mapsto \left(\bigcap_{X \subseteq Z \subseteq Y} G_D(Z), \bigcup_{X \subseteq Z \subseteq Y} G_D(Z) \right)$$

- $\mathcal{G}_D$ approximates G_D, as $\mathcal{G}_D(X, X) = (G_D(X), G_D(X))$.
- $\mathcal{G}_D$ is $\leq_i$-monotone: $(X_1, Y_1) \leq_i (X_2, Y_2)$ implies $X_1 \subseteq X_2 \subseteq Z \subseteq Y_2 \subseteq Y_1$.
- This construction is known as ultimate approximation (Denecker, Marek, and Truszczyński, 2004).

From AFs to ADFs: Defining Semantics

Definition

Let $D = (S, L, C)$ be an ADF. A pair (X, Y) is ...

- **admissible** iff $(X, Y) \leq_i \mathcal{G}_D(X, Y)$;
- **complete** iff $\mathcal{G}_D(X, Y) = (X, Y)$;
- **preferred** iff (X, Y) is $\leq_i$-maximal w.r.t. $\mathcal{G}_D(X, Y) = (X, Y)$;
- **grounded** iff $(X, Y) = \text{lfp}(\mathcal{G}_D)$.

From AFs to ADFs: Defining Semantics

Definition

Let $D = (S, L, C)$ be an ADF. A pair (X, Y) is ...

- **admissible** iff $(X, Y) \leq_i \mathcal{G}_D(X, Y)$;
- **complete** iff $\mathcal{G}_D(X, Y) = (X, Y)$;
- **preferred** iff (X, Y) is $\leq_i$-maximal w.r.t. $\mathcal{G}_D(X, Y) = (X, Y)$;
- **grounded** iff $(X, Y) = \mathrm{lfp}(\mathcal{G}_D)$.

Theorem

Let $F = (A, R)$ be an AF and D_F its corresponding ADF, and $X \subseteq Y \subseteq A$.

- (X, Y) is admissible for F iff (X, Y) is admissible for D_F;
- (X, Y) is complete for F iff (X, Y) is complete for D_F;
- (X, Y) is grounded for F iff (X, Y) is grounded for D_F;
- (X, X) is stable for F iff X is a model of D_F.

Towards Stable Model Semantics

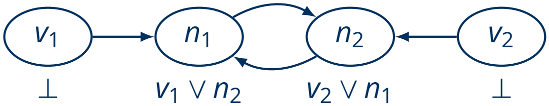
Consider this simplified model of a fuel system for an aircraft:
Node n_1 is pressurised by valve v_1 or node n_2; symmetrically for node n_2.

Towards Stable Model Semantics

Consider this simplified model of a fuel system for an aircraft:
Node n_1 is pressurised by valve v_1 or node n_2; symmetrically for node n_2.

We can model the behaviour of this system as an ADF as follows:

$D_2 =$

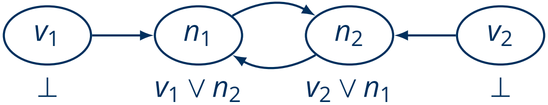


Towards Stable Model Semantics

Consider this simplified model of a fuel system for an aircraft:
Node n_1 is pressurised by valve v_1 or node n_2; symmetrically for node n_2.

We can model the behaviour of this system as an ADF as follows:

$D_2 =$

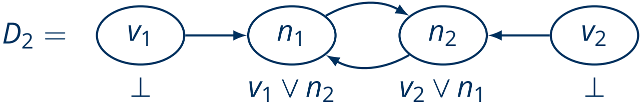


What are the models of D_2?

Towards Stable Model Semantics

Consider this simplified model of a fuel system for an aircraft:
Node n_1 is pressurised by valve v_1 or node n_2; symmetrically for node n_2.

We can model the behaviour of this system as an ADF as follows:



What are the models of D_2?

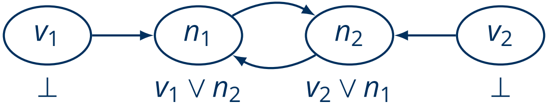
There are two models

Towards Stable Model Semantics

Consider this simplified model of a fuel system for an aircraft:
Node n_1 is pressurised by valve v_1 or node n_2; symmetrically for node n_2.

We can model the behaviour of this system as an ADF as follows:

$D_2 =$



What are the models of D_2?

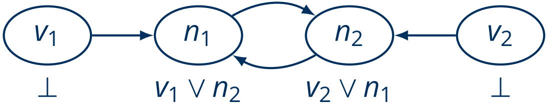
There are two models: $\emptyset$ and $\{n_1, n_2\}$.

Towards Stable Model Semantics

Consider this simplified model of a fuel system for an aircraft:
Node n_1 is pressurised by valve v_1 or node n_2; symmetrically for node n_2.

We can model the behaviour of this system as an ADF as follows:

$D_2 =$



What are the models of D_2?

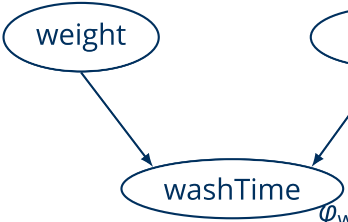
There are two models: $\emptyset$ and $\{n_1, n_2\}$.

Is this desired?

Weighted ADFs

$\varphi_{\text{weight}} = 0.8$

$\varphi_{\text{clean}} = 0.25$



$\varphi_{\text{washTime}} = \text{weight} \land \neg\text{clean}$

Weighted ADFs: Values I

Assume a complete lattice $(v, \leq_v)$, intutively representing the acceptance values.

Example

- $([0,1], \leq)$,
- $(\{\mathbf{t}, \mathbf{f}\}, \leq_t)$,
- $(\{(0,0),(0,1),(1,0),(1,1)\}, \leq_{\mathtt{prod}})$ with $\leq_{\mathtt{prod}}$ the product comparison.

Given a set of statements S, $int(v, S)$ consists of all functions $S \to v$.
Given $X, Y \in int(v, S)$, $X \leq_a Y$ iff $X(s) \leq_v Y(s)$ for every $s \in S$.

Example

Weighted ADFs: Values II

Given $S = \{\text{weight}, \text{clean}, \text{washTime}\}$, and $v = [0, 1]$,
$\{\text{weight} \mapsto 0.3, \text{clean} \mapsto 0.6, \text{washTime} \mapsto 1\} \in \mathit{int}(v, S)$.
We abbreviate this with $(0.3, 0.6, 1)$.
$(0.3, 0.6, 1) \leq_a (0.4, 0.7, 1)$.

Weighted ADFs: Definition

Definition Bart Bogaerts. 'Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 2686–2693

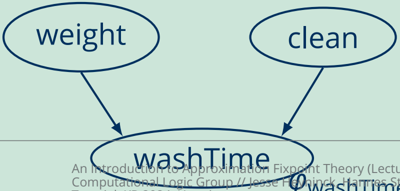
A **weighted abstract dialectical framework** (wADF) is a triple $D = (S, L, C)$ with

- a finite set S of statements (arguments),
- a set $L \subseteq S \times S$ of links, $(par(s) = \{r \in S \mid (r, s) \in L\})$
- a family $C = \{C_s\}_{s \in S}$ of acceptance conditions $C_s : int(v, S) \to v$.

Example: Washing machine

$\varphi_{\text{weight}} = 0.8$

$\varphi_{\text{clean}} = 0.25$



$\varphi_{\text{washTime}} = \max(v(\varphi_{\text{weight}}), (1 - v(\varphi_{\text{clean}})))$

Weighted ADFs: Operator

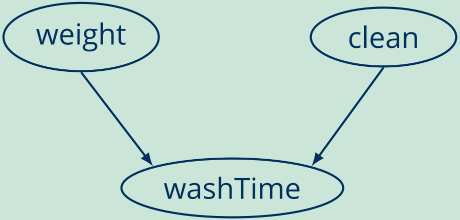
Definition

Let $D = (S, L, C)$ be an weighted ADF over v. A consequence operator is given by $G_D : int(v, S) \to int(v, S)$ with $G_D(x) : s \mapsto C_s(x)$.

Example

$\varphi_{\text{weight}} = 0.8$

$\varphi_{\text{clean}} = 0.25$



$\varphi_{\text{washTime}} = \max(v(\varphi_{\text{weight}}), (1 - \varphi(\text{clean}))$

$G_D(0.3, 0.6, 1) = (0.8, 0.25, 0.4)$.

$G_D(0.8, 0.25, 0.8) = (0.8, 0.25, 0.8)$.

Weighted ADFs: Approximator

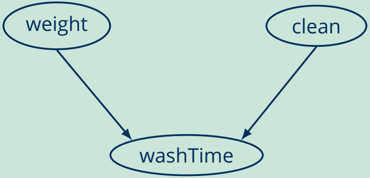
Definition

Let $D = (S, L, C)$ be an weighted ADF over v. Define the **ultimate approximator of** D as follows:

$$\mathcal{X}_{G_D} : int(v,S)^2 \to int(v,S)^2,$$

$$(X,Y) \mapsto \left(\bigwedge \{G_D(Z) \mid X \leqslant Z \leqslant Y\}, \bigvee \{G_D(Z) \mid X \leqslant Z \leqslant Y\}\right)$$

Example



$\varphi_{\text{weight}} = 0.8 \quad \varphi_{\text{clean}} = 0.25$

$\varphi_{\text{washTime}} = \max(v(\varphi_{\text{weight}}), (1 - \varphi(\text{clean}))$

$\mathcal{X}_{G_D}((0,0,0),(1,1,1)) =$
$((0.8, 0.25, 0), (0.8, 0.25, 1)),$
$\mathcal{X}_{G_D}((0.8, 0.25, 0), (0.8, 0.25, 1)) =$
$((0.8, 0.25, 0.8), (0.8, 0.25, 0.8)),$
$\mathcal{X}_{G_D}((0.8, 0.25, 0.8), (0.8, 0.25, 0.8)) =$
$((0.8, 0.25, 0.8), (0.8, 0.25, 0.8)).$

Weighted ADFs: Semantics and Classical ADFs

Definition

Let $D = (S, L, C)$ be an weighted ADF over v. A pair (X, Y) is …

- **admissible** iff $(X, Y) \leq_i \mathcal{X}_{G_D}(X, Y)$;
- **complete** iff $\mathcal{X}_{G_D}(X, Y) = (X, Y)$;
- **preferred** iff (X, Y) is $\leq_i$-maximal w.r.t. $\mathcal{X}_{G_D}(X, Y) = (X, Y)$;
- **grounded** iff $(X, Y) = \mathrm{lfp}(\mathcal{X}_{G_D})$.
- a **two-valued stable model** iff $X = Y$ and $\mathcal{S}\mathcal{X}_{G_D}(X, Y) = (X, Y)$;
- a **three-valued stable model** iff $X \leq_a Y$ and $\mathcal{S}\mathcal{X}_{G_D}(X, Y) = (X, Y)$;
- the **well-founded model** iff it is the least fixpoint of $\mathcal{S}\mathcal{X}_{G_D}$.

Relation to Classical ADFs

The weighted ADFs over $(\{\mathbf{t}, \mathbf{f}\}, \leq_t)$ are equivalent to the "classical" ADFs we've seen in the previous sections.

Stocktaking

- Weighted ADFs define a very rich class of formalisms that allow more fine-grained evaluation of arguments then ADFs.
- AFT-based development of their semantics was very straightforward: the main task was to generalise the notion of acceptance conditions.
- On the basis of that, the only task was to define a (non-monotonic) operator.
- Approximator and semantics are straightforward applications of the AFT-definitions.

Stratification

Motivation

```
p:- not q.
s:- p, not r.
r:- p, not s.
```

- We can split the search for fixpoints in two parts:
  - one related to $\{q, p\}$, and
  - and one related to $\{r, s\}$ based on our findings about $\{q, p\}$.
- Approach this topic purely algebraically, so it applies to any instantiation of AFT.

Preliminaries: Sub-lattices

Definition

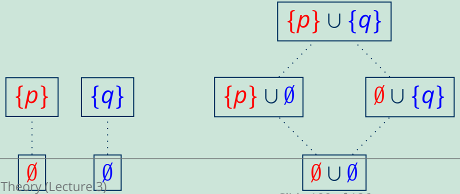
Let I be a set, which we call the index set, and for each $i \in I$, let L_i be a set. The product set $\bigotimes_{i \in I} L_i$ is the following set of functions:

$$\bigotimes_{i \in I} L_i = \{f \mid f : I \to \bigcup_{i \in I} L_i \text{ s.t. } \forall i \in I : f(i) \in L_i\}$$

$\bigotimes_{i \in I} L_i$ contains all ways of selecting one element of every set L_i.
For a finite $I = \{1, \ldots, n\}$, $\otimes_{i \in I} L_i$ is (isomorphic to) $L_1 \times \ldots \times L_n$.

Example

$L_1 = \{\emptyset, \{p\}\}$ and $L_2 = \{\emptyset, \{q\}\}$,
$\bigotimes_{i \in \{1,2\}} L_i$ contains
$f(1) = f(2) = \emptyset$, and
$f'(1) = \emptyset$ and $f'(2) = \{q\}$,



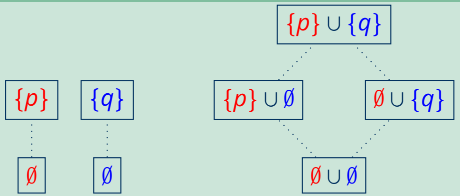
Preliminaries: sub-lattices

Definition

Given a product set $\bigotimes_{i\in I} L_i$ s.t. each L_j is partially ordered by $\leq_j$, the product order $\leq_\otimes$ on $\bigotimes_{j\in I} L_j$ is defined by: for all $x, y \in \bigotimes_{j\in I} L_j$, $x \leq_\otimes y$ iff for all $j \in I$, $x(j) \leq_j y(j)$.
It can be easily shown that if all $\langle L_j, \leq_j\rangle$ are (complete) lattices, then $\langle \bigotimes_{j\in I} L_j, \leq_\otimes\rangle$ is also a (complete) lattice, called the product lattice of the lattices L_j.

Example

As $(L_1 = \{\emptyset, \{p\}\}, \subseteq)$ and $(L_2 = \{\emptyset, \{q\}\}, \subseteq)$ are complete lattices, $(\bigotimes_{i\in\{1,2\}} L_i, \subseteq_\otimes)$ is a complete lattice.



Preliminaries: sub-lattices

Definition

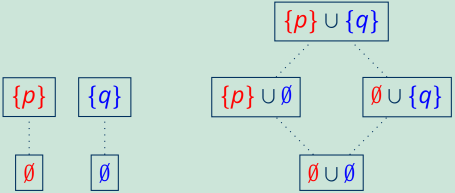
We denote, for a product lattice $\bigotimes_{i \in I} L_i$, $x \in \bigotimes_{i \in I} L_i$ and $j \in I$, $x_{|\leq j} = \bigotimes_{i \leq j} f(i)$.

Or, slightly abusing notation, $x_{|\leq j} = x_1 \otimes \ldots \otimes x_j$.

Example

$L_1 = \{\emptyset, \{p\}\}$ and $L_2 = \{\emptyset, \{q\}\}$

$\{p\} \cup \{q\}_{|\leq 1} = \{p\}$.



Stratification

Definition

An operator is stratifiable (over $\bigotimes_{i\in I} L_i$) iff for every $x^1, x^2 \in \bigotimes_{i\in I} L_i$ and every $j \in I$:

$$\text{if} \quad x^1_{|\leq j} = x^2_{|\leq j} \quad \text{then } O(x)_{|\leq j} = O(y)_{|\leq j}.$$

Example

For $P = \{p \leftarrow \sim q., \quad r \leftarrow p, \sim r., \quad s \leftarrow p, \sim s.\}$,
T_P is stratifiable over $\{p, q\} \otimes \{r, s\}$.
For example,

$$T_P(\{q, r\}) \cap \{p, q\} = \emptyset$$
$$T_P(\{q, s\}) \cap \{p, q\} = \emptyset$$

Results on Stratification

Theorem

Let $L = \bigotimes_{i \in I} L_i$ be a product lattice, $O : L \to L$ an operator on L and $\mathcal{A} : L^2 \to L^2$ an approximator of O. If $\mathcal{A}$ is stratifiable, so is O. Furthermore, the following holds for each pair $(x, y) \in L^2$:

- (x, y) is a fixpoint of $\mathcal{A}$ if and only if for each $i \in I$, $(x_{|\leq i}, y_{|\leq i})$ is a fixpoint of $\mathcal{A}_{|\leq i}$,
- (x, y) is the Kripke-Kleene fixpoint of $\mathcal{A}$ if and only if for each $i \in I$, $(x_{|\leq i}, y_{|\leq i})$ is the Kripke-Kleene fixpoint of $\mathcal{A}_{|\leq i}$,
- (x, y) is the well-founded fixpoint of $\mathcal{A}$ if and only if for each $i \in I$, $(x_{|\leq i}, y_{|\leq i})$ is the well-founded fixpoint of $\mathcal{A}_{|\leq i}$,
- (x, y) is a $\mathcal{A}$-stable fixpoint if and only if for each $i \in I$, $(x_{|\leq i}, y_{|\leq i})$ is an $\mathcal{A}_{|\leq i}$-stable fixpoint.

Stocktaking

Summary

- Purely algebraic definition and results on concept previously studied for logic programs.

Stocktaking

Summary

- Purely algebraic definition and results on concept previously studied for logic programs.
- Straightforwardly applies to any existing or future application of AFT.

Stocktaking

Summary

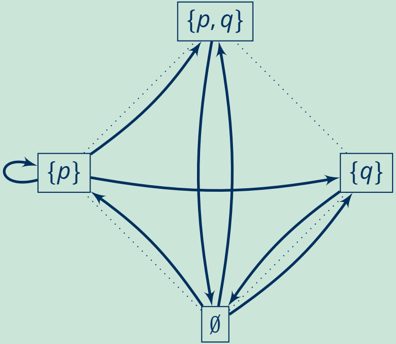
- Purely algebraic definition and results on concept previously studied for logic programs.
- Straightforwardly applies to any existing or future application of AFT.
- Allows for a language-independent study of concepts in NMR.

Non-Deterministic Operators

Disjunctive logic programming

Example

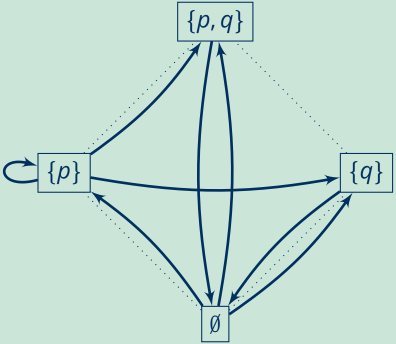
$\mathcal{P} = \{p \vee q \leftarrow \neg q\}$.



Disjunctive logic programming

Example

$\mathcal{P} = \{p \vee q \leftarrow \neg q\}$.



Such an operator cannot be captured in AFT!

Non-Deterministic Operators Pelov and Truszczynski, ‘Semantics of disjunctive programs with monotone aggregates-an operator-based approach.’

A non-deterministic operator on $\mathcal{L}$ is a function:

$$O_{\mathcal{L}} : \mathcal{L} \rightarrow 2^{\mathcal{L}} \setminus \{\emptyset\}$$

Non-Deterministic Operators Pelov and Truszczynski, 'Semantics of disjunctive programs with monotone aggregates-an operator-based approach.'

A non-deterministic operator on $\mathcal{L}$ is a function:

$$O_{\mathcal{L}} : \mathcal{L} \to 2^{\mathcal{L}} \setminus \{\emptyset\}$$

Output of $O_{\mathcal{L}}(x) = \{y_1, y_2, \ldots\}$ represents equally plausible choices we can make in view of x.

Example

Given a dlp $\mathcal{P}$ and a set of atoms x, we define:

- $\mathrm{HD}_{\mathcal{P}}(x) = \{\Delta \mid \bigvee \Delta \leftarrow \psi \in \mathcal{P}$ and $(x, x)(\psi) = \mathrm{T}\}$.
- $T_P(x) = \{y \subseteq \bigcup HD_{\mathcal{P}}(x) \mid \forall \Delta \in HD_{\mathcal{P}}(x),\ y \cap \Delta \neq \emptyset\}$.

Non-Deterministic Operators Pelov and Truszczynski, ‘Semantics of disjunctive programs with monotone aggregates-an operator-based approach.’

A non-deterministic operator on $\mathcal{L}$ is a function:

$$O_{\mathcal{L}} : \mathcal{L} \to 2^{\mathcal{L}} \setminus \{\emptyset\}$$

Output of $O_{\mathcal{L}}(x) = \{y_1, y_2, \ldots\}$ represents equally plausible choices we can make in view of x.

Example

Given a dlp $\mathcal{P}$ and a set of atoms x, we define:

- $\mathrm{HD}_{\mathcal{P}}(x) = \{\Delta \mid \bigvee \Delta \leftarrow \psi \in \mathcal{P}$ and $(x, x)(\psi) = \mathrm{T}\}$.
- $T_P(x) = \{y \subseteq \bigcup HD_{\mathcal{P}}(x) \mid \forall \Delta \in HD_{\mathcal{P}}(x),\ y \cap \Delta \neq \emptyset\}$.

For $\mathcal{P} = \{p \vee q \leftarrow \neg q\}$, we have:

- $HD_{\mathcal{P}}(\emptyset) = \{\{p, q\}\}$, and
- $T_P(\emptyset) = \{\{p\}, \{q\}, \{p, q\}\}$

Non-Deterministic Approximation Operators

$O_{\mathcal{L}}$ is approximated using a non-deterministic approximation operator $\mathcal{A} : \mathcal{L}^2 \rightarrow 2^{\mathcal{L}} \times 2^{\mathcal{L}}$:

- that is $\preceq_i^A$-monotonic,
- $\mathcal{A}(x, x) = O_{\mathcal{L}}(x) \times O_{\mathcal{L}}(x)$ for every $x \in \mathcal{L}$.

Non-Deterministic Approximation Operators

$O_{\mathcal{L}}$ is approximated using a non-deterministic approximation operator $\mathcal{A} : \mathcal{L}^2 \to 2^{\mathcal{L}} \times 2^{\mathcal{L}}$:

- that is $\preceq_i^A$-monotonic,
- $\mathcal{A}(x, x) = O_{\mathcal{L}}(x) \times O_{\mathcal{L}}(x)$ for every $x \in \mathcal{L}$.

Output of $\mathcal{A}(x, y) = \{x_1, x_2, \ldots\} \times \{y_1, y_2, \ldots\}$ is a set of lower bounds respectively upper bounds on choices $O(z) = \{z_1, z_2, \ldots\}$ (with $x \leq z \leq y$).

Non-Deterministic Approximation Operators

$O_{\mathcal{L}}$ is approximated using a non-deterministic approximation operator $\mathcal{A} : \mathcal{L}^2 \rightarrow 2^{\mathcal{L}} \times 2^{\mathcal{L}}$:

- that is $\preceq_i^A$-monotonic,
- $\mathcal{A}(x, x) = O_{\mathcal{L}}(x) \times O_{\mathcal{L}}(x)$ for every $x \in \mathcal{L}$.

Output of $\mathcal{A}(x, y) = \{x_1, x_2, \ldots\} \times \{y_1, y_2, \ldots\}$ is a set of lower bounds respectively upper bounds on choices $O(z) = \{z_1, z_2, \ldots\}$ (with $x \leq z \leq y$).
We denote first component of $\mathcal{A}(x, y)$ by $\mathcal{A}_l(x, y)$ and the second component by $\mathcal{A}_u(x, y)$.

Non-Deterministic Approximation Operators: Example

For a dlp $\mathcal{P}$ and an interpretation (x, y), we define:

- $\mathcal{HD}^{l}_{\mathcal{P}}(x, y) = \{\Delta \mid \bigvee \Delta \leftarrow \varphi \in \mathcal{P}, (x, y)(\varphi) \geq_t \mathsf{C}\}$,
- $\mathcal{HD}^{u}_{\mathcal{P}}(x, y) = \{\Delta \mid \bigvee \Delta \leftarrow \varphi \in \mathcal{P}, (x, y)(\varphi) \geq_t \mathsf{U}\}$,

Non-Deterministic Approximation Operators: Example

For a dlp $\mathcal{P}$ and an interpretation (x,y), we define:

- $\mathcal{HD}^l_{\mathcal{P}}(x,y) = \{\Delta \mid \bigvee\Delta \leftarrow \varphi \in \mathcal{P}, (x,y)(\varphi) \geq_t \mathsf{C}\}$,
- $\mathcal{HD}^u_{\mathcal{P}}(x,y) = \{\Delta \mid \bigvee\Delta \leftarrow \varphi \in \mathcal{P}, (x,y)(\varphi) \geq_t \mathsf{U}\}$,
- for $x = u, l$, $\mathcal{T}^x_P(x,y) = \{v \subseteq \bigcup \mathcal{HD}^x_{\mathcal{P}}(x,y) \mid \forall \Delta \in \mathcal{HD}^x_{\mathcal{P}}(x,y),\ v \cap \Delta \neq \emptyset\}$,
- $\mathcal{T}_P(x,y) = (\mathcal{T}^l_P(x,y), \mathcal{T}^u_P(x,y))$.

Non-Deterministic Approximation Operators: Example

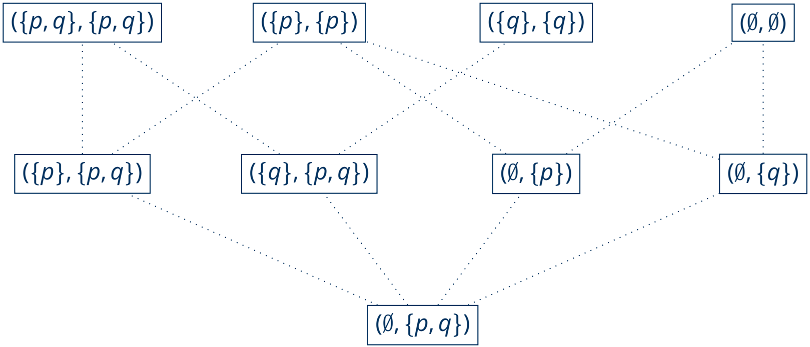
For a dlp $\mathcal{P}$ and an interpretation (x, y), we define:

- $\mathcal{HD}^l_{\mathcal{P}}(x, y) = \{\Delta \mid \bigvee \Delta \leftarrow \varphi \in \mathcal{P}, (x, y)(\varphi) \geq_t \mathsf{C}\}$,
- $\mathcal{HD}^u_{\mathcal{P}}(x, y) = \{\Delta \mid \bigvee \Delta \leftarrow \varphi \in \mathcal{P}, (x, y)(\varphi) \geq_t \mathsf{U}\}$,
- for $x = u, l$, $\mathcal{T}^x_P(x, y) = \{v \subseteq \bigcup \mathcal{HD}^x_{\mathcal{P}}(x, y) \mid \forall \Delta \in \mathcal{HD}^x_{\mathcal{P}}(x, y),\ v \cap \Delta \neq \emptyset\}$,
- $\mathcal{T}_P(x, y) = (\mathcal{T}^l_P(x, y), \mathcal{T}^u_P(x, y))$.

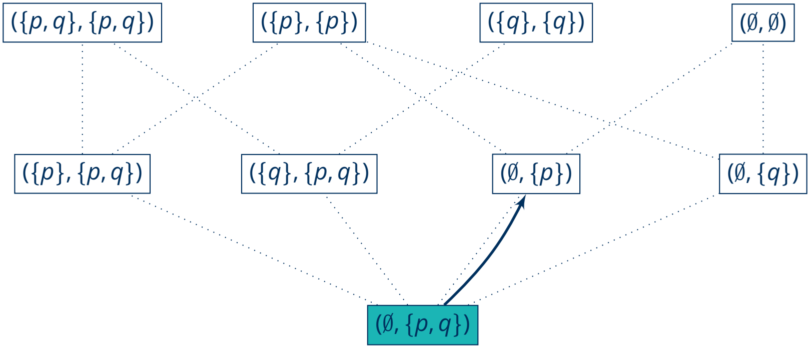
$\mathcal{P} = \{p \vee q \leftarrow \neg q\}$

$\mathcal{HD}^l(\emptyset, \{p, q\}) = \{\emptyset\}$.
$\mathcal{HD}^u(\emptyset, \{p, q\}) = \{\{p, q\}\}$.
$\mathcal{T}^l_P(\emptyset, \{p, q\}) = \{\emptyset\}$.
$\mathcal{T}^u_P(\emptyset, \{p, q\}) = \{\{p\}, \{q\}, \{p, q\}\}$.

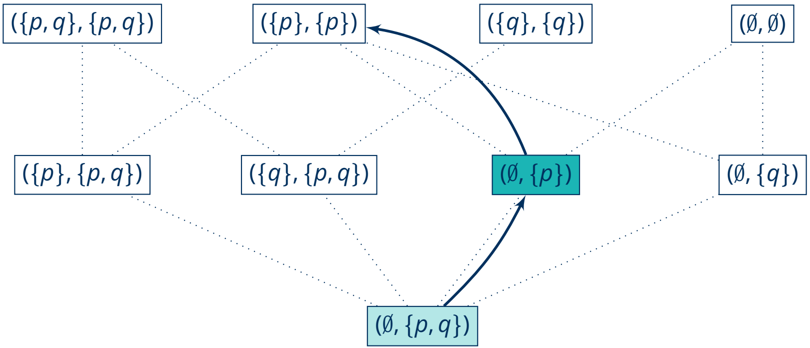
Kripke-Kleene Fixpoint (in deterministic AFT)



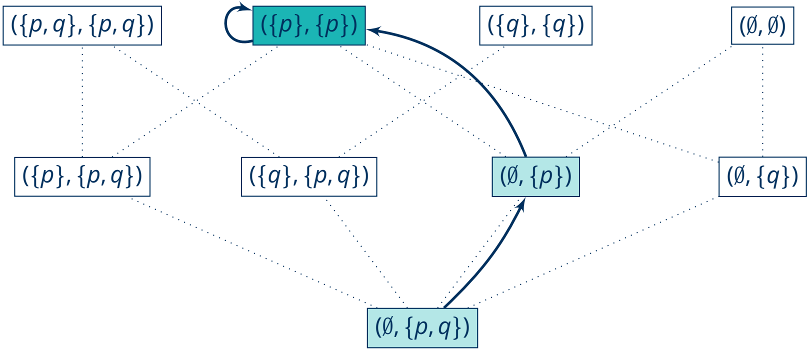
Kripke-Kleene Fixpoint (in deterministic AFT)



Kripke-Kleene Fixpoint (in deterministic AFT)



Kripke-Kleene Fixpoint (in deterministic AFT)



Kripke-Kleene-semantics for ndaos

($\leq_i$-minimal) Fixpoints of $\mathcal{A}$ (i.e. $x \in \mathcal{A}_l(x, y)$ and $y \in \mathcal{A}_u(x, y)$):

- $\vee$ Pairs (x, y)
- $\times$ not guaranteed to exist
- $\times$ not unique

Kripke-Kleene-semantics for ndaos

($\leq_i$-minimal) Fixpoints of $\mathcal{A}$ (i.e. $x \in \mathcal{A}_l(x, y)$ and $y \in \mathcal{A}_u(x, y)$):

∨ Pairs (x, y)

× not guaranteed to exist

× not unique

∨ **Theorem** Given a dlp $\mathcal{P}$ and some $x \subseteq y \subseteq \mathcal{A}_{\mathcal{P}}$, it holds that: (x, y) is a weakly supported model of $\mathcal{P}$ iff $(x, y) \in \mathcal{T}_P(x, y)$.

Kripke-Kleene-semantics for ndaos

($\leq_i$-minimal) Fixpoints of $\mathcal{A}$ (i.e. $x \in \mathcal{A}_l(x, y)$ and $y \in \mathcal{A}_u(x, y)$):

- ∨ Pairs (x, y)
- × not guaranteed to exist
- × not unique
- ∨ **Theorem** Given a dlp $\mathcal{P}$ and some $x \subseteq y \subseteq \mathcal{A}_{\mathcal{P}}$, it holds that: (x, y) is a weakly supported model of $\mathcal{P}$ iff $(x, y) \in \mathcal{T}_P(x, y)$.

Kripke-Kleene-State of $\mathcal{A}$:

- × Convex set of elements between set of lower bounds and set of upper bounds
- ∨ Existence guaranteed
- ∨ Unique
- ∨ Iterative construction

Construction of Kripke-Kleene state

Definition

Given a lattice $L = \langle \mathcal{L}, \leq \rangle$ and an element $X \in \mathcal{L}$, we define:

- the *upwards closure of X* is defined as $X\uparrow := \bigcup_{x \in X} \{y \in \mathcal{L} \mid x \leq y\}$,
- the *downwards closure of X* is defined as $X\downarrow := \bigcup_{x \in X} \{y \in \mathcal{L} \mid x \geq y\}$.

Fact

Given a lattice $L = \langle \mathcal{L}, \leq \rangle$ and a set $X \subseteq \mathcal{L}$, it holds that:

1. $X\uparrow \preceq_L^S X$ *and* $X \preceq_L^S X\uparrow$, *and*
2. $X\downarrow \preceq_L^H X$ *and* $X \preceq_L^H X\downarrow$.

Thus, upwards respectively downwards closure ensures anti-symmetry under $\preceq_L^S$ respectively $\preceq_L^H$.

Construction of the Kripke-Kleene state

For any pair of sets $X \times Y$ let:

$$\mathcal{A}'(X \times Y) = \bigcup_{x \in X, y \in Y} \mathcal{A}_l(x, y)\uparrow \times \bigcup_{x \in X, y \in Y} \mathcal{A}_u(x, y)\downarrow$$

Construction of the Kripke-Kleene state

For any pair of sets $X \times Y$ let:

$$\mathcal{A}'(X \times Y) = \bigcup_{x \in X, y \in Y} \mathcal{A}_l(x, y)\!\uparrow \times \bigcup_{x \in X, y \in Y} \mathcal{A}_u(x, y)\!\downarrow$$

$\mathcal{A}'$ is a $\preceq_i^A$-monotonic operator over the lattice consisting of pairs of upwards closed sets and downwards closed sets.

Construction of the Kripke-Kleene state

For any pair of sets $X \times Y$ let:

$$\mathcal{A}'(X \times Y) = \bigcup_{x \in X, y \in Y} \mathcal{A}_l(x, y)\uparrow \times \bigcup_{x \in X, y \in Y} \mathcal{A}_u(x, y)\downarrow$$

$\mathcal{A}'$ is a $\preceq_i^A$-monotonic operator over the lattice consisting of pairs of upwards closed sets and downwards closed sets.

Theorem

Let a complete lattice $L = \langle \mathcal{L}, \leq \rangle$ be given. Every $\preceq_i^A$-monotonic operator $\mathcal{A}' : \wp_\uparrow(\mathcal{L}) \times \wp_\downarrow(\mathcal{L}) \to \wp_\uparrow(\mathcal{L}) \times \wp_\downarrow(\mathcal{L})$ admits a unique $\preceq_i^A$-minimal fixpoint that can be constructed by iterative application of $\mathcal{A}'$ to $(\bot, \top)$.

Kripke-Kleene State: Example i

Let $\mathcal{P} = \{p \vee q \leftarrow\}$. We calculate $\mathsf{KK}(\mathcal{T}_P)$ as follows:

- $\mathcal{T}'_P(\emptyset, \{p,q\}) = \{\{p\}, \{q\}\ \{p,q\}\}\!\uparrow \times \{\{p\}, \{q\}, \{p,q\}\}\!\downarrow$.
- $\mathcal{T}'_P(\{\{p\}, \{q\}, \{p,q\}\}\!\uparrow \times \{\{p\}, \{q\}, \{p,q\}\}\!\downarrow) =$ $\{\{p\}, \{q\}, \{p,q\}\}\!\uparrow \times \{\{p\}, \{q\}, \{p,q\}\}\!\downarrow$ and thus a fixpoint is reached.

Kripke-Kleene State: Example ii

Let $\mathcal{P} = \{p \vee q \leftarrow, r \vee s \leftarrow \neg q\}$. We calculate $\mathrm{KK}(\mathcal{T}_P)$ as follows:

- $\mathcal{T}'_P(\emptyset, \{p, q, r, s\}) = \{\{p\}, \{q\}\}\!\uparrow \times \{\{p, r\}, \{p, s\}, \{q, r\}\{q, r\}\}\!\downarrow$.
- $\mathcal{T}'_P(\{\{p\}, \{q\}\}\!\uparrow \times \{\{p, r\}, \{p, s\}, \{q, r\}\{q, r\}\}\!\downarrow) =$
  $\{\{p\}, \{q\}\}\!\uparrow \times \{\{p, r\}, \{p, s\}, \{q, r\}\{q, r\}\}\!\downarrow$ and thus a fixpoint is reached.

Stable Operator for Deterministic Approximation Operators

Basic idea: look for smallest fixpoint that the upper bound allows us to construct:

$$S(\mathcal{A}_l)(y) = glb\{x \in \mathcal{L} \mid x = \mathcal{A}_l(x, y)\}.$$

Stable Operator for Deterministic Approximation Operators

Basic idea: look for smallest fixpoint that the upper bound allows us to construct:

$$S(\mathcal{A}_l)(y) = glb\{x \in \mathcal{L} \mid x = \mathcal{A}_l(x, y)\}.$$

(for finite lattices this is just the $\leq$-minimal fixpoint of $\mathcal{A}_l(., y)$)

Stable Operator for Deterministic Approximation Operators

Basic idea: look for smallest fixpoint that the upper bound allows us to construct:

$$S(\mathcal{A}_l)(y) = glb\{x \in \mathcal{L} \mid x = \mathcal{A}_l(x, y)\}.$$

(for finite lattices this is just the $\leq$-minimal fixpoint of $\mathcal{A}_l(., y)$)

$$S(\mathcal{A})(x, y) = (S(\mathcal{A}_l)(y), S(\mathcal{A}_u)(x)).$$

Stable Operator for Deterministic Approximation Operators

Basic idea: look for smallest fixpoint that the upper bound allows us to construct:

$$S(\mathcal{A}_l)(y) = glb\{x \in \mathcal{L} \mid x = \mathcal{A}_l(x, y)\}.$$

(for finite lattices this is just the $\leq$-minimal fixpoint of $\mathcal{A}_l(., y)$)

$$S(\mathcal{A})(x, y) = (S(\mathcal{A}_l)(y), S(\mathcal{A}_u)(x)).$$

Example

$S(\mathcal{T}_P)(y) = \min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{y})$.

Stable Operator for Deterministic Approximation Operators

Basic idea: look for smallest fixpoint that the upper bound allows us to construct:

$$S(\mathcal{A}_l)(y) = glb\{x \in \mathcal{L} \mid x = \mathcal{A}_l(x, y)\}.$$

(for finite lattices this is just the $\leq$-minimal fixpoint of $\mathcal{A}_l(., y)$)

$$S(\mathcal{A})(x, y) = (S(\mathcal{A}_l)(y), S(\mathcal{A}_u)(x)).$$

Example

$S(\mathcal{T}_P)(y) = \min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{y})$.

Fixpoints of $S(\mathcal{A})$ are *stable fixpoints*. The $\leq_i$-minimal fixpoint of the $\leq_i$-monotonic operator $S(\mathcal{A})$ is called the *well-founded* fixpoint.

Stable Operators for ndaos

Definition

Let an ndao $\mathcal{A} : \mathcal{L}^2 \to 2^{\mathcal{L}} \times 2^{\mathcal{L}}$ and some $x, y \in \mathcal{L}$ be given. Then we define:

- the *complete lower stable operator* as

$$C(\mathcal{A}_l)(y) = \{x \in \mathcal{L} \mid x \in \mathcal{A}_l(x, y) \text{ and } \neg\exists x' < y : x' \in \mathcal{A}_l(x', y)\}$$

- the *complete upper stable operator* as:

$$C(\mathcal{A}_u)(x) = \{y \in \mathcal{L} \mid y \in \mathcal{A}_u(x, y) \text{ and } \neg\exists y' < y : z \in \mathcal{A}_u(x, y')\}$$

- the *stable operator* as $S(\mathcal{A})(x, y) = C(\mathcal{A}_l)(y) \times C(\mathcal{A}_u)(x)$.
- a *stable fixpoint* of $\mathcal{A}$ as any $(x, y) \in \mathcal{L}^2$ s.t. $(x, y) \in S(\mathcal{A})(x, y)$.

Stable Operator: Example

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

$\min_{\subseteq}\{x \subseteq \mathcal{A}_{\mathcal{P}} \mid x \in \mathcal{T}_{P}(x, \emptyset)\} = \{\{p\}, \{q\}\}.$

Stable Operator: Example

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

$\min_{\subseteq}\{x \subseteq \mathcal{A}_{\mathcal{P}} \mid x \in \mathcal{T}_P(x, \emptyset)\} = \{\{p\}, \{q\}\}.$

Notice that taking the glb of fixpoints of $\mathcal{T}^l_P(., y)$ would be too weak (as we would derive neither p nor q).

Stable semantics for ndaos

Stable operators are approximation operators which give more precise approximations.

Stable semantics for ndaos

Stable operators are approximation operators which give more precise approximations.

Given an ndao $\mathcal{A}$:

- $\vee$ every stable fixpoint of $\mathcal{A}$ is a $\preceq_t^S$-minimal fixpoint of $\mathcal{A}$.
- $\times$ stable fixpoints might not exist.
- $\times$ $\leq_i$-minimal stable fixpoints might not be unique.

Stable semantics for ndaos

Stable operators are approximation operators which give more precise approximations.
Given an ndao $\mathcal{A}$:

- $\vee$ every stable fixpoint of $\mathcal{A}$ is a $\preceq_t^S$-minimal fixpoint of $\mathcal{A}$.
- $\times$ stable fixpoints might not exist.
- $\times$ $\leq_i$-minimal stable fixpoints might not be unique.

Theorem

Let a dlp $\mathcal{P}$ and a consistent $x \subseteq y \subseteq \mathcal{A}^2_{\mathcal{P}}$ be given.
Then (x, y) is a stable model of $\mathcal{P}$ iff $(x, y) \in S(\mathcal{T}_P^{cons})(x, y)$.

Well-founded state

Well-founded State of $\mathcal{A}$:

- $\times$ Convex set of elements between set of lower bounds and set of upper bounds.

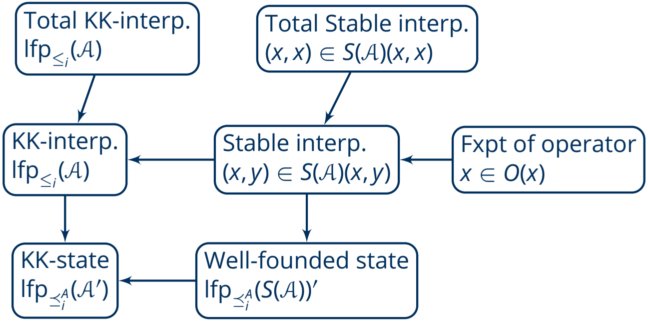
  Defined as Kripke-Kleene state of $S(\mathcal{A})$.
- $\vee$ Existence guaranteed.
- $\vee$ Unique.
- $\vee$ Iterative construction.
- $\vee$ More precise as the Kripke-Kleene state of $\mathcal{A}$.
- $\vee$ Approximates any fixpoint of $\mathcal{A}$ and O.
- $\vee$ WF($\mathcal{T}_P$) is (almost) equal to the well-founded semantics with disjunction from Joao Alcântara, Carlos Viegas Damásio and Luís Moniz Pereira. 'A well-founded semantics with disjunction'. In: *Logic Programming: 21st International Conference, ICLP 2005, Sitges, Spain, October 2-5, 2005. Proceedings 21*. Springer. 2005, pp. 341–355.

Well-founded State: Example

Let $\mathcal{P} = \{p \vee q \leftarrow \neg s; s \leftarrow r; r \leftarrow s\}$. We calculate $\mathrm{WF}(\mathcal{IC}_{\mathcal{P}})$ as follows:

- $S(\mathcal{T}_P)'(\emptyset, \{p, q\}) = \min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{\mathcal{A}_{\mathcal{P}}})\!\uparrow \times \min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{\emptyset})\!\downarrow = \{\emptyset\}\!\uparrow \times \{\{p\}, \{q\}\}\!\downarrow$.
- $S(\mathcal{T}_P)'^2(\emptyset, \{p, q\}) = (\min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{\{p\}}) \cup \min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{\{q\}}))\!\uparrow \times \min_{\subseteq} \mathrm{Mod}(\frac{\mathcal{P}}{\emptyset})\!\downarrow$ $= \{\{p\}, \{q\}\}\!\uparrow \times \{\{p\}, \{q\}\}\!\downarrow$ and thus a fixpoint is reached.

Summary


Total KK-interp.
$\mathrm{lfp}_{\leq_i}(\mathcal{A})$
Total Stable interp.
$(x, x) \in S(\mathcal{A})(x, x)$
KK-interp.
$\mathrm{lfp}_{\leq_i}(\mathcal{A})$
Stable interp.
$(x, y) \in S(\mathcal{A})(x, y)$
Fxpt of operator
$x \in O(x)$
KK-state
$\mathrm{lfp}_{\preceq_i^A}(\mathcal{A}')$
Well-founded state
$\mathrm{lfp}_{\preceq_i^A}(S(\mathcal{A}))'$

More results

- V Allows to generalize semantics for LPs with aggregates to the disjunctive case.
- V Application to conditional abstract dialectical frameworks.
- V Characterization of the semi-equilibrium semantics.
- V Choice rules
- ? Disjunctive default logic.

Bibliography I

- Alcântara, Joao, Carlos Viegas Damásio and Luís Moniz Pereira. 'A well-founded semantics with disjunction'. In: *Logic Programming: 21st International Conference, ICLP 2005, Sitges, Spain, October 2-5, 2005. Proceedings 21*. Springer. 2005, pp. 341–355.
- Alviano, Mario, Wolfgang Faber and Martin Gebser. 'Aggregate semantics for propositional answer set programs'. In: *Theory and Practice of Logic Programming* 23.1 (2023), pp. 157–194.
- Bogaerts, Bart. 'Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory'. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 2686–2693.
- Davey, B.A. and H.A. Priestley. *Introduction to Lattices and Order*. Second Edition. Cambridge University Press, 2002.

Bibliography II

- Faber, Wolfgang, Gerald Pfeifer and Nicola Leone. 'Semantics and complexity of recursive aggregates in answer set programming'. In: *Artificial Intelligence* 175.1 (2011), pp. 278–298.
- Gelfond, Michael and Yuanlin Zhang. 'Vicious circle principle and logic programs with aggregates'. In: *Theory and Practice of Logic Programming* 14.4-5 (2014), pp. 587–601.
- Kemp, David B and Peter J Stuckey. 'Semantics of Logic Programs with Aggregates.'. In: *ISLP*. Vol. 91. Citeseer. 1991, pp. 387–401.
- Liu, Lengning, Enrico Pontelli, Tran Cao Son and Miroslaw Truszczyński. 'Logic programs with abstract constraint atoms: The role of computations'. In: *Artificial Intelligence* 174.3-4 (2010), pp. 295–315.
- Mumick, Inderpal Singh, Hamid Pirahesh and Raghu Ramakrishnan. 'The magic of duplicates and aggregates'. In: *Proceedings of the 16th International Conference on Very Large Data Bases*. 1990, pp. 264–277.

Bibliography III

- Pelov, Nikolay, Marc Denecker and Maurice Bruynooghe. 'Well-founded and stable semantics of logic programs with aggregates'. In: *Theory and Practice of Logic Programming* 7.3 (2007), pp. 301–353.
- Pelov, Nikolay and Miroslaw Truszczynski. 'Semantics of disjunctive programs with monotone aggregates-an operator-based approach.'. In: *NMR*. Vol. 2004. Citeseer. 2004, pp. 327–334.
- Ross, Kenneth A. 'Modular stratification and magic sets for DATALOG programs with negation'. In: *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1990, pp. 161–171.