Foundations of logic programming semantics: an operator-based perspective

Jesse Heyninck

May 28, 2024

Open Universiteit, the Netherlands
University of Cape Town, South Africa

Logic Programming

- 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.

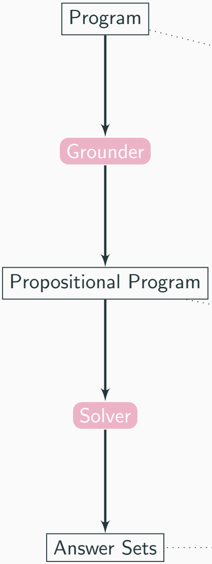
Example Applications: Student Projects

- Puzzles and games:
    - Rush hour
    - Rubics
    - Futoshiki
    - Kakurasu
    - IQ Puzzler Pro
- Generating healthy diets.
- Procedural content generation.
- Parsing grammatical structure of Latin.
- Minimum Sum Partition Problem.

Example Applications: Knowledge Representation

- Solvers or reasoners for:
    - Formal argumentation [DRWW20]
    - AGM Belief Revision [AP17, BNBPW04]
    - Boolean networks [KB19]
    - Ordered disjunction [BNS02]
    - Description Logics [Swi04]
- Inconsistency Measures [KT21]
- Linear temporal logic [KCG23]
- Logic programming (!?) [KRSW23]
- Axiom pinpointing in ontologies [HMP+23]
- ...

The ASP-workflow



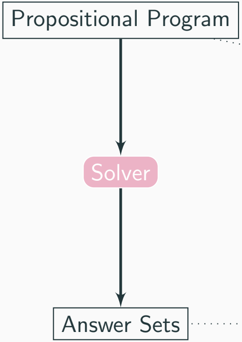
penguin(tweety).
bird(zazu).
bird(X)← penguin(X).
flies(X)← bird(X), ¬ ab(X).
ab(X)← penguin(X).

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).

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

The ASP-workflow

Focus of today:



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).

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

What are answer sets and what is so special about them?

Goals and Structure

- Provide a gentle introduction to the semantics of logic programming:
  - Supported models
  - Kripke-Kleene models
  - Stable models
  - Well-founded model
- Illustrate the operator-based approach to KR with a paradigmatic example.
  - Basic constructions of approximation fixpoint theory (for logic programs).
  - From logic programming to operators.
- Tutorial and invitation to AFT.

Almost nothing of this is my work

- Operator-based approach has driven logic programming since its inception [VGRS91, Fit06].
- Studied algebraically by Denecker, Marek and Truszczyński [DMT00].
- I extended and worked in this algebraic framework with Ofer Arieli and Bart Bogaerts, .

Goals and Structure

Syntax of Logic Programs

Semantics of Positive Programs

Semantics of Normal Logic Programs

Stable Semantics

Approximation Fixpoint Theory

- Operator-Based Studies: Modularity

Round up

Syntax of Logic Programs

Syntax of Logic Programs

Set of atoms $\mathcal{A} = \{a, b, c, p, q, r, a_1, a_2, \ldots\}$

$$a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m$$

- Program is a set of rules.
- Rule is positive if $m = 0$.
- Program is positive if all the rules are positive.

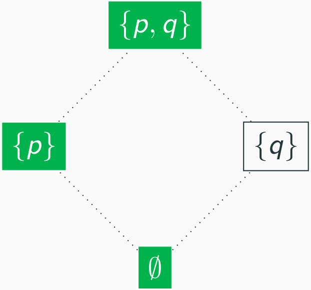
Semantics of Positive Programs

What are the semantics of logic programs?

$$p \leftarrow q.$$

Classical models? $\emptyset, \{p\}, \{p, q\}$.

Notice: a formula follows from every classical model if it follows from the minimal model $\emptyset$.



What are the semantics of logic programs?

$$p \leftarrow q., \quad q \leftarrow .$$

What are the semantics of logic programs?

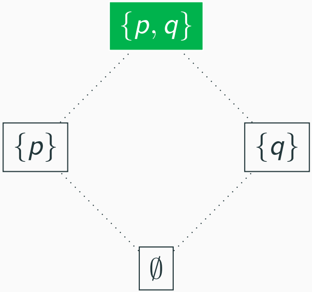
$$p \leftarrow q., \quad q \leftarrow .$$

Classical models? $\emptyset, \{p\}, \{p, q\}$.

What are the semantics of logic programs?

$$p \leftarrow q., \quad q \leftarrow .$$

Classical models? $\emptyset, \{p\}, \{p, q\}$.



$T_{\mathcal{P}}$-operator

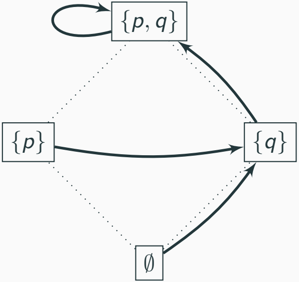
$$T_{\mathcal{P}} : \wp(\mathcal{A}_{\mathcal{P}}) \mapsto \wp(\mathcal{A}_{\mathcal{P}})$$

$$T_{\mathcal{P}}(x) = \{a \mid a \leftarrow b_1, \ldots, b_n \in \mathcal{P} \text{ and } b_1, \ldots, b_n \in x\}$$

Example

$$\mathcal{P} = \{p \leftarrow q., \quad q \leftarrow .\}$$

x	$\emptyset$	$\{p\}$	$\{q\}$	$\{p, q\}$
$T_{\mathcal{P}}(x)$	$\{q\}$	$\{q\}$	$\{p, q\}$	$\{p, q\}$



Models and Fixpoints

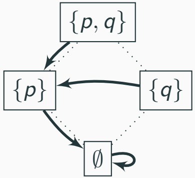
Definition
x is a post-fixpoint of $T_{\mathcal{P}}$ if $T_{\mathcal{P}}(x) \subseteq x$.

Intuition: everything I can derive from x using $\mathcal{P}$ is in x.

Models of $\mathcal{P}$.

Example

$$\mathcal{P} = \{p \leftarrow q.\}$$



Models and Fixpoints

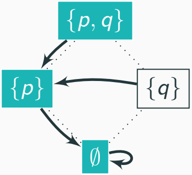
Definition
x is a post-fixpoint of $T_{\mathcal{P}}$ if $T_{\mathcal{P}}(x) \subseteq x$.

Intuition: everything I can derive from x using $\mathcal{P}$ is in x.

Models of $\mathcal{P}$.

Example

$$\mathcal{P} = \{p \leftarrow q.\}$$



Models and Fixpoints

Definition
x is a post-fixpoint of $T_{\mathcal{P}}$ if $T_{\mathcal{P}}(x) \subseteq x$.

Intuition: everything I can derive from x using $\mathcal{P}$ is in x.

Models of $\mathcal{P}$.

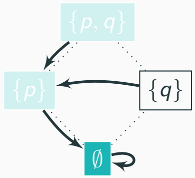
Definition
x is a fixpoint of $T_{\mathcal{P}}$ if $T_{\mathcal{P}}(x) = x$.

Intuition: same, plus everything in x is derivable.

Supported models of $\mathcal{P}$.

Example

$$\mathcal{P} = \{p \leftarrow q.\}$$



Models and Fixpoints

Definition
x is a post-fixpoint of $T_{\mathcal{P}}$ if $T_{\mathcal{P}}(x) \subseteq x$.

Intuition: everything I can derive from x using $\mathcal{P}$ is in x.

Models of $\mathcal{P}$.

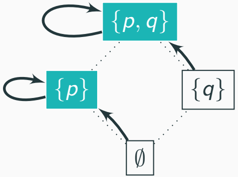
Definition
x is a fixpoint of $T_{\mathcal{P}}$ if $T_{\mathcal{P}}(x) = x$.

Intuition: same, plus everything in x is derivable.

Supported models of $\mathcal{P}$.

Example

$$\mathcal{P} = \{p \leftarrow ., \quad q \leftarrow q.\}$$



Models and Fixpoints

- For positive programs, $T_{\mathcal{P}}$ has a unique least fixpoint x.
- It is also the least pre-fixpoint.
- We can compute it by iterating $T_{\mathcal{P}}$ starting from $\emptyset$:
  $T_{\mathcal{P}}(\ldots T_{\mathcal{P}}(\emptyset)\ldots) = \bigcup_{i \geq 0} T^{i}_{\mathcal{P}}(\emptyset)$
  And this is possible in polynomial time.

Models and Fixpoints

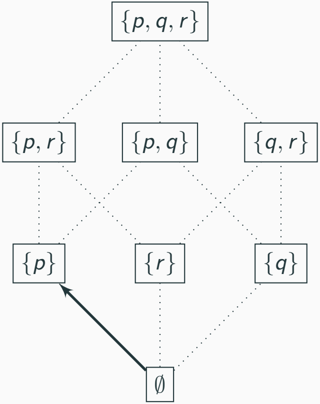
- For positive programs, $T_{\mathcal{P}}$ has a unique least fixpoint x.
  - Any fixpoint y of $T_{\mathcal{P}}$ will be a superset: $x \subseteq y$.
- It is also the least pre-fixpoint.
  - Any pre-fixpoint y of $T_{\mathcal{P}}$ will be a superset: $x \subseteq y$.
  - $\mathcal{P}$ has a least model,
    i.e. any model of $\mathcal{P}$ includes x.
  - If something follows from every model of $\mathcal{P}$, it follows from x.
- We can compute it by iterating $T_{\mathcal{P}}$ starting from $\emptyset$:
  $T_{\mathcal{P}}(\ldots T_{\mathcal{P}}(\emptyset) \ldots)$
  And this is possible in polynomial time.

Models and Fixpoints

- For positive programs, $T_{\mathcal{P}}$ has a unique least fixpoint x.
  - Any fixpoint y of $T_{\mathcal{P}}$ will be a superset: $x \subseteq y$.
- It is also the least pre-fixpoint.
  - Any pre-fixpoint y of $T_{\mathcal{P}}$ will be a superset: $x \subseteq y$.
  - $\mathcal{P}$ has a least model,
    i.e. any model of $\mathcal{P}$ includes x.
  - If something follows from every model of $\mathcal{P}$, it follows from x.
- We can compute it by iterating $T_{\mathcal{P}}$ starting from $\emptyset$:
  $T_{\mathcal{P}}(\dots T_{\mathcal{P}}(\emptyset)\dots)$
  And this is possible in polynomial time.
  Underlying result: A $\subseteq$-monotonic operator over a complete lattice admits a least fixpoint.

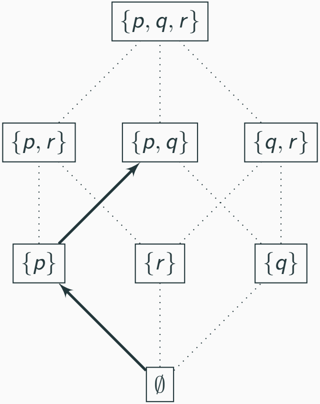
Least fixpoint computation

$$\mathcal{P} = \{p \leftarrow . \quad q \leftarrow p. \quad r \leftarrow p, q.\}$$



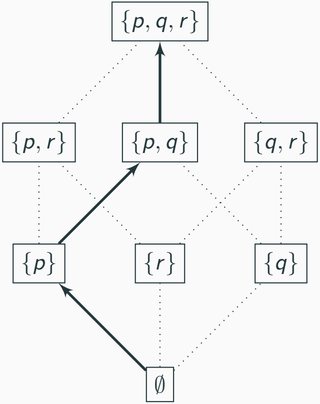
Least fixpoint computation

$$\mathcal{P} = \{p \leftarrow . \quad q \leftarrow p. \quad r \leftarrow p, q.\}$$



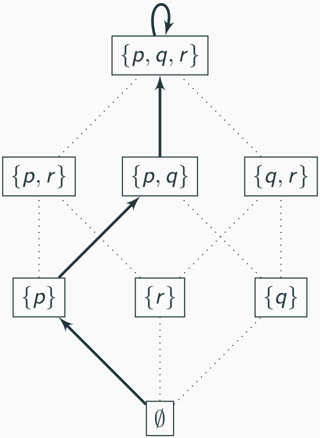
Least fixpoint computation

$$\mathcal{P} = \{p \leftarrow . \quad q \leftarrow p. \quad r \leftarrow p, q.\}$$



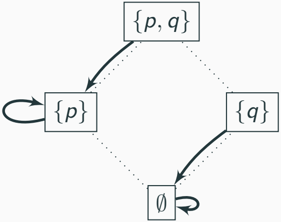
Least fixpoint computation

$$\mathcal{P} = \{p \leftarrow . \quad q \leftarrow p. \quad r \leftarrow p, q.\}$$



Least fixpoint $\neq$ unique fixpoint

Example ($\mathcal{P} = \{p \leftarrow p.\}$)



Semantics of Normal Logic Programs

Enters Negation

$$p \leftarrow \neg q$$

How to extend our operator?

Enters Negation

$$p \leftarrow \neg q$$

How to extend our operator?

Easy:

$$\begin{aligned} T_{\mathcal{P}}(x) = \{a \mid \quad & a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}, \text{ and} \\ & b_1, \ldots, b_n \in x, \text{ and } c_1, \ldots, c_m \notin x\} \end{aligned}$$

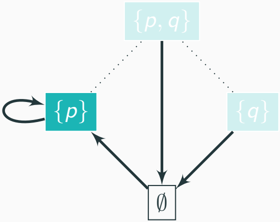
Enters Negation

$$p \leftarrow \neg q$$

How to extend our operator?

Easy:

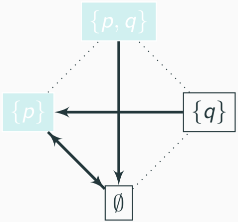
$$T_{\mathcal{P}}(x) = \{a \mid \quad a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}, \text{ and } \\ b_1, \ldots, b_n \in x, \text{ and } c_1, \ldots, c_m \notin x\}$$



Great, thanks for your attention

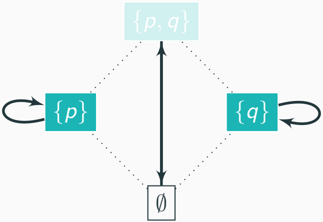
Great, thanks for your attention

$\mathcal{P} = \{p \leftarrow \neg p\}$



No unique fixpoint

$\mathcal{P} = \{p \leftarrow \neg q; q \leftarrow \neg p\}$



Problems with negation

- There might not be a fixpoint.
- There might be multiple minimal fixpoints.
- We don't know how to find fixpoints.

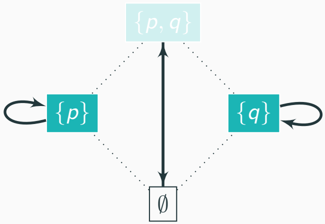
Problems with negation

- There might not be a fixpoint.
- There might be multiple minimal fixpoints.
- We don't know how to find fixpoints.
- Anyone sees what went wrong with our operator?

Problems with negation

- There might not be a fixpoint.
- There might be multiple minimal fixpoints.
- We don't know how to find fixpoints.
- Anyone sees what went wrong with our operator?

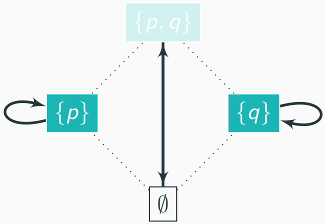
$\Rightarrow$ It is not a monotonic operator



Problems with negation

- There might not be a fixpoint.
- There might be multiple minimal fixpoints.
- We don't know how to find fixpoints.
- Anyone sees what went wrong with our operator?

$\Rightarrow$ It is not a monotonic operator



Solution: approximations of operators that are $\leq_i$-monotonic.

Commercial break

November 2-4, 2024, Hanoi, Vietnam
Co-located with KR 2024

- https://nmr.krportal.org/2024
- Submissions can be:
  - New material
  - Previously published papers
- Paper submission deadline: July 12th, 2024

Approximations

- Pairs of sets of atoms (x, y).
  - x contains all atoms that are definitely true.
  - y contains all atoms that are possibly true.

Approximations

- Pairs of sets of atoms (x, y).
  - x contains all atoms that are definitely true.
  - y contains all atoms that are possibly true.
- How to compare such pairs of sets?
  - $(x_1, y_1) \leq_t (x_2, y_2)$ if $x_1 \subseteq x_2$ and $y_1 \subseteq y_2$.
  - $(x_1, y_1) \leq_i (x_2, y_2)$ if $x_1 \subseteq x_2$ and $y_2 \subseteq y_1$.

Approximations

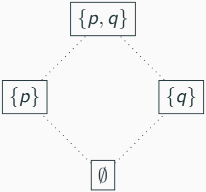
- Pairs of sets of atoms (x, y).
  - x contains all atoms that are definitely true.
  - y contains all atoms that are possibly true.
- How to compare such pairs of sets?
  - $(x_1, y_1) \leq_t (x_2, y_2)$ if $x_1 \subseteq x_2$ and $y_1 \subseteq y_2$.
  - $(x_1, y_1) \leq_i (x_2, y_2)$ if $x_1 \subseteq x_2$ and $y_2 \subseteq y_1$.

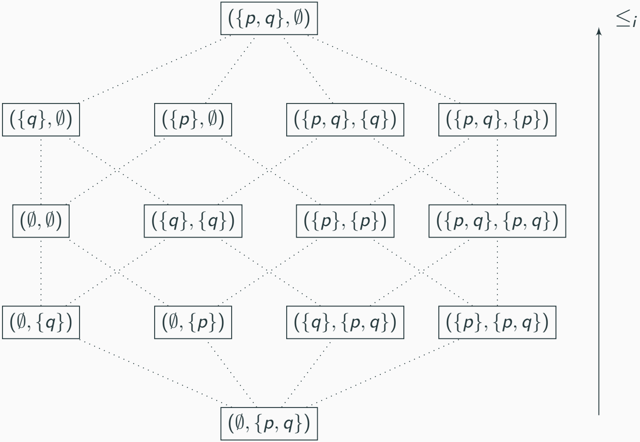
Example

$(\{p\}, \{p, q\})$: *p is true and q can be true.*

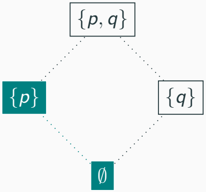
$(\{p\}, \{p\}) \leq_t (\{p\}, \{p, q\})$
$(\{p\}, \{p, q\}) \leq_i (\{p\}, \{p\})$

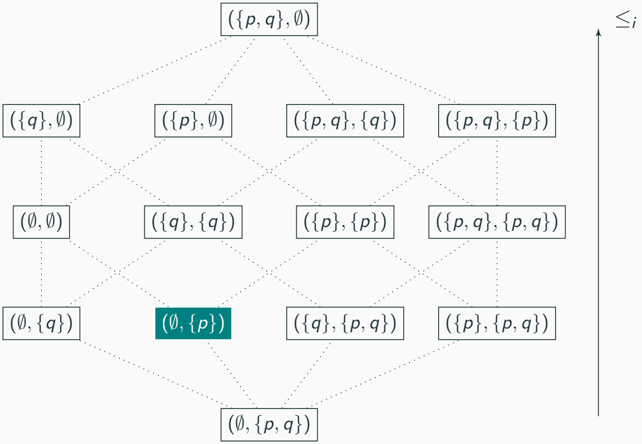
Graphical Depiction



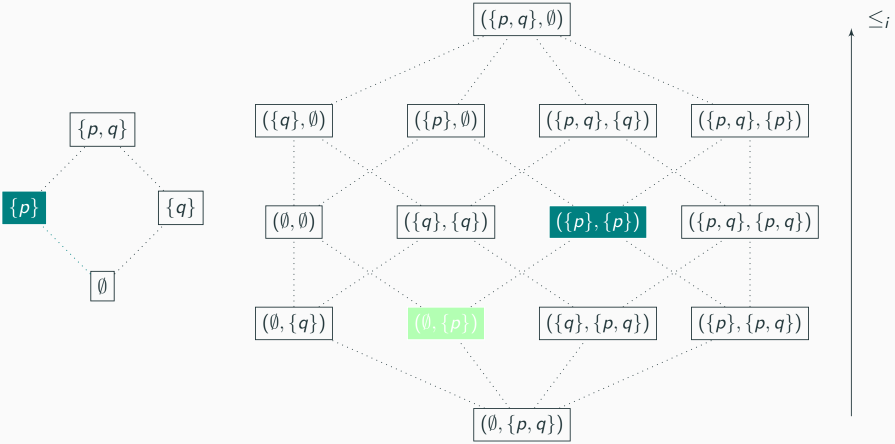


Graphical Depiction

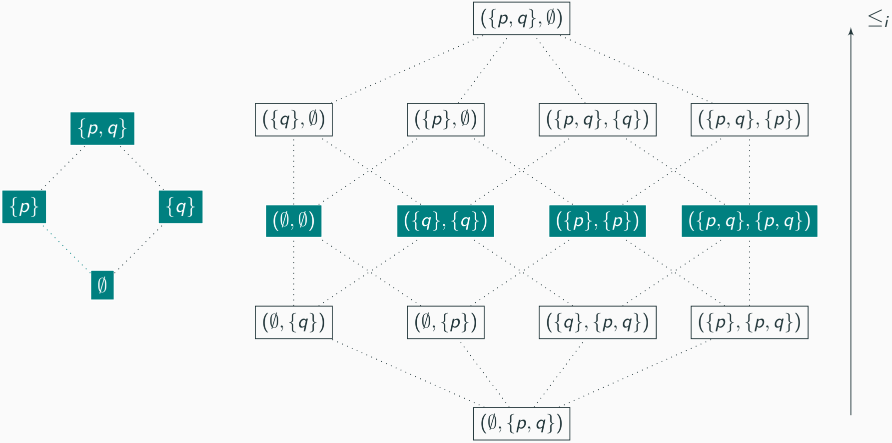




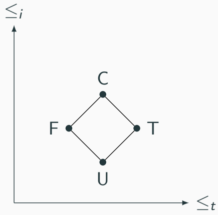
Graphical Depiction



Graphical Depiction



Approximations as Four-Valued Interpretations



- $-\mathsf{F} = \mathsf{T}$, $-\mathsf{T} = \mathsf{F}$, $-\mathsf{U} = \mathsf{U}$ and $-\mathsf{C} = \mathsf{C}$
- $(x, y)(p) = \begin{cases} \mathsf{T} & \text{if } p \in x \text{ and } p \in y, \\ \mathsf{U} & \text{if } p \notin x \text{ and } p \in y, \\ \mathsf{F} & \text{if } p \notin x \text{ and } p \notin y, \\ \mathsf{C} & \text{if } p \in x \text{ and } p \notin y. \end{cases}$
- $(x, y)(\neg\phi) = -(x, y)(\phi)$,
- $(x, y)(\psi \wedge \phi) = lub_{\leq_t}\{(x, y)(\phi), (x, y)(\psi)\}$,
- $(x, y)(\psi \vee \phi) = glb_{\leq_t}\{(x, y)(\phi), (x, y)(\psi)\}$.

Example

$(\{p\}, \{p, q\})(p) = \mathsf{T} \quad (\{p\}, \{p, q\})(q) = \mathsf{U} \quad (\{p\}, \{p, q\})(r) = \mathsf{F}$.

$(\{p\}, \{p, q\})(\neg p) = \mathsf{F} \quad (\{p\}, \{p, q\})(\neg q) = \mathsf{U}$

$(\{p\}, \{p, q\})(p \wedge q) = \mathsf{U} \quad (\{p\}, \{p, q\})(q \vee r) = \mathsf{U}$

Approximating $T_{\mathcal{P}}$ (from below)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

Approximating $T_{\mathcal{P}}$ (from below)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

$$\mathcal{IC}^{l}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$b_1, \ldots, b_n \in x \text{ and } c_1, \ldots, c_m \notin y\}$$

Approximating $T_{\mathcal{P}}$ (from below)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

$$\mathcal{IC}^{l}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$b_1, \ldots, b_n \in x \text{ and } c_1, \ldots, c_m \notin y\}$$

or, equivalently

$$\mathcal{IC}^{l}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$(x, y)(b_1 \wedge \ldots \ldots b_n \wedge \neg c_1 \wedge \ldots \wedge \neg c_m) \in \{\mathsf{T}, \mathsf{C}\}\}$$

Approximating $T_{\mathcal{P}}$ (from below)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

$$\mathcal{IC}^{l}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$b_1, \ldots, b_n \in x \text{ and } c_1, \ldots, c_m \notin y\}$$

or, equivalently

$$\mathcal{IC}^{l}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$(x, y)(b_1 \wedge \ldots \ldots b_n \wedge \neg c_1 \wedge \ldots \wedge \neg c_m) \in \{\mathsf{T}, \mathsf{C}\}\}$$

Example ($\{p \leftarrow p, \neg q\}$)
$\mathcal{IC}^{l}_{\mathcal{P}}(\{p\}, \{p, q\}) = \emptyset$
$\mathcal{IC}^{l}_{\mathcal{P}}(\{p\}, \{p\}) = \{p\}$

Approximating $T_{\mathcal{P}}$ (from above)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

Approximating $T_{\mathcal{P}}$ (from above)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

$$\mathcal{IC}^{u}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}, \\ b_1, \ldots, b_n \in y \text{ and } c_1, \ldots, c_m \notin x\}$$

Approximating $T_{\mathcal{P}}$ (from above)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

$$\mathcal{IC}^{u}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}, \\ b_1, \ldots, b_n \in y \text{ and } c_1, \ldots, c_m \notin x\}$$

or, equivalently

$$\mathcal{IC}^{u}_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}, \\ (x, y)(b_1 \wedge \ldots \ldots b_n \wedge \neg c_1 \wedge \ldots \wedge \neg c_m) \in \{\mathsf{U}, \mathsf{T}\}\}$$

Approximating $T_{\mathcal{P}}$ (from above)

$$\mathcal{IC}_{\mathcal{P}} : \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \times \mathcal{A}$$

We input an approximation and output an approximation.

$$\mathcal{IC}^u_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$b_1, \ldots, b_n \in y \text{ and } c_1, \ldots, c_m \notin x\}$$

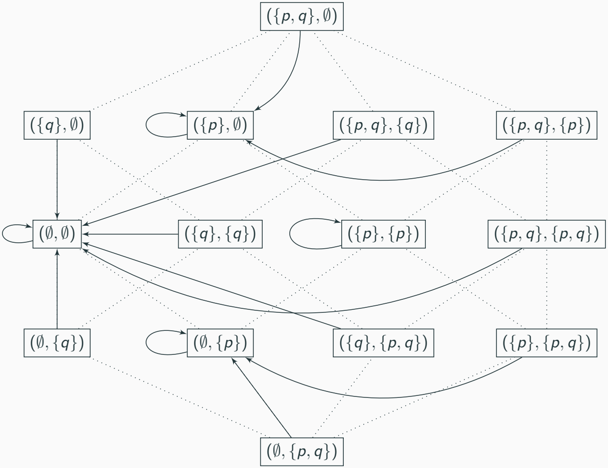
or, equivalently

$$\mathcal{IC}^u_{\mathcal{P}}(x, y) = \{a \in \mathcal{A} \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P},$$
$$(x, y)(b_1 \wedge \ldots \ldots b_n \wedge \neg c_1 \wedge \ldots \wedge \neg c_m) \in \{\mathsf{U}, \mathsf{T}\}\}$$

Example ($\{p \leftarrow p, \neg q\}$)
$\mathcal{IC}^u_{\mathcal{P}}(\{p\}, \{p, q\}) = \{p\}$
$\mathcal{IC}^u_{\mathcal{P}}(\{p\}, \{p\}) = \{p\}$

The Approximation Operator $\mathcal{IC}_\mathcal{P}$ (for $\mathcal{P} = \{p \leftarrow p, \neg q\}$)

$$\mathcal{IC}_\mathcal{P}(x, y) = (\mathcal{IC}^l_\mathcal{P}(x, y), \mathcal{IC}^u_\mathcal{P}(x, y))$$



Properties of $\mathcal{IC}_{\mathcal{P}}$

- $\mathcal{IC}_{\mathcal{P}}$ approximates $T_{\mathcal{P}}$:

$\mathcal{IC}_{\mathcal{P}}(x, x) = (T_{\mathcal{P}}(x), T_{\mathcal{P}}(x))$ for any $x \subseteq \mathcal{A}$.

Properties of $\mathcal{IC}_{\mathcal{P}}$

- $\mathcal{IC}_{\mathcal{P}}$ approximates $T_{\mathcal{P}}$:

$\mathcal{IC}_{\mathcal{P}}(x, x) = (T_{\mathcal{P}}(x), T_{\mathcal{P}}(x))$ for any $x \subseteq \mathcal{A}$.

- $\mathcal{IC}_{\mathcal{P}}$ is $\leq_i$-monotonic:

if $(x_1, y_1) \leq_i (x_2, y_2)$ then $\mathcal{IC}_{\mathcal{P}}(x_1, y_1) \leq_i \mathcal{IC}_{\mathcal{P}}(x_2, y_2)$.

Properties of $\mathcal{IC}_{\mathcal{P}}$

- $\mathcal{IC}_{\mathcal{P}}$ approximates $T_{\mathcal{P}}$:

$\mathcal{IC}_{\mathcal{P}}(x,x) = (T_{\mathcal{P}}(x), T_{\mathcal{P}}(x))$ for any $x \subseteq \mathcal{A}$.

- $\mathcal{IC}_{\mathcal{P}}$ is $\leq_i$-monotonic:

if $(x_1, y_1) \leq_i (x_2, y_2)$ then $\mathcal{IC}_{\mathcal{P}}(x_1, y_1) \leq_i \mathcal{IC}_{\mathcal{P}}(x_2, y_2)$.

We say $\mathcal{IC}_{\mathcal{P}}$ is an approximation operator. It is also symmetric, in the sense that $\mathcal{IC}_{\mathcal{P}}(x,y) = (\mathcal{IC}^l_{\mathcal{P}}(x,y), \mathcal{IC}^l_{\mathcal{P}}(y,x))$.

Kripke-Kleene Fixpoint

The $\leq_i$-monotonicity is our *indulgentia* back into Tarski's heaven:

Proposition
$\mathcal{IC}_\mathcal{P}$ *has a least fixpoint, obtainable as* $\bigsqcup_{i\geq 0} \mathcal{IC}^i_\mathcal{P}(\emptyset, \mathcal{A})$[3].

[3] $(x_1, y_1) \sqcup (x_2, y_2) = (x_1 \cup x_2, y_1 \cap y_2)$.

Kripke-Kleene Fixpoint

The $\leq_i$-monotonicity is our *indulgentia* back into Tarski's heaven:

Proposition
$\mathcal{IC}_{\mathcal{P}}$ has a least fixpoint, obtainable as $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$[3].

$\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$ is called the Kripke-Kleene Fixpoint

[3] $(x_1, y_1) \sqcup (x_2, y_2) = (x_1 \cup x_2, y_1 \cap y_2)$.

Kripke-Kleene Fixpoint

The $\leq_i$-monotonicity is our *indulgentia* back into Tarski's heaven:

Proposition
$\mathcal{IC}_{\mathcal{P}}$ has a least fixpoint, obtainable as $\bigsqcup_{i\geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$[3].

$\bigsqcup_{i\geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$ is called the Kripke-Kleene Fixpoint

Proposition
For any fixpoint of $x = T_{\mathcal{P}}(x)$, $\bigsqcup_{i\geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A}) \leq_i (x, x)$.

[3] $(x_1, y_1) \sqcup (x_2, y_2) = (x_1 \cup x_2, y_1 \cap y_2)$.

Kripke-Kleene Fixpoint

The $\leq_i$-monotonicity is our *indulgentia* back into Tarski's heaven:

Proposition
$\mathcal{IC}_{\mathcal{P}}$ has a least fixpoint, obtainable as $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$[3].

$\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$ is called the Kripke-Kleene Fixpoint

Proposition
For any fixpoint of $x = T_{\mathcal{P}}(x)$, $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A}) \leq_i (x, x)$.

Proposition
$\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$ is consistent (i.e. where $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A}) = (x, y)$, $x \subseteq y$).

[3] $(x_1, y_1) \sqcup (x_2, y_2) = (x_1 \cup x_2, y_1 \cap y_2)$.

Kripke-Kleene Fixpoint

The $\leq_i$-monotonicity is our *indulgentia* back into Tarski's heaven:

Proposition
$\mathcal{IC}_{\mathcal{P}}$ has a least fixpoint, obtainable as $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$[3].

$\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$ is called the Kripke-Kleene Fixpoint

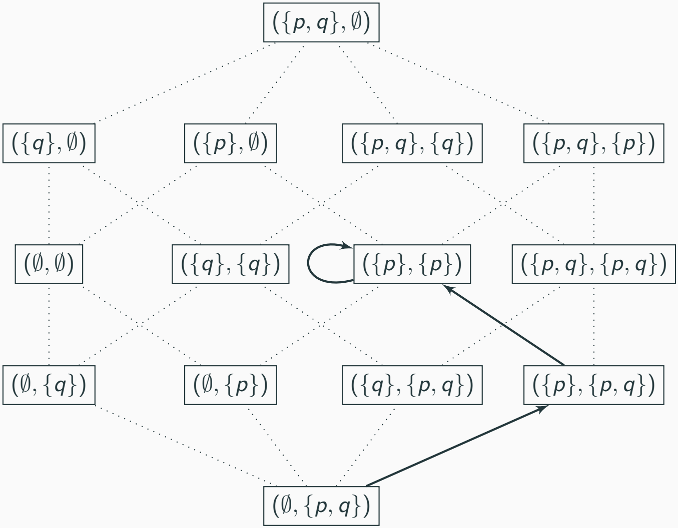
Proposition
For any fixpoint of $x = T_{\mathcal{P}}(x)$, $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A}) \leq_i (x, x)$.

Proposition
$\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A})$ is consistent (i.e. where $\bigsqcup_{i \geq 0} \mathcal{IC}^i_{\mathcal{P}}(\emptyset, \mathcal{A}) = (x, y)$, $x \subseteq y$).

If $(x, y) = \mathcal{IC}_{\mathcal{P}}(x, y)$ then we call it a partial supported model.

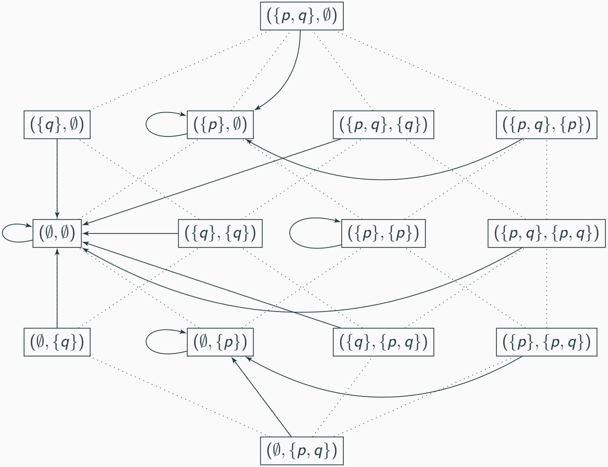
[3] $(x_1, y_1) \sqcup (x_2, y_2) = (x_1 \cup x_2, y_1 \cap y_2)$.

Example: $\mathcal{P} = \{p \leftarrow; q \leftarrow \neg p\}$



The Approximation Operator $\mathcal{IC}_{\mathcal{P}}$ (for $\mathcal{P} = \{p \leftarrow p, \neg q\}$)

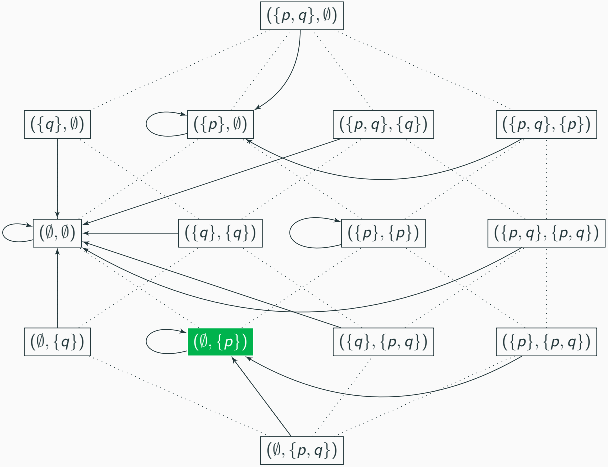
$$\mathcal{IC}_{\mathcal{P}}(x, y) = (\mathcal{IC}^{l}_{\mathcal{P}}(x, y), \mathcal{IC}^{u}_{\mathcal{P}}(x, y))$$



The Approximation Operator $\mathcal{IC}_\mathcal{P}$ (for $\mathcal{P} = \{p \leftarrow p, \neg q\}$)

$$\mathcal{IC}_\mathcal{P}(x, y) = (\mathcal{IC}^l_\mathcal{P}(x, y), \mathcal{IC}^u_\mathcal{P}(x, y))$$

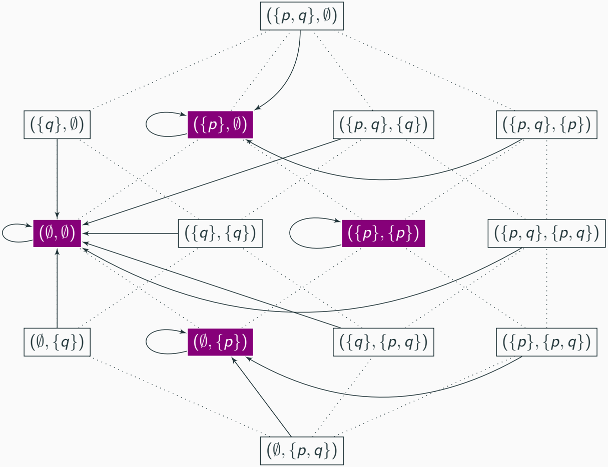
Kripke-Kleene Fixpoint



The Approximation Operator $\mathcal{IC}_\mathcal{P}$ (for $\mathcal{P} = \{p \leftarrow p, \neg q\}$)

$$\mathcal{IC}_\mathcal{P}(x, y) = (\mathcal{IC}^l_\mathcal{P}(x, y), \mathcal{IC}^u_\mathcal{P}(x, y))$$

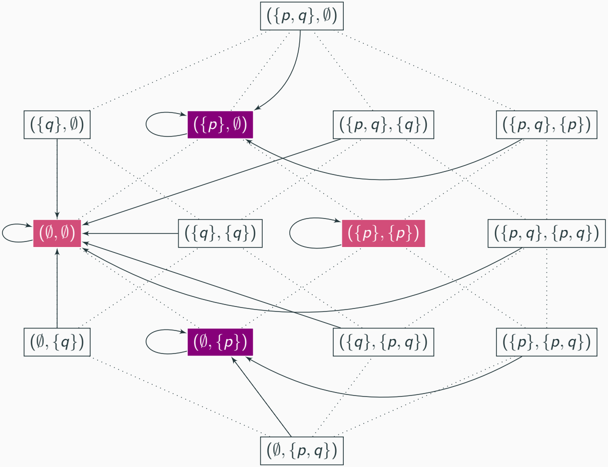
Partial Supported models



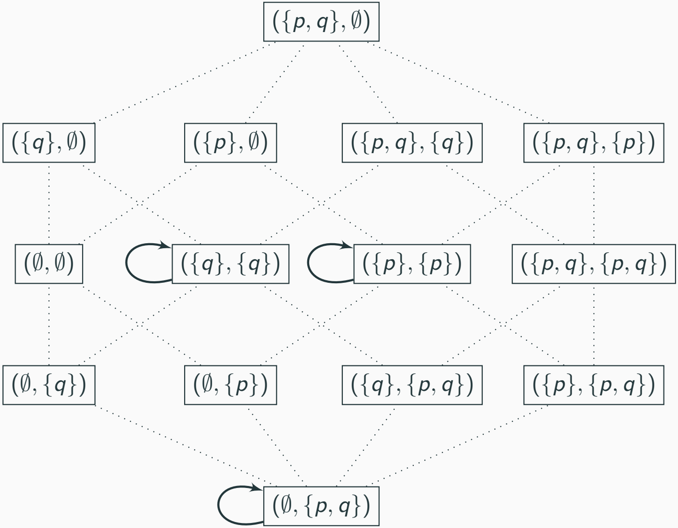
The Approximation Operator $\mathcal{IC}_{\mathcal{P}}$ (for $\mathcal{P} = \{p \leftarrow p, \neg q\}$)

$$\mathcal{IC}_{\mathcal{P}}(x, y) = (\mathcal{IC}^{l}_{\mathcal{P}}(x, y), \mathcal{IC}^{u}_{\mathcal{P}}(x, y))$$

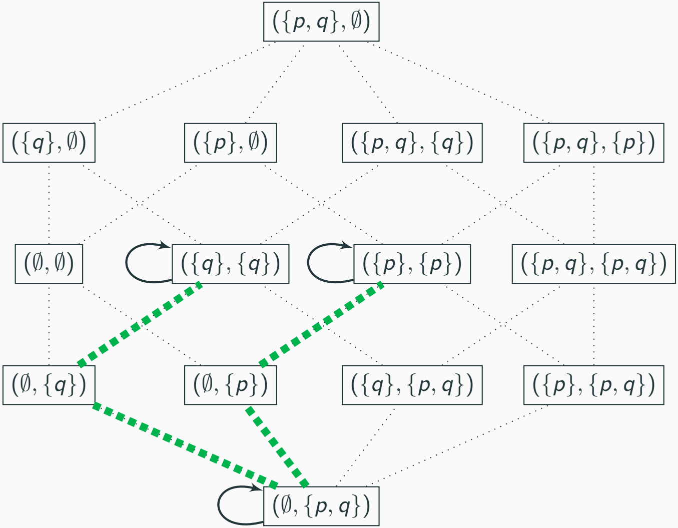
Supported models



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



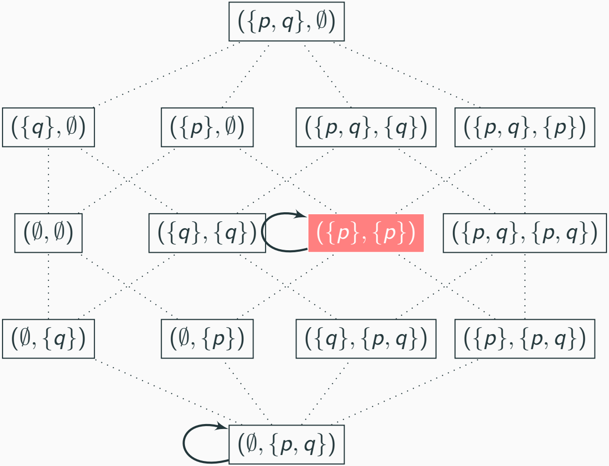
Example: $\mathcal{P} = \{p \leftarrow \neg q; q \leftarrow \neg p\}$



Stable Semantics

Example: Kripke-Kleene is rather weak

$$p \leftarrow \neg q;\ q \leftarrow q$$



Example: Kripke-Kleene is rather weak

$$\mathcal{P} = \{p \leftarrow \neg q;\ q \leftarrow q\}$$

Construction of the Kripke-Kleene fixpoint:

- $\mathcal{IC}_{\mathcal{P}}(\emptyset, \{p, q\}) = (\emptyset, \{p, q\})$.
- Fixpoint reached.

Can't get rid of the self-supporting atom q in the upper bound.

Example: Kripke-Kleene is rather weak

$$\mathcal{P} = \{p \leftarrow \neg q;\ q \leftarrow q\}$$

Construction of the Kripke-Kleene fixpoint:

- $\mathcal{IC}_{\mathcal{P}}(\emptyset, \{p, q\}) = (\emptyset, \{p, q\})$.
- Fixpoint reached.

Can't get rid of the self-supporting atom q in the upper bound.

Assuming that no atom is certainly true, construct the smallest upper bound possible:

$\mathcal{IC}^{u}_{\mathcal{P}}(\emptyset, \emptyset) = \{p\}$

$\mathcal{IC}^{u}_{\mathcal{P}}(\emptyset, \{p\}) = \{p\}$

Example: Kripke-Kleene is rather weak

$$\mathcal{P} = \{p \leftarrow \neg q;\ q \leftarrow q\}$$

Construction of the Kripke-Kleene fixpoint:

- $\mathcal{IC}_{\mathcal{P}}(\emptyset, \{p, q\}) = (\emptyset, \{p, q\})$.
- Fixpoint reached.

Can't get rid of the self-supporting atom q in the upper bound.

Assuming that no atom is certainly true, construct the smallest upper bound possible:

$\mathcal{IC}^{u}_{\mathcal{P}}(\emptyset, \emptyset) = \{p\}$

$\mathcal{IC}^{u}_{\mathcal{P}}(\emptyset, \{p\}) = \{p\}$

As $\mathcal{IC}^{u}_{\mathcal{P}}(\emptyset, \cdot)$ is a $\subseteq$-monotonic operator, it admits a least fixed point.

Stable Operator

$$S(\mathcal{IC}^{l}_{\mathcal{P}})(y) = lfp(\mathcal{IC}^{l}_{\mathcal{P}}(\cdot, y))$$

Stable Operator

$$S(\mathcal{IC}^{l}_{\mathcal{P}})(y) = lfp(\mathcal{IC}^{l}_{\mathcal{P}}(\cdot, y))$$

Example ($\{p \leftarrow \neg q;\ q \leftarrow q\}$)

$S(\mathcal{IC}^{l}_{\mathcal{P}})(\{p, q\}) = \emptyset$ *since:*

$\mathcal{IC}^{l}_{\mathcal{P}}(\emptyset, \{p, q\}) = \emptyset$: *fixpoint reached.*

Stable Operator

$$S(\mathcal{IC}^l_\mathcal{P})(y) = lfp(\mathcal{IC}^l_\mathcal{P}(\cdot, y))$$

$$S(\mathcal{IC}^u_\mathcal{P})(x) = lfp(\mathcal{IC}^u_\mathcal{P}(x, \cdot)) = lfp(\mathcal{IC}^l_\mathcal{P}(\cdot, x))$$

Example (${p \leftarrow \neg q;\ q \leftarrow q}$)

$S(\mathcal{IC}^l_\mathcal{P})(\{p, q\}) = \emptyset$ *since:*

$\mathcal{IC}^l_\mathcal{P}(\emptyset, \{p, q\}) = \emptyset$*: fixpoint reached.*

Stable Operator

$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = lfp(\mathcal{IC}^l_{\mathcal{P}}(\cdot, y))$$

$$S(\mathcal{IC}^u_{\mathcal{P}})(x) = lfp(\mathcal{IC}^u_{\mathcal{P}}(x, \cdot)) = lfp(\mathcal{IC}^l_{\mathcal{P}}(\cdot, x))$$

Example **(**$\{p \leftarrow \neg q;\ q \leftarrow q\}$**)**

$S(\mathcal{IC}^l_{\mathcal{P}})(\{p, q\}) = \emptyset$ *since:*
$\mathcal{IC}^l_{\mathcal{P}}(\emptyset, \{p, q\}) = \emptyset$*: fixpoint reached.*

$S(\mathcal{IC}^u_{\mathcal{P}})(\emptyset) = \{p\}$ *since:*
$\mathcal{IC}^u_{\mathcal{P}}(\emptyset, \emptyset) = \{p\}$
$\mathcal{IC}^u_{\mathcal{P}}(\emptyset, \{p\}) = \{p\}$*: fixpoint reached.*

Stable Operator

$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = lfp(\mathcal{IC}^l_{\mathcal{P}}(\cdot, y))$$

$$S(\mathcal{IC}^u_{\mathcal{P}})(x) = lfp(\mathcal{IC}^u_{\mathcal{P}}(x, \cdot)) = lfp(\mathcal{IC}^l_{\mathcal{P}}(\cdot, x))$$

$$S(\mathcal{IC}_{\mathcal{P}})(x, y) = (S(\mathcal{IC}^l_{\mathcal{P}})(y),\ S(\mathcal{IC}^u_{\mathcal{P}})(x))$$

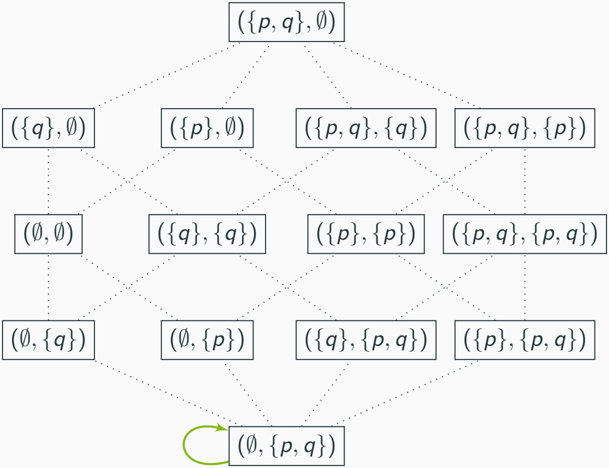
Example ($\{p \leftarrow \neg q;\ q \leftarrow q\}$)

$S(\mathcal{IC}^l_{\mathcal{P}})(\{p, q\}) = \emptyset$ *since:*
$\mathcal{IC}^l_{\mathcal{P}}(\emptyset, \{p, q\}) = \emptyset$*: fixpoint reached.*

$S(\mathcal{IC}^u_{\mathcal{P}})(\emptyset) = \{p\}$ *since:*
$\mathcal{IC}^u_{\mathcal{P}}(\emptyset, \emptyset) = \{p\}$
$\mathcal{IC}^u_{\mathcal{P}}(\emptyset, \{p\}) = \{p\}$*: fixpoint reached.*

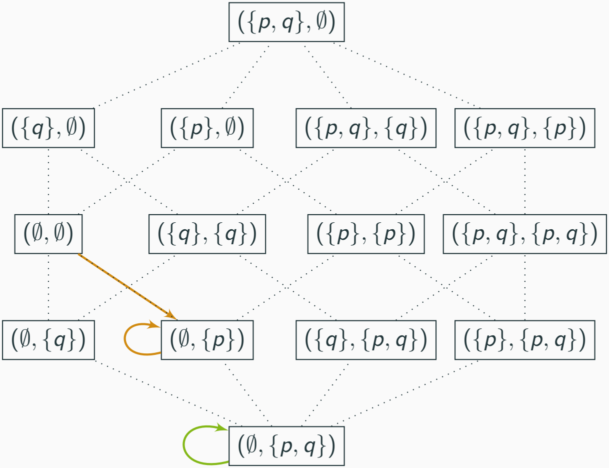
Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$$S(\mathcal{IC}^{l}_{\mathcal{P}})(\{p, q\}) = lfp(\mathcal{IC}^{l}_{\mathcal{P}}(\cdot, \{p, q\}))$$



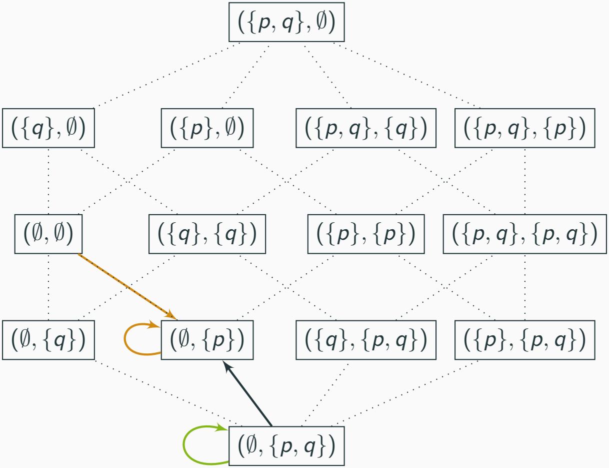
Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$S(\mathcal{IC}^u_{\mathcal{P}})(\emptyset) = lfp(\mathcal{IC}^u_{\mathcal{P}}(\emptyset, \cdot))$



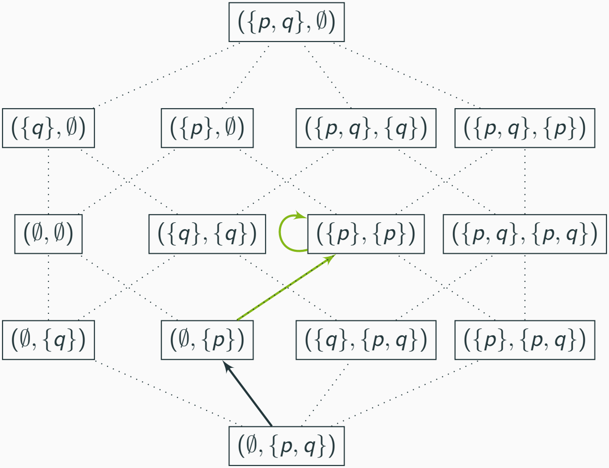
Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$$S(\mathcal{IC}_{\mathcal{P}})(\emptyset, \{p, q\}) = (S(\mathcal{IC}^{l}_{\mathcal{P}})(\{p, q\}),\ S(\mathcal{IC}^{u}_{\mathcal{P}})((\emptyset))$$



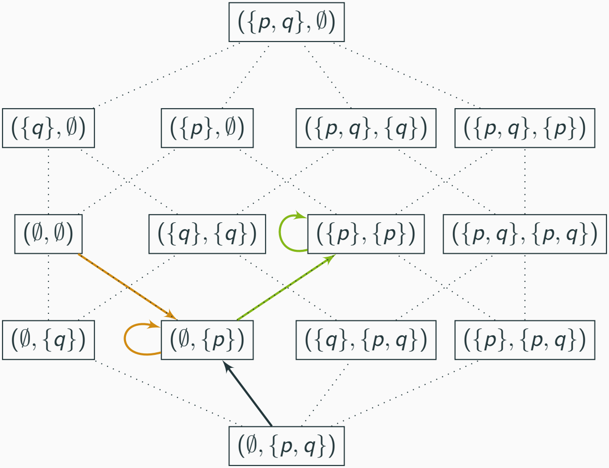
Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$$S(\mathcal{IC}^l_{\mathcal{P}})(\{p\}) = lfp(\mathcal{IC}^l_{\mathcal{P}}(\cdot, \{p\}))$$



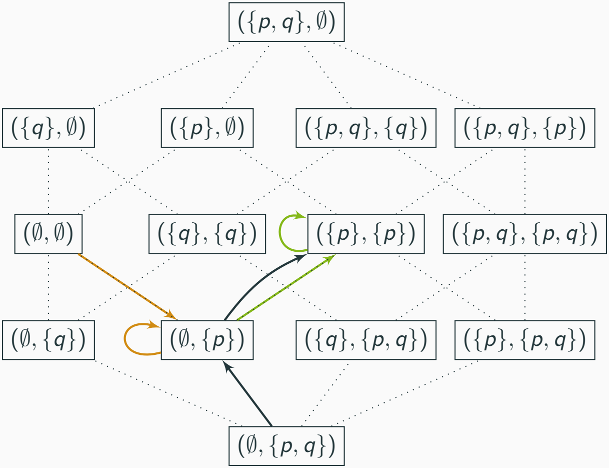
Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$S(\mathcal{IC}^u_{\mathcal{P}})(\emptyset) = lfp(\mathcal{IC}^u_{\mathcal{P}}(\emptyset, \cdot))$

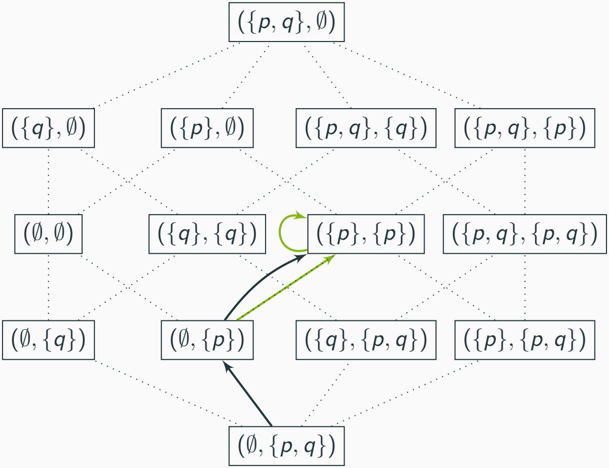


Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$$S(\mathcal{IC_P})(\emptyset, \{p\}) = (S(\mathcal{IC}^l_\mathcal{P})(\{p\}),\ S(\mathcal{IC}^u_\mathcal{P})(\emptyset))$$

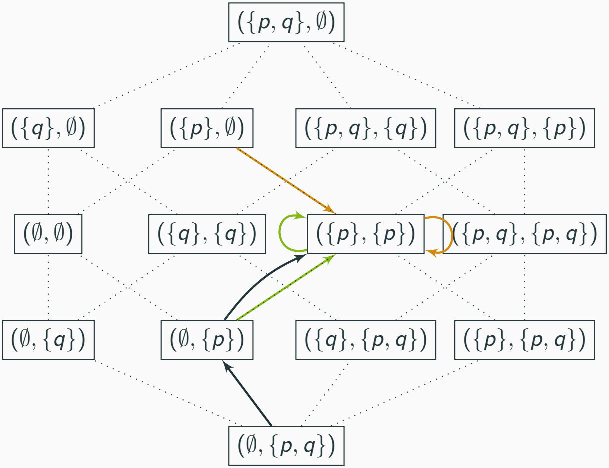


$S(\mathcal{IC}^l_{\mathcal{P}})(\{p\}) = lfp(\mathcal{IC}^l_{\mathcal{P}}(\cdot, \{p\}))$



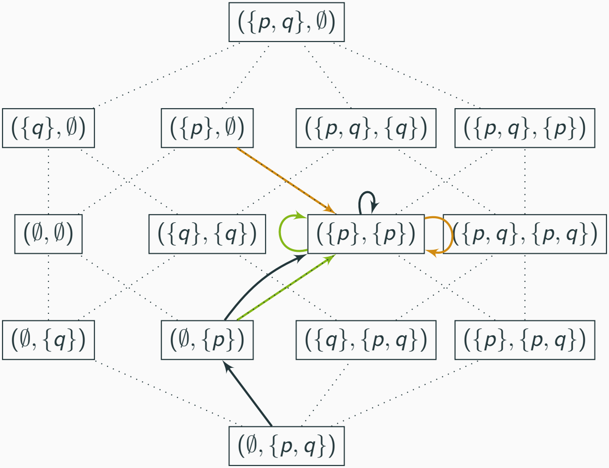
Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$S(\mathcal{IC}^u_\mathcal{P})(x) = lfp(\mathcal{IC}^u_\mathcal{P}(\{p\}, \cdot))$



Stable Operator: Example $\{p \leftarrow \neg q; q \leftarrow q\}$

$$S(\mathcal{IC}_{\mathcal{P}})(\{p\},\{p\}) = (S(\mathcal{IC}^{l}_{\mathcal{P}})(\{p\}),\ S(\mathcal{IC}^{u}_{\mathcal{P}})(\{p\}))$$



Stable Operator and Well-Founded Model

$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = \mathit{lfp}(\mathcal{IC}^l_{\mathcal{P}}(\cdot, y)) \qquad S(\mathcal{IC}^u_{\mathcal{P}})(x) = \mathit{lfp}(\mathcal{IC}^u_{\mathcal{P}}(x, \cdot))$$

$$S(\mathcal{IC}_{\mathcal{P}})(x, y) = (S(\mathcal{IC}^l_{\mathcal{P}})(y),\ S(\mathcal{IC}^u_{\mathcal{P}})(x))$$

- $S(\mathcal{IC}_{\mathcal{P}})$ is a $\leq_i$-monotonic operator, so it admits a least fixpoint.

We call this the well-founded model, denoted $\mathrm{WF}(\mathcal{P})$.

Stable Operator and Well-Founded Model

$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = \mathit{lfp}(\mathcal{IC}^l_{\mathcal{P}}(\cdot, y)) \qquad S(\mathcal{IC}^u_{\mathcal{P}})(x) = \mathit{lfp}(\mathcal{IC}^u_{\mathcal{P}}(x, \cdot))$$

$$S(\mathcal{IC}_{\mathcal{P}})(x, y) = (S(\mathcal{IC}^l_{\mathcal{P}})(y),\ S(\mathcal{IC}^u_{\mathcal{P}})(x))$$

• $S(\mathcal{IC}_{\mathcal{P}})$ is a $\leq_i$-monotonic operator, so it admits a least fixpoint. We call this the well-founded model, denoted $\mathrm{WF}(\mathcal{P})$.

• The well-founded model is more precise than the Kripke-Kleene fixpoint: $\mathrm{KK}(\mathcal{P}) \leq_i \mathrm{WF}(\mathcal{P})$.

Stable Operator and Well-Founded Model

$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = \mathit{lfp}(\mathcal{IC}^l_{\mathcal{P}}(\cdot, y)) \qquad S(\mathcal{IC}^u_{\mathcal{P}})(x) = \mathit{lfp}(\mathcal{IC}^u_{\mathcal{P}}(x, \cdot))$$

$$S(\mathcal{IC}_{\mathcal{P}})(x, y) = (S(\mathcal{IC}^l_{\mathcal{P}})(y),\ S(\mathcal{IC}^u_{\mathcal{P}})(x))$$

• $S(\mathcal{IC}_{\mathcal{P}})$ is a $\leq_i$-monotonic operator, so it admits a least fixpoint.
We call this the well-founded model, denoted $\mathrm{WF}(\mathcal{P})$.

• The well-founded model is more precise than the Kripke-Kleene fixpoint: $\mathrm{KK}(\mathcal{P}) \leq_i \mathrm{WF}(\mathcal{P})$.

• Any fixpoint of $S(\mathcal{IC}_{\mathcal{P}})$ is a minimal model of $\mathcal{P}$.
If $(x, y) = S(\mathcal{IC}_{\mathcal{P}})(x, y)$, we call it a (partial) stable model.
If $x = S(\mathcal{IC}_{\mathcal{P}})(x)$, we call it a stable model.

Stable Operator and Well-Founded Model

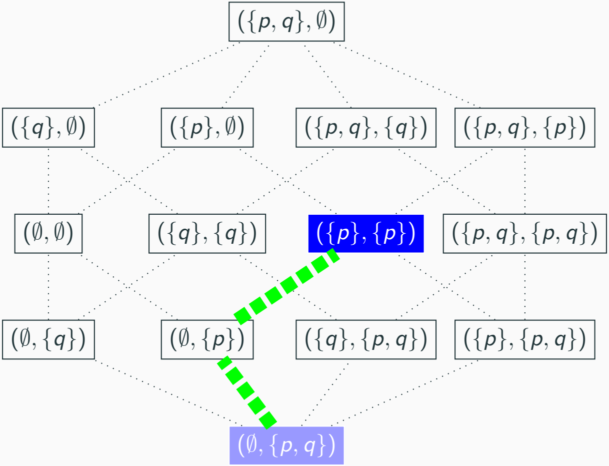
$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = \mathit{lfp}(\mathcal{IC}^l_{\mathcal{P}}(\cdot, y)) \qquad S(\mathcal{IC}^u_{\mathcal{P}})(x) = \mathit{lfp}(\mathcal{IC}^u_{\mathcal{P}}(x, \cdot))$$

$$S(\mathcal{IC}_{\mathcal{P}})(x, y) = (S(\mathcal{IC}^l_{\mathcal{P}})(y),\ S(\mathcal{IC}^u_{\mathcal{P}})(x))$$

- $S(\mathcal{IC}_{\mathcal{P}})$ is a $\leq_i$-monotonic operator, so it admits a least fixpoint. We call this the well-founded model, denoted $\mathrm{WF}(\mathcal{P})$.
- The well-founded model is more precise than the Kripke-Kleene fixpoint: $\mathrm{KK}(\mathcal{P}) \leq_i \mathrm{WF}(\mathcal{P})$.
- Any fixpoint of $S(\mathcal{IC}_{\mathcal{P}})$ is a minimal model of $\mathcal{P}$. If $(x, y) = S(\mathcal{IC}_{\mathcal{P}})(x, y)$, we call it a (partial) stable model. If $x = S(\mathcal{IC}_{\mathcal{P}})(x)$, we call it a stable model.
- If $T_{\mathcal{P}}$ has a least fixpoint, it coincides with the well-founded model.

Stable Operator: Example

$p \leftarrow \neg q; q \leftarrow q$



Stable Operator: Example 2

$$\mathcal{P} = \{p \leftarrow \neg q; \quad q \leftarrow \neg p; \quad r \leftarrow r; \quad s \leftarrow \neg r\}$$

- Kripke-Kleene fixpoint: $(\emptyset, \{p, q, r, s\})$.
- Well-founded model: $(\{s\}, \{p, q, s\})$.
- Stable models: $(\{p, s\}, \{p, s\})$, $(\{q, s\}, \{q, s\})$.

Stable Semantics and Reducts

$$\frac{\mathcal{P}}{x} = \{a \leftarrow b_1, \ldots, b_n \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}$$

$$c_1, \ldots, c_n \notin x\}$$

Definition

x is a stable model of $\mathcal{P}$ if it is a minimal model of $\frac{\mathcal{P}}{x}$.

Stable Semantics and Reducts

$$\frac{\mathcal{P}}{x} = \{a \leftarrow b_1, \ldots, b_n \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}$$
$$c_1, \ldots, c_n \notin x\}$$

Definition

x is a stable model of $\mathcal{P}$ if it is a minimal model of $\frac{\mathcal{P}}{x}$.

Example ($\mathcal{P} = \{p \leftarrow \neg p; q \leftarrow \neg p; p \leftarrow \neg q\}$)

$\frac{\mathcal{P}}{\{q\}} = \{p \leftarrow; q \leftarrow\}$. $\{q\}$ is not a minimal model of $\mathcal{P}$, thus $\{q\}$ is not a stable model.

$\frac{\mathcal{P}}{\{p\}} = \{p \leftarrow\}$. $\{p\}$ is a minimal model of $\mathcal{P}$.$\{q\}$ is not a minimal model of $\mathcal{P}$, thus $\{p\}$ is a stable model.

Stable Semantics and Reducts

$$\frac{\mathcal{P}}{x} = \{a \leftarrow b_1, \ldots, b_n \mid a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m \in \mathcal{P}$$

$$c_1, \ldots, c_n \notin x\}$$

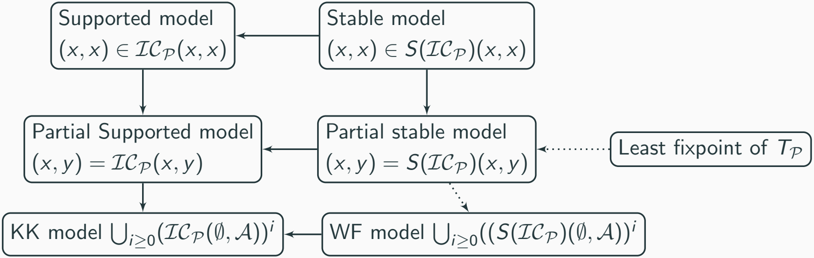
Definition
x is a stable model of $\mathcal{P}$ if it is a minimal model of $\frac{\mathcal{P}}{x}$.

Proposition
$S(\mathcal{IC}^l_{\mathcal{P}})(y)$ is the set of minimal models of $\frac{\mathcal{P}}{y}$.

Proposition
$(x, x) = S(\mathcal{IC}_{\mathcal{P}})(x, x)$ if and only if x is a stable model of $\mathcal{P}$ (iff $x = S(\mathcal{IC}^l_{\mathcal{P}})(x)$).

Approximation Fixpoint Theory

Recap



- Operator-based framework
    - Non-monotonic operator $T_{\mathcal{P}}$,
    - a $\leq_i$-monotonic approximation operator $\mathcal{IC}_{\mathcal{P}}$,
    - and its stable variant $S(\mathcal{IC}_{\mathcal{P}})$.
- Allows us to define semantics as fixpoints of these operators, with attractive properties:
    - KK and WF models exist, can be constructively found, and
    - approximate any fixpoint of $T_{\mathcal{P}}$.
- This story can be told for a great number of formalisms.

Lattices, bilattices, operators

Given a lattice $L = \langle \mathcal{L}, \leq \rangle$.

Interested in operator $O_{\mathcal{L}} : \mathcal{L} \to \mathcal{L}$ and its fixpoints.

- $(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \geq y_2$,
- $(x_1, y_1) \leq_t (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \leq y_2$.

Lattices, bilattices, operators

Given a lattice $L = \langle \mathcal{L}, \leq \rangle$.

Interested in operator $O_{\mathcal{L}} : \mathcal{L} \to \mathcal{L}$ and its fixpoints.

- $(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \geq y_2$,
- $(x_1, y_1) \leq_t (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \leq y_2$.

$\langle \mathcal{L}^2, \leq_i, \leq_t \rangle$ is called a bilattice. Approximate $O_{\mathcal{L}}$ with an approximation operator $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$, which is $\leq_i$-monotonic and for which $\mathcal{O}(x, x) = (O_{\mathcal{L}}(x), O_{\mathcal{L}}(x))$ for any $x \in \mathcal{L}$.

Lattices, bilattices, operators

Given a lattice $L = \langle \mathcal{L}, \leq \rangle$.

Interested in operator $O_{\mathcal{L}} : \mathcal{L} \to \mathcal{L}$ and its fixpoints.

- $(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \geq y_2$,
- $(x_1, y_1) \leq_t (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \leq y_2$.

$\langle \mathcal{L}^2, \leq_i, \leq_t \rangle$ is called a bilattice. Approximate $O_{\mathcal{L}}$ with an approximation operator $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$, which is $\leq_i$-monotonic and for which $\mathcal{O}(x, x) = (O_{\mathcal{L}}(x), O_{\mathcal{L}}(x))$ for any $x \in \mathcal{L}$.

Formalism	Lattice Elements	Order
Logic Programming	Possible worlds	$\subseteq$
Default Logic and AEL	Sets of possible worlds	$\supseteq$
Formal Argumentation	Sets of arguments	$\subseteq$
Weighted ADFs	Weighted worlds	Pointwise comparison
SHACL	Interpretations	Truth order

Operator-Based Semantics for Dialects of Logic Programming

- $\checkmark$ Aggregates in the body: $p \leftarrow \#\mathtt{sum}\{2 : p; q : 1; r : 1\} \geq 2$.
- $\checkmark$ Propositional formulas in the body: $p \leftarrow q \wedge (r \vee (s \wedge \neg t))$.
- $\checkmark$ Disjunctions in the head: $p \vee q \leftarrow q \wedge (r \vee (s \wedge \neg t))$.
- $\checkmark$ Choice constructs in the head: $\#\mathtt{count}\{p; q; r\} = 2 \leftarrow \neg r$.
- $\checkmark$ DL-based logic programs: $\mathsf{K}\, C(x) \leftarrow \neg p(X); C \sqsubseteq D$.
- $\checkmark$ 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: $tr(S, P, O) \leftarrow \& RDF[uri](S, P, O)$.

Operator-Based Semantics for other KR-formalisms

- autoepistemic logic [DMT03],
- default logic [DMT03],
- abstract argumentation [SW15],
- abstract dialectical frameworks [SW15],
- weighted abstract dialectical frameworks [Bog19],
- SCHACL [BJ21].

Operator-Based Studies

Top-Down approach:

- Instead of studying a concept for a specific framework, define and study it for operators over a lattice (and their approximations).
- We can then apply this concept to all formalisms that are or can be captured in AFT.

Examples:

- $\vee$ Stratification [VGD06]
- $\vee$ Conditional Independence [Hey23]
- $\vee$ Knowledge Compilation [BVdB15]
- $\vee$ Groundedness [BVdB15]
- $\vee$ Strong equivalence [Tru06]
- $\vee$ Argumentative dialogues [HA20]
- ? Belief dynamics
- ? Modular equivalence
- ? Neuro-symbolism

Operator-Based Studies: Modularity

Motivating Example

$$r: \ \mathtt{inf}(\mathtt{X}) \leftarrow \mathtt{inf}(\mathtt{Y}), \mathtt{cnct}(\mathtt{Y}, \mathtt{X}), \mathrm{not}\ \mathtt{vac}(\mathtt{X}).$$

Motivating Example

$$r_1: \ \texttt{inf}(\texttt{b}) \leftarrow \texttt{inf}(\texttt{a}), \texttt{cnct}(\texttt{a}, \texttt{b}), \mathrm{not}\ \texttt{vac}(\texttt{b}).$$

$$r_2: \ \texttt{inf}(\texttt{c}) \leftarrow \texttt{inf}(\texttt{d}), \texttt{cnct}(\texttt{d}, \texttt{c}), \mathrm{not}\ \texttt{vac}(\texttt{c}).$$

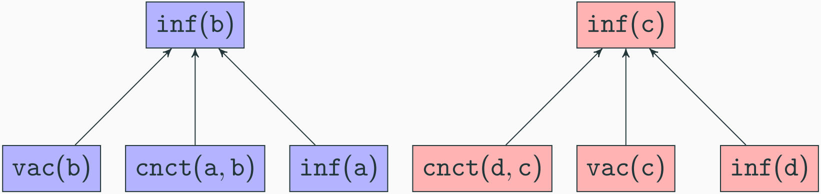
$$r_3: \ \texttt{inf}(\texttt{a}).,\ r_4: \ \texttt{cnct}(\texttt{a}, \texttt{b}).,\ r_5: \ \texttt{cnct}(\texttt{d}, \texttt{c}).$$

Motivating Example

$$r_1: \ \mathtt{inf}(\mathtt{b}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a}, \mathtt{b}), \mathrm{not}\ \mathtt{vac}(\mathtt{b}).$$

$$r_2: \ \mathtt{inf}(\mathtt{c}) \leftarrow \mathtt{inf}(\mathtt{d}), \mathtt{cnct}(\mathtt{d}, \mathtt{c}), \mathrm{not}\ \mathtt{vac}(\mathtt{c}).$$

$$r_3: \ \mathtt{inf}(\mathtt{a}).,\ r_4: \ \mathtt{cnct}(\mathtt{a}, \mathtt{b}).,\ r_5: \ \mathtt{cnct}(\mathtt{d}, \mathtt{c}).$$



Motivating Example

$$r_1 : \ \mathtt{inf}(\mathtt{b}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a}, \mathtt{b}), \mathrm{not}\ \mathtt{vac}(\mathtt{b}).$$

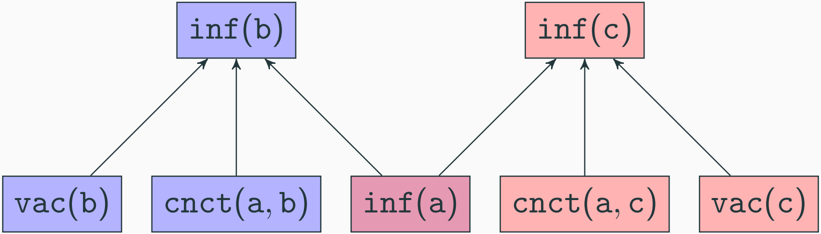
$$r_2 : \ \mathtt{inf}(\mathtt{c}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a}, \mathtt{c}), \mathrm{not}\ \mathtt{vac}(\mathtt{c}).$$

$$r_3 : \ \mathtt{inf}(\mathtt{a}).,\ r_4 : \ \mathtt{cnct}(\mathtt{a}, \mathtt{b}).,\ r_5 : \ \mathtt{cnct}(\mathtt{a}, \mathtt{c}).$$

$$r_1: \ \texttt{inf}(\texttt{b}) \leftarrow \texttt{inf}(\texttt{a}), \texttt{cnct}(\texttt{a}, \texttt{b}), \text{not } \texttt{vac}(\texttt{b}).$$

$$r_2: \ \texttt{inf}(\texttt{c}) \leftarrow \texttt{inf}(\texttt{a}), \texttt{cnct}(\texttt{a}, \texttt{c}), \text{not } \texttt{vac}(\texttt{c}).$$

$$r_3: \ \texttt{inf}(\texttt{a}).,\ r_4: \ \texttt{cnct}(\texttt{a}, \texttt{b}).,\ r_5: \ \texttt{cnct}(\texttt{a}, \texttt{c}).$$

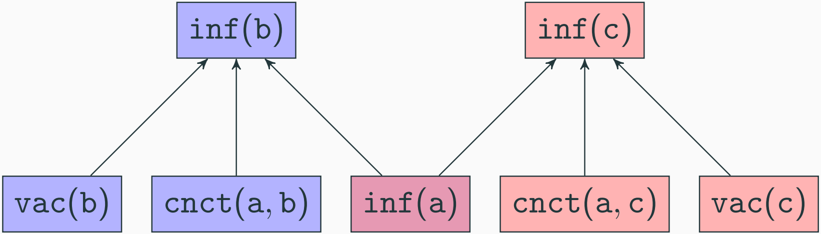


Motivating Example

$$r_1: \ \mathtt{inf}(\mathtt{b}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a},\mathtt{b}), \mathrm{not}\ \mathtt{vac}(\mathtt{b}).$$

$$r_2: \ \mathtt{inf}(\mathtt{c}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a},\mathtt{c}), \mathrm{not}\ \mathtt{vac}(\mathtt{c}).$$

$$r_3: \ \mathtt{inf}(\mathtt{a}).,\ r_4: \ \mathtt{cnct}(\mathtt{a},\mathtt{b}).,\ r_5: \ \mathtt{cnct}(\mathtt{a},\mathtt{c}).$$



Once we know `inf(a)` (or $\neg$`inf(a)`), we obtain two independent subprograms.

Conditional Independence w.r.t. an operator

Definition
Let $O : L_1 \otimes L_2 \otimes L_3 \to L_1 \otimes L_2 \otimes L_3$ *be given.*
$L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ *if there exist operators*

$$O_{1,3} : L_1 \otimes L_3 \to L_1 \otimes L_3 \text{ and } O_{2,3} : L_2 \otimes L_3 \to L_2 \otimes L_3$$

s.t. for $i, j \in \{1, 2\}$, $i \neq j$, *and for every* $x_i \otimes x_3 \in L_i \otimes L_3$ *and for every* $x_j \in L_j$ *it holds that:*

$$O(x_i \otimes x_j \otimes x_3)_{|i,3} = O_{i,3}(x_i \otimes x_3).$$

Conditional Independence w.r.t. an operator

Definition
Let $O : L_1 \otimes L_2 \otimes L_3 \to L_1 \otimes L_2 \otimes L_3$ *be given.*
$L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ *if there exist operators*

$$O_{1,3} : L_1 \otimes L_3 \to L_1 \otimes L_3 \ \text{ and } O_{2,3} : L_2 \otimes L_3 \to L_2 \otimes L_3$$

s.t. for $i, j \in \{1, 2\}$, $i \neq j$, *and for every* $x_i \otimes x_3 \in L_i \otimes L_3$ *and for every* $x_j \in L_j$ *it holds that:*

$$O(x_i \otimes x_j \otimes x_3)_{|i,3} = O_{i,3}(x_i \otimes x_3).$$

where $x_i \otimes x_j \otimes x_{3|i,3} = x_i \otimes x_3$

Search for Fixpoints can be Split

Proposition

Let an operator O s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ be given.
Then $x_1 \otimes x_2 \otimes x_3 = O(x_1 \otimes x_2 \otimes x_3)$ iff
$x_1 \otimes x_3 = O_{1,3}(x_1 \otimes x_3)$ and $x_2 \otimes x_3 = O_{2,3}(x_2 \otimes x_3)$.

Search for Fixpoints can be Split

Proposition

Let an operator O s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ be given.
Then $x_1 \otimes x_2 \otimes x_3 = O(x_1 \otimes x_2 \otimes x_3)$ iff
$x_1 \otimes x_3 = O_{1,3}(x_1 \otimes x_3)$ and $x_2 \otimes x_3 = O_{2,3}(x_2 \otimes x_3)$.

Example

$r_1: \ \mathtt{inf}(\mathtt{b}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a}, \mathtt{b}), \mathrm{not}\ \mathtt{vac}(\mathtt{b}).$

$r_2: \ \mathtt{inf}(\mathtt{c}) \leftarrow \mathtt{inf}(\mathtt{a}), \mathtt{cnct}(\mathtt{a}, \mathtt{c}), \mathrm{not}\ \mathtt{vac}(\mathtt{c}).$

$r_3: \ \mathtt{inf}(\mathtt{a})., \ r_4: \ \mathtt{cnct}(\mathtt{a}, \mathtt{b})., \ r_5: \ \mathtt{cnct}(\mathtt{a}, \mathtt{c}).$

We can look for supported models of $\mathcal{P}$ by looking for supported models of $\mathcal{P}_1$ and $\mathcal{P}_2$ and combining them afterwards.

Reasoning using Modularity

$r_1: \ \mathtt{inf}(\mathrm{b}) \leftarrow \mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{b}), \mathrm{not}\ \mathtt{vac}(\mathrm{b}).$

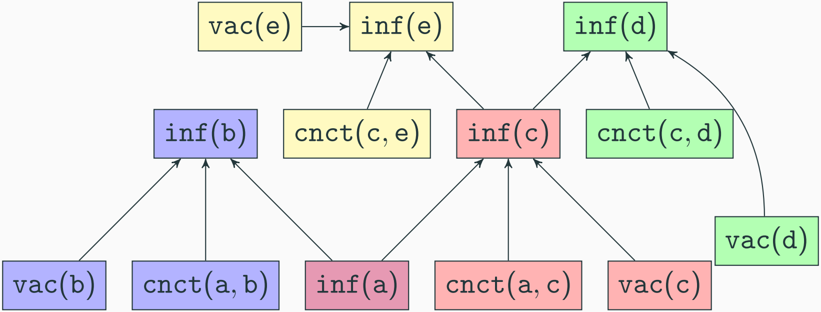
$r_2: \ \mathtt{inf}(\mathrm{c}) \leftarrow \mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{c}), \mathrm{not}\ \mathtt{vac}(\mathrm{c}).$

$r_3: \ \mathtt{inf}(\mathrm{a}).,\ r_4: \ \mathtt{cnct}(\mathrm{a}, \mathrm{b}).,\ r_5: \ \mathtt{cnct}(\mathrm{a}, \mathrm{c}).$

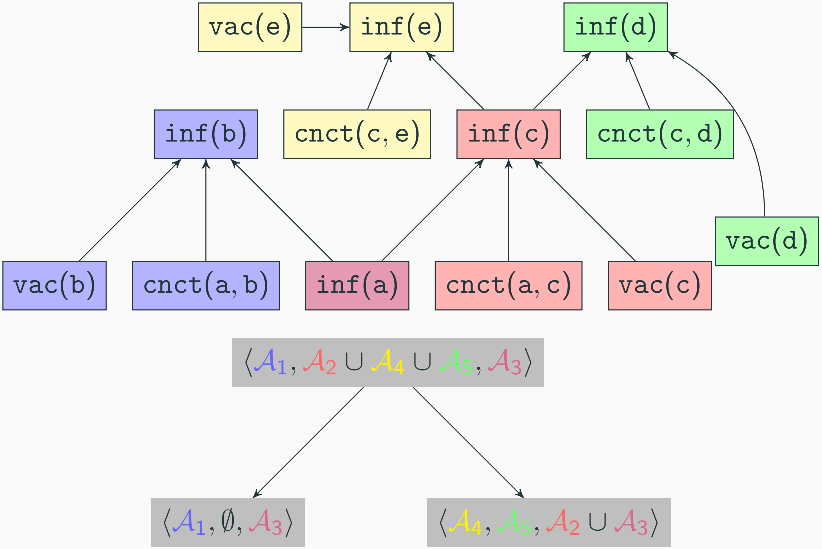
$r_6: \ \mathtt{inf}(\mathrm{d}) \leftarrow \mathtt{inf}(\mathrm{c}), \mathtt{cnct}(\mathrm{c}, \mathrm{d}), \mathrm{not}\ \mathtt{vac}(\mathrm{d}).$

$r_7: \ \mathtt{inf}(\mathrm{e}) \leftarrow \mathtt{inf}(\mathrm{c}), \mathtt{cnct}(\mathrm{c}, \mathrm{e}), \mathrm{not}\ \mathtt{vac}(\mathrm{e}).$

$r_8: \ \mathtt{cnct}(\mathrm{c}, \mathrm{d}).,\ r_9: \ \mathtt{cnct}(\mathrm{c}, \mathrm{e}).$



Reasoning using Modularity



Reasoning using Modularity

$$\begin{tikzcd}[column sep=small]
& \langle \mathcal{A}_1, \mathcal{A}_2 \cup \mathcal{A}_4 \cup \mathcal{A}_5, \mathcal{A}_3 \rangle \arrow[dl] \arrow[dr] & \\
\langle \mathcal{A}_1, \emptyset, \mathcal{A}_3 \rangle & & \langle \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_2 \cup \mathcal{A}_3 \rangle
\end{tikzcd}$$

- $\mathrm{WF}(\mathcal{P}_1 \cup \mathcal{P}_3) = \{\mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{b}), \mathtt{inf}(\mathrm{b})\}$
  Search space size: 2^4
- $\mathrm{WF}(\mathcal{P}_4 \cup \mathcal{P}_2 \cup \mathcal{P}_3) = \{\mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{c}), \mathtt{inf}(\mathrm{c}), \mathtt{cnct}(\mathrm{c}, \mathrm{e}), \mathtt{inf}(\mathrm{e})\}$
  Search space size: 2^7
- $\mathrm{WF}(\mathcal{P}_5 \cup \mathcal{P}_2 \cup \mathcal{P}_3) = \{\mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{c}), \mathtt{inf}(\mathrm{c}), \mathtt{cnct}(\mathrm{c}, \mathrm{d}), \mathtt{inf}(\mathrm{d})\}$
  Search space size: 2^7

Reasoning using Modularity

$$
\begin{tikzcd}[column sep=small]
& \langle \mathcal{A}_1, \mathcal{A}_2 \cup \mathcal{A}_4 \cup \mathcal{A}_5, \mathcal{A}_3 \rangle \arrow[dl] \arrow[dr] & \\
\langle \mathcal{A}_1, \emptyset, \mathcal{A}_3 \rangle & & \langle \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_2 \cup \mathcal{A}_3 \rangle
\end{tikzcd}
$$

- $\mathsf{WF}(\mathcal{P}_1 \cup \mathcal{P}_3) = \{\mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{b}), \mathtt{inf}(\mathrm{b})\}$
  Search space size: 2^4
- $\mathsf{WF}(\mathcal{P}_4 \cup \mathcal{P}_2 \cup \mathcal{P}_3) = \{\mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{c}), \mathtt{inf}(\mathrm{c}), \mathtt{cnct}(\mathrm{c}, \mathrm{e}), \mathtt{inf}(\mathrm{e})\}$
  Search space size: 2^7
- $\mathsf{WF}(\mathcal{P}_5 \cup \mathcal{P}_2 \cup \mathcal{P}_3) = \{\mathtt{inf}(\mathrm{a}), \mathtt{cnct}(\mathrm{a}, \mathrm{c}), \mathtt{inf}(\mathrm{c}), \mathtt{cnct}(\mathrm{c}, \mathrm{d}), \mathtt{inf}(\mathrm{d})\}$
  Search space size: 2^7
- $\mathsf{WF}(\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \cup \mathcal{P}_4 \cup \mathcal{P}_5) =$
  $\mathsf{WF}(\mathcal{P}_1 \cup \mathcal{P}_3) \cup \mathsf{WF}(\mathcal{P}_4 \cup \mathcal{P}_2 \cup \mathcal{P}_3) \cup \mathsf{WF}(\mathcal{P}_5 \cup \mathcal{P}_2 \cup \mathcal{P}_3)$.
  Original search space size: 2^{14}

Fixed-Parameter Complexity

Definition

Let an operator O over the power set lattice $\bigotimes_{i\in I}\mathcal{L}_i$ and CIT $T=(V,E,\nu)$ be given s.t. V_l are the leafs of T. The CIT-partition-size of O relative to (V,E,ν) is defined as

$$\max(\{|\bigotimes_{i\in I_j}\mathcal{L}_i \otimes \bigotimes_{i\in I_3}\mathcal{L}_i| \mid v\in V_\mathrm{l}, \nu(v)=\langle I_1,I_2,I_3\rangle, j=1,2\})$$

Fixed-Parameter Complexity

Definition
Let an operator O over the power set lattice $\bigotimes_{i\in I}\mathcal{L}_i$ and CIT $T=(V,E,\nu)$ be given s.t. V_l are the leafs of T. The CIT-partition-size of O relative to (V,E,ν) is defined as

$$\max(\{|\bigotimes_{i\in I_j}\mathcal{L}_i \otimes \bigotimes_{i\in I_3}\mathcal{L}_i| \mid v\in V_\text{l}, \nu(v)=\langle I_1, I_2, I_3\rangle, j=1,2\})$$

Proposition
Let a $\leq_\otimes$-monotonic operator O over the product lattice $\bigotimes_{i\in I}\mathcal{L}_i$ and CIT $T=(V,E,\nu)$ with CIT-partition-size s be given. Assume that $O(x,y)$ can be computed to a call to an NP-oracle. *The least fixpoint of O can be computed in time $O(f(s))$.*

Round up

Second Commercial break

Tutorial on Approximation Fixpoint Theory

Jesse Heyninck and Hannes Straß

part of the KR 2024 tutorial program

- https://jesseheyninck.github.io/AFTtutorial/
- Half day workshop
  - Motivation and history: a guided tour through the history of logic programming
  - An abstract view: from logic programming to (general) AFT
  - A second application: (weighted) abstract dialectical frameworks
  - More general insights: complexity, groundedness, modularity

Summary

- Operators as the core for understanding answer set semantics.
- Paved the road towards approximation fixpoint theory.
- Algebraic theory that allows language independent work on KR.
- Requires some buy-in, but in my view a great bargain.
- Interested in cooperating? Questions on AFT? Come talk to me.

Bibliography i

Theofanis I Aravanis and Pavlos Peppas.
Belief revision in answer set programming.
In *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, pages 1–5, 2017.

Bart Bogaerts and Maxime Jakubowski.
Fixpoint semantics for recursive shacl.
In *37th International Conference on Logic Programming*, pages 41–47. Open Publishing Association, 2021.

Jonathan Ben-Naim, Salem Benferhat, Odile Papini, and Eric Würbel.
An answer set programming encoding of prioritized removed sets revision: application to gis.
In *European Workshop on Logics in Artificial Intelligence*, pages 604–616. Springer, 2004.

Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen.
Implementing ordered disjunction using answer set solvers for normal programs.
In *Logics in Artificial Intelligence: 8th European Conference, JELIA 2002 Cosenza, Italy, September 23–26, 2002 Proceedings 8*, pages 444–456. Springer, 2002.

Bart Bogaerts.
Weighted abstract dialectical frameworks through the lens of approximation fixpoint theory.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2686–2693, 2019.

Bart Bogaerts and Guy Van den Broeck.
Knowledge compilation of logic programs using approximation fixpoint theory.
Theory and Practice of Logic Programming, 15(4-5):464–480, 2015.

Marc Denecker, Victor Marek, and Mirosław Truszczyński. **Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning.** In *Logic-based Artificial Intelligence*, volume 597 of *The Springer International Series in Engineering and Computer Science*, pages 127–144. Springer, 2000.

Marc Denecker, Victor Marek, and Mirosław Truszczyński. **Uniform semantic treatment of default and autoepistemic logics.** *Artificial Intelligence*, 143(1):79–122, 2003.

Bibliography v

Wolfgang Dvořák, Anna Rapberger, Johannes P Wallner, and Stefan Woltran.
Aspartix-v19-an answer-set programming based system for abstract argumentation.
In *International Symposium on Foundations of Information and Knowledge Systems*, pages 79–89. Springer, 2020.

Melvin Fitting.
Bilattices are nice things.
In *Self Reference*, volume 178 of *CSLI Lecture Notes*, pages 53–77. CLSI Publications, 2006.

Jesse Heyninck and Ofer Arieli.
Argumentative reflections of approximation fixpoint theory.
In *Computational Models of Argument*, pages 215–226. IOS Press, 2020.

Jesse Heyninck.
An algebraic notion of conditional independence, and its application to knowledge representation (preliminary report).
2023.

Ignacio Huitzil, Giuseppe Mazzotta, Rafael Peñaloza, Francesco Ricca, et al.
Asp-based axiom pinpointing for description logics.
In *CEUR WORKSHOP PROCEEDINGS*, volume 3515, pages 1–13. CEUR-WS, 2023.

Tarek Khaled and Belaid Benhamou.
An asp-based approach for attractor enumeration in synchronous and asynchronous boolean networks.
arXiv preprint arXiv:1909.08251, 2019.

Isabelle Kuhlmann, Carl Corea, and John Grant.
Non-automata based conformance checking of declarative process specifications based on asp.
In *International Conference on Business Process Management*, pages 396–408. Springer, 2023.

Roland Kaminski, Javier Romero, Torsten Schaub, and Philipp Wanko.
How to build your own asp-based system?!
Theory and Practice of Logic Programming, 23(1):299–361, 2023.

Isabelle Kuhlmann and Matthias Thimm.
Algorithms for inconsistency measurement using answer set programming.
In *19th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 159–168, 2021.

Hannes Strass and Johannes Peter Wallner.
Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory.
Artificial Intelligence, 226:34–74, 2015.

Terrance Swift.
Deduction in ontologies via asp.
In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 275–288. Springer, 2004.

Mirosław Truszczyński.
Strong and uniform equivalence of nonmonotonic theories–an algebraic approach.
Annals of Mathematics and Artificial Intelligence, 48(3-4):245–265, 2006.

Joost Vennekens, David Gilis, and Marc Denecker.
Splitting an operator: Algebraic modularity results for logics with fixpoint semantics.
ACM Transactions on computational logic (TOCL), 7(4):765–797, 2006.

Allen Van Gelder, Kenneth A Ross, and John S Schlipf.
The well-founded semantics for general logic programs.
Journal of the ACM, 38(3):619–649, 1991.