Splitting an operator

An algebraic modularity result and its application to logic programming

Joost Vennekens
David Gilis
Marc Denecker

KU Leuven, Belgium

Slides by Peter Baumgartner
MPII Saarbrücken, Germany

Logic Programming
↓
Abstraction Theory
↓
Stratification

Various Logic Program Semantics

- Assign "meaning" to a program / knowledge base: perfect model, stable models, **well-founded model**
- Normal (logic) programs: negation in rule body allowed.

$$win(X) \leftarrow move(X, Y),\ not\ win(Y) \quad (1)$$
$$move(c, d) \leftarrow \quad (2)$$
$$move(a, b) \leftarrow \quad (3)$$
$$move(b, a) \leftarrow \quad (4)$$

- **The** well-founded model:

True	Undefined	False
$win(c)$	$win(a)$	$win(d)$
	$win(b)$	

- **Two** stable models:

(i)

True	False
$win(c)$	$win(d)$
$win(a)$	$win(b)$

(ii)

True	False
$win(c)$	$win(d)$
$win(b)$	$win(a)$

More About Well-Founded Models

- See [VanGelder/Ross/Schlipf 89, Przymusinski 91]
- Generally accepted for "reasonable" sceptical reasoning
- "well-behaved" :
  - always exists, stratification not required
  - unique model
  - goal-oriented procedure exists
  - quadratic complexity
- *undef* is assigned to atoms which negatively depend on themselves, and for which no independent "well-founded" derivation exists
- XSB-Prolog system (Warren et. al., top-down system)
- SModels (Niemelä et. al., bottom-up system, also for stable model semantics)

"Building in" Information into Programs

- Program P

$$q \leftarrow \qquad r \leftarrow not\ s$$
$$p \leftarrow not\ q, s \qquad p \leftarrow not\ p$$

- Partial interpretation $\mathcal{J}$

True	Undefined	False
q	p, r	s

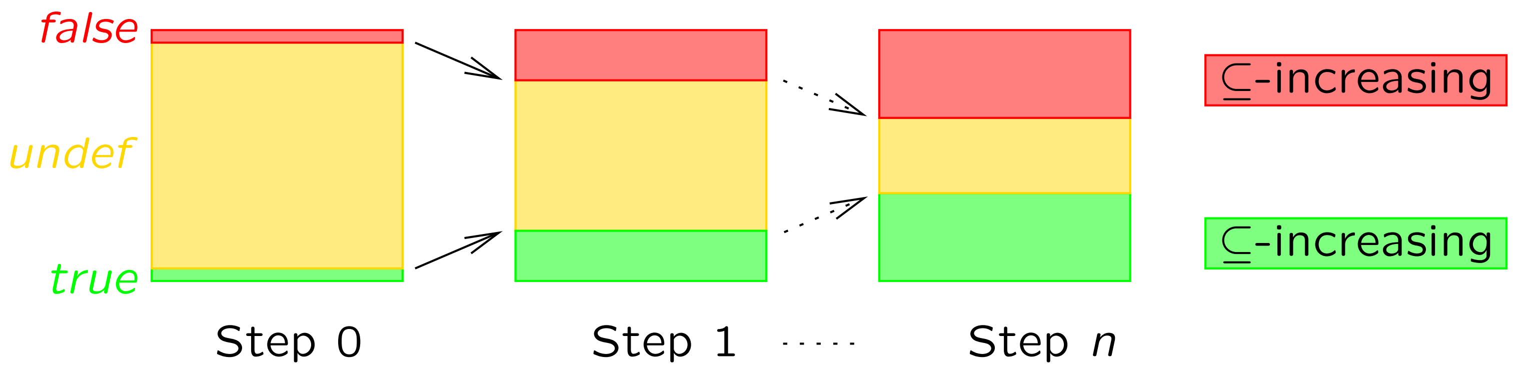
- Quotient program $\frac{P}{\mathcal{J}}$

$$q \leftarrow \qquad r \leftarrow true$$
$$p \leftarrow false, s \qquad p \leftarrow undef$$

- $\mathcal{I}$ is a partial model of $\frac{P}{\mathcal{J}}$ iff for all $Head \leftarrow Body$ in $\frac{P}{\mathcal{J}}$:
  - If $\mathcal{I}(Body) = true$ then $\mathcal{I}(Head) = true$
  - If $\mathcal{I}(Head) = false$ then $\mathcal{I}(Body) = false$
- **Least** partial model $LPM(\frac{P}{\mathcal{J}})$
  - $\mathcal{I}$ minimizes *true* atoms, and
  - $\mathcal{I}$ maximizes *false* atoms

True	Undefined	False
q, r	p	s

Well-Founded Models as Fixpoint Iteration



- Maintain two sets to represent $\mathcal{I}_i$:
  - The "*true*" atoms
  - The "*true* or *undef*" atoms
- Set $\mathcal{I}_0 =$ "all *undef*" and do $\mathcal{I}_{i+1} = LPM(\frac{P}{\mathcal{I}_i})$ until fixpoint, where
- seqeuence $(\mathcal{J}_0 =$ "all *false*"$), \mathcal{J}_1, \ldots, \mathcal{J}_{n-1}, (\mathcal{J}_n = \mathcal{J}_{n+1} = LPM(\frac{P}{\mathcal{I}_i}))$ obtained with operator associated to $(Head \leftarrow Body) \in \frac{P}{\mathcal{I}_i}$:

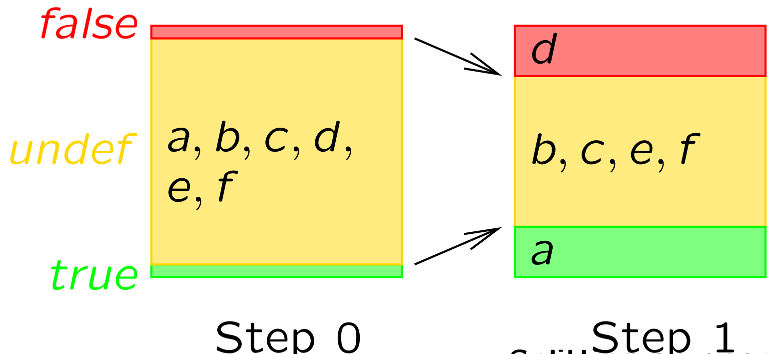
(i) If $\mathcal{J}_k(Body) = true$ then $\mathcal{J}_{k+1}(Head) = true$

(ii) If $\mathcal{J}_{k+1}(Head) = false$ then $\mathcal{J}_k(Body) = false$ iff

If $\underbrace{\mathcal{J}_k(Body) \neq false}_{\mathcal{J}_k(Body) \in \{true, undef\}}$ then $\underbrace{\mathcal{J}_{k+1}(Head) \neq false}_{\mathcal{J}_{k+1}(Head) \in \{true, undef\}}$

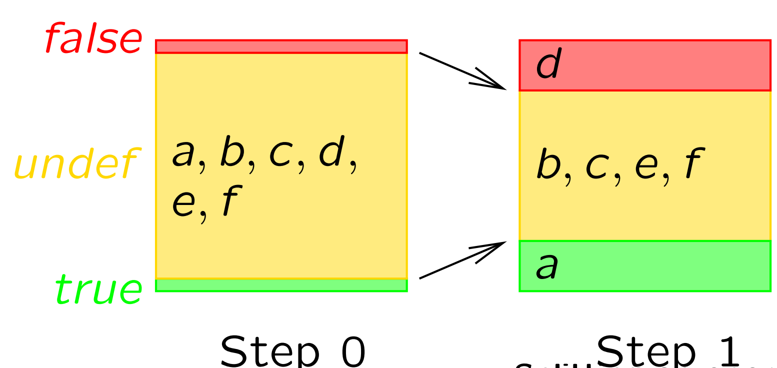
Computing Well-Founded Models, Step 0 $\mapsto$ Step 1

P
$a \leftarrow$
$c \leftarrow not\ b, a$
$b \leftarrow not\ c$
$e \leftarrow not\ d$
$f \leftarrow e$
$f \leftarrow not\ a$



Computing Well-Founded Models, Step 0 $\mapsto$ Step 1

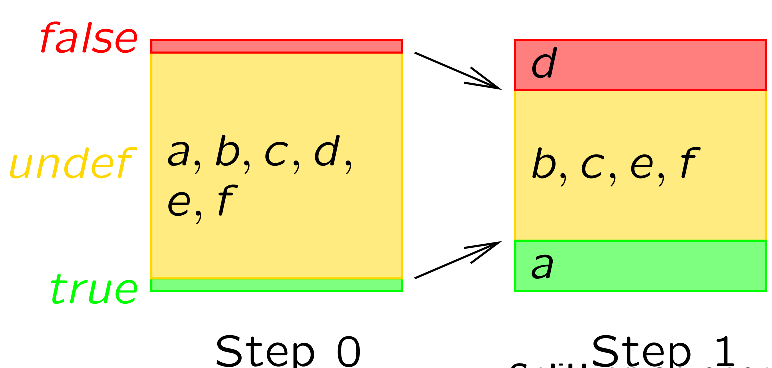
P	(i) build $P/$ [a, b, c, d, e, f]
$a \leftarrow$	$a \leftarrow$
$c \leftarrow \; not\ b, a$	$c \leftarrow undef, a$
$b \leftarrow \; not\ c$	$b \leftarrow undef$
$e \leftarrow \; not\ d$	$e \leftarrow undef$
$f \leftarrow \; e$	$f \leftarrow \; e$
$f \leftarrow \; not\ a$	$f \leftarrow undef$



Computing Well-Founded Models, Step 0 $\mapsto$ Step 1

P	(i) build $P/$ a, b, c, d, e, f
$a \leftarrow$	$a \leftarrow$
$c \leftarrow not\ b, a$	$c \leftarrow undef, a$
$b \leftarrow not\ c$	$b \leftarrow undef$
$e \leftarrow not\ d$	$e \leftarrow undef$
$f \leftarrow e$	$f \leftarrow e$
$f \leftarrow not\ a$	$f \leftarrow undef$

(ii) derive new *true* atoms a

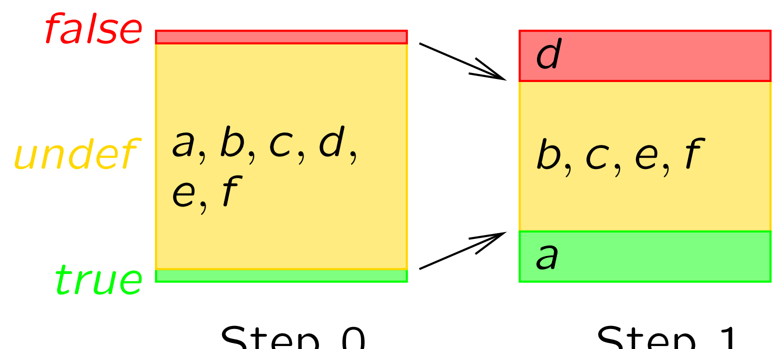


Computing Well-Founded Models, Step 0 $\mapsto$ Step 1

P	(i) build $P/$ [a, b, c, d, e, f]
$a \leftarrow$	$a \leftarrow$
$c \leftarrow not\ b, a$	$c \leftarrow undef, a$
$b \leftarrow not\ c$	$b \leftarrow undef$
$e \leftarrow not\ d$	$e \leftarrow undef$
$f \leftarrow e$	$f \leftarrow e$
$f \leftarrow not\ a$	$f \leftarrow undef$

(ii) derive new *true* atoms [a]

(iii) derive new *true* or *undef* atoms [a | b, c, e, f]



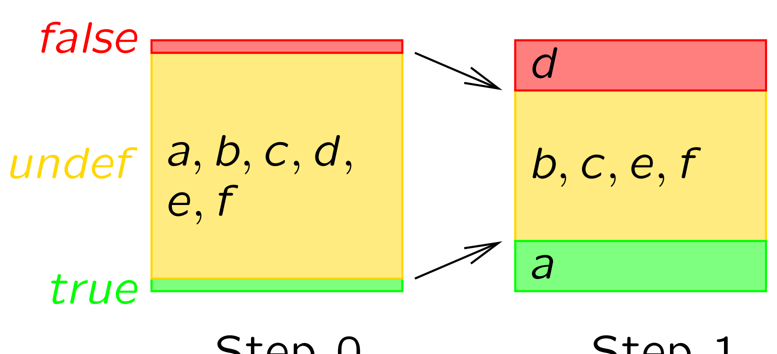
Computing Well-Founded Models, Step 0 $\mapsto$ Step 1

P	(i) build $P/$ [a, b, c, d, e, f]
$a \leftarrow$	$a \leftarrow$
$c \leftarrow not\ b, a$	$c \leftarrow undef, a$
$b \leftarrow not\ c$	$b \leftarrow undef$
$e \leftarrow not\ d$	$e \leftarrow undef$
$f \leftarrow e$	$f \leftarrow e$
$f \leftarrow not\ a$	$f \leftarrow undef$

(ii) derive new *true* atoms [a]

(iii) derive new *true* or *undef* atoms [a | b, c, e, f]

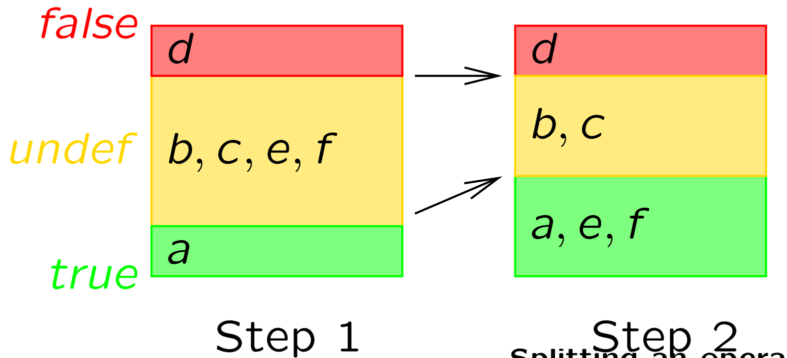
(iv) conclude new *false* atoms [d]



Computing Well-Founded Models, Step 1 $\longmapsto$ Step 2

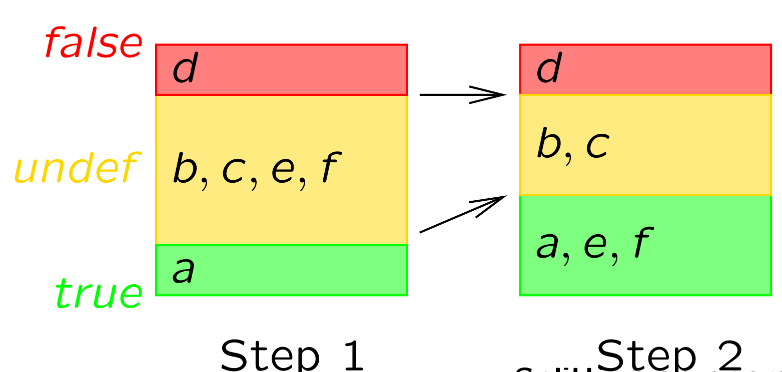
P

$$a \leftarrow$$
$$c \leftarrow not\ b, a$$
$$b \leftarrow not\ c$$
$$e \leftarrow not\ d$$
$$f \leftarrow e$$
$$f \leftarrow not\ a$$



Computing Well-Founded Models, Step 1 $\mapsto$ Step 2

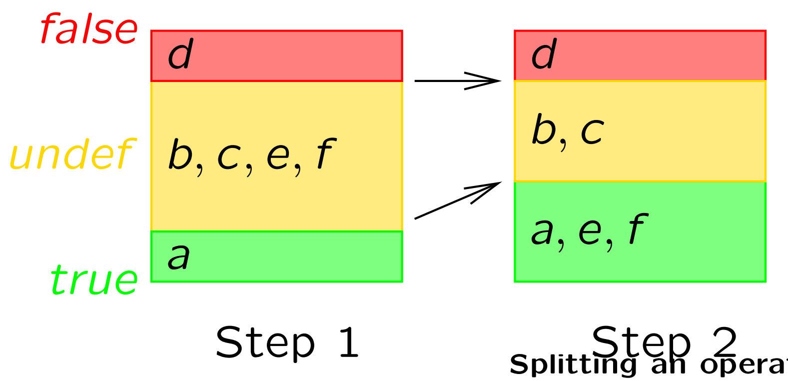
P	(i) build $P/$ a b, c, e, f d
$a \leftarrow$	$a \leftarrow$
$c \leftarrow \textit{not } b, a$	$c \leftarrow \textit{undef}, a$
$b \leftarrow \textit{not } c$	$b \leftarrow \textit{undef}$
$e \leftarrow \textit{not } d$	$e \leftarrow \textit{true}$
$f \leftarrow e$	$f \leftarrow e$
$f \leftarrow \textit{not } a$	$f \leftarrow \textit{false}$



Computing Well-Founded Models, Step 1 $\mapsto$ Step 2

P	(i) build $P/$ a | b, c, e, f | d
$a \leftarrow$	$a \leftarrow$
$c \leftarrow \textit{not } b, a$	$c \leftarrow \textit{undef}, a$
$b \leftarrow \textit{not } c$	$b \leftarrow \textit{undef}$
$e \leftarrow \textit{not } d$	$e \leftarrow \textit{true}$
$f \leftarrow e$	$f \leftarrow e$
$f \leftarrow \textit{not } a$	$f \leftarrow \textit{false}$

(ii) derive new *true* atoms a, e, f

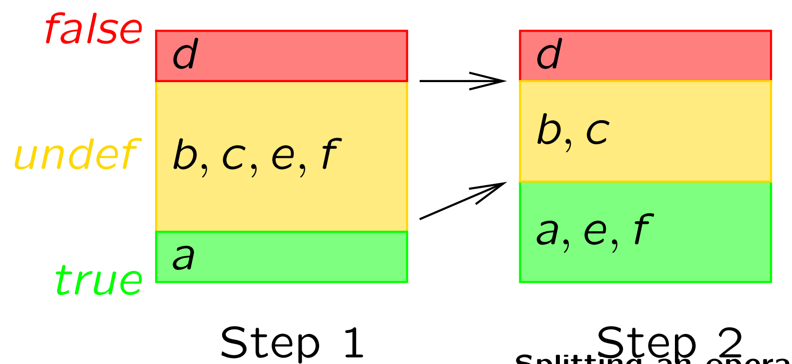


Computing Well-Founded Models, Step 1 $\longmapsto$ Step 2

P	(i) build $P/$ [a | b, c, e, f | d]
$a \leftarrow$	$a \leftarrow$
$c \leftarrow not\ b, a$	$c \leftarrow undef, a$
$b \leftarrow not\ c$	$b \leftarrow undef$
$e \leftarrow not\ d$	$e \leftarrow true$
$f \leftarrow e$	$f \leftarrow e$
$f \leftarrow not\ a$	$f \leftarrow false$

(ii) derive new *true* atoms [a, e, f]

(iii) derive new *true* or *undef* atoms [a, e, f \| b, c]



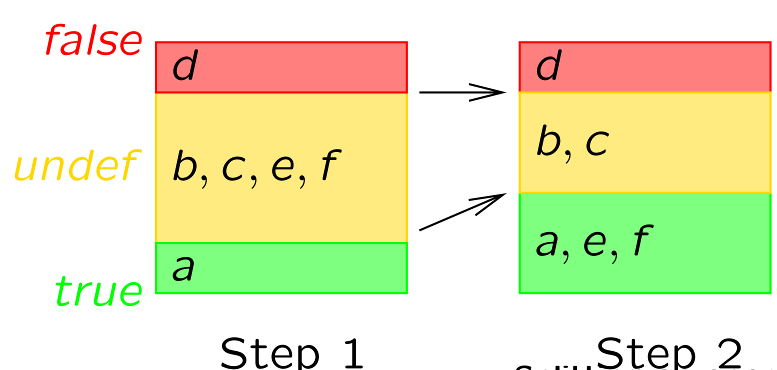
Computing Well-Founded Models, Step 1 $\mapsto$ Step 2

P	(i) build $P / \boxed{a}\boxed{b,c,e,f}\boxed{d}$
$a \leftarrow$	$a \leftarrow$
$c \leftarrow \; not\ b, a$	$c \leftarrow undef, a$
$b \leftarrow \; not\ c$	$b \leftarrow undef$
$e \leftarrow \; not\ d$	$e \leftarrow true$
$f \leftarrow \; e$	$f \leftarrow \; e$
$f \leftarrow \; not\ a$	$f \leftarrow false$

(ii) derive new *true* atoms $\boxed{a, e, f}$

(iii) derive new *true* or *undef* atoms $\boxed{a, e, f}\boxed{b, c}$

(iv) conclude new *false* atoms $\boxed{d}$



Abstraction Theory (Denecker, Marek and Truszczynsk

Recall Fitting operator for logic programs:

(i) If $\mathcal{I}_k(Body) = true$ then $\mathcal{I}_{k+1}(Head) = true$

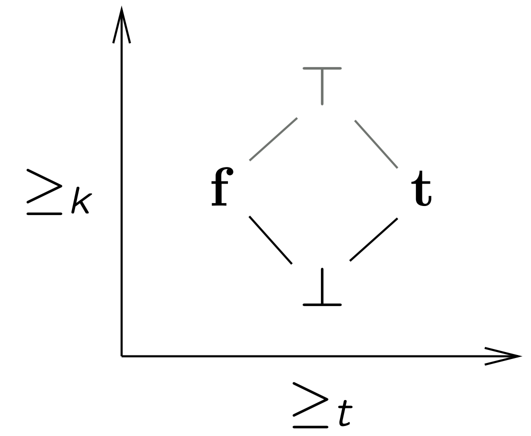
(ii) If If $\mathcal{I}_k(Body) \neq false$ then $\mathcal{I}_{k+1}(Head) \neq false$

Fitting: Semantics as fixpoints of certain derived operators

Abstraction Theory

- Operator (i) alone is sufficient, (ii) is derived (minor issue)
- Other major knowledge representation formalisms (Autoepistemic Logic, Default Logic) can be described by operators comparable to (i) with same monotonicity properties
- **Conclusion:** Develop theory on an abstract level.
- **Applications:**
  – Comparable (new) semantics for AEL and DL Logic as in logic programming
  – Abstract results on stratification

Ordering Interpretations



Ordering of truth values:

$\geq_k$ knowledge (precision, information) ordering
$\geq_t$ truth ordering

Maintain two sets $(X, Y) \in 2^\Sigma \times 2^\Sigma$ to represent an interpretation:

- The "*true*" atoms X
- The "*true* or *undef*" atoms Y

Further notions:

- (X, X) is **exact**
- (X, Y) is **consistent** iff $X \subseteq Y$

Ordering interpretations, bilattices $(2^\Sigma \times 2^\Sigma, \leq_k)$ and $(2^\Sigma \times 2^\Sigma, \leq_t)$:

$(X, Y) \leq_k (X', Y')$ iff $X \subseteq X'$ and $Y' \subseteq Y$ (Knowledge ordering)
$(X, Y) \leq_t (X', Y')$ iff $X \subseteq X'$ and $Y \subseteq Y'$ (Truth ordering)

Evaluation of Formulas

$$H_{(X,Y)}(\phi) = \begin{cases} \mathbf{t} & \phi \text{ is true in the interpretation defined by } (X, Y) \\ \mathbf{f} & \text{otherwise} \end{cases}$$

$$H_{(X,Y)}(p) = \begin{cases} \mathbf{t} & \text{if } p \in X \qquad (p \text{ an atom}) \\ \mathbf{f} & \text{otherwise} \end{cases}$$

$$H_{(X,Y)}(\phi \wedge/\vee \psi) = \begin{cases} \mathbf{t} & \text{if } H_{(X,Y)}(\phi) = \mathbf{t} \text{ and/or } H_{(X,Y)}(\phi) = \mathbf{t} \\ \mathbf{f} & \text{otherwise} \end{cases}$$

$$H_{(X,Y)}(\neg\phi) = \begin{cases} \mathbf{t} & \text{if } H_{(Y,X)}(\phi) = \mathbf{f} \\ \mathbf{f} & \text{otherwise} \end{cases}$$

Associating Operators to Programs

Let P be a Program. Define operator $U_P : 2^\Sigma \times 2^\Sigma \mapsto 2^\Sigma$:

$$U_P(X,Y) = \{p \in \Sigma \mid \text{there is } (p \leftarrow q, \neg r) \in P \text{ with } H_{X,Y}(q \wedge \neg r) = \mathbf{t}\}$$

Note: $H_{X,Y}(q \wedge \neg r) = \mathbf{t}$ iff q is *true* and r is *false* in (X, Y)

Special case

Well known two-valued operator $T_P : 2^\Sigma \mapsto 2^\Sigma$:

$$X \mapsto U_P(X,X)$$

Properties

- Fixpoints of T_P need not exist, take $P = \{p \leftarrow \neg p\}$
- Fixpoints of T_P are **two-valued supported models**
  E.g. fixpoints of $T_{\{p \leftarrow p\}}$ are $\{\}$ and $\{p\}$
- If P is definite then T_P is monotone; LFP is **minimal model**

Fitting Operator as Symmetric Application of U_P

Recall (X, Y) means (“*true* atoms”, “*true* or *undef* atoms”)

Recall

$$U_P(X, Y) = \{p \in \Sigma \mid \text{there is } (p \leftarrow q, \neg r) \in P \text{ with } H_{X,Y}(q \wedge \neg r) = \mathbf{t}\}$$

$H_{X,Y}(q \wedge \neg r) = \mathbf{t}$ iff q is *true* and r is *false* in (X, Y)

Now swap X and Y:

$$U_P(Y, X) = \{p \in \Sigma \mid \text{there is } (p \leftarrow q, \neg r) \in P \text{ with } H_{Y,X}(q \wedge \neg r) = \mathbf{t}\}$$

$H_{Y,X}(q \wedge \neg r) = \mathbf{t}$ iff q is *true* or *undef* and r is *false* or *undef* in (X, Y)

Define Fitting operator $\mathcal{T}_P(X, Y) = (U_P(X, Y), U_P(Y, X))$

$\mathcal{T}_P$ is $\leq_k$-monotone:

if $X \subseteq X'$ and $Y' \subseteq Y$

then $U_P(X, Y) \subseteq U_P(X', Y')$ and $U_P(Y', X') \subseteq U_P(Y, X)$

Intuition for $\mathcal{T}_P$

$$\mathcal{T}_P(X,Y)(p) = \begin{cases} \textit{true} & \text{if there is } (p \leftarrow q, \neg r) \in P \text{ where} \\ & q \text{ and } \neg r \text{ are } \textit{true} \text{ in } (X,Y) \\ \textit{true} \text{ or } \textit{undef} & \text{if there is } (p \leftarrow q, \neg r) \in P \text{ where} \\ & q \text{ and } \neg r \text{ are } \textit{true} \text{ or } \textit{undef} \text{ in } (X,Y) \\ \textit{false} & \text{otherwise} \end{cases}$$

Equivalently:

$$\mathcal{T}_P(X,Y)(p) = \begin{cases} \textit{true} & \text{if there is } (p \leftarrow q, \neg r) \in P \text{ where} \\ & q \text{ and } \neg r \text{ are } \textit{true} \text{ in } (X,Y) \\ \textit{false} & \text{if for all } (p \leftarrow q, \neg r) \in P \text{ it holds} \\ & q \text{ or } \neg r \text{ is } \textit{false} \text{ in } (X,Y) \\ \textit{true} \text{ or } \textit{undef} & \textit{otherwise} \end{cases}$$

Properties of $\mathcal{T}_P$

- $\mathcal{T}_P$ is $\leq_k$-monotone, thus least fixpoint exists;
  Bottom element is $(\{\}, \Sigma)$
  Gives **Kripke-Kleene semantics**, (or Fitting semantics)
- Examples

Program	Fixpoint iteration
$p \leftarrow \neg q$	$(\{\}, \{p, q\}) \rightarrow (\{\}, \{p\}) \rightarrow (\{p\}, \{p\})$
$p \leftarrow \neg p$	$(\{\}, \{p, q\}) \rightarrow (\{\}, \{p\})$
$p \leftarrow p$	$(\{\}, \{p, q\}) \rightarrow (\{\}, \{p\})$

Abstraction Theory (1)

- Given a lattice $(L, \leq)$ – concrete case $(2^\Sigma, \subseteq)$
- Bilattice $(L \times L, \leq_p)$ – concrete case $(2^\Sigma \times 2^\Sigma, \leq_k)$
- **Approximation:** any $\leq_p$-monotone operator $A : L \times L \mapsto L \times L$
  A can be written as

$$\underbrace{A(X,Y)}_{\mathcal{T}_P(X,Y)} = (\underbrace{A_1(X,Y)}_{U_P(X,Y)}, \underbrace{A_2(X,Y)}_{U_P(Y,X)})$$

- Derived operators (1) - holding an argument as parameter:

  $A^1(\cdot, Y) = \lambda X.A_1(X,Y)$ – concrete case $A^1(\cdot, Y) = \lambda X.U_p(X,Y)$
  $A^2(X, \cdot) = \lambda Y.A_2(X,Y)$ – concrete case $A^2(X, \cdot) = \lambda Y.U_p(Y,X)$

  Both A_1 and A_2 are $\leq$-monotone

Abstraction Theory (2)

- Derived operators (1) from above:

  $A^1(\cdot, Y) = \lambda X.A_1(X, Y)$
  $A^2(X, \cdot) = \lambda Y.A_2(X, Y)$

- Derived operators (2): $(C^{\downarrow}_{\mathcal{T}_P}(Y), C^{\uparrow}_{\mathcal{T}_P}(X)) = LPM(\frac{P}{(X,Y)})$

$$C^{\downarrow}_A(Y) = LFP(A^1(\cdot, Y))$$
$$C^{\uparrow}_A(X) = LFP(A^2(X, \cdot))$$

  Both $C^{\downarrow}_A$ and $C^{\uparrow}_A$ are $\leq$-antimonotone

- **Partial stable operator of A:**

$$\mathcal{C}_A(X, Y) = (C^{\downarrow}_A(Y), C^{\uparrow}_A(X))$$

  Because $C^{\downarrow}_A$ and $C^{\uparrow}_A$ are $\leq$-antimonotone, $\mathcal{C}_A$ is $\leq_p$-monotone

  $LFP(\mathcal{C}_{\mathcal{T}_P})$ (wrt. $\leq_k$) is the **well-founded model**
  Two-valued fixpoints of $\mathcal{C}_{\mathcal{T}_P}$ are the **stable models**

Summary - Abstraction Theory $\rightarrow$ Logic Programming

Start with an operator O – concrete case U_P.

Semantics of derived operators:

- $T_P(X) = U_P(X, X)$
  - Fixpoints: 2-valued supported models
- $\mathcal{T}_P(X, Y) = (U_P(X, Y), U_P(Y, X))$
  - Fixpoints: 3-valued supported models
  - LFP: Kripke-Kleene semantics
- Let $A = \mathcal{T}_P$. Partial stable operator $\mathcal{C}_A(X, Y) = (C_A^{\downarrow}(Y), C_A^{\uparrow}(X))$
  - Fixpoints: (partial) stable models
  - LFP: well-founded model

Application to Default Logic and Autoepistemic Logic

Default Logic and Autoepistemic Logic semantics can be described by suitable operators O. Then:

- Usual Moore semantics for AEL is given by 2-valued supported models ("$X \mapsto U_P(X,X)$")
- Usual Reiter semantics for DL is given by 2-valued stable models
- Intuitive mapping from DL to AEL:

Default logic inference rule:	Translation to Autoepistemic Logic:
$\dfrac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}$	$\mathbf{L}\alpha \land \neg\mathbf{L}\neg\beta_1 \land \cdots \land \neg\mathbf{L}\neg\beta_n \rightarrow \gamma$

Reiter semantics for DL is the same as the 2-valued stable model semantics for the translation!

Dependency Graph leads to Stratification

Example, $\Sigma = \{p, q, r\}$:

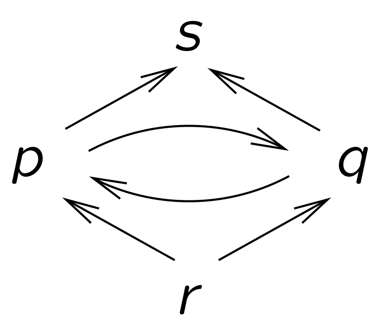
P:

$$s \leftarrow p, q \qquad (1)$$

$$p \leftarrow \neg q, \neg r \qquad (2)$$

$$q \leftarrow \neg p, \neg r \qquad (3)$$

Dependency graph:



$$\Sigma_2 = \{s\}$$

$$\Sigma_1 = \{p, q\}$$

$$\Sigma_0 = \{r\}$$

Suggests **splitting** $\Sigma = \Sigma_0 \dot{\cup} \Sigma_1 \dot{\cup} \Sigma_2$

Contribution: The program P is not stratified in the standard sense, but models can still be constructed in a stratified way $\Sigma_0 \rightarrow \Sigma_1 \rightarrow \Sigma_2$.

Stratification in Abstraction Theory - Product Lattices

So far: lattice $(2^\Sigma, \subseteq)$ and bilattice $(2^\Sigma \times 2^\Sigma, \leq_k)$

Now:

- Product lattice $(\bigotimes_{i=0,\ldots,n} 2^{\Sigma_i}, \subseteq)$, where
  - $(\bigotimes_{i=0,\ldots,n} 2^{\Sigma_i}, \subseteq) = (2^{\Sigma_0}, \ldots, 2^{\Sigma_n})$, and
  - $(x_0, \ldots, x_n) = x \subseteq y = (y_0, \ldots, y_n)$ iff $x_0 \subseteq y_0$ and ... and $x_n \subseteq y_n$
- Example: $\Sigma = \underbrace{\{r\}}_{\Sigma_0} \dot\cup \underbrace{\{p, q\}}_{\Sigma_1} \dot\cup \underbrace{\{s\}}_{\Sigma_2}$

  $x = (\{r\}, \{p\}, \{\}) \in \bigotimes_{i=0,1,2} 2^{\Sigma_i}$
  $y = (\{r\}, \{p, q\}, \{s\}) \in \bigotimes_{i=0,1,2} 2^{\Sigma_i}$
  It holds $x \subseteq y$
- Bilattice of product lattices $(\bigotimes_{i=0,\ldots,n} 2^{\Sigma_i} \times \bigotimes_{i=0,\ldots,n} 2^{\Sigma_i}, \text{“}\leq_k\text{”})$
- Product lattice of bilattices $(\bigotimes_{i=0,\ldots,n}(2^{\Sigma_i} \times 2^{\Sigma_i}), \text{“}\leq_k\text{”})$

Stratification in Abstraction Theory - Results

Notation: e.g. $x = (\{r\}, \{p\}, \{\})$. Then $x|_{\leq 1} = (\{r\}, \{p\})$

Definition: ("Applying O at stratum i does not depend from strata $> i$.")

Operator O on a product lattice L is **stratifiable** iff
for all $x, y \in L$ and all $i = 0, \ldots, n$:
if $x|_{\leq i} = y|_{\leq i}$ then $O(x)|_{\leq i} = O(y)|_{\leq i}$.

Theorem: ("Logic programming: splitting results in stratification")

Let P be a logic program and $(\Sigma_i)_{i=0,\ldots,n}$ a splitting.
Then the operator $\mathcal{T}_P$ on the bilattice of the product lattice
$(\bigotimes_{i=0,\ldots,n} 2^{\Sigma_i} \times \bigotimes_{i=0,\ldots,n} 2^{\Sigma_i}, \text{“}\leq_k\text{”})$ is stratifiable.

Theorem: ("Stratum-wise computation of fixpoints")

Let L be a product lattice, O a stratifiable operator and $x \in L$.
Then x is a fixpoint of O iff for all $i = 0, \ldots, n$:
$x|_i$ is a fixpoint of $O(x)|_i$ $\quad$ ($x|_i$ fixpoint of $O_i^{x|_{<i}}$).

$\longrightarrow$ similar result for least fixpoints

Stratification: Example

O is $\mathcal{T}_P$, where

P:

$$s \leftarrow p, q \quad (1)$$

$$p \leftarrow \neg q, \neg r \quad (2)$$

$$q \leftarrow \neg p, \neg r \quad (3)$$

Task: compute well-founded model x of P (i.e. least fixpoint of $\mathcal{T}_P$)

Construct well-founded models of $P_0^{x|_{<0}}$, $P_1^{x|_{<1}}$, $P_2^{x|_{<2}}$

$\Sigma_0 = \{r\}$, $P_0 = \emptyset$, $P_0^{x|_{<0}} = \emptyset$, well-founded model is $x|_{<1} = (\{\}, \{\})$

$\Sigma_1 = \{p, q\}$, $P_1 = \{(2), (3)\}$, with $x|_{<1}(r) =$ *false* have

$P_1^{x|_{<1}}$:

$$p \leftarrow \neg q, \mathbf{t} \quad (2')$$

$$q \leftarrow \neg p, \mathbf{t} \quad (3')$$

Well-founded model is $x|_{<2} = ((\{\}, \{\}), (\{\}, \{p, q\}))$

Stratification: Example

O is $\mathcal{T}_P$, where

P:

$$s \leftarrow p, q \tag{1}$$

$$p \leftarrow \neg q, \neg r \tag{2}$$

$$q \leftarrow \neg p, \neg r \tag{3}$$

Recall well-founded model $x|_{<2} = ((\{\}, \{\}), (\{\}, \{p, q\}))$

$\Sigma_2 = \{s\}$, $P_2 = \{(1)\}$,
with $x|_{<2}(r) = \mathit{false}$, $x|_{<2}(p) = \mathit{undef}$ and $x|_{<2}(q) = \mathit{undef}$ have

$P_2^{x|_{<2}}$:

$$s \leftarrow \mathbf{u}, \mathbf{u} \tag{1'}$$

Well-founded model is $x|_{<3} = ((\{\}, \{\}, \{\}), (\{\}, \{p, q\}, \{s\}))$

This is the well-founded model of P

Conclusions

- Abstraction theory: framework to explain and construct semantics of knowledge representation formalism in a uniform way
- Abstract concept of stratification: useful for own work