Ultimate approximations and their applications in knowledge representation

Marc Denecker
K.U. Leuven

Victor Marek and Mirek Truszczyński
University of Kentucky

Overview

- Logic programming — a brief intro to how lattices, operators and fixpoints come about
- An abstract algebraic setting for our work
- The theory of approximating operators — an algebraic account of semantics of logic programs (but also other knowledge-representation systems)
- Ultimate approximations
- Applications in logic programming
- Other applications — briefly

Logic programming

- A *clause*:

$$p \leftarrow q_1, \ldots, q_m, \neg r_1, \ldots, \neg r_n$$

  (p, q_i and r_j are atoms)

- A *Horn clause* — no negation in the body

$$p \leftarrow q_1, \ldots, q_m$$

- A *logic program* (or, just *program*): a set of clauses
- A *Horn program*: Horn clauses only
- Programs:
  - computational mechanisms (clauses as inference rules: derive p if all q_i can be derived and none of r_j can)
  - declarative specifications (definitions) of concepts (clauses as logic formulas)
- How to make these two views formal?
- How to reconcile them?

Logic programming

- Key semantic concept, an *interpretation*: a mapping from atoms to truth values
- Interpretations — world (state) descriptions, specify which elementary statements hold in the world
- Programs:
  - mechanisms to revise interpretations (procedural reading)
  - specification of a class of interpretations as a description of a concept or piece of domain knowledge (declarative reading)

Example I

- Consider a program P

$$p \leftarrow q$$
$$q \qquad \text{(shorthand for } q \leftarrow\text{)}$$
$$r \leftarrow q, r$$

- Consider an interpretation I_0 in which p, q and r are all false ($I_0 = \emptyset$; we represent interpretations as sets of atoms that are true)
- P revises I_0 to $I_1 = \{q\}$ (q is *computed* to be true)
- Then, P revises I_1 to $I_2 = \{p, q\}$ (p is now forced to be true, too)
- I_2 cannot be revised any more (no change from that point on)
- I_2 — a model of P, in fact, the least model
- (Not coincidentally — P is a Horn program)
- But there are other interpretations that do not change under P: $I_3 = \{p, q, r\}$

Example II

- Consider a program P

$$p \leftarrow \neg q$$
$$q \leftarrow \neg p$$
$$r \leftarrow q, r$$

- Let $I_0 = \emptyset$
- P revises I_0 to $I_1 = \{p, q\}$
- Then, P revises I_1 to $I_2 = \emptyset = I_0$
- The computational process does not terminate — a problem!
- But I_1 has some interesting aspect — does not revise outside itself (it is a *model*!)
- And, there are interpretations that do not change under P:
  - $I_3 = \{p\}$, $I_4 = \{q\}$, $I_5 = \{q, r\}$
  - special type of models

Thinking in terms of operators

- We assign to P an operator T_P mapping interpretations to interpretations
  - $T_P(I)$ consists of all atoms p such that there is a rule in P with head p and such that its body is true in I
- The domain of T_P — the complete lattice of all interpretations (the lattice of all subsets of a set, with inclusion as the ordering relation)
- The operator captures the concept of revising

  $T_P(I)$ is the revision of I by P
- The operator specifies classes of interpretations, for instance, as its:
  - *prefixpoints* ($T_P(I) \subseteq I$): interpretations that do not revise outside of themselves (models)
  - *fixpoints* ($T_P(I) = I$): interpretations that do not change under P (special type of models)

The setting

- $\langle L, \leq \rangle$ — a complete lattice
- $x \in L$ — represents information about the world
- Operator O on L — represents a mechanism to revise information
- Of particular interest:
  - *prefixpoints*: $O(x) \leq x$ — revising information does not lead outside x
  - *fixpoints*: $O(x) = x$ — revising information does not change x
- Goal: develop techniques to study fixpoints and prefixpoints of operators on lattices
- Basic tools:
  - Approximations and approximating operators
  - Monotone operators
  - Knaster-Tarski Theorem: a monotone operator on a complete lattice has a least fixpoint

Approximations and bilattices

- A pair $(x, y) \in L^2$ *approximates* an element z if $x \leq z \leq y$
- Orderings of approximations:
  - *information* (*precision*) ordering:
    $(x_1, y_1) \leq_i (x_2, y_2)$ if $x_1 \leq x_2$ and $y_2 \leq y_1$
  - *truth* ordering:
    $(x_1, y_1) \leq_t (x_2, y_2)$ if $x_1 \leq x_2$ and $y_1 \leq y_2$
- A pair (x, y) is *consistent* if $x \leq y$, and *inconsistent*, otherwise
- An element (x, y) is *complete* if $x = y$
- $\langle L^2, \leq_i, \leq_t \rangle$ is a bilattice!
- However:
  - only the ordering $\leq_i$ is really needed in our work
  - ordering $\leq_t$ plays only a marginal role
  - our theory does not exploit any interactions between the orderings

Approximating operators

- $A : L^2 \to L^2$ is an *approximating operator* if:
  - A is $\leq_i$-monotone
  - A is symmetric:
    $$A(x, y) = (A(y, x)_2, A(y, x)_1),$$
    where $A(x, y) = (A(x, y)_1, A(x, y)_2)$
- Symmetry implies *completeness*: for every $x \in L$, $A(x, x)$ is complete ($A(x, x) = (y, y)$, for some $\in L$)
- $A : L^2 \to L^2$ *approximates* an operator O on L if
  - A is approximating
  - $A(x, x) = (O(x), O(x))$
- Why symmetry?

Approximating operators — some properties

- Approximating operators are *consistent*: they map consistent approximations to consistent approximations
- Approximating operators have a $\leq_i$-least fixpoint (as they are $\leq_i$-monotone)
- This least fixpoint is consistent
- $LF(A)$
- For every fixpoint (x, y) of A

$$LF(A) \leq_i (x, y)$$

- If A (defined on L^2) approximates O (defined on L), complete fixpoints of A correspond to fixpoints of O

$$A(x, x) = (x, x) \quad \text{iff} \quad O(x) = x$$

- For every fixpoint x of O,

$$LF(A) \leq_i (x, x) \quad \text{or} \quad LF(A)_1 \leq x \leq LF(A)_2$$

Stable operators

- Are there classes of fixpoints of an approximating operator that are particularly interesting?
- Since $A : L^2 \rightarrow L^2$ is $\leq_i$-monotone,
$$A(\cdot, y)_1 \quad \text{and} \quad A(x, \cdot)_2$$
are monotone
- Therefore, the following operators on L are well defined:
$$A^s(x)_l = LF(A(\cdot, x)_1) \quad \text{and} \quad A^s(x)_u = LF(A(x, \cdot)_2)$$
- By the symmetry of A, $A^s_l = A^s_u$
- Thus, we set $A^s = A^s_l$ (and so, we also have $A^s = A^s_u$)
- We then define:
$$\mathcal{A}^s(x, y) = (A^s(y), A^s(x))$$
- A^s and $\mathcal{A}^s$ — stable operators for A

Properties of stable operators

- A^s is antimonotone ($x \leq y$ implies that $A^s(y) \leq A^s(x)$)
- Therefore, $\mathcal{A}^s$ is $\leq_i$-monotone
  (it is also $\leq_t$-antimonotone, but we do not need that fact)
- Thus, $\mathcal{A}^s$ is an approximating operator (it approximates A^s)
- (x, y) is a *stable* fixpoint of A if (x, y) is a fixpoint of $\mathcal{A}^s$
- The terminology is justified: if $\mathcal{A}^s(x, y) = (x, y)$, $A(x, y) = (x, y)$
- Even more: stable fixpoints are $\leq_t$-minimal fixpoints of A
  - Minimality important for knowledge representation applications
- We define $WFF(A) = LF(\mathcal{A}^s)$
- $WFF(A)$ approximates all stable fixpoints of A
- Clearly, $LF(A) \leq_i WFF(A)$

Properties of stable operators

- Let O — an operator on L
- Let A — an approximating operator for O
- If $\mathcal{A}^s(x,x) = (x,x)$, then $A^s(x) = x$, that is,
- $x = LF(A(\cdot, x)_1)$ and, consequently, $x = A(x,x)_1 = O(x)$
- Each such x — A-*stable* fixpoint of O
  - $\leq$-minimal fixpoints of O
  - All of them approximated by $WFF(A)$

A poset of consistent approximations

- Can we reproduce all essential elements of our approach in the setting of consistent elements of L^2?
- After all, only consistent elements of L^2 have a sound interpretation as an approximation
- Define $L^c = \{(x, y) \in L^2 : x \leq y\}$
- $\langle L^c, \leq_i \rangle$ not a lattice!
- However, $\langle L^c, \leq_i \rangle$ is a *chain-complete* poset:
  - it has a least element and every chain has a least upper bound in the poset
  - Knaster-Tarski Theorem extends to chain-complete posets (Markowsky)

Approximating operators on L^c

- Operator A (defined on L^c) is *approximating* if
  - A is $\leq_i$-monotone
  - A is complete
- Need for symmetry disappears!
- Completeness requirement can be relaxed

  w/out it:
  A approximates O if for every $X \in L$, $A(x, x) \leq_i (O(x), O(x))$
- $\leq_i$-monotonicity remains as the *only* essential requirement

Approximations on L^c vs. those on L^2

- Let A be an approximating operator on L^c
- Concepts of stable operator, A-stable fixpoint, well-founded fixpoint of A can be defined within the setting of L^c
- Definitions are not simple variations of those we have for the case L^2
- For instance, stable operator is defined on some consistent pairs only
- Connections to approximations on L^2
  - Every approximating operator on L^c extends to an approximating operator on L^2 (not straightforward as symmetry must be enforced)
  - In general, several extensions possible (all with the same consistent stable fixpoints, however)
  - The restriction of an approximating operator on L^2 is an approximating operator on L^c
- Thus L^2-approximation theory and L^c-approximation theory are equivalent as long as we are interested in consistent elements in L^c

Back to logic programming

- P — logic program
- L — lattice of interpretations (sets of atoms or truth assignments)
- T_P — an operator
- T_P has a counterpart in the setting of 3-valued interpretations
  - truth values: **t**, **f** and **u**
  - negation flips **t** and **f**, fixes **u**
  - $\mathbf{f} \leq \mathbf{u} \leq \mathbf{t}$; $\wedge$ — minimum; $\vee$ — maximum
- $\mathcal{T}_P(K)(p) = \mathbf{f}$ if no clause with p in the head
  $\mathcal{T}_P(K)(p) =$ maximum truth value (in K) of the body of a rule with p in the head, otherwise
- $\mathcal{T}_P$ has fixpoints!!

Back to logic programming

- 3-valued interpretations correspond to pairs of consistent 2-valued interpretations — pairs (I, J) of sets of atoms such that $I \subset J$
- Given such (I, J), define K:

$$\begin{aligned} K(a) &= \mathbf{t} && \text{if } a \in I \\ K(a) &= \mathbf{f} && \text{if } a \notin J \\ K(a) &= \mathbf{u} && \text{otherwise} \end{aligned}$$

- Given a 3-valued interpretation K, define

$$I = \{p\colon K(p) = \mathbf{t}\} \quad \text{and} \quad J = \{p\colon K(p) = \mathbf{t} \text{ or } \mathbf{u}\}$$

- Thus, $\mathcal{T}_P$ — an operator on L^c, where L is a lattice of all subsets of the set of atoms
- $\mathcal{T}_P$ is $\subseteq_i$-monotone and $\mathcal{T}_P(I, I) = (T_P(I), T_P(I))$
- $\mathcal{T}_P$ approximates T_P

Back to logic programming

- Models — prefixpoints of T_P
- Partial supported models — fixpoints of $\mathcal{T}_P$
- Supported models — complete fixpoints of $\mathcal{T}_P$ (fixpoints of T_P)
- $LF(\mathcal{T}_P)$ — approximates all supported models of P
- Partial stable models — fixpoints of $\mathcal{T}_P^s$
- Stable models — complete fixpoints of $\mathcal{T}_P^s$ (fixpoints of T_P^s)
- $WFF(\mathcal{T}_P)$ — well-founded model of P
- $WFF(\mathcal{T}_P)$ — approximates all stable fixpoints of P
- $LF(\mathcal{T}_P) \subseteq_i WFF(\mathcal{T}_P)$
- 4-valued operator $\mathcal{T}_P$ — approximation on L^2

How to pick an approximation

- In logic programming, $\mathcal{T}_P$ appeared naturally in the 3-valued and 4-valued settings
- Are there other "natural" approximation operators?
- Are there general ways to construct an approximation given a lattice operator?
- The setting of L^c makes it easier (and, as in the case of LP, we restrict to it now)
- Let A and B be approximating operators on L^c

$$A \preceq B \text{ if for every } (x, y) \in L^c,\ A(x, y) \leq_i B(x, y).$$

- $\preceq$ is a complete ordering of approximations for O
- Thus, it has a largest element, Ult_O — *ultimate* approximation to O
- Ultimate approximation to O satisfies:

$$Ult_O(x, y) = (glb(\{O(z) : x \leq z \leq y\},\ lub(\{O(z) : x \leq z \leq y\})$$

Properties of approximations

- If $A \preceq B$:
  - every complete fixpoint of A is a fixpoint of B
  - every complete stable fixpoint of A is a stable fixpoint of B
  - $LF(A) \leq_i LF(B)$
  - $WFF(A) \leq_i WFF(B)$

Properties of approximations

- Complete fixpoints of Ult_O = fixpoints of O
  - for every fixpoint x of O:
    $LF(Ult_O) \leq_i (x, x)$ or $LF(Ult_O)_1 \leq x \leq LF(Ult_O)_2$
  - *Best* common approximation of all fixpoints of O given by an approximating operator
- *Ultimate stable fixpoints* of O = complete Ult_O-stable fixpoints of O
  - for every approximation A of O: A-stable fixpoints of O are ultimate stable fixpoints of O
  - for every approximation A of O: $WFF(A) \leq_i WFF(Ult_O)$
  - for every ultimate stable fixpoint x of O:
    $WFF(Ult_O) \leq_i (x, x)$ or $WFF(Ult_O)_1 \leq x \leq WFF(Ult_O)_2$

Back to logic programming again

- Ultimate approximation to T_P more precise than $\mathcal{T}_P$
- In general — strictly more precise
- Every Gelfond-Lifschitz stable model of P is an ultimate stable model of P
- Ultimate well-founded semantics is more precise (in general, strictly) than VGRS well founded semantics
- Ultimate well-founded semantics approximates all G-L stable models of P (as each of them is an ultimate stable model of P)
- Ultimate approximation to T_P does not depend on the syntactic form of a program P
- Ultimate semantics (stable and wf) do not depend on a syntactic form of a program:

  Programs
  $\{p\}$ and $\{p \leftarrow p, \quad p \leftarrow \neg p\}$
  have the same ultimate approximation operator (but different operators $\mathcal{T}_P$)

Back to logic programming again

- Downside: higher computational complexity
  - existence of an ultimate stable model — Σ_2^P-complete (rather than NP-complete)
  - deciding status of an w/r to ultimate well-founded semantics — in Δ_2^P (rather than polynomial)
- However: for broad classes of programs the complexity does not grow
- All programs such that for every atom p at least one of the following conditions holds:
  - the body of each clause with the head p consists of at most two elements;
  - the body of each clause with the head p contains at most one positive literal;
  - the body of each clause with the head p contains at most one negative literal.

Concluding comments

- There is a purely algebraic foundation to semantics of logic programs
- Strips the theory of logic programming of inessential and potentially confusing details
- Proofs gain on clarity and simplicity
- The approach works equally well in streamlining the treatment of semantics of default logic and autoepistemic logics
- The approach allows to see a precise connection between LP, DL and AEL and strikingly close correspondence between these formalisms
- Perhaps most important contributions:
  - algebraic account of the well-founded semantics
  - ultimate approximation