Kripke structures on bilattices

Martin Baláž
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
balaz@ii.fmph.uniba.sk

June 1, 2004

Abstract

This paper extends stable model semantics of many valued logics for logic programs with explicit negation. We will use bilattices as truth value space introduced by [4] and studied in [2] and we will build a Kripke structure, which provides a more detailed semantics for logic programs. This approach is used for dynamic logic programs as it is described in [6].

Keywords: Extended logic program, bilattice, Kripke structure.

1 Introduction

The purpose of this paper is to define a Kripke structure associated with an extended logic program. Kripke structure can be viewed as a graph, where nodes are pairs of interpretations (I, J). Each node represents our knowledge about the world. The interpretations approximate the state K of the world, if $I \leq K \leq J$ [1].

In the beginning, we have no knowledge available. In other words an initial interpretation which always assigns the lowest truth value and another interpretation which always assigns the greatest truth value. By applying information contained in the rules of the program, we can bring those interpretations closer to each other. In the case we can not apply any rule, we accept the assumption of a default negation. Our goal is to reach a node of the form (I, I) which only approximates one interpretation I.

2 Lattices and bilattices

The set of truth values is *partially ordered* by a relation, which is reflexive, antisymmetric and transitive. A partially ordered set is a *lattice* if supremum and infimum exist for any two elements. We will use the notations $a \wedge b$ for infimum of a and b and $a \vee b$ for supremum of a and b and we will call $\wedge$ the *meet* and $\vee$ the *join*.

In general the supremum and infimum do not exist for any subset. A lattice L is called *complete* if supremum and infimum exist for any nonvoid subset of L. We will denote infinitary meet and join by $\bigwedge$ and $\bigvee$. The *bottom* is denoted by $\bot$ and the *top* by $\top$.

A lattice L has a *negation operation* if there is a mapping $\sim\colon L \to L$ such that $\sim\sim x = x$ and $x \leq y$ implies $\sim y \leq \sim x$.

The set L^S of all mappings from a set S to a lattice L with ordering $\leq$ defined as $f \leq g$ if $f(s) \leq g(s)$ for all $s \in S$ is a lattice. If L has negation operation, this induces a negation operator on L^S according to $(\sim f)(s) = \sim f(s)$.

Example 1. Sets $\{0, 1\}$, $\{0, \frac{1}{2}, 1\}$ and interval $\langle 0, 1\rangle$ with the linear ordering on real numbers are complete lattices. Generally, a lattice can have more negation operators. The mapping $x \mapsto 1 - x$ is a negation operator which will be used in the following examples.

A *bilattice* is a set B together with two partial orderings $\leq_t$ and $\leq_i$ such that $(B, \leq_t)$ and $(B, \leq_i)$ are lattices. The relation $\leq_t$ is an ordering of the degree of truth. The ordering $\leq_i$ is thought of as ranking degree of information.

A bilattice B has a *negation operation* if there is a mapping $\sim\colon B \to B$ such that it reverses the $\leq_t$ ordering, leaves unchanged the $\leq_i$ ordering and $\sim\sim a = a$. A bilattice B has a *conflation operation* if there is a mapping $-\colon B \to B$ such that it leaves the $\leq_t$ ordering unchanged, reverses the $\leq_i$ ordering and $--a = a$.

We say that $a \in B$ is *exact* if $a = -a$ and $a \in B$ is *consistent* if $a \leq_i -a$. The bilattice $(B, \leq_t, \leq_i)$ is *complete* if $(B, \leq_t)$ and $(B, \leq_i)$ are complete lattices.

The set B^S of all mappings from S to a bilattice B with the orderings $\leq_t$ and $\leq_i$ defined as $f \leq_t g$ (resp. $f \leq_i g$) if $f(s) \leq_t g(s)$ (resp. $f(s) \leq_i g(s)$) for all $s \in S$ is a bilattice. If B has a negation operation (resp. conflation operation), this induces the negation operation (reps. the conflation operation) on B^S according to $(\sim f)(s) = \sim f(s)$ (resp. $(-f)(s) = -f(s)$).

3 Extended language

We need to extend first order language [5] to contain explicit negation and nullary connectives. An *extended alphabet* $\mathcal{A}_L$ contains variables, constants, function and predicate symbols, nullary connectives from L, unary $\{not, \neg\}$ and binary $\{\wedge, \vee, \rightarrow\}$ connectives, quantifiers $\{\forall, \exists\}$ and punctuation symbols.

A variable or a constant is a term. If f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term. If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is an *atom*. *Objective literal* is an atom or the explicit negation of an atom. *Default literal* is an objective literal or the default negation of an objective literal.

An objective literal or a nullary connective is a *(well-formed) formula*. If F and G are formulas, then so are $(not\ F)$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$. If F is a formula and x is a variable, then $(\forall x F)$ and $(\exists x F)$ are formulas. An *extended first order language* $\mathcal{L}_L$ given by an extended alphabet $\mathcal{A}_L$ consists of the set of all formulas constructed from the symbols of the alphabet $\mathcal{A}_L$.

An *extended rule* is a formula of the form

$$\forall x_1 \ldots \forall x_m (L_1 \wedge \cdots \wedge L_n \rightarrow L)$$

where L is a default literal, each L_i is a default literal or nullary connective and $x_1 \ldots x_m$ are all the variables occurring in $L_1 \wedge \cdots \wedge L_n$. We will denote this clause by

$$L \leftarrow L_1, \ldots, L_n$$

L is called the *head* and $L_1 \wedge \cdots \wedge L_n$ is called the *body* of the extended rule. An *extended program* is a finite set of extended rules.

Example 2. $P = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, b \leftarrow not\ a, a \leftarrow not\ b, \neg b \leftarrow b\}$ is an extended program.

4 Interpretations and models

A *ground term* is a term containing no variables. Similarly, a *ground formula* is a formula not containing variables. The *Herbrand universe* $\mathcal{U}$ is the set of all ground terms and the *Herbrand base* $\mathcal{B}$ is the set of all ground objective literals. The *Herbrand interpretation* is a mapping from $\mathcal{B}$ to a complete lattice L.

An interpretation I can be extended to a *formula valuation* v_I (wrt I) of the closed formulas such that each objective literal is given its truth value according to I and each nullary connective is assigned to itself. If the formula

has the form *not* F, $F \wedge G$, $F \vee G$, than the truth value of the formula is $v_{\sim I}(F)$, $v_I(F) \wedge v_I(G)$, $v_I(F) \vee v_I(G)$. If the formula has the form $F \rightarrow G$, then the truth value of the formula is $\top$ if $v_I(F) \leq v_I(G)$. Otherwise, its truth value is $\bot$. If the formula has the form $\forall x F$, then the truth value of the formula is $\bigwedge_{d \in \mathcal{U}} v_I(F\{x/d\})$, where $\{x/d\}$ is the substitution that replaces each x in F by d. If the formula has the form $\exists x F$, then the truth value of the formula is $\bigvee_{d \in \mathcal{U}} v_I(F\{x/d\})$.

So far corresponding positive and negative objective literals have been independent. Next property constrains their truth values. The Herbrand interpretation I is called *coherent*, if $I(\neg A) \leq \sim I(A)$ holds for every atom A.

A Herbrand interpretation I *models an extended rule* c if $v_I(c) = \top$ and *models an extended program* P if I models each $c \in P$.

Example 3. $I_1 = \{a \mapsto \frac{3}{4}, b \mapsto \frac{1}{4}, \neg a \mapsto 0, \neg b \mapsto \frac{1}{4}\}$ is a coherent model of the program $P = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, b \leftarrow not\ a, a \leftarrow not\ b, \neg b \leftarrow b\}$.
$I_2 = \{a \mapsto \frac{1}{4}, b \mapsto \frac{3}{4}, \neg a \mapsto 0, \neg b \mapsto \frac{3}{4}\}$ is not a coherent model of P.

5 Stable models

An *extended definite program* does not contain any default negation. Let P be an extended definite program, $\mathcal{I}$ be a set of all interpretations of P and L be an objective literal. The operator $T_P : \mathcal{I} \rightarrow \mathcal{I}$ is defined as follows:

$$T_P(I)(L) = I(L) \vee \bigvee_{\substack{c \in P \\ head(c) = L}} v_I(body(c))$$

Let $T_P \uparrow 0(I) = \bot$. $T_P \uparrow \alpha(I) = T(T \uparrow (\alpha - 1)(I))$, if α is successor ordinal and $T_P \uparrow \alpha(I) = \bigvee_{\beta < \alpha} T \uparrow \beta(I)$, if α is limit ordinal.

Theorem 5.1. *Let P be an extended definite logic program. Then $T_P \uparrow \omega(\bot)$ is the least model of P.*

An *extended normal program* can contain a default negation in the bodies of the rules. Let I be an interpretation, P be an extended normal program. P_I is a program obtained from P by replacing all negative default literals L with nullary connectives $v_I(L)$. If I coincides with minimal model of P_I, we say that I is *stable interpretation* for P.

In the case of two valued logic the definition of stable interpretations [3] was extended to answer sets. Because in many valued logic the interpretation is defined as a function and not as a set, we will use the term "stable interpretation" instead of the term "answer set".

Lemma 5.2. *Every stable interpretation of an extended normal program is its model.*

An *extended generalized program* can contain a default negation in the bodies and also in the heads of the rules. Let P be an extended generalized program. We will denote P^+ rules with a positive default literal in the head and P^- rules with a negative default literal in the head. If I is a stable interpretation for P^+, we say that I is a stable interpretation for P.

6 Approximations

In the previous section, we have not used the rules in P^- for computation of the stable interpretation. They are only used as a consistency check for a model. Operators transforming approximations can use all of them. The rules from P^+ determine the lower bound and the rules from P^- the upper bound.

Let $(L, \leq)$ be a lattice. A bilattice $B(L)$ based on a lattice L is a set L^2 with two partial orderings:

- $(a_1, a_2) \leq_t (b_1, b_2) \Leftrightarrow a_1 \leq b_1, a_2 \leq b_2$
- $(a_1, a_2) \leq_i (b_1, b_2) \Leftrightarrow a_1 \leq b_1, b_2 \leq a_2$

If L has a negation operator, then $B(L)$ has a negation operator $\sim (a, b) = (\sim b, \sim a)$. The mapping $-(a, b) = (b, a)$ is a conflation operator for $B(L)$.

The *Herbrand approximation* is mapping from $\mathcal{B}$ to $B(L)$. Because the bilattice $(B(L))^{\mathcal{B}}$ is isomorphic with the lattice $B(L^{\mathcal{B}})$, the Herbrand approximation can be also viewed as a pair of Herbrand interpretations.

The definition of a bilattice based on a lattice implies that $A = (I, J)$ is exact if $I = J$ and consistent if $I \leq J$.

The approximation $A = (I, J)$ can be extended to a *formula valuation* v_A (wrt A) of the closed formulas such that each objective literal is given its approximation according to I and each nullary connective is assigned to itself. If the formula has the form *not* F, $F \wedge G$, $F \vee G$, then the truth value of the formula is $v_{\sim A}(F)$, $v_A(F) \wedge v_A(G)$, $v_A(F) \vee v_A(G)$. If the formula has the form $F \rightarrow G$, then the truth value of the formula is $\top$ if $v_A(F) \leq v_A(G)$. Otherwise, its truth value is $\bot$. If the formula has the form $\exists x F$, then the truth value of the formula is $\bigvee_{d \in \mathcal{U}} v_A(F\{x/d\})$, where $\{x/d\}$ is the substitution that replaces x by d. If the formula has the form $\forall x F$, then the truth value of the formula is $\bigwedge_{d \in \mathcal{U}} v_A(F\{x/d\})$.

The Herbrand approximation A is called *coherent* if $A(\neg B) \leq_t \sim A(B)$ holds for every atom B.

7 Kripke structures

Operators transforming approximations do not offer tools for a fine-grained semantics needed in the case of dynamic programs. Not all rules applied within one step of the iteration stay relevant after their update by a newer program. Therefore we should define a structure containing more fine-grained dependences between individual literals.

Example 4. Let $P = \{a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}, b \leftarrow not\ a, a \leftarrow not\ b, \neg b \leftarrow b\}$. We start with the approximation $w_1 = (\bot, \top)$. By applying information contained in the rules $a \leftarrow \frac{1}{4}, b \leftarrow \frac{1}{4}$, we can increase the lower bound of a and lower bound of b to $\frac{1}{4}$. By applying the coherence principle, we can decrease the upper bound of $\neg a$ and $\neg b$ to $\frac{3}{4}$. The rule $\neg b \leftarrow b$ increases the lower bound for $\neg b$ to $\frac{1}{4}$.

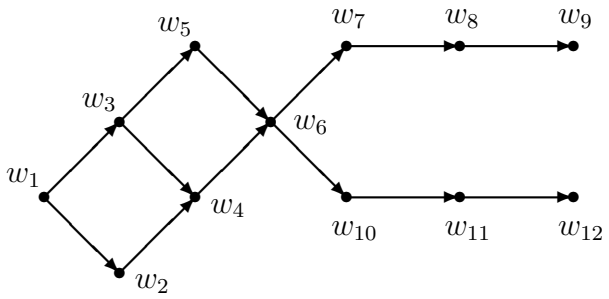
Further in the resulting interpretation $w_4 = \{a \in (\frac{1}{4}, \top), b \in (\frac{1}{4}, \top), \neg a \in (\bot, \frac{3}{4}), \neg b \in (\frac{1}{4}, \frac{3}{4})\}$, no rule can be directly applied. However we have two rules $b \leftarrow not\ a, a \leftarrow not\ b$ with a default negation in the body of which truth value is not exact. Now we can accept default negation under which the upper bound is decreased to the lower bound. We have two choices: $a \in (\frac{1}{4}, \frac{1}{4})$ because the rule $b \leftarrow not\ a$ or $b \in (\frac{1}{4}, \frac{1}{4})$ because the rule $a \leftarrow not\ b$.

In the first case, we get a new interpretation $w_8 = \{a \in (\frac{1}{4}, \frac{1}{4}), b \in (\frac{3}{4}, \top), \neg a \in (\bot, \frac{3}{4}), \neg b \in (\frac{1}{4}, \frac{1}{4})\}$ by applying the rule $b \leftarrow not\ a$ and by applying coherence principle . The rule $\neg b \leftarrow b$ causes an inconsistency for $\neg b \in (\frac{3}{4}, \frac{1}{4})$ then.

In the second case, we get a new interpretation $w_{11} = \{a \in (\frac{3}{4}, \top), b \in (\frac{1}{4}, \frac{1}{4}), \neg a \in (\bot, \frac{1}{4}), \neg b \in (\frac{1}{4}, \frac{3}{4})\}$ by applying the rule $a \leftarrow not\ b$ and by applying coherence principle.

In this state, no rule can be applied and no default assumption bringing new information can be accepted. Therefore we can decrease all upper bounds to the lower bounds and we get an exact approximation $w_{12} = \{a \in (\frac{3}{4}, \frac{3}{4}), b \in (\frac{1}{4}, \frac{1}{4}), \neg a \in (\bot, \bot), \neg b \in (\frac{1}{4}, \frac{1}{4})\}$.

Figure 1: Kripke structure associated with the program P



$$
\begin{array}{rcllll}
w_1 & = & \{a \in (\bot, \top), & b \in (\bot, \top), & \neg a \in (\bot, \top), & \neg b \in (\bot, \top)\} \\
w_2 & = & \{a \in (\frac{1}{4}, \top), & b \in (\bot, \top), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\bot, \top)\} \\
w_3 & = & \{a \in (\bot, \top), & b \in (\frac{1}{4}, \top), & \neg a \in (\bot, \top), & \neg b \in (\bot, \frac{3}{4})\} \\
w_4 & = & \{a \in (\frac{1}{4}, \top), & b \in (\frac{1}{4}, \top), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\bot, \frac{3}{4})\} \\
w_5 & = & \{a \in (\bot, \top), & b \in (\frac{1}{4}, \frac{3}{4}), & \neg a \in (\bot, \top), & \neg b \in (\frac{1}{4}, \frac{3}{4})\} \\
w_6 & = & \{a \in (\frac{1}{4}, \top), & b \in (\frac{1}{4}, \top), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\frac{1}{4}, \frac{3}{4})\} \\
w_7 & = & \{a \in (\frac{1}{4}, \frac{1}{4}), & b \in (\frac{1}{4}, \top), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\frac{1}{4}, \frac{3}{4})\} \\
w_8 & = & \{a \in (\frac{1}{4}, \frac{1}{4}), & b \in (\frac{3}{4}, \top), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\frac{1}{4}, \frac{1}{4})\} \\
w_9 & = & \{a \in (\frac{1}{4}, \frac{1}{4}), & b \in (\frac{3}{4}, \frac{1}{4}), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\frac{3}{4}, \frac{1}{4})\} \\
w_{10} & = & \{a \in (\frac{1}{4}, \top), & b \in (\frac{3}{4}, \frac{1}{4}), & \neg a \in (\bot, \frac{3}{4}), & \neg b \in (\frac{1}{4}, \frac{3}{4})\} \\
w_{11} & = & \{a \in (\frac{3}{4}, \top), & b \in (\frac{1}{4}, \frac{1}{4}), & \neg a \in (\bot, \frac{1}{4}), & \neg b \in (\frac{1}{4}, \frac{3}{4})\} \\
w_{12} & = & \{a \in (\frac{3}{4}, \frac{3}{4}), & b \in (\frac{1}{4}, \frac{1}{4}), & \neg a \in (\bot, \bot), & \neg b \in (\frac{1}{4}, \frac{1}{4})\}
\end{array}
$$

Definition 7.1. Kripke structure associated with an extended program P is a graph $\mathcal{K}_P = (W, \rho)$, where:

- W is a set of all coherent approximations for P.
- $\rho = \rho_1 \cup \rho_2 \cup \rho_3 \subseteq W^2$, $(w_1, w_2) \in \rho$ if w_1 is consistent and $w_1 \neq w_2$. If the node w is the source of a ρ_i-edge, it is not the source of any ρ_j-edge for $i < j$.
  - $((I_1, J_1), (I_2, J_2)) \in \rho_1$ if exists $c \in P$ with objective literal in the head and
    - $I_2(L) = I_1(L) \vee v_{(I_1, J_1)}(body(c))$ for $L = head(c)$
    - $I_2(L) = I_1(L)$ for $L \neq head(c)$
    - $J_2(\neg L) = J_1(\neg L) \wedge \sim v_{(I_1, J_1)}(body(c))$ for $L = head(c)$

* $J_2(\neg L) = J_1(\neg L)$ for $L \neq head(c)$

– $((I_1, J_1), (I_2, J_2)) \in \rho_1$ if exists $c \in P$ with default literal in the head and
  * $I_2(L) = I_1(L)$ for all L
  * $J_2(L) = J_1(L) \wedge \sim v_{(I_1,J_1)}(body(c))$ for $L = not\ head(c)$
  * $J_2(L) = J_1(L)$ for $L \neq not\ head(c)$

– $((I_1, J_1), (I_2, J_2)) \in \rho_2$ if exists $c \in P$ such that $v_{(I_1,J_1)}(body(c)) \leq v_{(I_1,J_1)}(head(c)) < v_{I_1}(body(c))$ and
  * $I_2(L) = I_1(L)$ for all L
  * $J_2(L) = I_1(L)$ for $not\ L \in body(c)$
  * $J_2(L) = J_1(L)$ for $not\ L \notin body(c)$

– $((I_1, J_1), (I_2, J_2)) \in \rho_3$ if
  * $I_2(L) = I_1(L)$ for all L
  * $J_2(L) = I_1(L)$ for all L

Let $e = (u, v)$ be an edge. We say that u is the source and v is the target for e. Let $\sigma = e_1, \ldots, e_n$ be a path. We say, that σ is u-rooted, if u is the source for e_1. We say, that σ terminates in v, if v is the target for e_n and is not the source for any edge.

Theorem 7.1. *Let $\sigma \in \mathcal{K}_P$ be a $\bot$-rooted path terminated in exact approximation $w = (I, I)$. Then I is a coherent stable model for P.*

Sketch of proof. The transformed program P_I will do the same thing as default assumptions in Kripke structure. The operator T_P can use the transformed rules corresponding to the rules in the path. The path terminates in exact approximation, that means there is not an edge leading to an inconsistent world. Therefore the interpretation I is a model for P. We have used in the path only direct conclusions and default assumptions, the interpretation I must conincide with minimal model of P_I. □

Theorem 7.2. *Let I be a coherent stable model for P. Then exists $\bot$-rooted path $\sigma \in K_P$ which terminates in the exact approximation $w = (I, I)$.*

Sketch of proof. Let I be a coherent stable model. The transformed program P_I represent default assumptions in $\rho_2 \cup \rho_3$-edges. If we use $\rho_1 - edges$ corresponding to the used tranformed rules in $T_P \uparrow \omega$, we get the path terminating in the exact approximation (I, I). □

8 Conclusion

In this paper, it has been described how Kripke structures can be built. This semantics characterization is more detailed than simple stable models. The computation of stable models included in the Kripke structure is using imformation in the rules with positive and negative default literal in the head.

References

[1] Marc Denecker, Victor Marek, and Miroslaw Truszczynski. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning.

[2] Melvin C. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11(2):91–116, 1989.

[3] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Fifth Int'l Conf.Symp. on Logic Programming*, pages 1070–1080. Seattle, 1988.

[4] Matthew L. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[5] J. W. Lloyd. *Foundations of Logic Programming, Second Edition.* Springer-Verlag, 1987.

[6] Ján Šefránek. A Kripkean semantics for dynamic logic programming. *Lecture Notes in Computer Science*, 1955:469–??, 2000.

Martin Baláž (Mgr) is a postgraduate student at the Faculty of Mathematics, Physics and Informatics of the Comenius University in Bratislava. His PhD-thesis supervisor is professor Ján Šefránek.