Derivation of Proof Methods by Abstract Interpretation

Giorgio Levi and Paolo Volpe

Abstract

We study the application of abstract interpretation to the design of inductive methods for verifying properties of logic programs.
We give a unified view of inductive assertion-based proof methods for logic programs, by systematically deriving them in a uniform way. The resulting verification framework allows us to reconstruct several existing verification methods, to understand the relation among them in terms of abstractions and to tackle the problem of establishing the completeness of a method.

Keywords: Inductive proof methods, abstract interpretation, abstract domains.

1 Introduction

Abstract interpretation theory [15] has been used recently in the framework of debugging, diagnosis and validation [12, 11, 7] and to derive proof methods [14]. In fact, the problem of verification of logic programs can be very naturally approached by abstract interpretation, since it is a typical semantics-based task.
In this paper we use abstract interpretation as a tool for systematically deriving sufficient partial correctness conditions for logic programs. The specifications can simply be viewed as a suitable intended abstract semantics and the existing notions of correctness and related verification methods [18, 5, 17, 3, 1] be explained as different abstractions.
Abstract interpretation allows to compare different semantics by reasoning in terms of abstraction. In the case of logic programs verification, this makes easier to compare the different techniques and to show the essential differences. Moreover we can systematically derive the (optimal) abstract semantics from the concrete one and from the abstraction. The resulting abstract semantics is a correct approximation of the concrete semantics by construction and no additional "correctness" theorems need to be proved. Thus, given a specific property (abstraction), the corresponding verification conditions are systematically derived from the framework and guaranteed to be indeed sufficient partial correctness conditions. In addition the verification method is guaranteed to be complete, if the abstraction is precise (also called complete in [15]).
Properties we deal with in this paper are abstractions of SLD-trees. The verification framework is based on a hierarchy of semantics [10, 9], whose collecting semantics is defined in a concrete domain of SLD-derivations. A generic (abstract) semantics in the

G. Levi and P. Volpe are with the Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy. e-mail: {levi, volpep}@di.unipi.it. Phone: +39-50-887248. Fax: +39-50-887226.

hierarchy, corresponding to the asbtraction α, is the least fixpoint of an operator T_P^α, systematically derived from the corresponding operator of the collecting semantics. In Section (2), we show the general results about sufficient partial correctness conditions and completeness of the verification methods. In the next Sections, we show some instances of the verification framework, with the aim of reconstructing existing verification methods. We start with more abstract properties, moving from the success behaviour (Section (3)), to the Input-Output behaviour and to the call behaviour (Section (4)). In particular we reconstruct the Drabent and Maluszynski's method [18], the Bossi and Cocco's method [5], and various verification methods reviewed in [3]. Finally in Section (5), we consider the case of intensional specifications given in a formal specification language.
We assume familiarity with the standard notions of lattice theory [4], abstract interpretation [15], logic programming [2] and verification methods [1, 13].

2 Description of the framework

Like in [10, 9], the concrete domain is given by the set of *pure collections*, $\mathbb{D} = (Pred \rightarrow \widetilde{\wp}(\mathcal{SLD}), \leq)$, where a pure collection is a function from the set $Pred = \{p(\mathbf{x}) \mid p$ predicate symbol and $\mathbf{x}$ a tuple of distinct variables$\}$ of *pure atoms* to an element of $\widetilde{\wp}(\mathcal{SLD})$, that is the suffix-closed sets of *SLD*-derivations. The order $\leq$ is the pointwise extension of the subset order of $\widetilde{\wp}(\mathcal{SLD})$.
Given a program P, we associate to it a continuous function $T_P : \mathbb{D} \rightarrow \mathbb{D}$ (also defined in [10]). The semantics $[\![P]\!]$ of P is defined as $lfp(T_P) = T_P^\omega = \lambda p(\mathbf{x}).\{\delta \in \mathcal{SLD} \mid first(\delta) = p(\mathbf{x}), clauses(\delta) \subseteq P\}$.

The main idea of this paper is to view (extensional or intensional) specifications as an abstract domain $(\mathcal{A}, \sqsubseteq)$. Under reasonable hypotheses, this relationship can be formalized through a Galois connection between $(\mathbb{D}, \leq)$ and $(\mathcal{A}, \sqsubseteq)$. Then each element d of $\mathbb{D}$ is *correct* with respect to specifications $\mathcal{S}$ such that $\alpha(d) \sqsubseteq \mathcal{S}$. In this paper, (abstract) domains of specifications are structure $(Pred \rightarrow \mathbb{S}, \sqsubseteq)$, whose elements Ψ can be thought as a set $\{\Psi_p\}_{p \in Pred}$, where each Ψ_p is a specification for what we observe in the set of derivations started by predicate p. In fact the generic correctness condition can be expressed as $\forall \delta \in [\![P]\!](p(x))\ \delta$ verifies Ψ_p, which can be rewritten, for a suitable *observable* α [10], as

$$\forall \delta \in [\![P]\!](p(x))\ \alpha(\delta) \sqsubseteq \Psi_p. \tag{1}$$

We can study then the verification problem using methods and results of abstract interpretation. In fact the problem of checking whether a program P verifies a specification $\mathcal{S}$ in $\mathcal{A}$, can be rephrased in abstract interpretation terms as

$$[\![P]\!] \leq \gamma(\mathcal{S}) \quad \text{or, equivalently, } \alpha([\![P]\!]) \sqsubseteq \mathcal{S}. \tag{2}$$

Since $[\![P]\!]$ is defined as the least fixpoint of the operator T_P, a sufficient condition for (2) to hold is

$$T_P(\gamma(\mathcal{S})) \leq \gamma(\mathcal{S}) \quad \text{or, equivalently, } T_P^\alpha(\mathcal{S}) \sqsubseteq \mathcal{S}, \tag{3}$$

where $T_P^\alpha = \alpha \circ T_P \circ \gamma$, is the *best abstraction* of T_P in $\mathcal{A}$. In fact $T_P^\alpha(\mathcal{S}) \sqsubseteq \mathcal{S}$ implies $lfp(T_P^\alpha) \sqsubseteq \mathcal{S}$ and, since $\alpha(lfp(T_P)) \sqsubseteq lfp(T_P^\alpha)$ (because α is correct), the condition $\alpha([\![P]\!]) \sqsubseteq \mathcal{S}$ can be derived.

Condition (3) can often be unfolded and transformed into a verification condition for P and $\mathcal{S}$, which may be the base for an inductive proof method. Obviously how to do it depends on the abstract domain and the abstraction. In this paper we show how the verification conditions of several well known methods can be derived.

If an inductive proof method is *complete*, then, when the program P is correct with respect to specification $\mathcal{S}$, there exists a property $\mathcal{R}$, stronger than $\mathcal{S}$, which verifies the verification condition. We have proved that, for verification conditions which have the form of condition (3) for a suitable α, the derived method is complete if and only if the abstraction is *precise with respect to* T_P, that is if $\alpha(\mathit{lfp}(T_P)) = \mathit{lfp}(T_P^{\alpha})$, as follows from next lemma.

Lemma 2.1 *Let (C, A, α, γ) be a Galois connection between the complete lattices C and A. Let $F : C \to C$ be a monotonic operator on C and $F^{\alpha} = \alpha \circ F \circ \gamma : A \to A$ be its best abstraction on A. Then*

$$\alpha(\mathit{lfp}(F)) \sqsubseteq \varphi \textit{ implies } \exists \psi \sqsubseteq \varphi \; F^{\alpha}(\psi) \sqsubseteq \psi$$

if and only if (C, A, α, γ) is precise with respect to F.

A sufficient condition for precision, generally easier to check, is *full precision*, that is $\alpha \circ T_P = T_P^{\alpha} \circ \alpha$. In this paper we always show the completeness of some methods by showing the full precision of the underlying abstraction.

3 Success behaviour of programs

Let us focus on the program behaviour with respect to the success of non-ground atoms. Let us suppose a set φ_p of atoms (the *extensional specification of a property*) associated to each predicate p. The program P is *success-correct* with respect to *success-properties* $\{\varphi_p\}_{p \in \mathit{Pred}}$ iff

$$\forall p(t) \in \mathit{Atoms}\; p(t) \overset{\theta}{\rightsquigarrow} \square \text{ implies } p(t)\theta \in \varphi_p.$$

Now a set $\{\varphi_p\}_{p \in \mathit{Pred}}$ is just a function $\varphi : \mathit{Pred} \to \wp(\mathit{Atoms})$. Then, the domain $\mathcal{C} = (\mathit{Pred} \to \wp(\mathit{Atoms}), \leq)$, ordered by the pointwise extension of subset order, can be viewed as the domain of specifications of the success behaviour of programs. The abstraction from the basic domain $\mathbb{D}$ is given by the function

$$\alpha(D) = \lambda p(\mathbf{x}).\{p(\mathbf{x})\theta \mid \delta \in D(p(\mathbf{x})), \partial_\theta(\delta) \text{ successful}\}, \tag{4}$$

where $\partial_\theta(\delta)$ is the instantiation of derivation δ with θ. α is indeed additive ($\alpha(D_1 \vee D_2) = \alpha(D_1) \vee \alpha(D_2)$), hence there exists an adjoint function $\gamma : \mathcal{C} \to \mathbb{D}$, giving a Galois connection between $\mathbb{D}$ and $\mathcal{C}$. Success-correctness can be rephrased as $\alpha([\![P]\!]) \leq \varphi$.
The best abstraction $T_P^{\mathcal{C}} = \alpha \circ T_P \circ \gamma$ in $\mathcal{C}$ of the concrete function T_P can explicitly be defined as

$$T_P^{\mathcal{C}}(I) = \lambda p(\mathbf{x}).\; \{p(\mathbf{t})\theta \mid \begin{array}{l} p(\mathbf{t}) \leftarrow p_1(\mathbf{t}_1), \ldots, p_n(\mathbf{t}_n) \in P, \\ \forall i \in \{1, \ldots, n\}\; p_i(\mathbf{t}_i)\theta \in I(p_i(\mathbf{x}))\}. \end{array}$$

and the abstract interpretation is shown to be *full precise*.

Lemma 3.1 *Let $\alpha : \mathbb{D} \to \mathcal{C}$ be defined as in (4). Then there exists a function $\gamma : \mathcal{C} \to \mathbb{D}$ such that $(\mathbb{D}, \mathcal{C}, \alpha, \gamma)$ is a Galois connection and is* full precise *with respect to T_P.*

As a consequence of lemma (3.1), the least fixpoint of $T_P^{\mathcal{C}}$ exists and $lfp(T_P^{\mathcal{C}}) = \alpha(\llbracket P \rrbracket)$. Indeed the semantics $(\mathcal{C}, T_P^{\mathcal{C}})$ is just a reformulation of the *C-semantics* [8, 19].
Recalling Section (2), a sufficient condition for success-correctness is $T_P^{\mathcal{C}}(\varphi) \leq \varphi$, from which a verification condition can be constructively derived by unfolding $T_P^{\mathcal{C}}$.

Theorem 3.2 *Let P be a logic program and $\{\varphi_p\}_{p \in Pred}$ be a success property. P is success-correct with respect to $\{\varphi_p\}_{p \in Pred}$ if for each clause $p(\mathbf{t}) \leftarrow p_1(\mathbf{t}_1), \ldots, p_n(\mathbf{t}_n)$ of P it is true that*

$$\forall \theta \bigwedge_{i=1}^{n} p_i(\mathbf{t}_i)\theta \in \varphi_{p_i} \; \textbf{\textit{implies}} \; p(\mathbf{t})\theta \in \varphi_p. \tag{5}$$

Example 3.1 Let P be the program

```
app([],Y,Y).
app([X:Xs],Ys,[X:Zs])←app(Xs,Ys,Zs).
```

Let $\varphi_{app} = \{app(t_1, t_2, t_3) \mid t_3$ ground implies t_1 and t_2 ground$\}$. It can easily be checked that for each θ, $app([], Y, Y)\theta \in \varphi_{app}$ and, if $app(Xs, Ys, Zs)\theta \in \varphi_{app}$, then $app([X : Xs], Ys, [X : Zs])\theta \in \varphi_{app}$. By theorem (3.2), if $app(t_1, t_2, t_3) \overset{\theta}{\rightsquigarrow} \Box$ and $t_3\theta$ is ground, then $t_1\theta$ and $t_2\theta$ are ground. ▮

By lemmas (2.1) and (3.1), it follows that the method is complete, that is if P is success-correct with respect to $\{\varphi_p\}_{p \in Pred}$, then there exists a success property $\{\psi_p\}_{p \in Pred}$ with $\forall p \; \psi_p \subseteq \varphi_p$, which verifies condition (5) for each clause of P.

4 Input/Output behaviour and call behaviour

The input/output behaviour, i.e., the relation between the arguments of a predicate at call time and their instantiation in case of success, can be specified by a set $\eta_q \subseteq Atoms \times Subst$ associated to each predicate q. The set of properties $\{\eta_q\}_{q \in Pred}$ can be viewed then as a specification of I/O patterns for logic programs. A program is *I/O-correct* with respect to *I/O properties* $\{\eta_q\}_{q \in Pred}$ iff

$$\forall p(t) \in Atoms \; p(t) \overset{\theta}{\rightsquigarrow} \Box \text{ implies } (p(t), \theta) \in \eta_p.$$

To capture these observations we can take the domain $\mathcal{S} = (Pred \to \wp(Atoms \times Subst), \leq)$. If $(p(t), \theta) \in \eta_p$, the intended meaning is that the predicate p computes substitution θ if called with argument t. The order is obtained again by pointwise extension of the subset order on $\wp(Atoms \times Subst)$. There exists a Galois connection from $\mathbb{D}$ to $\mathcal{S}$. The abstraction is defined as

$$\begin{aligned} \alpha(D) = \lambda p(\mathbf{x}).\ \{(p(\mathbf{t}), \eta) \mid \; & p(\mathbf{t}) \in Atoms, \delta \in D(p(\mathbf{x})) \text{ successful}, \\ & \theta = res(\delta), \exists \eta = mgu(p(\mathbf{x})\theta, p(\mathbf{t}))\} \end{aligned} \tag{6}$$

Program P is I/O-correct with respect to $\{\eta_q\}_{q \in Pred}$ iff $\alpha(\llbracket P \rrbracket) \leq \eta$. The best abstraction $T_P^{\mathcal{S}}$ of T_P in $\mathcal{S}$ can be explicitly defined, and indeed the semantics $(\mathcal{S}, T_P^{\mathcal{S}})$ is an isomorphic version of *S-semantics* [19]. As before the sufficient condition $T_P^{\mathcal{S}}(\eta) \leq \eta$ can be unfolded and a verification condition for I/O-correctness be obtained.

Theorem 4.1 *Let P be a logic program and $\{\eta_p\}_{p\in Pred}$ be an I/O property. P is I/O-correct with respect to $\{\eta_p\}_{p\in Pred}$ if for each clause $p(\mathbf{t}) \leftarrow p_1(\mathbf{t}_1), \ldots, p_n(\mathbf{t}_n)$ and for each $p(\mathbf{s}) \in Atoms$ it is true that*

$$\rho = mgu(p(\mathbf{t}), p(\mathbf{s})) \ \textbf{\textit{and}} \ \bigwedge_{i=1}^{n} (p_i(\mathbf{t}_i)\rho, \theta_i) \in \eta_{p_i} \ \textbf{\textit{implies}} \ (p(\mathbf{s}), \rho\circ\theta) \in \eta_p, \qquad (7)$$

where $\theta = mgu((p_1(\mathbf{t}_1), \ldots, p_n(\mathbf{t}_n))\rho, (p_1(\mathbf{t}_1)\rho\theta_1, \ldots, p_n(\mathbf{t}_n)\rho\theta_n))$.

The abstraction is precise and then by lemma (2.1), the proof method is complete.

Some verification conditions proposed in the literature (see, for example, [18], [5], [3], [6]), in addition to I/O correctness, take into account a property which is stronger than I/O correctness, i.e. *call correctness*, that is each predicate is called accordingly to a given specification. Since it depends on the selection rule, we assume a leftmost selection rule for *SLD* derivations.
The call and I/O behaviour can be specified by a set $\xi_q \subseteq Atoms\times(Atoms \cup Subst)$ associated to each predicate q. The idea is that, if the pair $(p(t), \theta) \in \xi_p$, then if p is called with argument t, it computes the substitution θ. If $(p(t), q(s)) \in \xi_p$ then a call $q(s)$ is generated in the derivation for $p(t)$ with a leftmost selection rule. A program P is *call-correct* with respect to *call-properties* $\{\xi_q\}_{q\in Pred}$ of *Call* iff for each predicate p

$$p(t) \overset{\theta}{\leadsto} \square \text{ implies } (p(t), \theta) \in \xi_p$$

and

$$p(t) \leadsto_L^* \langle q(s), \mathbf{G}\rangle \text{ implies } (p(t), q(s)) \in \xi_p.$$

A suitable domain is given by $Call = (Pred \rightarrow \wp(Atoms\times(Atoms \cup Subst)), \leq)$. The abstraction is

$$\begin{array}{rll} \alpha(D) = \lambda p(\mathbf{x}). & \{(p(\mathbf{t}), q(\mathbf{s})) \mid & \delta \in D(p(\mathbf{x})), \delta' = \partial_\theta(\delta), \mathit{first}(\delta') = p(\mathbf{t}), \\ & & \mathit{last}(\delta') = \langle q(\mathbf{s}), \mathbf{G}\rangle\} \\ & \bigcup \ \{(p(\mathbf{t}), \sigma) \mid & \delta \in D(p(\mathbf{x})), \delta' = \partial_\theta(\delta) \text{ successful}, \\ & & \mathit{first}(\delta') = p(\mathbf{t}), \mathit{res}(\delta') = \sigma\}. \end{array} \qquad (8)$$

With some effort the corresponding best abstract operator T_P^{Call} can be defined explicitly. No information about calls and successes is lost in the computation of abstract iterates.

Lemma 4.2 *Let $\alpha : \mathbb{D} \rightarrow Call$ be defined as in (8). Then there exists a function $\gamma : Call \rightarrow \mathbb{D}$ such that $(\mathbb{D}, Call, \alpha, \gamma)$ is a Galois connection and is* full precise *with respect to T_P.*

The usual sufficient verification condition is $T_P^{Call}(\xi) \leq \xi$, and because of lemma (4.2), the method is complete.

Theorem 4.3 *Let P be a logic program and $\{\xi_p\}_{p\in Pred}$ be a call-property. P is call-correct with respect to $\{\xi_p\}_{p\in Pred}$ if for each clause $p(\mathbf{t}) \leftarrow p_1(\mathbf{t}_1), \ldots, p_n(\mathbf{t}_n)$ of P it is true that*

$$\begin{array}{l} \forall p(\mathbf{s}) \in Atoms,\ \theta_0 = mgu(p(\mathbf{s}), p(\mathbf{t})),\ \forall\theta_1, \ldots, \theta_n \\ \forall k \leq n \ \textbf{\textit{if}}\ \forall i < k\, (p_i(\mathbf{t}_i)\theta_0\cdots\theta_{i-1}, \theta_i) \in \xi_{p_i} \ \textbf{\textit{and}}\ (p_k(\mathbf{t}_k)\theta_0\cdots\theta_{k-1}, q(\mathbf{r})) \in \xi_{p_k} \\ \hfill \textbf{\textit{then}}\ (p(\mathbf{s}), q(\mathbf{r})) \in \xi_p \\ \textbf{\textit{and}} \\ \hfill \textbf{\textit{if}}\ \forall i \leq n\, (p_i(\mathbf{t}_i)\theta_0\cdots\theta_{i-1}, \theta_i) \in \xi_{p_i} \ \textbf{\textit{then}}\ (p(\mathbf{s}), \theta_0\cdots\theta_n) \in \xi_p. \end{array}$$

4.1 The method of Drabent and Maluszynski

The method of Drabent and Maluszynski [18] can be derived by considering as specifications $\mathcal{DM}$ *pre-post properties* $\{pre^i \multimap post^i\}_{i\in I}$, where, for each $i \in I$, $pre^i \multimap post^i$ is a *basic pre-post specification* $\{pre^i_p \multimap post^i_p\}_{p\in Pred}$, with $pre_p \subseteq Atoms$ and each $post_p \subseteq Atoms \times Atoms$. A program P is $\mathcal{DM}$*-correct* with respect to $\{pre^i \multimap post^i\}_{i\in I}$ iff for each $i \in I$ and each predicate p

$$\forall p(t) \in pre^i_p\, p(t) \overset{\theta}{\leadsto} \Box \text{ implies } (p(t), p(t)\theta) \in post^i_p$$

and

$$\forall p(t) \in pre^i_p\, p(t) \leadsto^*_L \langle q(s), \mathbf{G}\rangle \text{ implies } q(s) \in pre^i_q.$$

The domain of such specifications can be defined as $\mathcal{DM} = (\wp(Pred \to \wp(Atoms)\times\wp(Atoms\times Atoms)), \supseteq)$ and each element $pre \multimap post = \{pre^i \multimap post^i\}_{i\in I}$ is concretizable into an element of *Call*

$$\begin{aligned}\gamma(pre \multimap post) = \lambda p(x).\ &\{(p(t),\theta) \mid \forall i \in I\, p(t) \in pre^i_p \Rightarrow (p(t), p(t)\theta) \in post^i_p\}\\ &\bigcup\{(p(t), q(s)) \mid \forall i \in I\, p(t) \in pre^i_p \Rightarrow q(s) \in pre^i_q\}.\end{aligned}$$

It can be checked that γ has a left adjoint α, hence there is a Galois connection between *Call* and $\mathcal{DM}$. By composing with the Galois connections between $\mathbb{D}$ and *Call*, we soon obtain a Galois connection between $\mathbb{D}$ and $\mathcal{DM}$.

As usual, the verification condition can be extracted from the prefixpoint relation of $T^{\mathcal{DM}}_P$, the best abstraction of T_P. It can be checked that when the property to verify is a single basic pre-post specification, that is a $\mathcal{DM}$ property $\{pre^i \multimap post^i\}_{i\in I}$ with I a singleton, then the verification condition is exactly (an extensional version of) the verification condition of the method of Drabent and Maluszynski.

Theorem 4.4 *Let P be a logic program and $\{pre_p \multimap post_p\}_{p\in Pred}$ be a basic pre-post property. Then P verifies the verification condition of the method of Drabent and Maluszynski with respect to $\{pre_p \multimap post_p\}_{p\in Pred}$ if and only if $T^{\mathcal{DM}}_P(pre \to post) \supseteq pre \to post$.*

The abstraction is precise (it suffices to prove the precision of the abstraction from *Call* and $\mathcal{DM}$ with respect to T^{Call}_P), hence, the derived method is complete.
Notice that this is weaker property than the completeness of the method of Drabent and Maluszynski. In the latter case, the stronger property to be found must be again a pre-post property $\{pre^i \multimap post^i\}_{i\in I}$ with I singleton.

4.2 The method of Bossi and Cocco, types and modes

The method of Bossi and Cocco [5] is obtained by considering as basic pre-post specifications a pair $pre_p \multimap post_p$ for each predicate p, where pre_p and $post_p$ are substitution closed subsets of atoms. A program P is $\mathcal{BC}$*-correct* with respect to $\mathcal{BC}$*−properties* $\{pre^i \multimap post^i\}_{i\in I}$ iff for each $i \in I$ and each predicate p

$$\forall p(t) \in pre^i_p\, p(t) \overset{\theta}{\leadsto} \Box \text{ implies } p(t)\theta \in post^i_p$$

and

$$\forall p(t) \in pre^i_p\, p(t) \leadsto^*_L \langle q(s), \mathbf{G}\rangle \text{ implies } q(s) \in pre^i_q.$$

We can consider as a domain

$$\mathcal{BC} = (\wp(Pred \to \wp_{\uparrow}(Atoms){\times}\wp_{\uparrow}(Atoms)), \supseteq),$$

where $\wp_{\uparrow}(Atoms)$ is the set of substitution closed sets of $Atoms$. Following [3], an element $pre \twoheadrightarrow post$ of $\mathcal{BC}$ can be viewed as an element of $\mathcal{DM}$ through the function $\gamma(pre \twoheadrightarrow post) = \{\lambda p(x).pre^i_p \twoheadrightarrow (Atoms{\times}post^i_p) \mid i \in I, (pre^i \twoheadrightarrow post^i) \in (pre \twoheadrightarrow post)\}$. It gives raise to a Galois connection between $\mathcal{DM}$ and $\mathcal{BC}$. Then a Galois connection $(\mathbb{D}, \mathcal{BC}, \alpha, \gamma)$ is obtained. Like in the previous cases a sufficient condition for correctness can be derived from the relation on $T^{\mathcal{BC}}_P$. The condition obtained in the case of singleton $\mathcal{BC}-$properties, is exactly (an extensional version of) the verification condition of the method of Bossi and Cocco [5].

Theorem 4.5 *Let P be a logic program and $\{pre_p \twoheadrightarrow post_p\}_{p \in Pred}$ be a basic $\mathcal{BC}-$property. Then P verifies the verification condition of the method of Bossi and Cocco if and only if $T^{\mathcal{BC}}_P(pre \to post) \supseteq pre \to post$.*

Also in this case completeness of the method can be shown by proving precision of the abstraction from $\mathcal{DM}$ and $\mathcal{BC}$. The same proviso applies about the difference between the completeness of the derived method derived and the method of Bossi and Cocco.
For what concerns types and modes, we can view the set of type assignments and mode assignments to predicates as further abstractions with respect to $\mathcal{BC}$ and can use the same techniques used to go from $\mathcal{DM}$ to $\mathcal{BC}$. The result is that we completely reconstruct the hierarchy of [3], by reducing the relationships between proof methods to Galois connections between the corresponding domains.

5 Intensional specifications

Specification can also be expressed in a formal specification language. Our idea is that this case corresponds to a further level of abstraction. We will consider the case of success-correctness only. Similar constructions can be given for the other notions of correctness.

Fixed a first order language $\mathcal{L} = \langle \Sigma, \Pi, V \rangle$ and a set $\mathcal{F}$ of formulas (also called *assertions*) of $\mathcal{L}$, we define an atom $p(t_1, \ldots, t_n)$ *to satisfy a property* $\Phi\,[x_1, \ldots, x_n]$ of $\mathcal{F}$, if, fixed a term-interpretation $\mathcal{I} = \langle Terms(\Sigma, V), \Sigma_{\mathcal{I}}, \Pi_{\mathcal{I}} \rangle$, that is the set of non-ground terms seen as an $\mathcal{L}$ structure, for each σ

$$\mathcal{I} \models_{\sigma[x_1,\ldots,x_n \backslash t_1,\ldots,t_n]} \Phi\,[x_1, \ldots, x_n]\,.$$

Example 5.1 For example $app([a]\,, [\,]\,, d)$ *satisfies* the formula $gr(x_3) \Rightarrow gr(x_1) \wedge gr(x_2)$. In fact it is true that $d \in gr_{\mathcal{I}}$ implies $[a] \in gr_{\mathcal{I}}$ and $[\,] \in gr_{\mathcal{I}}$, where $gr_{\mathcal{I}} = Terms(\Sigma, \emptyset)$ is the interpretation of gr. ▮

A program P *verifies* a set of assertions $\{\Theta_p\}_{p \in Pred}$, if $\forall p(t) \in Atoms\ p(t) \overset{\theta}{\rightsquigarrow} \square$ implies $p(t)\theta$ satisfies Θ_p. $\mathcal{F}$ can be preordered by $\Theta \preceq_{\mathcal{I}} \Phi$ iff $\mathcal{I} \models \Theta \Rightarrow \Phi$. We can take the induced equivalence and consider the partial order $\mathcal{A}_{\mathcal{I}} = (Pred \to \mathcal{F}/_{\equiv_{\mathcal{I}}}, \preceq_{\mathcal{I}})$, whose elements are sets $\{\Theta_p\}_{p \in Pred}$, where each Θ_p is a formula of $\mathcal{F}$ with free variables corresponding to arguments of p. The order is given by the pointwise extension of the order

between formulas of $\mathcal{F}$ (modulo $\equiv_{\mathcal{I}}$).
Consider the following function from $\mathcal{A}_{\mathcal{I}}$ to $\mathcal{C}$:

$$\gamma_{\mathcal{I}}(\Theta) = \lambda p(\mathbf{x}).\{p(\mathbf{t}) \in \mathit{Atoms} \mid p(\mathbf{t}) \text{ satisfies } \Theta_p\},$$

If $(\mathcal{F}/_{\equiv_{\mathcal{I}}}, \preceq_{\mathcal{I}})$ is a complete lattice, it can easily be checked that $\mathcal{A}_{\mathcal{I}}$ is a complete lattice and the function $\gamma_{\mathcal{I}}$ is meet-additive. Hence $\gamma_{\mathcal{I}}$ determines a Galois connection with $\mathcal{C}$. We can define the best abstraction of $T_P^{\mathcal{C}}$ on $\mathcal{A}_{\mathcal{I}}$ as

$$T_P^{\mathcal{I}}(\Theta) = \lambda p(\mathbf{x}). \bigvee_{p(\mathbf{t}) \leftarrow p_1(\mathbf{t}_1),\ldots,p_n(\mathbf{t}_n) \in P} \bigwedge \{\Phi \mid \mathcal{I} \models \bigwedge_{i=1}^{n} \Theta_{p_i}[\mathbf{x}_i \backslash \mathbf{t}_i] \Rightarrow \Phi_p[\mathbf{x} \backslash \mathbf{t}]\}.$$

It can be checked that P verifies assertions $\{\Theta_p\}_{p \in \mathit{Pred}}$ iff $\alpha_{\mathcal{I}}(\mathit{lfp}(T_P^{\mathcal{C}})) \leq \Theta$, where $\alpha_{\mathcal{I}}$ is the adjoint function of $\gamma_{\mathcal{I}}$. A verification condition can be derived from $T_P^{\mathcal{I}}(\Theta) \leq \Theta$.

Theorem 5.1 *Let P be a logic program and $\{\Phi_p\}_{p \in \mathit{Pred}}$ be assertions of $\mathcal{F}$. A sufficient condition for P to be $\mathcal{I}$-success-correct with respect to $\{\Phi_p\}_{p \in \mathit{Pred}}$is that for each clause $p(\mathbf{t}) \leftarrow p_1(\mathbf{t}_1), \ldots, p_n(\mathbf{t}_n)$ of P it is true that*

$$\mathcal{I} \models \bigwedge_{i=1}^{n} \Phi_{p_i}[\mathbf{x}_i \backslash \mathbf{t}_i] \Rightarrow \Phi_p[\mathbf{x} \backslash \mathbf{t}] \tag{9}$$

This is essentially the verification method proposed by Clark[8] and Deransart[17].

Example 5.2 Let us consider the program of example (3.1) and the assertion $\Phi_{app} = gr(x_3^{app}) \Rightarrow gr(x_1^{app}) \wedge gr(x_2^{app})$. For the clause $app([], Y, Y)$, it can easily be checked that $\mathcal{I} \models gr(Y) \Rightarrow gr([]) \wedge gr(Y)$. For the clause $app([X : Xs], Ys, [X : Zs]) \leftarrow app(Xs, Ys, Zs)$ it can be showed that $\mathcal{I} \models (gr(Zs) \Rightarrow gr(Xs) \wedge gr(Ys)) \Rightarrow (gr([X : Zs]) \Rightarrow gr([X : Xs]) \wedge gr(Ys))$.
By the above theorems, we conclude that the program of example (3.1) verifies the assertion $gr(x_3^{app}) \Rightarrow gr(x_1^{app}) \wedge gr(x_2^{app})$. ▮

A similar treatment can be given when the satisfaction for atoms is given with respect to a notion of derivability in a theory, which axiomatizes the properties of interest. Notice that if the relation $\models$ is decidable, we have an effective test to check the conditions. As an example, we could consider the decidable language of properties by Marchiori [20, 21], which allows us to express groundness, freeness and sharing of terms.
As for the completeness of the method, since it is equivalent to the precision of abstraction, it strongly depends on the choice of the language.

6 Conclusion

As already advocated by the Cousots [16, 14], abstract interpretation can be very helpful to understand, organize and synthesize proof methods for program verification. We have shown this in practice, by defining various proof methods, each obtained in a uniform way by unfolding pre-fixpoint relations on domains obtained by abstracting a concrete semantics of derivations. We have derived new more general methods, provided a unified view of a class of proof methods and made clear their mutual relationship.
It is worth noting, that, by using abstract interpretation theory, the definition of a new

verification method simply requires the formalization of the property we are interested in as an abstraction of SLD-derivations. Once we have the abstraction, we systematically derive the specific sufficient correctness conditions.
Our results can be applied to abstract diagnosis [11]. The verification condition on the abstract operator is essentially the same as the notion of partial correctness used there. Our conditions might then be used in a similar way to find potential errors. The new relevant features, to be added to abstract diagnosis, are intensional specifications and specifications given as pre- and post-conditions.
There are some interesting open problems related to intensional specifications and their relation to traditional abstract domains used in program analysis, developed to model specific properties (such as modes, types, groundness, etc.). Indeed we think that the logical domains derived for verification might be useful in this area.

References

[1] K. Apt. *From Logic Programming to Prolog.* Prentice Hall, 1997.

[2] K. R. Apt. Introduction to Logic Programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 495–574. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.

[3] K. R. Apt and E. Marchiori. Reasoning about Prolog Programs: from Modes through Types to Assertions. *Formal Aspects of Computing*, 3, 1994.

[4] G. Birkhoff. Lattice Theory. In *AMS Colloquium Publication, third ed.*, 1967.

[5] A. Bossi and N. Cocco. Verifying Correctness of Logic Programs. In J. Diaz and F. Orejas, editors, *Proc. TAPSOFT'89*, pages 96–110, 1989.

[6] J. Boye and J. Maluszynski. Directional Types and the Annotation Method. *Journal of Logic Programming*, 33(3):179–220, 1997.

[7] F. Bueno, P. Deransart, W. Drabent, G. Ferrand, M. Hermenegildo, J. Maluszynski, and G. Puebla. On the Role of Semantic Approximations in Validation and Diagnosis of Constraint Logic Programs. In *Proc. of the 3rd. Int'l Workshop on Automated Debugging–AADEBUG'97*, pages 155–170, Linkoping, Sweden, May 1997. U. of Linkoping Press.

[8] K. L. Clark. Predicate logic as a computational formalism. Res. Report DOC 79/59, Imperial College, Dept. of Computing, London, 1979.

[9] M. Comini. *An abstract interpretation framework for Semantics and Diagnosis of logic programs.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 1998.

[10] M. Comini, G. Levi, and M. C. Meo. Compositionality of *SLD*-derivations and their abstractions. In J. Lloyd, editor, *Proceedings of the 1995 Int'l Symposium on Logic Programming*, pages 561–575. The MIT Press, 1995.

[11] M. Comini, G. Levi, M. C. Meo, and G. Vitiello. Abstract Diagnosis. Submitted for publication, 1996.

[12] M. Comini, G. Levi, M. C. Meo, and G. Vitiello. Proving properties of logic programs by abstract diagnosis. In M. Dams, editor, *Analysis and Verification of Multiple-Agent Languages, 5th LOMAPS Workshop*, number 1192 in Lecture Notes in Computer Science, pages 22–50. Springer-Verlag, 1996.

[13] P. Cousot. Methods and Logics for Proving Programs. In J. V. Leeuwen, editor, *Formal Methods and Semantics*, volume B of Handbook of Theoretical Computer Science, pages 843–993. Elsevier Science Publishers B.V. (North-Holland), 1990.

[14] P. Cousot. Constructive Design of a Hierarchy of Semantics of a Transition system by Abstract Interpretation. *Electronic Notes in Theoretical Computer Science*, 6, 1997. URL:http://www.elsevier.nl/locate/entcs/volume6.html.

[15] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. Fourth ACM Symp. Principles of Programming Languages*, pages 238–252, 1977.

[16] P. Cousot and R. Cousot. Inductive Definitions, Semantics and Abstract Interpretation. In *Proc. Nineteenth Annual ACM Symp. on Principles of Programming Languages*, pages 83–94. ACM Press, 1992.

[17] P. Deransart. Proof Methods of Declarative Properties of Definite Programs. *Theoretical Computer Science*, 118(2):99–166, 1993.

[18] W. Drabent and J. Maluszynski. Inductive Assertion Method for Logic Programs. *Theoretical Computer Science*, 59(1):133–155, 1988.

[19] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative Modeling of the Operational Behavior of Logic Languages. *Theoretical Computer Science*, 69(3):289–318, 1989.

[20] E. Marchiori. A Logic for Variable Aliasing in Logic Programs. In G. Levi and M. Rodriguez-Artalejo, editors, *Proceedings of the 4th International Conference on Algebraic and Logic Programming (ALP'94)*, number 850 in LNCS, pages 287–304. Springer Verlag, 1994.

[21] E. Marchiori. Design of Abstract Domains using First-order Logic. In M. Hanus and M. Rodriguez-Artalejo, editors, *Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP'96)*, number 1139 in LNCS, pages 209–223. Springer Verlag, 1996.