Modular Abstract Diagnosis

Marco Comini and **Giorgio Levi**
Dipartimento di Informatica,
Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy.
comini, levi@di.unipi.it, http://www.di.unipi.it/~comini, levi

Giuliana Vitiello
Dipartimento di Informatica ed Applicazioni,
Università di Salerno, Baronissi (Salerno), Italy,
giuvit@dia.unisa.it

September 15, 1997

Abstract diagnosis [5, 6, 4] is a generalization of declarative diagnosis [10, 8, 7], where we consider operational properties, i.e., *observables*, which are properties which can be extracted from a goal computation. In a modular abstract diagnosis we are concerned with programs composed of separate modules. Our theory of abstract diagnosis can be applied to modular diagnosis even if our concrete semantics *is not* *OR*-compositional, while usually an *OR*-compositional semantics was needed (see, for example, modular analysis in [2]).

Abstract diagnosis consists of comparing the actual and the intended behaviors of the program P w.r.t. the observable property α and determining the wrong program components, when the two behaviors are different. The formulation is parametric w.r.t. the property α considered in the specification and in the actual behavior. Assume $\mathcal{F}[\![\mathsf{P}]\!]$ be the concrete semantics of a program P (in our case the least fixed point of the *s*-semantics immediate consequences operator $\mathcal{P}[\![\mathsf{P}]\!]$) and α be an observable, i.e., an abstract domain related to the concrete domain of answer substitutions by a Galois insertion, whose abstraction function is α. The standard correctness and completeness properties can now be defined as follows.

Definition 1 *[3, 4] Let* P *be a program and* α *be an observable.*

1. P *is* partially correct *w.r.t.* $\mathcal{I}_\alpha$ *if* $\alpha(\mathcal{F}[\![\mathsf{P}]\!]) \leq \mathcal{I}_\alpha$.

2. P *is* complete *w.r.t.* $\mathcal{I}_\alpha$ *if* $\mathcal{I}_\alpha \leq \alpha(\mathcal{F}[\![\mathsf{P}]\!])$.

3. P *is* totally correct *w.r.t.* $\mathcal{I}_\alpha$, *if* $\alpha(\mathcal{F}[\![\mathsf{P}]\!]) = \mathcal{I}_\alpha$.

Note that we are comparing the abstraction of the concrete semantics and the specification. Abstract diagnosis determines the errors without requiring to determine the symptoms in advance. This is obtained by systematically deriving all the incorrect clauses and uncovered elements. Incorrect clauses and uncovered elements are defined in terms of the abstract immediate consequences operator,

$$\begin{aligned}\mathcal{P}_\alpha[\![P]\!](\mathcal{K}_\alpha) = \lambda p(\mathbf{x}).\tilde{\bigcup}\{D \mid\ & c = p(\mathbf{x}) \leftarrow E, A_1, \ldots, A_n \text{ is a renamed} \\ & \text{clause of } P,\ \boldsymbol{z} = \mathit{var}(c), \text{ for } i \in [1, n],\ \mathbf{y}_i = \mathit{var}(A_i), \\ & D = (\alpha(\{E\}) \,\tilde{\otimes}^{\boldsymbol{z}}_{\mathbf{y}_1} \mathcal{K}_\alpha(A_1) \,\tilde{\otimes}^{\boldsymbol{z}}_{\mathbf{y}_2} \cdots \tilde{\otimes}^{\boldsymbol{z}}_{\mathbf{y}_n} \mathcal{K}_\alpha(A_n)) \,\tilde{\restriction}_{\mathbf{x}}\},\end{aligned}$$

where $\mathcal{K}_\alpha$ is an abstract interpretation and $\tilde{\bigcup}$, $\tilde{\otimes}$ and $\tilde{\restriction}$ are the abstract join, meet and restriction operators (automatically derived for each observable α from the concrete operators) [3, 4].

Definition 2 *Let* P *be a program. If there exists an* $\mathbb{A}$*-element* σ *such that* $\sigma \not\leq \mathcal{I}_\alpha$ *and* $\sigma \leq \mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}_\alpha)$*, then the clause* $c \in P$ *is* incorrect *on* σ*.*

Definition 3 *Let* P *be a program. An* $\mathbb{A}$*-element* σ *is* uncovered *if* $\sigma \leq \mathcal{I}_\alpha$ *and* $\sigma \not\leq \mathcal{P}_\alpha[\![P]\!](\mathcal{I}_\alpha)$*.*

The abstract immediate consequences operator can in general introduce an approximation. *Complete observables* are those observables such that $\alpha \circ \mathcal{P}[\![P]\!] = \mathcal{P}_\alpha[\![P]\!] \circ \alpha$, while for *approximate observables*[1] $\alpha \circ \mathcal{P}[\![P]\!] \leq \mathcal{P}_\alpha[\![P]\!] \circ \alpha$ holds. Complete observables (such as computed answers, correct answers, etc.) usually lead to infinite specifications (and therefore to non-effective definitions of incorrect clauses and uncovered elements). On the contrary, specifications are finite (and diagnosis is effective) in the case of approximate observables (such as *depth*(k) answers, types, groundness dependencies, etc.). However weaker result hold, because of approximation. Here are the basic results of abstract diagnosis.

Let α be an observable. Then

1. If there are no incorrect clauses in P, then P is partially correct w.r.t. α.

2. Let P be partially correct w.r.t. $\mathcal{I}_\alpha$. If there exists an uncovered $\mathbb{A}$-element, then P is not complete.

3. Let α be complete and P be a complete program w.r.t. $\mathcal{I}_\alpha$. If there exists an incorrect clause in P, then P is not partially correct.

4. Let α be a complete observable and assume $\mathcal{P}_\alpha[\![P]\!]$ has a unique fixpoint. If there are no uncovered $\mathbb{A}$-elements, then P is complete w.r.t. $\mathcal{I}_\alpha$.

As already said, in a modular abstract diagnosis we are concerned with programs composed of separate modules. The idea, of course, is that of performing the diagnosis in a modular way, i.e., module by module. Modular analysis is usually based on an *OR*-compositional semantics. For example, the compositional analysis framework in [2] is based on the *OR*-compositional version of the s-semantics [1]. Our concrete semantics *is not* *OR*-compositional and this is obviously true for all its abstractions. However, we can note that abstract diagnosis does not require to actually compute the abstract semantics, since it is simply based on one application of the abstract immediate consequences

[1] Approximate observables have noetherian domains.

operator to the specification. And the s-semantics immediate consequences operator is known to be *OR*-compositional (see for example [9]). The same result holds (by definition) for all the abstract immediate consequences operator corresponding to complete and approximate observables. The conclusion is that our theory of abstract diagnosis can directly be applied to modular diagnosis, as we will formally show in the following.

We assume a program P to be partitioned into *predicate-disjoint* modules [2], such that each predicate symbol is completely defined by a single module. Namely,

Definition 4 *A program partitioning* $P_1, \dots, P_n$ *is* predicate-disjoint *if* $\forall i \neq j.\ preds(P_i) \cap preds(P_j) = \emptyset$, *where* $preds(P) = \{p \mid p(\mathbf{t}) \leftarrow B \in P\}$.

Specifications $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$ are associated to modules $P_1, \dots, P_n$. Since the partition is predicate-disjoint, any $\mathcal{I}^i_\alpha$ is undefined for all the pure atoms with predicate *not* in $preds(P_i)$ and then $\forall i \neq j.\ dom(\mathcal{I}^i_\alpha) \cap dom(\mathcal{I}^j_\alpha) = \emptyset$. It is worth noting that our definition of partitioning does not require a *hierarchical decomposition*, since mutual recursion between modules is possible.

The overall specification is $\mathcal{I}_\alpha = \mathcal{I}^1_\alpha \tilde{\sqcup} \dots \tilde{\sqcup} \mathcal{I}^n_\alpha$, but note that any module P_i does not necessarily need to use all the other modules. Then we define an operator to take into account only the *relevant* specifications, i.e., $use(P_i) = \{\mathcal{I}^j_\alpha \mid$ module P_i uses (i.e., clause bodies contain) predicates which are defined by another module $P_j\}$[2]. If P_i is a *basic* module (i.e., does not use other modules) then $use(P_i) = \emptyset$.

The decomposition into modules allows us to define incorrect clauses and uncovered elements in a (more efficient) modular way.

Definition 5 *Let* $P_1, \dots, P_n$ *be a program partitioning and* c *be a clause in* P_i, *for some* $1 \leq i \leq n$. *If there exists an* $\mathbb{A}$*-element* σ *such that* $\sigma \not\leq \mathcal{I}^i_\alpha$ *and* $\sigma \leq \mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \tilde{\bigsqcup} use(P_i))$, *then the clause* $c \in P$ *is* m-incorrect *on* σ.

c is m-incorrect on σ if it derives, from the (relevant part of the entire) intended semantics, an element which is not in the module's intended semantics. Note that $\mathcal{P}_\alpha[\![\{c\}]\!]$ is the operator associated to the program $\{c\}$, consisting of the clause c only and that $\mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \tilde{\bigsqcup} use(P_i)) = \mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha)$.

Definition 6 *Let* $P_1, \dots, P_n$ *be a program partitioning and* $\sigma = p(\mathbf{x}) \mapsto D$ *be an* $\mathbb{A}$*-element such that* $p \in preds(P_i)$, *for some* $1 \leq i \leq n$. σ *is* m-uncovered *if* $\sigma \leq \mathcal{I}^i_\alpha$ *and* $\sigma \not\leq \mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \tilde{\bigsqcup} use(P_i))$.

An element in the intended semantics of a module is m-uncovered if there are no clauses in the module deriving it from the (relevant part of the entire) intended semantics.

It is worth noting that in the above definitions of m-incorrect clauses and m-uncovered elements, we compare two abstract interpretations which give a meaning only to those predicates which are defined inside the module and that we are only concerned with the specifications used by the module.

The following theorems show that the general results of abstract diagnosis do apply to modular abstract diagnosis.

[2]Note that $\tilde{\bigsqcup} use(P_i) \leq \mathcal{I}_\alpha$ and that $dom(\tilde{\bigsqcup} use(P_i)) \cap dom(\mathcal{I}^i_\alpha) = \emptyset$.

Theorem 7 *Let* $\mathsf{P} = \mathsf{P}_1 \cup \cdots \cup \mathsf{P}_n$*. If there are no m-incorrect clauses in any module* P_i*, then* P *is partially correct w.r.t.* α*.*

Proof. By hypothesis, for any i, $\forall c \in \mathsf{P}_i$. $\mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \bigsqcup use(\mathsf{P}_i)) \leq \mathcal{I}^i_\alpha$. Hence $\forall i$. $\mathcal{P}_\alpha[\![\mathsf{P}_i]\!](\mathcal{I}_\alpha) \leq \mathcal{I}^i_\alpha$. Now, by definition of $\mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{I}_\alpha)$, $\mathcal{P}_\alpha[\![\cup \mathsf{P}_i]\!](\mathcal{I}_\alpha) \leq \tilde{\bigsqcup} \mathcal{I}^i_\alpha = \mathcal{I}_\alpha$. Hence $\mathcal{I}_\alpha$ is a pre-fixpoint of $\mathcal{P}_\alpha[\![\mathsf{P}]\!]$ and then, since $\alpha(\mathcal{F}[\![\mathsf{P}]\!]) \leq \mathcal{F}_\alpha[\![\mathsf{P}]\!] = \mathit{lfp}\, \mathcal{P}_\alpha[\![\mathsf{P}]\!]$, by Tarski's theorem $\alpha(\mathcal{F}[\![\mathsf{P}]\!]) \leq \mathcal{I}_\alpha$. ■

Theorem 8 *Let* α *be a complete observable and* $\mathsf{P} = \mathsf{P}_1 \cup \cdots \cup \mathsf{P}_n$ *be a complete program w.r.t.* $\mathcal{I}_\alpha$*. If for some* i *there exists an m-incorrect clause in* P_i*, then* P *is not partially correct.*

Proof. By completeness of P and Theorem 3.5 in [3], $\mathcal{I}_\alpha \leq \alpha(\mathcal{F}[\![\mathsf{P}]\!]) = \mathcal{F}_\alpha[\![\mathsf{P}]\!]$. Then, by monotonicity of $\mathcal{P}_\alpha[\![\mathsf{P}]\!]$, $\mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{I}_\alpha) \leq \mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{F}_\alpha[\![\mathsf{P}]\!]) = \mathcal{F}_\alpha[\![\mathsf{P}]\!]$. Thus, $\sigma \not\leq \mathcal{I}^i_\alpha$ and $\sigma \leq \mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}_\alpha) = \mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \tilde{\bigsqcup} use(\mathsf{P}_i))$ implies $\sigma \not\leq \mathcal{I}^i_\alpha$ and $\sigma \leq \mathcal{F}_\alpha[\![\mathsf{P}]\!] = \alpha(\mathcal{F}[\![\mathsf{P}]\!])$, which means that P is not partially correct. ■

Theorem 9 *Let* $\mathsf{P} = \mathsf{P}_1 \cup \cdots \cup \mathsf{P}_n$ *be partially correct w.r.t.* $\mathcal{I}_\alpha$*. If there exists an m-uncovered* $\mathbb{A}$*-element, then* P *is not complete.*

Proof. Let $\sigma = p(\mathbf{x}) \mapsto D$ with $p \in preds(\mathsf{P}_i)$, for some $1 \leq i \leq n$. By Theorem 3.5 in [3] and partial correctness of P, $\mathcal{F}_\alpha[\![\mathsf{P}]\!] = \alpha(\mathcal{F}[\![\mathsf{P}]\!]) \leq \mathcal{I}_\alpha$. By monotonicity of $\mathcal{P}_\alpha[\![\mathsf{P}]\!]$,

$$\alpha(\mathcal{F}[\![\mathsf{P}]\!]) = \mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{F}_\alpha[\![\mathsf{P}]\!]) \leq \mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{I}_\alpha). \tag{1}$$

Now, $\sigma \leq \mathcal{I}^i_\alpha$ and $\sigma \not\leq \mathcal{P}_\alpha[\![\mathsf{P}_i]\!](\mathcal{I}_\alpha) = \mathcal{P}_\alpha[\![\mathsf{P}_i]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \tilde{\bigsqcup} use(\mathsf{P}_i))$ implies (since other modules cannot define predicate p) $\sigma \leq \mathcal{I}^i_\alpha$ and $\sigma \not\leq \mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{I}_\alpha)$. Thus $\sigma \leq \mathcal{I}^i_\alpha \leq \mathcal{I}_\alpha$ and, by (1), $\sigma \not\leq \alpha(\mathcal{F}[\![\mathsf{P}]\!])$, which means that P is not complete. ■

Theorem 10 *Let* α *be a complete observable,* $\mathsf{P} = \mathsf{P}_1 \cup \cdots \cup \mathsf{P}_n$ *be a program and assume* $\mathcal{P}_\alpha[\![\mathsf{P}]\!]$ *has a unique fixpoint. If there are no m-uncovered* $\mathbb{A}$*-elements, then* P *is complete w.r.t.* $\mathcal{I}_\alpha$*.*

Proof. Absence of m-uncovered $\mathbb{A}$-elements implies $\forall i. \mathcal{I}^i_\alpha \leq \mathcal{P}_\alpha[\![\mathsf{P}_i]\!](\mathcal{I}_\alpha) = \mathcal{P}_\alpha[\![\mathsf{P}_i]\!](\mathcal{I}^i_\alpha \tilde{\sqcup} \tilde{\bigsqcup} use(\mathsf{P}_i))$. Hence, $\mathcal{I}_\alpha \leq \mathcal{P}_\alpha[\![\mathsf{P}]\!](\mathcal{I}_\alpha)$, i.e., $\mathcal{I}_\alpha$ is a post-fixpoint of $\mathcal{P}_\alpha[\![\mathsf{P}]\!]$ and, by Tarski's theorem, $\mathcal{I}_\alpha \leq \mathit{gfp}(\mathcal{P}_\alpha[\![\mathsf{P}]\!])$. Since, by Theorem 3.5 in [3], $\alpha(\mathcal{F}[\![\mathsf{P}]\!]) = \mathcal{F}_\alpha[\![\mathsf{P}]\!] = \mathit{lfp}\, \mathcal{P}_\alpha[\![\mathsf{P}]\!]$ and, by hypothesis, $\mathit{gfp}(\mathcal{P}_\alpha[\![\mathsf{P}]\!]) = \mathit{lfp}(\mathcal{P}_\alpha[\![\mathsf{P}]\!])$, the program P is complete. ■

Example 11 ______________________________

The program in Figure 1 is a wrong version of a program verifying satisfiability of boolean formulas, built with *or* and *not*, which has a missing clause. We consider the observable *depth*(2) answers (τ_2) where we replace all the subterms at depth greater than 2 by a fresh variable. τ_2 is an approximate observable (see [3] for the precise definition of the corresponding abstract immediate consequences operator). The specification of module "sat" is

$$\begin{aligned}\mathcal{I}_{\tau_2}(\mathsf{sat}(\mathsf{X})) = \big\{ & \{\mathsf{X} = \mathsf{true}\}, \{\mathsf{X} = \mathsf{or}(\mathsf{Y}, \mathsf{true})\}, \{\mathsf{X} = \mathsf{or}(\mathsf{true}, \mathsf{Y})\}, \\ & \{\mathsf{X} = \mathsf{or}(\mathsf{not}(\hat{x}), \mathsf{Y})\}, \{\mathsf{X} = \mathsf{or}(\mathsf{Y}, \mathsf{not}(\hat{x}))\}, \{\mathsf{X} = \mathsf{not}(\mathsf{false})\}, \\ & \{\mathsf{X} = \mathsf{not}(\mathsf{not}(\hat{x}))\}, \{\mathsf{X} = \mathsf{not}(\mathsf{or}(\hat{x}, \hat{y}))\}, \\ & \{\mathsf{X} = \mathsf{or}(\mathsf{Y}, \mathsf{or}(\hat{x}, \hat{y}))\}, \{\mathsf{X} = \mathsf{or}(\mathsf{or}(\hat{x}, \hat{y}), \mathsf{Y})\} \big\}\end{aligned}$$

Module "sat"

```
sat(not(X)) :- inv(X).
sat(or(X,Y)) :- sat(X).
sat(true).

%the clause sat(or(X,Y)) :- sat(Y). is missing
```

Module "inv"

```
inv(false).
inv(not(X)) :- sat(X).
inv(or(X,Y)) :- inv(X),inv(Y).
```

Figure 1: The program of Example 11

while the specification of module "inv" is

$$\begin{aligned}\mathcal{K}_{\tau_2}(\mathsf{inv}(X)) = \big\{&\{X = \mathsf{false}\}, \{X = \mathsf{or}(\mathsf{false}, \mathsf{false})\}, \{X = \mathsf{not}(\mathsf{true})\},\\ &\{X = \mathsf{not}(\mathsf{not}(\hat{x}))\}, \{X = \mathsf{not}(\mathsf{or}(\hat{x}, \hat{y}))\},\\ &\{X = \mathsf{or}(\mathsf{false}, \mathsf{not}(\hat{x}))\}, \{X = \mathsf{or}(\mathsf{not}(\hat{x}), \mathsf{false})\},\\ &\{X = \mathsf{or}(\mathsf{false}, \mathsf{or}(\hat{x}, \hat{y}))\}, \{X = \mathsf{or}(\mathsf{or}(\hat{x}, \hat{y}), \mathsf{false})\}\big\}\end{aligned}$$

By applying the definitions, we find out that there are no m-incorrect clauses and then (by Theorem 7) the program is partially correct. Furthermore, since the $\mathbb{A}$-element $\sigma = \mathsf{sat}(X) \mapsto \{X = \mathsf{or}(x, \mathsf{true})\}$ is m-uncovered then (by Theorem 9) the entire program is not complete.

Up to now we have shown how to perform a modular diagnosis (Definitions 5 and 6), to prove partial correctness and completeness of the entire program (Definition 1). We consider now the more realistic case where we have a single module P_i and we want to prove its partial correctness and completeness, by assuming that all the other (possibly still not implemented) modules satisfy their specifications (i.e., are totally correct). We will still perform the diagnosis by using Definitions 5 and 6. However we have to introduce a new definition of partial correctness and completeness. This in turns requires the introduction of a new notion of concrete semantics of a module, where the unknown modules are replaced by their specifications. Since for unknown modules we only have abstract specifications, their concrete semantics will be the concretization of the abstract specifications. This is achieved by first defining the following concrete ($\mathcal{I}_\alpha$-augmented) immediate consequences operator:

$$\mathcal{P}^{\mathcal{I}_\alpha}[\![P_i]\!](\mathcal{K}) = \mathcal{P}[\![P_i]\!](\mathcal{K} \sqcup \bigsqcup \gamma(use(P_i))).$$

This operator is (obviously) continuous and then the concrete semantics of a program module P_i can be defined as

$$\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!] = \mathcal{P}^{\mathcal{I}_\alpha}[\![P_i]\!]{\uparrow}\omega\,.$$

This leads to the new notions of partial correctness and completeness of a module w.r.t. the intended abstract semantics of the whole program.

Definition 12 *Let $P_1, \dots, P_n$ be a program partitioning, α be an observable, and $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$ be an intended modules semantics. A module P_i is*

1. m-partially correct *w.r.t.* $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$ *if* $\alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!]) \leq \mathcal{I}^i_\alpha$.
2. m-complete *w.r.t.* $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$ *if* $\mathcal{I}^i_\alpha \leq \alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!])$
3. P *is* m-totally correct *w.r.t.* $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$, *if* $\alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!]) = \mathcal{I}^i_\alpha$

Theorem 13 *Let $P = P_1 \cup \dots \cup P_n$. If any module P_i, is m-partially correct (m-complete) w.r.t. $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$, then P is partially correct (complete) w.r.t. $\mathcal{I}_\alpha$.*

Theorem 14 *If there are no m-incorrect clauses in P_i, then P_i is m-partially correct w.r.t. $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$.*

Proof. By hypothesis, for any clause c in P_i, $\mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}^i_\alpha \,\tilde{\sqcup} \widetilde{\bigsqcup}\, use(P_i)) \leq \mathcal{I}^i_\alpha$. Hence $\mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha) \leq \mathcal{I}^i_\alpha$. Now, by definition, $\mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha) = \mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha)$ and then $\mathcal{I}^i_\alpha$ is a pre-fixpoint of $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$. Then, since $\alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!]) \leq \mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!] = \mathit{lfp}(\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]) \leq \mathcal{I}^i_\alpha$, P_i is m-partially correct w.r.t. $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$. ■

Theorem 15 *Let α be a complete observable and P_i be an m-complete module w.r.t. $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$, for some $1 \leq i \leq n$. If there exists an m-incorrect clause in P_i, then P_i is not m-partially correct.*

Proof. By m-completeness of P_i, $\mathcal{I}^i_\alpha \leq \alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!]) = \mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$. By monotonicity of $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$, $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha) \leq \mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]) = \mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$. On the other hand, by definition, $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha) = \mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha)$. Thus, $\sigma \not\leq \mathcal{I}^i_\alpha$ and $\sigma \leq \mathcal{P}_\alpha[\![\{c\}]\!](\mathcal{I}_\alpha)$ implies $\sigma \not\leq \mathcal{I}^i_\alpha$ and $\sigma \leq \mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$ i.e., P_i is not m-partially correct. ■

We can state two theorems about completeness of a module P_i by first introducing the following abstract immediate consequences operator.

$$\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{K}_\alpha) = \mathcal{P}_\alpha[\![P_i]\!](\mathcal{K}_\alpha \,\tilde{\sqcup}\, \widetilde{\bigsqcup}\, use(P_i)),$$

Theorem 16 *Let P_i be m-partially correct w.r.t. $\mathcal{I}^1_\alpha, \dots, \mathcal{I}^n_\alpha$, for some $1 \leq i \leq n$. If there exists an m-uncovered $\mathbb{A}$-element in P_i, then P_i is not m-complete.*

Proof. Since α is approximate,

$$\begin{aligned}
&(\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!] \circ \alpha)(\mathcal{J}) = && [\text{by definition}]\\
&\mathcal{P}_\alpha[\![P_i]\!](\alpha(\mathcal{J}) \,\tilde{\sqcup} (\widetilde{\bigsqcup}\, use(P_i)) \geq && [\text{since } \mathcal{P}_\alpha[\![\cdot]\!] \text{ is correct}]\\
&\alpha \circ \mathcal{P}[\![P_i]\!](\gamma\alpha(\mathcal{J}) \sqcup (\bigsqcup \gamma(use(P_i))) \geq && [\text{since } \gamma\alpha \sqsupset id]\\
&\alpha \circ \mathcal{P}[\![P_i]\!](\mathcal{J} \sqcup (\bigsqcup \gamma(use(P_i)))) \geq && [\text{by definition}]\\
&(\alpha \circ \mathcal{P}^{\mathcal{I}_\alpha}[\![P_i]\!])(\mathcal{J}).
\end{aligned}$$

Hence,

$$\alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!]) = \qquad [\text{since } \mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!] \text{ is a fixpoint}]$$

```
accept([a|Xs]) :- acc(Xs).

%the clause accept([]). is missing
```

Figure 2: The program of Example 18

$$\begin{array}{ll}
\alpha(\mathcal{P}^{\mathcal{I}_\alpha}[\![P_i]\!](\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!])) \leq & [\text{by previous result}] \\
\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!])) \leq & [\text{by monot. of } \mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!] \text{ and m-par. corr. of } P_i] \\
\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha) = & [\text{by definition}] \\
\mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha). &
\end{array}$$

Now, if σ is an m-uncovered $\mathbb{A}$-element (i.e., $\sigma \leq \mathcal{I}^i_\alpha$ and $\sigma \not\leq \mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha)$), then $\sigma \leq \mathcal{I}^i_\alpha$ and $\sigma \not\leq \alpha(\mathcal{F}^{\mathcal{I}_\alpha}[\![P_i]\!])$, i.e., P_i is not m-complete. ■

Theorem 17 *Let α be a complete observable and, for some $1 \leq i \leq n$, assume that $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$ has a unique fixpoint. If there are no m-uncovered $\mathbb{A}$-elements in P_i, then P_i is m-complete w.r.t. $\mathcal{I}^1_\alpha, \ldots, \mathcal{I}^n_\alpha$.*

Proof. Absence of m-uncovered $\mathbb{A}$-elements in P_i implies $\mathcal{I}^i_\alpha \leq \mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha)$. Since, by definition, $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha) = \mathcal{P}_\alpha[\![P_i]\!](\mathcal{I}_\alpha)$, $\mathcal{I}^i_\alpha \leq \mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!](\mathcal{I}^i_\alpha)$, i.e., $\mathcal{I}^i_\alpha$ is a post-fixpoint of $\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$ and, by Tarski's theorem, $\mathcal{I}^i_\alpha \leq gfp(\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!])$. Now, by hypothesis, $gfp(\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]) = lfp(\mathcal{P}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]) = \mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$. Thus, $\mathcal{I}^i_\alpha \leq \mathcal{F}^{\mathcal{I}_\alpha}_\alpha[\![P_i]\!]$, i.e., P_i is m-complete w.r.t. $\mathcal{I}^1_\alpha, \ldots, \mathcal{I}^n_\alpha$. ■

Example 18

The program P in Figure 2 is a wrong version of a module of an automaton which recognizes the language $L = \{(ab)^n \mid n \geq 0\} \cup \{(ab)^n a \mid n \geq 0\}$. Its specification w.r.t. the τ_2 observable (*depth*(2) answers) is

$$\mathcal{I}_{\tau_2}(\mathrm{accept}(X)) = \{\{X = []\}, \{X = [a]\}, \{X = [a, \hat{b}|\hat{x}]\}\}$$

while the specification of the other module is

$$\mathcal{K}_{\tau_2}(\mathrm{acc}(X)) = \{\{X = []\}, \{X = [b]\}, \{X = [b, \hat{a}|\hat{x}]\}\}.$$

Let $\mathcal{I}_\alpha := \mathcal{I}_{\tau_2} \tilde{\sqcup} \mathcal{K}_{\tau_2}$. The concrete semantics is

$$\begin{aligned}\mathcal{F}^{\mathcal{I}_\alpha}[\![P]\!](\mathrm{accept}(X)) = \{&\{X = [a]\}, \{X = [a, b]\}, \{X = [a, b, Y|Xs]\}, \\ &\{X = [a, b, Y, Z|Xs]\}, \{X = [a, b, Y, Z, W|Xs]\}, \ldots\}\end{aligned}$$

and its abstraction is

$$\tau_2(\mathcal{F}^{\mathcal{I}_\alpha}[\![P]\!])(\mathrm{accept}(X)) = \{\{X = [a]\}, \{X = [a, \hat{b}|\hat{x}]\}\}$$

By applying the module diagnosis we find out that P is m-partially correct (without actually computing $\mathcal{F}^{\mathcal{I}_\alpha}[\![P]\!]$) and that the $\mathbb{A}$-element $\mathrm{acc}(X) \mapsto \{X = []\}$ is m-uncovered and then (by Theorem 16) the module is not m-complete (and the entire program is not complete).

References

[1] A. Bossi, M. Gabbrielli, G. Levi, and M. C. Meo. A Compositional Semantics for Logic Programs. *Theoretical Computer Science*, 122(1–2):3–47, 1994.

[2] M. Codish, S. K. Debray, and R. Giacobazzi. Compositional Analysis of Modular Logic Programs. In *Proc. Twentieth Annual ACM Symp. on Principles of Programming Languages*, pages 451–464. ACM Press, 1993.

[3] M. Comini, G. Levi, M. C. Meo, and G. Vitiello. Abstract diagnosis. Submitted for publication. http://www.di.unipi.it/˜comini/Papers/, 1996.

[4] 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.

[5] M. Comini, G. Levi, and G. Vitiello. Abstract debugging of logic programs. In L. Fribourg and F. Turini, editors, *Proc. Logic Program Synthesis and Transformation and Metaprogramming in Logic 1994*, volume 883 of *Lecture Notes in Computer Science*, pages 440–450. Springer-Verlag, 1994.

[6] M. Comini, G. Levi, and G. Vitiello. Efficient detection of incompleteness errors in the abstract debugging of logic programs. In M. Ducassé, editor, *Proc. 2nd International Workshop on Automated and Algorithmic Debugging, AADEBUG'95*, 1995.

[7] G. Ferrand. Error Diagnosis in Logic Programming, an Adaptation of E. Y. Shapiro's Method. *Journal of Logic Programming*, 4:177–198, 1987.

[8] J. W. Lloyd. Declarative error diagnosis. *New Generation Computing*, 5(2):133–154, 1987.

[9] P. Mancarella and D. Pedreschi. An Algebra of Logic Programs. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 1006–1023. The MIT Press, 1988.

[10] E. Y. Shapiro. Algorithmic program debugging. In *Proc. Ninth Annual ACM Symp. on Principles of Programming Languages*, pages 412–531. ACM Press, 1982.