# ABSTRACT DIAGNOSIS

Comini, Levi, Meo & Vitiello, Abstract Diagnosis, JLP ~~(to appear)~~ 39, 1-3 (1999)

# THE DIAGNOSIS APPROACH TO PROGRAM VERIFICATION

- declarative diagnosis (debugging) [of logic programs]
  [Shapiro 83, Ferrand 87, Lloyd 87]

$P$ logic program
$I$ specification (intended declarative semantics of $P$)
$[\![P]\!]$ (actual) declarative semantics of $P$

- a method to prove the correctness and completeness of $P$ w.r.t. $I$
  - comparison between $I$ and $[\![P]\!]$
  - if they are different, locate the program bugs

- abstract diagnosis
  [Comini, Levi & Vitiello, META 94 – AADEBUG 95 – ILPS 95]

$P$ logic program
$\alpha$ observable property
- any abstraction of SLD-trees

$I_\alpha$ abstract specification (intended behavior w.r.t. $\alpha$)
$[\![P]\!]_\alpha$ (actual) behavior of $P$ w.r.t. $\alpha$
- more concrete than the declarative semantics
  (e.g. when $\alpha$ is computed answers)
- less concrete than the declarative semantics

# DECLARATIVE DIAGNOSIS
# VS.
# ABCTRACT DIAGNOSIS



- declarative diagnosis
  - declarative semantics as reference semantics
  - two steps method
    - symptoms detection (using testing techniques)
    - from symptoms to errors

- abstract diagnosis
  - the reference semantics can be any abstract semantics (including the declarative one)
  - no need to detect symptoms in advance
    - for some abstract semantics it may be non-effective

# DIAGNOSIS AND THE S-SEMANTICS



- the specification $\mathcal{I}$ is the intended s-semantics of P
- the actual semantics $[\![P]\!]$ is the semantics $\mathcal{O}[\![P]\!] = F[\![P]\!] = T_P \uparrow \omega$
- the two definition styles (bottom-up and top-down) of the s-semantics and its properties are relevant to diagnosis
  - the theory of diagnosis is based on the bottom-up characterization
  - the equivalent top-down characterization allows us to implement the diagnosers by means of simple meta-interpreters
  - the condensing property allows us to consider goal-independent specifications
    - if P is correct and complete w.r.t. the goal-independent specification $\mathcal{I}$ then it behaves correctly for all the goals
- the above properties hold for the abstractions of the s-semantics we will consider in abstract diagnosis
- our version of the s-semantics
  - CLP style, with equations on the Herbrand domain as constraints

# A COLLECTING S-SEMANTICS FOR CLP(H,=)

- program representation (for the sake of simplicity)

  pure atoms + constraints

  <u>example</u>

  $p(s(X), Y, [X|Z]) :- q(X, s(W)), p(W, s(Y), Z)$

  is represented as

  $p(X, Y, Z) :- X = s(X1), Z = [X1|Z1], W = s(W1), Y1 = s(Y),$
  $q(X, W), p(W1, Y1, Z1).$

- constraints = sets of equations

  $E$ — set of equations

  $soln(E)$ — set of solutions of $E$ (a solution is a grounding substitution $\vartheta$ such that all the equations in $E\vartheta$ are identities)

  $E_1, E_2$ — sets of equations

  $E_1 \preceq E_2$ iff $soln(E_1) \subseteq soln(E_2)$

  $\mathcal{E}$ — the set of all sets of equations (modulo equivalence)

# THE SEMANTIC DOMAIN



- the semantics maps each goal $G$ to an element of $\wp(\mathcal{E})$ (the set of answer constraints)

- partial order on $\wp(\mathcal{E})$

  $\mathcal{E}_1, \mathcal{E}_2$ elements of $\wp(\mathcal{E})$

  $\mathcal{E}_1 \sqsubseteq \mathcal{E}_2$ iff for each element $E \in \mathcal{E}_1$, there exists $E' \in \mathcal{E}_2$, such that $E \preceq E'$

- $(\wp(\mathcal{E}), \sqsubseteq)$ is a complete lattice

  - top element : $\mathcal{E}$
  - bottom element : $\phi$
  - least upper bound lub : set union
  - greatest lower bound glb

    $$glb(\{E_1, E_2, \dots\}, \{E_1', E_2', \dots\}) =$$
    $$\{E_1 \cup E_1', E_1 \cup E_2', \dots, E_2 \cup E_1', \dots\dots\}$$

    - $E_1 \cup E_2$ corresponds to unification and is transformation to solved form
      - satisfiability check
      - "normal form" for the equivalence class

# INTERPRETATIONS

- partial functions from <u>most general atomic goals</u> to elements of $\wp(\mathcal{E})$

- partial order on interpretations

$I_1, I_2$ interpretations $(\in \mathcal{I})$

$I_1 \leq I_2$ iff, for any most general atomic goal $A$, $I_1(A) \sqsubseteq I_2(A)$

- $(\mathcal{I}, \leq)$ is a complete lattice

# THE BOTTOM-UP CONSTRUCTION OF THE S-SEMANTICS

- the immediate consequences operator

  $\forall$ most general atom $p(X_1,\dots,X_n)$

  $T_P(\mathcal{I})(p(X_1,\dots,X_n)) =$

  $lub(\{\varepsilon \mid$

  $p(X_1,\dots,X_n) :- e_1,\dots,e_k \,\square\, q_1(X_1^1,\dots,X_{n_1}^1),\dots,q_m(X_1^m,\dots,X_{n_m}^m)$
  renamed clause of $P$

  $\mathcal{I}(q_1(X_1^1,\dots,X_{n_1}^1)) = \varepsilon_1,$
  $\vdots$
  $\mathcal{I}(q_m(X_1^m,\dots,X_{n_m}^m)) = \varepsilon_m,$

  $E = \{e_1,\dots,e_k\},$

  $\varepsilon = glb(\{\{E\},\varepsilon_1,\dots,\varepsilon_k\})|_{X_1,\dots,X_n}\})$

- the S-semantics is the interpretation

  $F[\![P]\!] = T_P \uparrow \omega$

# DIAGNOSIS W.R.T. COMPUTED ANSWERS

$P$ definite logic program

$F[\![P]\!] = T_P \uparrow \omega$ actual $\mathcal{S}$-semantics of $P$

$S$ intended semantics of $P$

- $P$ is <u>partially correct</u> w.r.t. $S$

$$F[\![P]\!] \leq S$$

- $P$ is <u>complete</u> w.r.t. $S$

$$S \leq F[\![P]\!]$$

  - termination issues are not addressed
    - the $\mathcal{C}$-semantics is too abstract

- <u>observation</u>

  $p(X_1, \ldots, X_n)$ most general atom

  $E$ set of equations (one answer constraint)

  an observation is a partial function $\vartheta$ which maps $p(X_1, \ldots, X_n)$ onto $\{E\}$

  - observations are interpretations

- an observation $\vartheta$ is an <u>incorrectness symptom</u>

  $\vartheta \leq F[\![P]\!]$ and $\vartheta \not\leq S$

  an answer constraint computed by $P$ not in the specification

- an observation $\vartheta$ is an <u>incompleteness symptom</u>

  $\vartheta \leq S$ and $\vartheta \not\leq F[\![P]\!]$

  an answer constraint in the specification which is not computed by $P$

# SYMPTOMS AND ERRORS



- some symptoms are not actual "bugs"

  they are just consequences of other "basic symptoms"

$P$ $\boxed{q(x) :- p(x).}$

$$S(p(x)) = \{\{x=a\}\}$$
$$S(q(x)) = \{\{x=a\}\}$$
$$F[\![P]\!](p(x)) = \phi$$
$$F[\![P]\!](q(x)) = \phi$$

$\mathcal{S}_1(p(x)) = \{\{x=a\}\}$ and $\mathcal{S}_2(q(x)) = \{\{x=a\}\}$ are both incompleteness symptoms, but $\mathcal{S}_2$ is just a consequence of $\mathcal{S}_1$

- some bugs cannot be detected by looking at the symptoms
  - an incorrectness may be hidden by an incompleteness and vice versa

$P$ $\boxed{\begin{array}{l} q(x) :- p(x). \\ p(x) :- x=b. \end{array}}$

$$S(q(x)) = \{\{x=b\}\}$$
$$F[\![P]\!](p(x)) = \{\{x=b\}\}$$
$$F[\![P]\!](q(x)) = \{\{x=b\}\}$$

- there exists only an incorrectness symptom

  $\mathcal{S}_1(p(x)) = \{\{x=b\}\}$
- if we fix this bug (by removing the second clause), we get an incompleteness symptom, since $F[\![P]\!](q(x)) = \phi$

- symptom detection requires a fixpoint computation

# INCORRECT CLAUSES AND UNCOVERED OBSERVATIONS

- the problems with symptoms can be solved by basing the diagnosis on the detection of
  - incorrect clauses
  - uncovered observations
    - detects basic and hidden bugs
    - requires one application of $T_P$ rather than a fixpoint computation
    - symptoms can be ignored

## incorrect clause

the clause $c \in P$ is incorrect on the observation $\sigma$ if

$$\sigma \not\leq S \quad \text{and} \quad \sigma \leq T_{\{c\}}(S)$$

(the clause c derives a wrong answer constraint from the specification)

## uncovered observation

the observation $\sigma$ is uncovered if

$$\sigma \leq S \quad \text{and} \quad \sigma \not\leq T_P(S)$$

(there are no clauses in P which can derive $\sigma$ from the specification)

- we will show that the diagnosis can be based on the detection of incorrect clauses and uncovered observations

# DIAGNOSIS OF CORRECTNESS: THEOREMS

## Theorem 1

if there are no incorrect clauses, then the program is partially correct

- if there are no incorrect clauses, $T_P(S) \leq S$
- $S$ is a pre-fixpoint of $T_P$
- $F[\![P]\!]$ is the least pre-fixpoint of $T_P$ $\rightarrow$ $F[\![P]\!] \leq S$

(if the program is not partially correct, then there exists an incorrect clause)

## Theorem 2

if the program is partially correct, then it is not always the case that there are no incorrect clauses

(an incorrectness may be hidden by an incompleteness)

## Theorem 3

if the program is complete, then
if there exists an incorrect clause on $\theta$, then
$\theta$ is an incorrectness symptom

- incorrect clauses are more meaningful than incorrectness symptoms
  - absence of incorrect clauses implies partial correctness
  - an incorrect clause always corresponds to a bug, while this is not the case for symptoms

# DIAGNOSIS OF COMPLETENESS

- cannot in general be based on the detection of uncovered observations

theorem

There exist a program P and a specification S, such that

- there are no uncovered observations
- P is not complete w.r.t. S

P $\boxed{p(x) :- p(x).}$

$S(p(x)) = \{true\}$

$F[\![P]\!](p(x)) = \phi$

$T_P(S)(p(x)) = \{true\}$

- S is a fixpoint of $T_P$ different from the least fixpoint $F[\![P]\!]$
  - clearly related to loops

- a theorem similar to theorem 1 holds, if we assume $T_P$ to have a unique fixpoint

# DIAGNOSIS OF COMPLETENESS: THEOREMS

## Theorem 4

if $T_P$ has a unique fixpoint and there are no uncovered observations, then the program is complete

( if $T_P$ has a unique fixpoint and P is not complete, then there exists an uncovered observation )

## Theorem 5

if the program is complete, then it is not always the case that there are no uncovered observations

( an incompleteness may be hidden by an incorrectness )

## Theorem 6

if the program is partially correct, then
if there exists an uncovered observation $\sigma$, then $\sigma$ is an incompleteness symptom

- uncovered observations are more meaningful than incompleteness symptoms
  - absence of uncovered observations implies completeness (if $T_P$ has one fixpoint only)
  - an uncovered observation always corresponds to a bug, while this is not the case for incompleteness symptoms!

# $T_P$ HAS ONE FIXPOINT ONLY, IF P IS AN ACCEPTABLE PROGRAM

acceptable programs were introduced to study the termination of pure PROLOG programs (Apt-Pedreschi, 93)

- they are exactly the left-terminating programs (all the LD-derivations for ground goals are finite)
- acceptability is undecidable, but all sensible programs turn out to be acceptable
  - all the pure PROLOG programs in the Sterling & Shapiro's book are acceptable
  - most wrong versions of sensible programs are acceptable (unless the bugs are just related to termination)
- the ground immediate consequence operator has one fixpoint only
- we have proved that the same result holds for the s-semantics $T_P$

# AN EXAMPLE

P

$$\boxed{\begin{array}{l} \text{ancestor}(X,Y) :- \text{parent}(Z,Y), \text{ancestor}(X,Z). \\ \text{ancestor}(X,Y) :- \text{parent}(Y,X). \\ \text{---- missing data base tuples ----} \end{array}}$$

$S(\text{parent}(X,Y)) = \{\{X=\text{terach}, Y=\text{abraham}\}, \{X=\text{abraham}, Y=\text{isaac}\}\}$

$S(\text{ancestor}(X,Y)) = \{\{X=\text{terach}, Y=\text{abraham}\}, \{X=\text{abraham}, Y=\text{isaac}\}, \{X=\text{terach}, Y=\text{isaac}\}\}$

$F[\![P]\!](\text{parent}(X,Y)) = \phi$

$F[\![P]\!](\text{ancestor}(X,Y)) = \phi$

$T_P(S)(\text{parent}(X,Y)) = \phi$

$T_P(S)(\text{ancestor}(X,Y)) = \{\{X=\text{abraham}, Y=\text{terach}\}, \{X=\text{isaac}, Y=\text{abraham}\}, \{X=\text{terach}, Y=\text{isaac}\}\}$

- the second clause is incorrect on the observations

  $\vartheta_1(\text{ancestor}(X,Y)) = \{\{X=\text{isaac}, Y=\text{abraham}\}\}$

  $\vartheta_2(\text{ancestor}(X,Y)) = \{\{X=\text{abraham}, Y=\text{terach}\}\}$

- the uncovered observations are

  $\vartheta_3(\text{parent}(X,Y)) = \{\{X=\text{terach}, Y=\text{abraham}\}\}$

  $\vartheta_4(\text{parent}(X,Y)) = \{\{X=\text{abraham}, Y=\text{isaac}\}\}$

  $\vartheta_5(\text{ancestor}(X,Y)) = \{\{X=\text{terach}, Y=\text{abraham}\}\}$

  $\vartheta_6(\text{ancestor}(X,Y)) = \{\{X=\text{abraham}, Y=\text{isaac}\}\}$

- there are no incorrectness symptoms, even if there is a wrong clause
- $\vartheta_7(\text{ancestor}(X,Y)) = \{\{X=\text{terach}, Y=\text{isaac}\}\}$ is not uncovered, even if it is an incompleteness symptom

# A MORE SYMMETRIC FORMULATION OF DIAGNOSIS

- proposed by (Ferrand 1993) for declarative diagnosis
- extends to abstract diagnosis

- the specification is a pair $(S^+, S^-)$
  - $S^+$ intended s-semantics ($lfp(T_P^s)$)
  - $S^-$ intended $gfp(T_P^s)$

- a new definition of completeness

  P is complete w.r.t. $(S^+, S^-)$ if

  $$S^- \leq gfp(T_P^s)$$

- the new definition of uncovered observation (error)

  the observation $\vartheta$ is uncovered if

  $$\vartheta \leq S^- \text{ and } \vartheta \not\leq T_P^s(S^-)$$

- the completeness theorem

  if there are no uncovered observations, then the program is complete

- no need for the unique fixpoint assumption

  reduces to the standard definition under the unique fixpoint assumption

- requires a more complex specification

# ORACLE-BASED TOP-DOWN DIAGNOSIS

## bottom-up diagnosis

- comparison between S and $T_P(S)$
- S must be specified in advance

## top-down diagnosis

- S can be implemented by an oracle (querying the user)

$$\mathcal{A}(p(X_1,\dots,X_n)) = \{E \mid E \text{ is an intended answer constraint of } ?\text{-}\ p(X_1,\dots,X_n)\}$$

- the diagnosis is expressed in terms of oracle simulation
  - one resolution step with program clauses
  - answers for the remaining goals from the oracle

# THE BOTTOM-UP CONSTRUCTION OF THE S-SEMANTICS

## ORACLE SIMULATION

- the immediate consequences operator

$\mathcal{T}_P$

resolution in P

answers from the oracle $\mathcal{A}$

$\mathcal{A}$

composition of answer constraints

- the S-semantics is the interpretation

$$F[\![P]\!] = T_P \uparrow \omega$$

- $\mathcal{T}_P = T_P(\mathcal{A}) = T_P(S)$

# TOP-DOWN DIAGNOSIS : THEOREMS

**Theorem**

The clause $c$ is incorrect on the observation $\mathcal{O}$ iff

$$\mathcal{O} \leq \mathcal{S}_{\{c\}} \quad \text{and} \quad \mathcal{O} \not\leq \mathcal{R}$$

**Theorem**

The observation $\mathcal{O}$ is uncovered iff

$$\mathcal{O} \leq \mathcal{R} \quad \text{and} \quad \mathcal{O} \not\leq \mathcal{S}_P$$

- can be implemented as metainterpreters
  - one oracle only
  - no need to start from symptoms
    - we only need to start from (finitely many) most general atomic goals
  - more efficient interaction with the user is possible
    - oracles for conjunctive "instantiated" goals

# EFFECTIVITY



- the diagnosis is not effective, unless the intended S-semantics S is finite
    - the bottom-up diagnosis is unfeasible, if S is infinite
    - in the top-down diagnosis, the oracle may return infinitely many answers to some queries

- we need finite approximations
    - partial specifications [Comini, Levi & Vitiello, ILPS 95]
    - widening techniques
    - $\boxed{\text{(finite) abstract domains}}$

# ABSTRACT DIAGNOSIS

- the property we consider
    - in the specification $S_\alpha$
      in the actual semantics of P

  is an <u>observable</u> $\alpha$
    - any abstraction of SLD-trees

- abstract diagnosis is based on a semantic framework where observables (and the corresponding semantics) are related (and formally derived) using abstract interpretation theory [Comini, Levi ILPS 94, Comini, Levi & Vitiello ILPS 95]
    - a taxonomy of observables
        - each item has suitable precision and compositionality properties
    - the kernel semantics collects SLD-trees
        - here we will take the (more abstract) S-semantics as collecting semantics

# OBSERVABLES AND DIAGNOSIS



- the observable $\alpha$ is a Galois insertion between $\wp(\mathcal{E})$ and an abstract domain $(D, \leq)$

- the observable must have a correct abstract immediate consequences operator $T_P^{\alpha}$, whose least fixpoint is the abstract semantics $F_\alpha[\![P]\!]$
  - the theory of diagnosis is based on the fixpoint semantics

- the observable must be condensing, i.e. the abstract behavior for any goal $G$ must be uniquely determined by the goal-independent denotation $\alpha(F[\![P]\!])$
  - if this is not the case, we cannot specify the goal-independent behavior only

- there exist two classes of observables for which the above properties hold
  - precise observables, for which $\alpha(F[\![P]\!]) = F_\alpha[\![P]\!]$
    - all the results of diagnosis w.r.t. computed answers hold
    - useful to reconstruct declarative diagnosis, where specifications can still be infinite
  - approximate observables, for which $\alpha(F[\![P]\!]) \leq F_\alpha[\![P]\!]$
    - includes finite domains, such as depth(k), POS (groundness) and various modes and types domains

# FROM THE S-SEMANTICS TO THE ABSTRACT SEMANTICS

- an abstract domain $(D_\alpha, \leq_\alpha)$ complete lattice
- a Galois insertion $(\alpha, \gamma)$ between $(\wp(\mathcal{E}), \leq)$ and $(D_\alpha, \leq_\alpha)$ satisfying the axioms of approximate denotables (9.1)
- the abstract immediate consequences operator $T_P^\alpha$

  $\forall$ most general atom $p(X_1, \dots, X_n)$

  $T_P^\alpha(I)(p(X_1, \dots, X_n)) =$

  $lub_\alpha(\{\varepsilon \mid$

  $p(X_1, \dots, X_n) \text{:-} e_1, \dots, e_k \,\square\, q_1(X_1^1, \dots, X_{n_1}^1), \dots, q_m(X_1^m, \dots, X_{n_m}^m)$ renamed clause of $P$

  $I(q_1(X_1^1, \dots, X_{n_1}^1)) = \varepsilon_1,$

  $\vdots$

  $I(q_m(X_1^m, \dots, X_{n_m}^m)) = \varepsilon_m,$

  $E = \alpha(\{e_1, \dots, e_k\}),$

  $\varepsilon = glb_\alpha(\{\{E\}, \varepsilon_1, \dots, \varepsilon_k\})|_{\alpha\, X_1, \dots, X_n}\})$

- the abstract semantics $F_\alpha[\![P]\!] = T_P^\alpha \uparrow \omega$ satisfies

  $\alpha(F[\![P]\!]) \leq F_\alpha[\![P]\!]$

- can be also computed as standard semantics of the "abstract program (Giacobazzi, Debray & Levi, JLP 94)

# DIAGNOSIS W.R.T. APPROXIMATE OBSERVABLES



$\alpha$ — approximate observable

$P$ — definite logic program

$F_\alpha[\![P]\!] = T_P^\alpha \uparrow \omega$ — abstract semantics of $P$

$\alpha(F[\![P]\!]) = \alpha(T_P \uparrow \omega) \leq F_\alpha[\![P]\!]$ — abstraction of the semantics of $P$

$S_\alpha$ — intended abstract behavior of $P$

- $P$ is <u>partially correct</u> w.r.t. $S_\alpha$

$$\alpha(F[\![P]\!]) \leq_\alpha S_\alpha$$

- $P$ is <u>complete</u> w.r.t. $S_\alpha$

$$S_\alpha \leq_\alpha \alpha(F[\![P]\!])$$

- <u>incorrect clause</u>

the clause $c \in P$ is incorrect on the (abstract) observation $\vartheta$ if

$$\vartheta \not\leq_\alpha S_\alpha \quad \text{and} \quad \vartheta \leq_\alpha T^\alpha_{\{c\}}(S_\alpha)$$

($c$ derives a wrong abstract constraint from the specification)

- <u>uncovered observation</u>

the (abstract) observation $\vartheta$ is uncovered if

$$\vartheta \leq_\alpha S_\alpha \quad \text{and} \quad \vartheta \not\leq_\alpha T_P^\alpha(S_\alpha)$$

(no clause in $P$ derives $\vartheta$ from the specification)

# THE DIAGNOSIS THEOREMS

## Theorem 1

if there are no incorrect clauses, then the program is partially correct

- if there are no incorrect clauses, $T_P^{\alpha}(S_\alpha) \leq S_\alpha$
- $S_\alpha$ is a prefixpoint of $T_P^{\alpha}$
- $F_\alpha[\![P]\!] \leq_\alpha S_\alpha \quad \rightarrow \quad \alpha(F[\![P]\!]) \leq_\alpha S_\alpha$

- theorem 3 does not hold for approximate observables

  if the program is complete, then
  if there exists an incorrect clause on $O$, then
  $O$ is not always an incorrectness symptom

  - the incorrectness might be generated by the approximation of the abstract semantics

- theorem 4 does not hold for approximate observables

  absence of uncovered observations, even under the unique fixpoint assumption, does not imply program completeness

  incompleteness bugs might be hidden by the approximation of the abstract semantics

## Theorem 6

if the program is partially correct, then
if there exists an uncovered observation $O$, then
$O$ is an incompleteness symptom

- weaker results
  - absence of incorrect clauses implies partial correctness
  - uncovered observations correspond to bugs
  - equivalent top-down characterization

## AN APPROXIMATE OBSERVABLE: DEPTH(k)



- the abstract domain of the observable $\tau_k$
  - the concrete domain, where
    an equation $x = t$ is replaced by
    an equation $x = t'$
    every subterm of $t$ at depth $\geq k$ is replaced by a fresh variable

- the abstract immediate consequences operator

$$T_P^{\tau_k}(I)(p(x_1,\dots,x_n)) =$$

$$\bigcup \{ \varepsilon \mid$$

$$p(x_1,\dots,x_n) :- e_1,\dots,e_k \,\square\, g_1(x_1',\dots,x_{n_1}'),\dots, g_m(x_1^m,\dots,x_{n_m}^m)$$

is a renamed clause of $P$

$$I(g_1(x_1',\dots,x_{n_1}')) = \varepsilon_1,$$

$$\vdots$$

$$I(g_m(x_1^m,\dots,x_{n_m}^m)) = \varepsilon_m$$

$$\varepsilon = \tau_k(glb(\{\tau_k(\{e_1,\dots,e_k\}), \varepsilon_1,\dots,\varepsilon_m\}))|_{x_1\dots x_n} \}$$

$glb$ is the one of the concrete domain

- $\tau_k$ is an approximate observable $\longrightarrow$
  - it is condensing
  - $\tau_k(F[\![P]\!]) \leq T_P^{\tau_k} \uparrow \omega = F_{\tau_k}[\![P]\!]$

# AN EXAMPLE WITH DEPTH(K) - ANSWERS

P

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

- wrong version (missing clause) of an automaton which recognizes the language $L = \{(ab)^n \mid n \geq 0\} \cup \{(ab)^n a \mid n \geq 0\}$
- the specification of the intended depth(2)-answers

$$S(accept(X)) = \{\{X=[\,]\}, \{X=[a]\}, \{X=[a,b]\}, \{X=[a,b|Y]\}\}$$

$$S(acc(X)) = \{\{X=[\,]\}, \{X=[b]\}, \{X=[b,a]\}, \{X=[b,a|Y]\}\}$$

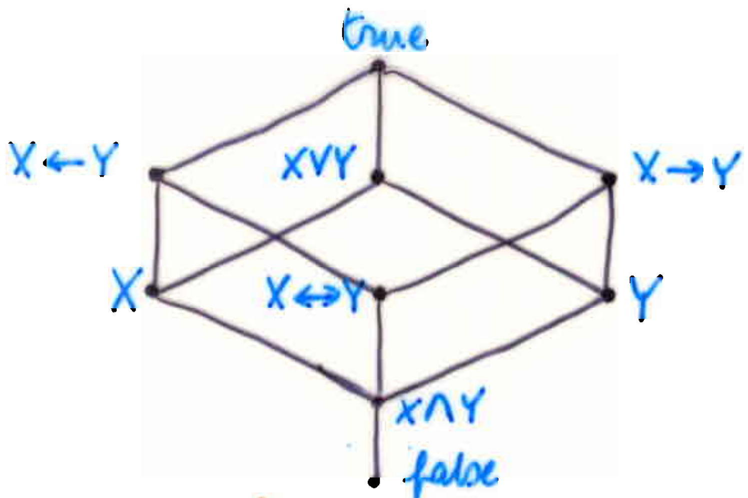
- by applying the $T_p^{\tau_2}$ operator we find out that
  - P is partially correct w.r.t. S
  - there exists an uncovered element

    $acc(X) \mapsto \{X=[\,]\}$

    which shows that there is a missing clause for acc

# AN APPROXIMATE OBSERVABLE: REPRESENTING GROUNDNESS BY POS

- the abstract domain $(POS, \leq_{POS})$ [propositional formulas]
  (shown for two variables)




$\alpha_{POS} : \wp(\mathcal{E}) \rightarrow POS$

$\gamma_{POS} : POS \rightarrow \wp(\mathcal{E})$

$false \rightarrow \emptyset$ (no answers)

$true \rightarrow \mathcal{E}$ (no groundness information)

$X \rightarrow \{ E \mid \exists t \text{ ground}, X = t \in E \}$ (X is ground)

$X \wedge Y \rightarrow \{ E \mid \exists t_1, t_2 \text{ ground}, \{X = t_1, Y = t_2\} \subseteq E \}$ (both X and Y are ground)

$X \vee Y \rightarrow \{ E \mid \exists t \text{ ground}, X = t \in E \text{ or } Y = t \in E \}$ (either X or Y is ground)

$X \rightarrow Y \rightarrow \{ E \mid \exists t, Y \text{ occurs in } t, X = t \in E \}$ (if X is ground, then Y is ground)

$X \leftrightarrow Y \rightarrow \{ E \mid \exists t, Y \text{ is the only variable in } t, X = t \in E \}$ (X is ground iff Y is ground)

- $(POS, \leq_{POS})$ is an approximate observable

# THE ABSTRACT IMMEDIATE CONSEQUENCES OPERATOR



$\forall$ most general atom $p(X_1, \ldots, X_n)$

$$T_P^{POS}(I)(p(X_1, \ldots, X_n)) =$$

$$lub^{POS}\Big(\{\varepsilon \mid$$

$$p(X_1, \ldots, X_n) :- c_1, \ldots, c_k \,\square\, q_1(X_1^1, \ldots, X_{n_1}^1), \ldots, q_m(X_1^m, \ldots, X_{n_m}^m)$$

(renamed clause of P)

$$I(q_1(X_1^1, \ldots, X_{n_1}^1)) = \varepsilon_1,$$

$$\vdots$$

$$I(q_m(X_1^m, \ldots, X_{n_m}^m)) = \varepsilon_m,$$

$$\varepsilon = glb^{POS}\Big(\{\alpha_{POS}(\{c_1, \ldots, c_k\}), \varepsilon_1, \ldots, \varepsilon_m\}\Big)\Big|^{POS}_{X_1, \ldots, X_n}\}\Big)$$

- this is the only occurrence of "concrete constraints" in the definition
  - it can be eliminated by considering the abstract program, where, in each clause,

    $c_1, \ldots, c_k$ is replaced by $\alpha_{POS}(\{c_1, \ldots, c_k\})$

- since POS is an approximate observable, we know that it is condensing
  - $\alpha_{POS}(F_\phi[\![P]\!]) \leq_{POS} T_P^{POS} \uparrow \omega = F_{POS}[\![P]\!]$

# ABSTRACT DIAGNOSIS: EXAMPLE 2

$$
\begin{aligned}
&p(X) :- X = f(Y) \,\square\, q(Y).\\
&q(X) :- X = a.\\
&r(X) :- Y = g(X) \,\square\, p(Y).\\
&s(X,Y) :- r(X).\\
&s(X,Y) :- Y = a.
\end{aligned}
$$

$$
\begin{aligned}
&F[\![P]\!](p(X)) = \{\{X = f(a)\}\}\\
&F[\![P]\!](q(X)) = \{\{X = a\}\}\\
&F[\![P]\!](r(X)) = \phi\\
&F[\![P]\!](s(X,Y)) = \{\{Y = a\}\}
\end{aligned}
$$

$$
\begin{aligned}
&\alpha_{pos}(F[\![P]\!])(p(X)) = X\\
&\alpha_{pos}(F[\![P]\!])(q(X)) = X\\
&\alpha_{pos}(F[\![P]\!])(r(X)) = false\\
&\alpha_{pos}(F[\![P]\!])(s(X,Y)) = Y
\end{aligned}
$$

$$
\begin{aligned}
&F_{pos}[\![P]\!](p(X)) = X\\
&F_{pos}[\![P]\!](q(X)) = X\\
&F_{pos}[\![P]\!](r(X)) = X\\
&F_{pos}[\![P]\!](s(X,Y)) = X \vee Y
\end{aligned}
$$

- the abstract semantics is not precise, i.e. $F_{pos}[\![P]\!] \neq \alpha_{pos}(F[\![P]\!])$

# ABSTRACT DIAGNOSIS: EXAMPLE 2

```
p(X) :- X=f(Y) □ q(Y).
q(X) :- X=a.
r(X) :- Y=g(X) □ p(Y).
s(X,Y) :- r(X).
s(X,Y) :- Y=a.
```

$$F[\![P]\!](p(X)) = \{\{X=f(a)\}\}$$
$$F[\![P]\!](q(X)) = \{\{X=a\}\}$$
$$F[\![P]\!](r(X)) = \phi$$
$$F[\![P]\!](s(X,Y)) = \{\{Y=a\}\}$$

$$\alpha_{pos}(F[\![P]\!])(p(X)) = X$$
$$\alpha_{pos}(F[\![P]\!])(q(X)) = X$$
$$\alpha_{pos}(F[\![P]\!])(r(X)) = false$$
$$\alpha_{pos}(F[\![P]\!])(s(X,Y)) = Y$$

$$F_{pos}[\![P]\!](p(X)) = X$$
$$F_{pos}[\![P]\!](q(X)) = X$$
$$F_{pos}[\![P]\!](r(X)) = X$$
$$F_{pos}[\![P]\!](s(X,Y)) = X \vee Y$$

- the abstract semantics is not precise, i.e. $F_{pos}[\![P]\!] \neq \alpha_{pos}(F[\![P]\!])$

$$S_{pos}(p(X)) = X$$
$$S_{pos}(q(X)) = X$$
$$S_{pos}(r(X)) = false$$
$$S_{pos}(s(X,Y)) = Y$$

$$T_P^{pos}(S_{pos})(p(X)) = X$$
$$T_P^{pos}(S_{pos})(q(X)) = X$$
$$T_P^{pos}(S_{pos})(r(X)) = X$$
$$T_P^{pos}(S_{pos})(s(X,Y)) = Y$$

- P is partially correct and complete
- however, the third clause is incorrect on $\mathcal{O}_1(r(X)) = X$ (because of the approximation in the abstract $T_P$)

# DIAGNOSIS AND VERIFICATION OF PARTIAL CORRECTNESS

- a specification like

$$S(+(X,Y,Z)) = X \wedge Y \wedge Z$$

reads as

"every successful computation of ?- +(X,Y,Z) binds X, Y, and Z to ground terms"

- it looks like an equation (postcondition)

- diagnosis can be compared to techniques for verifying partial correctness

(Apt 93, 94, 96)

# WHAT IS AN ASSERTION?



- verification
  - an intensional set of concrete atoms
    - closed under instantiation
    - the intensional definition is given in terms of concepts (being a list, being ground, .... ) which have to be defined elsewhere (e.g. the theory of groundness)

$$\{ +(X,Y,Z) \mid X \text{ is ground} \wedge Y \text{ is ground} \wedge Z \text{ is ground} \}$$

  - infinitely many concrete atoms

- diagnosis
  - an extensional set of abstract atoms

$$\{ +(X,Y,Z) :- X \wedge Y \wedge Z \}$$

  - one abstract atom

- the constraint on being closed under instantiation rules out some interesting properties (e.g. computed answers, freeness, .. )

- in the case of precise observables (e.g. correct answers) intensional assertions can lead to finite assertions, but

"no additional concepts, apart from the abstract operations, are needed in abstract diagnosis"

# WHAT IS A SPECIFICATION?



- verification
    - a pair of equations
        - precondition and postcondition
            - the precondition specifies the class of goals we are interested in
- diagnosis
    - one equation
        - postcondition only
            - "goal-independent" verification, or verification for "most general atomic goals"

# WHAT IS PARTIAL CORRECTNESS?

## verification

- there exist two different formalizations
  - the first one corresponds to our notion of partial correctness

    ($\approx$ the actual semantics is included in the specification)
  - the second one (strongest postcondition) corresponds to our notion of partial correctness and completeness

    ($\approx$ the actual semantics is the same as the specification)

# THE BASIC TECHNICAL TOOL

## verification

- the definition of well-asserted (greater and) programs w.r.t. a specification
  - a simple inductive definition which generates a set of formulas which have to be proved in the theory which formalized the "external" concepts used in the assertions

## diagnosis

- the application of the abstract $T_P$ to the specification
  - abstract computation instead of theorem proving

# HOW DO WE PROVE PARTIAL CORRECTNESS?

- partial correctness 1

  verification

  - partial correctness is essentially proving the well-assertedness

  diagnosis

  - absence of incorrect clauses

- partial correctness 2 (= partial correctness + completeness)

  verification

  - requires the construction of a concrete semantics (least Herbrand model of a suitable program, obtained from P and the preconditions)

  diagnosis

  - absence of uncovered elements
    - if the observable is precise and the abstract $T_P$ has one fixpoint only
    - if the observable is "precise for P" and the abstract $T_P$ has one fixpoint only

# WHAT IF THE PARTIAL CORRECTNESS PROOF FAILS?

## verification

- no useful information

## diagnosis

- uncovered elements always correspond to incompleteness bugs
- incorrect clauses correspond to incorrectness bugs, if the domain is precise

# THE STRONG POINTS

## verification

- the ability to handle specifications given in terms of pre- and post-conditions

## diagnosis

- generates information useful for debugging
- one step of abstract computation instead of theorem proving
- possibly better for proving completeness
- can use tools from abstract interpretation theory (domain composition, domain refinement) to improve the precision and the expressive power of specifications

$\Downarrow$

extension of diagnosis to handle pre and post-conditions

# MODULAR DIAGNOSIS

- the program is splitted into $n$

  predicate-disjoint modules

  - any predicate is fully defined by a single module
  - it is not a hierarchical decomposition

    (mutual recursion across modules is allowed)

- each module $P_i$ has a specification $I_\alpha^i$

- $use(P_i) = \{ I_\alpha^j \mid$ the body of a clause in $P_i$ contains a predicate defined in module $P_j \}$

- two different (strongly related) techniques

  1. (modular diagnosis)

     - all the program components are known
     - the diagnosis is performed component-wise

  2. (modular development and diagnosis)

     - we have one component only (and all the related specifications)
     - we perform the diagnosis of a single module

# MODULAR DIAGNOSIS AND COMPOSITIONALITY

- modular analysis is usually based on OR-compositional semantics
  - for example, [Codish, Debray, Giacobazzi, POPL 93] use the OR-compositional version of the s-semantics [Bossi, Gabbrielli, Levi, Meo, TCS 94]
- our semantics are not OR-compositional
- abstract diagnosis does not require to actually compute the abstract semantics
  - both top-down and bottom-up diagnosis just use the abstract immediate consequences operators $T_P^\alpha$, which are indeed OR-compositional
- we do not need any modification in the basic framework

# MODULAR DIAGNOSIS

$$P = P_1 \cup \dots \cup P_m$$

$$I_\alpha = I_\alpha^1 \cup \dots \cup I_\alpha^m$$

correctness $\quad \alpha(F[\![P]\!]) \leq_\alpha I_\alpha$

completeness $\quad I_\alpha \leq_\alpha \alpha(F[\![P]\!])$

## m-incorrect clause

the clause $c \in P_i$ is m-incorrect on the (abstract) element $\sigma$ if

$$\sigma \not\leq_\alpha I_\alpha^i \quad \text{and} \quad \sigma \leq_\alpha T^\alpha_{\{c\}}(I_\alpha^i \,\tilde{\cup}\, use(P_i))$$

## m-uncovered element

the (abstract) element $\sigma$ is uncovered if

$$\sigma \leq_\alpha I_\alpha^i \qquad \sigma \not\leq_\alpha T^\alpha_{P_i}(I_\alpha^i \,\tilde{\cup}\, use(P_i))$$

- the results
  - if there are no m-incorrect clauses, P is correct
  - if P and $\alpha$ are complete, m-incorrect clauses always correspond to incorrectness bugs
  - if P is correct, m-uncovered elements always correspond to incompleteness bugs
  - if $\alpha$ is complete and $lfp(T_P^\alpha) = gfp(T_P^\alpha)$, then if there are no m-uncovered elements, P is complete

# MODULAR DEVELOPMENT AND DIAGNOSIS

$P_i$

$use(P_i)$

- we want to perform the diagnosis of $P_i$ under the assumption that all the other (possibly not yet implemented) modules are correct and complete (w.r.t. their specifications)

- we cannot construct the concrete semantics of $P_i$ since the other components are unknown
- we can take as concrete semantics of the unknown modules the concretization of their abstract specifications

$$T_{P_i}^{I_\alpha}(I) = T_{P_i}(I \cup \gamma(use(P_i)))$$

$$F^{I_\alpha}[\![P_i]\!] = T_{P_i}^{I_\alpha} \uparrow \omega$$

module correctness $\quad \alpha(F^{I_\alpha}[\![P_i]\!]) \leq_\alpha I_\alpha^i$

module completeness $\quad I_\alpha^i \leq_\alpha \alpha(F^{I_\alpha}[\![P_i]\!])$

- the results

$$P = P_1 \cup \dots \cup P_n$$

- if all the $P_i$'s are correct (complete), then $P$ is correct / complete)
- if there are no m-incorrect clauses in $P_i$, then $P_i$ is m-correct
- if $P_i$ and $\alpha$ are complete, then if there exists an m-incorrect clause in $P_i$, $P_i$ is not m-correct
- if $P_i$ is correct, and there exists an m-uncovered element in $P_i$, $P_i$ is not m-complete
- if $\alpha$ is complete (and the m-suitable $T_{P_i}$'s have a unique fixpoint), then if there are no m-uncovered elements in $P_i$, $P_i$ is m-complete