On Completeness in Abstract Model Checking from the Viewpoint of Abstract Interpretation

Patrick COUSOT
École Normale Supérieure, 45 rue d’Ulm
75230 Paris cedex 05, France
mailto:Patrick.Cousot@ens.fr http://www.di.ens.fr/~cousot

Réunion Workshop on Implementation of Logics, November 11-12, 2000

Abstraction in Program Analysis & Model Checking

Abstract interpretation has been successfully applied in:
• static program analysis (by approximation of the semantics);
• model checking (state explosion & infinite state models).

Abstraction in Model Checking

Main abstractions in model checking:
• Implicit abstraction: to design the model of reference;
• Polyhedral abstraction (with widening): synchronous, real-time & hybrid system verification;
• Finitary abstraction (without widening): hardware & protocol verification

1 Abstracting concrete transition systems to abstract transition systems so as to reuse existing model checkers in the abstract.

Motivations & Results
Abstraction in Program Analysis & Model Checking

- The abstraction must always be **sound**;
- For **completeness**:
  - in **static program analysis**: not required (possible uncertainty);
  - in **model checking**: required\(^2\) (formal verification method\(^3\)).

\(^2\) allowing only for yes/no answers, all uncertainty resulting only from getting out of computer resources.
\(^3\) otherwise model-checking would be a mere debugging method or equivalent to program/model analysis.

Discovery of Abstractions

- in **static program analysis**:
  - task of the program analyzer designer,
  - find a **sound** abstraction providing useful information for all programs,
  - essentially manual,
  - partially automated e.g. by combination & refinement of abstract domains;
- in **model checking**:
  - task of the user,
  - find a **sound & complete** abstraction required to verify one model,
  - looking for automation (e.g. starting from a trivial or user provided guess and refining by trial and error).

Informal Objective of the Talk

- Understand the **logical nature** of the problem of finding an **appropriate abstraction** (for proving safety properties).

Formalization of the Problem
Fixpoint Checking

- Model-checking safety properties of transition systems:
  \[ \text{lfp} \leq \lambda X. I \lor F(X) \leq S ? \]

- Program static analysis by abstract interpretation:
  \[ \gamma(\text{lfp} \leq \lambda X. \alpha(I \lor F(\gamma(X)))) \leq S ? \]

Soundness / (Partial) Completeness

Soundness: a positive abstract answer implies a positive concrete answer. So no error is possible when reasoning in the abstract;

Completeness: a positive concrete answer can always be found in the abstract;

Partial completeness: in case of termination of the abstract fixpoint checking algorithm, no positive answer can be missed.
Soundness / (Partial) Completeness

**Soundness**: a positive abstract answer implies a positive concrete answer. So no error is possible when reasoning in the abstract;

**Completeness**: a positive concrete answer can always be found in the abstract;

**Partial completeness**: in case of termination of the abstract fixpoint checking algorithm, no positive answer can be missed.

*Termination/resource limitation* is therefore considered a separate problem (widening/narrowing, etc.).

Practical Question

Is it possible to **automatize the discovery of complete abstractions**?

Objective of the Talk (Formally)

Constructively characterize the abstractions \( \langle \alpha, \gamma \rangle \) for which abstract fixpoint algorithms are partially complete.

Concrete Fixpoint Checking
Concrete Fixpoint Checking Problem

- Complete lattice \( \langle L, \leq, 0, 1, \lor, \land \rangle \);
- Monotonic transformer \( F \in L \xrightarrow{\text{mon}} L \);
- Specification \( \langle I, S \rangle \in L^2 \);

\[
\text{lfp} \leq \lambda X. I \lor F(X) \leq S?
\]

Example (contd.)

- Safety specification: \( S \subseteq \Sigma \)
- Reachable states from \( I \):

\[
\text{post}[\tau^*](I) = \text{lfp} \leq \lambda X. I \cup \text{post}[\tau](X);
\]

- Satisfaction of the safety specification \( (\text{post}[\tau^*](I) \subseteq S) \):

\[
\text{lfp} \leq \lambda X. I \lor \text{post}[\tau](X) \leq S?
\]

Concrete Fixpoint Checking Algorithm

Algorithm 1

\[
X := I; \quad \text{Go} := (X \leq S);
\]

\[
\text{while Go do}
\]

\[
X' := I \lor F(X);
\]

\[
\text{Go} := (X \neq X') \& (X' \leq S);
\]

\[
X := X';
\]

\[
\text{od};
\]

\[
\text{return } (X \leq S);
\]

---

Example

- Set of states: \( \Sigma \);
- Initial states: \( I \subseteq \Sigma \);
- Transition relation: \( \tau \subseteq \Sigma \times \Sigma \);
- Transition system: \( \langle \Sigma, \tau, I \rangle \);
- Complete lattice: \( \langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap \rangle \);
- Right-image of \( X \subseteq \Sigma \) by \( \tau \):

\[
\text{post}[\tau](X) \triangleq \{ s' | \exists s \in X : \langle s, s' \rangle \in \tau \};
\]

- Reflexive transitive closure of \( \tau \): \( \tau^* \)
**Partial correctness of Alg. 1**

Alg. 1 is **partially correct**: if it ever terminates then it returns \( \text{lfp} \preceq \lambda X. I \lor F(X) \leq S \).

**Concrete Invariants**

- \( A \in L \) an **invariant** for \( \langle F, I, S \rangle \) if and only if \( I \preceq A \land F(A) \leq A \land A \preceq S \);

**Note 1** (Floyd’s proof method): \( \text{lfp} \preceq \lambda X. I \lor F(X) \leq S \) if and only if there exists an invariant \( A \in L \) for \( \langle F, I, S \rangle \);

**Note 2**: if Alg. 1 terminates successfully, then it has computed an invariant \( \chi = \text{lfp} \preceq \lambda X'. I \lor F(X') \).

**Galois connection**

A **Galois connection**, written
\[
\langle L, \preceq \rangle \Leftarrow \Rightarrow \rightarrow \langle M, \sqsubseteq \rangle,
\]

is such that:
- \( \langle L, \preceq \rangle \) and \( \langle M, \sqsubseteq \rangle \) are posets;
- the maps \( f \in L \mapsto M \) and \( g \in M \mapsto L \) satisfy
\[
\forall x \in L : \forall y \in M : f(x) \sqsubseteq y \text{ if and only if } x \preceq g(y).
\]

**Concrete Adjoinedness**

In general, \( F \) has an **adjoint** \( \tilde{F} \) such that \( \langle L, \preceq \rangle \Leftarrow \Rightarrow \rightarrow \langle L, \leq \rangle \).
Example of Concrete Adjoinedness

- $\tau^{-1}$ is the inverse of $\tau$;
- $\text{pre}[\tau] \triangleq \text{post}[\tau^{-1}]$;
- Set complement $\neg X \triangleq \Sigma \setminus X$;
- $\text{pre}[\tau](X) \triangleq \neg \text{pre}[\tau](\neg X)$;

$$
\langle \varphi(\Sigma), \subseteq \rangle \xrightarrow{\text{pre}[\tau]} \langle \varphi(\Sigma), \subseteq \rangle.
$$

The Complete Lattice of Concrete Invariants

- The set $\mathcal{I}$ of invariants for $\langle F, I, S \rangle$ is a complete lattice $\langle \mathcal{I}, \leq, \text{lfp} \leq \lambda X. I \lor F(X), \text{gfp} \leq \lambda X. S \land \overline{F}(X), \lor, \land \rangle$.

Fixpoint Concrete Adjoinedness

$$
\langle L, \leq \rangle \xrightarrow{\lambda \text{S}. \text{gfp} \leq \lambda X. S \land \overline{F}(X)} \lambda I. \text{lfp} \leq \lambda X. I \lor F(X)
$$

Proof:

$$
\text{lfp} \leq \lambda X. I \lor F(X) \leq S
\iff \exists A \in L : I \leq A \land F(A) \leq A \land A \leq S \quad (1)
\iff \exists A \in L : I \leq A \land A \leq \overline{F}(A) \land A \leq S
\iff I \leq \text{gfp} \leq \lambda X. S \land \overline{F}(X).
$$

Dual Concrete Fixpoint Checking Algorithm

**Algorithm 2**

1. $Y := S$; $Go := (I \leq Y)$;
2. while $Go$ do
   1. $Y' := S \land \overline{F}(Y)$;
   2. $Go := (Y \neq Y') \land (I \leq Y')$;
   3. $Y := Y'$;
5. od;
6. return $(I \leq Y)$;

---

Partial correctness of Alg. 2

Alg. 2 is partially correct: if it ever terminates then it returns \( lfp \leq \lambda X. I \lor F(X) \leq S \).

On (Dual) Fixpoint Checking

\[ lfp \leq \lambda X. I \lor F(X) \leq S \]

if and only if

\[ I \leq gfp \leq \lambda X. S \land \tilde{F}(X) \]

if and only if

\[ lfp \leq \lambda X. I \lor F(X) \leq gfp \leq \lambda X. S \land \tilde{F}(X) \]

The Adjoined Concrete Fixpoint Checking Algorithm

Algorithm 3

\[
X := I; \quad Y := S; \quad Go := (X \leq Y);
\]

while Go do

\[
X' := I \lor F(X); \quad Y' := S \land \tilde{F}(Y);
\]

\[
Go := (X \neq X') \land (Y \neq Y') \land (X' \leq Y');
\]

\[
X := X'; \quad Y := Y';
\]

od;

return \( X \leq Y \);

Partial correctness of Alg. 3

Alg. 3 is partially correct: if it ever terminates then it returns \( lfp \leq \lambda X. I \lor F(X) \leq S \).
Abstract Fixpoint Checking

Example: the Recurrent Abstraction in Abstract Model-Checking

- State abstraction: \( h \in \Sigma \mapsto \Sigma; \)
- Property abstraction: \( \alpha_h(X) \triangleq \{ h(x) \mid x \in X \} = \text{post}[h]; \)
- Property concretization: \( \gamma_h(Y) \triangleq \{ x \mid h(x) \in Y \} = \text{pre}[h]; \)
- Galois connection:

\[
\langle \wp(\Sigma), \subseteq \rangle \leftarrow \alpha_h \rightarrow \langle \wp(\Sigma'), \subseteq \rangle.
\]

Example (rule of signs): \( \Sigma = \mathbb{Z} \) so choose \( h(z) \) to be the sign of \( z \).

\(^6\) Considering the function \( h \) as a relation.

Abstract Fixpoint Checking Algorithm

Algorithm 4

\[
X := \alpha(I); \quad Go := (\gamma(X) \leq S);
\]
while \( Go \) do

\[
X' := \alpha(I \lor F(\gamma(X))); \quad Go := (X \neq X') \& (\gamma(X') \leq S);
\]

\[
X := X'; \quad \text{od};
\]

return if \( (\gamma(X) \leq S) \) then \text{true} else \text{I don't know};

\(^7\) In P. Cousot & R. Cousot, POPL'77, \((\gamma(X) \leq S) \in X \subseteq S'\) where \( S' = \alpha(S) \).
Partial correctness of Alg. 4

Alg. 4 is partially correct: if it terminates and returns "true" then \( \text{ifp} \leq \lambda X. I \lor F(X) \leq S \).

Example of Dual Abstraction

If

- \( \langle L, \leq, 0, 1, \lor, \land, \neg \rangle \) is a complete boolean lattice;
- \( \langle M, \sqsubseteq, \bot, \top, \sqcap, \sqcup, \sim \rangle \) is a complete boolean lattice;
- \( \langle L, \leq \rangle \overset{\gamma}{\longrightarrow} \langle M, \sqsubseteq \rangle \);

\( \tilde{\alpha} \triangleq \sim \circ \alpha \circ \sim \) and \( \tilde{\gamma} \triangleq \neg \circ \gamma \circ \neg \)

then

\( \langle L, \geq \rangle \overset{\tilde{\gamma}}{\longrightarrow} \langle M, \sqsupseteq \rangle \)

Example of Dual Abstraction (Contd.)

For the recurrent abstraction in abstract model-checking \( \alpha_h(X) \)

\( \triangleq \{ h(x) \mid x \in X \} = \text{post}[h] \) we have:

- \( \langle \varphi(\Sigma), \sqsubseteq \rangle \overset{\text{pre}[h]}{\longrightarrow}^{\text{post}[h]} \langle \varphi(\Sigma), \sqsubseteq \rangle \);
- \( \widehat{\text{pre}}[h](X) = \neg \text{pre}[h](\neg X) \) and \( \widehat{\text{post}}[h](X) = \neg \text{post}[h](\neg X) \), so:
- \( \langle \varphi(\Sigma), \sqsupseteq \rangle \overset{\text{pre}[h]}{\longrightarrow}^{\text{post}[h]} \langle \varphi(\Sigma), \sqsupseteq \rangle \).
Abstract Adjoinedness

\( \langle L, \leq \rangle \xrightarrow{\gamma}{\alpha} \langle M, \sqsubseteq \rangle, \langle L, \leq \rangle \xrightarrow{\tilde{F}} \langle L, \leq \rangle \text{ and } \langle L, \geq \rangle \xrightarrow{\gamma}{\alpha} \langle M, \sqsupseteq \rangle \) imply:

\( \langle M, \sqsubseteq \rangle \xrightarrow{\tilde{\alpha} \circ \tilde{F} \circ \gamma}{\tilde{\alpha} \circ \tilde{F}} \langle M, \sqsubseteq \rangle. \)

Partial correctness of Alg. 5

Alg. 5 is partially correct: if it terminates and returns "true" then \( \text{lfp} \leq \lambda X. I \lor F(X) \leq S \).

The Dual Abstract Fixpoint Checking Algorithm

Algorithm 5

Y := \( \tilde{\alpha}(S) \); Go := \( (I \leq \tilde{\gamma}(Y)) \);
while Go do
    \( Y' := \tilde{\alpha}(S \land \tilde{F}(\tilde{\gamma}(Y))) \);
    Go := \( (Y \neq Y') \& (I \leq \tilde{\gamma}(Y')) \);
    Y := Y';
od;
return if \( (I \leq \tilde{\gamma}(Y)) \) then true else I don’t know;

The Particular Case of Complement Abstraction

1. \( \langle L, \leq, 0, 1, \lor, \land, \neg \rangle \) is a complete boolean lattice;
2. \( \langle M, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \sim \rangle \) is a complete boolean lattice;
3. \( \langle L, \leq \rangle \xrightarrow{\gamma}{\alpha} \langle M, \sqsubseteq \rangle \); 
4. \( \langle L, \leq \rangle \xrightarrow{\tilde{F}} \langle L, \leq \rangle \);
5. \( \tilde{F} \triangleq \neg \circ F \circ \neg, \tilde{\alpha} \triangleq \sim \circ \alpha \circ \sim \) and \( \tilde{\gamma} \triangleq \neg \circ \gamma \circ \sim \).
Algorithm 6 becomes:

\[
Z := \alpha(\neg S); \quad Go := (I \land \gamma(Z) = 0);
\]

while \( Go \) do
\[
Z' := \alpha(\neg S \lor F(\gamma(Z)));
Go := (Z \neq Z') \land (I \land \gamma(Z') = 0);
Z := Z';
\]
end while

return if \((I \land \gamma(Z) = 0)\) then true else I don’t know;

Partial correctness of Alg. 6

Alg. 6 is partially correct: if it terminates and returns “true” then \( \text{lfp} \leq \lambda X. I \lor F(X) \leq S \).
Further Requirements for Program Static Analysis

- In program static analysis, one cannot compute $\gamma$, $\tilde{\gamma}$ and $\leq$ and sometimes neither $I$ nor $S$ may even be machine representable;
- So Alg. 7, which can be useful in model-checking, is of limited interest in program static analysis;
- Such problems do no appear in abstract model checking since the concrete model is almost always machine-representable (although sometimes too large).

Example: the Recurrent Abstraction in Abstract Model-Checking

Continuing with the abstraction of p. 31 with

$$\alpha \triangleq \text{post}[h] \quad \gamma \triangleq \text{pre}[h]$$

and

$$\tilde{\alpha} \triangleq \text{post}[h] \quad \tilde{\gamma} \triangleq \text{pre}[h],$$

we have:

1. $\forall X \in L : \gamma \circ \tilde{\alpha}(X) \subseteq X$;
2. $\forall X \in L : X \subseteq \tilde{\gamma} \circ \alpha(X)$.

Additional Hypotheses

In order to be able to check termination in the abstract, we assume:

1. $\forall X \in L : \gamma \circ \tilde{\alpha}(X) \leq X$;
2. $\forall X \in L : X \leq \tilde{\gamma} \circ \alpha(X)$.

The Adjoined Abstract Fixpoint Abstract Checking Algorithm

Algorithm 8

1. $X := \alpha(I)$; $Y := \tilde{\alpha}(S)$; $Go := (X \subseteq Y)$;
2. while $Go$ do
3. 
4. $X' := \alpha(I) \sqcup \alpha \circ F \circ \gamma(X)$; $Y' := \tilde{\alpha}(S) \sqcap \tilde{\alpha} \circ \tilde{F} \circ \tilde{\gamma}(Y)$;
5. $Go := (X \neq X') \& (Y \neq Y') \& (X' \subseteq Y')$;
6. $X := X'$; $Y := Y'$;
7. od;
8. return if $X \subseteq Y$ then true else I don’t know;
Partial correctness of Alg. 8

Alg. 8 is partially correct: if it ever terminates and returns "true" then $\text{ifp} \leq \lambda X. I \lor F(X) \leq S$.

Partially Complete Abstraction (definition)

Definition 9 The abstraction $\langle \alpha, \gamma \rangle$ is partially complete if, whenever Alg. 4 terminates and $\text{ifp} \leq \lambda X. I \lor F(X) \leq S$ then the returned result is "true".

Characterization of Partially Complete Abstractions for Algorithm 4

Theorem 10 The abstraction $\langle \alpha, \gamma \rangle$ is partially complete for Alg. 4 if and only if $\alpha(L)$ contains an abstract value $A$ such that $\gamma(A)$ is an invariant for $\langle F, I, S \rangle$.

Observations: this notion of partial completeness is different from the notions of fixpoint completeness ($\alpha(\text{fixpoint}) = \text{fixpoint } \alpha + G = \text{fixpoint } (\alpha \circ G + \gamma)$) and the stronger one of local completeness ($\alpha \circ G = \alpha \circ G \circ \gamma$) considered in P. Cousot & R. Cousot, POPL '79.
Characterization of Partially Complete Abstractions for Algorithm 4

**Theorem 10** The abstraction \( \langle \alpha, \gamma \rangle \) is partially complete for Alg. 4 if and only if \( \alpha(L) \) contains an abstract value \( A \) such that \( \gamma(A) \) is an invariant for \( \langle F, I, S \rangle \).

**Intuition:** finding a partially complete abstraction is logically equivalent to making an invariance proof.

---

**The Most Abstract Partially Complete Abstraction (Definition)**

**Definition 11** The most abstract partially complete abstraction \( \langle \alpha, \gamma \rangle \), if it exists, is defined such that:
1. The abstract domain \( \overline{M} = \overline{\alpha}(L) \) has the smallest possible cardinality;
2. If another abstraction \( \langle \alpha', \gamma' \rangle \) is a partially complete abstraction with the same cardinality, then there exists a bijection \( \beta \) such that \( \forall x \in \overline{M} : \gamma'((\beta(x))) \leq \gamma(x) \).

---

**The Least Abstract Partially Complete Abstraction (Definition)**

**Definition 13** Dually, the least abstract partially complete abstraction \( \langle \alpha, \gamma \rangle \), if it exists, is defined such that:
1. The abstract domain \( \overline{M} = \overline{\alpha}(L) \) has the smallest possible cardinality;
2. If another abstraction \( \langle \alpha', \gamma' \rangle \) is a partially complete abstraction with the same cardinality, then there exists a bijection \( \beta \) such that \( \forall x \in \overline{M} : \gamma((\beta(x))) \leq \gamma'(\beta(x)) \).

---

9. Otherwise stated, the abstract values in \( \overline{\alpha}(L) \) are more approximate than the corresponding elements in \( \alpha'(L) \).

10. Otherwise stated, the abstract values in \( \overline{\alpha}(L) \) are less approximate than the corresponding elements in \( \alpha'(L) \).
Characterization of the Least Abstract Complete Abstraction

**Theorem 14**  Dually, the least abstract partially complete abstraction for Alg. 4 is such that:

- if \( I = 1 \) then \( M = \{ T \} \) where \( \alpha \overset{\Delta}{=} \lambda X \cdot T \) and \( \gamma \overset{\Delta}{=} \lambda Y \cdot 1; \)
- if \( I \neq 1 \) then \( M = \{ \bot, T \} \) where \( \bot \in \bot \subseteq T \subseteq T \) with \( \langle \alpha, \gamma \rangle \) such that:
  \[
  \alpha(X) \overset{\Delta}{=} \text{if } X \leq \text{lfp} \leq X. I \lor F(X) \text{ then } \bot \text{ else } T
  \]
  \[
  \gamma(\bot) \overset{\Delta}{=} \text{lfp} \leq X. I \lor F(X)
  \]
  \[
  \gamma(T) \overset{\Delta}{=} 1
  \]

The Complete Lattice of Minimal Complete Abstractions for Alg. 4

**Theorem 16**  
- The relation \( \langle \{ \bot, T \}, \subseteq, \alpha, \gamma \rangle \preceq \langle \{ \bot', T' \}, \subseteq', \alpha', \gamma' \rangle \) if and only if \( \gamma(\bot) \leq \gamma'(\bot') \) is a pre-ordering on \( \mathcal{A} \).
- Let \( \langle \{ \bot, T \}, \subseteq, \alpha, \gamma \rangle \cong \langle \{ \bot', T' \}, \subseteq', \alpha', \gamma' \rangle \) if and only if \( \gamma(\bot) = \gamma'(\bot') \) be the corresponding equivalence.
- The quotient \( \mathcal{A}/\cong \) is a complete lattice for \( \preceq \) with infimum \( \langle M, \subseteq, \alpha, \gamma \rangle \) and supremum \( \langle \bar{M}, \subseteq, \bar{\alpha}, \bar{\gamma} \rangle \).

Intuition for Minimal Partially Complete Abstractions

**Theorem 15**  
- The set \( \mathcal{A} \) of partially complete abstractions of minimal cardinality for Alg. 4 is the set of all abstract domains \( \langle M, \subseteq, \alpha, \gamma \rangle \) such that \( M = \{ \bot, T \} \) with \( \bot \subseteq \bot \subseteq T \subseteq T \), \( \langle L, \preceq \rangle \overset{\alpha}{\mapsfrom} \langle M, \subseteq \rangle \), \( \gamma(\bot) \in I \) and \( \bot = T \) if and only if \( \gamma(T) \in I \).

Similar results hold for the other Algs. 6, 7 & 8.
On the Automatic Inference of Partially Complete Abstractions

- The automatic inference/refinement of abstractions is an active subject of research \(^{12}\);
- Automating the abstraction is logically equivalent to discovering an invariant and checking a proof obligation (Th. 10);

\(^{12}\) Graf & Loiseaux, CAV’93; Loiseaux, Graf, Sifakis, Bouajjani & Bensalem FMSD(6:1)’95, Graf & Saidi, CAV’97; Bensalem, Lakhnech & Owre CAV’98; Colon & Uribe, CAV’98; Abdulla, Annichini, Bensalem, Bouajjani, Habermehl & Lakhnech, CAV’99; Das, Dill & Park, CAV’99; Saidi & Shankar, CAV’99; Saidi, SAS’00; Baumgartner, Tripp, Aziz, Singhal & Andersen, CAV’00; Clarke, Grumberg, Jha, Lu & Veith, CAV’00; etc.
On the Automatic Inference of Partially Complete Abstractions (contd.)

Will the empirical methods (presently) used in abstract model-checking be able to automatize the discovery of partially complete abstractions? 13

May be not so abstract model-checking will eventually boil down to incomplete abstract interpretations as used in program analysis or program debugging, using a simultaneous simulation of program executions (although the current per-example reasoning can go on for ever).

THE END, THANK YOU.