A minimalistic look at widening operators

David Monniaux

A minimalistic look at widening operators

David Monniaux

February 20, 2009

Abstract

We consider the problem of formalizing the familiar notion of widening in abstract interpretation in higher-order logic. It turns out that many axioms of widening (e.g. widening sequences are increasing) are not useful for proving correctness. After keeping only useful axioms, we give an equivalent characterization of widening as a well-founded tree. In type systems supporting dependent products and sums, this tree can be made to reflect the condition of correct termination of the widening sequence.

1 Abstract interpretation

We shall first recall the usual definitions of abstract interpretation and widening operators.

Abstract interpretation is a framework for formalizing approximation relationships arising in program semantics and static analysis [Cousot and Cousot, 1992]. *Soundness* of the abstraction is expressed by the fact that the approximation takes place in a controlled direction. If, through static analysis, we wish to derive information that some event is unreachable, then we can try computing some kind of superset of the set of reachable states (an *over-approximation*) and show that this set does not intersect the event. If we wish to obtain a set of initial states that necessarily result in some event further down the program, we compute some *under-approximation* of the set of initial states that verify that property. Because most static analysis takes place with over-approximations, we shall only consider this case in this article.

Most introductory material on abstract interpretation describe abstraction as a *Galois connection* between a concrete space S (say, the powerset $\mathcal{P}(\Sigma)$ of the set of states Σ of the program) and an abstract space $S^\sharp$. For instance, if the program state consists in three integer variables, $\Sigma = \mathbb{Z}^3$, $S = \mathcal{P}(\mathbb{Z}^3)$ and the abstract state can be, for instance, $\{\bot\} \cup I^3$, where I is the set of well-formed intervals (a, b) ($a \in \mathbb{Z} \cup \{-\infty\}$, $b \in \mathbb{Z} \cup \{+\infty\}$ and $a \leq b$) and $\bot$ is a special element meaning "unreachable". S and $S^\sharp$ are ordered; here, S is ordered by set inclusion $\subseteq$ and $S^\sharp$ is ordered by $\sqsubseteq$: $\bot \sqsubseteq x^\sharp$ for all x in $S^\sharp$, and $((a_1, b_1), (a_2, b_2), (a_3, b_3)) \sqsubseteq ((a'_1, b'_1), (a'_2, b'_2), (a'_3, b'_3))$ if for all $1 \leq i \leq 3$, $a'_i \leq a_i$ and $b'_i \geq b_i$.

S and $S^\sharp$ are connected by an *abstraction function* α and a *concretization function* γ. γ maps any abstract state $x^\sharp$ to the set of concrete states that it represents; here, $\gamma((a_1, b_1), (a_2, b_2), (a_3, b_3))$ is the set of triples (v_1, v_2, v_3) such that for all $1 \leq i \leq 3$, $a_i \leq v_i \leq b_i$. α maps a set x of concrete states to the "best" (least) abstract element $x^\sharp$ such that $x \subseteq \gamma(x^\sharp)$. Here, if $x \subseteq \mathbb{Z}^3$, then

for all $1 \leq i \leq 3$, $a_i = \inf_{(v_1,v_2,v_3)\in x} v_i$ and $b_i = \sup_{(v_1,v_2,v_3)\in x} v_i$. $\subseteq$ and $\sqsubseteq$ must be compatible: if $x^\sharp \sqsubseteq y^\sharp$, then $\gamma(x^\sharp) \subseteq \gamma(y^\sharp)$.

Abstract interpretation replaces a possibly infinite number of concrete program execution, which cannot be simulated in practice, by a simpler "abstract" execution. For instance, one may replace running a program using our three integer variables over all possible initial states by a single abstract execution with interval arithmetic. The resulting final interval is guaranteed to contain all possible outcomes of the concrete program. More formally, if one has a transition relation $\tau \subseteq \Sigma \times \Sigma$, one defines the forward concrete transfer function $f_\tau : S \to S$ as $f_\tau(x) = \{\sigma' \mid \sigma \to_\tau \sigma' \wedge \sigma \in x\}$. $f_\tau(W)$ is the set of states reachable in one forward step from W. We say that ${f_\tau}^\sharp(x^\sharp)$ is a correct abstraction for f_τ if for all $x^\sharp$, $f_\tau \circ \gamma(x^\sharp) \subseteq \gamma \circ {f_\tau}^\sharp(x^\sharp)$. This means that if we have a superset of the concrete precondition, we get a superset of the concrete postcondition.

As usual in program analysis, obtaining loop invariants is the hardest part. Given a set $x_0 \subseteq \Sigma$ of initial states, we would like to obtain a supserset of the set of reachable states $x_\infty = \{\sigma' \mid \sigma \to_\tau^* \sigma' \wedge \sigma \in x_0\}$. Equivalently, one may define the sets of states x_n reachable in at most n steps from x_0 by induction: $x_{n+1} = f_\tau(x_n)$, then take their union $\bigcup_{n=0}^\infty x_n$, which is the least fixed point of $x \mapsto x_0 \cup f_\tau(x)$. Also equivalently, this is the least inductive invariant containing x_0, that is, the least set x_∞ containing x_0 and stable by τ ($f_\tau(x_\infty) \subseteq x_\infty$): $x_\infty = \bigcap\{x \mid f_\tau(x) \subseteq x\}$.

If $x_0^\sharp$ is an abstraction of x_0 ($x_0 \subseteq \gamma({x_0}^\sharp)$), and we define $x_{n+1}^\sharp = f_\tau^\sharp(x_n^\sharp)$, then for all n, ${x_n}^\sharp$ is an abstraction of x_n. If $S^\sharp$ is a complete lattice (any subset of $S^\sharp$ has a least upper bound and a greatest lower bound), then $x_\infty^\sharp = \bigsqcap_{n=0}^\infty x_n^\sharp$ exists and is an abstraction of x_∞. This result can be also obtained from the fact that $f_\tau^\sharp(x_\infty^\sharp) \sqsubseteq x_\infty^\sharp$.

For practical implementations, we cannot afford to wait until convergence at infinity. If $S^\sharp$ has infinite ascending sequences, then we need some way to enforce convergence in a finite amount of iterations. In some cases, least inductive invariants may be computed directly (e.g. as solutions to constraint solving problems), but often, one has to use a *widening operator* $\triangledown$ that verifies three properties: $x^\sharp \sqsubseteq x^\sharp \triangledown y^\sharp$, $y^\sharp \sqsubseteq x^\sharp \triangledown y^\sharp$, and for any sequence $v_n^\sharp$, a sequence of the form $u_{n+1}^\sharp = u_n^\sharp \triangledown v_n^\sharp$ is ultimately stationary.

We can then use $u_0^\sharp = \bot$, $u_{n+1}^\sharp = x_0^\sharp \sqcup f_\tau^\sharp(u_n^\sharp)$, and name $u_\infty^\sharp$ the stationary limit. $u_\infty^\sharp$ is an abstraction of x_∞. This can be proved by: $x_0^\sharp \sqcup f_\tau^\sharp(u_\infty^\sharp) \sqsubseteq u_\infty^\sharp$, thus $x_0 \cup \gamma \circ f_\tau^\sharp(u_\infty^\sharp) \subseteq \gamma(u_\infty^\sharp)$. Since $f_\tau(u_\infty^\sharp) \circ \gamma(u_\infty^\sharp) \subseteq \gamma \circ f_\tau^\sharp(u_\infty^\sharp)$, it follows that $f_\tau(u_\infty^\sharp) \circ \gamma(u_\infty^\sharp) \subseteq \gamma(u_\infty^\sharp)$, and thus $\gamma(u_\infty^\sharp)$ is an inductive invariant for τ that contains x_0, and thus contains the least of such invariants, x_∞.

Remark that whether $S^\sharp$ is a complete lattice, whether $f_\tau^\sharp$ is monotonic, are irrelevant to this result. The only relevant axioms are that f_τ is monotonic (which is always the case if it derives from a transition relation), that the concrete space is a complete lattice (again, this is always the case if it is the powerset of the state space), that the abstraction is sound, that the abstract ordering $\sqsubseteq$ is compatible with the concrete ordering $\subseteq$, and that the widening system always terminates with some $u_{n+1}^\sharp$ such that $u_{n+1}^\sharp \sqsubseteq u_n^\sharp$. This motivates our work.

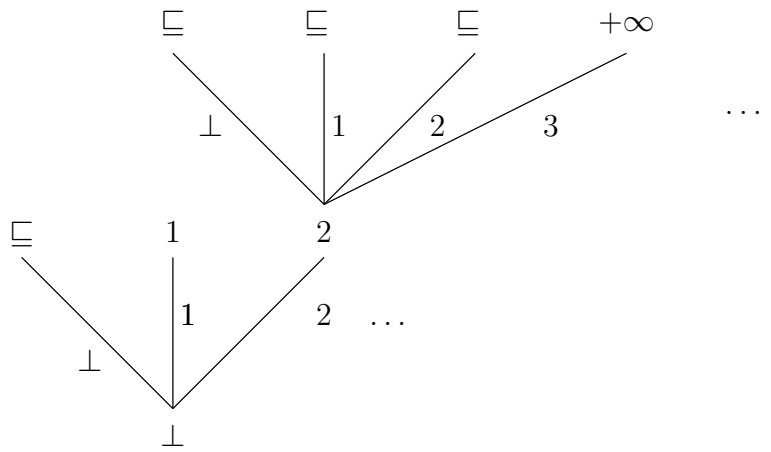

Figure 1: Interpretation of widening as a well-founded tree

2 Relaxation of conditions and interpretation in inductive types

When formalizing the notion of widening in the Coq proof assistant, we realized that the conditions described above were too restrictive: most axioms about widenings are actually not needed for proving the soundness of analyses. Pichardie [2005, §4.4] already proposed a relaxation of these conditions, but his definition of widenings is still fairly complex. We propose here a drastically reduced definition of widenings, which subsumes both the $\sqsubseteq$ ordering and the $\triangledown$ operator.

Definition 1. A widening system is an algorithm that proposes successive abstract elements $u_0^\sharp, u_1^\sharp, \ldots, u_n^\sharp$ to an abstract transformer $\phi^\sharp : S^\sharp \to S^\sharp$, and receives $\phi^\sharp(u_n^\sharp)$ (in practical use, $\phi^\sharp$ will correspond to the concrete transformer ϕ of a loop or, more generally, of a monotonic system of semantic equations). It can then either terminate with some guarantee that $\gamma \circ \phi^\sharp(u_n^\sharp) \subseteq \gamma(u_n^\sharp)$, either propose the next element $u_{n+1}^\sharp$. The system never provides infinite sequences.

It is obvious that any widening that verifies the conditions of §1 also verifies these conditions. Note that Def. 1 is strictly laxer than §1. For instance, we make no requirement that $\gamma(u_n^\sharp) \subseteq \gamma(u_{n+1}^\sharp)$; a widening system could first try some ascending sequence $u_0^\sharp, \ldots, u_n^\sharp$, realize that it is probably a bad idea to go this way, and restart with another sequence $u_{n+1}^\sharp, \ldots$.

Definition 1 can be easily recast as couple of mutually inductive types :

$$\begin{array}{ll} \mathit{widening} & \cong S^\sharp \times (S^\sharp \to \mathit{answer}) \\ \mathit{answer} & \equiv \mathit{termination} \mid \mathit{next} \text{ of } \mathit{widening} \end{array} \tag{1}$$

These types define a tree. A run of the widening system, that is, a sequence $u_0^\sharp, u_1^\sharp, \ldots, u_n^\sharp$, corresponds to a path in the tree. The absence of infinite widening sequences means that the tree should be well-founded. Note that, even in an eager language such as Objective Caml, this tree is never constructed in

memory: its nodes are constructed on demand by application of the function $S^\sharp \to answer$.

In a higher-order type system with dependent sums and products such as the Calculus of inductive constructions (as in Coq), the above inductive datatype can be adorned with proof terms. A tree node *widening* is a pair $(u^\sharp, a)$. $a(v^\sharp)$ is either *termination*, carrying a proof term stating that $\gamma(v^\sharp) \subseteq \gamma(u^\sharp)$, or another *widening* tree node.

3 Implementation in Coq

We implemented this definition into Coq, and it gave a very terse and usable definition of widening. We assume we have an abstract domain `S` with a decidable ordering `domain_le` (representing $\sqsubseteq$):

```
Variable S : Set.
Hypothesis domain_le : S -> S -> Prop.
Hypothesis domain_le_decide :
    forall x y : S,
      { domain_le x y } + {~ (domain_le x y) }.

Inductive widening: Set :=
  widening_intro : forall x : S,
    (forall y : S, widening + {domain_le y x}) -> widening.
```

`(abstract_lfp_rec f widening)` computes x such that `(domain_le` x x`)` using the widening chain `widening`:

```
Section Recursor.
Variable f : S -> S.

Fixpoint abstract_lfp_rec
  (iteration_step : widening) :
  { lfp : S | domain_le (f lfp) lfp } :=
  let (x, xNext) := iteration_step in
  match xNext (f x) with
  | inleft next_widening => abstract_lfp_rec next_widening
  | inright fx_less_than_x => exist (fun x => domain_le (f x) x)
      x fx_less_than_x
  end.

End Recursor.
```

For ease of use, we pack `S`, `domain_le`, an abstraction relation `domain_abstracts` and other related constructs into one single `domain` record. (domain_abstracts $x^\sharp$ x) means that $x \subseteq \gamma(x^\sharp)$.

In numerical abstract domains, it is common to use "widening up to" or "widening with thresholds": one keeps an ascending sequence $z_1^\sharp, \ldots, z_n^\sharp$ of "special" values, and $x^\sharp \triangledown y^\sharp$ is the least element $z_k^\sharp$ greater than $x^\sharp \sqcup y^\sharp$. This is easily achieved within this framework by a "widening transformer": taking a widening W as input and a finite list l of values, it outputs a widening W' that first applies the thresholds and, as a last resort, calls W. (Please note that `Variable T : domain` is a parameter including the original widening ramp.)

```
Section Widening_ramp.
Variable T : domain.

Fixpoint ramp_widening_chain_search (bound : (domain_set T))
  (ramp : (list (domain_set T))) { struct ramp } :
  (list (domain_set T)) :=
...

Fixpoint ramp_widening_chain (ramp : (list (domain_set T))) :
  (widening_chain (domain_set T) (domain_le T)) := ...
```

One can choose to delay widening by n steps of $\sqcup$ after each widening step. This is again implemented as a "widening transformer":

```
Definition delayed_widening_each_step :
  nat -> (widening_chain (domain_set T) (domain_le T)).
```

We can similarly build a product domain $S_1^\sharp \times S_2^\sharp$. The widening on couples $(a_1, a_2)\nabla(b_1, b_2) = (a_1 \nabla_1 b_1, a_2 \nabla_2 b_2)$ is implemented by a "widening transformer" taking one widening W_1 on $S_1^\sharp$ and a widening W_2 on $S_2^\sharp$ as inputs, and producing a widening on $S_1^\sharp \times S_2^\sharp$ by syntactic induction on W_1 and W_2: if $a_1 \sqsubseteq_1 b_1 \wedge a_2 \sqsubseteq_2 b_2$, then $(a_1, a_2) \sqsubseteq (b_1, b_2)$ for the product ordering and one terminates; if $a_1 \sqsubseteq_1 b_1$ but $a_2 \not\sqsubseteq_2 b_2$ then one stays on a_1 but moves one step into W_2 (and *mutatis mutandis* reversing the coordinates); if $a_1 \sqsubseteq_1 b_1$ and $a_2 \not\sqsubseteq_2 b_2$, then one moves into both W_1 and W_2. This implements the usual widening on products. This construct can be generalized to any finite products of domains.

4 Conclusion

By seeing the combination of the computational ordering $\sqsubseteq$ and the widening operator ∇ as a single inductive construct, one obtains an elegant characterization extending the usual notion of widening in abstract interpretation, suitable for implementation in higher order logic.

References

P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. of Logic and Computation*, pages 511–547, Aug. 1992.

D. Pichardie. *Interprétation abstraite en logique intuitionniste : extraction d'analyseurs Java certifiés*. PhD thesis, Université Rennes 1, 2005. In French.