Learning based Widening

Vivek Notani
Roberto Giacobazzi
University of Verona
Italy
IMDEA Software Institute
Spain
vivek.notani@univr.it,roberto.giacobazzi@univr.it

Abstract

Even though design of a widening operator is an integral step in the design of an abstract interpreter using an infinite domain, not much work has been done to systematize the design of widening operator. While there exist works that derive widening of higher-level domains by lifting the widening of the base-level domain, the design of widening for base-level domains remains largely creative process, especially for numerical domains. This is partly because the definition of widening requires convergence and termination while providing for very weak algebraic properties, making the problem of creating generic widening construction techniques hard.

We provide a template widening operator, parametric on the shape of the domain, by leveraging a duality between Supervised Machine Learning (Regression) and Abstract Interpretation (Widening).

CCS Concepts •**Theory of computation → Program analysis;**

Keywords Abstract Interpretation, Widening, Machine Learning, Regression

ACM Reference format:
Vivek Notani and Roberto Giacobazzi. 2017. Learning based Widening. In *Proceedings of ACM SIGPLAN Workshop on Tools for Automatic Program Analysis, New York, NY, USA, August 29, 2017 (TAPAS'17),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

Problem & Motivation

Consider the typical creative design process [6] of an abstract interpreter faced by Alice. Alice starts with a set of representative programs (benchmarks) and tries to extract the (often geometrical) structure of their invariants by dynamically monitoring their behavior. Then Alice needs to generalize her intuitions towards a domain of approximate objects (abstract domain) and approximate transfer functions (abstract semantics) for each operation of the programming language [4]. Finally, she designs the widening operations for fix-point acceleration. In-fact, the last step is largely creative and often expensive.

TAPAS'17, New York, NY, USA
2017. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

In this work, we provide a way to automatically derive widening operator for Alice, once she has designed the domain, by leveraging a duality between Supervised Machine Learning (Regression) and Abstract Interpretation (Widening).

Novel Approach

Our thesis is that widening, like regression, is an inductive learner. Regression generalizes known states to a hypothesis. Widening generalizes abstract states on a iteration chain to a post fix point. While Regression usually aims to minimize the total error (sum of false positives and false negatives), Widening aims for soundness and hence errs on the side of false positives to have zero false negatives. We use this duality to derive a generic widening operator from regression on the set of abstract states.

Widening by Learning

Intervals was the first infinite abstract domain introduced by Cousot [5]. Next came the polyhedra [8] and octagons [10]. The widening for polyhedra has since been improved upon [1]. We observe that all of these widening work by setting unstable bounds to infinity. For example $[1,2]\nabla[1,3] = [1,\infty]$.

We view the problem of identifying unstable bounds as a learning theory problem. We start by representing an abstract domain as a constraint system. Set of states are then represented by a conjunction of constraints CS.

Consider a constraint system defined as the three tuple $\langle CS, \top, \bot\rangle$ where $CS = \langle CS_{var}, \cap, \phi\rangle$ is either empty set ϕ, a constraint or a conjunction of constraints from the constraint set $CS_{var} = \{\vec{P}^T.\vec{X} \leqslant \lambda\}$ and $\vec{X}$, $\vec{P}^T$ and λ are defined below.

$$\begin{aligned} \vec{X} &= \phi(\vec{X'}) \\ P &\text{ weight vector} \\ \lambda &\text{ constant} \\ X'_i &\in \text{ Var} \end{aligned} \tag{1}$$

We may also represent the entire constraint set as $CS = \{P'^T.\vec{X} \leqslant \lambda'\}$ where P' is $m \times n$ matrix with column vectors $\langle P_1, P_2, ...P_n\rangle$, $\vec{X}$ is a vector with m dimensions and λ' is a column vector in n dimensions.

In this domain model, we define widening ∇ as a binary operator on the constraint set CS as follows:

$$\begin{aligned}&\nabla : CS \times CS \to CS \text{ such that:}\\ &\quad CS_1 \nabla CS_2 \supseteq CS_1, CS_2 \text{ and}\\ &\quad \forall \{CS_i\} \subseteq CS \text{ chain } \{CS_i^{\nabla}\} \text{ is stable finitely}\end{aligned} \tag{2}$$

We view this as a two step process-

1. Learn: Here, widening operator uses the abstract states fed to it at two different iterations to learn an approximate transformer (hypothesis in machine learning terminology) that relates abstract states to iteration count. Thus the long chain is approximated to a transformer.
2. Drop: Here, we obtain an upper-approximation of the constraint set that is comprised of only the stable constraints. The approximated transformer is stabilized in a sound fashion by an existential elimination of iteration count in the constraints which usually amounts to dropping the unstable constraint.

We propose to use linear regression to learn instability in constraints. Constraint set obtained by dropping unstable constraints would then satisfy both the conditions above and hence, provide a suitable widening operator.

To learn the instability, we use the linear model to learn the relationship between an element in matrix P' (and λ') and the iterator i. A linear model will always learn the element of P' (and λ') as either a constant or some linear function of i. Clearly, a constant element corresponds to a constant co-efficient w.r.t. iterations in the chain. Any other linear function corresponds to an instability as we move up the chain. Note that using a more precise model may get a better approximation of how the co-efficient varies with each iteration; However, since we only care to learn whether or not the coefficient is constant, linear regression is the most efficient solution.

Given two points, linear regression simply learns the straight line joining the two points. Hence, given two constraint sets $CS_{i=1} = \{{}_1P'^T.\vec{X} \leqslant_1 \lambda'\}$ and $CS_{i=2} = \{{}_2P'^T.\vec{X} \leqslant_2 \lambda'\}$, the learned constraint set would be: $CS_i = \{{}_iP'^T.\vec{X} \leqslant_i \lambda'\}$ where ${}_iP'$ and ${}_i\lambda'$ as defined below.

$$\begin{aligned}&{}_iP' = [{}_iP'_{k,l}] \text{ where } k \in [1,m],\ l \in [1,n] \text{ and}\\ &{}_i\vec{P'}_l = ({}_2\vec{P'}_l -_1 \vec{P'}_l).(i-1) +_1 \vec{P'}_l\\ &{}_i\vec{\lambda'} = [{}_i\lambda'_{l,1}] \text{ where } l \in [1,n]\\ &{}_i\lambda'_l = ({}_2\lambda'_{l,1} -_1 \lambda'_{l,1}).(i-1) +_1 \lambda'_{l,1}\end{aligned} \tag{3}$$

Next, to obtain $CS_1 \nabla CS_2$, we want an existential elimination of i. So we drop the rows in matrix ${}_3P'^T$ and ${}_3\lambda'$, that have a non-zero coefficient to i. Note since we want to drop unstable constraints, dropping a row in one matrix ($P'^T or \lambda'$) is accompanied by dropping the corresponding row in the other matrix as well. The widening result is thus defined as $CS_1 \nabla CS_2 = \{P'^T.\vec{X} \leqslant \lambda'\}$ where:

$$\begin{aligned}&{}_iP' = [{}_iP'_{k,l}] \text{ where } k \in [1,m],\ l \in [1,n] \text{ and}\\ &{}_i\vec{P'}_l =_1 \vec{P'}_l \text{ if } \forall k \in [1,m] {}_2P'_{k,l} =_1 P'_{k,l}\\ &\quad = 0 \text{ otherwsise}\\ &{}_i\vec{\lambda'} = [{}_i\lambda'_{l,1}] \text{ where } l \in [1,n]\\ &{}_i\lambda'_l =_1 \lambda'_{l,1} \text{ if } ({}_2\lambda'_{l,1} =_1 \lambda'_{l,1}) \wedge_i \vec{P'}_l \neq 0\\ &\quad = 0 \text{ otherwise}\end{aligned} \tag{4}$$

Dropping a conjunct results in a more relaxed constraint set. Also, since the resultant constraint set is independent of i, it is indeed a fix-point to the chain. Hence, it satisfies the conditions of widening as defined in equation-2.

Example: Interval

Next we describe the process via an example. Consider P : `x = 1; y = 0; while x < 10 {x++; y++}`. We use the interval [5] domain I to infer the invariant that defines the bounds on x and y. Intervals in two dimensions are conjunction of four lines: $a_1 \leqslant x \leqslant a_2 \wedge a_3 \leqslant y \leqslant a_4$ where real valued $a_1, a_2, ..., a_4$. The constraint sets at iteration i=2 and i=3 can be expressed in the matrix form as:

$$CS_{i=2} = \begin{bmatrix}1&0\\-1&0\\0&1\\0&-1\end{bmatrix} \times \begin{bmatrix}x\\y\end{bmatrix} \leqslant \begin{bmatrix}2\\-1\\1\\0\end{bmatrix}$$

$$CS_{i=3} = \begin{bmatrix}1&0\\-1&0\\0&1\\0&-1\end{bmatrix} \times \begin{bmatrix}x\\y\end{bmatrix} \leqslant \begin{bmatrix}3\\-1\\2\\0\end{bmatrix}$$

Learn step gives the new constraint set as a linear function of the iterator i:

$$CS_i = \begin{bmatrix}1&0\\-1&0\\0&1\\0&-1\end{bmatrix} \times \begin{bmatrix}x\\y\end{bmatrix} \leqslant \begin{bmatrix}i\\-1\\i-1\\0\end{bmatrix}$$

Next, in Drop step we drop all rows with i in either matrix to obtains the widening.

$$CS_2 \nabla CS_3 = \begin{bmatrix}-1&0\\0&-1\end{bmatrix} \times \begin{bmatrix}x\\y\end{bmatrix} \leqslant \begin{bmatrix}-1\\0\end{bmatrix}$$

Clearly the result is a stable fix point and a super-set of all iterates. Additionally, widening obtained via our template operator is the same as that obtained by traditional methods.

Example: Octagon Widening

Consider P : `x = 1; y = 0; while x < 10 {x++; y++}`. We must use the domain of octagons O to infer the invariant that relates x and y [10]. Octagons are conjunction of eight lines: $a_1 \leqslant x \leqslant a_2 \wedge a_3 \leqslant y \leqslant a_4 \wedge a_5 \leqslant a_6 x + y \leqslant a_7 \wedge a_8 \leqslant -a_6 x + y \leqslant a_9$ where real valued $a_1, a_2, ..., a_9$. In order to compute $[\![P]\!]^O$ during the fourth iteration inside the loop (point 2), we use widening to generalize observations: octagons at $i = 2$ and $i = 3$ are described below.

$$\begin{aligned} point2(i=2) = \\ 1 \leqslant x \leqslant 2 \ \wedge \ 0 \leqslant y \leqslant 1 \\ 1 \leqslant x+y \leqslant 3 \\ -1 \leqslant -x+y \leqslant -1 \end{aligned} \qquad \begin{aligned} point2(i=3) = \\ 1 \leqslant x \leqslant 3 \ \wedge \ 0 \leqslant y \leqslant 2 \\ 1 \leqslant x+y \leqslant 5 \\ -1 \leqslant -x+y \leqslant -1 \end{aligned}$$

$$\begin{aligned} point2(i=2) \nabla point2(i=3) = \\ 1 \leqslant x \ \wedge \ 0 \leqslant y \\ 1 \leqslant x+y \\ -1 \leqslant -x+y \leqslant -1 \end{aligned} \tag{5}$$

The widening on octagons extrapolates unstable bounds to infinity (equation-5). In this case abstract interpretation finds $a_1, a_2, ..., a_9$ such that the octagon is a sound program invariant.

With linear regression we can automatically determine unstable bounds and therefore design the corresponding widening operation. The constraint sets at i=2 and i=3 can be expressed in the matrix form as:

$$CS_{i=2} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \leqslant \begin{bmatrix} 2 \\ -1 \\ 1 \\ 0 \\ 3 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

$$CS_{i=3} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \leqslant \begin{bmatrix} 3 \\ -1 \\ 2 \\ 0 \\ 5 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

Given the two constraint sets as input to the linear regressor, we obtain the new constraint set as a linear function of the iterator i as defined by equation-4:

$$CS_i = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \leqslant \begin{bmatrix} i \\ -1 \\ i-1 \\ 0 \\ 2i-1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

Next, we drop all rows with i in either matrix to obtains the widening.

$$CS_2 \nabla CS_3 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} \leqslant \begin{bmatrix} -1 \\ 0 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

Clearly the widening obtained above from regression is the same as that obtained by traditional methods in equation-5.

Conclusion

Our key result is the duality between supervised machine learning (Regression) and abstract interpretation (Widening). Indeed both regression and widening are inductive learners- they generalize from observations; However, until now they have been studied separately as distinct fields of study. We hope to change that. Further, we used this relationship between widening and regression to come up with a widening operator parametric on the domain.

Related Works

Even though design of a widening operator is an integral step in the design of an abstract interpreter using an infinite domain, not much work has been done to systematize the design of widening operator. There is currently no known algorithm for automatically deriving a widening operator for any given domain.

Most work on the design of widening operators has been restricted to specific domains- type graphs [11], ellipsoid domain for digital filters [9], etc. However, some work has been done recently to study the widening operator properties to support systematic construction of widening [3, 7]. In [2] authors describe three generic methods to derive widening for powerset domains by lifting the base-level abstract domains. In [3], authors show how widening operators can be combined in the cartesian and reduced product of abstract domains. However, all these works provide for deriving widening of higher level domains by lifting widening of base-level domains. Our work provides for a way to automatically derive widening of base-level numerical domains.

References

[1] Roberto Bagnara, Patricia M Hill, Elisa Ricci, and Enea Zaffanella. 2005. Precise widening operators for convex polyhedra. *Science of Computer Programming* 58, 1-2 (2005), 28–56.

[2] Roberto Bagnara, Patricia M Hill, and Enea Zaffanella. 2004. Widening operators for powerset domains. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 135–148.

[3] Agostino Cortesi. 2008. Widening operators for abstract interpretation. In *2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*. IEEE, 31–40.

[4] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL'77*. ACM, 238–252.

[5] Patrick Cousot and Radhia Cousot. 1976. Static determination of dynamic properties of programs. In *Dunod*.

[6] Patrick Cousot and Radhia Cousot. 1979. Systematic design of program analysis frameworks. In *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM, 269–282.

[7] Patrick Cousot and Radhia Cousot. 1992. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In *International Symposium on Programming Language Implementation and Logic Programming*. Springer, 269–295.

[8] Patrick Cousot and Nicolas Halbwachs. 1978. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM, 84–96.

[9] Jérôme Feret. 2004. Static analysis of digital filters. In *European Symposium on Programming*. Springer, 33–48.

[10] Antoine Miné. 2006. The octagon abstract domain. *Higher-order and symbolic computation* 19, 1 (2006), 31–100.

[11] Pascal Van Hentenryck, Agostino Cortesi, and Baudouin Le Charlier. 1994. Type analysis of Prolog using type graphs. *ACM SIGPLAN Notices* 29, 6 (1994), 337–348.