CHALMERS
UNIVERSITY OF TECHNOLOGY

Fuzzy Set Abstraction

Downloaded from: https://research.chalmers.se, 2026-02-14 22:25 UTC

Citation for the original published paper (version of record):
Lidman, J., Svenningsson, J. (2018). Fuzzy Set Abstraction. Electronic Notes in Theoretical Computer Science, 334: 17-29. http://dx.doi.org/10.1016/j.entcs.2018.03.003

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 334 (2018) 17–29

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Fuzzy Set Abstraction

Jacob Lidman and Josef Svenningsson[1]

Computer science and engineering,
Chalmers University of Technology,
Gothenburg, Sweden

Abstract

Program analysis plays a key part in improving modern software. Static (sound) analyses produce globally correct, but often pessimistic results while dynamic (complete) analyses yield highly precise results but with limited coverage. We present the Fuzzy set abstraction which generalizes previous work based on 3-valued logic. Our abstraction allows for hybrid analysis where static results are refined dynamically through the use of fuzzy control systems.

Keywords: Abstract interpretation, static program analysis, dynamic program analysis

1 Introduction

Static and dynamic analysis are complementary. Static analysis is sound because it summaries all possible executions, whereas dynamic analysis provides more precise information because it summaries the executions which actually happen in practice.

Over-approximation in static analysis is sometimes a sever problem for applications that rely on the results. Static alias analysis often produce point-to sets several times larger than dynamic alias analysis[8],[9] and in extension inhibits several opportunities for parallelization.

Being able to combine both kinds of analyzes can greatly improve results, for instance in non-functional verification (e.g. deducing worst-case benefit of compiler optimizations) when pessimistic assumptions about input state/environment is used. In this case, sound results are interesting at compile-time so that optimizations that are guaranteed to be detrimental is not applied. In contrast, complete results are interesting at run-time where the actual set of inputs are known and hence the benefit of an optimization can be accurately evaluated. The fuzzy dataflow framework[5] showed how program analyzes based on fuzzy logic can uncover

[1] Email: lidman@chalmers.se, josefs@chalmers.se

https://doi.org/10.1016/j.entcs.2018.03.003
1571-0661/© 2018 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

optimization opportunities that classical frameworks would not. The generalization to many-valued fuzzy logics allow program properties to be true or false to a certain degree. The truth values are elements of the unit interval[2] and denote bias of the program property. For instance, a result of 0.1875 would indicate that the property tends to false since it is closer to 0 (false) than to 1 (true).

The increased expressiveness offered by using fuzzy logic in program analysis however motivates additional research into the properties of the analysis framework, in particular soundness and completeness.

We introduce the Fuzzy Set Abstraction that generalize the three-valued logic abstraction[10]. We present the theoretical foundation of the fuzzy set abstraction and prove soundness for the static analysis (Section 3.1). We also present a dynamic analysis (Section 3.2) where we use an adaptive fuzzy inference system from fuzzy control theory to gradually specialize the analysis results to improve accuracy.

2 Preliminaries

We briefly introduce several concepts from the fuzzy set community. Our static analyses manipulate fuzzy sets using predicate transformers, expressed using fuzzy logic (Section 2.1), and collector functions motivated by possibility theory (Section 2.2). Similarly our dynamic analyses start from the results of the static analysis and iteratively specialize it to increase the accuracy of our results. This process relies on a fuzzy classifier (Section 2.3).

2.1 Fuzzy set and logic

Fuzzy sets assign a partial membership to each element as opposed to classical sets where the membership is binary. We use the common point-wise ordering to relate two fuzzy sets: $\langle S, \mu_A \rangle \leq \langle S, \mu_B \rangle \Leftrightarrow \forall s \in S : \mu_A(s) \leq \mu_B(s)$.

Definition 2.1 Let S be a set of elements and μ_S a *membership function* that assigns a membership value from the unit interval $[0, 1]$ to each element. Then $\langle S, \mu_S \rangle$ is a fuzzy set.

A fuzzy set over a singleton set can be considered a description of partial truth. Fuzzy logic defines logical connectives to manipulate such fuzzy sets. Here, complement $\tilde{\neg}$ is often defined as negation (i.e., $1 - x$) and, in the Min-max fuzzy logic, the max operation is used for disjunction $\tilde{\vee}$ and min operator for conjunction $\tilde{\wedge}$.

Definition 2.2 Fuzzy logics $\langle \tilde{\wedge}, \tilde{\vee}, \tilde{\neg} \rangle$ satisfy the De Morgans laws. $\tilde{\wedge}$ and $\tilde{\vee}$ are two binary functions $[0, 1]^2 \to [0, 1]$ that are commutative, associative and monotonically decreasing/increasing and have identity elements ($x \,\tilde{\wedge}\, 1 = x$ and $x \,\tilde{\vee}\, 0 = x$). Similarly $\tilde{\neg}$ is a unary function $\tilde{\neg} : [0, 1] \to [0, 1]$ that is decreasing, involutory (i.e. $\tilde{\neg}(\tilde{\neg}(x)) = x$) and satisfy the boundary conditions $\tilde{\neg}(0) = 1$ and $\tilde{\neg}(1) = 0$[3]. The

[2] To guarantee termination we use a finite congruence set of the unit interval.

[3] In the fuzzy logic literature the conjunction operator is called a Triangular norm (T-norm), the disjunction operator called Triangular conorm (S-norm) and complement the C-norm

logical connectives are lifted point-wise for non-singleton sets of elements.

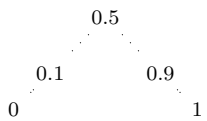
2.2 Possibility theory

Possibility theory is a non-classical theory for reasoning about uncertainty. Whereas probabilities are self-dual, i.e. $P(x) = 1 - P(\overline{x})$, possibilities Π are dually related to necessities N, i.e. $\Pi(x) = 1 - N(\overline{x})$. Underlying these two measures is a possibility distribution π that encode the partial knowledge of a universe Ω: for $x \in 2^{\Omega}$, $\pi(x) = 1$ mean x is totally possible and $\pi(x) = 0$ mean x is rejected as impossible [4].

Definition 2.3 The **possibility and necessity measure** is defined by $\Pi(X) = \sup_{u \in X} \pi(u)$ and $N(X) = \inf_{u \notin X} 1 - \pi(u)$ respectively, where $\pi : 2^{\Omega} \to [0, 1]$ is a *possibility distribution* that given a universe of discourse Ω with measurable subsets, satisfies:

- $\pi(\emptyset) = 0$
- $\pi(\Omega) = 1$
- For any set U of pair-wise disjoint subsets $U_i \subseteq \Omega$: $\pi(\bigcup_i U_i) = \sup_i \pi(U_i)$

Although concepts parallel to probability theory (e.g., independence, conditioning) can be defined for possibility theory [2] we restrict attention to partial orders between distributions. An information order sort distributions based on their informative content by comparing a measure to its negation (i.e., $\Pi(\overline{x}) = 1 - \Pi(x)$ or $N(\overline{x}) = 1 - N(x)$). The measure provides the least amount of information when it is equal to its negation, i.e. $\Pi(x) = \Pi(\overline{x}) = 0.5$. We define a semi-lattice [5] $\langle [0, 1], \preceq_{\sqcup}, \uplus_{\sqcup} \rangle$ over a set of events isomorphic to the unit interval (i.e. $[0, 1] = 2^{\Omega}$) such that $x \preceq_{\sqcup} y$ mean $P(x)$ is more informative than $P(y)$:



$$x \uplus_{\sqcup} y = \begin{cases} \max(x, y) & x, y \leq 0.5 \\ \min(x, y) & x, y \geq 0.5 \\ 0.5 & \text{otherwise} \end{cases}$$

Conceptually the join operation returns the consensus of both inputs, i.e. the least informative of the two or the middle ground (0.5) if they are conflicting (one input is < 0.5 and one i > 0.5). The information order is used in our static analysis in Section 3.1 to merge results from different control paths and formalize a concretization function.

2.3 Takagi-Sugeno Adaptive-Network-based fuzzy inference system (TS-ANFIS)

TS-ANFIS implements classification as inference in a system of fuzzy IF-THEN rules. Each rule is composed of an antecedent and consequence part. The antecedent is composed of a fuzzy set for each input variable of the classifier and the consequence is a polynomial mapping the input variables to the output domain. A two rule example classifier is shown in Figure 1 for two input variables x_0 and x_1 where the

[4] Possibility measures are special cases of plausibility functions in Dempster-Shafer theory [2], a generalization of Bayesian probability theory

[5] Using the duality theorem the inverse order of this join semi-lattice is a meet semi-lattice, where in our case the elements would correspond to measures of $N(\overline{x})$ since $\Pi(x) = 1 - N(\overline{x})$.

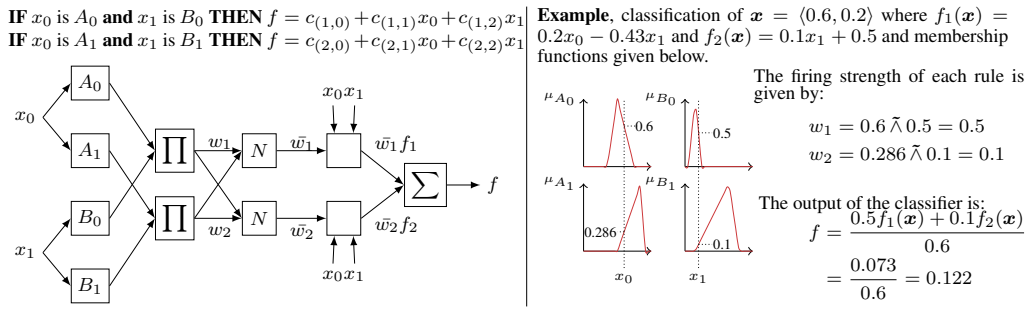


Fig. 1. First-order Takagi-Sugeno ANFIS with two rules and two variables

fuzzy sets in the antecedent part is called A_0 (A_1) and B_0 (B_1) for the first (second) rule respectively and similarly the coefficients of the polynomial of the consequence part is denoted $c_{1,i}$ ($c_{2,i}$). The classifier is composed of five layers. The first three layers look up the fuzzy membership degrees of the input vector and compute the normalized fuzzy conjunction of this collection, producing the normalized *firing strength* of the rule, i.e. the fuzzy membership or weight of the rule. In the example the fuzzy membership of the input vector is $\mu_{A_0}(x_0) = 0.6$ and $\mu_{B_0}(x_1) = 0.5$. The fuzzy conjuncture of the min-max fuzzy-logic evaluates to $w_1 = min(0.5, 0.6) = 0.5$, similarly $w_2 = 0.1$, and hence the normalized weight is $\bar{w}_1 = \frac{w_1}{w_1+w_2} = \frac{0.5}{0.6}$. The next layer weight the consequent output $f_i(x)$ with the normalized firing strength. The final layer sums each weighted rule classification. Returning to the example we get $f_1(\boldsymbol{x}) = 0.034$ and $f_2(\boldsymbol{x}) = 0.56$ hence the output of the classifier is $\frac{0.5}{0.6}0.034 + \frac{0.1}{0.6}0.56 = 0.122$. Since the consequent part is a polynomial and the classification can be improved online using algorithms for statistical regression/adaptive filtering, e.g. Least Mean Square (LMS). We use TS-ANFIS and LMS when we can guarantee convergence for the dynamical analysis in Section 3.2.

3 Fuzzy set abstraction

Program analyses reason about the dynamics of the *individuals* of the analysis (e.g. memory stores, redundant expressions) with respect to a set of *properties*. The 3-valued logic analysis of Sagiv et al.[10] uses logical predicates to state when individuals possess a property, and logical formulas to model how statements update predicates. Their framework relies on first-order 3-valued Kleene logic with transitive closure and bi-lattice theory [4], where inference values are ordered both according to a *information order* and a *truth order*. The framework crucially enables summarizing predicates over a potentially infinite number of individuals and interpretations in a concrete semantics to a finite, tractable program analysis.

Our interest in increasing the precision of analyses and incorporating dynamic information leads us to consider analyses where individuals can possess properties to a certain degree. To this end we use a family of fuzzy logics.

One of the simplest forms of fuzzy logics is Kleene's 3-valued logic as used by Sagiv et al.[10] which we get by restricting the min-max fuzzy logic to three

values. We will therefore use their program analysis framework as a basis for our own fuzzy set abstraction which is described in Section 3.1. Although analyzes in our framework are decidable and sound the resulting abstract description could, in the worst-case, be very large. Therefore we also consider cases where the resulting description is kept to a minimum. Analyses in this approximation yield a single interpretation representing the maximum interpretation.

3.1 Static analysis

Similar to Sagiv et al.[10] we start with a given set of predicates, a *vocabulary*, $\mathcal{P} = \{p_1, ..., p_n\}$. A program statement, which defines the new state for each property of each individual, is modeled as a *predicate transformer*, i.e. a logical formula over $\mathcal{P}$ that transforms a predicate into a new predicate.

Definition 3.1 Let $\mathcal{P}$ be a vocabulary. A **program statement** is a set of predicate transformers, one for each predicate $p \in \mathcal{P}$. A predicate transformer is described by a **logical formula** ϕ from the grammar below. A **Flow graph** is a edge-labeled connected graph $G = \langle V \cup \{v_{start}\}, E, C \rangle$ where v_{start} is the unique start node with zero in-degree, nodes $v \in V$ are program statements, edges $e \in E \subset V \times V$ denote control transfer between two vertices and C maps an edge to its control transfer condition.

$$\langle \phi \rangle ::= \bot \mid \top \mid p(v_1, ..., v_k) \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \forall v : \phi \mid \exists v : \phi \mid TCv_1, v_2 : \phi$$

The concrete semantics defines the set of possible 2-valued logical structures that satisfy a predicate transformer. The semantics of a formula is defined in the standard way for 2-valued logics with transitive closure.

Definition 3.2 [Sagiv et al.[10], def. 3.2] Let U^S be a universe of individuals and $\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2 ... \cup \mathcal{P}^k$ a set of unary, binary,..., k-ary predicates.

- **2-valued interpretation** is a structure $S = \left\langle U^S, i^S \right\rangle$ where $i^S(p^*)$ is a map from $\left(U^S\right)^*$ to a truth value (i.e., 0 or 1), $p^* \in \mathcal{P}^*$. The set of 2-valued interpretations is denoted 2-$STRUCT[\mathcal{P}]$.
- An **assignment** Z^S is a mapping from variables to individuals, i.e. $Z : V \mapsto U^S$. The assignment is **complete** if Z is total.
- The free variables of a formula ϕ is defined in the standard way. If free$(\phi) = \emptyset$ we say ϕ is a **closed formula**.
- **2-valued meaning** $[\![\phi]\!]_2^S(Z)$ of a closed formula ϕ and a complete assignment Z yields a truth value and defined inductively in Figure 2 (left)
- S and Z **satisfy** ϕ if $[\![\phi]\!]_2^S(Z) = 1$. We denote this by $S, Z \models \phi$, or $S \models \phi$ if ϕ is satisfied for all Z.

A statement updates each property (of each individual) according to a predefined transfer function. The semantics of a statement should hence generate new predicates, defined in terms of the predicates of its predecessors.

Definition 3.3 [Sagiv et al.[10], def. 3.3] Let $\mathcal{P} = \mathcal{P}^1 \cup \mathcal{P}^2 \cup ...\mathcal{P}^k$ be a vocabulary

$$
\begin{aligned}
&[\![\bot]\!]_2^S(Z) = 0\\
&[\![\top]\!]_2^S(Z) = 1\\
&[\![\neg\phi]\!]_2^S(Z) = 1 - [\![\phi]\!]_2^S(Z)\\
&[\![p\,(v_1, ..., v_k)\,]\!]_2^S(Z) = i^S(p)\,(Z(v_1), ..., Z(v_k))\\
&[\![\phi_1 \wedge \phi_2]\!]_2^S(Z) = \min\left([\![\phi_1]\!]_2^S(Z), [\![\phi_2]\!]_2^S(Z)\right)\\
&[\![\phi_1 \vee \phi_2]\!]_2^S(Z) = \max\left([\![\phi_1]\!]_2^S(Z), [\![\phi_2]\!]_2^S(Z)\right)\\
&[\![\forall v : \phi]\!]_2^S(Z) = \min_{u \in U^S} [\![\phi]\!]_2^S\,(Z[v \mapsto u])\\
&[\![\exists v : \phi]\!]_2^S(Z) = \max_{u \in U^S} [\![\phi]\!]_2^S\,(Z[v \mapsto u])\\
&[\![TC\ v_1, v_2 : \phi]\!]_2^S(Z) =\\
&\qquad \max_{\substack{u_{[0,n]} \in U\\ Z(v_1) = u_0\\ Z(v_2) = u_n}} \min_{i=0}^{n-1} [\![\phi]\!]_2^S \left(Z\begin{bmatrix} v_1 \mapsto u_i\\ v_2 \mapsto u_{i+1}\end{bmatrix}\right)
\end{aligned}
$$

$$
\begin{aligned}
&[\![\bot]\!]_{[0,1]_q}^S(Z) = 0.0\\
&[\![\top]\!]_{[0,1]_q}^S(Z) = 1.0\\
&[\![\neg\phi]\!]_{[0,1]_q}^S(Z) = \tilde{\neg}\left([\![\phi]\!]_{[0,1]_q}^S(Z)\right)\\
&[\![p\,(v_1, ..., v_k)\,]\!]_{[0,1]_q}^S(Z) = i^S(p)\,(Z(v_1), ..., Z(v_k))\\
&[\![\phi_1 \wedge \phi_2]\!]_{[0,1]_q}^S(Z) = \tilde{\wedge}\left([\![\phi_1]\!]_{[0,1]_q}^S(Z), [\![\phi_2]\!]_{[0,1]_q}^S(Z)\right)\\
&[\![\phi_1 \vee \phi_2]\!]_{[0,1]_q}^S(Z) = \tilde{\vee}\left([\![\phi_1]\!]_{[0,1]_q}^S(Z), [\![\phi_2]\!]_{[0,1]_q}^S(Z)\right)\\
&[\![\forall v : \phi]\!]_{[0,1]_q}^S(Z) = \tilde{\bigwedge}_{u \in U^S} [\![\phi]\!]_{[0,1]_q}^S\,(Z[v \mapsto u])\\
&[\![\exists v : \phi]\!]_{[0,1]_q}^S(Z) = \tilde{\bigvee}_{u \in U^S} [\![\phi]\!]_{[0,1]_q}^S\,(Z[v \mapsto u])\\
&[\![TC\ v_1, v_2 : \phi]\!]_{[0,1]_q}^S(Z) =\\
&\qquad \tilde{\bigvee}_{\substack{u_{[0,n]} \in U\\ Z(v_1) = u_0\\ Z(v_2) = u_n}} \tilde{\bigwedge}_{i=0}^{n-1} [\![\phi]\!]_{[0,1]_q}^S \left(Z\begin{bmatrix} v_1 \mapsto u_i\\ v_2 \mapsto u_{i+1}\end{bmatrix}\right)
\end{aligned}
$$

Fig. 2. Definition of the semantics of classical first-order logic with transitive closure (left) and first-order fuzzy logic with transitive closure over the congruence domain $[0,1]_q$ (right)

of unary, binary,... k-ary predicates, let ϕ_p^w be the formula with free variables $v_1, ..., v_x$ which updates predicate p at statement w. Given a structure S we define the semantics of S after w as

$$[\![st(w)]\!]_2(S) = \left\langle U^S, \bigcup_{x=1}^{k} \lambda p \in \mathcal{P}^x \lambda u_1, ..., \lambda u_x . [\![\phi_p^w]\!]_2^S([v_1 \mapsto u_1, ..., v_x \mapsto u_x])\right\rangle.$$

The 2-valued semantics define the (possibly infinite) set of logical structures that a node may see on entry. The collecting semantics is later defined as the least fixed-point of the 2-valued semantics.

Definition 3.4 Let VS map $v \in V$ to a set of 2-valued logical structures. The 2-valued semantics of a flow graph $G = \langle V, E, C\rangle$ is given by

$$[\![G]\!]_2(VS) = \lambda v. \begin{cases} VS(v_{start}) & v = v_{start}\\ \bigcup_{w \to v \in E} \left\{[\![st(w)]\!]_2(S) | S \in VS(w) \text{ and } S \models C(w, v)\right\} & \text{otherwise}\end{cases}$$

The natural order on $[0,1]$ has infinite height, hence includes infinite chains. Fixpoint iteration in such a domain may not terminate. Widening operators solve this problem by traversing the domain in a finite number of steps, e.g. by a sequence of *jumps* in the domain followed by the top/bottom element. However, its very hard to control the degree of over-approximation introduced by a widening operator. Rather than introducing a widening operator we let our fuzzy abstraction use a finite congruence domain of the unit interval. To trade expressiveness for tractability, or vice versa, we define the abstract semantics over a family of congruence domains over $[0,1]$: the ϵ_q domains, where $q \in \mathbb{N} - \{0\}$. The domains are linearly ordered such that a truth value t in ϵ_q is split to two truth values (not including 0.5) in ϵ_{q+1}. We illustrate the congruence relation in Figure 3 for the ϵ_1 (i.e., 3-valued Kleene logic used by Sagiv et al.[10]), ϵ_2 and ϵ_3. The values in the light gray box of ϵ_2 (i.e. 1/4, 1/2 and 3/4) are therefor mapped to 1/2 in ϵ_1 and similarly the dark gray boxes of ϵ_3 denote sets of values that are mapped to a single value in ϵ_2. The congruence

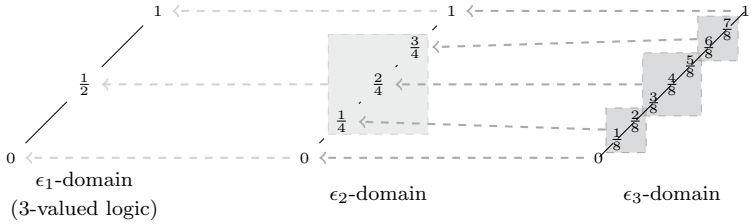


Fig. 3. Equivalence partitioning of the ϵ_1, ϵ_2 and ϵ_3 graded truth orders

relation is consistent with the information order introduced in Section 2, i.e., when interpreting a ϵ_{q+1} value in a ϵ_q domain the value is never more *informative*.

Definition 3.5 The ϵ_q equivalence class of $x \in [0,1]$, denoted $[x]_q$, is defined by rounding x to the closest multiple of $\frac{1}{2^q}$ towards $\frac{1}{2}$. The set of all ϵ_q equivalence classes is $[0,1]_q = \{\frac{1}{2^q}x | x \in \mathbb{N}_{2^q}\}$ where $\mathbb{N}_{2^q} = \left\{x \mid x \in \mathbb{N} \wedge 0 \leq x \leq 2^q\right\}$.

The abstract semantics over ϵ_q is defined in terms of the fuzzy logics from Section 2. The definition uses concepts such as interpretation and assignment which is similar to that of the concrete semantics but where *truth value* is now an element of $[0,1]_q$ rather than $\{0,1\}$.

Definition 3.6 [Sagiv et al.[10], def. 4.2] The $[0,1]_q$**-valued interpretation** and assignment is analogous to the concrete semantics.

- Given a Fuzzy logic $\langle \tilde{\wedge}, \tilde{\vee}, \tilde{\neg} \rangle$ the **fuzzy** ϵ_q **meaning** $[\![\phi]\!]^S_{[0,1]_q}(Z)$ of a closed formula ϕ and a complete assignment Z yields a truth value in $[0,1]_q$ and defined inductively in Figure 2 (right)
- The definition of the semantics of a statement (i.e., $[\![st(w)]\!]_{[0,1]_q}$) is analogous to the concrete case in Definition 3.3 but the formulas are interpreted in a given fuzzy logic.
- S and Z **potentially satisfy** ϕ if $[\![\phi]\!]^S_{[0,1]_q}(Z) > 0$. We denote this by $S, Z \models_{[0,1]_q} \phi$, or $S \models_{[0,1]_q} \phi$ if ϕ is satisfied for all Z.

Embeddings was introduced by Sagiv et al.[10] to relate 2-valued and 3-valued interpretations. Informally they relate logical structures that conform to an information order. The embeddings cluster individuals and decide the value of their properties in terms of the corresponding values of the members of the cluster. Importantly, the class of embeddings that minimize information loss is termed *tight* and are used to define the abstract semantics of a flow-graph. Note that although we choose to cluster individuals here it is also possible to cluster predicates[7] as in a predicate abstraction.

Definition 3.7 Let $S = \left\langle U^S, i^S \right\rangle$ and $S' = \left\langle U^{S'}, i^{S'} \right\rangle$ be two $[0,1]_q$-interpretations. A surjective function $f : U^S \to U^{S'}$ is a $\preceq$**-embedding** (denoted by $S \preceq^f S'$) if it

satisfies below relation. The embedding is **tight** if the relation holds for equality.

$$\forall x \in [1,k], \forall p \in \mathcal{P}^x : i^{S'}(p)(u'_1, ..., u'_x) \succeq \biguplus_{\substack{(u_1,...,u_x) \in (U^S)^x \\ f(u_i) = u'_i, 1 \leq i \leq x}} i^S(p)(u_1, ..., u_x)$$

The abstract semantics need to preserve information loss to be conservative (i.e. should not spuriously introduce more precise results). For this reason our abstract semantics is monotonically increasing with respect to an information order $\preceq$.

Lemma 3.8 (Sagiv et al.[10], Lemma 4.4) *Let $S = \left\langle U^S, i^S \right\rangle$ and $S' = \left\langle U^{S'}, i^{S'} \right\rangle$ be two structures where $S = S'$ and let $f : U^S \to U^{S'}$ be a $\preceq$-embedding. Then for every complete assignment Z, all k-arity $p \in \mathcal{P}^k$: $i^S \preceq i^{S'} \Rightarrow [\![\phi]\!]^S_{[0,1]_q} \preceq [\![\phi]\!]^{S'}_{[0,1]_q}$ where $i^S \preceq i^{S'}$ is the point-wise extension of $\preceq$.*

Intuitively, given a finite number of individuals (predicates) we should be able to cluster (even an infinite number of) predicates (individuals) into a finite number of clusters such that all predicates (individuals) in a cluster are equivalent with respect to the individuals (predicates). We refer to this special embedding as *normalization* [6]. More specifically the number of clusters from the normalizer is bounded by $|U^{S'}| \leq (1 + 2^q)^{\sum_{i=1}^{k} |\mathcal{P}^i|^i}$ in a $[0,1]_q$ domain.

Definition 3.9 Let $\mathcal{P} = \mathcal{P}^1 \cup ... \cup \mathcal{P}^k$ be the sets of unary ($\mathcal{P}^1$), ..., k-ary ($\mathcal{P}^k$) predicates. Two individuals $u_i, u_i \in U^S, u_i \neq u_j$ are **equivalent** if all predicates evaluate to the same values for u_i and u_j, i.e. if $\forall p \in \mathcal{P}^1 : p(u_i) = p(u_j)$ and $\forall p \in \mathcal{P}^2, u' \in U^S : [p(u_i, u') = p(u_j, u')] \wedge [p(u', u_i) = p(u', u_j)]$ and analogously for all $\mathcal{P}^i, 3 \leq i \leq k$. The **normalizer** f_{Norm} is a $\preceq$-embedding that generates a new logical structure $\left\langle \hat{U}^S, \hat{i}^S \right\rangle$ that maps equivalent individuals from U^S to the same individual in $\hat{U}^S$.

Example 3.10 Let $S = \left\langle \{u_1, u_2, u_3, u_4, u_5, u_6\}, i^S \right\rangle$ be as in left most table below. Here u_2, u_4 and u_6 are equivalent w.r.t all predicates (i.e., p, q and r). Shown in the right most table below is the resulting structure when applying the normalizer, where equivalent individuals are represented by a new individual u_{246}.

Individual	Unary predicate		
	p	**q**	**r**
u_1	1	0	1
u_2	1	1	0
u_3	0	1	0
u_4	1	1	0
u_5	1	1	1
u_6	1	1	0

Individual	Unary predicate		
	p	**q**	**r**
u_1	1	0	1
u_{246}	1	1	0
u_3	0	1	0
u_5	1	1	1

Using normalization we can hence produce a logical structure with a finite number of individuals, and by extension, bound the number of logical structures. We

[6] Sagiv et al.[10] refer to this as the *canonical abstraction*

also consider the extreme case of normalization, when the set of logical structures is a singleton. To accommodate both cases we parameterize the abstract semantics on a function F^n, the *collector function* that takes n sets (representing the results from n incoming edges) of logical structures and produce a single set of logical structures. We consider the following F_* and $\preceq_*$ combinations:

(i) $(n \geq 1)$ The *bounded union* semantics with $\preceq_\cup$ and $F^n_\cup(\boldsymbol{S}) = \bigcup_{i=1}^{n} S_i$.

(ii) $(n = 1)$ The *maximum* semantics with $\preceq_\sqcup$ and
$F^n_\sqcup(\boldsymbol{S}) = \left\langle U^{\hat{S}}, \bigcup_{x=1}^{k} \{\lambda u_1, ..., \lambda u_x. \left(\biguplus_\sqcup\right)_{y=1}^{n} i^{S_y}(p)\left([v_1 \mapsto u_1, ..., v_x \mapsto u_x]\right) | p \in \mathcal{P}^x\}\right\rangle$.

We introduce an abstract semantics as the fixed-point, with respect to set-inclusion (when n is bounded) or the information order (when $n = 1$) from Section 2.2, of the equation system below and show that it over-approximates the collecting semantics.

Definition 3.11 Let F^n_* be a collector function and f be a tight $\preceq_*$-embedding and VS map $v \in V$ to a set of bounded $[0,1]_q$-valued logical structures. The $[0,1]_q$-**valued semantics** of a flow graph $G = \langle V, E, C\rangle$ is given by

$$[\![G]\!]_{[0,1]_q}(VS) = \lambda v. \begin{cases} F^1_*\left(VS(v_{start})\right) & v = v_{start} \\ F^n_*\left(\left\{ f\left([\![st(w)]\!]_{[0,1]_q}(S)\right) \middle| \begin{array}{l} S \in VS(w) \text{ and} \\ S \models_{[0,1]_q} C(w,v) \end{array} \right\}_{w \to v \in E}\right) & \text{otherwise} \end{cases}$$

Computing the fixed-point proceeds by iteratively applying a monotonic increasing transfer function and terminates in bounded time because the set of possible logical structures is bounded. The result of the analysis is sound as per Theorem 3.12. Note that the result of a fuzzy set abstraction using $F^n_\sqcup$ is the most accurate maximum logical structure a ϵ_q domain can provide, i.e. within a factor $\frac{1}{2^q}$ of the true value.

Theorem 3.12 *For a flow graph $G = \langle V, E, C\rangle$ and initial assumptions $\boldsymbol{S_{entry}} = \{S_0, ..., S_n\} \subset [0,1]_q$-$STRUCT[\mathcal{P}], n \in \mathbb{N}$ the collecting semantics of the flow graph is denoted $\mathbb{C}$ and is defined using the 2-valued semantics:*

$$\mathbb{C} = lfp_{\subseteq}[\![G]\!]_2(\{v_{start} \to S_{entry}\} \cup \{v \to \emptyset | v \in V\text{-}\{v_{start}\}\})$$

The abstract semantics with the bounded union- and the maximum collector functions is similarly defined using $[0,1]_q$-valued semantics:

$$\mathbb{A}_{F_\cup} = lfp^{F_\cup}_{\subseteq}[\![G]\!]_{[0,1]_q}(\{v_{start} \to S_{entry}\} \cup \{v \to \emptyset | v \in V\text{-}\{v_{start}\}\})$$
$$\mathbb{A}_{F_\sqcup} = lfp^{F_\sqcup}_{\preceq_\sqcup}[\![G]\!]_{[0,1]_q}(\{v_{start} \to F^n_\sqcup(S_{entry})\} \cup \{v \to \boldsymbol{\frac{1}{2}} | v \in V\text{-}\{v_{start}\}\})$$
$$\text{where } \boldsymbol{\frac{1}{2}} = \left\langle U^S, \bigcup_{x=1}^{k} \bigcup_{p \in \mathcal{P}^x} \{\lambda u_1, ..., u_x. i^S(p)\left([v_1 \mapsto u_1, ..., v_x \mapsto u_x]\right) = 0.5\}\right\rangle$$

The corresponding concretization functions are given by $\gamma_{F_\cup}$ and $\gamma_{F_\sqcup}$ where $[S]_1$ denotes the interpretation of structure S in 3-valued logic, i.e. the point-wise extension

of Definition 3.5.

$$\gamma_{F_\sqcup}(S) = \{S^\# \in 2\text{-}STRUCT[\mathcal{P}] | S^\# \preceq_\sqcup [S]_1\}$$
$$\gamma_{F_\sqcup}(\langle U, i\rangle) = \Big\{\Big\langle U, i^\#\Big\rangle \in [0,1]_q\text{-}STRUCT[\mathcal{P}] \mid \forall x \in [1,k], p \in \mathcal{P}^x, u_1, ..., u_x \in U :$$
$$\Big|i(p)\left([v_1 \mapsto u_1, ..., v_x \mapsto u_x]\right) - i^\#(p)\left([v_1 \mapsto u_1, ..., v_x \mapsto u_x]\right)\Big| \leq \frac{1}{2^q}\Big\}$$

With the above definitions in place we can state or main theorem: The concrete semantics is approximated by the composition of the abstract semantics and the concretization function:

$$\forall v : \begin{cases} \mathbb{C}(v) \subseteq \bigcup_{s \in \mathbb{A}_{F_\sqcup}(v)} \gamma_{F_\sqcup}(s) \\ \mathbb{C}(v) \subseteq \gamma_{F_\sqcup}\left(\mathbb{A}_{F_\sqcup}(v)\right) \end{cases}$$

3.2 Dynamical analysis

We next explore an dynamical analysis which starts from the results of the static analysis with an ϵ_q domain and improves the accuracy by specializing the analysis result given a sequence of dynamical assignments of a property. The aim is to improve completeness, possibly sacrificing soundness, and compute a new fuzzy set with increased classification accuracy. The resulting fuzzy set is described implicitly by an TS-ANFIS. We consider improving the accuracy with respect to the least square error between the example(s) and the analysis results.

Our dynamical analysis instantiate an TS-ANFIS and use gradient descent (GD) to improve classification online. GD is similar to Newton's method and so we use results from Newtonian program analysis[3] in formalizing our dynamic analysis.

Esparza et al. [3] showed that polynomials $f(\boldsymbol{x}) = \sum_i c_i \phi_i(\boldsymbol{x}) = \sum_i c_i \prod_j x_{i,j}^{a_{i,j}}$ over ω-continuous (commutative) semirings admit fixed-points due to Kleene's fixed-point theorem. Since membership of a fuzzy set is an element of the unit interval $[0,1]$ we use the special case of polynomials over the real semiring $\langle \mathbb{R} \cup \{\infty\}, +, \cdot, 0, 1\rangle$. We let $\frac{\partial f}{\partial x_i}$ denote the (partial) derivative of f with respect to the variable x_i, $\nabla f = \sum_i \frac{\partial f}{\partial x_i} e_i$ denote its gradient and $Df\big|_\nu(\boldsymbol{x}) = \nabla f(\nu) \cdot \boldsymbol{x}$ the differential of a polynomial f. We stress that our polynomials are evaluated on a congruence domain of the real semiring, i.e. an ϵ_q domain. In this article we consider the case when the resulting set of logical structures from the static analysis is a singleton, i.e., the result was computed using the maximum abstract semantics above [7].
The logical structure, i.e. $S = \langle U, i\rangle$, can be considered a set of classifiers, one for each k-ary predicate $p \in i$, where the variables are assigned an individual (from U) as input value. Given p we create a two rule TS-ANFIS where the variables $x_{[1,k]}$ are real-valued encodings of the individuals, e.g. the encoding $u_i \mapsto i$.

[7] It is possible to extend our scheme to an arbitrary $n \in \mathbb{N}$ by building an TS-ANFIS with 2^n rules. But this obviously comes at a high cost and for lack of a better scheme we postpone formalization to future work

Newton's method	$f^+_{Newton}(\boldsymbol{\nu}^i)=lfp\Big(Df\Big\vert_{\boldsymbol{\nu}^i}(f(\boldsymbol{\nu}^i)-\boldsymbol{\nu})\Big)$	$f^-_{Newton}(\boldsymbol{\nu}^i)=gfp\Big(Df\Big\vert_{\boldsymbol{\nu}^i}(\boldsymbol{\nu}-f(\boldsymbol{\nu}^i))\Big)$
GD on $\langle \boldsymbol{x}, y\rangle$	$f^+_{GD}(\boldsymbol{\nu}^i)=\mu\cdot lfp\Big(\nabla\big(\boldsymbol{\nu}^i\phi_i(\boldsymbol{x})-y\big)^2\Big)$	$f^-_{GD}(\boldsymbol{\nu}^i)=\mu\cdot gfp\Big(\nabla\big(y-\boldsymbol{\nu}^i\phi_i(\boldsymbol{x})\big)^2\Big)$

Table 1
Newton/GD fixed-point equation

- (Rule 1) p is the fuzzy set in the premise part and $f(x_1, ..., x_k) = 1$ is the consequent polynomial.
- (Rule 2) $\tilde{\neg} p$ is the fuzzy set in the premise part and $f(x_1, ..., x_k) = 0$ is the consequent polynomial.

The output of the initial TS-ANFIS is hence equal to the predicate p. Our dynamical analysis is iteratively given an example assignment $\boldsymbol{x}$ and the corresponding *actual* fuzzy membership y. It computes a new fixed-point when $\boldsymbol{x}$ is the input producing the *expected* fuzzy membership and updates the consequent polynomials $f(\boldsymbol{x})$ based on the error. The algorithm used for rule 1 minimizes the polynomial coefficient corresponding to the constant (which were initialized to 1) and maximizes the remaining coefficients (which was initialized to 0). The polynomial for rule 2 remains constant. Table 1 shows the fixed-point equations. Note that elements of a semiring do not in general have an additive inverse, hence subtraction may seem erroneous. But as noted by Esparza et al. [3] for the particular case when the function is a polynomial in a semi-ring the difference is always positive, a consequence of a generalized form of Taylors theorem. For a commutative semiring the accuracy increases exponentially with each iteration[6]. The number of elements in a ϵ_q domain is 2^q. Hence each iteration of our dynamical analysis produce the fixed-point in the ϵ_{q+1} domain.

Lemma 3.13 shows that the fixed-point systems admit a solution. As all ascending/descending chains are finite in an ϵ_q domain the sequence will, assuming $\boldsymbol{x}$ is monotonically decreasing/increasing, converge to a new consequent polynomial.

Lemma 3.13 *Let $f(\boldsymbol{x})$ be a polynomial. Then the ascending/descending sequence $\left(\nu^i\right)_{i\in\mathbb{N}}$ is increasing/decreasing monotonically and converges to unique least/greatest fixed-point:*

- $0 \leq \nu^i \leq f^+(\nu^i) \leq \nu^{i+1} \leq \sup^{\leq}_{j\in\mathbb{N}} \nu^j$
- $\inf^{\leq}_{j\in\mathbb{N}} \nu^j \leq \nu^{i+1} \leq f^-(\nu^i) \leq \nu^i \leq 1$

The fixed-point equations admit a recursive closed-form that we state in Lemma 3.14. Our results show that we can guarantee validity in the real semiring by imposing constraints on the inputs rather than resorting to a non-negative version of the GD algorithm [1].

Lemma 3.14 *Given a $\langle \boldsymbol{x}, y\rangle$ and a polynomial $f(\boldsymbol{x})$ let $e(k) = f(\boldsymbol{x}) - y$ for f^+ and $e(k) = y - f(\boldsymbol{x})$ for f^-. We can compute f^* using $c^{k+1}_{i,j} = c^k_{i,j} + 2\mu e_i(k)\phi(\boldsymbol{x})$.*

Finally, we show in Theorem 3.15 that our dynamical analysis indeed improve the analysis results for a set of examples.

Theorem 3.15 *For a flow graph $G = \langle V, E, C\rangle$ let $\mathbb{A} : V \mapsto [0, 1]_q$-$STRUCT[\mathcal{P}]$*

be the analysis result with the $F^n_{\sqcup}$ *abstract semantics and* $\langle \boldsymbol{x}, y\rangle_{1\leq i\leq N}$ *a sequence of examples such that* $\forall j : \left(x_j\right)_{i+1} \leq \left(x_j\right)_i$ *and* $y_{i+1} \leq y_i$*. Then each iteration* i *of the GD algorithm, applied to the example* $\langle \boldsymbol{x}, y\rangle_i$*, updates the polynomials such that the coefficients are non-negative and in the limit converge to a polynomial that minimize the least square error to the set of examples.*

4 Conclusion

We have introduced the Fuzzy Set Abstraction which enables stating properties about programs which are true or false to a certain degree. This opens up new possibilities for speculative optimizations which can now use information about which values, branches, etc. are more likely than others.

We have also shown how to perform hybrid analysis by refining the result of a static analysis online. This has been done using TS-ANFIS, an adaptive fuzzy inference system from control theory. This result paves the way for importing other results from the rich literature of fuzzy control theory.

References

[1] Chen, J., C. Richard, J. C. M. Bermudez and P. Honeine, *Nonnegative least-mean-square algorithm*, IEEE Transactions on Signal Processing **59** (2011), pp. 5225–5235.

[2] Dubois, D., H. Prade and H. Prade, "Fundamentals of Fuzzy Sets," The Handbooks of Fuzzy Sets, Springer US, 2000.

[3] Esparza, J., S. Kiefer and M. Luttenberger, *Newtonian program analysis*, J. ACM **57** (2010), pp. 33:1–33:47.

[4] Ginsberg, M., *Multivalued logics: A uniform approach to inference in artificial intelligence*, Computational Intelligence **4** (1988), pp. 265–316.

[5] Lidman, J. and J. Svenningsson, *Bridging static and dynamic program analysis using fuzzy logic*, Quantitative Aspects of Programming Languages and Systems (QAPL) (2017).

[6] Luttenberger, M. and M. Schlund, "Convergence of Newton's Method over Commutative Semirings," Springer Berlin Heidelberg, Berlin, Heidelberg, 2013 pp. 407–418.

[7] Manevich, R., E. Yahav, G. Ramalingam and M. Sagiv, "Predicate Abstraction and Canonical Abstraction for Singly-Linked Lists," Springer Berlin Heidelberg, Berlin, Heidelberg, 2005 pp. 181–198.

[8] Mock, M., M. Das, C. Chambers and S. J. Eggers, *Dynamic points-to sets: A comparison with static analyses and potential applications in program understanding and optimization*, in: *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, PASTE '01 (2001), pp. 66–72.

[9] Ribeiro, C. and M. Cintra, *Quantifying uncertainty in points-to relations*, in: *Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science **4382**, Springer Berlin Heidelberg, 2007 pp. 190–204.

[10] Sagiv, M., T. Reps and R. Wilhelm, *Parametric shape analysis via 3-valued logic*, ACM Trans. Program. Lang. Syst. **24** (2002), pp. 217–298.

A Omitted proofs

Proof. of Theorem 3.12. We first consider the case when using the $F_{\cup}$ function, Note that $\gamma(S)$ creates the set of all 2-valued logical structures that are $\preceq_{\cup} S$,

quantized to the ϵ_1 domain (i.e., the 3-valued domain in Sagiv et al.[10] by Definition 3.5). Since generating S (i.e., the fixed-point iteration) was done conservatively by Theorem 3.8 we can re-use the argument from Sagiv et al.[10] Theorem 4.9 (i.e., that an embedding is extensive w.r.t. to the information order: $[\![\phi]\!]_2^S(Z) \preceq [\![\phi]\!]_3^{S'}(f \circ Z)$), Theorem 6.1 (i.e., that we are conservative w.r.t. a branch condition and program statement) and Theorem 6.2 (i.e., $\forall v : C(v) \subseteq \bigcup_{s \in A(v)} \gamma(s)$).

We next consider the case when using the $F_{\sqcup}$ function. Note that, as per Knaster-Tarskis fixed-point theorem the abstract semantics is well-defined. We argue that the result is a conservative approximation (i.e., $\forall v : \mathbb{C}(v) \subseteq \gamma_{F_{\sqcup}}\left(\mathbb{A}_{F_{\sqcup}}(v)\right)$) based on proof by contradiction on the existence of a fixed-point for the collectors $F_{\sqcup}^n$: Assume an fixed-point S_∞ has been reached then $S_\infty(v), v \in V$ is the unique maximal element in ϵ_q. $\gamma_{F_{\sqcup}}(S_\infty(v))$ is the set of $[0,1]_q$ logical structures where the predicates are within $\frac{1}{2^q}$ from $S_\infty(v)$. If S_∞ is not a conservative approximation then there exists an element $S_* \notin \gamma_{F_{\sqcup}}(S_\infty(v))$ that is further away from $\gamma_{F_{\sqcup}}(S_\infty(v))$ than $\frac{1}{2^q}$ which contradict that S_∞ is the unique fixed-point. □

Proof. of 3.13 This is a special case of Theorem 3.10 in Esparza et al. [3] with the (non-negative) real semiring. □

Proof. of 3.14. As the fixed-point iteration is performed over a totally ordered set (the least square error) the minimal/maximal and infimum/supremum element coincide. We consider finding the coefficients c_i of a polynomial $f(\boldsymbol{x}) = \sum_i c_i \phi_i(\boldsymbol{x}) = \sum_i c_i \prod_j x_{i,j}^{a_{i,j}}$ that minimize the least square error, $e(\boldsymbol{x})^2 = \left(f(\boldsymbol{x}) - y\right)^2$ at a particular $\langle \boldsymbol{x}, y \rangle$. Since $e(\boldsymbol{x})^2$ is convex we know that globally minimal solution occur where gradient is zero:

$$\begin{aligned}
\frac{\partial \left(f(\boldsymbol{x}) - y\right)^2}{\partial \boldsymbol{c}} &= 2\left(f(\boldsymbol{x}) - y\right) \cdot \frac{\partial \sum_i c_i \phi_i(\boldsymbol{x}) - y}{\partial \boldsymbol{c}} \\
&= 2\left(f(\boldsymbol{x}) - y\right)\phi(\boldsymbol{x}) \\
&= 2e(\boldsymbol{x})\phi(\boldsymbol{x})
\end{aligned}$$

Hence $max_{c_i}\left(f(\boldsymbol{x}) - y\right)^2 = 2e(\boldsymbol{x})\phi(\boldsymbol{x})$ and since the expression only depends on variables in the current iteration the sequence of updates is given by $\boldsymbol{c_{i+1}} = \boldsymbol{c_i} + 2\mu e(\boldsymbol{x})\phi(\boldsymbol{x})$. Similary $min_{c_i}\left(y - f(\boldsymbol{x})\right)^2 = -2e(\boldsymbol{x})\phi(\boldsymbol{x})$ and hence $\boldsymbol{c_{i+1}} = \boldsymbol{c_i} - \left(-2e(\boldsymbol{x})\phi(\boldsymbol{x})\right) = \boldsymbol{c_i} + 2e(\boldsymbol{x})\phi(\boldsymbol{x})$. □

Proof. of 3.15. Lemma 3.14 shows that each iteration finds the coefficients that minimize the least square error. Toghether with monotonicity of $\boldsymbol{x}$ & y and Lemma 3.13 the sequence is guaranteed to converge eventually. □