Automated Checking of Implicit Assumptions on Textual Data

Radwa Sherif Abdelbar
Supervisors: Dr. Caterina Urban, Alexandra Bugariu

July 19, 2018

Abstract

1 Introduction

2 Static Analysis

In this section, we aim to present a generic framework for tracking assumptions on input data using backward static analysis, which builds on the work done in [1]. We present a top-level domain, the Assumption Domain, parametrized by a list of abstract domains which we call sub-domains and an internal stack. The different operators and transformations of the analysis are invoked on these sub-domains independently, as well as on the stack. When a program point is encountered at which input is being read, the relevant sub-domain is queried for information about the variable being read as input, and this information is stored on the stack.

Below we present our domain using a top-down approach. In Section 2.1, we present our top level domain and explain its operations. Then in Sections 2.2 we present the properties of the sub-domains and some additional functions which need to be introduced to abstract domains in order for them to work properly with our analysis. In Section 2.3, we then present the Input Stack data structure and explain how it interacts with the sub-domains through the Assumption Domain. Finally, in Section 2.4, we present possible example sub-domains, particularly the ones we have focused on in this project and used in our implementation. We also trace a code example as a demonstration of how our analysis operates.

2.1 The Assumption Domain

$D \equiv SUBD^n \times STACK$

Where $SUBD$ is the set of abstract domains that can keep track of some property about the program variables, and $STACK$ is a set of stacks of assumptions on input values of the program. This domain acts as a top-level domain under which any sequence of chosen abstract domains can operate as independent sub-domains. The stack is then used to record the values of program variables once they are read as inputs to the program. We shall discuss $SUBD$ and $STACK$ in more detail later in this thesis.

We define the Assumption Domain formally as follows:

- An element $d \in D = \{((S_i)_{i=1}^n, Q) | S_i \in SUBD \wedge Q \in STACK\}$.
- A concretization function $\gamma_D(d) = (\bigcup_{i=1}^{n} \gamma_{S_i}(S_i), \gamma_{STACK}(Q))$ [Comment: I am not sure if the best way to make a distinction between program variables and stack is a tuple.]
- A partial order $\sqsubseteq_D$ such that if S_{1i} and S_{2i} belong to the same abstract domain $SUBD_i$, we get that $((S_{1i})_{i=1}^n, Q_1) \sqsubseteq_D ((S_{2i})_{i=1}^n, Q_2) \iff \bigwedge_{i=1}^{n} (S_{1i} \sqsubseteq_{SUBD_i} S_{2i}) \wedge Q_1 \sqsubseteq_{STACK} Q_2$
- A minimum element $\bot_D = ((\bot_{SUBD_i})_{i=1}^n, \bot_{STACK})$.
- A maximum element $\top_D = ((\top_{SUBD_i})_{i=1}^n, \top_{STACK})$
- A join operator $\sqcup_D$ such that $((S_{1i})_{i=1}^n, Q_1) \sqcup_D ((S_{2i})_{i=1}^n, Q_2) = (((S_{1i} \sqcup_{SUBDi} S_{2i})_{i=1}^n, Q_1 \sqcup_{STACK} Q_2))$
- A meet operator $\sqcap_D$ such that $((S_{1i})_{i=1}^n, Q_1) \sqcap_D ((S_{2i})_{i=1}^n, Q_2) = (((S_{1i} \sqcap_{SUBDi} S_{2i})_{i=1}^n, Q_1 \sqcap_{STACK} Q_2))$
- A backward assignment operator $\overleftarrow{[\![X := expr]\!]}((S_i)_{i=1}^n, Q) = ((\overleftarrow{[\![X := expr]\!]}(S_i))_{i=1}^n, \overleftarrow{[\![X := expr]\!]}(Q))$
- A widening operator ∇_D such that $((S_{1i})_{i=1}^n, Q_1) \nabla_D ((S_{2i})_{i=1}^n, Q_2) = ((S_{1i} \nabla_{SUBD_i} S_{2i})_{i=1}^n, Q_1 \nabla_{STACK}$

2.2 The Sub-domains

2.3 The Input Assumption Stack

2.4 Example Sub-domains

[Examples of sub-domains plugged into our analysis.]

3 Implementation

4 Evaluation

References

[1] Madelin Schumacher. Automated generation of data quality checks. Master's thesis, ETH Zurich, 2018.