A Similarity Criterion For Sequential Programs Using Partial Functions - Basic Formulation

Abhinav Aggarwal and Padam Kumar

Indian Institute of Technology Roorkee

Basic Idea

Sequential programs perform transformations on their inputs similar to composite **partial functionals**. These functionals are formed by treating individual program statements as partial functions and then composing them in their order of appearance in the original program. The representation is used to establish the following:

- **Truth preservation of Boolean conditions** under the transformation performed by the program
- **Similarity** in the nature of truth preservation by two programs
- Existence of **sub-structure isomorphism** between two programs based on the truth preservation properties
- A measure, weaker than Turing-reducibility, to **compare computational complexity** of two given programs

Introduction

Let A, B and C be three sets, such that $f : A \to B$ and $g : B \to C$ are defined only on elements satisfying conditions C_1 in A and C_2 in B, respectively. The diagram below commutes only if there exists an $x \in A$ that satisfies C_1 and $f(x) \in B$ that satisfies C_2. This property is called **Truth Preservation** of C_1 by C_2 and is important to study the nature of inputs two seemingly different programs operate on as well as the kind of transformation they simulate.

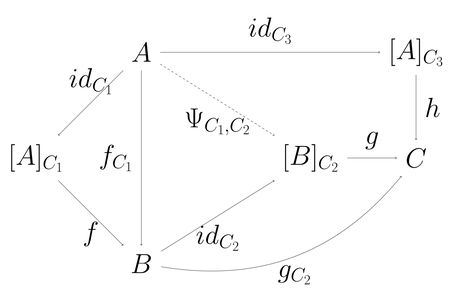

Figure 1: Commutative diagram to show the existence of truth preserving partial function, Ψ_{C_1,C_2}

Order of truth preservation

Given a Boolean condition C, a function $f : A \to A$ has an **order** of truth preservation, defined as :

$$m = \max\left(-1, \min_{x \in [A]_{C_A}} n_x\right)$$

This is useful to study the behavior of f in its iterative application over an input set, as may be the case with recursive calls and `while`-loops.

```
Initialize x ∈ A
while C_A is True on x do
   Set x ← f(x)
end while
Print x
```

Figure 2: A general `while`-loop. For a given value of x, the loop executes exactly m times, since the truth of C_A is rendered false by $f^{(m+1)}(x)$.

Properties of truth-preservation order

Following are some conditions under which truth preservation order may change :

1. The theoretical value of order may be different from its actual value on a t-bit machine.
2. **Condition strengthening** $C'_A \implies C_A$: Order may reduce
3. **Condition weakening** $C_A \implies C'_A$: Order may increase
4. **Nesting of loops**
5. **Smaller/Larger state variable set**
6. **Different state variable set with smaller/larger cardinality**
7. **Conditional functions**

Based on these properties of the truth preservation, conditions for program isomorphism are obtained.

Interpreting Maps between partial functions

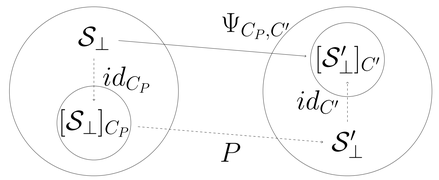

Figure 4: Viewing a program P as a truth preserving function

1. Every sequential computer program, P, is defined on only some inputs which lead the machine under consideration to halt in finite time. The set of all inputs is

$$\mathcal{S}_\perp = \{\perp\} \cup \{x_1 : T_{x_1}, x_2 : T_{x_2}, \dots, \}$$

2. S_P, of $\mathcal{S}_\perp$ - the set of all valid inputs for P (i.e. for which P will make our machine halt in finite time) - selection of this subset is through specifying a condition, $C_P \in \mathcal{C}_{\mathcal{S}_\perp}$ such that $C_P = [x \in S_P]$.
3. The set of output state variables is $\mathcal{S}'_\perp$. Not all values in this set may be achievable by P - only a subset, say $[\mathcal{S}_\perp]_{C'}$, is achievable - select this subset through the application of $id_{C'}$ on $\mathcal{S}'_\perp$.
4. This way, P acts as a truth preserving function from $\mathcal{S}_\perp$ to $\mathcal{S}'_\perp$, with respect to the conditions C_P and C', respectively.

Example :

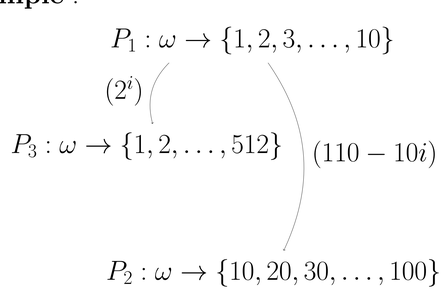

Figure 5: Maps between three programs, P_1, P_2 and P_3

Important Result

Undecidability result : The problem of determining truth preservation order or the existence of type-0/type-1 arrows is undecidable. This can be verified by reducing the Halting problem into this problem.

The undefined state : $\perp$

How to explain the behavior pf a partial function on the remaining elements? If $\mathcal{S}$ denotes the domain of f, called the *State variable set*, lift this domain as

$$\mathcal{S}_\perp = \mathcal{S} \cup \{\perp\}$$

Here, $\perp$ = assignment of the state variables interpreted as an *undefined* state, since we do not know how the state variables, or any function, would behave on reaching here - **Acts as a trap** in the machine : entering here means no coming out.

$$id_C(x) = \begin{cases} x & \text{if } x \in [A]_C \\ \perp & \text{otherwise} \end{cases}$$

Infinite loops

Functions with **infinite truth preservation order** - studied using **transfinite induction** over the iteration space.

```
while C(x) is True do
   Set x ← f(x)
end while
Print x
```

Figure 3: Code to demonstrate infinite order of truth preservation of C by f

Cases when this can happen - Fixed points, periodic points, countably infinite domain and range