A Similarity Criterion For Sequential Programs Using Partial Functions - Transformational and Sub-structure transformational Isomorphism

Abhinav Aggarwal and Padam Kumar

Indian Institute of Technology Roorkee

Basic Idea

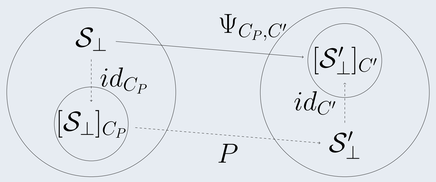

Figure 1: Program P as a truth preserving function

Treating id_{C_P} and $id_{C'}$ as programs P_1 and P_2, through $\Psi_{C_P,C'} = (id_{C'} \circ (P \circ id_{C_P}))$, we go from the inputs of $P_1 : \mathcal{S}_\perp \to [\mathcal{S}_\perp]_{C_P}$ to the output of $P_2 : \mathcal{S}'_\perp \to [\mathcal{S}'_\perp]_{C'}$ through an intermediate transformation of state variables, performed by P. In a way, there is a way to **mimic the operations performed by P_1 through the operations performed by P_2** and the existence of $\Psi_{C_P,C'}$ proves this point for us. The situation is similar to **sub-program isomorphism**, in which P_2 is **semantically isomorphic, i.e. performs the same transformation of its input variables**, to P_1. The computation performed in P_1 is in some sense, similar to that performed in P_2 and once again, $\Psi_{C_P,C'}$ captures this similarity.

Model of Computation

- Turing machine model with a work tape and a print tape
- The state variable set is totally ordered, i.e. there exists a relation $\prec_{\mathcal{S}_{i,\perp_i}}$ such that for all $x_i, y_i \in \mathcal{S}_{i,\perp_i}$, either $x_i \prec_{\mathcal{S}_{i,\perp_i}} y_i$ or $y_i \prec_{\mathcal{S}_{i,\perp_i}} x_i$ or $x_i = y_i$.
- The program P_i induces a total order on its inputs, say a *Program order*, denoted by $\prec_{P_i}$, such that for all $x_i, y_i \in \mathcal{S}_{i,\perp_i}$, we have $x_i \prec_{S_{i,\perp_i}} y_i$ implies $P_i(x_i) \preceq_{P_i} P_i(y_i)$, i.e. the program P_i, when seen as a mapping $P_i : (\mathcal{S}_{i,\perp_i}, \preceq_{\mathcal{S}_{i,\perp_i}}) \to (\mathcal{S}_{i,\perp_i}, \preceq_{P_i})$, is order preserving (may not be strictly monotonic)

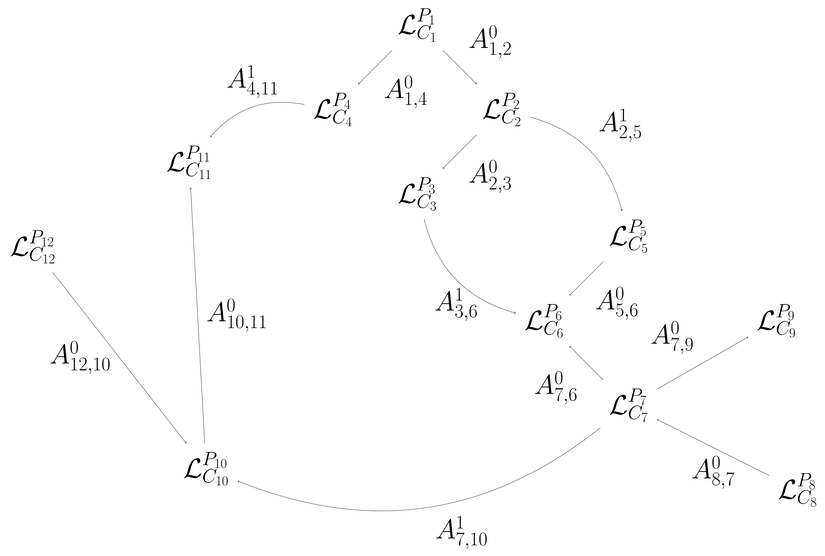

Figure 2: Reduction of type-0 arrows in a commuting diagram

Type-0 arrow - Transformational Isomorphism

Main idea - Representability of every program inside an appropriately designed `while`-loop, using some auxillary variables, if necessary, and then studying the termination properties of that program with respect to the termination of the loop that envelopes it.

$$\mathcal{L}^f_C(x) = \begin{cases} \mathcal{L}^f_C(f(x)) & \text{if } C(x) \text{ is } True \\ x & \text{otherwise} \end{cases}$$

Let $C_i \in \mathcal{C}_{\mathcal{S}_{i,\perp_i}}$ and $C_j \in \mathcal{C}_{\mathcal{S}_{j,\perp_j}}$ be two Boolean conditions. For two given programs $P_i : \mathcal{S}_{i,\perp_i} \to \mathcal{S}_{i,\perp_i}$ and $P_j : \mathcal{S}_{j,\perp_j} \to \mathcal{S}_{j,\perp_j}$, let an injective (one-one) order-preserving transformation function between their ordered domains be $T_{i,j} : \left(\mathcal{S}_{i,\perp_i}, \preceq_{S_{i,\perp_i}}\right) \to \left(\mathcal{S}_{j,\perp_j}, \preceq_{S_{j,\perp_j}}\right)$. Then there exists an **arrow of type-0**, $A^0_{i,j} \in \mathcal{A}^0_{i,j}$, from $\mathcal{L}^{P_i}_{C_i}$ to $\mathcal{L}^{P_j}_{C_j}$, denoted by $A^0_{i,j} : \langle T_{i,j} \rangle$, if the truth preservation orders of all elements in the domains of P_i and P_j are the same, with respect to C_i and C_j, respectively.

$$\Theta_{C_i,P_i}(x) = \Theta_{C_j,P_j}(T_{i,j}(x)) \quad \forall x \in S_{i,\perp_i}$$

Here, $\mathcal{A}^0_{i,j}$ is the set of all arrows of type-0 that can exist between $\mathcal{L}^{P_i}_{C_i}$ and $\mathcal{L}^{P_j}_{C_j}$, one for each order-preserving injection, $T_{i,j}$. The existence of $A^0_{i,j}$ establishes a **Transformational Isomorphism** between $\mathcal{L}^{P_i}_{C_i}$ and $\mathcal{L}^{P_j}_{C_j}$, denoted by $\mathcal{L}^{P_i}_{C_i} \overset{0}{\sim} \mathcal{L}^{P_j}_{C_j}$. This type of isomorphism is transitive in nature :

$$\left(\mathcal{L}^{P_i}_{C_i} \overset{0}{\sim} \mathcal{L}^{P_j}_{C_j}\right) \wedge \left(\mathcal{L}^{P_j}_{C_j} \overset{0}{\sim} \mathcal{L}^{P_k}_{C_k}\right) \implies \left(\mathcal{L}^{P_i}_{C_i} \overset{0}{\sim} \mathcal{L}^{P_k}_{C_k}\right)$$

Properties :

1. Independent of the programming language used to code the programs
2. Hardware-independent space - only the nature of computation is compared
3. Stronger relation than the mapping- reducibility
4. The truth preservation orders for all programs in their respective domains is invariant under the transformation function. For Turing machines, this translates to a one-one correspondance between the number of state transitions taken.
5. Non-recursive enumerable in nature

Type-1 arrow - Sub-structure Transformational Isomorphism

In most cases, there will **only be a small part of $\mathcal{L}^{P_2}_{C_2}$ that is similar to the computation performed by $\mathcal{L}^{P_1}_{C_1}$, and not the whole program itself**. For example, if $\mathcal{L}^{P_1}_{C_1}$ is a loop, counting down from 10 to 1 in unit steps, and $\mathcal{L}^{P_2}_{C_2}$ is a computer program which contains a loop counting up from 50 to 100 in unit steps, surrounded by some other computation (like function calls etc.) then $\mathcal{L}^{P_2}_{C_2}$ contains a loop that counts up in ten unit steps, surrounded obviously, by the code to make up for the remaining computation.

Definition : Let $C_i \in \mathcal{C}_{\mathcal{S}_{i,\perp_i}}$, $C_j \in \mathcal{C}_{\mathcal{S}_{j,\perp_j}}$, $C_k \in \mathcal{C}_{\mathcal{S}_{k,\perp_k}}$ and $C_l \in \mathcal{C}_{\mathcal{S}_{l,\perp_l}}$ be four Boolean conditions. For two given programs $\mathcal{L}^{P_i}_{C_i} : \mathcal{S}_{i,\perp_i} \to \mathcal{S}_{i,\perp_i}$ and $\mathcal{L}^{P_j}_{C_j} = (\mathcal{L}^{P_m}_{C_m} \circ (\mathcal{L}^{P_k}_{C_k} \circ \mathcal{L}^{P_l}_{C_l}))$, for some $\mathcal{L}^{P_k}_{C_k}, \mathcal{L}^{P_l}_{C_l}, \mathcal{L}^{P_m}_{C_m} \in \mathcal{P}$ and $\mathcal{L}^{P_l}_{C_l} : \mathcal{S}_{l,\perp_l} \to \mathcal{S}_{l,\perp_l}$, then there exists an **arrow of type-1**, $A^1_{i,j} \in \mathcal{A}^1_{i,j}$, from $\mathcal{L}^{P_i}_{C_i}$ to $\mathcal{L}^{P_j}_{C_j}$, denoted by $\mathcal{L}^{P_i}_{C_j} \xrightarrow{A^1_{i,j}} \mathcal{L}^{P_j}_{C_j}$, if the following conditions hold:

1. $\mathcal{L}^{P_i}_{C_i}$ and $\mathcal{L}^{P_k}_{C_k}$ are transformationally isomorphic, i.e. $\mathcal{L}^{P_i}_{C_i} \overset{0}{\sim} \mathcal{L}^{P_k}_{C_k}$, through some arrow $A^0_{i,k} : \langle T_{i,k} \rangle$
2. The set of all valid input state assignments for $\mathcal{L}^{P_k}_{C_k}$ is a subset of the range of $\mathcal{L}^{P_l}_{C_l}$, i.e. $T_{i,k}(\mathcal{S}_{i,\perp_i}) \subseteq Im(\mathcal{S}_{l,\perp_l})$.

Here, $\mathcal{A}^1_{i,j}$ is the set of all arrows of type-1 that can exist between $\mathcal{L}^{P_i}_{C_i}$ and $\mathcal{L}^{P_j}_{C_j}$, one for every sub-program $\mathcal{L}^{P_k}_{C_k}$ of $\mathcal{L}^{P_j}_{C_j}$ such that $\mathcal{L}^{P_i}_{C_i} \overset{0}{\sim} \mathcal{L}^{P_k}_{C_k}$. The existence of $A^1_{i,j}$ establishes a **Sub-structure Transformational Isomorphism** between $\mathcal{L}^{P_i}_{C_i}$ and $\mathcal{L}^{P_j}_{C_j}$, denoted by $\mathcal{L}^{P_i}_{C_i} \overset{1}{\sim} \mathcal{L}^{P_j}_{C_j}$.

Properties :

1. The existence of $A^0_{i,j}$ implies the existence of $A^1_{i,j}$
2. *Cross Transitivity* between type-0 and type-1 arrows.
3. $\overset{1}{\sim}$ induces a partial order on the elements of $\mathcal{P}$.
4. Every diagram that commutes, containing arrows of type-0 between programs, can be converted into a diagram consisting of type-1 arrows only.