

# 1 Congruence with Isomorphic Binary Expression Nodes

**Theorem 1.1.** *The inferred type of a binary expression node from any valid abstract syntax tree is unaffected by the ordering of its two children nodes.*

*Proof.* Let  $\Sigma$  be the set of all nodes from a valid abstract syntax tree. We can then define  $\Gamma$  to be the set of all possible data types in our language, and similarly define  $F$  to be the set of all valid data types in our program, such that  $F \subset \Gamma$ .

Let us then describe the process of inferring a data type from any node  $\delta \in \Sigma$  as the transformation function  $\lambda(\delta) \rightarrow \gamma, \forall \delta \in \Sigma$  where  $\gamma \in \Gamma$ .

Let us also define the binary operation of comparing two data types as  $\cap : \Gamma \times \Gamma \rightarrow \Gamma$  such that  $\gamma_i \cap \gamma_j \rightarrow \gamma_k$  where  $\gamma_k \in F \leftrightarrow \gamma_i \equiv \gamma_j$  otherwise  $\gamma \notin F$ .

To infer the data type of a binary expression, we can further describe that particular function as  $\lambda(\hat{\delta}) = \lambda(\alpha) \cap \lambda(\beta)$  where  $\alpha$  and  $\beta$  represent the left and right children respectfully of the binary node  $\hat{\delta}$ .

To show that the inferred type of a node is unaffected by the ordering of its two children nodes, we need to show that the binary operation  $\cap$  is commutative such that  $\gamma_i \cap \gamma_j \equiv \gamma_j \cap \gamma_i$ .

It shows from our definition of  $\cap$  that for  $\gamma_i \cap \gamma_j$  to produce a data type namely  $\gamma_k$  such that  $\gamma_k \in F$ , it is required that both  $\gamma_i$  and  $\gamma_j$  must be the equivalent. Therefore  $\gamma_i \equiv \gamma_j \equiv \gamma$ , which implies that it is equivalent to say that  $\gamma_i \cap \gamma_i \rightarrow \gamma_k$  for some  $\gamma_k \in F$ .

Therefore, it must also be true to say that  $\gamma_j \cap \gamma_i \rightarrow \gamma_k$  for some  $\gamma_k \in F$  since it is empirically true that  $\gamma_i \cap \gamma_j \equiv \gamma_i \cap \gamma_i \equiv \gamma_j \cap \gamma_i$ .

Since  $\gamma_j \cap \gamma_i \equiv \gamma_i \cap \gamma_j$ , we can conclude that  $\lambda(\hat{\delta}) = \lambda(\alpha) \cap \lambda(\beta)$  where  $\alpha$  and  $\beta$  represent the left and right children respectfully of the binary node  $\hat{\delta}$  is equivalent to  $\lambda(\hat{\delta}) = \lambda(\beta) \cap \lambda(\alpha)$ .

Therefore, the proposition that the inferred type of a binary expression node from any valid abstract syntax tree is unaffected by the ordering of its two children nodes.  $\square$

**Corollary 1.1.1.** *Two binary expression nodes from any valid abstract syntax trees are said to have the same type if both nodes are isomorphic to one another.*

*Proof.* Let us assume that we have two binary expression nodes namely  $\alpha$  and  $\beta$  from some abstract syntax tree which are isomorphic to one another such that there exists a bijection  $\phi : \alpha \rightarrow \beta$  which preserves the integrity of each node through their transformations.

Therefore, if one node  $\alpha$  has some inferred type  $\gamma$ , then it is known from 1.1 that all isomorphic variations of  $\alpha$  will have the same inferred type  $\gamma$ .  $\square$