```java
package player;

import java.util.HashSet;
import java.util.Set;

import edu.neu.ccs.demeterf.demfgen.lib.List;
import edu.neu.ccs.satsolver.InputInitialI;
import gen.Constraint;
import gen.RawMaterial;
import gen.Type;
import gen.TypeInstance;

/**
 * Implements the InputInitialI interface.
 */
public class InputInitial implements InputInitialI {
    private HashSet<InputPair> pairs = new HashSet<InputPair>();

    /**
     * Build a type and compute the weighted pairs for all relations.
     */
    public InputInitial(RawMaterial rm) {
        Type t = buildType(rm);
        computeWeightedPairs(t, rm);
    }

    /**
     * An individual relation with weight 1.
     */
    public InputInitial(Type t) {
        computeWeightedPairs(t);
    }

    /**
     * Gets the pairs of relation numbers and fractional weights.
     */
    public Set<InputPair> getPairs() {
        return pairs;
    }

    /**
     * Build a type (no duplicates).
     */
    private Type buildType(RawMaterial rm) {
        List<TypeInstance> result = List.<TypeInstance> create();

        for (Constraint c : rm.instance.cs) {
            TypeInstance t = new TypeInstance(c.r);

            if (!result.contains(t)) {
                result = result.append(t);
            }
        }

        return new Type(result);
    }

    private void computeWeightedPairs(Type t) {
        for (TypeInstance i : t.instances) {
            pairs.add(new InputPair(i.r.v, 1));
        }
    }

    /**
     * Compute the fractional weight for each relation number.
     */
    private void computeWeightedPairs(Type t, RawMaterial rm) {
        double total = 0;

        // The total number of weighted constraints
        for (Constraint c : rm.instance.cs) {
            total = total + c.w.v;
        }

        // The weight for each relation
        for (TypeInstance i : t.instances) {
            double weight = 0;

            // Add up all weights for the given relation
            for (Constraint c : rm.instance.cs) {
                if (c.r.v == i.r.v) {
                    weight = weight + c.w.v;
                }
            }

            // Get the fraction of weight
            double fraction = weight / total;

            pairs.add(new InputPair(i.r.v, fraction));
        }
    }
}
```