Project: Solving proximity constraints

Jan-Michael Holzinger[*] Sophie Hofmanninger[†]

JKU Linz — SS2019

Version Number	Changes Summary	Author
0.1		Jan-Michael
0.2	added System Model	Jan-Michael

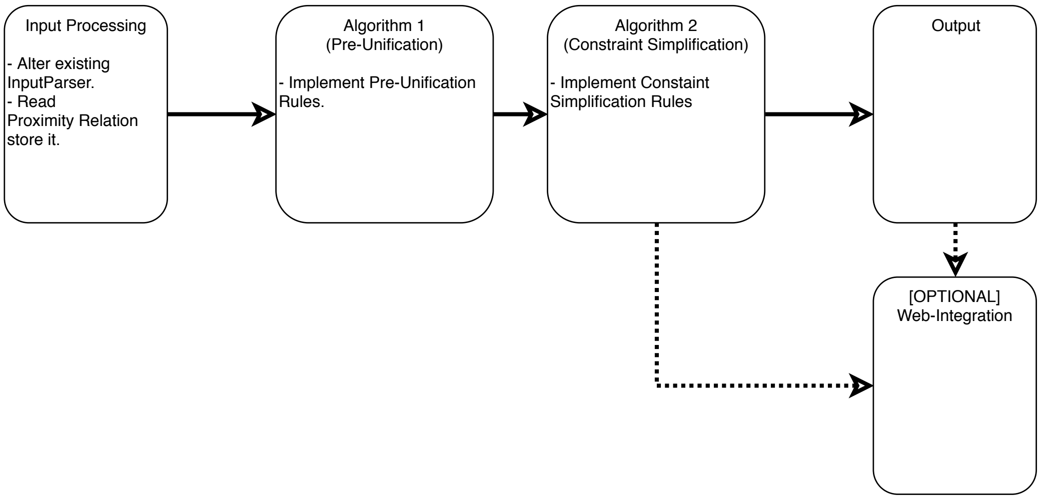
1 System Overview

We split the problem in 4 (5) smaller tasks:

1. Input Processing,
2. Pre-Unification,
3. Constraint Simplification,
4. Output.

O. Web-Integration.

[*]jan.holzinger@gmx.at
[†]sophie@hofmanninger.co.at


Input Processing
- Alter existing InputParser.
- Read Proximity Relation store it.
Algorithm 1
(Pre-Unification)
- Implement Pre-Unification Rules.
Algorithm 2
(Constraint Simplification)
- Implement Constaint Simplification Rules
Output
[OPTIONAL]
Web-Integration

1.1 Input Processing

The first idea here is to use and probably extend the given code, to proximity relations. The class that can be used for this is InputParser.

For the Proximity Relations $\mathcal{R}$ and the λ-*cut* we have the following idea:

1. We try to get the number of constants, variables and function symbols (n_1, n_2, n_3) respectively.
2. We let the user input 3 $n_i \times n_i$ symmetric matrices that constist of values in $[0, 1]$ for $i = 1, 2, 3$. These matrices must have a 1 in the main diagonal.
3. We let the user input $\lambda \in [0, 1]$ and calculate the set $\mathcal{R}_\lambda$.

1.2 Algorithms

We implement the Algorithms in an own class, that has two static functions, preUnification and CS.

1.2.1 Pre-Unification Algorithm

The preUnification method consists of a loop, that runs until either $\mathrm{P} = \emptyset$ or it is detected, that there is no solution to the problem.
Inside the loop body, the 7 pre-unification rules are iteratively applied to the first element (which gets popped by doing so).

The method changes the problems constraints and pre-unifier accordingly.

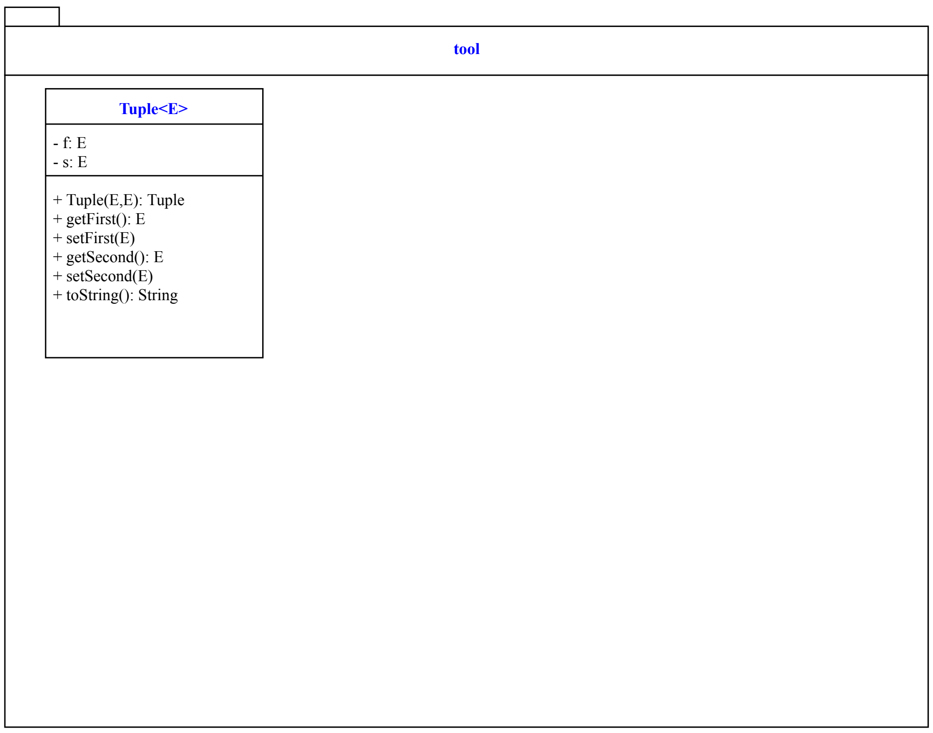
1.2.2 Constraint Simplification Algorithm

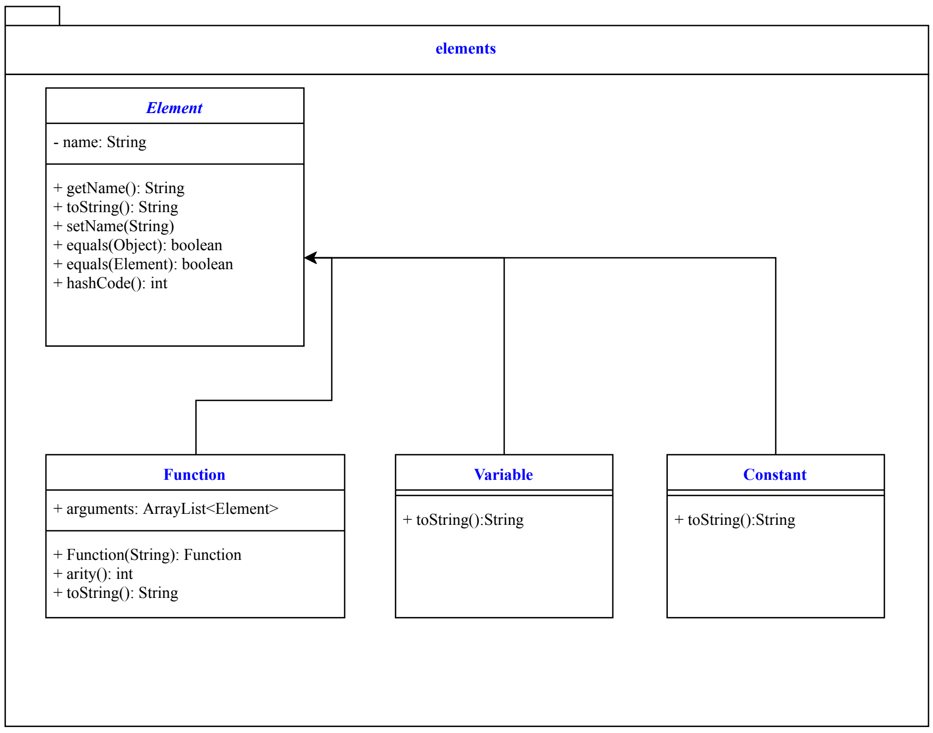
1.3 Output

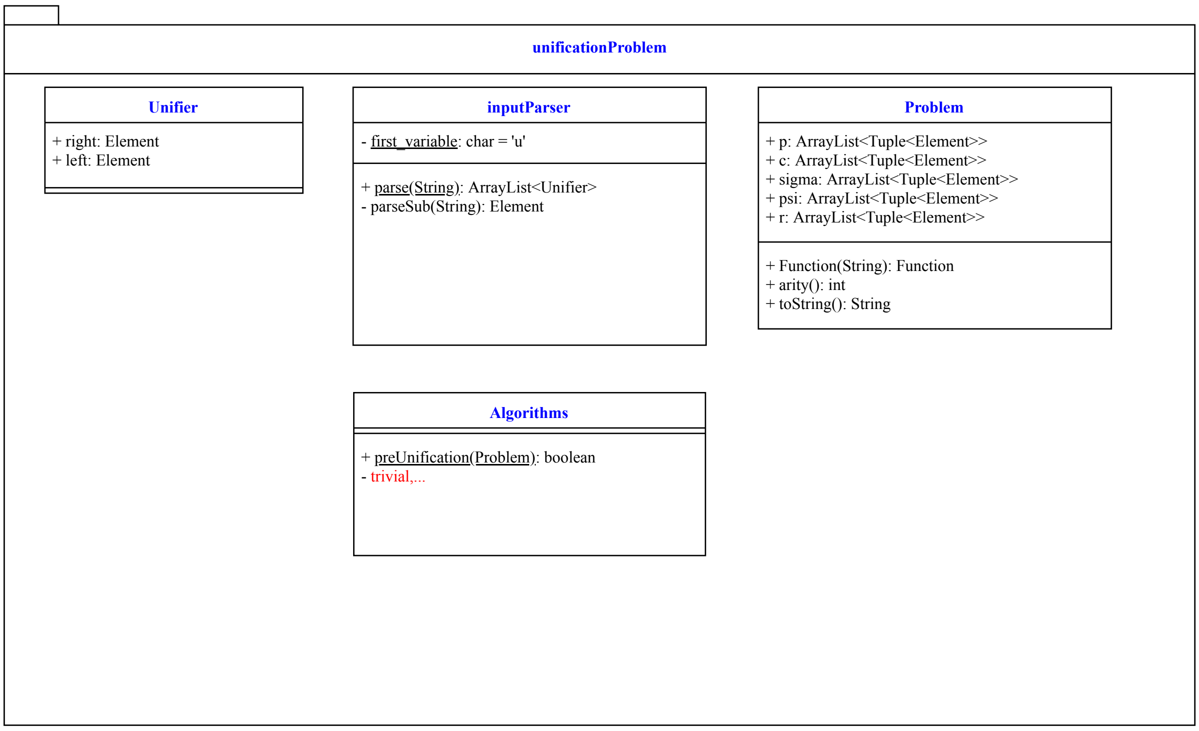
2 System Model

The program consists of 3 packages,

- tool
- elements
- unificationProblem


tool
Tuple<E>
- f: E
- s: E
+ Tuple(E,E): Tuple
+ getFirst(): E
+ setFirst(E)
+ getSecond(): E
+ setSecond(E)
+ toString(): String


elements
Element
- name: String
+ getName(): String
+ toString(): String
+ setName(String)
+ equals(Object): boolean
+ equals(Element): boolean
+ hashCode(): int
Function
+ arguments: ArrayList<Element>
+ Function(String): Function
+ arity(): int
+ toString(): String
Variable
+ toString():String
Constant
+ toString():String


unificationProblem
Unifier
+ right: Element
+ left: Element
inputParser
- first_variable: char = 'u'
+ parse(String): ArrayList<Unifier>
- parseSub(String): Element
Problem
+ p: ArrayList<Tuple<Element>>
+ c: ArrayList<Tuple<Element>>
+ sigma: ArrayList<Tuple<Element>>
+ psi: ArrayList<Tuple<Element>>
+ r: ArrayList<Tuple<Element>>
+ Function(String): Function
+ arity(): int
+ toString(): String
Algorithms
+ preUnification(Problem): boolean
- trivial,...