JOHANNES KEPLER
UNIVERSITÄT LINZ

Solving Proximity Constraints

Sophie Hofmanninger & Jan-Michael Holzinger

26.06.2019

Johannes Kepler University, A-4040 Linz, Austria

Motivation

For proving theorems, a frequently occurring problem is to find common instances of formulae.

Example 1

Let f be a function, a, b constants and x a variable. The two expressions

$$f(a, x) \quad \text{and} \quad f(a, b)$$

can be unified with $\{x \mapsto b\}$.

Motivation

For proving theorems, a frequently occurring problem is to find common instances of formulae.

Example 2

Let f, g be functions, a, b constants and x a variable. The two expressions

$$f(a, x) \quad \text{and} \quad g(a, b)$$

cannot be unified as $f \neq g$.

Motivation

In 1965 Robinson presented his unification algorithm and solved this problem, his algorithm was improved for better(=faster) performance since.

If we consider now the unification problem

$$f(a, x) \simeq^{?} g(a, b)$$

again, we might wonder, if we could not ignore $f \neq g$, if they are "close" to each other, i.e. if they are equal in a fuzzy logic sense. Being close is represented as a proximity relation, which are symmetric and reflexive, but not necessarily transitive. C. Pau and T. Kutsia solved this problem, presenting an algorithm, which we implemented.

Introduction

The Algorithm consists of two sub-algorithms and works on (modifies) 4 sets:

- P: unification problem to be solved ,
- C: neighbourhood constraint,
- σ: set of pre-unifier,
- Φ: name-class mapping,

where Algorithm 1 modifies P, C, and σ and Algorithm 2 modifies C and Φ. If Algorithm 1 was successful, $P = \emptyset$, if Algorithm 2 was successful $C = \emptyset$.

pre-Unification rules

(Tri) $\{x \simeq^? x\} \uplus P; C; \sigma \Rightarrow P; C; \sigma$

(Dec)
$\{F(\overline{s_n}) \simeq^? G(\overline{t_n})\} \uplus P; C; \sigma \Rightarrow \{\overline{s_n \simeq^? t_n}\} \cup P; \{F \approx^? G\} \cup C; \sigma$

(VE) $\{x \simeq^? t\} \uplus P; C; \sigma \Rightarrow \{t' \simeq^? t\} \cup P x \mapsto t'; C; \sigma\{x \mapsto t'\}$

(Ori) $\{t \simeq^? x\} \uplus P; C; \sigma \Rightarrow \{x \simeq^? t\} \cup P; C; \sigma$

(Cla) $\{F(\overline{s_n}) \simeq^? G(\overline{t_n})\} \uplus P; C; \sigma \Rightarrow \bot$ if $m \neq n$

(Occ)
$\{x \simeq^? t\} \uplus P; C; \sigma \Rightarrow \bot$ if there is an occurrence cycle of x in t

(VO)
$\{x \simeq^? y, \overline{x_n \simeq^? y_n}\}; C; \sigma \Rightarrow \{\overline{x_n \simeq^? y_n}\}\{x \mapsto y\}; C; \sigma\{x \mapsto y\}$

Rules for Neighbourhood Constraints

(FFS) $\{f \approx^? g\} \uplus C; \Phi \Rightarrow C; \Phi;$ if $\mathcal{R}(f, g) \geq \lambda$

(NFS) $\{N \approx^? g\} \uplus C; \Phi \Rightarrow C; \mathit{update}(\Phi, N \rightarrow \mathbf{pc}(g, \mathcal{R}, \lambda))$

(FSN) $\{g \approx^? N\} \uplus C; \Phi \Rightarrow \{N \approx^? g\} \cup C; \Phi$

(NN1)
$\{N \approx^? M\} \uplus C; \Phi \Rightarrow C; \mathit{update}(\Phi, N \rightarrow \{f\}, M \rightarrow \mathbf{pc}(f, \mathcal{R}, \lambda))$,
where $N \in \mathit{dom}(\Phi)$, $f \in \Phi(N)$

(NN2) $\{M \approx^? N\} \uplus C; \Phi \Rightarrow \{N \approx^? M\} \cup C; \Phi$, where
$M \notin \mathit{dom}(\Phi)$, $N \in \mathit{dom}(\Phi)$

(Fail1) $\{f \approx^? g\} \uplus C; \Phi \Rightarrow \bot$, if $\mathcal{R}(f, g) < \lambda$

(Fail2) $C; \Phi \Rightarrow \bot$, if there exists $N \in \mathit{dom}(\Phi)$ such that $\Phi(N) = \emptyset$

Simple example about how both algorithms work

Example 3

Let p, q and f be functions, b, c, c' constants and x, z variables. Then the following unification problem has a solution:

$$p(x, z) =^? q(f(b), f(x)) \quad \text{with} \quad R = \{(b, c'), (c', c), (p, q)\}$$

Simple example - pre-Unification fails

Example [Fail pU]

Examples where the pre-Unification algorithm fails:

$$\text{(Occ)} \quad p(x) =^? q(f(x)) \tag{1}$$

$$\text{(Cla)} \quad p(a, b) =^? q(f(x)) \tag{2}$$

Simple example - Constraint Simplification fails

Example [Fail CS]

Let p and f be functions, a, b constants and x, y variables. Then for the following unification problem only the pre-Unification algorithm is successful:

$$p(a, x, a) =^{?} q(y, b, x) \quad \text{with} \quad R = \{(b, c), (p, q)\}$$

Simple example cont.

pre-Unification

...

$C = \{p \approx^? q, N_1 \approx^? a, N_2 \approx^? b, a \approx^? N_2\}$

Constraint Simplification

$C = \{p \approx^? q, N_1 \approx^? a, N_2 \approx^? b, a \approx^? N_2\}$

$\Phi = \{\}$

$\Rightarrow_{\mathsf{FFS}}$

$C = \{N_1 \approx^? a, N_2 \approx^? b, a \approx^? N_2\}$

$\Phi = \{\}$

$\Rightarrow_{\mathsf{NFS}^2}$

Simple example cont.

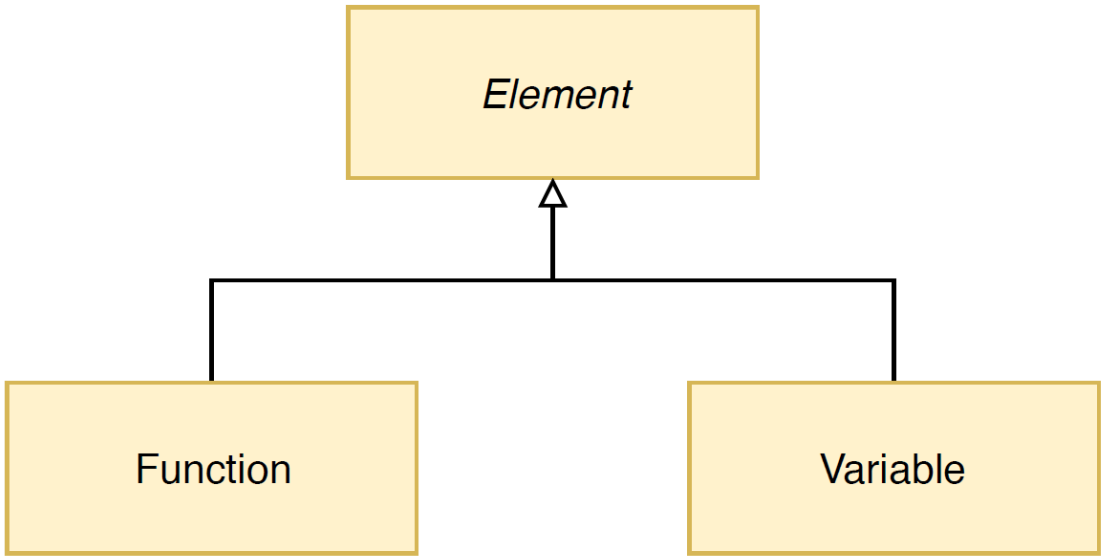
$$C = \{a \approx^? N_2\}$$
$$\Phi = \{N_1 \mapsto \{a\}, N_2 \mapsto \{b, c\}\}$$
$$\Rightarrow^{\text{FSN}}$$
$$C = \{N_2 \approx^? a\}$$
$$\Phi = \{N_1 \mapsto \{a\}, N_2 \mapsto \{b, c\}\}$$
$$\Rightarrow^{\text{NFS}}$$
$$C = \{\}$$
$$\Phi = \{N_1 \mapsto \{a\}, N_2 \mapsto \emptyset\}$$
$$\Rightarrow^{\text{Fail2}}$$
$$\bot$$

System Model

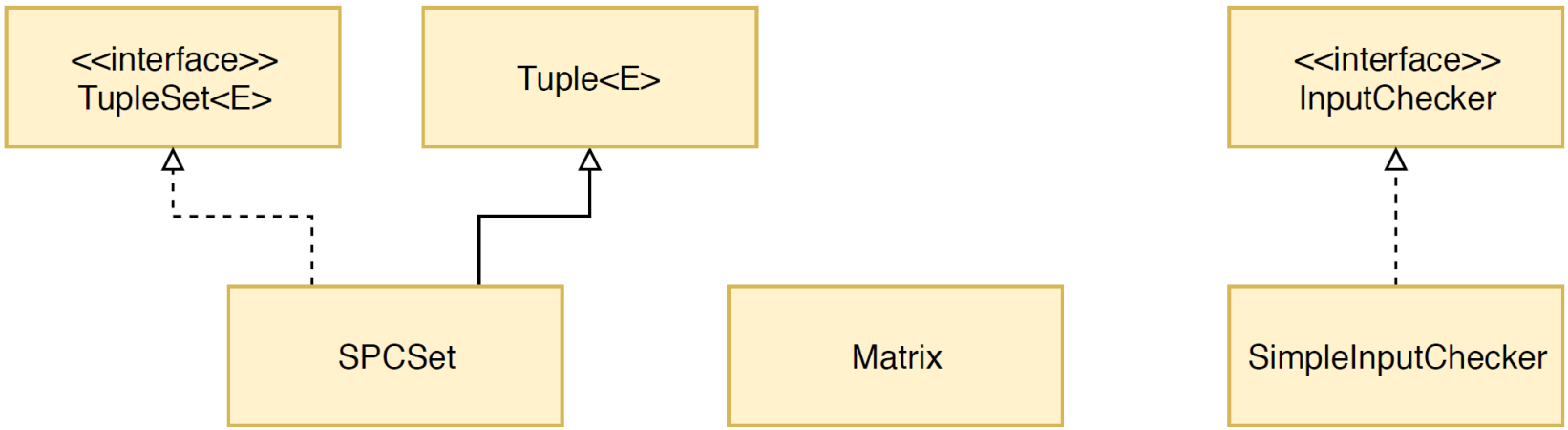
Project consists of 4 packages:

- elements : deals with elements
- tool : offers important tools (e.g. read input)
- unificationProblem : treats the unification problem
- userInterfaces : allow user interfaces

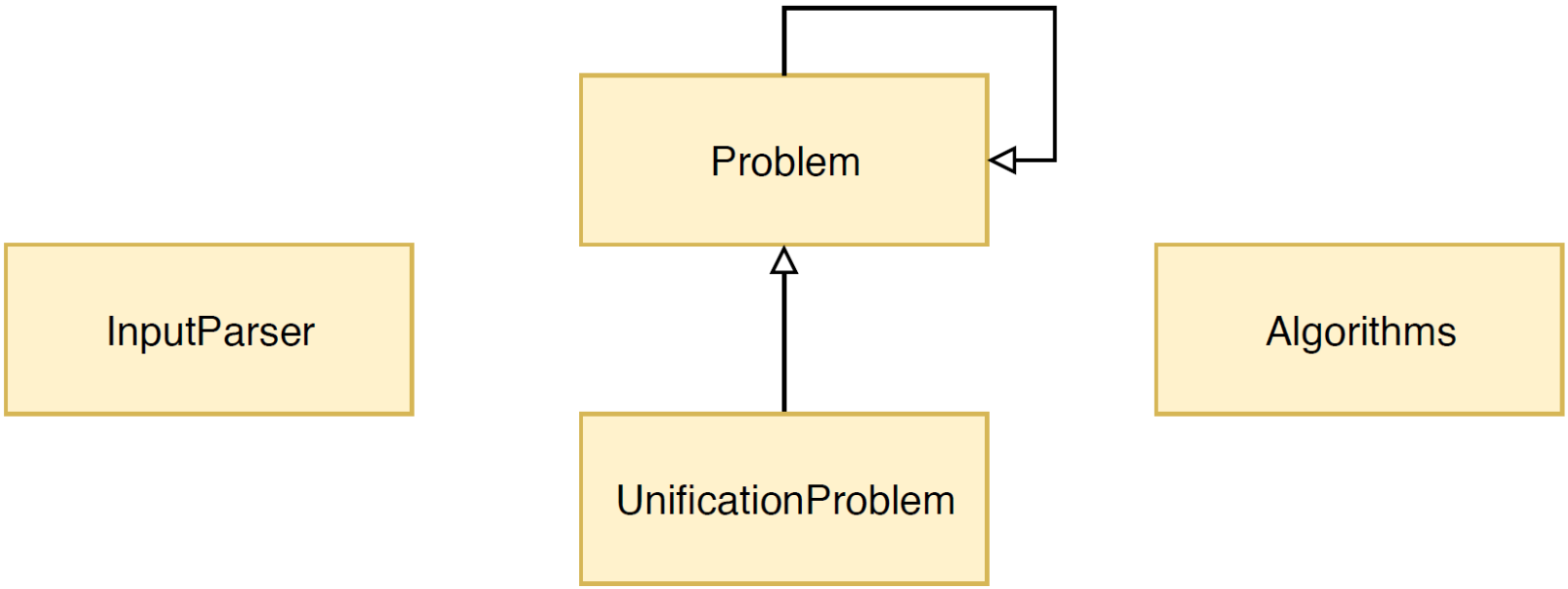
Package elements



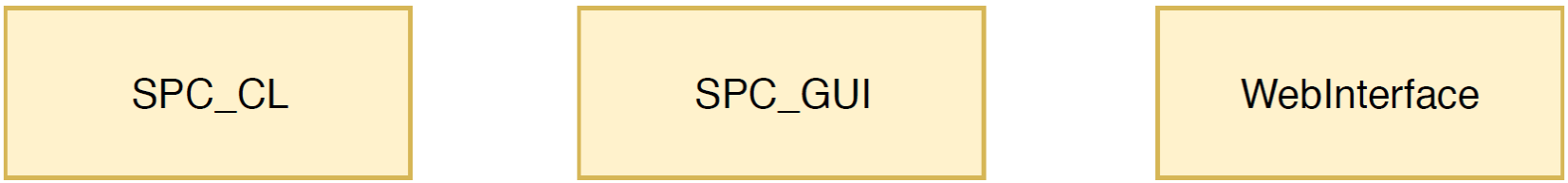
Package tool



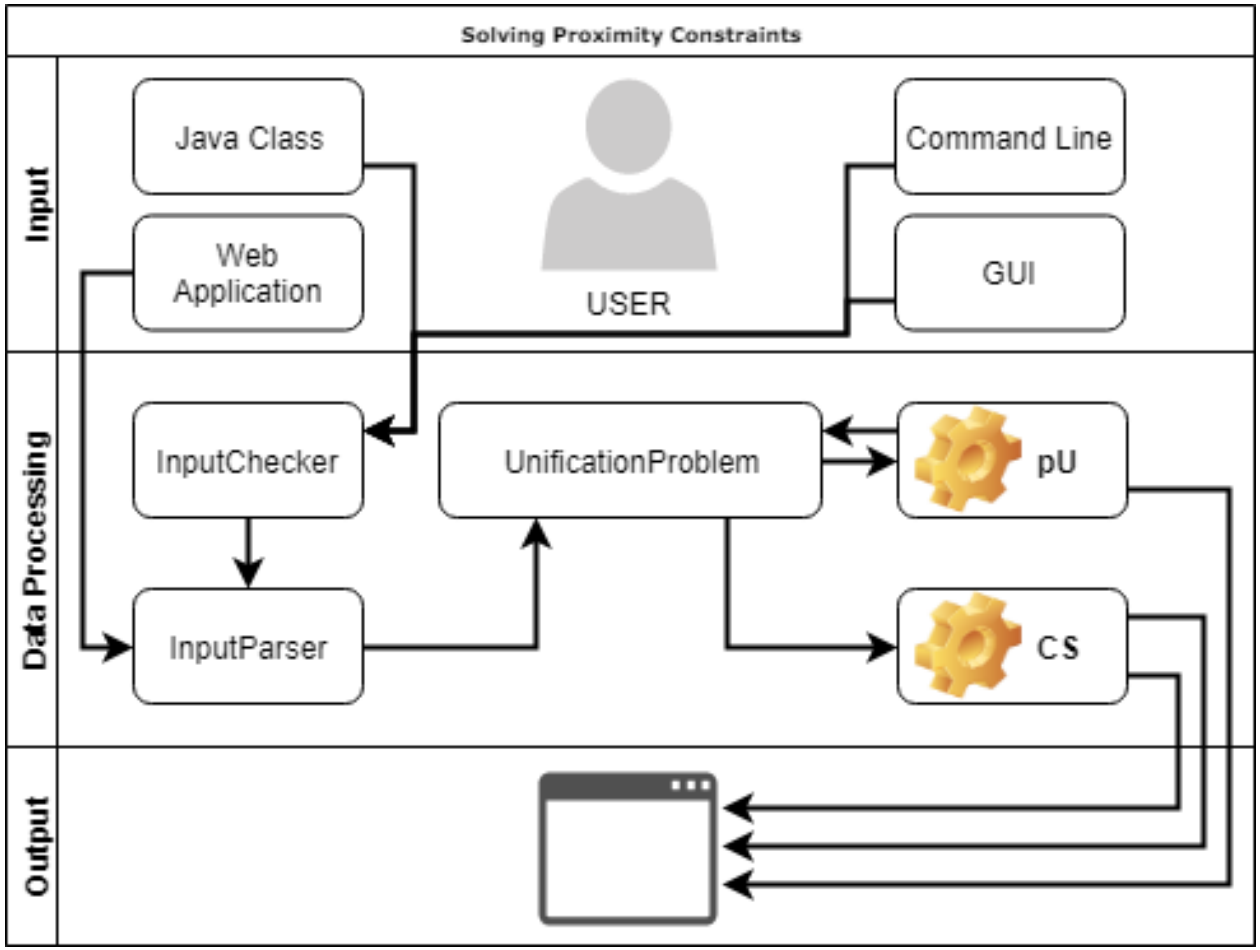
Package unificationProblem



Package userInterfaces



Workflow


Solving Proximity Constraints
Input
Java Class
Web Application
USER
Command Line
GUI
Data Processing
InputChecker
UnificationProblem
pU
InputParser
CS
Output

Redmine/UML

We found Redmine and its features very helpful for Software development. Mainly, we used it to keep track of work that has to be done, that was done, and also see what the other person is working on. We also tried out the Forum, to communicate when we needed, but we found Whatsapp more suitable for this.
For communication: In our opinion, communication is one of the most important things when working in a team, but we tried to restrict ourselves to the communications channels offered by the lecture, so basically Redmine, meeting a few minutes before the lecture, and Whatsapp that is like the Forum + notification when read.
From a very early stage on, we also used UML, to express and communicate our ideas about the project and the implementation as well as to structure it.

Git/Javadoc

As it is a requirement for the project, we of course used git alot. We tried to use branches and merge. As suggested, we committed (and most of the times also pushed) as soon as we further developed the project. As we did not work on the project for 8 hours straight every day in the week, this pretty much models the usage in a company, where you pull at the beginning, then work/commit to save your work and at the end of the day (or at a "milestone") push.
It was very important to us to use Javadoc from the beginning on for almost each method. The reason for that was that eclipse has the nice feature of displaying it as a tooltip, and so you can understand easily the behaviour of the method, even if the other person implemented it.

JUnit/Jenkins

For testing, that was very necessary, to find bugs, we used JUnit 5, although we later found out it is not too easy to use it from command line/Jenkins. It also was a good feeling, that whenever one had altered parts of the code, and run JUnit everything worked out again. At the beginning we had the feeling that Jenkins would not be as useful in our project as it might be in others, but actually, it was nice to use it to simply run all tests. Even though realizing that JUnit 5 has (also) strict naming conventions for test classes took us some time.