REFINITY **to Model and Prove Program Transformation Rules**

APLAS 2020 · December 2nd, 2020

Dominic Steinhöfel

steinhoefel@cs.tu-darmstadt.de
@DomSteinhoefel
dominic-steinhoefel.de

Software Engineering Group · CS Department · TU Darmstadt

DEMO TIME!

Photo by Markus Spiske on Unsplash

Photo by Wojciech Then on Unsplash

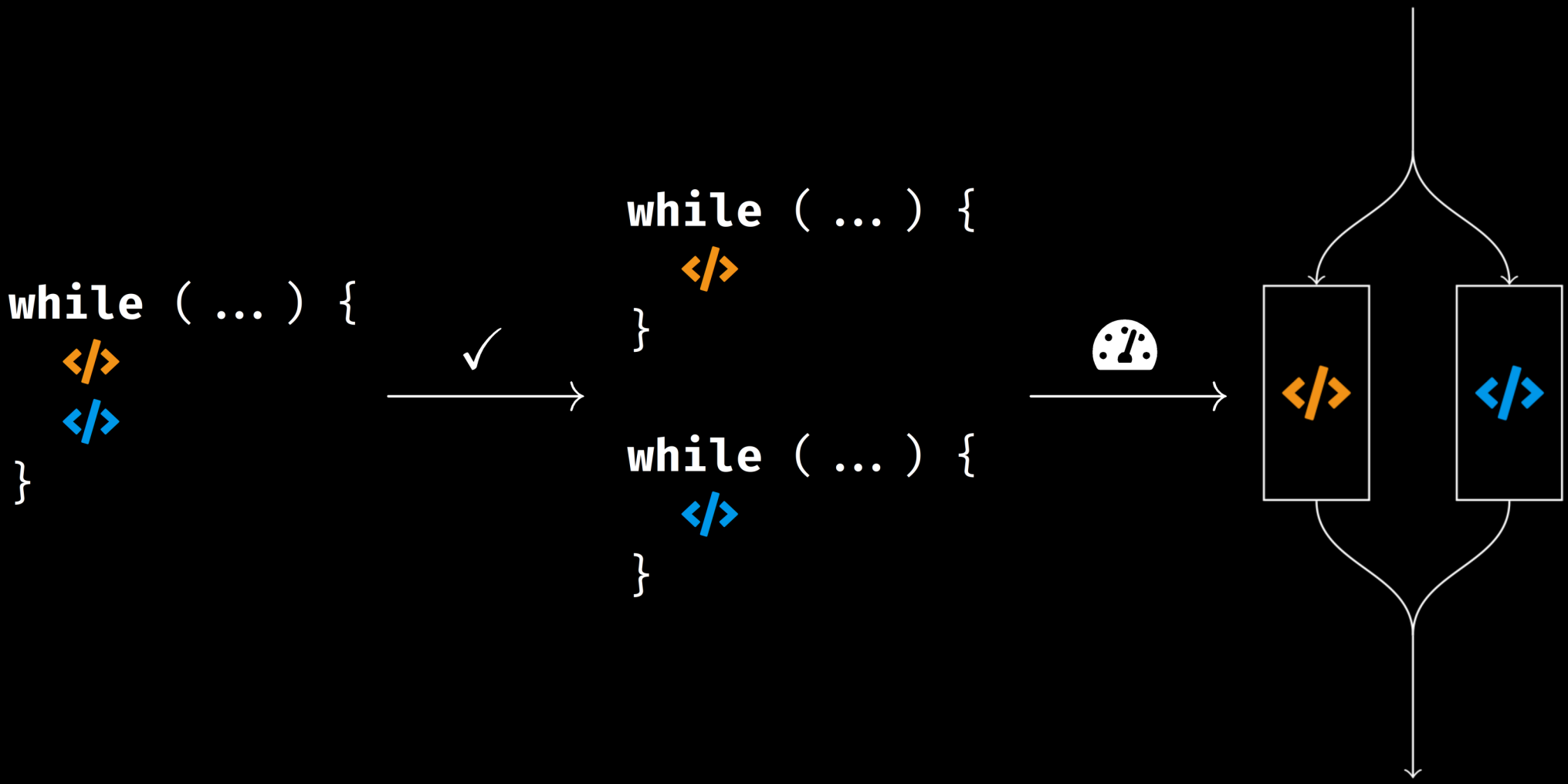
Refactoring

```
try {
  </>
} catch ( ... ) {
  </>
}
```


```
if (</>) {
  </>
} else ( ... ) {
  </>
}
```

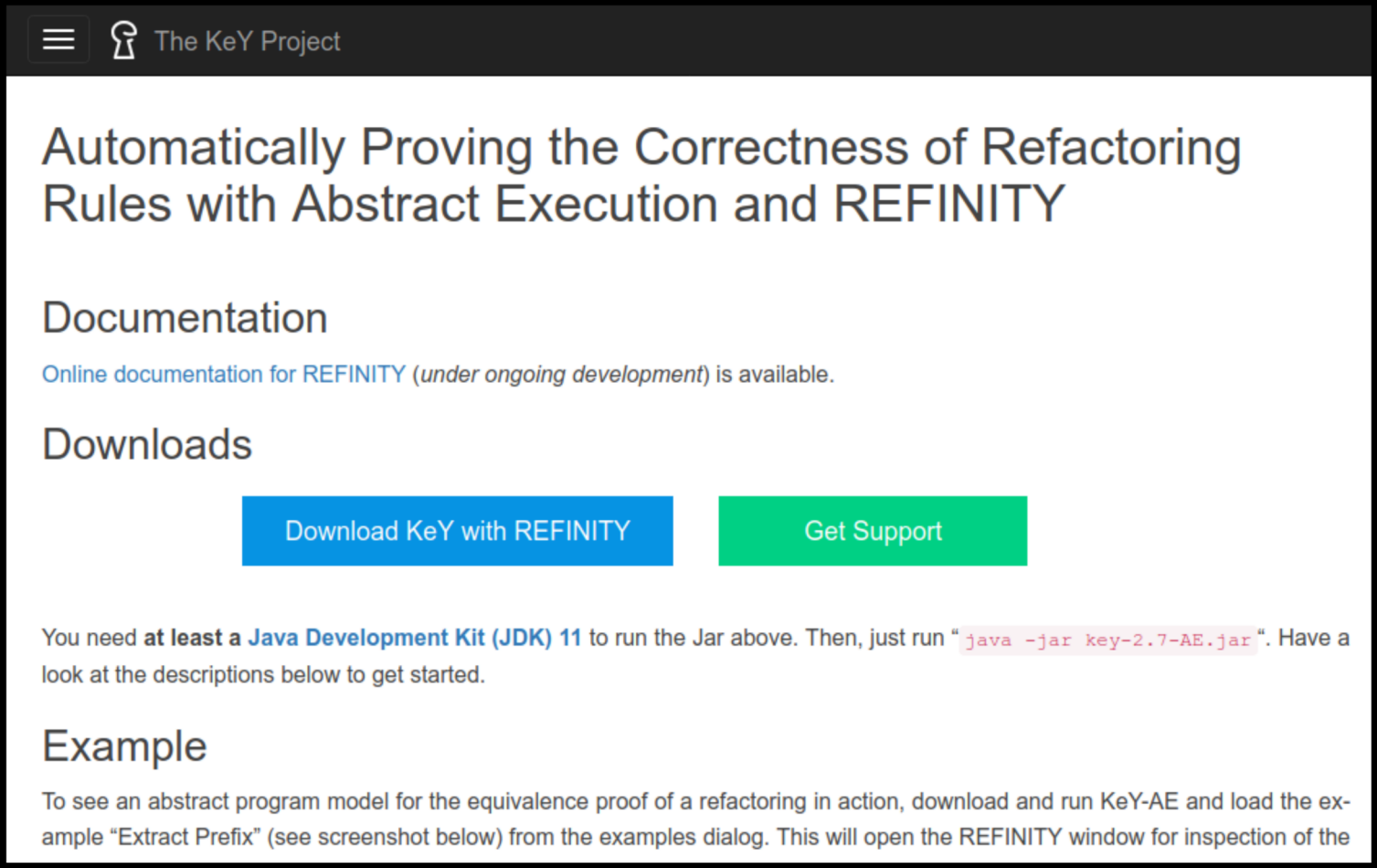
Abstract Cost Analysis

Restructuring for Parallelization



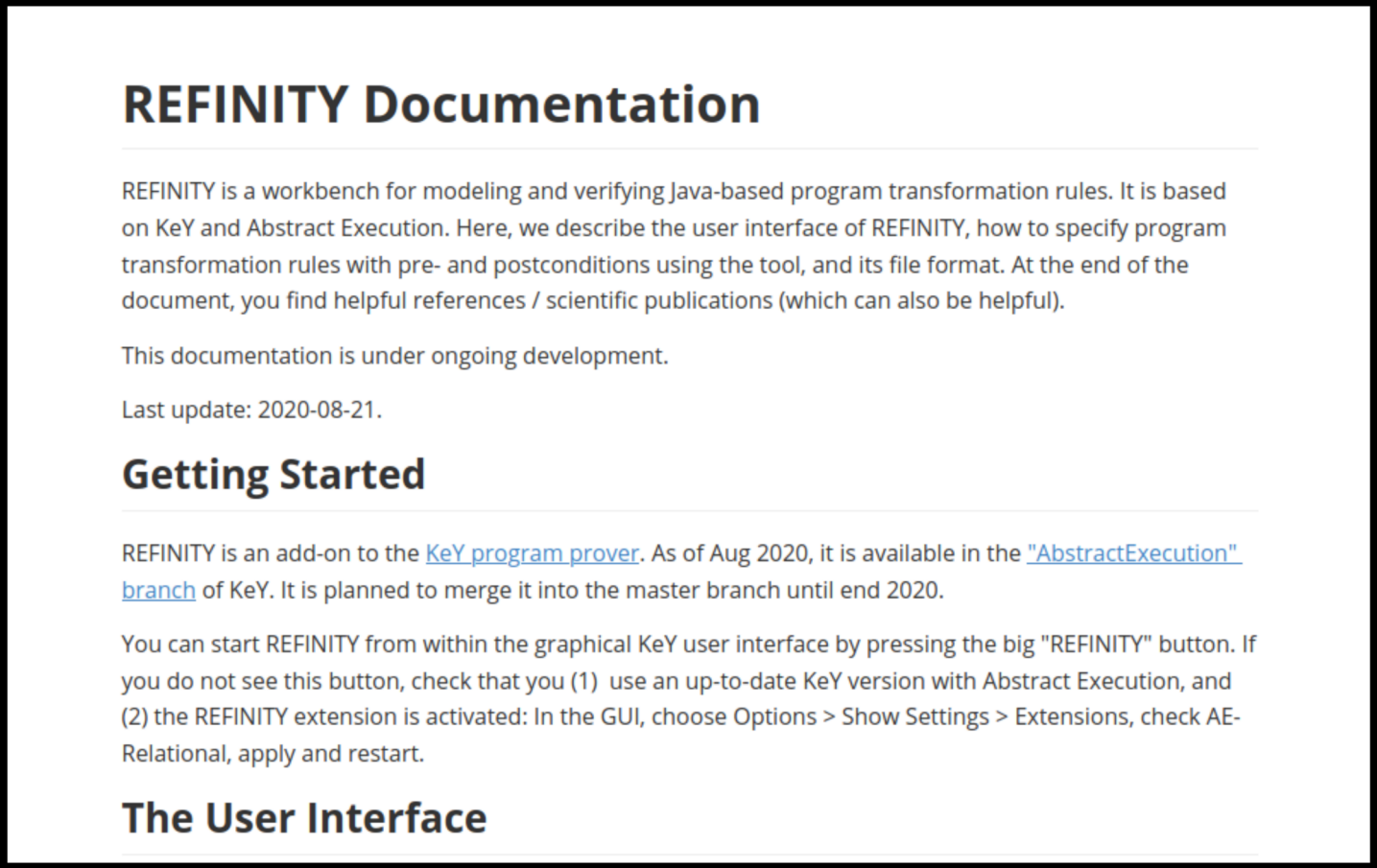

REFINITY Resources
Photo by Tyler Casey on Unsplash

Download



key-project.org/REFINITY

Documentation



REFINITY Documentation

REFINITY is a workbench for modeling and verifying Java-based program transformation rules. It is based on KeY and Abstract Execution. Here, we describe the user interface of REFINITY, how to specify program transformation rules with pre- and postconditions using the tool, and its file format. At the end of the document, you find helpful references / scientific publications (which can also be helpful).

This documentation is under ongoing development.

Last update: 2020-08-21.

Getting Started

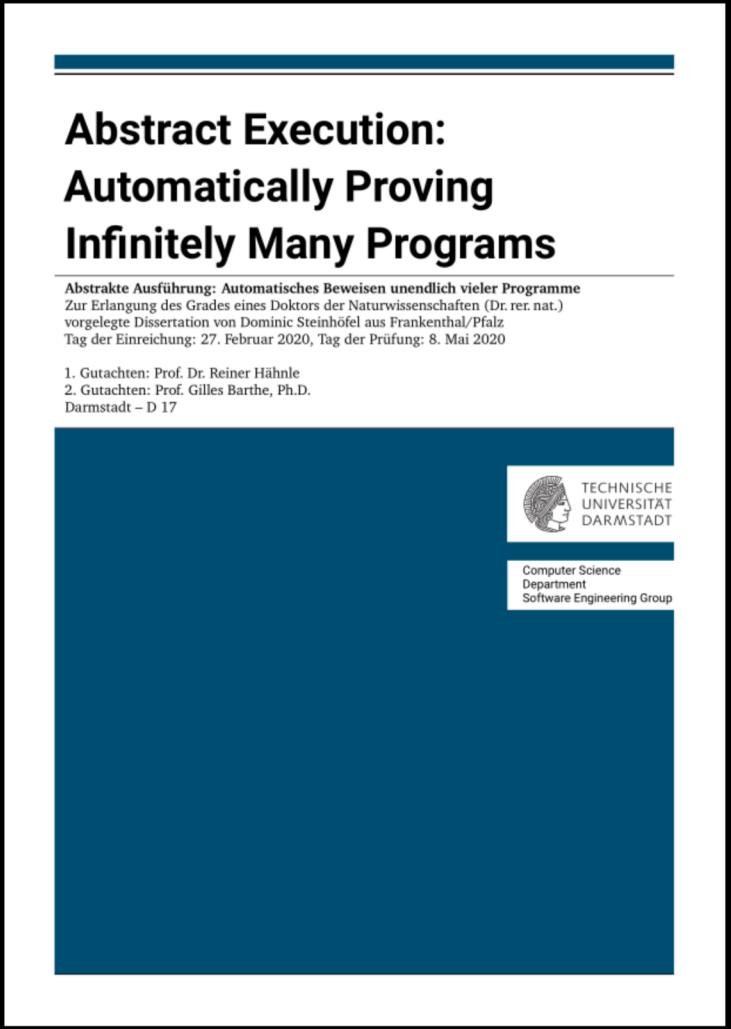
REFINITY is an add-on to the KeY program prover. As of Aug 2020, it is available in the "AbstractExecution" branch of KeY. It is planned to merge it into the master branch until end 2020.

You can start REFINITY from within the graphical KeY user interface by pressing the big "REFINITY" button. If you do not see this button, check that you (1) use an up-to-date KeY version with Abstract Execution, and (2) the REFINITY extension is activated: In the GUI, choose Options > Show Settings > Extensions, check AE-Relational, apply and restart.

The User Interface

key-project.org/material/REFINITYDoc/

More Details & Background



Abstract Execution: Automatically Proving Infinitely Many Programs

Abstrakte Ausführung: Automatisches Beweisen unendlich vieler Programme
Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
vorgelegte Dissertation von Dominic Steinhöfel aus Frankenthal/Pfalz
Tag der Einreichung: 27. Februar 2020, Tag der Prüfung: 8. Mai 2020

1. Gutachten: Prof. Dr. Reiner Hähnle
2. Gutachten: Prof. Gilles Barthe, Ph.D.
Darmstadt – D 17

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science
Department
Software Engineering Group

https://doi.org/10.25534/tuprints-00008540

Photo by Mohamed Ajufaan on Unsplash

Transformation rules as pairs of schematic programs

Proof generation and certificate checking

REFINITY

Editing support: Shortcuts, syntax checking, tool tips, ...

Instance checking

steinhoefel@cs.tu-darmstadt.de
@DomSteinhoefel
dominic-steinhoefel.de