**Evaluation “*Extended-KeY-Sourceview”* Plugin**

In der beiliegenden zip Datei findet sich eine KeY.jar und 2 java Programme ( CaesarChiffre1.java und CaesarChiffre2.java ).  
Beide implementieren eine Caesar Verschlüsselung mit JML Spezifikation und beide haben jeweils einen Fehler weshalb KeY die Spezifikation nicht beweisen kann.

Der Fehler der ersten Datei soll mit dem Extended-SourceView-Prototypen gefunden werden und der Zweite ohne Quellcodeansicht nur mit dem Sequenzen Fenster.

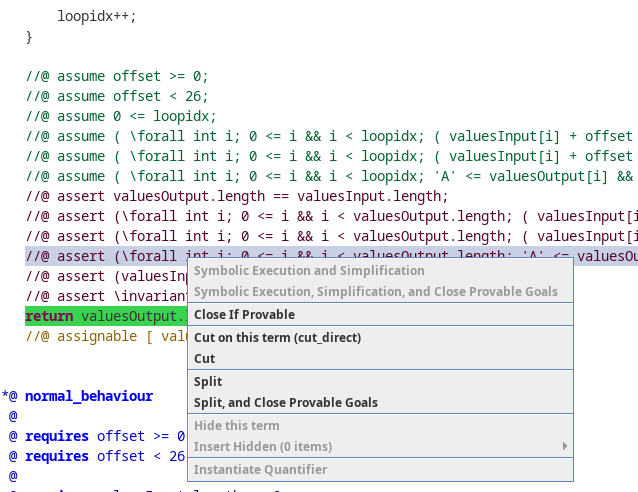
Vorbereitung

* Die zip Datei mit beiden Programmen entpacken.
* Bitte den .key Ordner im Home Verzeichnis temporär umbenennen, damit KeY mit den Standardeinstellungen läuft. Am Ende kann der Ordner wieder zurück umbenannt werden.
* Bitte die Zeit merken an denen die beiden Teile jeweils begonnen wurden, damit man im folgenden Fragebogen die Bearbeitungsdauer beider Aufgabenteile angeben kann.

Teil 1: Anleitung (Programm 1):

1. KeY.jar starten und CaesarChiffre1.key öffnen.
2. Den Methodenkontrakt der Methode *calcChiffre()* auswählen
3. Das Unterfenster „Sequent“ minimieren, so dass nur „Loaded Proofs“, „Proof“, und „Source“ sichtbar sind.
4. Ziel ist es zu erkennen an welcher Stelle im Quellcode/Spezifikation der Fehler eingebaut ist welcher das (automatische) Schließen des Beweises verhindert.

Hinweise

Illustration 1: Kontextmenü in der Quellcodeansicht

* In der Quellcodeansicht wird die Sequenz (durch das Extended-SourceView-plugin) als JML Einschübe angezeigt
* Interaktionen können in der Quellcodeansicht durchgeführt werden indem man an einer beliebigen Position rechts klickt.  
  Term-bezogene Interaktionen erfordern es auf einen bestimmten Term rechts zu klicken.
* Da die Sequenzenansicht ausgeblendet ist können dort keine Interaktionen durchgeführt werden. Dies ist nur auf im Proof-Fenster und dem Source-Fenster möglich.

Anleitung (Programm 2):

1. KeY.jar starten und CaesarChiffre2.key öffnen.
2. Das Source-Fenster verbergen und das Sequent-Fenster wieder anzeigen.  
   Nur die die Fenster „Loaded Proofs“, „Proof“, und „Sequent“ sollten sichtbar sein.
3. Den Quellcode von CaesarChiffre2 kann man sich in einem externen Texteditor anzeigen.
4. Dann, äquivalent zu Teil 1, versuchen den Fehler zu identifizieren.

Danach bitte den Fragebogen auf der nächsten Seite ausfüllen und per Email zurückschicken.

Name

|  |
| --- |
|  |

Was ist der Fehler in Teil 1.

|  |
| --- |
|  |

Was ist der Fehler in Teil 2.

|  |
| --- |
|  |

Wie viel Zeit hast du für Teil 1 gebraucht?

|  |
| --- |
|  |

Wie viel Zeit hast du für Teil 2 gebraucht?

|  |
| --- |
|  |

Wie hast du die Position der JML Einschübe interpretiert?  
Gib eine kurze Erkläung an was, deiner Meinung nach, die Position aussagt.

|  |
| --- |
|  |

Welche Bedeutung hast du der \old<*$num*>(*$term*) Syntax zugeordnet?  
Gib eine kurze Erkläung an, was ein solcher Ausdruck bedeutet.

|  |
| --- |
|  |

Welche Teile der Präsentation würdest du ändern?  
Was hat dir an der UI gefehlt?

|  |
| --- |
|  |

Welche zusätzlichen Interaktionen in der Quellcodeansicht hättest du noch gebraucht, bzw. gerne zur Verfügung gehabt?

|  |
| --- |
|  |

Vergleiche die Sequenzansicht und die neue Quellcodeansicht.

Mit welcher Ansicht konntest du besser den aktuellen Beweiszustand verstehen?

|  |
| --- |
|  |

Zusätzliche Kommentare / Freitext:

|  |
| --- |
|  |

*Vielen Dank für die Teilnahme :)*