Bachelorarbeit

TCT Web Interface

Abdullatif Yesil
abdullatif.yesil@student.uibk.ac.at

10. März 2011

Betreuer: Martin Avanzini, MSc.
Priv.-Doz. Dr. Georg Moser
Dipl.-Ing. Andreas Schnabl

Zusammenfassung (Englisch)

Within this project, a webinterface for TcT is developed. The interface is autonomous in terms of the browser. Easy operability was of high importance. All the functionalities that are made available are as self-explanatory as possible and have a clear structure. The TcT can be called by an automatic or a semi-automatic mode. The latter enables a facile configuration of each TcT strategies. The administration of the strategies is dynamic and depends on the used version of TcT . The input of the rewriting system can bedone directly with in the interface or by a file. Should a file be uploaded, also an archive of files can be uploaded, so that the contained rewriting systems can be treated iterativly by the TcT .

In diesem Projekt wurde ein Webinterface für TcT entwickelt. Die Oberfläche ist Browserunabhängig und es wurde besonderen Wert auf einfache Bedienbarkeit gelegt: alle bereitgestellten Funktionalitäten sind möglichst selbsterklärend und strukturiert aufgebaut. Das Webinterface erlaubt den Aufruf von TcT anhand eines automatischen und eines semi-automatischen Modus. Letzterer ermöglicht eine einfache Konfiguration aller Strategien von TcT . Die Verwaltung der Strategien ist dynamisch und hängt von der jeweils verwendeten Version von TcT ab. Die Eingabe des Termersetzungssystems ist entweder direkt im Interface oder über ein File möglich. Soll ein File hochgeladen werden, kann auch ein Archiv von Files hochgeladen werden, sodass die darin enthaltenen Ersetzungssysteme iterativ von TcT behandelt werden.

Inhaltsverzeichnis

1 Einleitung

Wie der Begriff *"Termersetzungssysteme"* von sich aus bereits erläutert, führt ein Termersetzungssystem Ersetzungen von Termen aus. Hierbei stellt sich zunächst die Frage was eine Termersetzung von der Begrifflichkeit her zu bedeuten hat.

Die Termersetzung, ein Begriff aus der Mathematik, besagt, dass Terme nach bestimmten Regeln umgestellt werden können, ohne ihre Semantik zu verändern. Es handelt sich somit um eine Äquivalenzumformung. Eine Termersetzung führt im Regelfall zu einer Verbesserung der Termeigenschaften. Diese Optimierung kann darin liegen, dass eine bessere Lesbarkeit geschaffen wird oder aber der Wegfall redundanter Teilterme die Schreibweise eine Erleichterung erfährt. Durch eine Termersetzungsregel r kann ein Term t durch einen anderen Term t' ersetzt werden, ohne seine Bedeutung zu verlieren.

TCT ("Tyrolean Complexity Tool") ist ein Softwarewerkzeug zur automatischen Bestimmung der Komplexität eines Termersetzungssystem. TCT wurde von meinen Betreueren im Rahmen des Projektes "Derivationskomplexitätsanalyse", finanziert durch den Fond wissenschaftlicher Forschung (FWF) entwickelt. FWF Projektnummer ist P20133 erstellt[1]. Bei Eingabe eines Termersetzungssystem R wird, abhängig von den gewählten Parametern, verifizert, ob R polynomielle Derivations- oder Laufzeitkomplexität aufweist. Ist dies der Fall wird eine asymptotisch Komplexität in groß-O Notation ausgegeben. Bisher war TCT nur von der Kommandozeile aus aufrufbar. TCT ist open-source und frei [2] erhältlich.

Die Aufgabe in diesem Projekt besteht darin, eine Web-Benutzeroberfläche für das TCT zu entwickeln. Diese Webseite wurde mit Google Web Toolkit[3] und SmartGWT[4] Frameworks in JAVA geschrieben und AJAX-Technologie verwendet.

[1] http://www.fwf.ac.at/de/abstracts/abstract.asp?L=D&PROJ=P20133
[2] http://cl-informatik.uibk.ac.at/research/projects/derivational-complexity-analysis/
[3] http://code.google.com/webtoolkit/
[4] http://code.google.com/p/smartgwt/

2 Termersetzungssysteme

Die Termersetzungssysteme (TRS) sind ein formales Berechnungsmodell in der Theoretischen Informatik.[1] Sie bilden insbesondere die Grundlage der Logik- und funktionalen Programmierung. Ferner spielen sie eine wichtige Rolle beim Wortproblem und bei der Terminierungsanalyse.

Termersetzungssysteme sind Mengen von so genannten Termersetzungsregeln. Diese Mengen kann man sich wie Gleichungssysteme zwischen Termen vorstellen, bei dem die Gleichungen nur von links nach rechts angewendet werden dürfen.

Weitere Information zu Termersetzungssysteme sind in [2] zu finden.

Durch eine Termersetzungsregel r kann ein Term t durch einen anderen Term t' ersetzt werden, ohne seine Bedeutung zu verlieren.

Beispiel 2.1. *Sei das Termersetzungssystem R gegeben durch*

$$\begin{array}{llll} 1\colon & x - 0 \to x & 3\colon & 0 \div s(y) \to 0 \\ 2\colon & s(x) - s(y) \to x - y & 4\colon & s(x) \div s(y) \to s((x - y) \div s(y)) \end{array}$$

Das Beispiel beschreibt die Division zweier Zahlen, die in unärer Kodierung (mit 0 und Nachfolger s) dargestellt werden.

Beispiel 2.2. Wir nehmen

$$s(s(0)) - s(0) \to_R$$

und verfeinern mit Regeln in R (Beispiel 2.1)

$$s(s(0)) - s(0) \to_R s(0) - 0 \to_R s(0)$$

$s(0)$ *ist Normalform.*

TRS Format:

Der eingegebene TRS-Text soll in der für das TRS Format geeigneten Grammatik eingegeben werden. Dieses Format wird in Rahmen der "Termination Problems Data Base"(TPDB)[2] verwendet.

Beispiel 2.3. Für Beispiel 2.1

[1] http://de.wikipedia.org/wiki/Termersetzungssystem
[2] http://www.termination-portal.org/wiki/TPDB

```
(VAR x y )
(RULES
minus(x, 0) -> x
minus(s(x), s(y)) -> minus(x, y)
quot(0, s(y)) -> 0
quot(s(x), s(y)) -> s(quot(minus(x, y), s(y)))
)
```

Definition 2.4. die derivationelle Komplexität in Bezug auf eine TRS R [4]

$$dc_R(n) = dh(n, \ddot{a}ll\ terms", \rightarrow_R)$$

Definition 2.5. die Laufzeit der Komplexität in Bezug auf eine TRS R

$$rc_R(n) = dh(n, "basic\ terms", \rightarrow_R)$$

Definition 2.6.

$$dh(n, T, \rightarrow) = max\{dh(t, \rightarrow) | \exists t$$

3 TCT (Tyrolean Complexity Tool)

3.1 Was ist TCT

Durch das Anwenden des TCT [1] (Tyrolean Complexity Tool) Softwarewerkzeugs, wird die Komplexität eines Termersetzugssystems automatisch bestimmt. Das Softwarewerkzeug wurde im Rahmen des Projektes "Derivational Complexity Analysis"über den Fond wissenschaftlicher Forschung (FWF). FWF Projektnummer ist P20133. Wird ein Termersetzugssystem r eingegeben, so wird anhand der gewählten Parameter verifizert, ob es sich bei r um eine polynomielle Derivations- oder eine Laufzeitkomplexität handelt. Falls eine solche Komplexität gegeben ist, so erfolgt eine asymptotische Komplexität in groß-O Notation.

Das Softwarewerkzeug ist open-source und somit kostenlos[2] erhältlich. Bisher konnte TCT nur von der Kommandozeile aus aufgerufen werden.

Es existieren diverse Arten, die *Komplexität* [4] eines Termersetzugssystems auszudrücken. Termersetzungssystem R als primitivste Methode kann die Anzahl der Schritte gezählt werden, die durch R von einem Term t aus durchgeführt werden können. Die Anzahl der Schritte wird in Bezug zu dem Term t gesetzt. Die "Größe"von t setzt sich aus der Addition der vorkommenden Symbole zusammen.

Basierend auf diesem Ansatz, können zwei Funktionen differenziert werden: Die Derivationskomplexität (*"derivational complexity"*) sowie die Laufzeitkomplexität (*"runtime complexity"*) einer Termersetzungssystems. Bei der Derivationskomplexität ist die maximale Anzahl der Schritte die in einem Ersetzungssystem vorgenommen werden können, bezogen auf die Größe des Startterms bedeutend. Eine Verfeinerung findet bei der Laufzeitkomplexität statt, da die Menge der betrachteten Startterme geeignet eingeschränkt werden kann.

Beide Komplexitätsmaßnahmen können durch das TCT berechnet und zertifiziert werden. Die obere Schranke, die das Programm automatisch errechnet, wird schließlich in groß-O-Notation dargestellt. Hierbei wird zunächst die Termination des Termersetzungssystems gezeigt und aus dieser bestehenden Information die obere Schranke abgeleitet.

3.2 TCT Ausführen

TCT ist ein in einer Konsole ausführbares Programm.

Standard Ausführung der TCT ist;

```
~$ tct <option>* <file>
```

[1] http://cl-informatik.uibk.ac.at/research/projects/derivational-complexity-analysis/
[2] http://cl-informatik.uibk.ac.at/software/tct/

Ein paar Option Argument Beispiele:

- $-a$: Antwort Kategorie (Komplexitätsmaßnahmen)
- $-t$: Zeit Eingabe für Timeout
- $-s$: Strategie

Beispiel 3.1. Etwa wie in diesem Beispiel:

```
$ TCT -a rc -s "matrix" #3.1.xml

YES(?,O(n^1))

'matrix-interpretation of dimension 2'
--------------------------------------
Answer:           YES(?,O(n^1))
Input Problem:    runtime-complexity with respect to
 Rules:
   {  minus(x, 0()) -> x
    , minus(s(x), s(y)) -> minus(x, y)
    , quot(0(), s(y)) -> 0()
    , quot(s(x), s(y)) -> s(quot(minus(x, y), s(y)))}

Proof Output:
 The following argument positions are usable:
   Uargs(minus) = {1}, Uargs(0) = {}, Uargs(s) = {1},
   Uargs(quot) = {1}
 We have the following constructor-restricted matrix
interpretation:
 Interpretation Functions:
  minus(x1, x2) = [1 0] x1 + [0 0] x2 + [1]

                  [0 1]      [0 0]      [0]

  0() = [1]

        [0]

  s(x1) = [1 0] x1 + [3]

          [0 1]      [0]

  quot(x1, x2) = [2 0] x1 + [0 1] x2 + [2]

                 [0 0]      [0 1]      [0]
```

Zur Veranschaulichung wurde im oberen Beispiel die sogenannte Matrixmethode angewandt [1]. Für diese Methode ist es von Notwendigkeit, die Terme

über die Vektoren von natürlichen Zahlen zu interpretieren und ein sich hierfür eignendes Maß zu definieren, das mit jedem Ersetzungsschritt reduziert wird. Hieraus kann automatisch eine Aussage über die Laufzeitkomplexität getroffen werden.

Weitere Information zu dieser Methode der Komplexitätsbestimmung von Termersetzungssystem sind in [3] zu finden.

3.3 TCT-Info

Die Strategie sind dynamisch und hängt von der jeweils verwendeten Version von TCT ab. Diese Informationen werden sich in einer Datei befunden. Diese Datei Name ist *tct_info*. Jeder Prozess bzw. Strategie beinhaltet zwei Argument-Typen. Optional und Positional Arguments und jedes Argument hat außerdem verschidene Flag-Typen (*Nat*, *Bool*, *Select*, *Subprocessor*). Diese Flags sind für beide Argument-Typen gültig. Optionale Argumente können auf Wunsch des Benutzer aktiviert werden, allerdings sollten alle positionalen Argumente eingestellt werden.

Beispiel 3.2. *Einfache TCT -Info Code*

```
Strategy {name = wdg; //Strategie Name ist wdg
 description = ["...."];
 synopsis = "wdg #OPTIONALARGS #1";
 positionalArguments = [(1 ,
     Argument {name = "subprocessor";
        isOptional = False;
        description = "The processor t....";
        values = "<processor>";})];
     Argument {name = "uargs";
        isOptional = True;
        description = "This argument sp.....";
        values = "<bool>";
        default = On;}];
}
```

4 Aufgabe

Die Aufgabe in diesem Projekt besteht darin, eine Web-Benutzeroberfläche für TCT zu entwickeln. Die Oberfläche hat Browser unabhängig und auf Standard konfigurierte Apache-Server[1] lauffähig zu sein.

Die Funktionalität der Oberfläche ist in zwei Modi konzipiert. Bei dem Ersten handelt es sich um ein Full Automatic Mode. In diesem Modus können kurze Termersetzungssysteme (TRS) Texte über das Text-Eingabefeld eingegeben werden, die Timeout Zeit kann bestimmt und eine Kategorie/Komplexitätsmaßnahme (Derivationskomplexität, Laufzeitkomplexität) ausgewählt werden. Auch kann das TRS-Text als TPDB- oder XML-Datei oder sowie mehrerer TRSs über zip-, tar.gz- oder tgz-Datei hochgeladen werden. Die Ergebnisse werden auf einer separaten Seite dargestellt. Auf dieser neuen Seite steht ebenfalls die Zeit für das Timeout, die gewählte Kategorie, die Strategie, der Dateiname, die Dauer der Ausführung, das Ergebnis, sowie der Beweis. Für Archiv Dateien steht die Zeit für das Timeout, die gewählte Kategorie, die Strategie, der Archive-Dateiname und die Statistik der Ergebnisse. Jegliche Dateien, die in der Archive-Datei stehen, werden nacheinander ausgeführt und in einer Tabelle angezeigt.

Der zweite Modus ist ein Semi Automatic Mode. Dieser Modus fungiert als eine Erweiterung des ersten Modus. Zusätzlich zu der vorherigen Funktionalität kommt die Definition der Beweisstrategie hinzu. Durch das Aufrufen einer Seite werden jedes mal die TCT -Strategien aktualisiert. Die Strategien können auch *"rekursiv"* definiert werden, so kann die Strategie beispielsweise *wdg* als Parameter einer Substrategie verlangen.

[1] http://tomcat.apache.org/

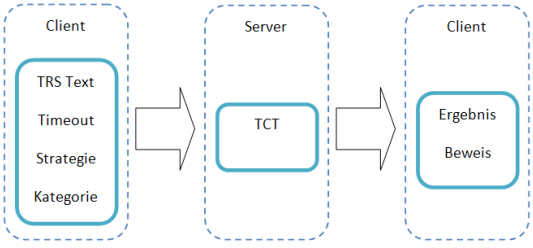


Abbildung 4.1: Schematische Beschreibung des WEB Interface: Die Aufgabe besteht darin, dem Server zugeschickte Termersetzungssysteme(TRS)-Eingaben mit einer gewählten Kategorie (Komplexitätsmaßnahme) und Strategie im TCT-Programm auszuführen und die Ergebnisse an den Benutzer zurückzuschicken.

5 Hintergrund

5.1 Benutzte Technologien

- Server: Apache-Server Tomcat 6[1]
- Web Technik: AJAX[2]
- Web Framework: Google Web Toolkit[3]
- Web Framework Erweiterung: SmartGWT[4]

5.1.1 AJAX

AJAX ist ein Methode der Datenübertragung zwischen einem Client und dem Server. Es ermöglicht das Durchführen von Anfragen, während eine Seite angezeigt wird, und diese Seite zu verändern, ohne sie komplett neu laden zu müssen. Dieser Vorgang benötigt keine zusätzlichen Browser Plugins .

Es gibt noch weitere Alternativen. Zum Beispiel kann mit Flash oder Javaapplets die selbe Funktionalität erhalten werden, diese Alternative braucht jedoch Browser Plugins, verursacht eine längere Ladezeit und beansprucht auch in Still Zustand einen Task Speicher.

AJAX hat auch Nachteile, zB:

- AJAX basiert auf Javascript, was in manchen Browsern ausgeschaltet ist - die Funktionen wären damit hinfällig.
- der 'Zurück'-Button des Browsers funktioniert nicht mehr effektiv, da die dynamischen AJAX-Schritte nicht in der Historie des Browsers gespeichert werden.

[1] http://tomcat.apache.org/
[2] http://www.webmasterpro.de/coding/article/ajax-einfuehrung-uebersicht.html
[3] http://code.google.com/webtoolkit/
[4] http://code.google.com/p/smartgwt/

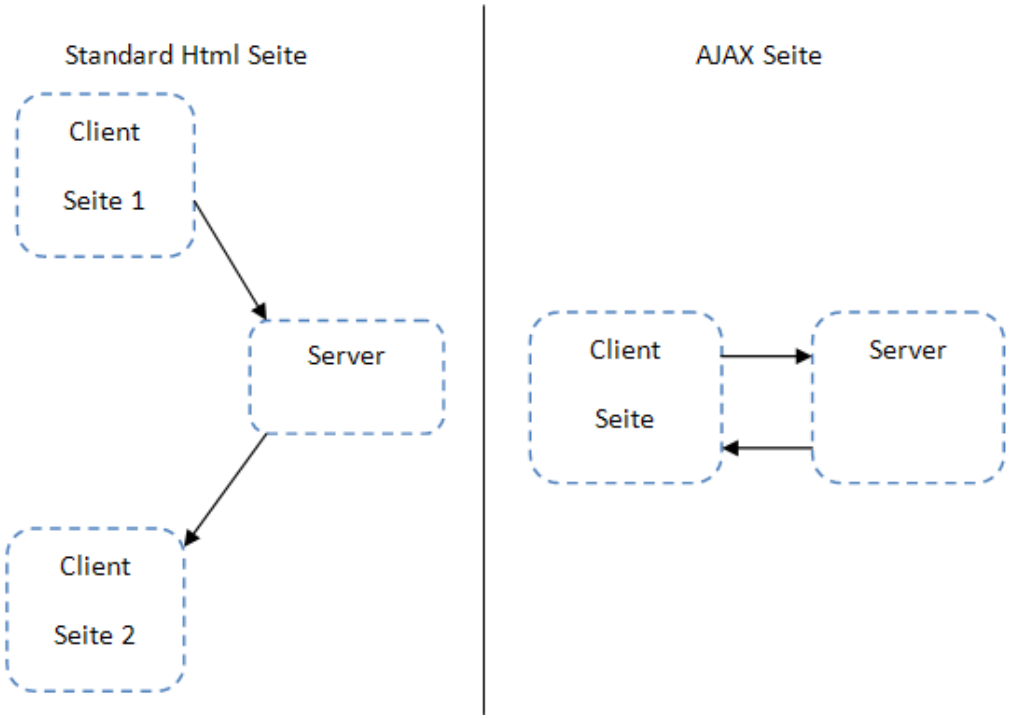


Abbildung 5.1: Standard Html und AJAX Datenübertragung Vergleich

5.1.2 Google Web Toolkit (GWT)

Google Web Toolkit (GWT) ist ein Framework zur Entwicklung von Webanwendungen. Es wurde von Google als freie Software veröffentlicht. Die Besonderheit hierbei ist, dass die gesamte Entwicklung von Client und Server AJAX Anwendungen auf der Basis von Java realisiert werden kann.

Beispiel 5.1. In diesem Beispiel sendet der Client eine Zeichenkette nach Server, der Server einem String die Zeichenfolge hinzuzufügt und sendet es zurück. Schließlich zeigt die Client kommende Antwort vom Server in Browser:
Simple Source Code - Client:

```
public class MyApplication implements EntryPoint  {

    public void onModuleLoad ()
   {
        // define the service you want to call
        MyServiceAsync svc = (MyServiceAsync)
                        GWT.create(MyService.class);
        ServiceDefTarget endpoint =
                         (ServiceDefTarget) svc;
        endpoint.setServiceEntryPoint("/myService");

        // define a handler for what to do when the
        // service returns a result
```

```
        AsyncCallback callback = new AsyncCallback()
        {
            public void onSuccess (Object result)
            {
                RootPanel.get().add(new
                                    HTML(result.toString()));
           }

            public void onFailure (Throwable ex)
            {
                RootPanel.get().add(new
                                    HTML(ex.toString()));
            }
        };
        //execute the service
        svc.myMethod("Hello World!", callback);
    }
}

public interface MyService extends RemoteService
{
    public String myMethod (String s);
}

public interface MyServiceAsync
{
    public void myMethod(String s,
                                    AsyncCallback callback);
}
```

Simple Source Code - Server:

```
public class MyServiceImpl extends RemoteServiceServlet
        implements MyService  {
    public String myMethod (String s)
    {
       return "You sent us '" + s + "'";
    }
}
```

Ausgabe: Nach mit GWT Compiler kompilieren, wird es mit einer Web Browser aufgerufen und folgende Seite erstellt:

5.1.3 SmartGWT

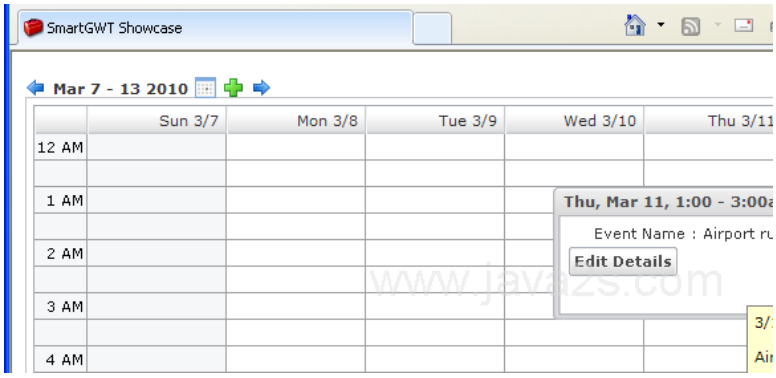
Smart GWT ist ein Framework auf Basis des Google Web Toolkits, das es erlaubt, die GWT-Widget-Bibliothek für die Entwicklung von Benutzeroberflächen einzusetzen und diese zwecks Datenverwaltung mit der Server-Seite zu verbinden. SmartGWT bildet das GWT-Widget-Bibliothek zu SmartClient[5]. SmartClient hat spezielles Client/Server Architekturmuster, das bestimmte Aspekte der Verteilung genauer festlegt[6].

Beispiel 5.2. *Simple Source Code - Kalender Beispiel[7]: Calendar Klasse wurde in SmartClient vordefiniert. Wir können diese Klasse in unsere GWT Applikation benutzen.*

```
public class Showcase implements EntryPoint{
    public void onModuleLoad() {
    RootPanel.get().add(getViewPanel());
    }
    public Canvas getViewPanel() {
      Calendar calendar = new Calendar();
      calendar.setData(CalendarData.getRecords());
      return calendar;
    }
}
class CalendarData {
//Meetings..
//Events..
...
}
```

Ausgabe:

[5]http://www.smartclient.com/
[6]http://de.wikipedia.org/wiki/Smart_Client
[7]http://www.java2s.com/Code/Java/GWT/SimpleeventcalendarSmartGWT.htm


SmartGWT Showcase
Mar 7 - 13 2010
Sun 3/7
Mon 3/8
Tue 3/9
Wed 3/10
Thu 3/11
12 AM
1 AM
2 AM
3 AM
4 AM
Thu, Mar 11, 1:00 - 3:00a
Event Name : Airport ru
Edit Details
www.javazs.com

6 Implementierung

6.1 Upload

Um eine Datei hinaufladen, werden zusätzlich zwei Apache Bibliotheken benutzt. Die Bibliotheken sind, um eine Datei hochzuladen, unter *commons-fileupload-1.2.2.jar* und um die aufgeladene Datei zu finden, unter *commons-io-2.0.jar* aufzusuchen.

Beispiel 6.1. Simple Source Server Code: *HttpServlet* Klasse und *Servlet* Klasse befinden sich in $commons - fileupload - 1.2.2.jar$

```
public class UploadFileServlet extends HttpServlet
                                implements Servlet {
        protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws
                        ServletException, IOException {
        response.setContentType("text/plain");
        FileItem uploadItem = getFileItem(request);
        new File(sessionID).mkdirs();
        File saveTo = new File(newFileSaveUrl);
        try {
                uploadItem.write(saveTo);
        } catch (Exception e) {
}
```

6.2 Ergebnis Kategorie

Die Ergebnisse werden in vier Kategorien eingeteilt und für jede Kategorie wurde eine eigene Default-Strategie eingestellt.

- dc: derivational complexity
- idc: innermost derivational complexity
- rc: runtime complexity
- irc: innermost runtime complexity

  Als Beispiele geben wir der Strategie für 'irc'

  ```
  (best (matrix :dim 2) (pop*))
  ```

Diese Strategien wurden in einer Txt Datei unter unterschiedlichen Namen gespeichert.

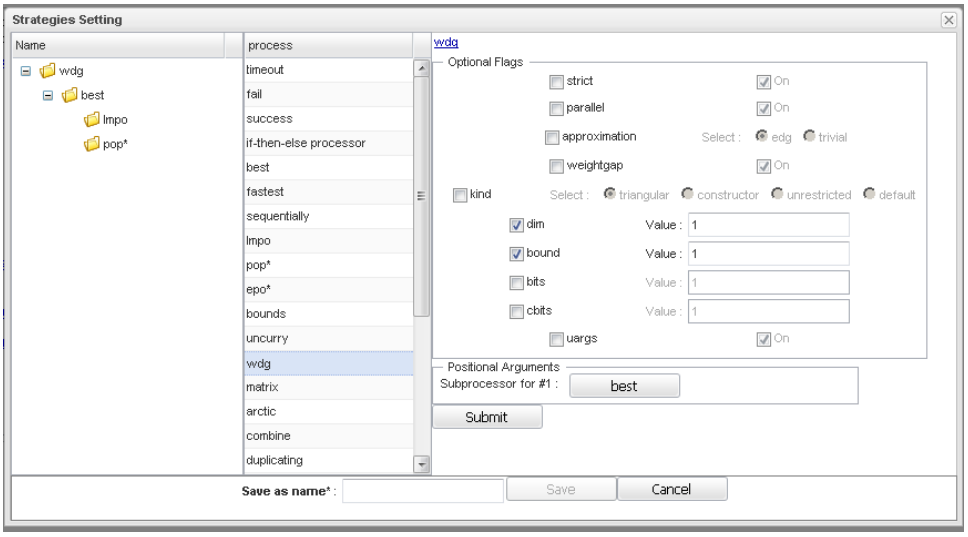
6.2.1 TCT-Info

Klickt der Benutzer die Semi Modus Open Box an, so werden zunächst die aktuellen Strategien aufgerufen. Diese stehen in der *tct-info.txt*-Datei. Diese *tct-info.txt*-Datei wird dynamisch die Zusammensetzung der Strategien bestimmt und mittels des tct_info zur Verfügung gestellt.

6.2.2 Strategie Wizard

Wizard:

Den Wizard wird in drei Spalten angezeigt.



Auf der linken Seite können die Kategorien der Ausführung äls Baum anzeigenëingestellt werden. Das mittlere Fenster beinhaltet die Liste der Prozessoren und die gespeicherten Strategien. Auf dem rechten Fenster werden die Details des im linken oder mittleren Fenster gewählten Prozesses angezeigt.

Rekursive Erstellung:

Jeder Prozess kann ein oder mehrere Kindprozessoren haben. Die Einstellung dieser Kindprozessoren erfolgt über neue Fenster. Auch der Subprozessor kann wiederum Subprozessoren haben.

Baum Erstellen:

Mit dem Finish Button können schließlich die eingestellten Strategien erstellt und als Baum dargestellt werden.

Speichern

Mit dem Save Button wird die Strategie unter einem eigenen Namen gespeichert und das Fenster wird geschlossen.

6.2.3 Ausführen

TCT Ausführen

Den Quellcode für das Ausführen TCT ist:

```
String [] cmd = {"./tct_","-t",timeout,
                 "-a",category,
                 "-s",strategy,file};
Process pr = rt.exec(cmd);
```

Die Ausführungszeitvon TCT wird in Sekunden gemessen.

Ergebnisse Abholung:

Da TCT ein in einer Konsole ausführbares Programm ist, werden die Ergebnisse auch in einer Konsole dargestellt und mit der Scanner-Methode gelesen.

Damit die Ergebnisse auf der Web-Seite richtig angezeigt werden, wurden HTML-Tags eingefügt und Hintergrund wird mit der Art des Ergebnisses (YES, NO, MAYBE, ERROR, TIMEOUT) gefärbt.

Beispiel 6.2. *Result*[1]+ = *s.replaceAll*(,"* *")
Leere Zeichen wird mit HTML Leerzeichen Code ' ' ersetzt.

6.2.4 Ergebnis Seiten

Die Ergebnis-Seiten sind auf einem extra Tab Fenster mit Datei Typen, Text/eine TRS/TXT Datei oder eine Archive-Datei, auf verschiedenen Seitentypen dargestellt.

```
if (result[0][0].equals("--FILES")) {
        //Wenn mehrere File Haben
        // Iterative Aufruf ...
} else if (result[0][0].equals("--FILE")) {
        //Wenn eine File haben
        // Einzelner Aufruf ...
}
```

Einzelnes Ergebnis:

Auf dieser Seite steht das Timeout, die Kategorie, die Strategie und der Dateiname/TextArea. Weiterhin sind die Ausführungszeit, das Ergebnis und der Beweis oder wenn ein Fehler aufgetreten ist, eine geschriebene Begründung dargestellt.

Archive-Datei Ergebnis:

Diese Seite zeigt ebenfalls das Timeout, die Kategorie, die Strategie und den Dateinamen. Dann stehen die Ergebnisse als Statistik gruppiert und die Anzahl wird angegeben. Jede Datei wird in einer Tabelle angezeigt. Zunächst sind die Ergebnisse von allen Dateien als "waitßu schreiben.

Alle Dateien müssen gleichzeitig ausgeführt und alle Ergebnisse als iterative aktualisiert werden.

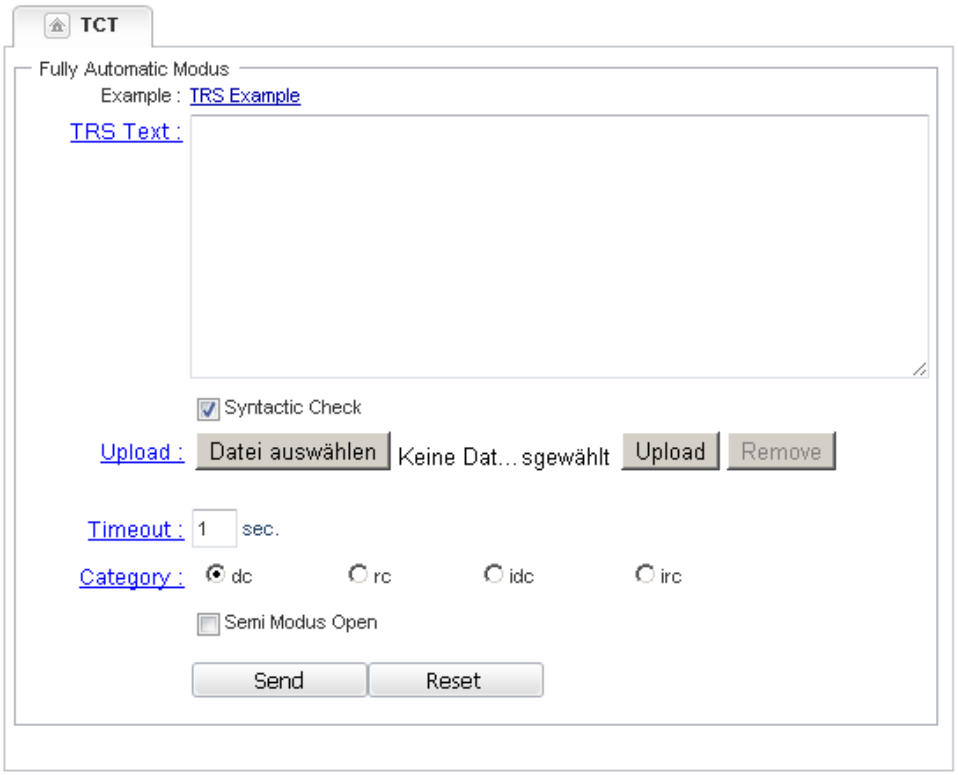
6.2.5 Dynamische Strategien

Implimentierung mit Hilfe von Code sich Kapitel 7.3

7 Verwendung

7.1 Hauptseite (Full Automatic)

Screenshot der Hauptseite:

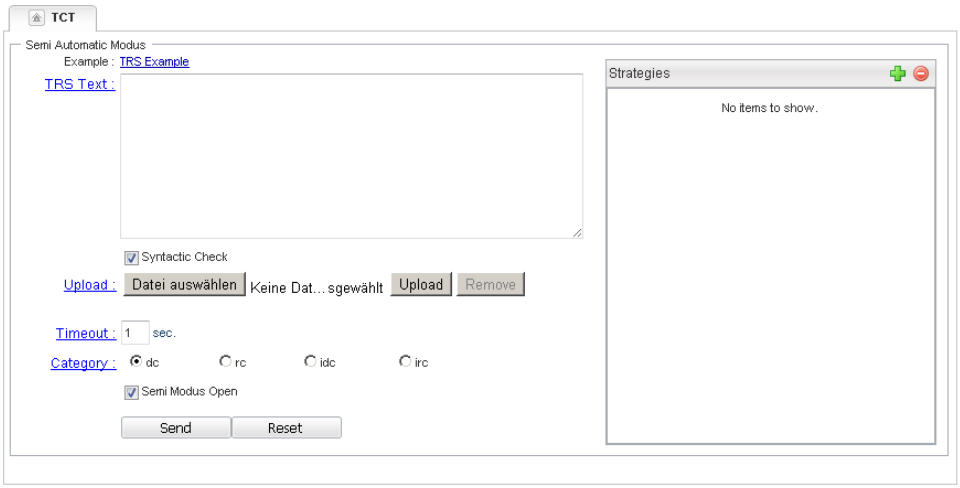


Aus dieser Bild sieht man das Hauptfenster in das man die Daten eingibt um das TRS dann an TCT zu senden und zurück erhalten. In Initalzustand sehen wir den Fully Automatic Modus und hier kann man im Text Eingabefeld ein TRS hineinschreiben, oder auch ein Archive- oder TRS-Datei hochladen, des weiteren Timeout und Kategorie(Komplexitätsmaßnahme) einstellen.

7.2 Erweiterter Modus (Semi Automatic)

Im Semi Automatic Modus erweitert sich das Fenster und ein Strategy Verwaltungsfenster wird darsgetellt. In diesem Fenster kann man beliebige Strategien erstellen, löschen, gehörige Prozesse als Baum darstellen und editieren.

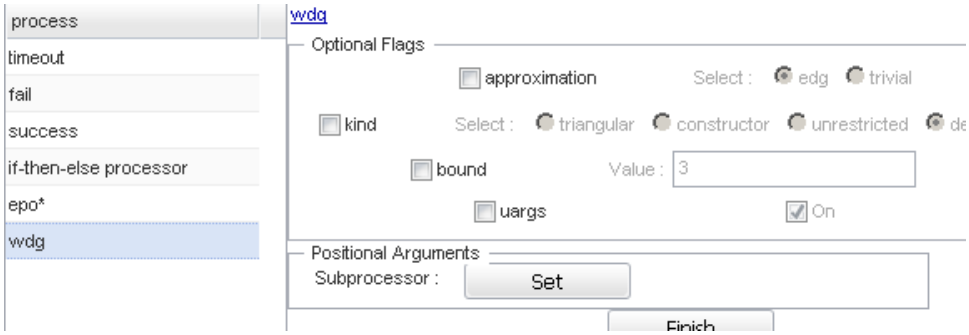
Screenshot der Semi Automatic:



Aus dieser Bild sieht man das Semi Automatic Modus und hier kann man seine spezielle Strategien erstellen, bearbeiten und löschen. Wenn der Benutzer dem "+"Button drückt, wird der Wizard Fenster eröffnet.

7.3 TCT-Info

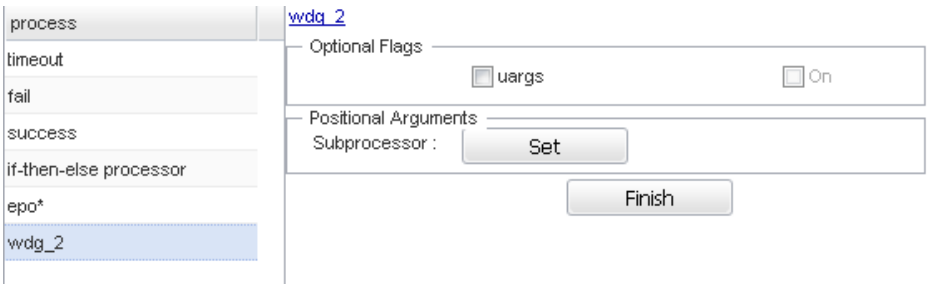
Screenshot nach aktuelle tct-info.txt:



Geänderte tct-info.txt Code

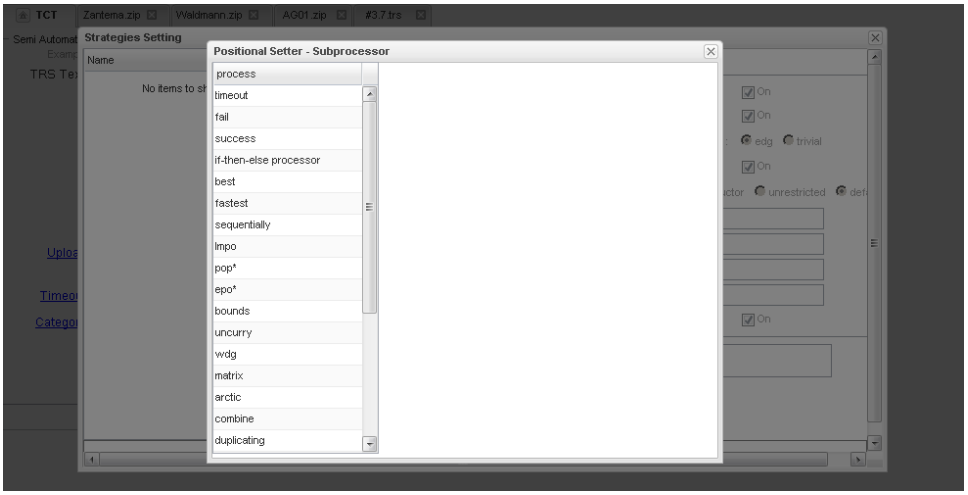
```
Strategy {name = wdg_2;
    description = [];
    synopsis = "wdg_2 #OPTIONALARGS #1";
    positionalArguments = [(1 ,
          Argument {name = "subprocessor";
                isOptional = False;
                description = "The processor .....";
                values = "<processor>";})];
          Argument {name = "uargs";
                isOptional = True;
                description = "This argument .....";
                values = "<bool>";
                default = Off\}];
}
```

Screenshot nach geänderte tct-info.txt:



7.4 Strategie rekursiv definieren

Screenshot des Wizards für rekursive Strategien:



Auf diesem Bild sieht man zwei Fenster. Das dunklere Fenster ist die Hauptseite des Wizard, und wenn gewählte Prozess ein Subprozessor haben kann, und der Benutzer Set Button klicken wird, der zweite Fenster wird starten und der Benutzer kann hier Einstellungen des Subprozessoren machen.

Zur Erinnerung: der Subprozessor kann wiederum Subprozessoren haben.

7.5 Archive Datei Ergebnisse Seite

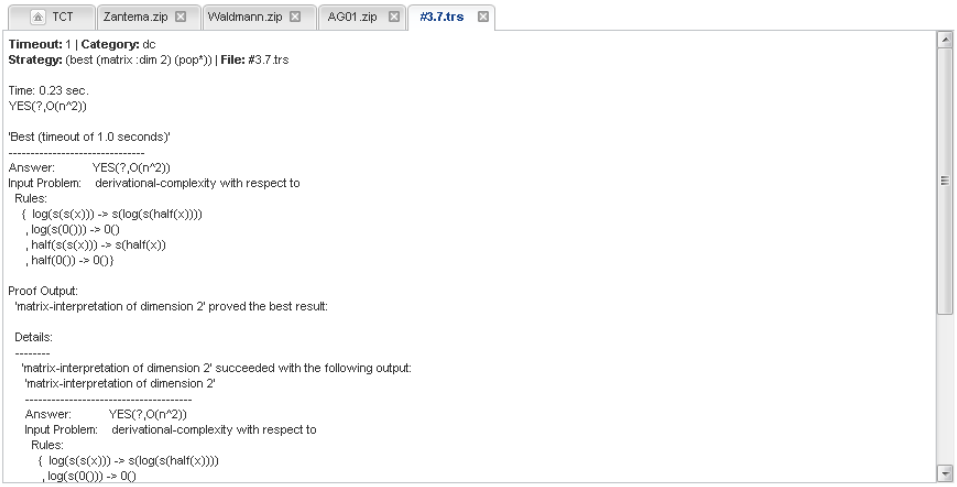
Screenshot der Archive Datei Ergebnisse Seite:

Auf diesem Bild sieht man eine Ergebnis Seite für eine Archiv datei. Auf das oberen Seite des Fensters befinden sich Informationen über Timeout, Kategorie, Strategie und Dateiname. Danach behandelt sich die Statistik (Zusammenfassung) über die Daten, womit wieviele Erfolge, Misserfolge bzw. Zeitüberschreitungen es gab. Schließlich die zeigt, für jede Datei ein einzelne Ergebnis in einer Tabelle.

Zur Erinnerung: Tabelle und Statistik werden laufend aktualiesiert.

7.6 Einzeldatei oder Text Ergebnisse Seite

Screenshot der Ergebnis Seite:



Aus diesem Bild sieht man die Detailanzeige für ein TRS. Die Ergebnis Seite wird in einen eigenen Tab angezeigt. Auf den oberen Seite des Fensters befinden sich die Informationen über Timeout, Kategorie, Strategie und Dateiname. Danach befindet sich die Ausführungszeit, das Ergebnis und schließlich den Beweis des Ergebnisses.

8 Erweiterung

Das Interface erledigt im Moment seine Aufgaben ausreichend, allerdings könnte man noch weitere Funktionen einbinden. Es könnte eine einfache Eingabesyntaxüberprüfung für die eingegebenen TRSs implimentiert werden, da bisher nur Klammerfehler überprüft werden können.

Des Weiteren könnte dafür gesorgt werden, dass, wenn das Programm tctinfo aufgerufen wird, eine direkte Strategienliste vorhanden wäre, ohne dass das File "tct-info.txt"verwendet werden müsste. Eine zusätzliche Idee wäre es, für die gespeicherte Strategien eine Datenbank einzurichten. Wenn der Benutzer später darauf zurückgreifen möchte, könnte er seine früher definierten Strategien somit problemlos wieder aufrufen.

9 Erfahrung

In der nächsten Web Generation Systeme sollen auf jeder Änderung aufbauen können, ohne sie komplett zu laden. Es gibt verschiedene Möglichkeiten für diese Funktion. Flash, Java Applet, AJAX etc.

Auf dieser Web-Seite wurde dies mit AJAX ausgeführt. AJAX braucht kein extra Plugin und funktioniert auf allen populären Rechnern und Smart Handys.

AJAX Scripts wurden mit Hilfe von Google Web Toolkit unter Java geschrieben. Mit GWT könnte man die gesamte Entwicklung von Client- und Server-AJAX-Anwendungen unter Java schreiben. Die mit dem GWT Java-Javascript Compiler erhaltenen Scripten funktioniert auf populären Servern auch ohne Einstellungen. Mit Hilfe von SmartGWT, einer Erweiterung von GWT, kann man ein spezielles Design darstellen und eine hohe Funktionalität erreichen.

Danksagung

Ich möchte mich an dieser Stelle recht herzlich bei Herren Martin Avanzini, Doz. Dr. Georg Moser und Andreas Schnabl für die Thema und ihre unkomplizierte und hilfreiche Unterstützung bei Fragen.

Des Weiteren möchte ich mich meiner Familie und meine Verlobte danken, die mir zur Seite stehen.

Literaturverzeichnis

[1] J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(3):195–220, 2008.

[2] M. Middeldorp and S. Winkler. Term rewriting, 2010. `http://cl-informatik.uibk.ac.at/teaching/ws10/trs/slides/1x2.pdf`.

[3] G. Moser. Proof theory at work: Complexity analysis of term rewrite systems. *CoRR*, abs/0907.5527, 2009. Habilitation Thesis.

[4] G. Moser. Complexity analysis of term rewrite, 2010. `http://cl-informatik.uibk.ac.at/ georg/talks/isr10/ohp/lecture1-2x2.pdf`.