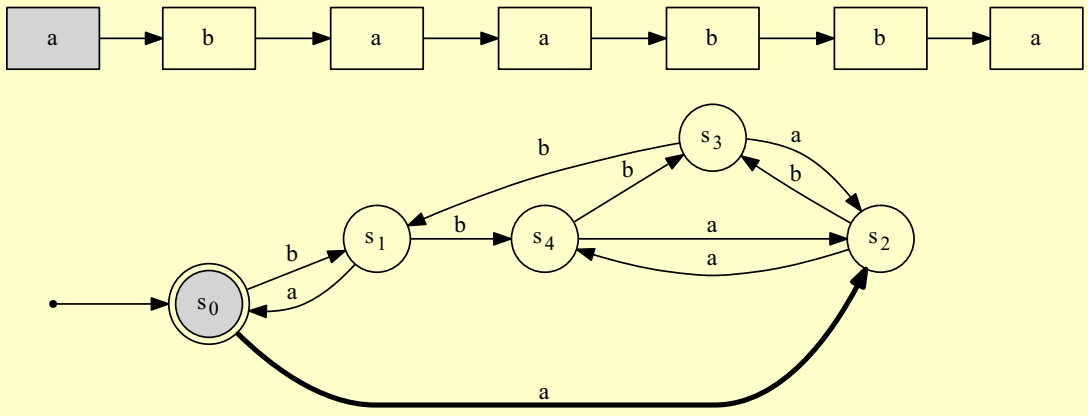
XWizard: **Das Online-Informatik-Werkzeug**

– Handbuch für Studierende –

Lukas König, Friederike Pfeiffer-Bohnen, Alexander Dorsch



Skript ID-10700

XWizard: **Das Online-Informatik-Werkzeug**

– Handbuch für Studierende –

Contents

1 Was ist XWizard?

XWizard ist ein frei verfügbares (Web-) Tool, mit dem viele Arten von Objekten aus verschiedenen Bereichen der Informatik erzeugt, bearbeitet und für Präsentationen, Veröffentlichungen usw. im PDF-Format aufbereitet werden können. Zu diesen Objekten gehören beispielsweise Turingmaschinen, Kellerautomaten, endliche Automaten, Chomsky-Grammatiken uvm. Zum Bearbeiten dieser Objekte bietet der XWizard eine Vielzahl von Algorithmen (deren Anzahl künftig weiter steigen wird). Die XWizard-PDF-Ausgaben sind meist intuitiv und übersichtlich, können aber auch individuell angepasst werden. Das Tool ist sowohl für den studentischen Gebrauch geeignet als auch (und besonders!) auf die typischen Arbeiten zugeschnitten, die bei Lehrpersonen anfallen, etwa das Erstellen von Aufgaben (das X in XWizard steht für "eXercise" – und auch für "beliebiges").

XWizard kann eine große Anzahl verschiedener Objekttypen erzeugen. Um einen Überblick über die Mächtigkeit zu erhalten sind im folgenden Abschnitt die wichtigsten Objekttypen jeweils mit Hauptmerkmalen aufgelistet. Um den Umgang mit diesen intuitiv zu erlernen, können sie aufgerufen, bearbeitet und verändert werden. An den XWizard-Grundlagen interessierte Leser können aber auch gleich zu Abschnitt 3 springen.

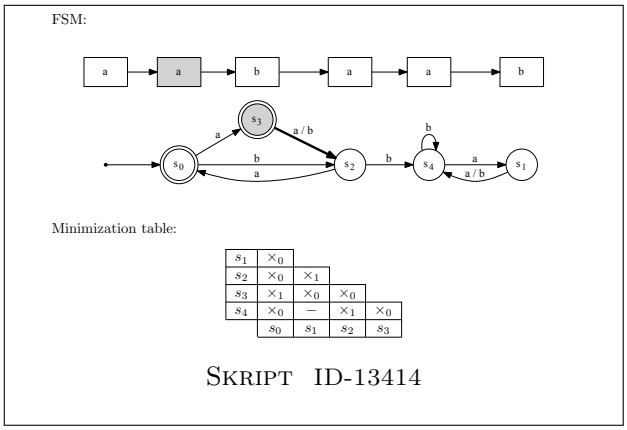
1.1 Native Objekttypen

Die grundlegenden XWizard Objekttypen können einer der zwei Gruppen "Theoretische Informatik" oder "Technische Informatik" zugeordnet werden (Man muss sich allerdings bewusst sein, dass nicht alle einwandfrei nur zu einer Gruppe zuzuordnen sind; in Zukunft können sich diese Gruppen auch verändern). Die folgende Liste folgt ebendieser Kategorisierung. Unter dem Namen jedes Objekttyps sind die Hauptmerkmale aufgeführt.

Unter den Abbildungen befinden sich Verlinkungen zu den zugehörigen Objekten auf der XWizard-Webseite, mit welchen man beliebig experimentieren kann, um sich informell die grundlegenden Ideen anzueignen.

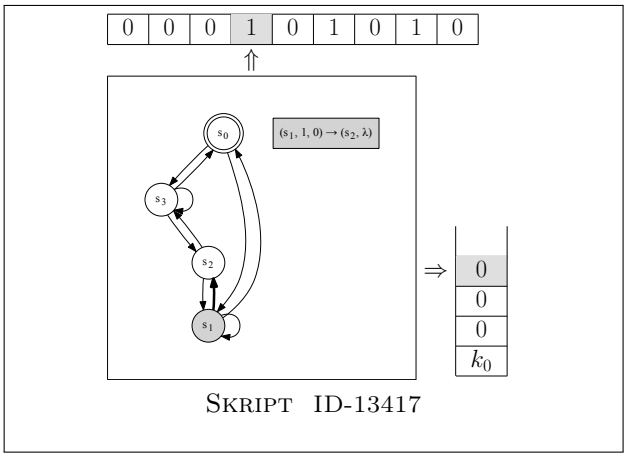
Theoretische Informatik

- **Endliche Automaten (EAs):**
  - Selbst-definierte oder zufällige EAs
  - Schrittweise Simulation (det. oder nicht-det.) von frei wählbaren Eingabewörtern
  - Minimierung
  - Erzeugung von äquivalenten deterministischen EAs
  - Konversion in äquivalente
    - Kellerautomaten
    - Turingmaschinen
    - Chomsky-Grammatiken
    - Reguläre Ausdrücke
  - ...



- **Kellerautomaten (KAs):**
  - Vorimplementierte Beispiele
  - Beliebig selbst-definierte KAs
  - Schrittweise Simulation (det. oder nicht-det.) von selbstgewählten Eingabewörtern
  - Anzeige mehrerer Berechnungsabläufe gleichzeitig
  - Konversion in äquivalente
    - Chomsky-Grammatiken
    - Turingmaschinen

    in Bearbeitung
  - ...



Skript ID-13417

- **Turingmaschinen (TMs):**
  - Vorimplementierte Beispiele
  - Beliebige selbst-definierte oder zufällige TMs
  - Simulation (det. oder nicht-det.) von beliebig gewählten Eingabewörtern
  - Anzeige mehrerer Berechnungsabläufe gleichzeitig
  - Konversion in äquivalente Chomsky-Grammatiken

    in Bearbeitung
  - Schrittweise Simulation

    in Bearbeitung
  - ...

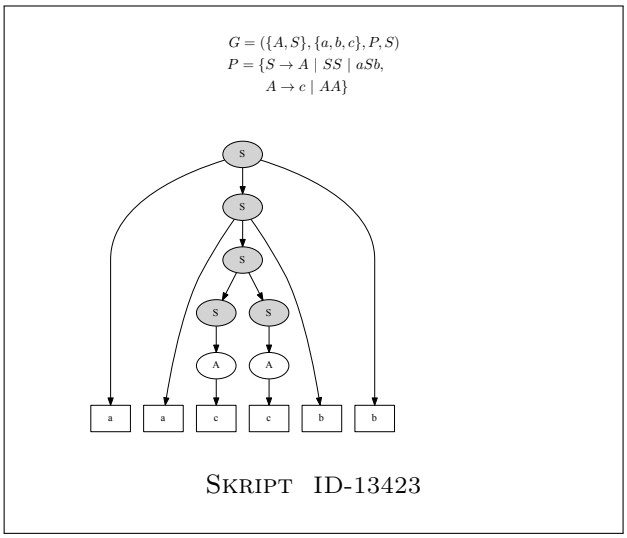
	1	$\star$
A	$(B, 1, L)$	$(B, 1, R)$
B	$(C, \star, L)$	$(A, 1, L)$
C	$(D, 1, L)$	$(H, 1, R)$
D	$(A, \star, R)$	$(D, 1, R)$
H		

Tape	Transition
$\hat{\star}$	$(A, \star) \to (B, 1, R)$
$1\hat{\star}$	$(B, \star) \to (A, 1, L)$
$\hat{1}1$	$(A, 1) \to (B, 1, L)$
$\hat{\star}11$	$(B, \star) \to (A, 1, L)$
$\hat{\star}111$	$(A, \star) \to (B, 1, R)$
$1\hat{1}11$	$(B, 1) \to (C, \star, L)$
$\hat{1} \star 11$	$(C, 1) \to (D, 1, L)$
$\hat{\star}1 \star 11$	$(D, \star) \to (D, 1, R)$
$1\hat{1} \star 11$	$(D, 1) \to (A, \star, R)$
$1 \star \hat{\star}11$	$(A, \star) \to (B, 1, R)$
$1 \star 1\hat{1}1$	$(B, 1) \to (C, \star, L)$
$1 \star \hat{1} \star 1$	$(C, 1) \to (D, 1, L)$
$1\hat{\star}1 \star 1$	$[(D, \star) \to (D, 1, R)]$

Skript ID-16304

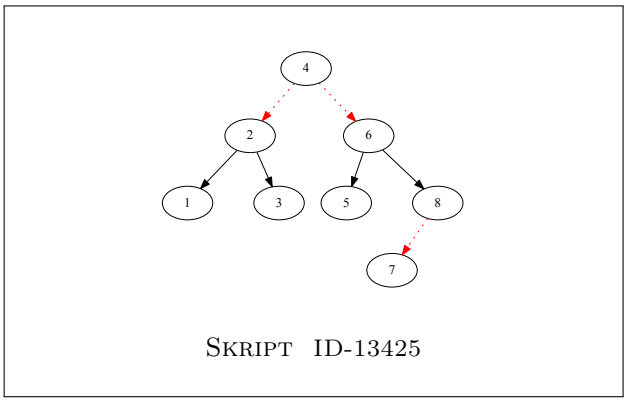
- **Chomsky Grammatiken:**
  - Vorimplementierte Beispiele
  - Beliebige selbst-definierte oder zufällige Grammatiken
  - Syntaxbaumvisualisierung
  - Ableitung einzelner Wörter
  - Baum mit allen ableitbaren Wörtern, beschränkt durch Ableitung oder Wortlänge
  - Konversion in äquivalente
    - ϵ-freie Grammatiken
    - Chomsky Normalform
    - Greibach Normalform
    - Kuroda Normalform
    - Kellerautomaten
  - ...

  (Die meisten Algorithmen sind nur auf Grammatiken des Typs 3 / Typs-2 (oder Typ-1: Kuroda NF) anwendbar.)



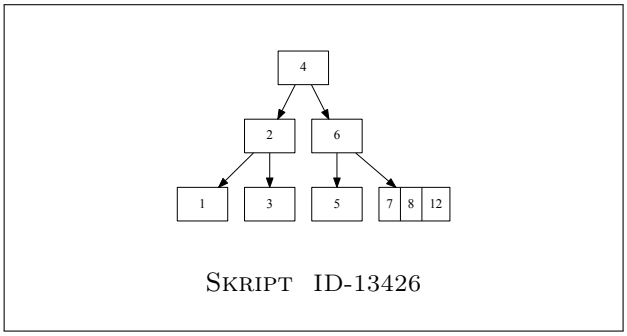
SKRIPT ID-13423

- **Rot-Schwarz-Bäume:**
  - Vorimplementierte Beispiele
  - Zufällige Rot-Schwarz Bäume gegeben durch
    - Einfügungsreihenfolge oder
    - explizite Baumdefinition
  - Konversion in äquivalente
    - 2-3-4 Bäume
  - Basierend auf Strings oder Zahlen



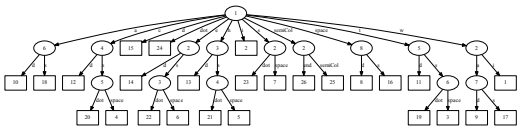
SKRIPT ID-13425

- **2-3-4 Bäume:**
  - Vorimplementierte Beispiele
  - Zufällige 2-3-4 Bäumge gegeben durch
    - Einfügungsreihenfolge oder
    - explizite Baumdefinition
  - Konversion in äquivalente
    - 2-3-4 Bäume
  - Basierend auf Strings oder Zahlen



SKRIPT ID-13426

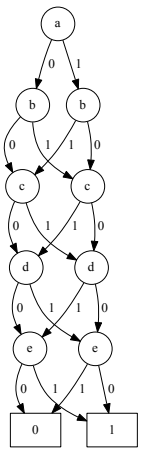
- **PAT Trees:**
  - Vorimplementierte Beispiele
  - Zufällige PAT Trees durch Textbasis gegeben
  - Bis jetzt nur Visualisierungen



Skript ID-13428

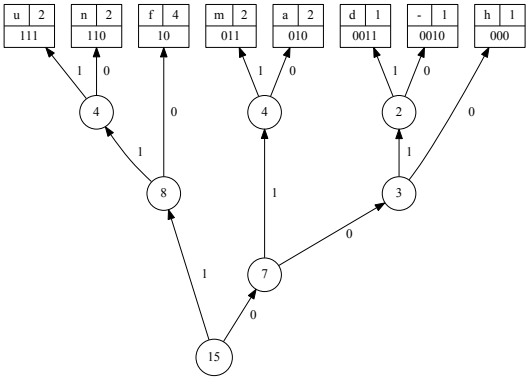
Technische Informatik

- **Binary Decision Diagrams (BDDs):**
  - Vorimplementierte Beispiele
  - Berechnung von BDDs von zufälligen bool'schen Funktionen
  - Visualisierung von BDDs
  - Visualisierung der schrittweisen Herleitung von BDDs



Skript ID-62

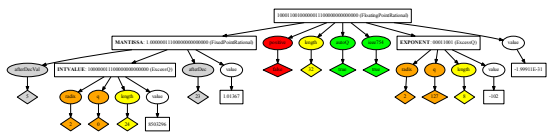
- **Huffmankodierung:**
  - Vorimplementierte Beispiele
  - Berechnung von Huffmanbäumen für zufällige Symbolverteilungen
  - Zwei verschiedene Visualisierungsarten (bottom-up und top-down)



Skript ID-72

- **Zahldarstellungen:**
  - ExcessQ Darstellung
  - Fixpunktdarstellung
  - Gleitpunktdarstellung (inc. IEEE754)
  - Einerkomplementdarstellung
  - Zweierkomlementdarstellung
  - Umrechnung zwischen diesen Zahldarstellungen und der Dezimaldarstellung
  - Zufällige Basen von Binär bis 36.

$1\ 00011001\ 00000011100000000000000_{ieee754} = -1.99911E-31$



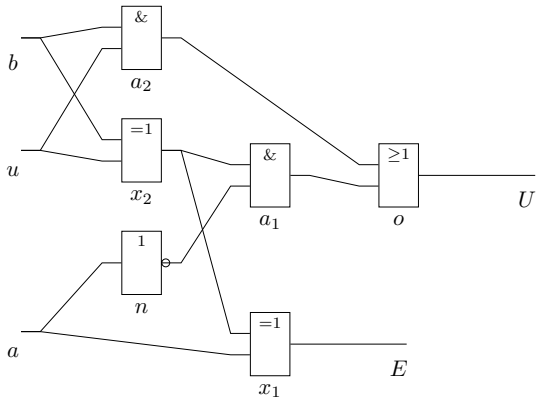
Skript ID-15538

Die nachfolgend beschriebenen Objekttypen befinden sich noch im Aufbau und haben daher nur wenig Funktionalität:

- **Logische Schaltungen:**

  in Bearbeitung

- (Bis jetzt können logische Schaltungen nur angezeigt werden.)



Skript ID-8861

- **Reguläre Ausdrücke:**

  in Bearbeitung (Alles, was reguläre Ausdrücke bis jetzt können, ist, einige Wörter der durch sie definierten Sprache aufzulisten, siehe Beispiel.)

$(a+b)^\star c + \emptyset^\star$

First 16 words: $\lambda, c, ac, bc,$
$aac, abc, bac, bbc, aaac, aabc, abac, abbc, baac, babc,$
$bbac, bbbc, \ldots$

Skript ID-14238

Der sehr spezielle Objekttyp “MARB” (zum Beispiel Skript ID-1847) und der “Taschenrechner” (Skript ID-235) sowie der Meta-Typ `MetaProperties` (Skript ID-15847) werden hier nicht aufgeführt. Eine Kurzhilfe für diese wird auf der XWizard-Webseite gegeben. Der `MetaProperties`-Typ kann verwendet werden, um eine Übersicht über die Objekttypen zu erzeugen, siehe Seite 12. Außerdem wird XWizard kontinuierlich weiterentwickelt, was bedeutet, dass sowohl neue Objekttypen als auch Verbesserungen der bereits verfügbaren Objekttypen jederzeit möglich sind.

1.2 PDF-Generatortypen

Neben den vorgefertigten Objekttypen kann XWizard beliebigen LaTeX- und Graphviz-Code bearbeiten. Diese Besonderheit wurde ursprünglich aus rein technischen Gründen[1] implementiert. Einmal verfügbar, eröffnete sie die Möglichkeit, komplexe Objekte, die weit über die ursprünglichen XWizard-Typen hinausgehen, zu erzeugen. Eine detaillierte Beschreibung dieser Möglichkeiten, geht über diese auf

[1] Da alle XWizard-Objekte mit PDF-LaTeX und/oder dem Graphviz DOT-Prozessor erstellt werden (oder GNUPlot und JavaPDF in alten Versionen der Downloadversion VFP, siehe unten).

den “Durchschnittsnutzer” ausgerichteten Dokumentation hinaus. Für weitere Details zur Anpassung der XWizardAusgabe, zum Erzeugen komplexer Objekte, die Merkmale mehrerer Basistypen vereinen, und der Erstellung von “Miniaufgaben” für Studierende (vergleiche Abschnitt 4), finden sich im Handbuch “XWizard für Unterrichtende”.

Als erfreulicher Nebeneffekt können Studierende, die mit LaTeX oder Graphviz nicht vertraut sind, XWizard als einfaches Lerntool verwenden, das beliebige LaTeX-Dokumente kompilieren kann und keine Installation schwergewichtiger Programme auf einem eigenen Rechner benötigt.

2 Zugriff auf XWizard

Auf XWizard kann einfach per

`www.xwizard.de`

zugegriffen werden, oder aber, indem man die Skript-Verlinkungen anklickt (oder scannt), wie zum Beispiel:

Skript ID-10700

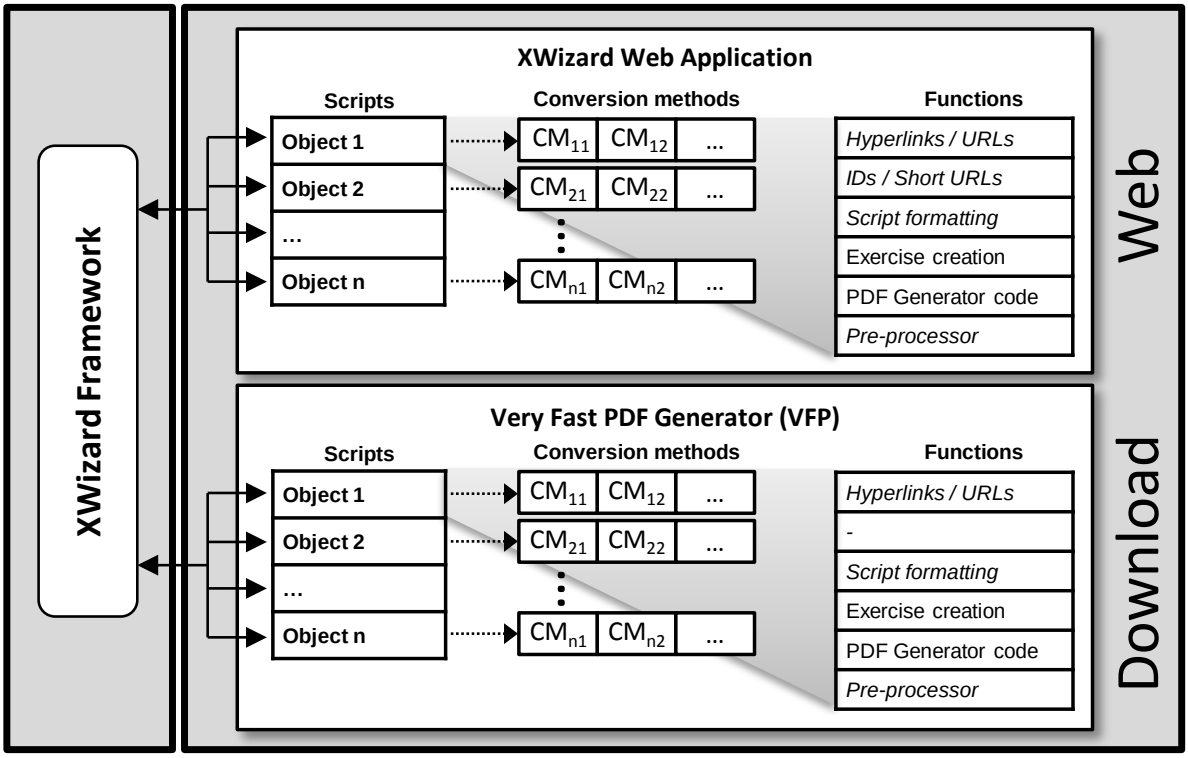
Zusätzlich zu der Webversion gibt es eine Downloadversion mit dem Namen **Very Fast PDF Generator (VFP)**. Diese kann (mitsamt Installationsinstruktionen) über den folgenden Link bezogen werden:

`https://sourceforge.net/projects/xwiz`

Beide Versionen sind weitgehend identisch. Deshalb wird der Ausdruck XWizard im Folgenden für beide Versionen verwendet, es sei denn, es wird speziell auf eine der beiden verwiesen. So wie zum Beispiel im Folgenden:

In XWizard kann man zwischen Englisch und Deutsch als Sprache wählen; bis jetzt ist VFP dagegen nur in Englisch verfügbar.

Die folgende Abbildung zeigt den Aufbau der XWizardKomponenten. Der XWizardWorkflow, basierend auf Skripten, Konversionsmethoden etc. wird im nächsten Abschnitt beschrieben.



XWizard und alle dazu gehörenden Implementierungen sind *freie Software*[2]. Sie dürfen für beliebige Zwecke verwendet werden (kommerzielle ausgenommen), und der Quellcode darf studiert, verändert und weiter verbreitet werden, solange das im Einklang mit den unter folgendem Link abrufbaren Rechtsbelehrungen passiert: `http://www.xwizard.de:8080/Wizz?impressum&lang=ger`.

3 Arbeiten mit XWizard

Es gibt zwei Hauptmechanismen, die man bei der Benutzung von XWizard verwenden kann: **Skripte**, die ein spezifisches Objekt definieren, und **Konversionsmethoden**, die einen Mechanismus zum Anwenden von Algorithmen auf ein Objekt bieten (zum Beispiel; Minimierung eines endlichen Automaten). Genauer gesagt bilden Skripte den universellen Rahmen, um XWizard zu kontrollieren. Das bedeutet, dass im Prinzip jegliche Funktionalität durch das Schreiben von Skripten ausgeschöpft werden kann. Konversionsmethoden werden als Spezialfall in diesem Rahmen implementiert. Sie bieten eine einfache graphische Schnittstelle für den Benutzer und sind nicht mit dem Schreiben von Skripten verbunden. Viele typische Anwendungen von XWizard können mithilfe von Konversionsmethoden verwendet werden, mit Skripten kann man alle abdecken.

3.1 Skripte

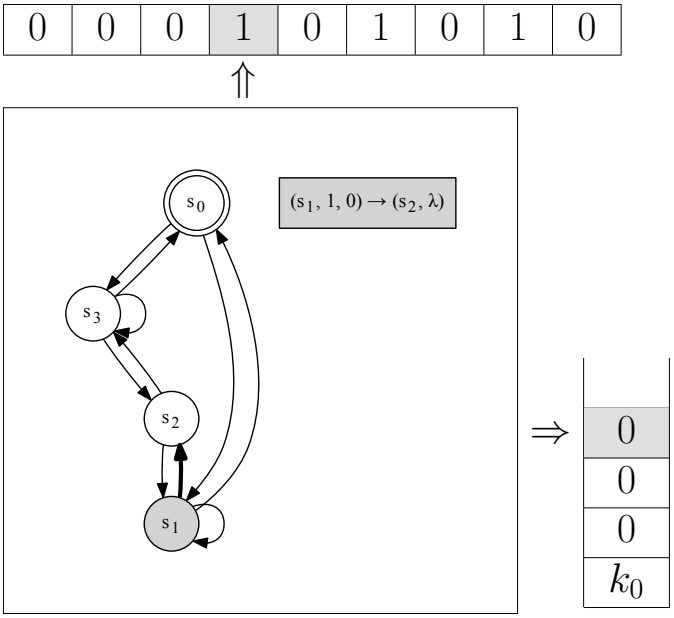
Die grundlegende Funktionsweise des XWizards beruht einfach darauf, ein Skript einzulesen, dieses in ein PDF-Bild zu übersetzen und anzuzeigen. Ein Skript besteht üblicherweise aus drei Teilen (vier, falls man die Ausführung einer Konversionsfunktion mitzählt, siehe unten), wie im folgenden Beispiel eines Kellerautomaten aufgeführt:

[2] `https://en.wikipedia.org/wiki/Free_software`

```
pda:                                                          ⇐ Skriptpräambel
    (s0, 0, k) => (s1, 0k);
    (s0, 1, k) => (s3, 1k);
    (s1, 0, 0) => (s1, 00);
    (s1, 1, 0) => (s2, lambda);
    (s1, lambda, k) | (s3, lambda, k) => (s0, k);
    (s2, lambda, 0) => (s1, lambda);                          } Hauptteil des Skripts
    (s2, lambda, k) => (s3, bk);
    (s3, 0, 1) => (s3, b);
    (s3, 0, b) => (s3, lambda);
    (s3, 1, 1) => (s3, 11);
    (s3, 1, b) => (s3, b1);
--declarations--
    e=#n#;
    s0=s0;
    F=s0;
    kSymb=k;                                                  } Variablendefinitionen
    inputs=000101010;
    simSteps=3;
--declarations-end--
```

Alle Skripte enthalten die folgenden drei Teile: Eine **Präambel** legt den Typ des Objekts, der durch das Skript definiert wird, fest; der **Hauptteil** definiert die eigentliche Struktur des Objekts; Variablen im Teil **Variablendefinitionen** können dazu genutzt werden, weitere Eigenschaften festzulegen (sowohl der komplette Variablendeklarationsteil, als auch einzelne Variablen können weggelassen werden. In diesem Fall werden die entsprechenden Variablen auf Standartwerte gesetzt.)

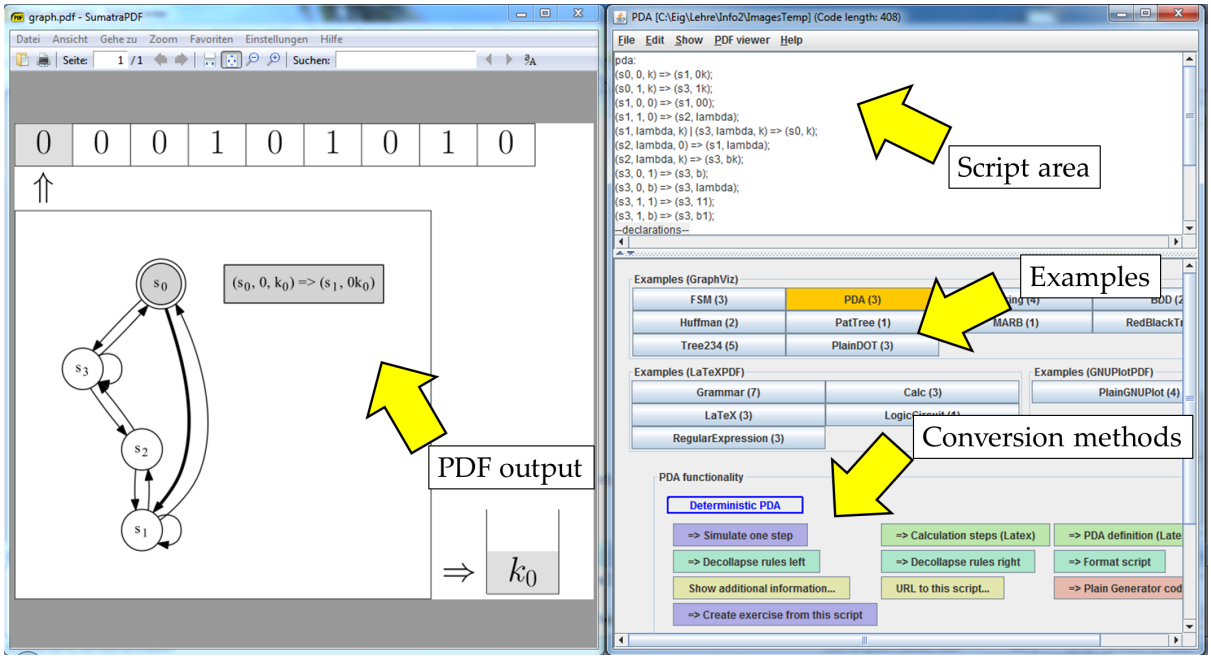
Dieses Beispielskript erzeugt die folgende Abbildung (im Hintergrund werden Graphviz und LaTeX verwendet):



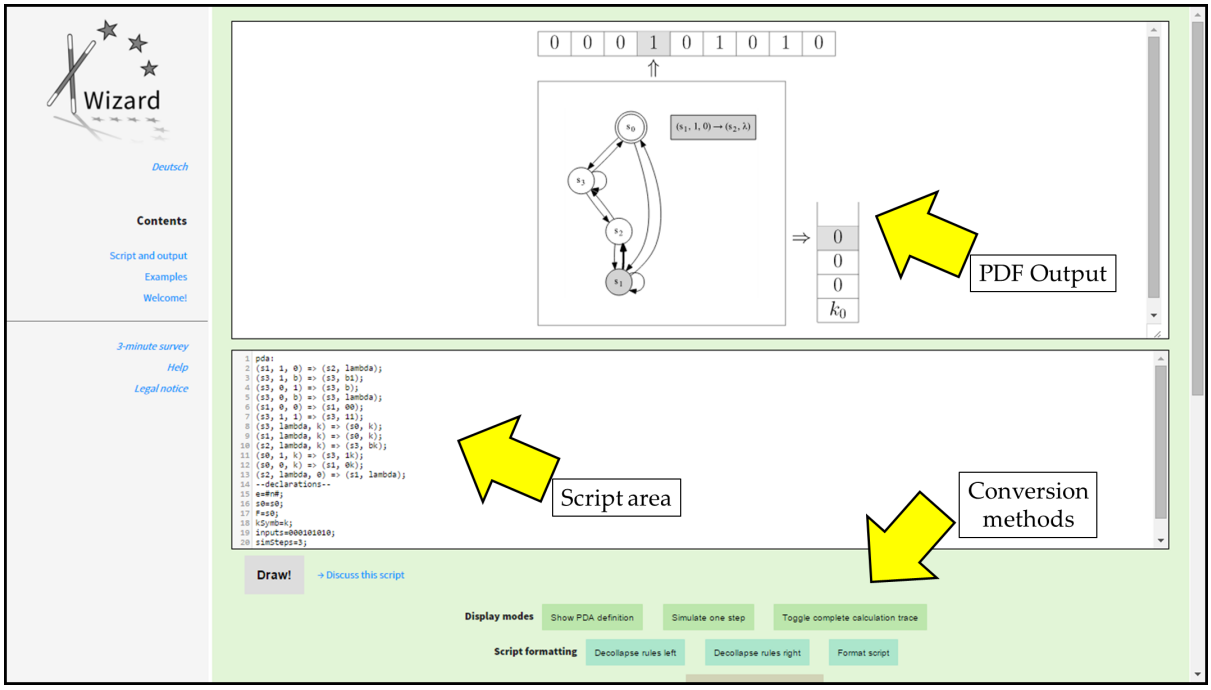
Skript ID-13417

Das Skript kann sowohl in VFP als auch in XWizard eingefügt werden, um die in den Screenshots angezeigte Ausgabe zu erzeugen (Klicken auf den Link oder Scannen des QR-Codes leiten weiter zu XWizard und führen das Skript automatisch aus).

VFP:



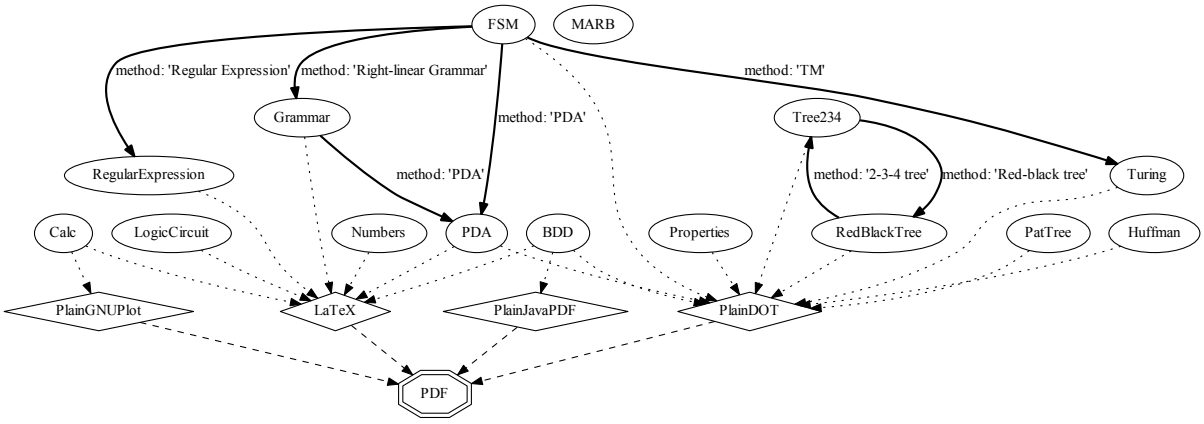
XWizard:



Um die Berechnung des angezeigten Kellerautomaten zu simulieren, kann man die "simSteps"-Variable im Skript erhöhen; ebenso kann man die **Konversionsmethode** anklicken (vergleiche dazu auch die Beschreibung von Konversionsmethoden weiter unten).

Bei der Verwendung von VFP wird die PDF-Ausgabe automatisch mit Veränderung des Skripts aktualisiert. Bei XWizard wird die Bildausgabe generiert, nachdem der "Draw!"-Knopf auf der Webseite geklickt wurde. Eine PDF kann durch Drücken des "Download PDF"-Links abgerufen werden; dieser erscheint beim Scrollen **unter** die PDF Ausgabe.

Einen Überblick über die verfügbaren Skripttypen zum Zeitpunkt des Verfassens dieses Dokuments ist durch das folgende Diagramm dargestellt(durchgezogene und gepunktete Pfeile bezeichnen Konversionen zwischen verschiedenen Skripttypen durch Konversionsmethoden; gestrichelte Pfeile bezeichnen den PDF-Generierungsprozess; rautenförmige Knoten repräsentieren die direkten PDF-erzeugende Skripttypen – hierbei handelt es sich vor allem um Graphviz und LaTeX (GNUPlot und JavaPDF sind außer Gebrauch, siehe unten).



Eine aktuelle Version der Abbildung erhält man über den folgenden Link (beachten Sie, dass sie selbst über ein einfaches XWizard Skript generiert ist):

Skript ID-15812

Für alle dieser Objekttypen gibt es Beispielskripte auf der XWizard Webseite:

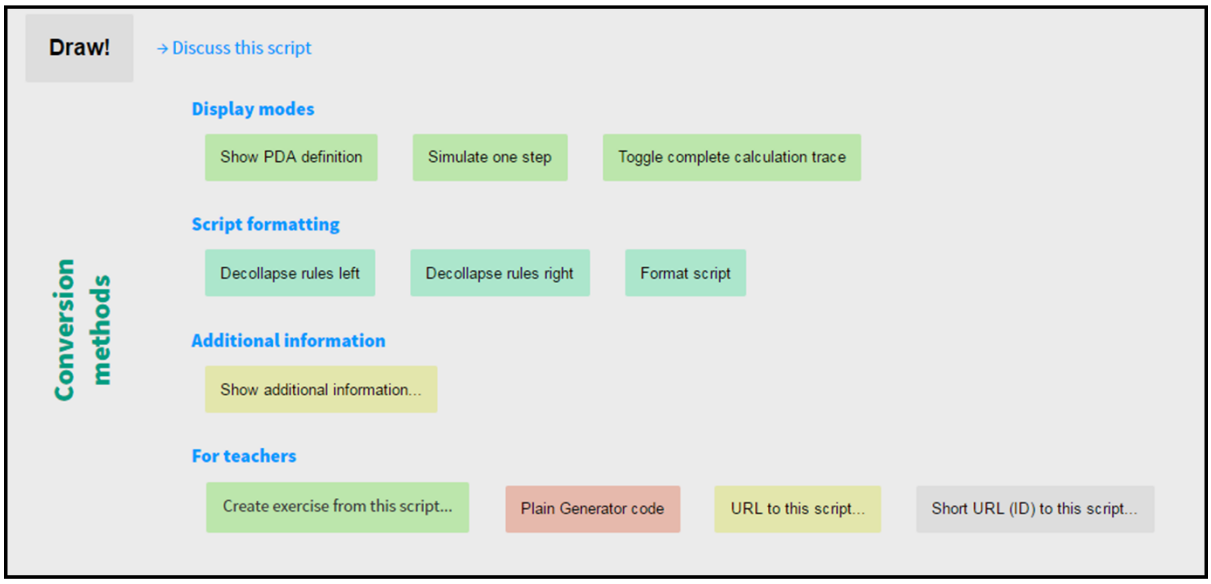
`http://www.xwizard.de:8080/Wizz?lang=eng&hide#Examples`

3.2 Konversionsmethoden

Neben dem Erzeugen und Anzeigen von Objekten, ist eine der Hauptfunktionen von XWizard das Anwenden von Algorithmen auf Objekte, zum Beispiel die Visualisierung der schrittweisen Simulation eines Kellerautomaten oder der Minimierung eines endlichen Automaten. Algorithmen, werden durch Konversionsmethoden auf Objekte angewendet. Das bedeutet technisch, dass ein Skript in ein anderes transformiert wird.

Wenn man eine Konversionsmethode anwendet, so wird das zu dem aktuellen Objekt gehörende Skript ersetzt durch das neue Skript, das von der Konversionsmethode erzeugt wurde, und das Objekt, das von dem neuen Skript definiert wird, wird generiert und angezeigt.

Der einfachste Weg, eine Konversionsmethode auf ein Skript anzuwenden, ist, den entsprechenden Knopf im "Konversionsmethoden"-Bereich anzuklicken. Der folgende Bildschirmausschnitt zeigt beispielhaft die verfügbaren Konversionsmethoden für Skripte eines Kellerautomaten in XWizard:



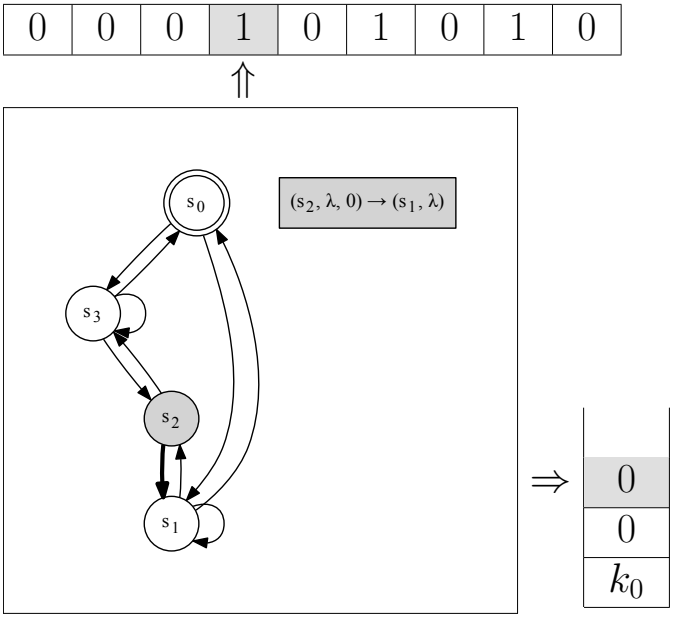
Wenn man zum Beispiel den Button "Simuliere einen Schritt" betätigt, so wird das Kellerautomaten-Skript in ein sehr ähnliches Skript übersetzt, wobei lediglich der Displaymodus zur schrittweisen Visualisierung gewechselt wurde; bei wiederholtem Klicken wird die `simSteps` Variable erhöht, wodurch der Kellerautomat einen Zeitschritt weiter simuliert wird.

Allgemein können die Konversionsmethoden in der Kategorie "Anzeigemodi" dazu benutzt werden, zwischen der schrittweisen Darstellung und der Komplettansicht, einer Visualisierung der Kellerautomatendefinition sowie Kombinationen dieser Ansichten (vergleiche die Auflistung der Kellerautomaten-Features in Abschnitt 1.1), zu wechseln.

Das durch die Konversionsmethode erzeugte zugehörige Skript lautet wie folgt:

```
pda:
    (s1, 1, 0) => (s2, lambda);
    (s3, 1, b) => (s3, b1);
    (s3, 0, 1) => (s3, b);
    (s3, 0, b) => (s3, lambda);
    (s1, 0, 0) => (s1, 00);
    (s3, 1, 1) => (s3, 11);
    (s3, lambda, k) => (s0, k);
    (s1, lambda, k) => (s0, k);
    (s2, lambda, k) => (s3, bk);
    (s0, 1, k) => (s3, 1k);
    (s0, 0, k) => (s1, 0k);
    (s2, lambda, 0) => (s1, lambda);
--declarations--
    e=#n#;
    s0=s0;
    F=s0;
    kSymb=k;
    inputs=000101010;
    simSteps=4;              /* Value increased by conversion method. */
    maxNondetCalcDepth=12
--declarations-end--
```

Das daraus hervorgehende Objekt sieht wie folgt aus:



Skript ID-C22055

Klicken Sie auf den QR Code und probieren Sie es aus! (Im Online-Skript ist zusätzlich eine Animationsanweisung definiert, die bei Klicken in die Grafik den Effekt dieser Methode zeigt, ohne dass der Button angeklickt werden muss. Animationen und andere weiterführende Eigenschaften werden im "XWizard-Handbuch für Lehrer" im Detail erläutert.)

3.3 Anwenden von Konversionsmethoden per Skript

Das Anwenden von Konversionsmethoden per Skript ist ein weiterführendes technisches Detail, das nur für fortgeschrittene Nutzer von Interesse ist. Daher kann der folgende Abschnitt von den meisten Lesern übersprungen werden.

Wie oben bereits erwähnt läuft XWizard komplett über Skripte; das bedeutet insbesondere, dass sogar die Anwendung einer Konversionsmethode über einen besonderen Skriptaufruf angestoßen werden kann. Skripte, die diesen Befehl enthalten, werden **conversion scripts** (deutsch: Konversionsskripte) genannt und sie beginnen wie üblich mit den drei Teilen, die das Skript festlegen, der umgewandelt werden soll. Als vierten Teil wird ein **conversion command** (deutsch: Konversionsbefehl) in die letzte Zeile des Skripts geschrieben. Dieser sieht wie folgt aus:

```
**>CM-NAME<**
```

dabei ist `CM-NAME` der englische Name der anzuwendenden Konversionsmethode. Für den Fall, dass die Konversionsmethode Parameter erfordert lautet der Befehl wie folgt:

```
**>CM-NAME[p1, p2, ...]<**
```

dabei sind `p1, p2, ...` die Methodenparameter – diese können in Anführungszeichen gesetzt werden, falls Sonderzeichen, wie beispielsweise Leerzeichen oder Kommata, benötigt werden:

```
["p, a, r, 1", "p, a, r, 2", ...]
```

Die "Simuliere einen Schritt"-Methode kann beispielsweise auf ein Kellerautomaten-Skript angewendet werden, indem man die rot-gefärbte letzte Zeile hinzufügt:

```
pda:
    (s1, 1, 0) => (s2, lambda);
    (s3, 1, b) => (s3, b1);
    (s3, 0, 1) => (s3, b);
    (s3, 0, b) => (s3, lambda);
    (s1, 0, 0) => (s1, 00);
    (s3, 1, 1) => (s3, 11);
    (s3, lambda, k) => (s0, k);
    (s1, lambda, k) => (s0, k);
    (s2, lambda, k) => (s3, bk);
    (s0, 1, k) => (s3, 1k);
    (s0, 0, k) => (s1, 0k);
    (s2, lambda, 0) => (s1, lambda);
--declarations--
    e=#n#;
    s0=s0;
    F=s0;
    kSymb=k;
    inputs=000101010;
    simSteps=3;
    maxNondetCalcDepth=12
--declarations-end--
**>Simulate one step<**   /* This is a conversion command. */
```

Wenn dieses Skript eingegeben wird, so ist das Ergebnis exakt dasselbe, wie für den Fall, dass man das entsprechende Feld zur Konversion des Skriptes angeklickt hat (ohne den Konversionsbefehl).

3.4 Syntax und Funktionen der Basisskripttypen

XWizard wird ständig weiterentwickelt, daher wäre es unpraktisch, alle Skripttypen und Funktionen in diesem eher statischen Handbuch zu beschreiben. Im Gegensatz dazu enthält die XWizard-Webseite eine umfassende Hilfeseite (in Englisch und Deutsch) mit Beschreibungen von allen wichtigen Skripttypen, Konversionsmethoden und zusätzlichen Funktionen. Beachten Sie deshalb für aktuelle Beschreibungen der meisten XWizard-Funktionen bitte die folgenden Hilfeseiten:

`http://www.xwizard.de:8080/Wizz?help&lang=eng` *(englische Hilfeseite)*
`http://www.xwizard.de:8080/Wizz?help&lang=ger` *(deutsche Hilfeseite)*

3.5 URLs und Kurz-URLs zu XWizard-Skripten

XWizard-Skripte können über URLs sehr einfach mit anderen Nutzern ausgetauscht werden. Dabei können entweder **normale URLs** oder **Kurz-URLs** über die folgenden Knöpfe verwendet werden:

(1) "Erstelle URL zu diesem Skript" oder

(2) "Erstelle kurze URL (ID) zu diesem Skript"

(Anzumerken ist, dass (2) keine Konversionsmethode ist und folglich nicht in VFP verfügbar ist.) Falls man eine normale URL (1) anlegt, so wird das Skript in eine URL transformiert, was bedeutet, dass der komplette Skriptstring in der URL kodiert wird. Im Gegensatz dazu wird beim Anlegen der Kurz-URL (2) das Skript in einer Datenbank abgelegt und mit einer Nummer versehen, der Skript-**ID**. Die erzeugte URL beruht im Wesentlichen nur noch auf der ID und ist deshalb normalerweise wesentlich kürzer:

(1) Beispiel einer normalen URL zu einem BDD-Skript:

`http://www.xwizard.de:8080/Wizz?template=bdd%3A+a%2Cb%2Cc%2Cd%2Ce%3A+01101001100101101001011001101001`

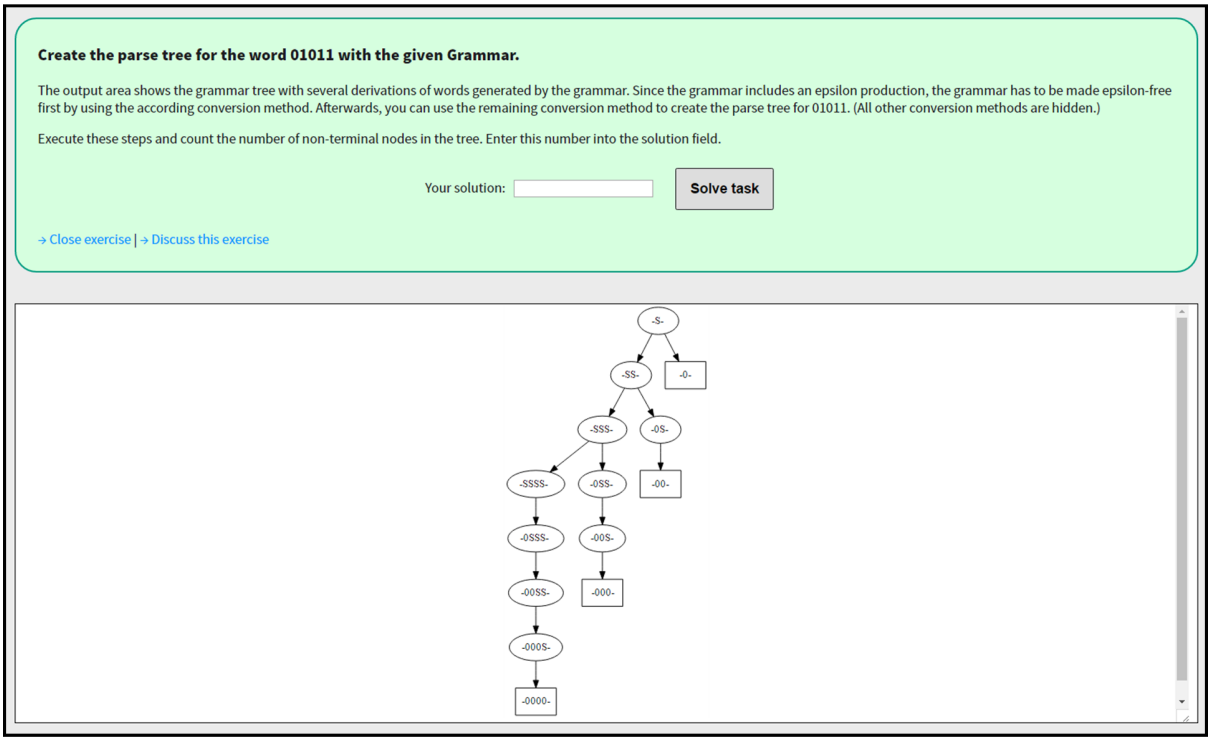
(2) Beispiel einer Kurz-URL zum selben Skript:

`http://www.xwizard.de:8080/Wizz?template=ID-16130`

Beide URL-Typen sind prinzipiell äquivalent, was die Funktionalität angeht. Allerdings können normale URLs so lang werden, dass manche Browser diese nicht akzeptieren; außerdem können Anti-Malware-Programme aufgrund der unüblichen URL-Form Alarm schlagen. Folglich sind in vielen Fällen Kurz-URLs zu bevorzugen. Nichtsdestotrotz muss angemerkt werden, dass Kurz-URLs von der XWizard-Datenbank abhängen, wohingegen normale URLs als **Stand-alone** funktionieren. Die Datenbank wird natürlich gewartet und archiviert, allerdings kann eine normale URL von diesem Standpunkt aus als ein wenig verlässlicher betrachtet werden als eine Kurz-URL.

4 Lösen von Aufgaben mit XWizard

Neben dem bis jetzt beschriebenen **Standardmodus** kann XWizard auch in einem **Aufgabenmodus** betrieben werden. Der Aufgabenmodus wird aktiviert, wenn ein Skript die Kodierung einer "Aufgabe" in seinem Variablendefinitions-Teil enthält (siehe folgendes Beispiel). Eine Aufgabe ist dabei eine Frage, die etwa von einer Lehrperson einer Gruppe von Studierenden gestellt wird und die die Studierenden lösen sollen. Die Lehrperson kann den Studierenden einen Link zu der Aufgabe zukommen lassen, genau so, wie das auch bei normalen Skripten der Fall war. Im Aufgabenmodus wird die zu lösende Frage über dem Skriptbereich von XWizard angezeigt. Der Nutzer kann alle **verfügbaren** Konversionsmethoden zum Lösen der Aufgabe verwenden; durch die Aufgabendefinition können jedoch nach Wunsch manche Konversionsmethoden verborgen werden. Der folgende Bildschirmausschnitt zeigt eine Beispielaufgabe auf der XWizardWebseite. (Beachten Sie, dass Aufgaben derzeit nicht durch VFP angezeigt werden können.)



Skript ID-11020

Der Anwender kann eine vermutete Lösung ins Textfeld “Lösung” eingeben. Falls diese korrekt ist, gibt XWizard eine Gratulationsnachricht aus, und, falls vom Ersteller der Aufgabe zur Verfügung gestellt, auch eine zusätzliche Erklärung und ein “Geheimwort” (bis jetzt hat das Geheimwort noch keine weitere Bedeutung, es wird lediglich als kleine Belohnung für das Lösen der Aufgabe angezeigt). XWizard wechselt wieder in den Normalbetrieb, wenn man den “Aufgabe schließen”-Link anklickt. Das folgende Skript zeigt, wie eine Aufgabe im Variablen-Teil (in rot gefärbt) kodiert wird.

```
grammar:
    A => A, A | 0 | epsilon;
    E => A, 1, A;
    S => E, E, E | S, S | 0;
--declarations--
    e=#tit=~Legen Sie den Syntaxbaum für das Wort 01011 mit der gegebenen Grammatik an.~,
    exp=~<P>Im Ausgabebereich wird der Grammatikbaum mit einigen Ableitungen von Wörtern,
    die von der Grammatik erzeugt wurden, angezeigt. Da die Grammatik einen
    Epsilonübergang enthält, muss sie zuerst epsilonfrei gemacht werden, indem man die
    zugehörige Konversionsmethode anwendet. Danach kann man die verbleibenden
    Konversionsmethoden dazu verwenden, den Syntaxbaum für 01011 anzulegen. (Alle
    anderen Konversionsmethoden sind verborgen).</P><P>Führen Sie diese Schritte aus und
    zählen Sie die Anzahl der Nonterminalknoten im Baum. Geben Sie diese Nummer in das
    Lösungsfeld ein.</P>~,
       sol=~6~,
       cod=~parser-guru~,
       met=~.*Epsilon.*|.*Parse.*~,
       cur=~.*~,
       tar=~.*~,
       crypt=~false~,
       excrypt=~false~,#;
    N=S,E,A;
    T=0,1;
    S=S;
    displayMode=2;
    maxdepth=8;
    cutNonTerminalBranches=true;
    cutTerminalDoubleBranches=true;
    maxLengthWords=4;
    multiLetterSymbolsHaveIndex=true;
    parseTreeNum=0;
--declarations-end--
```

In diesem Fall wird die Aufgabe als einfacher, menschenlesbarer Text in das Skript eingegeben – was das Schummeln einfach macht. Beispielsweise könnte die Lösung (“6”) aus dem Wert der Variable `sol` abgelesen werden. Um das zu verhindern, kann der Ersteller der Aufgabe zwei Stufen der Verschlüsselung auf das Skript anwenden: entweder wird nur die Aufgabendefinition verschlüsselt (Variable `excrypt`) oder das gesamte Skript (Variable `crypt`). Genaueres zur Verschlüsselung der Aufgaben kann man im Dokument “Handbuch für Unterrichtende” nachlesen.

Natürlich ist die Ausführung von Konversionsmethoden per Skript, wie in Abschnitt 3.3 beschrieben, nicht für verschlüsselte Skripte verfügbar. Anderenfalls könnten verborgene Konversionsmethoden doch wieder ausgeführt werden.

5 Rechtliche Hinweise

Die folgenden rechtlichen Hinweise finden Sie auch in einer aktuellen Fassung unter:

`http://www.xwizard.de:8080/Wizz?impressum&lang=ger`

Easy Agent Simulation und das darin enthaltene Teilprogramm Very Fast PDF Generator (auch PDF-XWizard oder XWizard-Desktopversion genannt) sowie XWizard, die Web-Version des letztgenannten, sind Open-Source-Programme; sämtliche Quelltexte, insbesondere Programmcode in Java, SQL, XML, HTML, LaTeXCode, GraphViz, XWizard-SCRIPT usw., ob nativ oder automatisch generiert, sind geschützt durch die Creative Commons by-nc-sa-Lizenz.

Alle Quelltexte sowie Javadoc für die meisten Java-Klassen können bei Sourceforge heruntergeladen werden:

- EAS (inklusive VFP) auf Sourceforge: `https://sourceforge.net/projects/easyagentsimulation`
- XWizard auf Sourceforge: `https://sourceforge.net/projects/xwiz`

Kurzgefasst dürfen Sie:

- Teilen – das Material in jedwedem Format oder Medium vervielfältigen und weiterverbreiten,
- Bearbeiten – das Material remixen, verändern und darauf aufbauen.

Der Lizenzgeber kann diese Freiheiten nicht widerrufen solange Sie sich an die Lizenzbedingungen halten.

Unter folgenden Bedingungen:

- Namensnennung – Sie müssen angemessene Urheber- und Rechteangaben machen, einen Link zur Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Diese Angaben dürfen in jeder angemessenen Art und Weise gemacht werden, allerdings nicht so, dass der Eindruck entsteht, der Lizenzgeber unterstütze gerade Sie oder Ihre Nutzung besonders.
- Nicht kommerziell – Sie dürfen das Material nicht für kommerzielle Zwecke nutzen.
- Weitergabe unter gleichen Bedingungen – Wenn Sie das Material remixen, verändern oder anderweitig direkt darauf aufbauen, dürfen Sie Ihre Beiträge nur unter derselben Lizenz wie das Original verbreiten.

Keine weiteren Einschränkungen – Sie dürfen keine zusätzlichen Klauseln oder technische Verfahren einsetzen, die anderen rechtlich irgendetwas untersagen, was die Lizenz erlaubt.

Detaillierte Lizenzbestimmungen (Deutsch): `http://creativecommons.org/licenses/by-nc-sa/3.0/de`

Detaillierte Lizenzbestimmungen (unported): `http://creativecommons.org/licenses/by-nc-sa/3.0/deed.en`

Viel Vergnügen mit XWizard!

Dieses Dokument wurde am 3. Februar 2017 kompiliert.