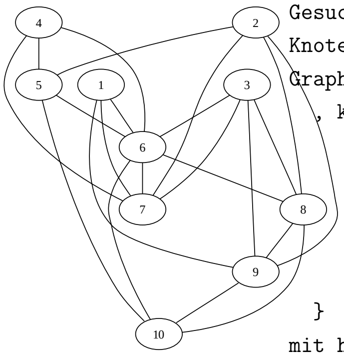
Das System autotool zur automatischen Erzeugung und Kontrolle von Übungsaufgaben zur Informatik

Mirko Rahn, Uni Karlsruhe
Alf Richter, Uni Leipzig
Johannes Waldmann, HTWK Leipzig

Beispiel: Graphenfärbung (Aufgabe)




```
Gesucht ist eine konfliktfreie
Knoten-Färbung des Graphen
Graph { knoten = mkSet [ 1 , 2 , 3 , 4
  , kanten = mkSet [ kante 1 6 , kante
     , kante 2 7 , kante 2 8 , kante 2
     , kante 3 8 , kante 3 9 , kante 4
     , kante 5 6 , kante 5 10 , kante
     , kante 8 9 , kante 8 10 , kante
     ]
  }
mit höchstens 3 verschiedenen Farben.
```


Beispiel: Graphenfärbung (Lösung)

Eingabe:

```
listToFM [ ( 1 , C ) , ( 2 , C ) , ( 3 , B ) , ( 4 , B ) , ( 5 ,
    , ( 6 , A ) , ( 7 , A ) , ( 8 , C ) , ( 9 , C ) , ( 10 , C )
```

Bewertung:

```
Ist die Menge
    Knotenmenge des Graphen =
    mkSet [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 ]
Teilmenge der Menge
    gefärbte Knoten = mkSet [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9
? Ja.
Diese Kante(n) verlaufen zwischen gleichfarbigen Knoten:
    [ kante 1 9 , kante 2 8 , kante 2 9 , kante 4 5 , kante 6 7
    , kante 8 9 , kante 8 10 , kante 9 10 ]
```


Typische Anwendung

Problem (Bsp: COL)

- Instanz: ein Graph G und eine Zahl k
- Lösung: eine k-Färbung von G

Ablauf mit autotool :

- Tutor konfiguriert Generator
- Student betrachtet Aufgabe:
  autotool erzeugt persönliche Instanz
- Student gibt (vermutete) Lösung ein
- autotool verifiziert Lösung,
  gibt ausführlichen Bericht (sofort)

Beispiel: Graphenfärbung (Konfiguration)

das hat der Tutor eingestellt:

- Semantik: Aufgabentyp: `Col-Quiz` und Parameter
  `Config { nodes = 10 , edges = 30 , chi = 3 }`
- Verwaltung:
  Hochschule, Vorlesung, Aufgabe,
  Bearbeitungszeitraum, Wichtigkeit

Wertung von Aufgaben

Aufgaben markieren als

- Demo (zu leicht, illustrieren Prinzip)
- Mandatory (Pflicht, Einsenden einer korrekten Lösung innerhalb der Deadline genügt)
- Optional (zu schwer, zum Basteln, Highscore)

Student kann auch außerhalb der Deadlines

- Aufgaben bearbeiten (wird korrigiert, nicht gespeichert)
- vorige Bewertung ansehen

nützlich z. B. zur Prüfungsvorbereitung

Aufgabenbereiche

- *formale Sprachen*: Grammatiken, reg. Ausdrücke
- *Automaten/Berechnungsmodelle*: endlich (Wort, Baum), Keller, Turing, Registermaschine, (prim.) rek. Funktionen
- *Graphen*: Parameter, Färbungen, Wege, . . .
- *diskrete Mathematik und Logik*: Zahlensysteme, Relationen, Boolesche Fkt., Modelle von PL-Formeln. . .
- *Datenstrukturen*: Suchbäume, . . .
- *Codes, Kompression*: Huffman, Burrows-Wheeler, Lempel-Zhiv

autotool als Verifizierer

- Der Idealfall sind Probleme aus NP (z. B. COL):
  - Student muß lange überlegen/suchen (N)
  - autotool kann schnell verifizieren (P)
- Manchmal sind die Lösungen länger (PCP)
- oder das Verifizieren dauert länger (Äquivalenz regulärer Ausdrücke)
- Manchmal kann man nicht verifizieren, sondern nur testen (Äquivalenz von CFG).

bietet immer viele Möglichkeiten für Diskussionen mi
Studenten über Berechenbarkeit und Komplexität

Einsatz in Lehrveranstaltungen

- Automaten und Sprachen, Berechenbarkeit und Komplexität (Uni Leipzig ab 2001, Uni Halle ab 2006)
- Automaten und Sprachen im Compilerbau (HTWK Leipzig, ab 2003)
- Datenstrukturen, diskrete Mathematik in Grundlagen der Informatik (Nebenfach) (HTWK L ab 2003)
- Datenstrukturen, disk. Math. in Grundl. Inf. (Nebenfach) (Uni Karlsruhe ab 2005, Uni Halle ab 2006)

Erfahrungen

- `autotool` zur Unterstützung des Übungsbetriebes. etwa die Hälfte der Aufgaben mit `autotool` korrigieren ... die andere Hälfte: schriftliche (Beweis-)Aufgaben
- wird von Studenten gut angenommen, sind erfreut über sofortige, ausführliche Antwort.
- Highscore-Wertung (mit Preisen) schafft zusätzlichen Anreiz

Installation, Nutzung

separate Installation:

- halbwegs schnellen Rechner mit GNU/Linux, Apache Webserver, GHC-Compiler, MySQL-Datenbankserver
- dafür Administrator, der System und Datenbank einrichtet und im Betrieb bei Bedarf Patches einspielt und kompiliert

zentrale Installation: Benutzung eines gemeinsamen `autotool`-Servers (@ HTWK) durch mehrere Einrichtungen (HTWK, Uni Leipzig, Uni Halle)

Bestandteile des `autotool`

- Semantik-Bibliothek (Automaten, Grammatiken, Graphen, ...)
- Generator-Programme
- Korrektur-Programme
- Datenbank (2002) Konfiguration der Generatoren, Aufgaben erreichte Punkte
- Web-Schnittstelle für Studenten (2003), für Tutoren (2005)

`autotool` intern

implementiert in Haskell (purely functional, strictly typed, polymorphic, lazy)

von Waldmann, Rahn, Richter seit ca. 2001

einige Teile waren Belegarbeiten für Vorlesung Funktionale Programmierung (Gerber)

Umfang:

- Bibliothek (allg. Datenstrukturen und endliche Automaten): 300 Module, 15 kLOC;
- Tool: 600 Module, 45 kLOC

(10 Zeilen pro Arbeitstag (Fred Brooks, 1972) $\rightarrow$ ...)

Weiterentwicklung

derzeit:

- Wartung und Ergänzung durch Waldmann (Leipzig) und Rahn (Karlsruhe) nach „Eigenbedarf“

demnächst

- Semantik-Dienste (Aufgaben-Erzeugung und -Korrektur) als Webservice (XML-RPC):
  - kann von anderen E-Learn-Portalen benutzt werden (Prototypen: Lips, Elate)
  - Service-Schnittstelle kann durch andere Semantik-Provider implementiert werden

Informatik I (als Nebenfach)

- Einführung Algorithmen, Sortieren: *Sortiernetze*
- Komplexität (Suchprobleme): *COL* (NP), *Lunar Lockout* (PSPACE),*PCP* (RE)
- Programmierung: einfache Programme: *Collatz(/Inverse)*; Typprüfungen: *mehrsortige Algebra*
- Datenstrukturen: *Suchbäume (Einfügen/Löschen)*

Sortiernetze

Finden Sie ein Sortiernetz für 5 Eingänge mit weniger als 10 Komparatoren.

```
mkNetz [ ( 1 , 4 ) , ( 3 , 4 ) , ( 2 , 3 ) , ( 1 , 2 ) , ( 3 , 5

Diese Eingabe wird
nicht korrekt geordnet:
[ 5 , 1 , 2 , 3 , 4 ]
Das Netz berechnet
die Ausgabe:
[ 1 , 3 , 2 , 5 , 4 ]
```

```
---------------------4--o--4---
                        v
---3--o--5--o--5--------v------
      v     v           v
------v--2--o--2--o--2--o--2---
      v           v
------v--------1--o--1--o--3---
      v                 v
---5--o--3-----------3--o--1---
```

Diskussion: Spezifikation, Korrektheitsbeweise, untere Schranken.

Suchproblem: Lunar Lockout

Geben Sie eine Zugfolge an, die den Roboter (Großbuchstabe) ins Ziel (entsprechender Klein- buchstabe) bringt:

```
. D . . .
. . . . C
B e . . .
. . . . .
. . E . .
. . . . A
```

Lösung:

```
 [ ( "A" , N ) , ("A", W), ("A", N)
, ("C", W), ( "E" , N ), ("E", W) ]
```

Diskussion: Begriff Konfiguration; Anzahl der Konfigurationen als Funktion von Breite, Höhe, Anzahl.

Suchproblem: PCP

Lösen Sie diese Instanz des Postschen Korrespondenz-Problems:

```
 PCP [ ( "aa" , "ba" ) , ( "ab" , "a" ) , ( "c" , "a" )
     , ( "bac" , "accbac" ) ]
gelesen: [ 2 , 1 , 2 , 4 ]

Aus Ihrer Folge entstehen die Zeichenketten:
    abaaabbac
    abaaaccbac
Die eine muß ein Präfix der anderen sein,
nach Löschen des gemeinsamen Präfixes  "abaaa"
entstehen jedoch die Reste  ( "bbac" , "ccbac" )
```

Diskussion: unbeschränkter Suchraum, Halteproblem

Aufgaben zu Collatz-Folgen

(Bsp: $7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1$)
direkt:

```
Gesucht sind Länge und maximales Element
der Collatz-Folge mit Startzahl 48863
```

invers:

```
Gesucht ist eine Startzahl,
deren Collatz-Folge diese Parameter hat:
Parameter { length = 247 , top = 481624 }
```

Diskussion: einfache imperative Programme mit Verzweigungen und Schleifen.

Typen (mehrsortige Algebren)

Gesucht ist ein Ausdruck vom Typ `boolean` in der Signatur

```
char a;
String b;
static String c ( boolean x );
static Bar d ( String x , char y , String z );
static boolean e ( Bar x , Bar y );
```

Lösung:

```
e (d (b, a, b), d (b, a, b))
```

Diskussion: Syntax und Semantik von Ausdrücken, Typprüfung (abstrakte Interpretation)

Datenstrukturen: Bäume

Rekonstruktion: Gesucht ist ein binärer Baum t mit der Knoten-Reihenfolgen:

```
Preorder (t) = [k,j,f,l,a,h,b,c,i,e,m,d,g]
Inorder  (t) = [f,j,l,k,c,b,i,h,e,a,d,m,g]
```

Suchbäume (unbalanicert, 2/3):

```
Auf den Baum: (Bild)
sollen diese Operationen angewendet werden
(wobei Sie  Any  geeignet ersetzen sollen):
    [ Any , Any, Insert 433, Any  ]
so daß dieser Baum entsteht: (Bild)
```


Informatik II (als Nebenfach)

- Aussagenlogik: *SAT, Boolesche Funktionen*
- Zahlendarstellungen: *Basiswechsel, Gleitkomma-Approximationen*
- Codes: *Hamming-Abstände*
- Kompression: *Huffman, Lempel-Zhiv*
- Verschlüsselung: *Erweiterter Euklidischer Algorithmus, RSA*

Aussagenlogik

- Finden Sie eine erfüllende Belegung für die Formel $(p \wedge q \wedge \neg t) \vee (p \wedge r \wedge s) \vee (p \wedge s \wedge t) \vee (p \wedge s \wedge r) \vee (p \wedge t \wedge \neg s) \vee (q \wedge t \neg r) \vee \ldots$
- Gesucht ist ein aussagenlogischer Ausdruck, der äquivalent ist zu:

  ```
  ((y == ! z) || x && x) || y
  ```

  und nur diese Operatoren enthält:

  ```
  mkSet [ <= , false ]
  ```

Diskussion: Erfüllbarkeit, Entscheidbarkeit, Komplexität, Boolesche Basisfunktionen

Zahlendarstellungen

- Stellen Sie

  ```
  Zahl { basis = 3
    , ziffern = [1,0,1,0,0,1,1,0,0,1,2,1,0]
    }
  ```

  in der Basis 5 dar.
- Welche Gleitkommazahl mit diesen Eigenschaften

  ```
  Config  { basis = 2
    , max_stellen_mantisse = 3
    , max_stellen_exponent = 3 }
  ```

  ist eine gute Näherung für 4 / 7 ?

Codes: Hamming-Abstand

Gesucht ist ein Code (als Liste von Wörtern über L, R) mit diesen Eigenschaften:

```
Config { width = ( Fixed , 4 )
    , size = ( Atleast , 5 )
    , distance = ( Atleast , 2 )
    , optimize = Size }
```

eine Lösung

```
[ [L,R,R,L], [R,L,L,R], [L,L,L,L]
, [R,R,R,R], [L,L,R,R] ]
```

Diskussion: Fehlererkennung, -korrektur; Dreiecksungleichung, Schranken für Codegröße

Huffman-Codes

gesucht ist ein optimaler Präfix-Code über dem Code-Alphabet `[L, R]` für die Buchstabenanzahl

```
[ ( 'a' , 11 ) , ( 'b' , 47 ) , ( 'c' , 6 )
, ( 'd' , 20 ) , ( 'e' , 30 ) , ( 'f' , 31 ) ]
```

Form der Lösung:

```
Code [ ( 'a' , [ R ] ) , ( 'b' , [ L , R ] )
    , ...
    , ( 'f' , [ L , L , L , L , L , R ] ) ]
```


Lempel-Zhiv-Komppression

Finden Sie eine möglichst gute Komprimierung von

```
"010010100100101001010010010100010"
```

nach dem Verfahren `Lempel_Ziv_77`

Form der Lösung:

```
[ Letter '0'
, Letter '1'
, Block {  width = 2, dist = 0 }
, Block {  width = 3, dist = 1 }
, ...
]
```

erzeugt `0 1 01 010 ...`

Kryptografie (RSA)

- Gegeben ist das Zahlenpaar `(a, b) = ( 2548 , 1496 )`. Gesucht ist ein Paar `(c, d)` von Zahlen mit der Eigenschaft `a * c + b * d = ggT (a, b)`.
- Gesucht sind 2 Zahlen `x_1 .. x_3` mit `x_i > 1` und `product [ x_1 , ... , x_3 ] = 580932019`
- Finden Sie den Klartext für eine RSA-Verschlüsselung mit

```
Config { public_key = ( 1691, 2809 )
       , message = 1404 }
```


Math. Grundlagen d. Informatik (Uni Halle

- Mengen und algebraische Strukturen
  Mengenoperationen (Algebraic-Set)
  Verknüpfungen von Relationen (Algebraic-Relation)
  mehrsortige Algebren (Sorten)
- Graphen
  Circle, *Wegematrix* (Way), *Bitpartit* (Bi),
  Färbung (Col), *Hamilton*
  Selbstkomplementärer Graph, *Nachbar*
  Operationen auf Graphen (Algebraic-Graph)

Logik

- Aussagenlogik
  erfüllende Belegung (SAT)
  äquivalente boolesche Ausdrücke (Boolean)
  Beweise im Hilbert-Kalkül (Hilbert)
- Prädikatenlogik
  Modelle für Formeln (Find-Model)

Mengenoperationen

```
Gesucht ist ein Ausdruck mit dieser Bedeutung:
{1, 5, {}, {4}}
Sie dürfen diese Symbole benutzen
    zweistellige : [ + , - , & ]
    einstellige  : [ pow ]
    nullstellige : [ 0 , 1 , 2 , 3 , 4 , 5 , 6]
und diese vordefinierten Konstanten:
    A = {1, 3, 5, 6}
    B = {2, 3, 6}
```

Lösung:

```
  A - B + pow (4)
```


Relationen

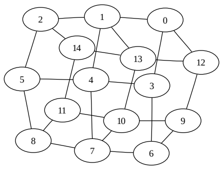
```
Gesucht ist ein Ausdruck mit dieser Bedeutung:
   {(2 , 3) , (4 , 1)}
Sie dürfen diese Symbole benutzen
    zweistellige : [ + , - , & , * ]
    einstellige  : [ inv , tcl , rcl ]
    nullstellige : [ ]
und diese vordefinierten Konstanten:
    R = {(1 , 2) , (3 , 4)}
    S = {(2 , 3) , (4 , 1) , (5 , 2)}
```

Lösung:

```
  inv (R * S * R)
```

Operationen auf Graphen

Gesucht ist ein Ausdruck mit dieser Bedeutung:



der nur diese Symbole enthält:

```
    Binu { binary = [ * , % , + ] , unary = [ co ]
         , nullary = [ K1 , K2 , K3 , K4 , K5 , P3 , P4 , P5
                     , C3 , C4 , C5] }
```

Lösung:

```
 C5 % P3
```


Hilbert-Kalkül

Gesucht ist eine Ableitung für die Formel

```
    p -> p
```

im Hilbert-Kalkül mit den Axiomen

```
    { H1 = A -> (B -> A)
    , H2 = (A -> (B -> C)) -> ((A -> B) -> (A -> C))
    , H3 = (A -> B) -> (not B -> not A) , H4 = A -> (not A -> B)
    , H5 = (not A -> A) -> A
    }
```

Lösung:

```
let { F1 = sub H1 { A = p , B = q -> p }
    , F2 = sub H2 { A = p , B = q -> p, C = p}
    , F3 = mopo F1 F2
    , F4 = sub H1 { A = p, B = q}}
in mopo F4 F3
```


Modelle (Prädikatenlogik)

Finden Sie für die Formel

```
    forall x . exists y. R (x , y) && (not P (y))
```

ein Modell (eine Interpretation) der Größe

```
    3
```

Lösung:

```
Interpretation { struktur =
Struktur { universum = mkSet [ 1 , 2, 3]
         , predicates = listToFM [ ( P , {} )
                , ( R , {(1,1), (2,2), (3,1)} ) ]
         , functions = listToFM [ ]}
         , belegung = listToFM [ ]
         }
```