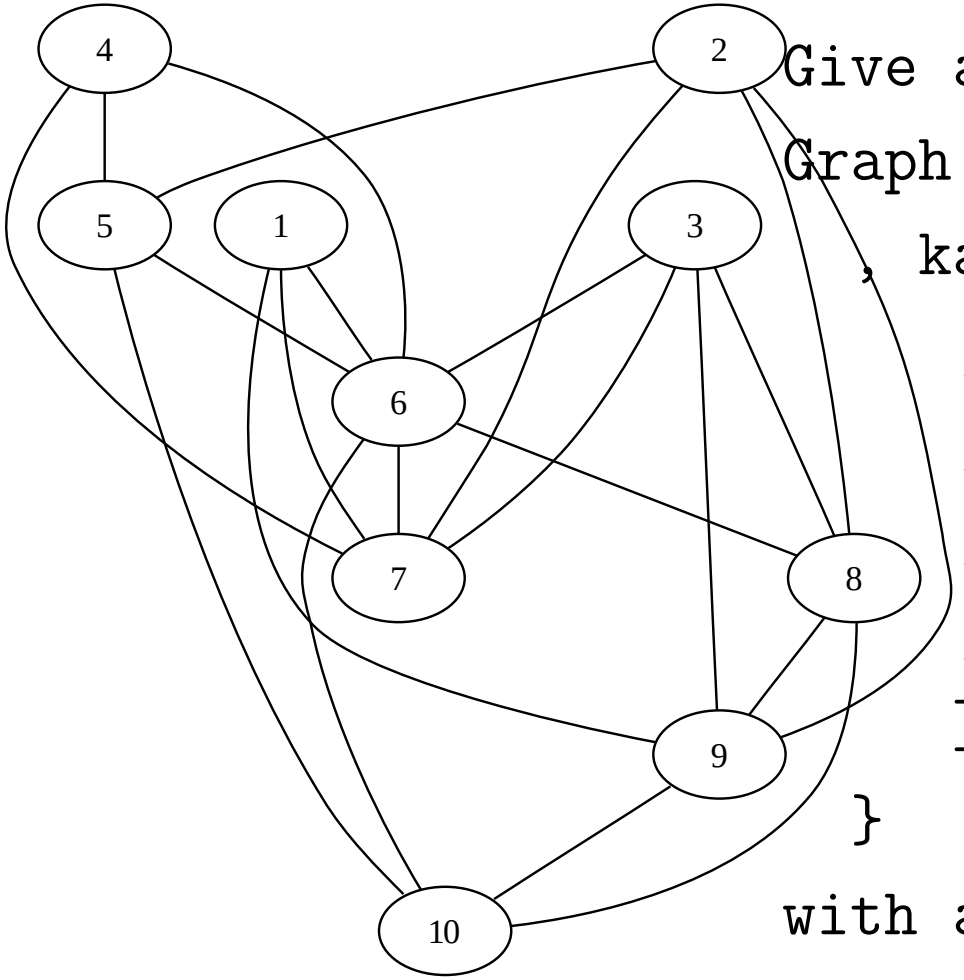
The Leipzig autotool E-Learning/E-Testing system

Mirko Rahn, Univ. Karlsruhe
Alf Richter, Univ. Leipzig
Johannes Waldmann, HTWK Leipzig

Example: Graph Colouring (Instance)




```
Give a conflict-free node colouring
Graph { knoten = mkSet [ 1 , 2 , 3 ,
  , kanten = mkSet [ kante 1 6 , kan
    , kante 2 7 , kante 2 8 , kante
    , kante 3 8 , kante 3 9 , kante
    , kante 5 6 , kante 5 10 , kant
    , kante 8 9 , kante 8 10 , kant
    ]
  }
with at most 3 different colours.
```

Example: Graph Colouring (Solution)

Input

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

Grading:

```
is the set
    nodes of graph =
    mkSet [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 ]
a subset of the set
    coloured nodes = mkSet [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 ,
? Yes.
These edges connect nodes of equal colour:
    [ kante 1 9 , kante 2 8 , kante 2 9 , kante 4 5 , kante 6
    , kante 8 9 , kante 8 10 , kante 9 10 ]
```

Typical `autotool` Use Case

Problem (Ex: COL)

- Instance: graph G, number k
- Solution: a k-colouring of G

workflow `autotool` :

- *tutor* configures *generator*
- *student* starts working:
  `autotool` generates problem instance
- student types in candidate solution
- `autotool` verifies candidate,
  reports back (verbose, immediately)

xample: Graph Colouring (Configuration

the tutor did choose this:

- semantics:
  problem type: `Col-Quiz` and parameters for generator

  ```
  Config { nodes = 10 , edges = 30 , chi = 3
  ```

- bookkeeping:
  school, lecture, exercise, time span, rating (level)

Problem levels

problems are marked as

- Demo (“too easy”, for illustration)
- Mandatory (must submit at least one correct solution before deadline, any number of attempts
- Optional (“too hard”, prize questions etc.)

even after deadline, student can

- work on problems (will be graded, but not counted)
- review previous graded answer

useful e.g. when preparing for exams

Problem Domains

- *formal languages*: grammars, regular expressions
- *automata, models of computation* finite automata, stack automata, turing machine, register machine, (primitive) recursive function
- *graphs*: parameters, colourings, paths . . .
- *logic*: sets, relations, boolean functions, predicate logic . . .
- *number theory*: gcd, RSA
- *data structures*: search trees, . . .
- *codes, compressions*: Huffman, Burrows-Wheeler, Lempel-Zhiv

`autotool` as a verifier

- ideally, problem is in NP (e.g. COL):
  - student has to guess (N)
  - `autotool` has to check (P)
- sometimes solutiones are a bit longer (PCP)
- or verification takes a bit longer (equivalence of regular expressions)
- sometimes verification is impossible,
  then replaced by testing (equivalence of CFG)

lots of opportunities to discuss with students abou
decidability and complexity

Find a small context-free grammar

for $L = \Sigma^* \setminus \{ww \mid w \in \Sigma^*\}$ where $\Sigma = \{0, 1\}$.
solution plan

- $L \to AB, L \to BA$ (start)
- $A \to 0, A \to SAS$ (0 in the middle)
- $B \to 1, B \to SBS$ (1 in the middle)
- $S \to 0, S \to 1$ (any letter)

which words are missing? (easy)
how to create them with *two* (not three) additional rules?

Fun with Regular Expressions (I)

of course this works:

- given an extended reg. exp. (complement, intersection, shuffle, . . .)
- find an equivalent simple reg. exp.

but finding a *small* answer is hard (PSPACE?)
try $a^{2*} \sqcup b^{2*}$ and then $a^{2*} \sqcup b^{2*} \sqcup c^{2*}$

Fun with Regular Expressions (II)

star height of an expression: maximal nesting number of stars
(extended) star height of a language: minimal star height of equiv. (extended) reg. exp.
Conjecture: $L \in \mathrm{REG} \Rightarrow \mathrm{ESH}(L) \leq 1$.
exercise:

- given any (simple) reg. exp.
- find equiv. extended reg. exp. of ESH ≤ 1

Using autotool

- 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)

Experiences

- use `autotool` for about half the exercises
- other half would be *formal proofs* (university) . . . or *programming* (university of applied sciences)
- students like it, and appreciate immediate and verbose response.
- additional incentive: high score competition, with prizes

Requirements

Client (student, tutor): any web browser
Server:

- for running: standard GNU/Linux machine with (Apache) web server, MySQL data base server
- for building: GHC Haskell compiler, some libraries (from hackage)
- tricky parts (currently not well-documented): build from source, initialize data base
- one central server can be used for several institutions with several lectures each

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

programmers: Waldmann, Rahn, Richter, since 2001

implementation language: Haskell (purely functional, strictly typed, polymorphic, lazy)

code size

- library (general domain knowledge)
  300 modules, 15 kLOC;
- autotool (problem generators, graders)
  600 modules, 45 kLOC;

Implementation: “class” design

Relation between (p)roblem type, (i)nstance, (s)olution

```
-- internal API, also provided via XML-RPC
class ToDoc s , Reader s =>
      Exercise p i s | p i -> s where
   describe :: p -> i -> Doc
   initial  :: p -> i -> s
   grade    :: p -> i -> s -> Reporter Grade
       ( instance MonadWriter Doc Reporter .
-- currently about 80 instances like
instance
  Exercise Col (Graph v, Int) (Map v Int) ..
```

Implementation: Multilingual output

(proof of concept)

```
inform $ fsep
    [ M.make [ ( M.DE, T.text "mit hÃ¶chstens" )
             , ( M.UK, T.text "with at most" )   ]
    , toDoc c
    , M.make [ ( M.DE, T.text "verschiedenen Farben." )
             , ( M.UK, T.text "different colours." ) ]
    ]
```

all `Doc` combinators are lifted to `Multilingual Doc`

```
data Multilingual a =
     Multilingual ( Map Language a )
```

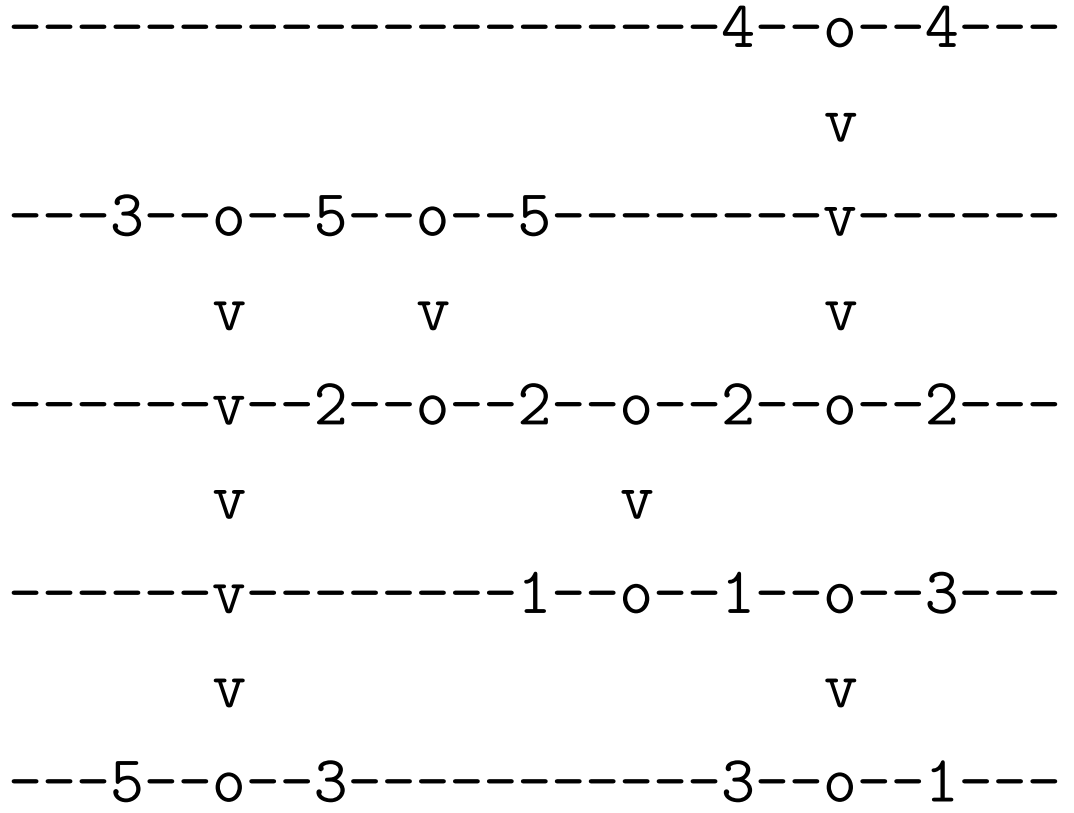
ntroduction to CS I (as a minor subject

- introduction to algorithms, sorting:
  sorting networks
- introduction to complexity (search problems)
  COL (NP), *Lunar Lockout* (PSPACE),*PCP* (RE)
- introduction to programming
  simple (imperative): *Collatz(/Inverse)*;
  type checking: *many-sorted algebras*
- data structures
  search trees (insert/delete)

Sorting Networks

Find a sorting network for 5 inputs
with less than 10 comparator circuits

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

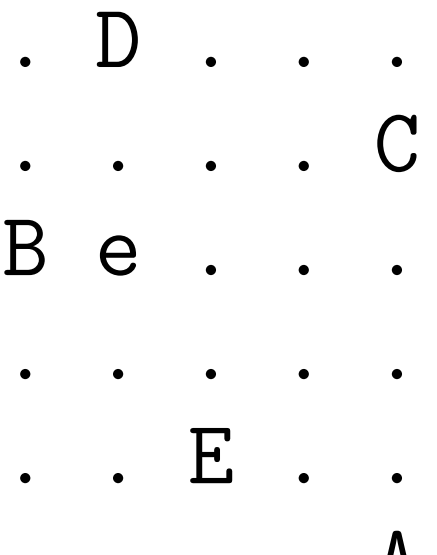


```
This input
is not handled correctly:
[ 5 , 1 , 2 , 3 , 4 ]
The output
of the network is:
[ 1 , 3 , 2 , 5 , 4 ]
```

discuss: specification, correctness, lower bounds

Search problem: Lunar Lockout

Cars at A,B,C,D,E; task: move E to e. Car stops only when hitting other car.



Solution

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

discuss: configuration, number of configurations bound

search problem: PCP

solve this instance of Post's Correspondence Problem:

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


your input creates these sequences
    abaaabbac
    abaaaccbac
One must be a prefix of the other.
After deleting the common prefix "abaaa",
remainders are  ( "bbac" , "ccbac" )
```

discuss: unbounded search space, halting problem

Collatz Sequence ($3n+1$)

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

direct:

```
find length and maximal element
of the Collatz sequence starting at 48863
```

inverse:

```
find the starting number
for a Collatz sequencw with
Parameter { length = 247 , top = 481624 }
```

discuss: simple (imperative) programming with loops and branches

Types (many-sorted algebras)

Construct an expression of type `boolean` given the signature

```
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 );
```

solution

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

discuss: syntax and semantics of expressions, type checking (abstract interpretation)

Data structures: trees

Reconstruction: Find a binary tree t with node sequences:

```
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]
```

Search trees (unbalanced, 2/3):

```
Fill in the missing operations such that
tree t1 is transformed into tree t2
    [ Any , Any, Insert 433, Any  ]
```

Introduction to CS (minor subject) II

- propositional logic: *SAT, boolean functions*
- number systems
  change of basis, floating point approximations
- codes: *Hamming-distance*
- compression: *Huffman, Lempel-Zhiv*
- cryptography
  gcd (extended), RSA

Aussagenlogik

- find a satisfying assignment for
  $(p \wedge q \wedge \neg t) \vee (p \wedge r \wedge s) \vee (p \wedge s \wedge t) \vee (p \wedge s \wedge r) \vee (p \wedge t \wedge \neg s) \vee (q \wedge t \wedge \neg r) \vee \ldots$
- find an expression that is equivalent ot

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

  and uses only the operators

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

discuss: satisfiability, decidability, complexity, bases for boolean functions

Number systems

- Convert

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

  to basis 5

- Among floating point numbers with

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

  find the best approximation to 4 / 7.

Codes: Hamming-distance

Find a code (as set of words over L,R) with

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

solution

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

discuss: error detection, error correction, triangle inequality, bounds

Huffman-Codes

find an optimal binary prefix-free code for the frequencies

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

form of solution

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

Lempel-Zhiv-compression

find a good compressed representation for

"0100101001001010010100100101010010"

using `Lempel_Ziv_77`

shape of solution

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

generates `0 1 01 010 ...`

cryptography (RSA)

- given the pair of numbers `(a, b) = ( 2548 , 1496 )`, find a pair of numbers `(c, d)` such that `a * c + b * d = gcd (a, b)`.
- find numbers `x_1 .. x_3` with `x_i > 1` and `product [ x_1 , ... , x_3 ] = 580932019`
- find the cleartext for this RSA key and message

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

Mathematics for CS (Univ. Halle)

- Sets and algebras
  operations on sets (Algebraic-Set)
  operations on relations (Algebraic-Relation)
  many-sorted algebras (Sorten)
- graphs
  Circle, *Bitpartit* (Bi),
  colouring (Col), *Hamilton*
  self dual graphs,
  graph operations (Algebraic-Graph)

Logic

- propositional
  satisfiability (SAT)
  equivalent boolean expressions (Boolean)
  derivations in Hilbert calculus (Hilbert)
- predicate
  models (Find-Model)

Set operations

```
Find an expression with value
{1, 5, {}, {4}}
You may use these symbols:
    binary : [ + , - , & ]
    unary  : [ pow ]
    nullary : [ 0 , 1 , 2 , 3 , 4 , 5 , 6]
and these constants
    A = {1, 3, 5, 6}
    B = {2, 3, 6}
```

solution:

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

Relations

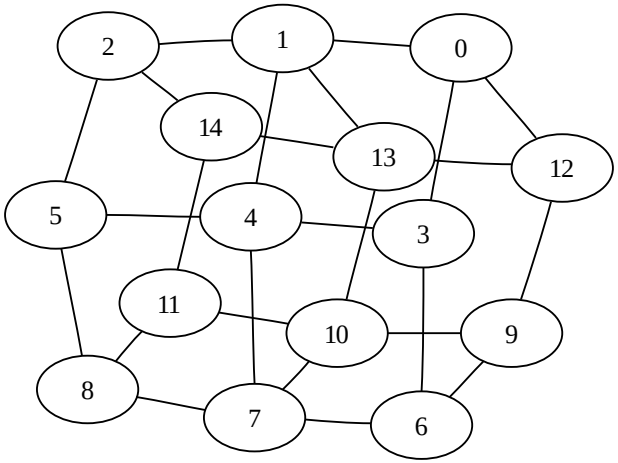
```
Find an expression with value
   {(2 , 3) , (4 , 1)}
You may use these symbols
    binary : [ + , - , & , * ]
    unary  : [ inv , tcl , rcl ]
    nullary : [ ]
and these constants
    R = {(1 , 2) , (3 , 4)}
    S = {(2 , 3) , (4 , 1) , (5 , 2)}
```

solution:

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

Graph operations

Find an expression with value



using these symbols

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

solution:

```
C5 % P3
```

Hilbert calculus

```
find a derivation for
    p -> p
using the axioms
    { H1 = A -> (B -> A)
    , H2 = (A -> (B -> C)) -> ((A -> B) -> (A -> C))
    , H3 = (A -> B) -> (not B -> not A) , H4 = A -> (not A ->
    , H5 = (not A -> A) -> A
    }
```

solution

```
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
```

Predicate logic (models)

```
For the formula
    forall x . exists y. R (x , y) && (not P (y))
find a model of size
    3
```

solution:

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

Problem types

given some “semantics” function, ask the student:

- certificate:
  find a (small) object with property . . .
- forward:
  what is the result for input . . .
- backward:
  what is the input if the result is . . .
- holes:
  fill in the missing steps such that input . . . gives result . . .

Current Work

- provide *semantics* as (stateless) service
- use existing E-Learning frameworks for *bookkeeping*
- better *user interface* (for semantics)
  - structured input (XML, schema-aware editor)
  - structured output, multilingual