ISP AUFGABE 2

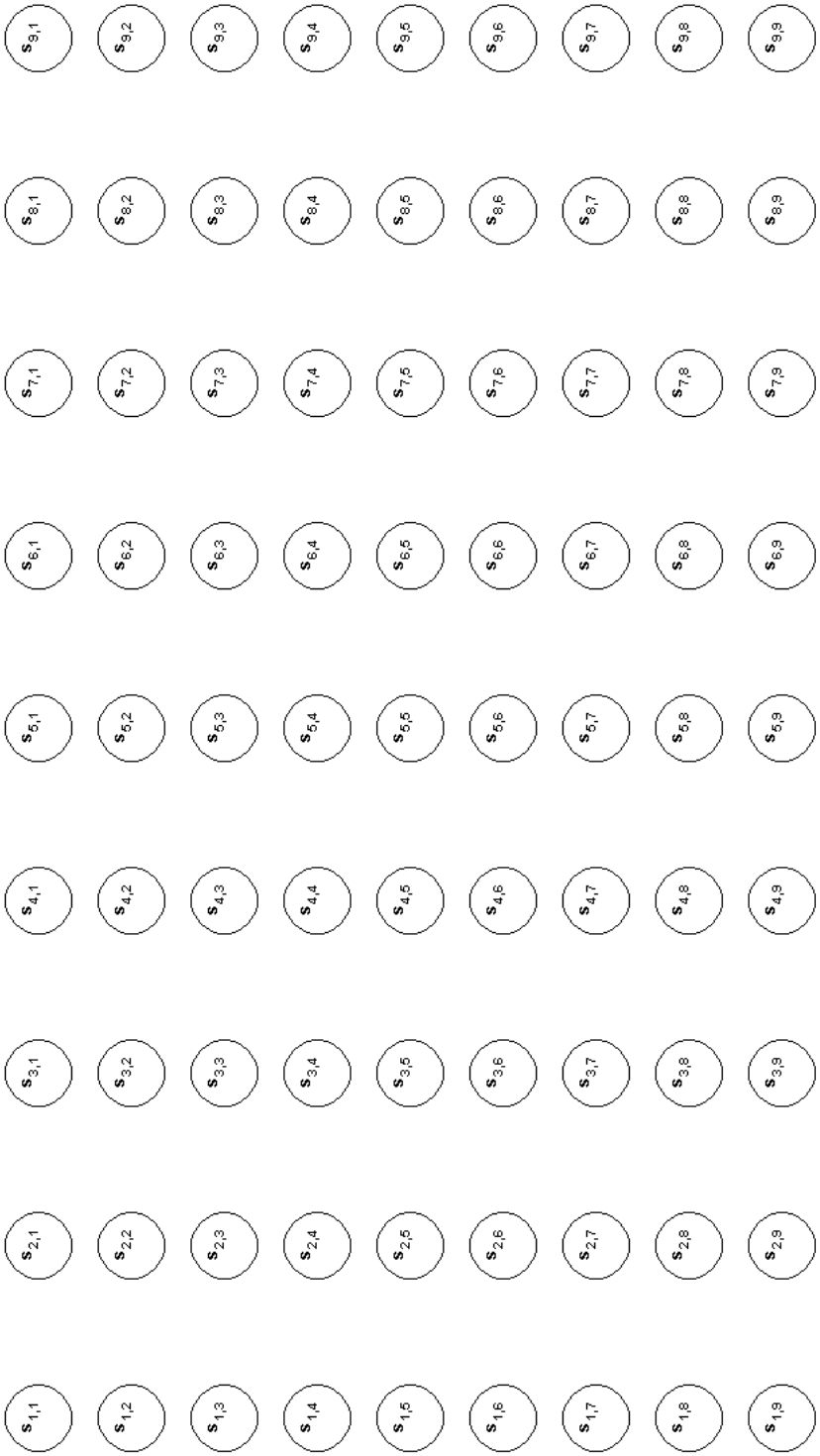
Gruppe 3

Dimitri Meier, Saeed Shanidar, Andreas Berks
dimitri.meier@haw-hamburg.de,
saeed.shanidar@haw-hamburg.de,
andreas.berks@haw-hamburg.de

Inhaltsverzeichnis

1) Constraint-Netz

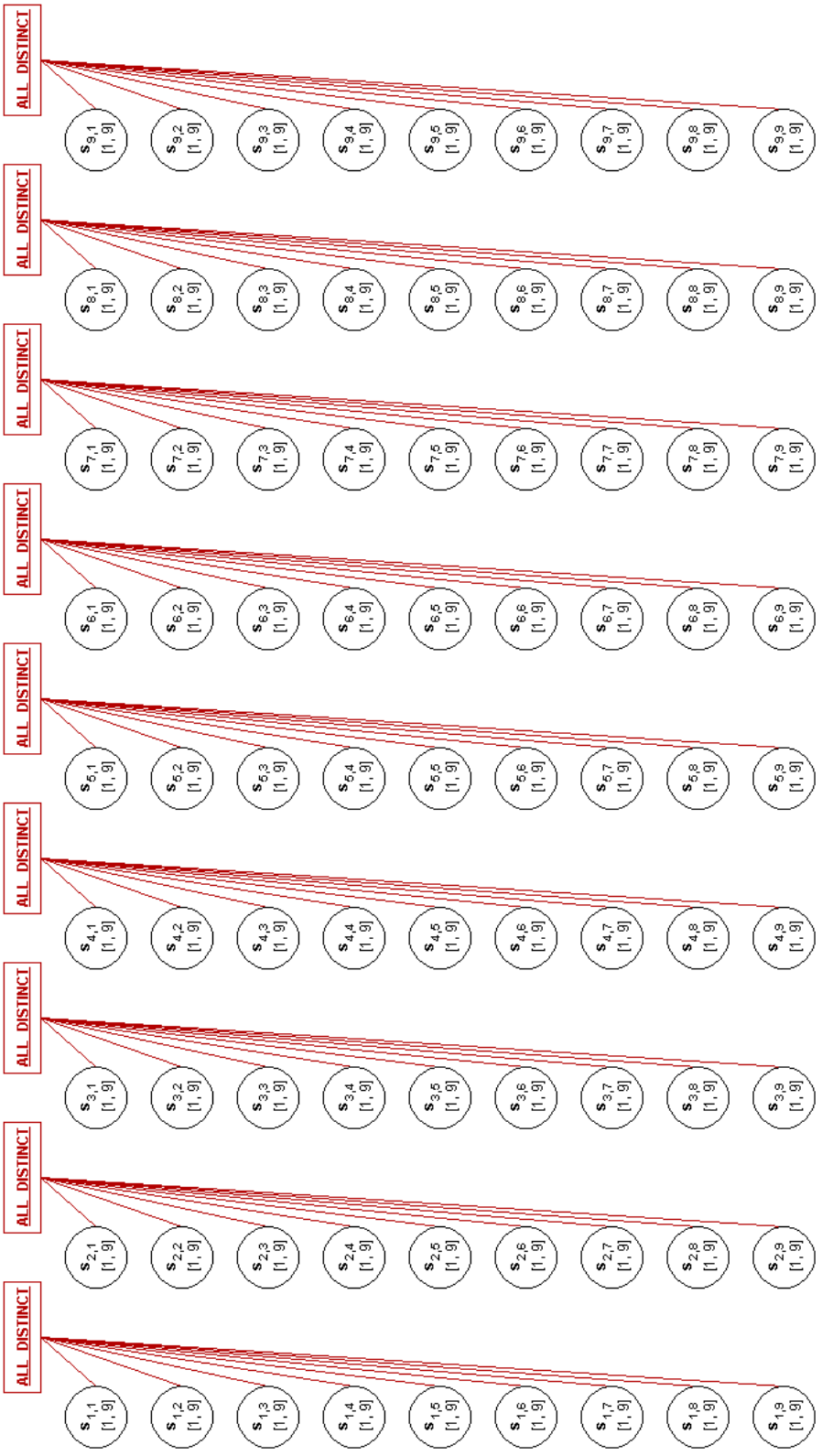
xxx



xxx

$s_{9,1}$ [1, 9]
$s_{9,2}$ [1, 9]
$s_{9,3}$ [1, 9]
$s_{9,4}$ [1, 9]
$s_{9,5}$ [1, 9]
$s_{9,6}$ [1, 9]
$s_{9,7}$ [1, 9]
$s_{9,8}$ [1, 9]
$s_{9,9}$ [1, 9]
$s_{8,1}$ [1, 9]
$s_{8,2}$ [1, 9]
$s_{8,3}$ [1, 9]
$s_{8,4}$ [1, 9]
$s_{8,5}$ [1, 9]
$s_{8,6}$ [1, 9]
$s_{8,7}$ [1, 9]
$s_{8,8}$ [1, 9]
$s_{8,9}$ [1, 9]
$s_{7,1}$ [1, 9]
$s_{7,2}$ [1, 9]
$s_{7,3}$ [1, 9]
$s_{7,4}$ [1, 9]
$s_{7,5}$ [1, 9]
$s_{7,6}$ [1, 9]
$s_{7,7}$ [1, 9]
$s_{7,8}$ [1, 9]
$s_{7,9}$ [1, 9]
$s_{6,1}$ [1, 9]
$s_{6,2}$ [1, 9]
$s_{6,3}$ [1, 9]
$s_{6,4}$ [1, 9]
$s_{6,5}$ [1, 9]
$s_{6,6}$ [1, 9]
$s_{6,7}$ [1, 9]
$s_{6,8}$ [1, 9]
$s_{6,9}$ [1, 9]
$s_{5,1}$ [1, 9]
$s_{5,2}$ [1, 9]
$s_{5,3}$ [1, 9]
$s_{5,4}$ [1, 9]
$s_{5,5}$ [1, 9]
$s_{5,6}$ [1, 9]
$s_{5,7}$ [1, 9]
$s_{5,8}$ [1, 9]
$s_{5,9}$ [1, 9]
$s_{4,1}$ [1, 9]
$s_{4,2}$ [1, 9]
$s_{4,3}$ [1, 9]
$s_{4,4}$ [1, 9]
$s_{4,5}$ [1, 9]
$s_{4,6}$ [1, 9]
$s_{4,7}$ [1, 9]
$s_{4,8}$ [1, 9]
$s_{4,9}$ [1, 9]
$s_{3,1}$ [1, 9]
$s_{3,2}$ [1, 9]
$s_{3,3}$ [1, 9]
$s_{3,4}$ [1, 9]
$s_{3,5}$ [1, 9]
$s_{3,6}$ [1, 9]
$s_{3,7}$ [1, 9]
$s_{3,8}$ [1, 9]
$s_{3,9}$ [1, 9]
$s_{2,1}$ [1, 9]
$s_{2,2}$ [1, 9]
$s_{2,3}$ [1, 9]
$s_{2,4}$ [1, 9]
$s_{2,5}$ [1, 9]
$s_{2,6}$ [1, 9]
$s_{2,7}$ [1, 9]
$s_{2,8}$ [1, 9]
$s_{2,9}$ [1, 9]
$s_{1,1}$ [1, 9]
$s_{1,2}$ [1, 9]
$s_{1,3}$ [1, 9]
$s_{1,4}$ [1, 9]
$s_{1,5}$ [1, 9]
$s_{1,6}$ [1, 9]
$s_{1,7}$ [1, 9]
$s_{1,8}$ [1, 9]
$s_{1,9}$ [1, 9]

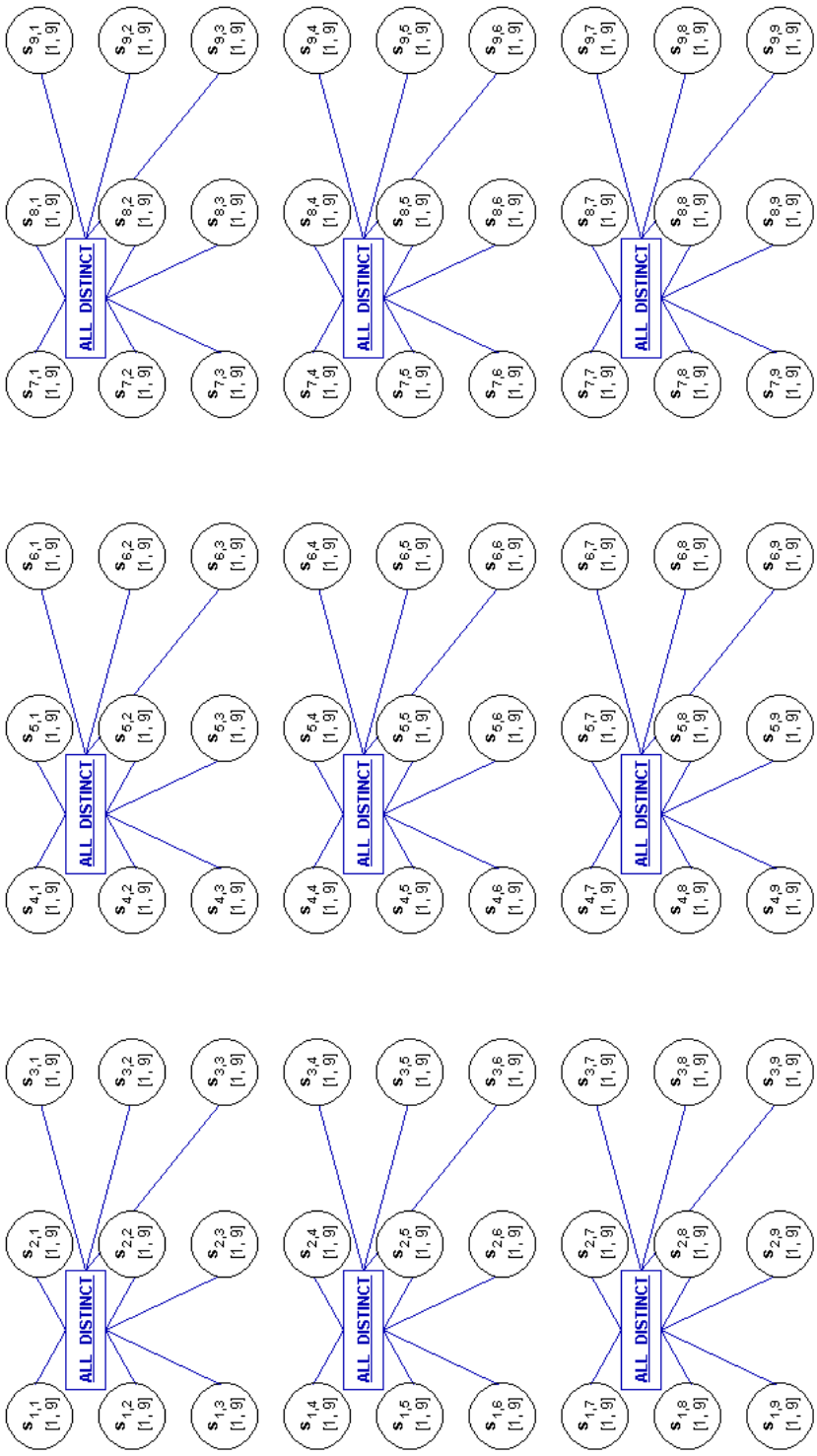
xxx


ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
s1,1 [1, 9]
s1,2 [1, 9]
s1,3 [1, 9]
s1,4 [1, 9]
s1,5 [1, 9]
s1,6 [1, 9]
s1,7 [1, 9]
s1,8 [1, 9]
s1,9 [1, 9]
s2,1 [1, 9]
s2,2 [1, 9]
s2,3 [1, 9]
s2,4 [1, 9]
s2,5 [1, 9]
s2,6 [1, 9]
s2,7 [1, 9]
s2,8 [1, 9]
s2,9 [1, 9]
s3,1 [1, 9]
s3,2 [1, 9]
s3,3 [1, 9]
s3,4 [1, 9]
s3,5 [1, 9]
s3,6 [1, 9]
s3,7 [1, 9]
s3,8 [1, 9]
s3,9 [1, 9]
s4,1 [1, 9]
s4,2 [1, 9]
s4,3 [1, 9]
s4,4 [1, 9]
s4,5 [1, 9]
s4,6 [1, 9]
s4,7 [1, 9]
s4,8 [1, 9]
s4,9 [1, 9]
s5,1 [1, 9]
s5,2 [1, 9]
s5,3 [1, 9]
s5,4 [1, 9]
s5,5 [1, 9]
s5,6 [1, 9]
s5,7 [1, 9]
s5,8 [1, 9]
s5,9 [1, 9]
s6,1 [1, 9]
s6,2 [1, 9]
s6,3 [1, 9]
s6,4 [1, 9]
s6,5 [1, 9]
s6,6 [1, 9]
s6,7 [1, 9]
s6,8 [1, 9]
s6,9 [1, 9]
s7,1 [1, 9]
s7,2 [1, 9]
s7,3 [1, 9]
s7,4 [1, 9]
s7,5 [1, 9]
s7,6 [1, 9]
s7,7 [1, 9]
s7,8 [1, 9]
s7,9 [1, 9]
s8,1 [1, 9]
s8,2 [1, 9]
s8,3 [1, 9]
s8,4 [1, 9]
s8,5 [1, 9]
s8,6 [1, 9]
s8,7 [1, 9]
s8,8 [1, 9]
s8,9 [1, 9]
s9,1 [1, 9]
s9,2 [1, 9]
s9,3 [1, 9]
s9,4 [1, 9]
s9,5 [1, 9]
s9,6 [1, 9]
s9,7 [1, 9]
s9,8 [1, 9]
s9,9 [1, 9]

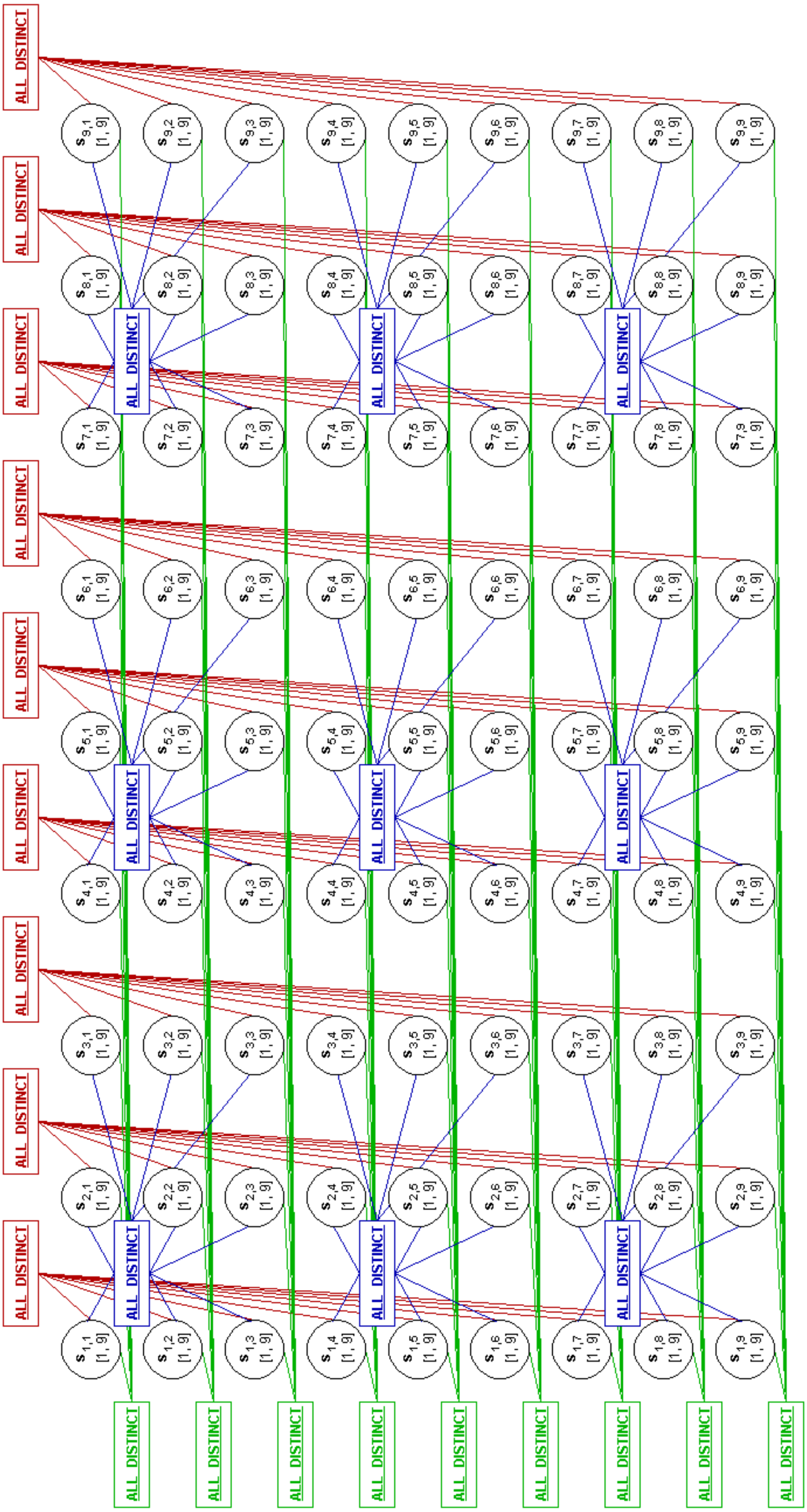
xxx

ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT

XXX


ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT
ALL_DISTINCT

xxx


ALL_DISTINCT
s9,1 [1, 9] s9,2 [1, 9] s9,3 [1, 9] s9,4 [1, 9] s9,5 [1, 9] s9,6 [1, 9] s9,7 [1, 9] s9,8 [1, 9] s9,9 [1, 9]
ALL_DISTINCT
s8,1 [1, 9] s8,2 [1, 9] s8,3 [1, 9] s8,4 [1, 9] s8,5 [1, 9] s8,6 [1, 9] s8,7 [1, 9] s8,8 [1, 9] s8,9 [1, 9]
ALL_DISTINCT
s7,1 [1, 9] s7,2 [1, 9] s7,3 [1, 9] s7,4 [1, 9] s7,5 [1, 9] s7,6 [1, 9] s7,7 [1, 9] s7,8 [1, 9] s7,9 [1, 9]
ALL_DISTINCT
s6,1 [1, 9] s6,2 [1, 9] s6,3 [1, 9] s6,4 [1, 9] s6,5 [1, 9] s6,6 [1, 9] s6,7 [1, 9] s6,8 [1, 9] s6,9 [1, 9]
ALL_DISTINCT
s5,1 [1, 9] s5,2 [1, 9] s5,3 [1, 9] s5,4 [1, 9] s5,5 [1, 9] s5,6 [1, 9] s5,7 [1, 9] s5,8 [1, 9] s5,9 [1, 9]
ALL_DISTINCT
s4,1 [1, 9] s4,2 [1, 9] s4,3 [1, 9] s4,4 [1, 9] s4,5 [1, 9] s4,6 [1, 9] s4,7 [1, 9] s4,8 [1, 9] s4,9 [1, 9]
ALL_DISTINCT
s3,1 [1, 9] s3,2 [1, 9] s3,3 [1, 9] s3,4 [1, 9] s3,5 [1, 9] s3,6 [1, 9] s3,7 [1, 9] s3,8 [1, 9] s3,9 [1, 9]
ALL_DISTINCT
s2,1 [1, 9] s2,2 [1, 9] s2,3 [1, 9] s2,4 [1, 9] s2,5 [1, 9] s2,6 [1, 9] s2,7 [1, 9] s2,8 [1, 9] s2,9 [1, 9]
ALL_DISTINCT
s1,1 [1, 9] s1,2 [1, 9] s1,3 [1, 9] s1,4 [1, 9] s1,5 [1, 9] s1,6 [1, 9] s1,7 [1, 9] s1,8 [1, 9] s1,9 [1, 9]
ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT
ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT
ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT
ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT ALL_DISTINCT

xxx

2) Lösung in Prolog

xxx

```
sudoku(Number) :-
      sudoku(Number, Sudoku),
      solveSudoku(Sudoku),
      printSudoku(Sudoku).
```

xxx

```
solveSudoku(Sudoku) :-
      checkLength(Sudoku),
      checkDomain(Sudoku),
      checkRows(Sudoku),
      checkCols(Sudoku),
      checkBlocks(Sudoku).
```

xxx

```
checkLength(Sudoku) :- length(Sudoku, 9), maplist(checkLengthInner, Sudoku).
checkLengthInner(Row) :- length(Row, 9).
```

xxx

```
checkDomain(Sudoku) :-
      append(Sudoku, SudokuFlatted),
      SudokuFlatted ins 1..9.
```

xxx

```
checkRows(Sudoku) :- maplist(all_distinct, Sudoku).
```

xxx

```
checkCols(Sudoku) :-
      transpose(Sudoku, SudokuTransposed),
      checkRows(SudokuTransposed).
```

xxx

```
checkBlocks(Sudoku) :-
      Sudoku =
            [[R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9],
             [R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9],
             [R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9],
             [R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9],
             [R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9],
             [R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9],
             [R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9],
             [R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9],
             [R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9]],
      SudokuBlock =
            [[R1C1, R1C2, R1C3, R2C1, R2C2, R2C3, R3C1, R3C2, R3C3],
             [R1C4, R1C5, R1C6, R2C4, R2C5, R2C6, R3C4, R3C5, R3C6],
             [R1C7, R1C8, R1C9, R2C7, R2C8, R2C9, R3C7, R3C8, R3C9],
             [R4C1, R4C2, R4C3, R5C1, R5C2, R5C3, R6C1, R6C2, R6C3],
             [R4C4, R4C5, R4C6, R5C4, R5C5, R5C6, R6C4, R6C5, R6C6],
             [R4C7, R4C8, R4C9, R5C7, R5C8, R5C9, R6C7, R6C8, R6C9],
             [R7C1, R7C2, R7C3, R8C1, R8C2, R8C3, R9C1, R9C2, R9C3],
             [R7C4, R7C5, R7C6, R8C4, R8C5, R8C6, R9C4, R9C5, R9C6],
             [R7C7, R7C8, R7C9, R8C7, R8C8, R8C9, R9C7, R9C8, R9C9]],
      checkRows(SudokuBlock).
xxx
  printSudoku(Sudoku) :- maplist(writeln, Sudoku).
xxx
```

3) Optimierung der Performance

3.1) Schritt 1: maplist durch Rekursion ersetzen

xxx

```
-checkLength(Sudoku) :- length(Sudoku, 9), maplist(checkLengthInner, Sudoku).
-checkLengthInner(Row) :- length(Row, 9).
+checkLength(Sudoku) :- length(Sudoku, 9), checkLengthInner(Sudoku).
+checkLengthInner([]).
+checkLengthInner([Row|Rest]) :- length(Row, 9), checkLengthInner(Rest).
```

xxx

```
-checkRows(Sudoku) :- maplist(all_distinct, Sudoku).
+checkRows([]).
+checkRows([Row|Rest]) :- all_distinct(Row), checkRows(Rest).
```

xxx

```
-printSudoku(Sudoku) :- maplist(writeln, Sudoku).
+printSudoku([]).
+printSudoku([Row|Rest]) :- writeln(Row), printSudoku(Rest).
```

xxx

Funktion	Inferenzen vorher	Inferenzen nachher	Differenz
checkLength	50	41	-9
checkDomain	4010	4010	0
checkRows	42635	42634	-1
checkCols	214651	214650	-1
checkBlocks	276664	276663	-1
printSudoku	20	19	-1
Gesamt	**538030**	**538017**	**-13**

xxx

3.2) Schritt 2: Rekursionen durch explizierte Unifikation ersetzen

xxx

```
-checkLength(Sudoku) :- length(Sudoku, 9), checkLengthInner(Sudoku).
-checkLengthInner([]).
-checkLengthInner([Row|Rest]) :- length(Row, 9), checkLengthInner(Rest).
+checkLength(Sudoku) :-
+     Sudoku =
+          [[_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_],
+           [_,_,_,_,_,_,_,_,_]].
```

xxx

```
  checkDomain(Sudoku) :-
-     append(Sudoku, SudokuFlatted),
+     Sudoku =
+          [[R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9],
+           [R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9],
+           [R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9],
+           [R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9],
+           [R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9],
+           [R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9],
+           [R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9],
+           [R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9],
+           [R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9]],
+     SudokuFlatted =
+          [R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9,
+           R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9,
+           R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9,
+           R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9,
+           R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9,
+           R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9,
+           R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9,
+           R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9,
+           R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9],
      SudokuFlatted ins 1..9.
```

xxx

```
-checkRows([]).
-checkRows([Row|Rest]) :- all_distinct(Row), checkRows(Rest).
+checkRows(Sudoku) :-
+     Sudoku = [Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9],
+     all_distinct(Row1),
+     all_distinct(Row2),
+     all_distinct(Row3),
+     all_distinct(Row4),
+     all_distinct(Row5),
+     all_distinct(Row6),
+     all_distinct(Row7),
+     all_distinct(Row8),
+     all_distinct(Row9).
```

xxx

```
  checkCols(Sudoku) :-
-     transpose(Sudoku, SudokuTransposed),
+     Sudoku =
+          [[R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9],
+           [R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9],
+           [R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9],
+           [R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9],
+           [R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9],
+           [R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9],
+           [R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9],
+           [R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9],
+           [R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9]],
+     SudokuTransposed =
+          [[R1C1, R2C1, R3C1, R4C1, R5C1, R6C1, R7C1, R8C1, R9C1],
+           [R1C2, R2C2, R3C2, R4C2, R5C2, R6C2, R7C2, R8C2, R9C2],
+           [R1C3, R2C3, R3C3, R4C3, R5C3, R6C3, R7C3, R8C3, R9C3],
+           [R1C4, R2C4, R3C4, R4C4, R5C4, R6C4, R7C4, R8C4, R9C4],
+           [R1C5, R2C5, R3C5, R4C5, R5C5, R6C5, R7C5, R8C5, R9C5],
+           [R1C6, R2C6, R3C6, R4C6, R5C6, R6C6, R7C6, R8C6, R9C6],
+           [R1C7, R2C7, R3C7, R4C7, R5C7, R6C7, R7C7, R8C7, R9C7],
+           [R1C8, R2C8, R3C8, R4C8, R5C8, R6C8, R7C8, R8C8, R9C8],
+           [R1C9, R2C9, R3C9, R4C9, R5C9, R6C9, R7C9, R8C9, R9C9]],
      checkRows(SudokuTransposed).
```

xxx

```
-printSudoku([]).
-printSudoku([Row|Rest]) :- writeln(Row), printSudoku(Rest).
+printSudoku(Sudoku) :-
+     Sudoku = [Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9],
+     writeln(Row1),
+     writeln(Row2),
+     writeln(Row3),
+     writeln(Row4),
+     writeln(Row5),
+     writeln(Row6),
+     writeln(Row7),
+     writeln(Row8),
+     writeln(Row9).
```

xxx

Funktion	**Inferenzen vorher**	**Inferenzen nachher**	**Differenz**
checkLength	41	1	-40
checkDomain	4010	3906	-104
checkRows	42634	42625	-9
checkCols	214650	214417	-233
checkBlocks	276663	276653	-10
printSudoku	19	10	-9
Gesamt	**538017**	**537612**	**-405**

xxx

3.3) Schritt 3: Entfernen von Unterfunktionen

xxx

```
sudoku(Number) :-
    sudoku(Number, Sudoku),
    time(solveSudoku(Sudoku)).

solveSudoku(Sudoku) :-
    % check length
        Sudoku =
         [[R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9],
          [R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9],
          [R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9],
          [R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9],
          [R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9],
          [R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9],
          [R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9],
          [R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9],
          [R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9]],

    % check domain
    SudokuFlatted =
        [R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9,
         R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9,
         R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9,
         R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9,
         R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9,
         R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9,
         R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9,
         R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9,
         R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9],
    SudokuFlatted ins 1..9,

    % check rows
    Sudoku = [Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9],
    all_distinct(Row1),
    all_distinct(Row2),
    all_distinct(Row3),
    all_distinct(Row4),
    all_distinct(Row5),
    all_distinct(Row6),
    all_distinct(Row7),
    all_distinct(Row8),
    all_distinct(Row9),
```

```
% check cols
all_distinct([R1C1, R2C1, R3C1, R4C1, R5C1, R6C1, R7C1, R8C1, R9C1]),
all_distinct([R1C2, R2C2, R3C2, R4C2, R5C2, R6C2, R7C2, R8C2, R9C2]),
all_distinct([R1C3, R2C3, R3C3, R4C3, R5C3, R6C3, R7C3, R8C3, R9C3]),
all_distinct([R1C4, R2C4, R3C4, R4C4, R5C4, R6C4, R7C4, R8C4, R9C4]),
all_distinct([R1C5, R2C5, R3C5, R4C5, R5C5, R6C5, R7C5, R8C5, R9C5]),
all_distinct([R1C6, R2C6, R3C6, R4C6, R5C6, R6C6, R7C6, R8C6, R9C6]),
all_distinct([R1C7, R2C7, R3C7, R4C7, R5C7, R6C7, R7C7, R8C7, R9C7]),
all_distinct([R1C8, R2C8, R3C8, R4C8, R5C8, R6C8, R7C8, R8C8, R9C8]),
all_distinct([R1C9, R2C9, R3C9, R4C9, R5C9, R6C9, R7C9, R8C9, R9C9]),

% check blocks
all_distinct([R1C1, R1C2, R1C3, R2C1, R2C2, R2C3, R3C1, R3C2, R3C3]),
all_distinct([R1C4, R1C5, R1C6, R2C4, R2C5, R2C6, R3C4, R3C5, R3C6]),
all_distinct([R1C7, R1C8, R1C9, R2C7, R2C8, R2C9, R3C7, R3C8, R3C9]),
all_distinct([R4C1, R4C2, R4C3, R5C1, R5C2, R5C3, R6C1, R6C2, R6C3]),
all_distinct([R4C4, R4C5, R4C6, R5C4, R5C5, R5C6, R6C4, R6C5, R6C6]),
all_distinct([R4C7, R4C8, R4C9, R5C7, R5C8, R5C9, R6C7, R6C8, R6C9]),
all_distinct([R7C1, R7C2, R7C3, R8C1, R8C2, R8C3, R9C1, R9C2, R9C3]),
all_distinct([R7C4, R7C5, R7C6, R8C4, R8C5, R8C6, R9C4, R9C5, R9C6]),
all_distinct([R7C7, R7C8, R7C9, R8C7, R8C8, R8C9, R9C7, R9C8, R9C9]),

% print rows
writeln(Row1),
writeln(Row2),
writeln(Row3),
writeln(Row4),
writeln(Row5),
writeln(Row6),
writeln(Row7),
writeln(Row8),
writeln(Row9).
```

xxx

Funktion	Inferenzen vorher	Inferenzen nachher	Differenz
checkLength	1		
checkDomain	3906		
checkRows	42625		
checkCols	214417		
checkBlocks	276653		
printSudoku	10		
Gesamt	**537612**	**537605**	**-7**

xxx

3.4) Verworfene Schritte

3.4.1) Lösung 1: Ohne Nutzung von Flatten

xxx

```
      % check domain
-     SudokuFlatted =
-          [R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9,
-           R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9,
-           R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9,
-           R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9,
-           R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9,
-           R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9,
-           R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9,
-           R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9,
-           R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9],
-     SudokuFlatted ins 1..9,
+     Sudoku = [Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9],
+     Row1 ins 1..9,
+     Row2 ins 1..9,
+     Row3 ins 1..9,
+     Row4 ins 1..9,
+     Row5 ins 1..9,
+     Row6 ins 1..9,
+     Row7 ins 1..9,
+     Row8 ins 1..9,
+     Row9 ins 1..9,
```

xxx

	Inferenzen vorher	Inferenzen nachher	Differenz
Gesamt	**537605**	**537885**	**+280**

xxx

3.4.2) Lösung 2: Ohne Unifikation der Zeilen

xxx

```
      % check rows
-     Sudoku = [Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9],
-     all_distinct(Row1),
-     all_distinct(Row2),
-     all_distinct(Row3),
-     all_distinct(Row4),
-     all_distinct(Row5),
-     all_distinct(Row6),
-     all_distinct(Row7),
-     all_distinct(Row8),
-     all_distinct(Row9),
+     all_distinct([R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9]),
+     all_distinct([R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9]),
+     all_distinct([R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9]),
+     all_distinct([R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9]),
+     all_distinct([R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9]),
+     all_distinct([R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9]),
+     all_distinct([R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9]),
+     all_distinct([R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9]),
+     all_distinct([R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9]),
```

xxx

```
      % print sudoku
-     writeln(Row1),
-     writeln(Row2),
-     writeln(Row3),
-     writeln(Row4),
-     writeln(Row5),
-     writeln(Row6),
-     writeln(Row7),
-     writeln(Row8),
-     writeln(Row9).
+     writeln([R1C1, R1C2, R1C3, R1C4, R1C5, R1C6, R1C7, R1C8, R1C9]),
+     writeln([R2C1, R2C2, R2C3, R2C4, R2C5, R2C6, R2C7, R2C8, R2C9]),
+     writeln([R3C1, R3C2, R3C3, R3C4, R3C5, R3C6, R3C7, R3C8, R3C9]),
+     writeln([R4C1, R4C2, R4C3, R4C4, R4C5, R4C6, R4C7, R4C8, R4C9]),
+     writeln([R5C1, R5C2, R5C3, R5C4, R5C5, R5C6, R5C7, R5C8, R5C9]),
+     writeln([R6C1, R6C2, R6C3, R6C4, R6C5, R6C6, R6C7, R6C8, R6C9]),
+     writeln([R7C1, R7C2, R7C3, R7C4, R7C5, R7C6, R7C7, R7C8, R7C9]),
+     writeln([R8C1, R8C2, R8C3, R8C4, R8C5, R8C6, R8C7, R8C8, R8C9]),
+     writeln([R9C1, R9C2, R9C3, R9C4, R9C5, R9C6, R9C7, R9C8, R9C9]).
```

xxx

	Inferenzen vorher	Inferenzen nachher	Differenz
Gesamt	**537605**	**537605**	**0**

xxx