TUTORIAL 8 · [COMPUTATION AND LOGIC]

OBJECTIVES

In this tutorial, you will:

– learn to apply the Tseytin transformation

– use the arrow rule to count satisfying valuations

– complete a Killer Sudoku solver.

TASKS

Exercises 1–5 are mandatory. Exercises 6 and 7 are optional.

SUBMIT a file `cl-tutorial-8` (image or pdf) with your answers that do not require programming, and the file `cl-tutorial-8-code.hs` with your Haskell code.

DEADLINE Saturday, 14th of November, 4 PM UK time

Good Scholarly Practice

Please remember the good scholarly practice requirements of the University regarding work for credit.

You can find guidance at the School page

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct.

This also has links to the relevant University pages. Please do not publish solutions to these exercises on the internet or elsewhere, to avoid others copying your solutions.

EXERCISE 1

MANDATORY | BEFORE TUTORIAL SESSION

Read Chapter 20 (*Efficient CNF Conversion*) of the textbook.

Recall the laws of Boolean algebra on page 192:

$$
\begin{array}{ccc}
a \leftrightarrow b = (a \to b) \land (b \to a) & & a \to b = \neg a \lor b \\
\neg(a \lor b) = \neg a \land \neg b & \neg 0 = 1 \quad \neg\neg a = a \quad \neg 1 = 0 & \neg(a \land b) = \neg a \lor \neg b \\
a \lor (b \land c) = (a \lor b) \land (a \lor c) & a \lor 1 = 1 = \neg a \lor a & (a \land b) \lor c = (a \lor c) \land (b \lor c) \\
a \land (b \lor c) = (a \land b) \lor (a \land c) & a \land 0 = 0 = \neg a \land a & (a \lor b) \land c = (a \land c) \lor (b \land c) \\
a \lor a = a = 0 \lor a & a \lor b = b \lor a & a \lor (b \lor c) = (a \lor b) \lor c \\
a \land a = a = 1 \land a & a \land b = b \land a & a \land (b \land c) = (a \land b) \land c
\end{array}
$$

Use these laws to convert the following expression to CNF:

$$r \leftrightarrow (s \leftrightarrow t)$$

Check if your result corresponds to the CNF given in the book:

$$(r \lor s \lor t) \land (r \lor \neg s \lor \neg t) \land (\neg r \lor s \lor \neg t) \land (\neg r \lor \neg s \lor t)$$

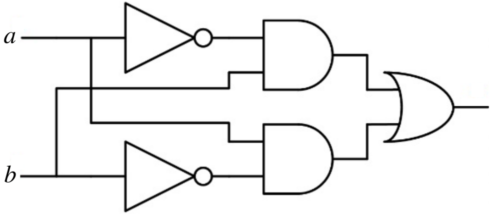
SOLUTION TO EXERCISE 1

$$
\begin{aligned}
& r \leftrightarrow (s \leftrightarrow t) \\
& = r \leftrightarrow ((s \rightarrow t) \wedge (t \rightarrow s)) \\
& = r \leftrightarrow ((\neg s \vee t) \wedge (\neg t \vee s)) \\
& = (r \rightarrow ((\neg s \vee t) \wedge (\neg t \vee s))) \wedge (((\neg s \vee t) \wedge (\neg t \vee s)) \rightarrow r) \\
& = (\neg r \vee ((\neg s \vee t) \wedge (\neg t \vee s))) \wedge (\neg((\neg s \vee t) \wedge (\neg t \vee s)) \vee r) \\
& = (\neg r \vee ((\neg s \vee t) \wedge (\neg t \vee s))) \wedge ((\neg(\neg s \vee t) \vee \neg(\neg t \vee s)) \vee r) \\
& = (\neg r \vee ((\neg s \vee t) \wedge (\neg t \vee s))) \wedge ((\neg\neg s \wedge \neg t) \vee (\neg\neg t \wedge \neg s) \vee r) \\
& = (\neg r \vee ((\neg s \vee t) \wedge (\neg t \vee s))) \wedge ((s \wedge \neg t) \vee (t \wedge \neg s) \vee r) \\
& = (\neg r \vee \neg s \vee t) \wedge (\neg r \vee \neg t \vee s) \wedge (s \vee t \vee r) \wedge (s \vee \neg s \vee r) \wedge (\neg t \vee t \vee r) \wedge (\neg t \vee \neg s \vee r) \\
& = (\neg r \vee \neg s \vee t) \wedge (\neg r \vee \neg t \vee s) \wedge (s \vee t \vee r) \wedge (1 \vee r) \wedge (1 \vee r) \wedge (\neg t \vee \neg s \vee r) \\
& = (\neg r \vee \neg s \vee t) \wedge (\neg r \vee \neg t \vee s) \wedge (s \vee t \vee r) \wedge (\neg t \vee \neg s \vee r)
\end{aligned}
$$

EXERCISE 2

MANDATORY | BEFORE TUTORIAL SESSION

Consider the following circuit:

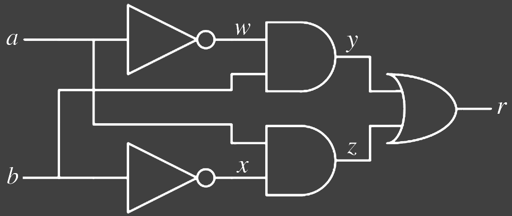


Give an equivalent logical expression.

Apply the Tseytin transformation to give an equisatisfiable CNF expression.

SOLUTION TO EXERCISE 2

We start by labelling the internal wires:



SOLUTION TO EXERCISE 2 (CONT.)

Writing down the list of equivalences and their corresponding CNF representations gives:

$$
\begin{aligned}
w \leftrightarrow \neg a : &\qquad (a \vee w) \wedge (\neg w \vee \neg a) \\
x \leftrightarrow \neg b : &\qquad (b \vee w) \wedge (\neg x \vee \neg b) \\
y \leftrightarrow w \wedge b : &\qquad (\neg y \vee w) \wedge (\neg y \vee b) \wedge (\neg w \vee \neg b \vee y) \\
z \leftrightarrow a \wedge x : &\qquad (\neg z \vee a) \wedge (\neg z \vee x) \wedge (\neg a \vee \neg x \vee z) \\
r \leftrightarrow y \vee z : &\qquad (\neg y \vee r) \wedge (\neg z \vee r) \wedge (\neg r \vee y \vee z)
\end{aligned}
$$

SOLUTION TO EXERCISE 2 (CONT.)

We then take the conjunction of these, set r to 1, and simplify, and we get:

$$
\begin{aligned}
&(a \vee w) \wedge (\neg w \vee \neg a) \\
&\wedge (b \vee x) \wedge (\neg x \vee \neg b) \\
&\wedge (\neg y \vee w) \wedge (\neg y \vee b) \wedge (\neg w \vee \neg b \vee y) \\
&\wedge (\neg z \vee a) \wedge (\neg z \vee x) \wedge (\neg a \vee \neg x \vee z) \\
&\wedge (y \vee z)
\end{aligned}
$$

EXERCISE 3

MANDATORY | BEFORE TUTORIAL SESSION

Apply the Tseytin transformation to the expression

$$(\neg a \vee c) \wedge (b \rightarrow ((a \vee c) \leftrightarrow d))$$

to give an equisatisfiable CNF expression.

Hint: Start with the innermost sub-expression.

SOLUTION TO EXERCISE 3

Starting with the innermost sub-expression:

$$
\begin{array}{ll}
x_1 \leftrightarrow (a \lor c) & (\neg a \lor c) \land (b \rightarrow (x_1 \leftrightarrow d)) \\
x_2 \leftrightarrow \neg a & (x_2 \lor c) \land (b \rightarrow (x_1 \leftrightarrow d)) \\
x_3 \leftrightarrow (x_1 \leftrightarrow d) & (x_2 \lor c) \land (b \rightarrow x_3) \\
x_4 \leftrightarrow (x_2 \lor c) & x_4 \land (b \rightarrow x_3) \\
x_5 \leftrightarrow (b \rightarrow x_3) & x_4 \land x_5 \\
x_6 \leftrightarrow (x_4 \land x_5) & x_6
\end{array}
$$

SOLUTION TO EXERCISE 3 (CONT.)

Converting these to CNF, taking the conjunction, setting x_6 to 1 and simplifying gives:

$$
\begin{aligned}
&(\neg a \vee x_1) \wedge (\neg c \vee x_1) \wedge (\neg x_1 \vee a \vee c) \\
&\wedge (a \vee x_2) \wedge (\neg x_2 \vee \neg a) \\
&\wedge (x_3 \vee x_1 \vee d) \wedge (x_3 \vee \neg x_1 \vee \neg d) \wedge (\neg x_3 \vee x_1 \vee d) \wedge (\neg x_3 \vee \neg x_1 \vee d) \\
&\wedge (\neg x_2 \vee x_4) \wedge (\neg c \vee x_4) \wedge (\neg x_4 \vee x_2 \vee c) \\
&\wedge (x_5 \vee b) \wedge (x_5 \vee \neg x_3) \wedge (\neg x_5 \vee \neg b \vee x_3) \\
&\wedge x_4 \wedge x_5
\end{aligned}
$$

EXERCISE 4

Read Chapter 23 (*Counting Satisfying Valuations*) of the textbook.

Use the arrow rule to count the number of combinations of values for literals that satisfy the following CNF expressions over the atoms A, B, C, D, E, F, G, H:

1. $E \vee F$
2. $(E \vee F) \wedge (\neg A \vee B) \wedge C$
3. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg D \vee F) \wedge (\neg E \vee F) \wedge (\neg F \vee D)$

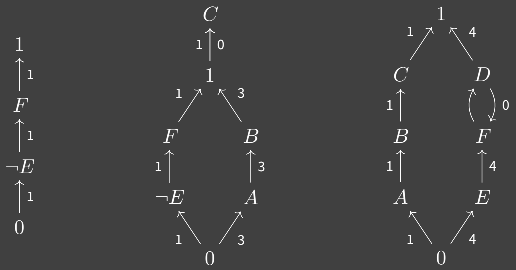
Hint: What do you need to do to the result of the calculation using the arrow rule to take account of atoms that aren't used in the expression?

SOLUTION TO EXERCISE 4

The CNF expressions are equivalent to the following conjunctions of implications:

1. $\neg E \to F$
2. $(\neg E \to F) \wedge (A \to B) \wedge (1 \to C)$
3. $(A \to B) \wedge (B \to C) \wedge (D \to F) \wedge (E \to F) \wedge (F \to D)$

These give the following diagrams of upward-pointing implications:



SOLUTION TO EXERCISE 4 (CONT.)

When a CNF expression doesn't use n atoms, the calculation using the arrow rule needs to be multiplied by 2^n: that is the number of possible combinations of values for those atoms, for each combination of values for the atoms that are present. For our diagrams, this gives:

1. $1+1+1=3$ combinations of values for the 2 atoms that are present, so $3 \times 2^6 = 192$ combinations for all 8 atoms
2. $0+3+3+3=9$ combinations of values for the 5 atoms that are present, so $9 \times 2^3 = 72$ combinations for all 8 atoms
3. $4+0+4+4=12$ combinations of values for the 6 atoms that are present, so $12 \times 2^2 = 48$ combinations for all 8 atoms

EXERCISE 5

MANDATORY | BEFORE TUTORIAL SESSION

Use the arrow rule to count the number of combinations of values for literals that satisfy the following CNF expressions:

1. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee B)$
2. $(\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg D \vee \neg A) \wedge (\neg E \vee \neg A) \wedge (A \vee C)$

The following task is optional:

3. $(A \vee B) \wedge (\neg B \vee \neg C) \wedge (\neg C \vee D) \wedge (\neg A \vee \neg E) \wedge (\neg E \vee D)$

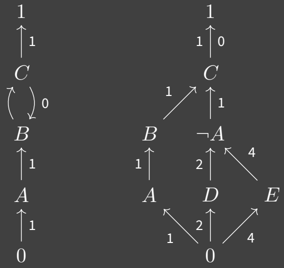
SOLUTION TO EXERCISE 5

The CNF expressions are equivalent to the following conjunctions of implications:

1. $(A \rightarrow B) \wedge (B \rightarrow C) \wedge (C \rightarrow B)$
2. $(A \rightarrow B) \wedge (B \rightarrow C) \wedge (D \rightarrow \neg A) \wedge (E \rightarrow \neg A) \wedge (\neg A \rightarrow C)$
3. $(\neg A \rightarrow B) \wedge (B \rightarrow \neg C) \wedge (C \rightarrow D) \wedge (A \rightarrow \neg E) \wedge (E \rightarrow D)$

SOLUTION TO EXERCISE 5 (CONT.)

The first and second expressions give the following diagrams of upward-pointing implications:



1. $1 + 0 + 1 + 1 = 3$ combinations
2. $0 + 1 + 4 + 4 = 9$ combinations

SOLUTION TO EXERCISE 5 (CONT.)

3. It's hard to draw a diagram of the implications that is convenient for counting cuts. We'll do a case split.

   If $A = 1$, then $\neg E = 1$ follows from $A \to \neg E$, and we have the following diagram from the remaining implications $(B \to \neg C) \land (C \to D)$, using the contrapositive $C \to \neg B$ of $B \to \neg C$:

$$\begin{tikzcd}
& 1 & \\
\neg B \arrow[ur, "1"] & & D \arrow[ul, "2"'] \\
& C \arrow[ul, "1"] \arrow[ur, "2"'] & \\
& 0 \arrow[u, "1"] &
\end{tikzcd}$$

   This gives $2 + 2 + 1 = 5$ possible cuts.

If $A = 0$, then $B = 1$ follows from $\neg A \to B$ and $\neg C = 1$ follows from $B \to \neg C$, leaving the following diagram from the remaining implication $E \to D$:

$$\begin{array}{c} 1 \\ \uparrow \\ D \\ \uparrow \\ E \\ \uparrow \\ 0 \end{array}$$

Adding the results for both cases gives a total of $5 + 3 = 8$ combinations of values for literals that satisfy the CNF expression.

EXERCISE 6

OPTIONAL | BEFORE TUTORIAL SESSION

Use Haskell to check your solutions for exercises 4 and 5.

Hint: Use the DPLL implementation from `cl-tutorial-8-code.hs` and/or a modification to the function

```
satisfiable :: Eq a => Wff a -> Bool
satisfiable p = or [ eval e p | e <- envs (atoms p) ]
```

from the FP tutorial 6 that counts satisfying environments instead of checking for their existence.

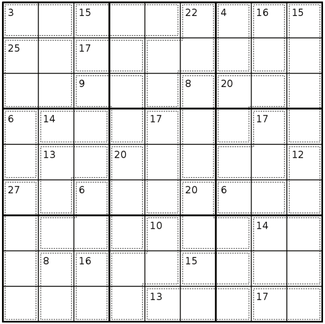
SOLUTION TO EXERCISE 7

To count the satisfying environments you could use, for example,

```
satisfyingEnvCount :: Eq a => Wff a -> Int
satisfyingEnvCount p =
    length [ e | e <- envs (atoms p), eval e p ]
```

EXERCISE 7

Killer Sudoku is a variant of the Sudoku puzzle. Like a standard Sudoku, each column, each row, and each 3×3 square must contain the numbers 1 to 9 exactly once. Killer Sudoku contains shapes marked by a dotted line (as in the image below). All the digits in a shape must add up to the total in the top corner of that shape.



EXERCISE 7

OPTIONAL | BEFORE TUTORIAL SESSION

Read the Haskell implementation of Killer Sudoku in the file `cl-tutorial-8-code.hs`.

Complete the implementation by defining the following functions:

```
scores :: Int -> Int -> [[Int]]
```

that takes two natural numbers, `n` and `m`, and returns the list of all lists `ds` of digits from `[1..9]` such that (1) `length ds == n`, and (2) `sum ds == m`;

```
mustSumTo :: Int -> Shape -> Form (Int,Int,Int)
```

that takes an integer `k` and a shape `sh` and produces a `Form` that rejects all patterns of scores whose sum is not `k`.

SOLUTION TO EXERCISE 7

```
scores :: Int -> Int -> [[Int]]
scores 0 0 = [[]]
scores 0 _ = []
scores n m = [ k : ss | k <- [1..9], ss <- scores (n-1) (m-k)]
```

```
mustSumTo :: Int -> Shape -> Form (Int,Int,Int)
mustSumTo k sh =
  And [ deny ( sh >>*< ss )
      | k’ <- [n..9*n], k’ /= k,
        ss <- scores n k’ ]
  where n = length sh
```