

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 Please do not publish solutions to these exercises on the internet or elsewhere, to avoid others copying your solutions.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:

$a \leftrightarrow b = (a \rightarrow b) \wedge (b \rightarrow a)$		$a \rightarrow b = \neg a \vee b$
$\neg(a \vee b) = \neg a \wedge \neg b$	$\neg 0 = 1 \quad \neg \neg a = a \quad \neg 1 = 0$	$\neg(a \wedge b) = \neg a \vee \neg b$
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \vee 1 = 1 = \neg a \vee a$	$(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$
$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	$a \wedge 0 = 0 = \neg a \wedge a$	$(a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c)$
$a \vee a = a = 0 \vee a$	$a \vee b = b \vee a$	$a \vee (b \vee c) = (a \vee b) \vee c$
$a \wedge a = a = 1 \wedge a$	$a \wedge b = b \wedge a$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$

 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:

$$(r \vee s \vee t) \wedge (r \vee \neg s \vee \neg t) \wedge (\neg r \vee s \vee \neg t) \wedge (\neg r \vee \neg s \vee t)$$

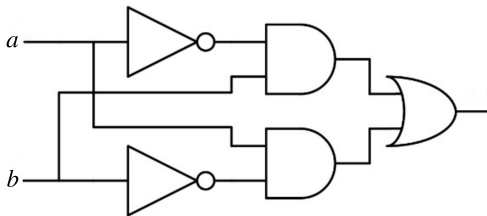
SOLUTION TO EXERCISE 1

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

EXERCISE 2

MANDATORY | BEFORE TUTORIAL SESSION

Consider the following circuit:

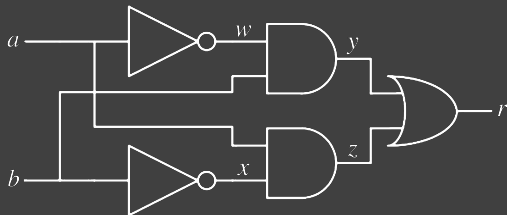


Give an equivalent logical expression.

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:

$$w \leftrightarrow \neg a : \quad (a \vee w) \wedge (\neg w \vee \neg a)$$

$$x \leftrightarrow \neg b : \quad (b \vee x) \wedge (\neg x \vee \neg b)$$

$$y \leftrightarrow w \wedge b : \quad (\neg y \vee w) \wedge (\neg y \vee b) \wedge (\neg w \vee \neg b \vee y)$$

$$z \leftrightarrow a \wedge x : \quad (\neg z \vee a) \wedge (\neg z \vee x) \wedge (\neg a \vee \neg x \vee z)$$

$$r \leftrightarrow y \vee z : \quad (\neg y \vee r) \wedge (\neg z \vee r) \wedge (\neg r \vee y \vee z)$$

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 | 1 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:

$x_1 \leftrightarrow (a \vee c)$	$(\neg a \vee c) \wedge (b \rightarrow (x_1 \leftrightarrow d))$
$x_2 \leftrightarrow \neg a$	$(x_2 \vee c) \wedge (b \rightarrow (x_1 \leftrightarrow d))$
$x_3 \leftrightarrow (x_1 \leftrightarrow d)$	$(x_2 \vee c) \wedge (b \rightarrow x_3)$
$x_4 \leftrightarrow (x_2 \vee c)$	$x_4 \wedge (b \rightarrow x_3)$
$x_5 \leftrightarrow (b \rightarrow x_3)$	$x_4 \wedge x_5$
$x_6 \leftrightarrow (x_4 \wedge x_5)$	x_6

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

MANDATORY | 🕒 BEFORE TUTORIAL SESSION

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

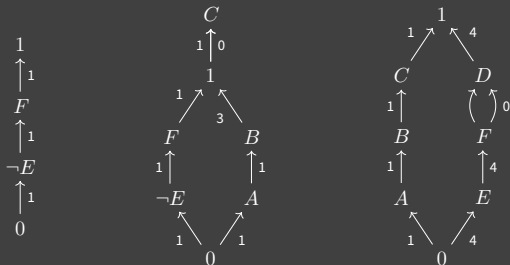
Hint: What do you need to do to the result of the calculation using the arrow rule to take account of atoms that aren't used in the expression?

SOLUTION TO EXERCISE 4

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

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

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:

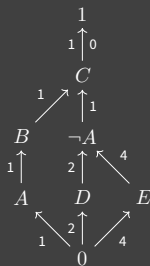
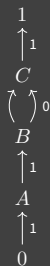
1. $(A \rightarrow B) \wedge (B \rightarrow C) \wedge (C \rightarrow B)$

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:



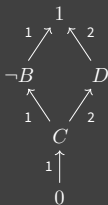
1. $1 + 0 + 1 + 1 = 3$ combinations

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 $A \rightarrow \neg E$, and we have the following diagram from the remaining implications $(B \rightarrow \neg C) \wedge (C \rightarrow D)$, using the contrapositive $C \rightarrow \neg B$ of $B \rightarrow \neg C$:



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

SOLUTION TO EXERCISE 5 (CONT.)


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



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


OPTIONAL |  BEFORE TUTORIAL SESSION


 **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.

3		15			22	4	16	15
25		17						
		9			8	20		
6	14			17			17	
	13		20					12
27		6			20	6		
				10			14	
	8	16			15			
				13			17	

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
```