Problem Sheet 9

Henri Sota
h.sota@jacobs-university.de
Computer Science 2022

November 15, 2019

Problem 9.1

$$S = A \dot{\vee} B \dot{\vee} C_{in}$$
$$C_{out} = (A \wedge B) \vee (C_{in} \wedge (A \dot{\vee} B))$$

A	**B**	$\mathbf{C_{in}}$	$\mathbf{C_{out}}$	**S**
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

a) Write both functions as a disjunction of product terms.
In order to express the boolean expressions S and C_{out} in terms of a disjunction of product terms, we first need to substitute both $\dot{\vee}$ operators in our expression for the conjunction of the disjunction and the negation of the conjunction between 2 variables, which expressed using operators and variables X and Y:

$$\begin{aligned} X \dot{\vee} Y &= (X \vee Y) \wedge \neg(X \wedge Y) \\ &= (X \vee Y) \wedge (\neg X \vee \neg Y) \\ &= ((X \vee Y) \wedge \neg X) \vee ((X \vee Y) \wedge \neg Y) \\ &= (\neg X \wedge Y) \vee (X \wedge \neg Y) \end{aligned}$$

Therefore the expressions are:

$$\begin{aligned} S &= A \dot{\vee} B \dot{\vee} C_{in} \\ &= (\neg((\neg A \wedge B) \vee (A \wedge \neg B)) \wedge C_{in}) \vee (((\neg A \wedge B) \vee (A \wedge \neg B)) \wedge \neg C_{in}) \\ &= (((A \vee \neg B) \wedge (\neg A \vee B)) \wedge C_{in}) \vee ((\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in})) \\ &= (((\neg A \wedge \neg B) \vee (A \wedge B)) \wedge C_{in}) \vee ((\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in})) \\ &= ((\neg A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge C_{in})) \vee ((\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in})) \\ &= (\neg A \wedge \neg B \wedge C_{in}) \vee (A \wedge B \wedge C_{in}) \vee (\neg A \wedge B \wedge \neg C_{in}) \vee (A \wedge \neg B \wedge \neg C_{in}) \end{aligned}$$

$$
\begin{aligned}
C_{out} &= (A \land B) \lor (C_{in} \land ((\neg A \land B) \lor (A \land \neg B))) \\
&= (A \land B) \lor ((C_{in} \land \neg A \land B) \lor (C_{in} \land A \land \neg B)) \\
&= (((C_{in} \land \neg A \land B) \lor (C_{in} \land A \land \neg B)) \lor A) \land (((C_{in} \land \neg A \land B) \lor (C_{in} \land A \land \neg B)) \lor B) \\
&= (((C_{in} \land \neg A \land B) \lor A) \lor ((C_{in} \land A \land \neg B) \lor A)) \land (((C_{in} \land \neg A \land B) \lor B) \lor ((C_{in} \land A \land \neg B) \lor B)) \\
&\vdots \\
&= (A \land B) \lor (A \land C_{in}) \lor (B \land C_{in})
\end{aligned}
$$

We arrive at the same conclusion using DNF on both expressions:

$$S = (\neg A \land B \land \neg C_{in}) \lor (A \land \neg B \land \neg C_{in}) \lor (\neg A \land \neg B \land C_{in}) \lor (A \land B \land C_{in})$$

$$C_{out} = (\neg A \land B \land C_{in}) \lor (A \land \neg B \land C_{in}) \lor (A \land B \land \neg C_{in}) \lor (A \land B \land C_{in})$$

Simplifying C_{out} using Quine-McCluskey:

$$C_{out} = (A \land B) \lor (A \land C_{in}) \lor (B \land C_{in})$$

b) Write both functions as a conjunction of sum terms.
In order to write the functions as a conjunction of sum terms, we can use CNF from their truth table:

$$S = (A \lor B \lor C_{in}) \land (\neg A \lor \neg B \lor C_{in}) \land (A \lor \neg B \lor \neg C_{in}) \land (\neg A \lor B \lor \neg C_{in})$$

$$
\begin{aligned}
C_{out} &= (A \lor B \lor C_{in}) \land (A \lor \neg B \lor C_{in}) \land (\neg A \lor B \lor C_{in}) \land (A \lor B \lor \neg C_{in}) \\
&= (A \lor B) \land (A \lor C_{in}) \land (B \lor C_{in})
\end{aligned}
$$

c) Write both functions using only not ($\neg$) and not-and ($\barwedge$) operations.

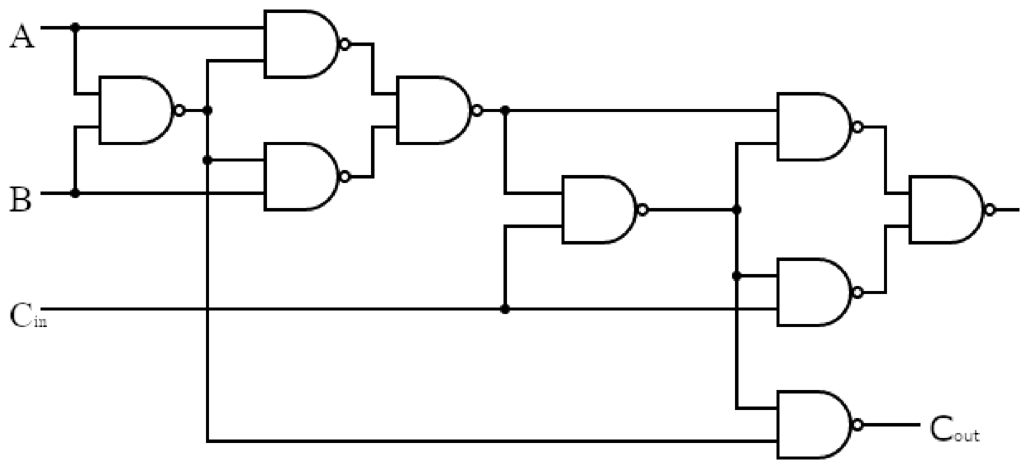
$\mathbf{A}$	$\mathbf{B}$	$\mathbf{A \barwedge \neg B}$	$\mathbf{\neg A \barwedge B}$	$\mathbf{(A \barwedge \neg B) \barwedge (\neg A \barwedge B)}$	$\mathbf{C_{in}}$	$\mathbf{S}$
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0
0	0	1	1	0	1	1
0	1	1	0	1	1	0
1	0	0	1	1	1	0
1	1	1	1	0	1	1

$$S = (C_{in} \barwedge (A \barwedge \neg B) \barwedge (\neg A \barwedge B)) \barwedge (\neg C_{in} \barwedge (A \barwedge \neg B) \barwedge (\neg A \barwedge B))$$

$\mathbf{A}$	$\mathbf{B}$	$\mathbf{C_{in}}$	$\mathbf{\neg(A \barwedge B)}$	$\mathbf{\neg A \barwedge \neg B}$	$\mathbf{O}$	$\mathbf{\neg(C_{in} \barwedge O)}$	$\mathbf{C_{out}}$
0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	0	1
0	0	1	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	1	1	1	0	1	1

$$\begin{aligned} A \wedge B &= \neg(A \barwedge B) \\ A \vee B &= \neg A \barwedge \neg B \\ O &= (A \barwedge \neg B) \barwedge (\neg A \barwedge B) = A \dot{\vee} B \\ C_{out} &= (\neg(\neg(A \barwedge B)) \barwedge \neg(\neg(C_{in} \barwedge O))) \end{aligned}$$

d) In a digital circuit, we can easily reuse common terms. Draw a small digital circuit implementing S and C_{out} using NAND gates only.



Problem 9.2

a) Let *op* be an associative operation with e as the neutral element:

```
op is associative: (x op y) op z = x op (y op z)
e is neutral element: e op x = x and x op e = x
```

Then the following holds for finite lists xs:

```
foldr op e xs = foldl op e xs
```

To prove this, I've tried two methods: one by going through the whole list and one by using induction.
First Proof

Proof.

$$\begin{aligned}
\text{foldl op e } (x_1, x_2, \ldots, x_n, x_{n+1}) &= ((\ldots(((\text{e op } x_1) \text{ op } x_2) \text{ op } x_3) \ldots \text{ op } x_n) \text{ op } x_{n+1}) \\
&= ((\ldots((x_1 \text{ op } x_2) \text{ op } x_3) \ldots \text{ op } x_n) \text{ op } x_{n+1}) \\
&= ((\ldots((x_1 \text{ op } (x_2 \text{ op } x_3) \ldots \text{ op } x_n) \text{ op } x_{n+1})^1 \\
&\vdots \\
&= (x_1 \text{ op } (x_2 \text{ op } \ldots (x_n \text{ op } x_{n+1}) \ldots)) \\
&= (x_1 \text{ op } (x_2 \text{ op } \ldots (x_n \text{ op } (x_{n+1} \text{ op e})) \ldots))^2 \\
&= \text{foldr op e } (x_1, x_2, \ldots, x_n, x_{n+1}) \\
&= \text{foldr op e xs}*
\end{aligned}$$

1 Apply associativity for each consecutive group of terms as above until we reach the end

2 Proof can be given just for x_n too as it requires one less step when applying associativity

* xs stands for the list $(x_1, x_2, \ldots, x_n, x_{n+1})$

■

Second Proof

Proof. In order to prove this using induction, we need to define the base case which should give the same result for both, *foldl* and *foldr*.
Base Case: Our base case in this case is that applying the foldl and foldr function to an empty list would give the empty list as result:

```
foldl op e [] = []
foldr op e [] = []
```

Therefore base case is proved:

```
foldl op e [] = foldr op e []
```

Induction Step:

```
foldl op e xs = foldr op e xs
```

Assume that the induction hypothesis is true for list of n elements. Proving that our foldl and foldr evaluate to the same value for $n+1$ elements. ($x : x_n$ stands for a list $n+1$ elements, while x_n is a list of n elements.
Starting from left hand side of the equation:

$$\begin{aligned}\text{foldl op e (x:}x_n\text{)} &= \text{op (op e x) (foldl op e } x_n\text{)}\\ &= \text{op x (foldl op e } x_n\text{)}\end{aligned}$$

Expanding the right hand side of the equation:

$$\begin{aligned}\text{foldr op e (x:}x_n\text{)} &= \text{op (op e x) (foldr op e } x_n\text{)}\\ &= \text{op x (foldr op e } x_n\text{)}\end{aligned}$$

Using induction hypothesis we can prove that both sides are equal:

$$\begin{aligned}\text{foldl op e (x:}x_n\text{)} &= \text{op (op e x) (foldl op e } x_n\text{)}\\ &= \text{op x (foldr op e } x_n\text{)}\\ &= \text{foldr op e (x:}x_n\text{)}\end{aligned}$$

∎

b) Let $op1$ and $op2$ be two operations for which:

```
x `op1` (y `op2` z) = (x `op1` y) `op2` z
x `op1` e = e `op2` x
```

holds. Then the following holds for finite lists xs:

```
foldr op1 e xs = foldl op2 e xs
```

To prove this, I've tried two methods: one by going through the whole list and one by using induction.
First Proof

Proof.

$$
\begin{aligned}
\text{foldl op2 e } (x_1, x_2, \ldots, x_n, x_{n+1}) &= ((\ldots(((\text{e op2 } x_1) \text{ op2 } x_2) \text{ op2 } x_3) \ldots \text{ op2 } x_n) \text{ op2 } x_{n+1}) \\
&= ((\ldots(((x_1 \text{ op1 e}) \text{ op2 } x_2) \text{ op } x_3) \ldots \text{ op } x_n) \text{ op } x_{n+1}) \\
&= ((\ldots((x_1 \text{ op1 } (\text{e op2 } x_2) \text{ op2 } x_3) \ldots \text{ op2 } x_n) \text{ op2 } x_{n+1})^1 \\
&\vdots \\
&= (x_1 \text{ op1 } (x_2 \text{ op1 } \ldots (x_n \text{ op1 } (\text{e op2 } x_{n+1}) \ldots)) \\
&= (x_1 \text{ op1 } (x_2 \text{ op1 } \ldots (x_n \text{ op1 } (x_{n+1} \text{ op1 e})) \ldots))^2 \\
&= \text{foldr op1 e } (x_1, x_2, \ldots, x_n, x_{n+1}) \\
&= \text{foldr op1 e xs}*
\end{aligned}
$$

1 Apply associativity for each consecutive group of terms as above until we reach the end

2 Proof can be given just for x_n too as it requires one less step when applying associativity

* xs stands for the list $(x_1, x_2, \ldots, x_n, x_{n+1})$

■

Second Proof

Proof. In order to prove this using induction, we need to define the base case which should give the same result for both, *foldl* and *foldr*.
Base Case: Our base case in this case is that applying the foldl and foldr function to an empty list would give the empty list as result:

```
foldl op1 e [] = []
foldr op2 e [] = []
```

Therefore base case is proved:

```
foldl op1 e [] = foldr op2 e []
```

Induction Step:

```
foldr op1 e xs = foldr op2 e xs
```

Assume that the induction hypothesis is true for list of n elements. Proving that our foldl and foldr evaluate to the same value for $n + 1$ elements.
Using induction hypothesis we can prove that right hand side is equal to the left hand side:

$$
\begin{aligned}
\text{foldl op2 e } (\text{x:}x_n) &= (\text{foldl op2 } (\text{op2 e x}) \; x_n) \\
&= (\text{foldl op2 } (\text{op1 x e}) \; x_n) \\
&= \text{op1 x } (\text{foldl op2 e } x_n) \\
&= \text{op1 x } (\text{foldr op1 e } x_n) \\
&= \text{foldr op1 e } (\text{x:}x_n)
\end{aligned}
$$

■

c) Let *op* be an associative operation and *xs* a finite list. Then

```
foldr op a xs = foldl op' a (reverse xs)
```

holds with

```
x op' y = y op x
```

To prove this, I've tried two methods: one by going through the whole list and one by using induction.

First Proof

Proof.

$$\begin{aligned}\text{foldr op a } (x_1, x_2, \ldots, x_n, x_{n+1}) &= (x_1 \text{ op } (x_2 \text{ op} \ldots (x_n \text{ op } (x_{n+1} \text{ op a})) \ldots))\\ &= (x_1 \text{ op } (x_2 \text{ op} \ldots (x_n \text{ op (a op' } x_{n+1})) \ldots))^1\\ &= (x_1 \text{ op } (x_2 \text{ op} \ldots (\text{a op' } x_{n+1}) \text{ op' } x_n)) \ldots))^1\\ &\vdots\\ &= (((\ldots((\text{a op' } x_{n+1}) \text{ op } x_n) \text{ op'} \ldots) \text{ op' } x_2) \text{ op' } x_1)\\ &= \text{foldl op' a (reverse xs)}*\end{aligned}$$

1 Apply formula x op' y = y op x

* xs stands for the list $(x_1, x_2, \ldots, x_n, x_{n+1})$

■

Second Proof

Proof. In order to prove this using induction, we need to define the base case which should give the same result for both, *foldl* and *foldr*.

Base Case: Our base case in this case is that applying the foldl and foldr function to an empty list would give the empty list as result:

```
foldr op a [] = []
foldr op' a (reverse []) = []
```

Therefore base case is proved:

```
foldr op a [] = foldr op' a (reverse []) = []
```

Induction Step:

```
foldr op a xs = foldl op' a (reverse xs)
```

Assume that the induction hypothesis is true for list of n elements. Proving that our foldl and foldr evaluate to the same value for $n + 1$ elements.

Using induction hypothesis we can prove that right hand side is equal to the left hand side:

$$\begin{aligned}\text{foldl op' a (reverse (xs))} &= \text{foldl op' a (reverse (x:}x_n))\\ &= (\text{foldl op' a (reverse } x_n) \text{ x})\\ &= (\text{foldl op' a (reverse } x_n)) \text{ op' x}\\ &= \text{x op (foldr op a } x_n)\\ &= \text{foldr op a (x:}x_n)\end{aligned}$$

■