

CSE 460: Lecture 8

Data Structures and Computational Techniques for Tabular Method

John W. Lockwood

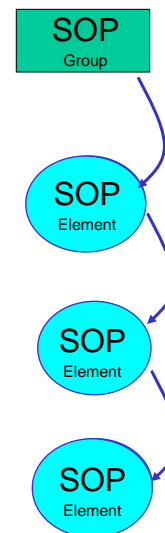
Washington University

Spring 2006

<http://www.arl.wustl.edu/~lockwood/class/cse460/>

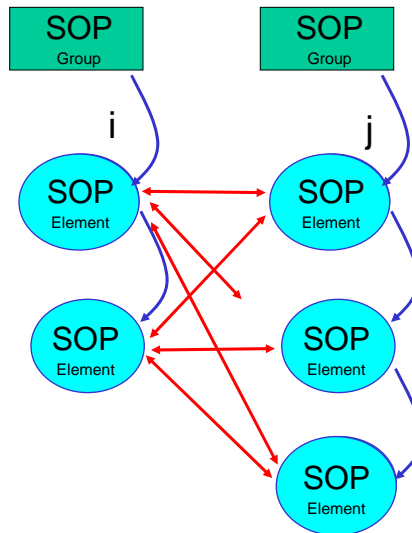
SOP Group

- SOP Group = Collection of SOP Elements
 - Allow scanning of elements
 - Allow insertion of unique members
 - Avoid duplicates
 - Allow modification of group members.



Comparing SOP Groups

For each $SOP_i \in \text{Group 1}$
 For each $SOP_j \in \text{Group 2}$
 Compare SOP_i to SOP_j



SOP Group Matrix : In class example

- Consider $f(w,x,y,z)$
 - Four-variable Function
- Arrange columns by Dontcare_count
 - $i \in \{0 \dots n\}$
 - $n+1$ columns
- Arrange rows by One_Count
 - $j \in \{0 \dots n\}$
 - $n+1$ rows

	DontCare Count				
	DC ₀	DC ₁	DC ₂	DC ₃	DC ₄
OC ₀	0000	000- 0-00			
OC ₁	0001 0100	00-1 -001 -100			
OC ₂	0011 1001 1100				
OC ₃					
OC ₄	1111				

SOP Group Matrix : In class example

- For each $DC \in \{0..3\}$
- For each $OC \in \{0 .. DC-1\}$
 - Compare
 - SOP_Group (DC,OC) to SOP_Group (DC,OC+1)
 - If (Difference=1 term),
 - Insert to SOP_Group (DC+1,OC)
 - Unmark Primes

		DontCare Count				
		DC ₀	DC ₁	DC ₂	DC ₃	DC ₄
One Count	OC ₀	0000	000- 0-00			
	OC ₁	0001 0100	00-1 -001 -100			
	OC ₂	0011 1001 1100				
	OC ₃					
	OC ₄	1111				

Extending program to find Prime Implicants

```
class sop_term {
public:
    // Homework 2 structure
    unsigned int value; // up to 32-bits: 0=false, 1=true
    unsigned int mask; // up to 32-bits: 0=Care, 1=DontCare
    // Homework 3 extensions
    int one_count;
    int dontcare_count;
    int is_prime;
    sop_term* next;
};
```

Your structures may vary!

Cygwin Tool Environment

- Provides
 - 32-bit C/C++ GNU software
 - gcc / g++ / gdb / Perl, Java
 - Runs on
 - Win95, Win98, WinME, NT, XP
 - Apps source-code compatible
 - With tools in Linux/UNIX
- Download from
 - <http://www.cygwin.com/>
- Freeware (GPL)



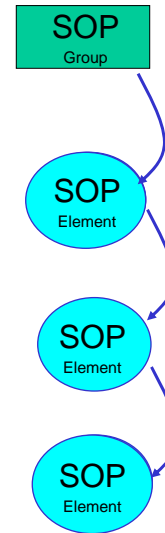
Suggestions for Software Implementation

- Create plan for the software design
 - Define: Class / Data structures
 - Define: Functions / Methods
- Implement one component at a time
- Debug each component
- Display intermediate steps

Implementation of Lists and Groups

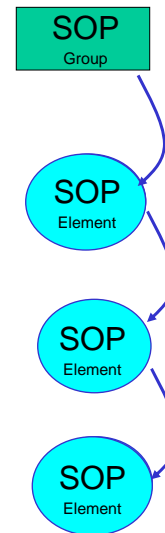
- Iterating through a group
 - Start at the Head of the list
 - Iterate through each term

```
~/coe460/mp/mp4
void sop_group::display() {
    sop_term* si = head;
    if (head==NULL)
        return;
    else
        do {
            if (si->is_prime)
                si->display();
            si = si->next;
        } while (si!=NULL);
}
```



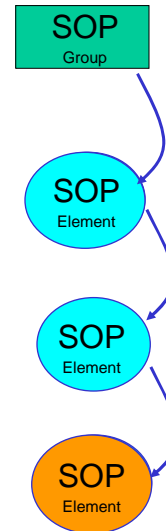
SOP Group

- SOP Group = Collection of SOP Elements
 - Allow scanning of elements
 - Allow insertion of unique members
 - Avoid duplicates
 - Allow modification of group members.



Implementation of Lists and Groups

- Adding an element to a list
 - Avoid inserting same term twice
 - Use `Is_Equal(s1, s2)`
- Cases to Consider:
`sop_group::append(sop_term *sin)`
 - Empty List?
 - Head of list `Is_Equal` new element?
 - Any elements in list = new element?
 - Append new element to end of list



Implementation Example (1)

- Implement program
- Compile
 - `g++ -o <output> <source.cpp>`
- Generate input
 - Edit data input file
- Run Program
- Generate Minterms

```
lockwood@WJL ~/coe460/mp/mp4
$ g++ -o parsevar parsevar.cpp
lockwood@WJL ~/coe460/mp/mp4
$ cat datain
3
---
done
lockwood@WJL ~/coe460/mp/mp4
$ parsevar.exe <datain
Function has 3 variables
==== Generate Minterms ====
000
001
010
011
100
101
110
111
```

Processing of SOP Groups

	Initial Terms	DontCare Count = 1	DontCare Count = 2
One_Count=0	SOP Group	SOP Group	SOP Group
One_Count=1	SOP Group	SOP Group	
One_Count=2	SOP Group		

Implementation Example (2)

- Minterm Expansion Example
 - 3 variable function = “---”
 - Expand into Minterms
 - (Same as before)
- Classify Minterms by one_count
- Insert sop_terms into Table
 - Rows : one_count
 - Columns : dontcare_count
 - Avoid duplicates

```

===== Count one terms =====
one_count=0
000
001
one_count=1
010
011
one_count=2
100
101
one_count=1
110
one_count=2
111
one_count=3

===== Create Table =====
Table[0][0] =
000
Table[0][1] =
001
010
100
Table[0][2] =
011
101
110
Table[0][3] =
111
Table[1][1][0] =
Table[1][1][1] =
Table[1][1][2] =
Table[1][2][0] =
Table[1][2][1] =
Table[1][2][2] =
Table[1][2][3] =
Table[1][3][0] =
Table[1][3][1] =
Table[1][3][2] =

```

Implementation Example (3)

- Combine Groups in Table
 - That differ by a single value
 - Unmark primes that combine
 - Insert result into next column
- Display table

```

===== Merge Groups in Table =====
Merge[0][0] with [1][0] to get [0][1]
Compare:
000
001
Appending to Table[0][1]
00
Compare:
000
001
Appending to Table[0][1]
00
Compare:
000
100
Appending to Table[0][1]
00
== Round[0] ==
Table[0][0] =
Table[0][1] =
Table[0][2] =
011
101
Table[0][3] =
111
Table[1][0] =
000
000
Table[1][1][0] =
Table[1][1][1] =
Table[1][1][2] =
Table[1][1][3] =
Table[1][1][4] =
Table[1][1][5] =
Table[1][1][6] =
Table[1][1][7] =
Table[1][1][8] =
Table[1][1][9] =
Merge[1][0] with [2][0] to get [1][1]
Compare:
001
011
Appending to Table[1][1]
001
Compare:
001
101
Appending to Table[1][1]
001
Compare:
001
110
Compare:

```

SOP Group Matrix : In class example

- For each $DC \in \{0..3\}$
- For each $OC \in \{0 .. DC-1\}$
 - Compare
 - SOP_Group (DC,OC) to SOP_Group (DC,OC+1)
 - If (Difference=1 term),
 - Insert to SOP_Group (DC+1,OC)
 - Unmark Primes

DontCare Count

	DC ₀	DC ₁	DC ₂	DC ₃	DC ₄
OC ₀	0000 000- 0-00				
OC ₁	0001 0100	00-1 -001 -100			
OC ₂	0011 1001 1100				
OC ₃					
OC ₄	1111				

One Count

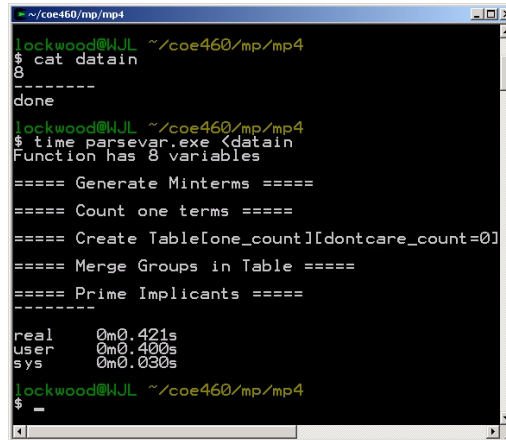
Performance & Measurement Analysis

- Performance Testing

- Reduce I/O for Benchmarking Code
- C/C++ Debugging
 - `#define DEBUG(x) x`
- C/C++ Performance
 - `#define DEBUG(x)`

- Measurement

- Unix `time` command
- `time <program> [arguments]`
- Report user time



```
lockwood@WJL ~/coe460/mp/mp4
$ cat datain
8
-----
done
lockwood@WJL ~/coe460/mp/mp4
$ time parsevar.exe <datain
Function has 8 variables
==== Generate Minterms ====
==== Count one terms ====
==== Create Table[one_count][dontcare_count=0]
==== Merge Groups in Table ====
==== Prime Implicants ====
-----
real    0m0.421s
user    0m0.400s
sys     0m0.030s
lockwood@WJL ~/coe460/mp/mp4
$ -
```