

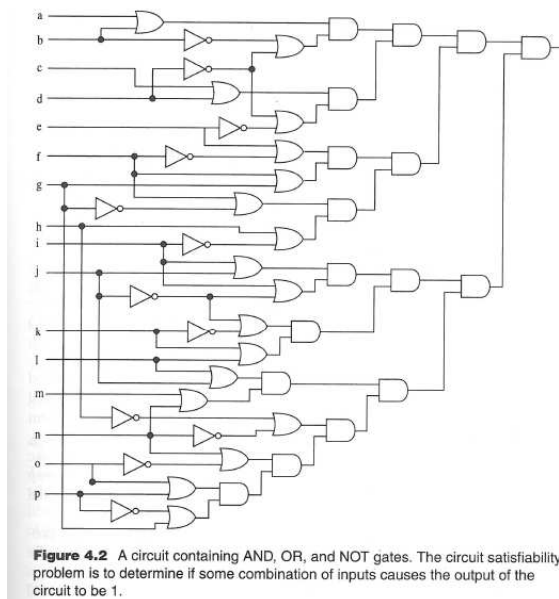
# INF3380 Exercise Set 6

for lecture “Recap of MPI programming”

## Exercise 1

Write a simple MPI program to measure the overhead of point-to-point communication involving `MPI_Send` and `MPI_Recv`. (Hint: you can start two MPI processes, between which a message is bounced back and forth a number of times.)

## Exercise 2 (circuit satisfiability)



- There are 16 input ports, each accepts a Boolean value (**true** or **false**);
- Total number of combinations:  $2^{16} = 65,536$ ;
- *Sketch* an MPI program to try out all the 65,536 combinations in parallel.

**Hint:** If we denote the Boolean values assigned to the 16 input ports as `b0`, `b1`, `b2`, ..., `b15`, then the above figure is equivalent with finding whether a

Boolean circuit of form

$(b_0 \mid\mid b_1) \ \&\& \ (!b_1 \mid\mid !b_3) \ \&\& \ (b_2 \mid\mid b_3) \ \&\& \ (!b_3 \mid\mid !b_4) \ \dots$

evaluates to `true` with suitable combinations of the 16 input Boolean values.

### Exercise 3

Write an MPI program that parallelizes the following numerical computation:

```
#include <malloc.h>
#include <math.h>

/* allocating three 1D arrays um, u, up of length M+2 */
/* ... */

double x, dx = 1.0/(M+1);
double t, dt = dx;
double *tmp;
int i;

for (i=0; i<=M+1; i++) {
    x = i*dx;
    um[i] = sin(2.0*M_PI*x);
}

for (i=1; i<=M; i++)
    u[i] = um[i] + 0.5*(um[i-1]-2*um[i]+um[i+1]);

u[0] = u[M+1] = 0.0;

t = dt;
while (t<1.0) {
    t += dt;
    for (i=1; i<=M; i++)
        up[i] = um[i]+u[i-1]+u[i+1];
    up[0] = up[M+1];

    /* shuffle the three arrays */
    tmp = um;
    um = u;
    u = up;
    up = tmp;
}
```

**Hint:** The above computation may arise from numerically solving a very simple 1D wave equation. Ignoring the mathematical and numerical details, it is sufficient to notice that three arrays are involved, where computing values in array `up` needs values from arrays `u` and `um`. In particular, values of array `up` can be computed in any random order (thus the existence of parallelism).