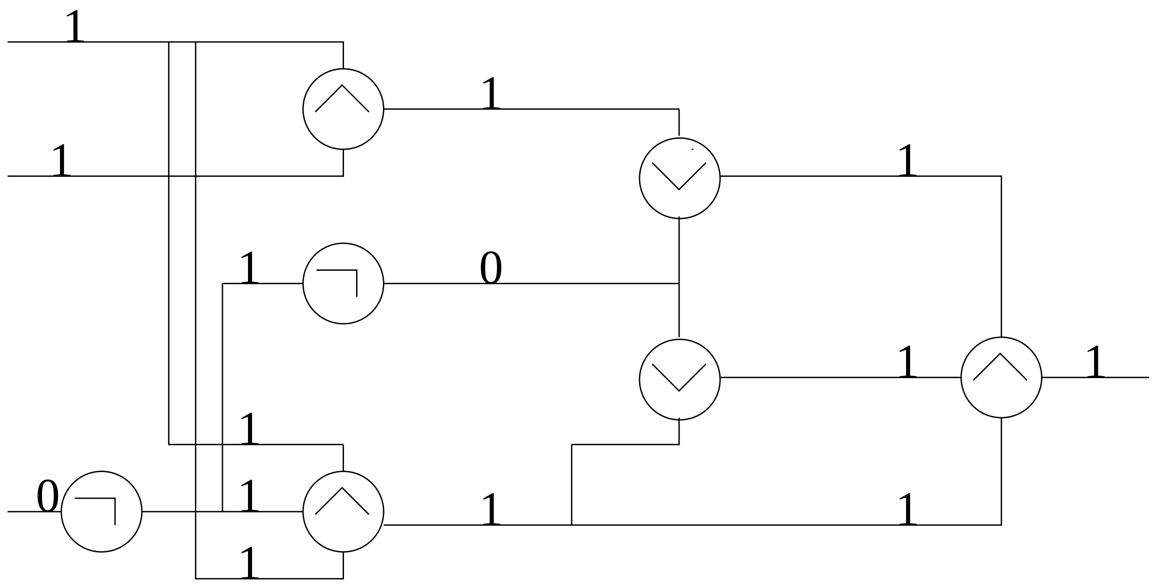
Circuit Satisfiability

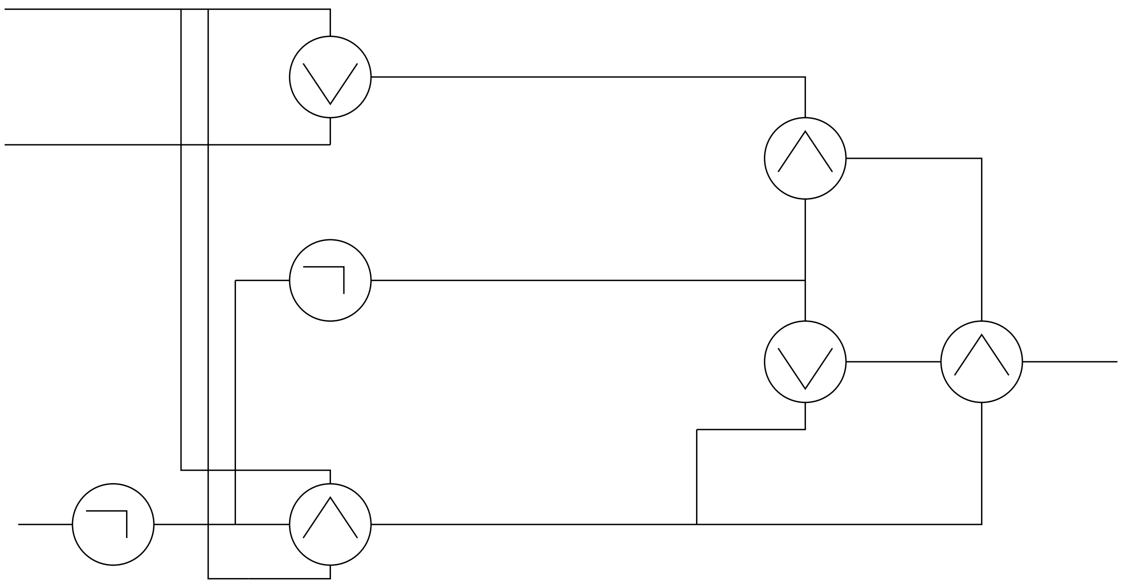
- Upto this point we have been avoiding the main issue.
- To prove a problem that belongs to the class $\mathcal{NP}$ complete.
- We can of course use reducibility as a tool to prove $\mathcal{NP}$ completeness of the many problems.
- But we must at least prove the class $\mathcal{NP}$ complete is not empty by showing one problem that is $\mathcal{NP}$ complete.
- The formal proof of this is beyond the socpe of the present course.
- We can just discuss this informally.
- Circuit satisfiability is one such problem.

Circuit Satisfiability



- The circuit is satisfiable if we assign 1, 1, 0 to input line from top to bottom

Circuit Satisfiability



- The circuit is not satisfiable. Check with all 8 possible truth value assignements.

Circuit Satisfiability

- We have already proved that there is an $\mathcal{NP}$ algorithm for this problem.
- So, circuit satisfiability problem is in $\mathcal{NP}$.
- To provide an informal proof that C-SAT is $\mathcal{NP}$ complete, we need some basic understanding about h/w.
- A computer program is stored in memory as a sequence of instructions.
- A special memory location called `program counter` keeps track of the instruction to be executed next.
- PC is auto-incremented whenever an instruction is fetched.
- This causes the instruction to be executed sequentially.

Memory Configuration

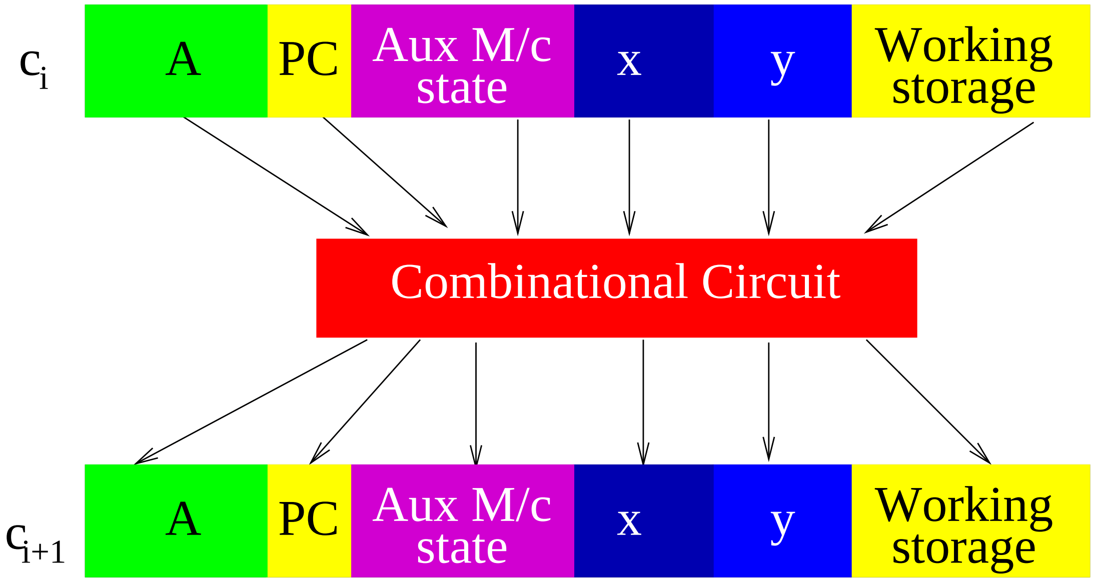
- PC can be changed by the execution of an instruction which causes a value to be written to it.
- This can cause a conditional branch or a loop breaking from the normal sequential execution of the instructions.
- At any point of time the entire state of the computation can be represented by the contents of computer memory.
- We can refer to particular state of computer memory as a configuration.
- Execution of an instruction, therefore, can be viewed as mapping one configuration to next.

C-SAT is NP Complete

- We need to prove any string of a language L in $\mathcal{NP}$ can be converted to as a string of C-SAT language in p-time.
- Since $L \in \mathcal{NP}$ there exists a $\mathcal{NP}$ algorithm $\mathbf{A}$ which runs in p-time.
- Let $T(n)$ be worst case time of $\mathbf{A}$ on n-length input string from L.
- So $T(n) = O(n^k)$ for some constant $k \geq 1$.
- Computation $\mathbf{A}$ is a sequence of confgiurations.
- Each configuration can be partitioned as
  - PC, program for $\mathbf{A}$,
  - Input string, guess string,
  - Working storage, auxiliary machine state.

C-SAT is NP Complete

- Essentially the mapping from one configuration to other is indicated as follows:



- A combinational circuit that implements the computer h/w maps a configuration c_i to c_{i+1}.
- Output (0 or 1) is written at some designated location of working storage when $\mathbf{A}$ completes.

C-SAT is NP Complete

- Since algorithm requires time of $T(n)$, with starting configuration c_0 the output appears in $c_{T(n)}$.
- Let reduction F construct a single combinational circuit that computes all configurations $c_1, c_2, \ldots, c_{T(n)}$ from given initial configuration c_0.
- Basically, the $T(n)$ copies of circuits that leads from c_0 to $c_{T(n)}$ are glued together by having the output of ith circuit which produces configuration c_i fed as input to combinational $i+1$st circuit producing configuration c_{i+1}.
- Clearly F is p-time as $T(n)$ is.
- So, given a string x we have produced a reduction F which computes a circuit $\mathbf{C}=f(x)$ in p-time such that it is satisfiable if algorithm $\mathbf{A}$ with guess y produces A(x)=1.

C-SAT is NP Complete

- Let us reiterate what reduction F does.

Given input string x it must compute $C = f(x)$ which is satisfiable if and only if there is an $\mathcal{NP}$ algorithm $\mathbf{A}$ for accepting x.

- When F obtains an input x it first computes $n = |x|$ and construct a combinational circuit C' consisting of $T(n)$ copies of circuits as indicated earlier.
- The input to C' is an initial configuration corresponding to a computation $\mathbf{A}$ with guess y.
- The output is $c_{T(n)}$.

C-SAT is NP Complete

- The circuit $C = f(x)$ that F computes is obtained by modifying C'.
  - The inputs to C' corresponding to the program $\mathbf{A}$, the initial PC, the input string x, the initial state of memory wired directly to the known values.
  - The only remaining input is the guess string y.
  - The output of $c_{T(n)}$ except the one bit corresponding to output of $\mathbf{A}$ are ignored.
- Now the circuit C constructed as such computes $C(y) = 1$ if $A(x) = 1$.
- Clearly, y can be considered as input values that satisfies the circuit C.
- Thus, $\mathbf{A}$ accepts x only when there is an assignment of values to input lines of circuit C such that it becomes satisfiable.

C-SAT is NP Complete

- As L is an arbitrary language in $\mathcal{NP}$, we have provided a proof of the fact that any recognition of language in class $\mathcal{NP}$ reduced to C-SAT problem.
- The reduction is obviously polinomial.
- Hence, C-SAT $\mathcal{NP}$ hard.
- Therefore, C-SAT is $\mathcal{NP}$ complete.

