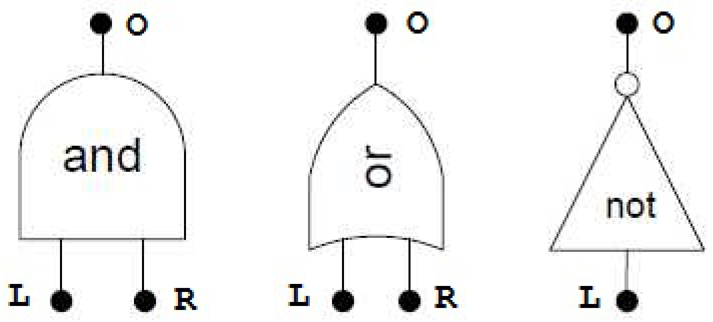
Metamodel of Logic Circuits

A **logic diagram** consists of the following symbols (AND, OR, NOT) called **gates**:

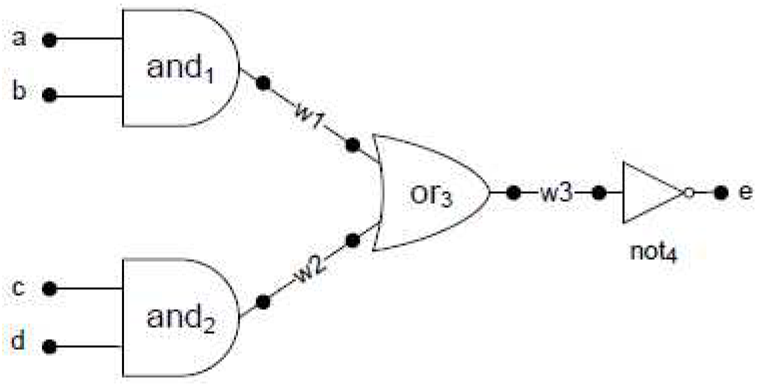


Every **gate** has has pins. The **and** and **or** gates above has an output pin (**O**) and left and right input pints (**L,R**). The **not** gate has a single input pin (**L**) and a single output pin (O). All gates have unique names.. Pin names are inferred -- e.g., gate **and** has pins **and1.L**, **and1.R**, and **and1.O**.

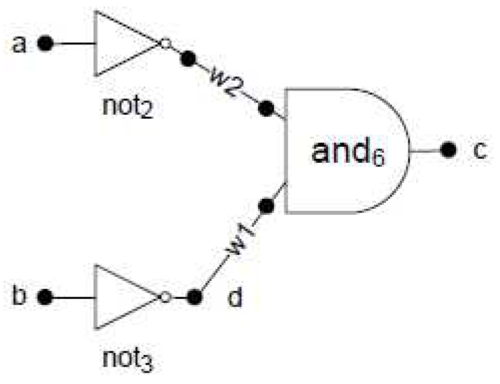
A **wire** connects the output pin of one gate to an input of one or more gates. Note, there can be only ONE wire connected to an input pin, although MANY wires can leave an output pin.

A **port** is a labeled pin. A port serves as the input or output (never both) of a circuit. All ports have unique names.

A logic diagram **D1** is shown below. There are 4 input ports (**a**, **b**, **c**, **d**) and one output port (**e**). There are 4 gates: **and1**, **and2**, **or3**, **not4**. Port **c = and2.L**. Port **e = not4.O**. There are 3 wires (**w1**, **w2**, *w3*). Wire **w2** connects **and2.O** to **or3.R**. Wire W3 connects **or3.O** to **Not4.L**.



Here is diagram **D2** with 4 ports, three gates and 2 wires. Note that port **d** is an output port (i.e., an output pin that is labeled "**d**"). Port a is an input port (i.e., an input pin that is labeled "**a**").



What to Submit to Canvas

You are to create an MDE metamodel (class diagram + constraints) for which each of the above logic circuts are models (instances).

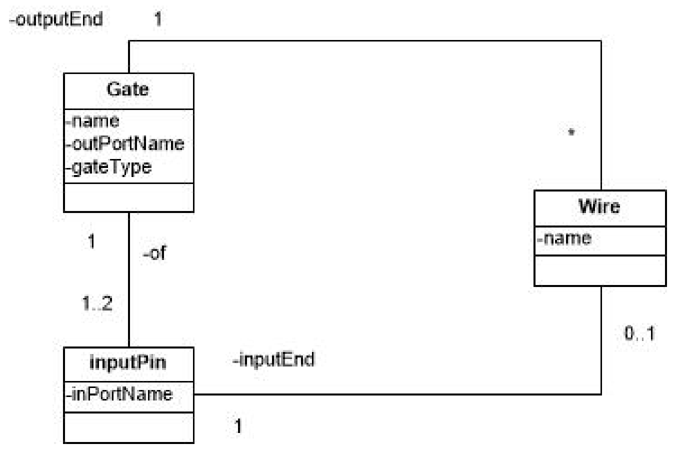
Submit a PDF with the following information:

1. define a UML class diagram to any logic circuit composed of and, or, and not gates.
2. derive the database tables (no rows, only attributes) of your class diagram.
3. populate your database with rows for diagram D1.
4. populate another database with rows for diagram D2.
5. List constraints (in English) that would you want to enforce on your database.

A Solution

As we will see in class, this is one solution. There are many (equivalent) solutions.

1. A UML class diagram for logic circuits composed of **and**, **or**, and **not** gates is:



Note:

- input pins are treated differently from output pins in that there can be several (1..2) input pins. There is ALWAYS one output pin. No need to have a separate object for an output pin when the gate itself suffices. (Yes, I suppose you can argue that this is an optimization. But it does remove unnecessary redundancy).
- A pin becomes a PORT when it is given a name. An empty (string) name means there is no name or “null”.

2. Table definitions:

gate

G#	name	outPortName	gatetype

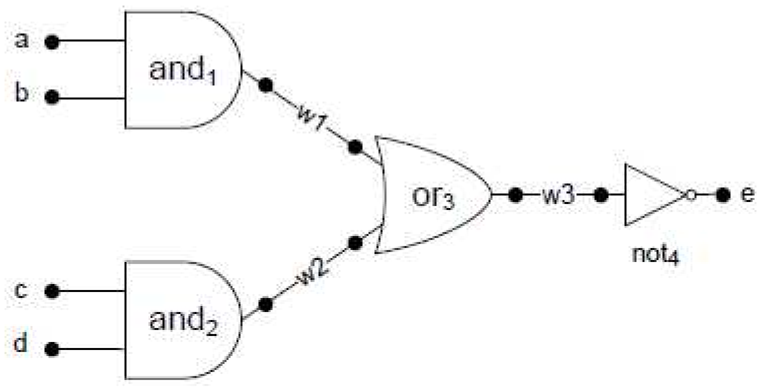
inputPin

IP#	inPortName	of (G#)

wire

W#	name	inputEnd (G#)	outputEnd (IP#)

3. Diagram D1 database



gate

G#	name	outPortName	gatetype
a1	and1		and
a2	and2		and
o3	or3		or
n4	not4	e	not

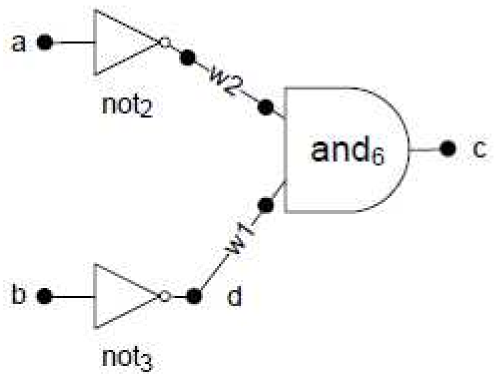
inputPin

IP#	inPortName	of (G#)
a1L	a	a1
a1R	b	a1
a2L	c	a2
a2R	d	a2
o3L		o3
o3R		o3
n4L		n4

wire

W#	name	inputEnd (G#)	outputEnd (IP#)
w1	w1	a1	o3L
w2	w2	a2	o3R
w3	w3	o3	n4L

4. Diagram D2 database:



G#	name	outPort Name	gatetype
n2	not2		not
n3	not3	d	not
a6	and6	c	and

IP#	inPortName	of (G#)
n2L		n2
n3L	c	n3
a6L	d	a6
a6R		a6

W#	name	inputEnd (G#)	outputEnd (IP#)
w1	w1	n3	a6L
w2	w2	n2	a6R

5. Constraints

- Every wire connects an input pin to an output pin.
- Input pins can only be connected to one wire
- Gate ids, names are unique
- Gate type is an enum {and, or, not}
- Input pins id, names are unique
- Wire ids, names are unique
- And gates have 2 input pins
- Or gates have 2 input pins
- Not gates have one input pin
- All gates have one output pin (although this need not be checked as it is hardwired).