

Homework # 5

Due Wednesday, March 8, 2006, at 1:30 PM
Collaboration is allowed and encouraged

1. Composition of Relations

In this problem, let the relations r , s , t , $\neg\blacktriangleleft$, $\textcircled{1}$, and $\textcircled{0}$ be defined as follows:

$$\begin{aligned} r &= \{\langle 1, 0, 0 \rangle, \langle 0, 1, 1 \rangle, \langle 0, 1, 0 \rangle, \langle 0, 0, 1 \rangle\} \\ s &= \{\langle 1, 1, 1 \rangle, \langle 1, 1, 0 \rangle, \langle 1, 0, 1 \rangle, \langle 1, 0, 0 \rangle, \langle 0, 1, 1 \rangle, \langle 0, 1, 0 \rangle, \langle 0, 0, 1 \rangle\} \\ t &= \{\langle 1, 0, 0 \rangle, \langle 0, 1, 1 \rangle, \langle 0, 0, 1 \rangle\} \\ \neg\blacktriangleleft &= \{\langle 1, 1, 1 \rangle, \langle 0, 0, 0 \rangle\} \\ \textcircled{1} &= \{\langle 1 \rangle\} \\ \textcircled{0} &= \{\langle 0 \rangle\} \\ \odot &= \{\langle 0 \rangle, \langle 1 \rangle\} \end{aligned}$$

You will probably want to develop your own notation for drawing r , s , and t .

- Show that r can implement s , and show that t can implement $\neg\blacktriangleleft$. *To show an implementation, draw the network of relations and say in one or two sentences how the circuit works.*
- Show that r cannot implement $\neg\blacktriangleleft$. *Hint: Show that any circuit where $\{r, \textcircled{0}, \textcircled{1}\}$ implements $\neg\blacktriangleleft$ can be reduced to a circuit with fewer r relations. Consider wire values near one of the wires leaving the circuit.*
- Show that s cannot implement t . *Hint: Think about parts (a) and (b).*
- In figure 1 we see a graphical representation of which Boolean relations on three variables can implement which others. Do the implementations (and the impossibility of other implementations) that you found in parts (a) through (c) match the information in the figure? Suppose that you have found an implementation for every line in the figure, but you have not yet proved the impossibility of any implementation except that in part (b). Based on the lines in the figure and your result from part (b), what relations must be impossible for s to implement?
- Show that $\{\langle 1, 1, 0, 0, 0 \rangle, \langle 0, 0, 1, 1, 0 \rangle, \langle 0, 0, 0, 0, 1 \rangle\}$ is a universal relation, meaning that it can be used to implement any Boolean relation of any number of variables. How would it fit into the diagram?

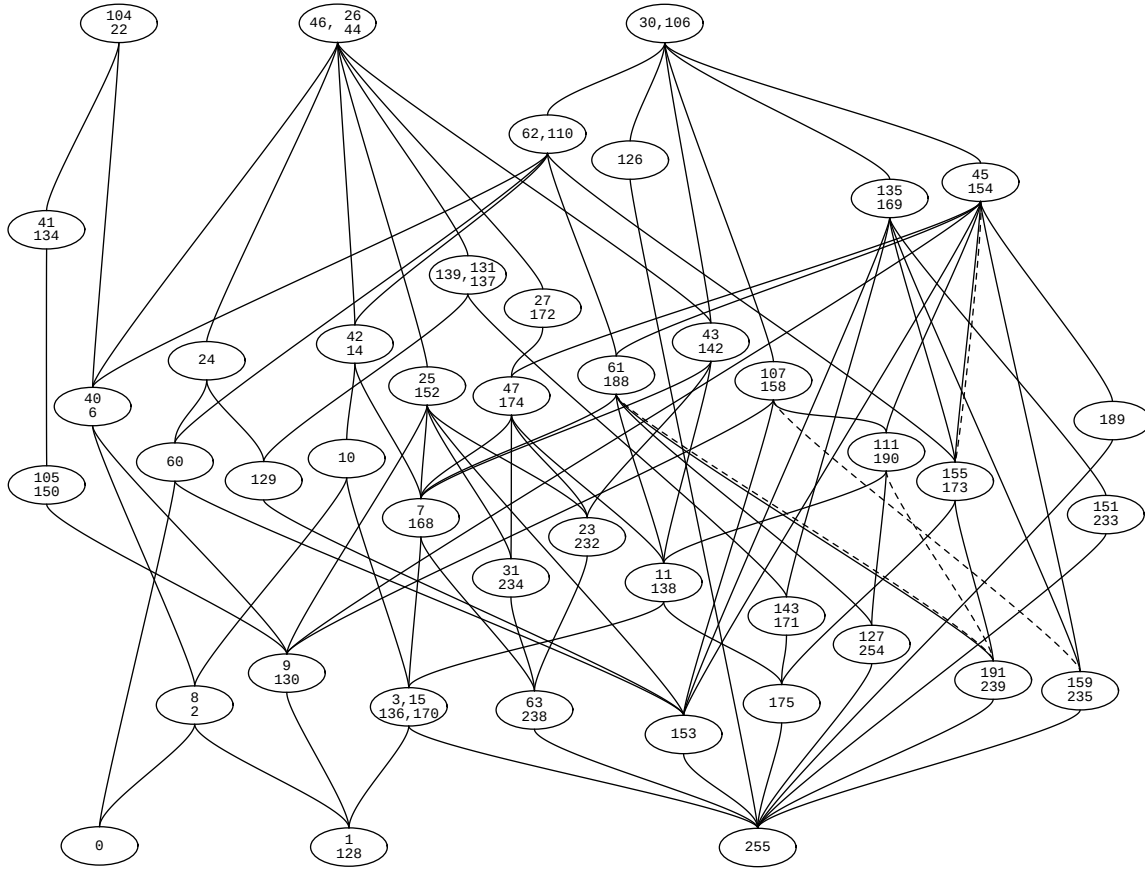


Figure 1: Which ternary boolean relations can implement which others. Each number indicates a relation by treating each acceptable triple as a three digit binary number, and then calculating $\sum_{n \in \text{triples}} 2^n$. For example, the number 30 (in the top right oval) is $2^4 + 2^3 + 2^2 + 2^1$, so the triples accepted by that relation are $\langle 1, 0, 0 \rangle$, $\langle 0, 1, 1 \rangle$, $\langle 0, 1, 0 \rangle$, and $\langle 0, 0, 1 \rangle$ (this is the relation $A = \text{NOR}(B, C)$). Where two relations can implement each other, they are shown in the same oval, separated by a comma. A surprising feature of the hierarchy is that this is not very common. If two relations are dual to each other (by flipping 0's and 1's), then they are shown in the same oval, one above the other, in which case the oval really represents two distinct points in the hierarchy, one for each relation. Lines between two such ovals only indicate implementability between the upper relations in each oval, and between the lower relations in each oval. A dotted line indicates crossing to the other side of the duality symmetry, so the upper relation in the upper oval can implement the lower relation in the lower oval, and the lower relation in the upper oval can implement the upper relation in the lower oval (these two implementations being equivalent by duality). Where three relations are listed in an oval, the two atop each other are duals of each other, but each of the three can implement the other two. The oval containing four relations represents two points in the hierarchy: one for the upper two relations, and one for their duals, the lower two relations.

FAQ on implementations:

- Self-loops (edges from a vertex to itself) are allowed.
- Multiple edges (more than one edge between the same two vertices) are allowed.
- Disconnected networks (where some vertices have no path of wires connecting them) are allowed.
- Ignored wires (a dangling edge which is not considered to be a variable of the implemented relation) are not allowed. If you want to ignore a wire, you must implement the happy relation on one variable (the “don’t care” relation, $\odot$).
- Constant-valued wires ($\textcircled{0}$ and $\textcircled{1}$) are not available by default. If you need a constant, you must implement it.
- Fan-out ($\neg\blacktriangleleft$) is not available by default. If you need it, you must implement it.

2. Full Networks

In this problem we will use Full Networks to decide whether some relations can implement another. Since you showed in problem 1a that the relation

$$t = \{\langle 1, 0, 0 \rangle, \langle 0, 1, 1 \rangle, \langle 0, 0, 1 \rangle\}$$

can implement $\neg\blacktriangleleft$, we know that $\neg\blacktriangleleft$ can be freely used in networks built out of t relations, and the availability of $\neg\blacktriangleleft$ is what allows the Full Network to be used.

- Draw the Full Network on 2^1 wires for $\{t, \neg\blacktriangleleft\}$. Can t implement $\textcircled{1}$? Can t implement $\textcircled{0}$?
- Draw the Full Network on 2^2 wires for $\{t, \neg\blacktriangleleft\}$. Are there any unimplementable relations accepting two tuples?
- Draw the Full Network on 2^3 wires for $\{t, \neg\blacktriangleleft\}$. Explain how you are sure you didn’t forget any part of your Full Network. Are there any unimplementable relations accepting three tuples?

3. Guessing Secrets

In the game “20 questions,” you (the “seeker”) try to find out what object your friend (the “adversary”) is thinking of, by asking yes/no questions. Consider a variant of this game, where your friend is allowed to think of any number of objects (the “secret set” of objects), rather than just a single secret object. Then, for any question you ask, your friend gets to answer the question (truthfully) based on any one of the objects in the secret set (whichever one they want). Your task is to try to figure out what the secret set is.

Assume the world contains n objects, and for any set of objects, there is a yes/no question of the form, “Is it in the set?” Assume that the adversary always gives the same answer if asked the same question multiple times.

For example, suppose the n objects are the backpacks in the classroom. Your friend might pick the set of backpacks on the floor to be the secret set. Then you might ask “Is it blue?” and your friend would pick one of the backpacks on the floor to use as the basis of the answer to this question. Your friend will have to answer “yes” if all of the backpacks on the floor are blue, and “no” if none of them are blue. If some of the backpacks on the floor are blue and some are not, then your friend might say “yes” or might say “no,” but the answer to this particular question will always be the same. You might then ask “Is it in the front row?” and your friend would again pick some backpack (maybe the same one, maybe a different one) in the secret set to serve as the basis for the answer to this question.

If your friend always uses the same backpack as the basis for the answer to all of your questions, then you will be able to figure out which backpack that is, but you will not be able to figure out that the secret set of backpacks is those that are on the floor. All you will know is that the secret set of backpacks, whatever it is, must include that particular backpack. If you consider all such sets of backpacks (i.e., all sets containing that particular backpack), then you are considering a set of sets, which we will call a family of sets so as not to get too confused. This family of sets has the property that you can never figure out which member of the family is your friend’s secret set.

- (a) Show that the adversary can pick three objects a, b, c and then answer all your questions so that you cannot tell whether the secret set is $\{a, b\}$ or $\{b, c\}$ or $\{a, c\}$, although you can tell that it must be one of these or a superset of one of these.
- (b) Show that the adversary can pick any family of intersecting sets (that is, any two sets in the family have a non-empty intersection), and then answer all of your questions so that you cannot tell which of these sets the adversary is using as the secret set, although you can tell that one of these sets (or a superset of one of these sets) must be the secret set. (Part (a) used such a family of three intersecting sets.)
- (c) We can think of each set as a bitvector where the k ’th bit indicates whether the k ’th object is in the set. We can then think of a family of intersecting sets as a function from bitvectors to $\{0, 1\}$, where 0 means not in the family and 1 means in the family. Show that the function for any given family of intersecting sets will preserve the NAND relation (the relation $\{\langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 0 \rangle\}$), and conversely, any function that preserves the NAND relation corresponds to a family of intersecting sets.
- (d) Show that a Boolean function is NAND-preserving iff it has the property that when written in a sum-of-products form, any two terms in the sum share an unnegated variable. (*Note: “iff” means “if and only if”*)
- (e) *Extra credit:* Show that a maximal family of intersecting sets (i.e., a family where no more sets can be added without destroying the intersecting set property of the family) corresponds to a function that preserves both NAND and OR (where OR is the relation $\{\langle 0, 1 \rangle, \langle 1, 0 \rangle, \langle 1, 1 \rangle\}$).