School of Computing and Information Systems
COMP30026 Models of Computation Problem Set 4

16–20 August 2021

Content: Solving problems with propositional logic (part 1), translating natural language to predicate logic (part 2).

Part 1 - Propositional Logic

P4.1 On the Island of Knights and Knaves, everyone is a knight or knave. Knights always tell the truth. Knaves always lie. You meet three people, let us call them A, B and C. A says to you: "If I am a knight then my two friends here are knaves." Use propositional logic to determine whether this statement gives you any real information. What, if anything, can be deduced about A, B, and C?

P4.2 Low-level programming languages (including C) offer bitwise operators, including bitwise negation, conjunction, disjunction and exclusive or. Bitwise means the logical operation is applied bit for bit, across whole words, with 0 representing false and 1 representing true. In such languages an expression like $53 \oplus 42$ evaluates to 31, because the exclusive or operator $\oplus$ is applied bit by bit to the binary representations of 53 and 42:

Assignment Project Exam Help

https://eduassistpro.github.io/

$$\begin{array}{cccccccc} \hline \ldots & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

and this yields the binary representation of 31.

Add WeChat edu_assist_pro

When the task is to swap the contents of two registers on a mach gisters available, an old idea is the "xor trick". To swap the contents of R_1 and R_2, rather than call upon a third register, we can perform three xor operations on R_1 and R_2.

$$\begin{array}{rcl} R_1 & := & R_1 \oplus R_2 \\ R_2 & := & R_1 \oplus R_2 \\ R_1 & := & R_1 \oplus R_2 \end{array}$$

Explain why that has the desired effect.

P4.3 Given these six formulas:

$$\begin{array}{ll} F_1: & (P \Rightarrow \neg Q) \wedge (P \Rightarrow \neg R) \\ F_2: & (P \Rightarrow \neg Q) \vee (P \Rightarrow \neg R) \\ F_3: & (Q \wedge R) \Rightarrow \neg P \\ F_4: & P \vee Q \vee R \\ F_5: & \neg P \vee \neg Q \vee \neg R \\ F_6: & P \Rightarrow \neg(Q \wedge R) \end{array}$$

group logically equivalent formulas together.

P4.4 A graph colouring is an assignment of colours to nodes so that no edge in the graph connects two nodes of the same colour. The graph colouring problem asks whether a graph can be coloured using some fixed number of colours. The question is of great interest, because many scheduling problems are graph colouring problems in disguise. The case of three colours is known to be hard (NP-complete).

How can we encode the three-colouring problem in propositional logic, in CNF to be precise? (One reason we might want to do so is that we can then make use of a SAT solver to determine colourability.) Using propositional variables

- B_i to mean node i is blue,
- G_i to mean node i is green,
- R_i to mean node i is red;
- E_{ij} to mean i and j are different but connected by an edge,

write formulas in CNF for these statements:

(a) Every node (0 to n inclusive) is coloured.
(b) Every node has at most one colour.
(c) No two connected nodes have the same colour.

Assignment Project Exam Help

For a graph with $n + 1$ nodes, what is the size of the CNF formula?

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

P4.5 In the Week 3 lecture we saw that conjunctive normal form (CNF) and disjunctive normal form (DNF) are not *canonical*. For example, $(\neg P \vee \neg Q \vee \neg R) \wedge (P \vee Q \vee R) \wedge (P \vee \neg R) \wedge (\neg Q \vee R)$ and the logically equivalent $P \wedge \neg Q$ are both in reduced CNF. It would be nice if equivalent formulas were syntactically identical. For one thing, equivalence checking would then become much simpler, just checking that the given formulas are the same.

It turns out that we can achieve this if we use exclusive or ($\oplus$) instead of or ($\vee$). XNF (exclusive-or normal form) can express every possible Boolean function. In XNF, a Boolean function is expressed as a sum of products, with addition corresponding to exclusive or, $\oplus$, and multiplication corresponding to conjunction, $\wedge$. For example, instead of $P \wedge (Q \Leftrightarrow R)$ we write $P \oplus (P \wedge Q) \oplus (P \wedge R)$. Actually we usually abbreviate this to $P \oplus PQ \oplus PR$, with the understanding that a "product" PQ is a conjunction. We call an expression like this—one that is written as a sum of products—a *polynomial*. Each summand (like PQ) is a *monomial*.

The connective $\neg$ is not used in XNF, only $\oplus$ and $\wedge$. To compensate, one of the truth value constants, $\mathbf{t}$, is needed. If F is a formula in XNF then $F \oplus \mathbf{t}$ is its negation. The other constant, $\mathbf{f}$, is not needed. This is because $\mathbf{f}$ is like zero: It is a "neutral element" for $\oplus$ (that is, $F \oplus \mathbf{f}$ is just F) and it is an "annihilator" for $\wedge$ ($F \wedge \mathbf{f}$ is just $\mathbf{f}$). So we can't have $\mathbf{f}$ *in* a monomial, because the entire monomial disappears if $\mathbf{f}$ enters it. And we don't need $\mathbf{f}$ *as* a monomial, because "adding" $\mathbf{f}$ really adds nothing.

Assignment Project Exam Help

In this "Boolean ring" algebra, we are really dealing with arithmetic modulo 2, with $\wedge$ playing the role of multiplica these in XNF:

(a) $\neg P$ (b) https://eduassistpro.github.io/ Q (f) $P \vee (Q \wedge R)$

Note that $\wedge$ distributes over $\oplus$, but not the other way rou lling out" properties. If we conjoin the monomials P $PQRST$; that is, "duplicates disappear". However, if we have $F =$ $\mu_2 \oplus \mu_3 \oplus \mu_4$ and we want to find $F \oplus F'$, then we find that "duplicates cancel out pairwise": $F \oplus F' = \mu_1 \oplus \mu_4$. This happens because $F \oplus F = \mathbf{f}$ for all Boolean functions F.

Add WeChat edu_assist_pro

Given this, turn these into XNF:

(g) $\neg(P \oplus Q)$ (h) $(P \oplus Q) \wedge R$ (i) $(PQ \oplus PQR \oplus R) \wedge (P \oplus Q)$

(j) $Q \wedge (P \oplus PQ \oplus \mathbf{t})$ (k) $Q \vee (P \oplus PQ)$

Part 2 - Predicate Logic

P4.6 Translate the following formulas into predicate logic, using $D(x)$ for "x is a duck", $M(x)$ for "x is muddy", $F(x, y)$ for "x follows y" and $R(x, y)$ for "x is muddier y". Use the constant a as a name for Jemima, and b for Louise.

(i) Every duck who follows a muddy duck is muddy

(ii) Any muddy duck is muddier than any duck that isn't muddy

(iii) Jemima is muddier than any duck who is muddier than Louise

(iv) There is a duck who is muddier than Louise, but not as muddy as Jemima

(v) Given any two ducks, one is muddier than the other

(vi) Given any three ducks, if the first is muddier than the second, and the second is muddier than the third, then the first is muddier than the third

P4.7 Translate the following formulas into predicate logic, using $D(x)$ for "x is a duck", $M(x)$ for "x is muddy", $E(x)$ for "x is an egg", $L(x, y)$ for "x lays y", and $R(x, y)$ for "x is muddier y". Use the constant a as a name for Jemima, and b for Louise.

Assignment Project Exam Help

(i) Not every duck lays an egg

(ii) Every duck does

(iii) There is a duck that

(iv) There isn't a duck who lays every egg

(v) Louise is not muddier than every duck

(vi) Louise is not muddier than any duck

ayed it

uddy duck

(i

duck lays

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

P4.8 Consider the following predicates: $C(x)$, which stan x M $x)$, which stands for "x is a mouse", and $L(x, y)$, which stands for "x likes y". Express the statement "No mouse likes a cat who likes mice" as a formula in first-order predicate logic. Here "likes mice" means "likes every mouse".

P4.9 For any formula G and variable x, $\neg\forall x G \equiv \exists x \neg G$, and $\neg\exists x G \equiv \forall x \neg G$. Interpret the formula $\neg\forall x(D(x) \Rightarrow \exists y(E(y) \wedge L(x, y)))$ in natural language, then use these equivalences to "push the negation" through each of the quantifiers and connectives, and re-interpret the result in natural language. Reflect on why these are saying the same thing.

P4.10 In the following formulas, identify which variables are bound to which quantifiers, and which variables are free.

(i) $\forall y(D(x) \wedge \exists x(E(y) \Leftrightarrow L(x, y)))$

(ii) $\exists z(E(z) \wedge M(y)) \Rightarrow \forall y(E(z) \wedge M(y))$

(iii) $\exists x((E(x) \wedge M(y)) \Rightarrow \forall y(E(x) \wedge M(y)))$

(iv) $\forall z(\exists z(D(z)) \Rightarrow D(z))$

(v) $\exists u((D(z) \wedge \forall x(M(x) \Rightarrow D(x))) \Rightarrow \forall z(L(x, z)))$

(vi) $\forall x(\forall y(M(x) \Rightarrow D(x)) \wedge \exists y(D(y) \wedge \forall y(L(y, x))))$