Assessment 1

Logic and Regular Languages

Norbert Logiewa
nl253

November 14, 2017

Propositional Logic

1. Model as a propositional formal the poroperty that the inital state cannot be reached from c or b

Answer:
$model \rightarrow \neg((c \rightarrow a') \vee (b \rightarrow a'))$

2. Prove the following sequent:

$$a \rightarrow b'.b \rightarrow c'.c \rightarrow b' \vdash (a \vee b) \rightarrow (b' \vee c')$$

1. $a \rightarrow b'.b \rightarrow c'.c \rightarrow b'$ [premise]
2. $(a \rightarrow b') \wedge (b \rightarrow c') \wedge (c \rightarrow b') \wedge_i$
3. $(a \rightarrow b') \wedge (c \rightarrow b') \wedge (b \rightarrow c')$ commutativity of $\wedge$
4. $((a \wedge c) \rightarrow b') \wedge (b \rightarrow c')$ distributivity of $\rightarrow$

3. Model the property that the automate can only be in one stat at a time via a defintion of a propositional function

Answer:
$goodState(a, b, c) \equiv (a \rightarrow b') \oplus (b \rightarrow c') \oplus (c \rightarrow b')$

4. Negate the formula and convert it to CNF

$$((a \rightarrow b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a')) \rightarrow (b \rightarrow \neg a')$$

---

1. $\neg(((a \rightarrow b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a')) \rightarrow (b \rightarrow \neg a'))$ negation of the formula
2. $\neg(((\neg a \vee b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a')) \rightarrow (\neg b \vee a'))$
3. $\neg(\neg((\neg a \vee b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a')) \vee (\neg b \vee a'))$
4. $\neg((\neg(a \vee b') \vee \neg(a \wedge \neg b) \vee \neg(b' \wedge \neg a')) \vee (\neg b \vee a'))$
5. $\neg(\neg(a \vee b') \vee \neg(a \wedge \neg b) \vee \neg(b' \wedge \neg a')) \wedge \neg(\neg b \vee a')$
6. $(a \vee b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a') \wedge \neg(\neg b \vee a')$
7. $(a \vee b') \wedge \neg(\neg a \vee b) \wedge \neg(\neg b' \vee a') \wedge \neg(\neg b \vee a')$

8. $(a \vee b') \wedge \neg(\neg a \vee b) \wedge \neg(\neg b' \vee a') \wedge \neg(\neg b \vee a')$ commutativity
9. $(a \vee b') \wedge \neg(\neg a \vee b) \wedge \neg(\neg b' \vee a') \wedge \neg(\neg b \vee a')$
10. $(a \vee b') \wedge a \wedge \neg b \wedge \neg(\neg b' \vee a') \wedge \neg(\neg b \vee a')$
11. $(a \vee b') \wedge a \wedge \neg b \wedge b' \wedge \neg a' \wedge \neg(\neg b \vee a')$
12. $(a \vee b') \wedge a \wedge \neg b \wedge b' \wedge \neg a' \wedge b$

$$(a \vee b') \wedge a \wedge \neg b \wedge b' \wedge \neg a' \wedge b$$

5. Explain the principle of unit propagation from DPLL, and use it to show that CNF formula of the previous question is unsatisfiable.

Answer:
a

Unit propagation refers to a technique that is used to simplify logical formulas. When we have a clause that *only* contains a single literal such as x, in all other clauses that contain x, we can substitute it for $True$.

Answer:
b

We can use it to show that the previous formula is unsatisfiable by demonstrating that it is a contradiction. We simplify it and continue to substitute $True$ for single literal clauses. When we rearrange the formula, it becomes clear that we have $b \wedge \neg b$, this is a contradiction.

$$(True \vee b') \wedge True \wedge \neg b \wedge b' \wedge \neg a' \wedge b$$
$$(True \vee True) \wedge True \wedge \neg b \wedge True \wedge \neg a' \wedge b$$
$$\neg b \wedge True \wedge \neg a' \wedge b$$
$$\neg b \wedge \neg a' \wedge b$$
$$\neg b \wedge b \wedge \neg a'$$

6. Comment on the validity of the original formula.

Answer:
The formula is not satisfiable and therefore not valid as satisfiability is a prerequisite for validity. In other words, because that logical statement was a contradiction (*assumed something can be False and True at the same time*), there was no way in which it could have been true and because of that it is not valid.

First Order Logic

1. Write a formula in first-order logic, using the parent relation, that states that two entities x and y are siblings if they share a parent

Answer:
$\forall i.\forall j.siblings(i_0, j_0) \equiv (parent(x, i_0) = parent(x, j_0))$

2. Assuming there is an equality relation $\equiv$ on P such that $x \equiv y$ means x and y are the same entity, write a formula in first order logic stating that every entity has two distinct parents.

Answer:
$\forall i.\exists! y.\exists! x(parent(x, i) \land parent(y, i) \land \neg(x \equiv y))$

3. Prove that the following sequent is valid:

$$parent(p, q).\forall x.\forall y(parent(x, y) \rightarrow \exists z.parent(z, x)) \vdash \exists.parent(z, p)$$

---

1. $parent(p, q).\forall x.\forall y(parent(x, y) \rightarrow \exists z.parent(z, x))$ [premise]
2. —
3. $\exists.parent(z, p)$ [conclusion] 1 – ?

---

4. Interpret the following formula into simple English statement

Answer:
$\forall x.\exists y.parent(y, x) \rightarrow \forall a.\exists b.parent(a, b)$

Answer:
If all children have a parent, then all parents have a child.

5. The sequent below is not valid. Show this by proving a counter-example: a concrete definition of the universe P (a set) and the relation parent. Explain why this is a counterexample.

$$\vdash \forall x.\exists y.parent(y, x) \rightarrow \forall a.\exists b.parent(a, b)$$

In English
if all children have a parent then all parents have a child.

1. $\forall x.\exists y.parent(y,x) \rightarrow \forall a.\exists b.parent(a,b)$ [premise]

---

2. $P = \{\ x \mid x\ \}$

---

3. $False$ [conclusion]

6. What is the smallest possible counterexample universe P and relation parent. Explain your reasoning.

7. Prove that the following sequent is valid.

$$\forall x.\forall y.parent(x,y) \rightarrow \neg(x \equiv y) \vdash \neg\exists x.parent(x,x)$$

ie. if someone is a parent and has a child, then that child cannot be it's parent.

1. $\forall x.\forall y.parent(x,y) \rightarrow \neg(x \equiv y)$ [premise]

---

2. $parent(x_i,y_j) \rightarrow \neg(x_i \equiv y_j)\ \forall_e\ 1$
3. $\neg parent(x_i,y_j) \vee \neg(x_i \equiv y_j)$ because $P \rightarrow Q \equiv \neg P \vee Q$
4. $\neg(parent(x_i,y_j) \wedge (x_i \equiv y_j))$ De Morgan's Law

---

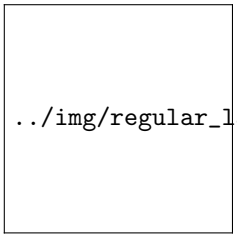
5. $\neg\exists x.parent(x,x)[conclusion]$

Regular Lanugages and Finite Automata

1. For the reular expression $(A|AB|ABB)*$ do the following steps:

a) translate it into an NFA

Answer:



b) remove (semantic preserving) the ϵ transitions

c) create the corresponding DFA

d) translate it into an NFA

Justify the steps.

2. Consider the following language: a word is in the language if and only if it contains an even number of As an odd number of Bs and precisely one C.

a) Design a DFA for this language

b) Explain your design (what information do you "store " in your states?)