Assessment 1

Regular Languages and Logic

Norbert Logiewa
nl253

November 2017

Propositional Logic

1. Model as a propositional formal the property that the initial 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. $(a \rightarrow b') \wedge (b \rightarrow c') \wedge (c \rightarrow b')$ [$\wedge_i$] 1
3. $(a \rightarrow b') \wedge (c \rightarrow b') \wedge (b \rightarrow c')$ [commutativity of $\wedge$] 2
4. $(a \wedge c \rightarrow b') \wedge (b \rightarrow c')$ [distributivity of $\rightarrow$] 3
5. $a \wedge c \rightarrow b'$ [$\wedge_{e2}$] 4
6. $a \rightarrow b'$ [$\wedge_{e2}$] 5
7. $a \vee b \rightarrow b'$ [$\vee_{i2}$] 6

---

8. $(a \vee b) \rightarrow (b' \vee c')$ [$\vee_i$] 7 [conclusion]

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

Answer:
$A \oplus B \equiv (A \vee B) \wedge (\neg A \vee \neg B)$ [definiton of the XOR operator]
$goodState(a, b, c) \equiv (a \rightarrow b') \oplus (b \rightarrow c') \oplus (c \rightarrow b')$

4. Negate the formula $((a \rightarrow b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a')) \rightarrow (b \rightarrow \neg a')$ and convert it to CNF.

(a) Negate the formula $\neg(((a \rightarrow b') \wedge (a \wedge \neg b) \wedge (b' \wedge \neg a')) \rightarrow (b \rightarrow \neg a'))$

(b) Remove $\rightarrow$ using $A \rightarrow B \equiv \neg A \vee B$ property

i. $\neg(((\neg a \vee b') \wedge (\neg a \vee \neg b) \wedge (b' \wedge \neg a')) \rightarrow (\neg b \vee \neg a'))$

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

(c) Push $\neg$ inwards using De Morgan's Laws

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

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

(d) Distribute

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

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

(e) Simplify (remove duplicate elements because $A \wedge A \equiv A$)

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

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

Parenthesis are dropped because: $A \wedge B \wedge C \equiv A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$ so they all need to be $True$ anyway

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

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

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

1. **Answer**:
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 replace it with $True$ ie. $\top$.

2. **Answer**:
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 $True$ ie $\top$ for single literal clauses.

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

   (b) $\neg a' \wedge a' \wedge b \wedge \top \wedge (\neg a \vee \neg b) \wedge (\neg a \vee \top)$ replace all b's with $\top$

   (c) $\neg a' \wedge a' \wedge \top \wedge \top \wedge (\neg a \vee \bot) \wedge (\neg a \vee \top)$ replace all bs with $\top$

   (d) $\bot \wedge \top \wedge \top \wedge \top \wedge (\neg a \vee \bot) \wedge (\neg a \vee \top)$ replace all a's with $\top$

   When we rearrange and simplify the formula, it becomes clear that we have $\top\bot$, this is a contradiction, something cannot be $True$ and $False$ at the same time ($\bot \wedge \top \equiv \bot$).

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* $\bot$ *and True* $\top$ *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:
$\forall i.\forall j.siblings(i,j) \equiv (parent(x,i) = parent(x,j))$

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:
$\exists!x.P(x) \equiv \exists x(P(x) \wedge \forall y(P(y) \rightarrow y = x))$ [1]
$\forall i.\exists!y.\exists!x(parent(x,i) \wedge parent(y,i) \wedge \neg(x \equiv y))$

3. Prove that the following sequent is valid:

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

---

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

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

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.

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

Answer:
If this is not valid then: $\forall x.\exists y.parent(y,x) \wedge \neg(\forall a.\exists b.parent(a,b))$, because the only way implication can be wrong is if we are concluding $False$ from $True$ ie $\neg(A \rightarrow B) \equiv A \wedge \neg B$. So I need to prove this is true by defining the Universe P (a set):

$$P = \{\ x \mid \exists y.parent(y,x)\}$$

and the case that contradicts it:

$$\exists a.\exists b.\neg parent(a,b)$$

The reason this would be a counterexample is that it shows that there is an object for which the predicate *parent* does not hold, which contradicts this logical statement which is universally quantified ie. makes a claim about all objects.

[1] There exists exactly one, see https://math.stackexchange.com/questions/228285/how-can-i-get-the-negation-of-exists-unique-existential-quantification

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

Answer:
The smallest possible counterexample universe (a set) would consist of the minimum number of elements for which the predicate does not hold (here this is two). Normally one counterexample is sufficient to contradict any statement that is universally quantified. In other words $\exists(A).\neg P(A)$ contradicts $\forall A.P(A)$, but because this predicate is a statement about two objects A and B we do need both to show a single case where P does not hold.

7. Prove that the following sequent is valid.

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

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

---

2. $parent(x_0,y_0) \rightarrow \neg(x_0 \equiv y_0)$ $\forall_e$ 1
3. $\neg parent(x_0,y_0) \vee \neg(x_0 \equiv y_0)$ because $P \rightarrow Q \equiv \neg P \vee Q$
4. $\neg(parent(x_0,y_0) \wedge (x_0 \equiv y_0))$ De Morgan's Law
5. $\neg parent(x_0,x_0)$ because if $(A \equiv B)$ then we can replace all B with A or A with B
6. $\neg\exists x_0.parent(x_0,x_0))$ $\exists_i$

---

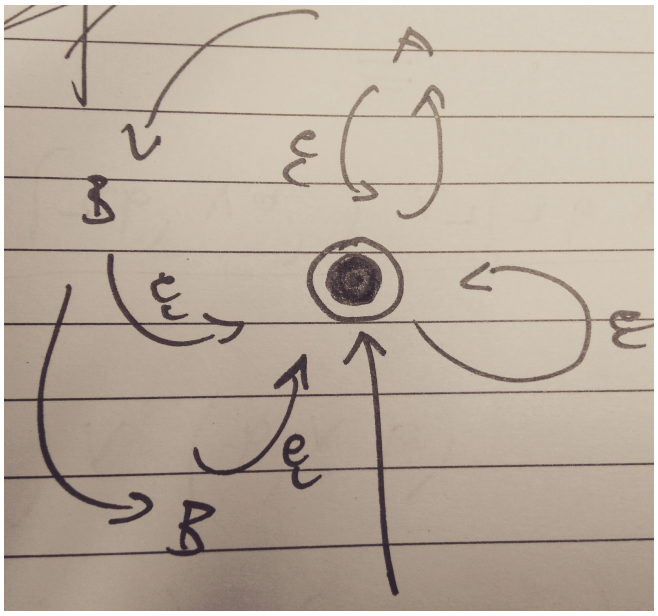
7. $\neg\exists x_0.parent(x_0,x_0))$ $\exists_i$

Regular Lanugages and Finite Automata

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

- translate it into an NFA

Answer:

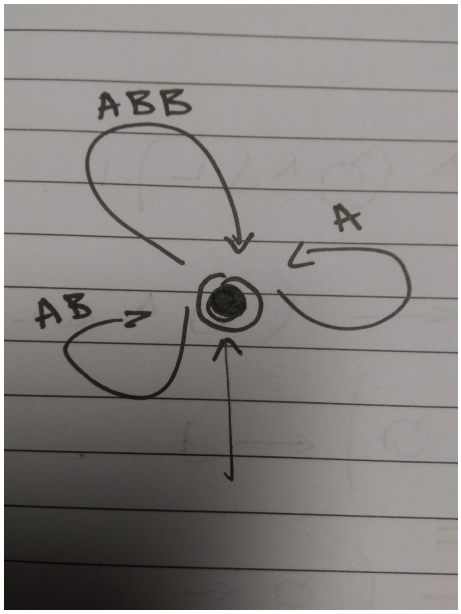


- remove (semantic preserving) the ϵ transitions

Answer:
–

- create the corresponding DFA

Answer:



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.

Answer:
This is not possible because Finite Automata *cannot store any states* – they don't have memory and cannot keep track of how many As and Bs have been encountered. They only memory they have is the state they are in. Generally speaking FSM lack the expressivenes to solve this problem and parsing needs to be used in such cases.

References

[noitemsep]https://en.wikipedia.org/wiki/Conjunctive_normal_form https://en.wikipedia.org/wiki/De_Morgan%27s_laws https://en.wikipedia.org/wiki/Distributive_property https://en.wikipedia.org/wiki/Exclusive_or https://en.wikipedia.org/wiki/Finite-state_machine https://en.wikipedia.org/wiki/Modus_ponens https://en.wikipedia.org/wiki/Uniqueness_quantificatio https://math.stackexchange.com/questions/228285/how-can-i-get-the-negation-of-exists-unique-exis https://stackoverflow.com/questions/133601/can-regular-expressions-be-used-to-match-nested-patte https://en.wikipedia.org/wiki/Negation