Logik	WS 2009/2010	LVA 703019

EXAM 2 April 9, 2010

This exam consists of <u>five</u> exercises. The available points for each item are written in the margin. You need at least 50 points to pass.

1 Consider the boolean function $f(x,y,z) = 1 \oplus y \oplus xy \oplus xz$.

[10] (a) Give a binary decision tree for f with the variable ordering $[x,y,z]$ and use the reduce algorithm to construct an equivalent reduced OBDD.

[3] (b) Show that $\neg$ can be expressed in terms of f.

[7] (c) Prove that $\{f\}$ is adequate.

2 In this exercise we consider polynomials over the variable x which are constructed according to the following grammar:

$$P ::= x \mid N \mid P + P \mid P * P$$
$$N ::= \text{(some number)}$$

All of the following parts can be done independently!

[5] (a) Write a Prolog predicate `derive/2` to compute derivatives of polynomials. Your program does *not* have to simplify the resulting polynomial. For example, for the query `derive(3*x,D)` a possible answer might be `D = 0*x+1*3`.

Hints: `number/1` checks whether the input is a number; $(p * q)' = p' * q + q' * p$.

[10] (b) Write a Prolog predicate `parse/2` to parse polynomials. Here, the input is a list over the alphabet $\{x, +, *, (,)\} \cup \mathbb{Z}$. The grammar is like the one above except that there is an additional production for parentheses:

$$P ::= (P)$$

The parser should not make any assumptions about precedence of operators. Hence, the query `parse([15,*,'(',x,+,3,*,x,')'],P)` has two solutions: `P = 15*(x+(3*x))` and `P = 15*((x+3)*x)`.

[5] (c) Using `parse/2`, write a Prolog predicate `parse_unique/2` which takes the same input list as for `parse/2` and returns one of three possible results: `invalid`, if the input cannot be parsed; `multiple`, if there is more than one possibility to parse the input; and the unique polynomial, otherwise.

For example, `parse_unique([15,*,'(',x,+,3,*,x,')'],P)` yields `P = multiple` and `parse_unique([15,*,'(',x,+,x,')'],P)` yields `P = 15*(x+x)`.

Hint: use `findall/3`.

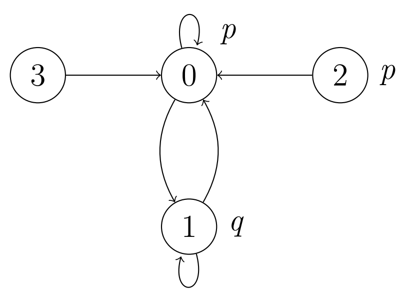
$\boxed{3}$ For each of the following sequents of predicate logic, either give a natural deduction proof or find a model which does not satisfy it:

[6] (a) $\forall x\, P(x, f(x)), \forall x\, \forall y\, \forall z\, (P(x,y) \land P(y,z) \to P(x,z)) \vdash \exists x\, P(x,x)$

[7] (b) $\forall x\, (\neg Q(x) \to Q(g(x))), Q(a) \vdash \exists x\, (Q(x) \land Q(g(g(x))))$

[7] (c) $\forall x\, R(x, h(x)), \forall x\, \forall y\, \neg(R(x,y) \land R(y,x)) \vdash \neg\exists x\, (x = h(x))$

$\boxed{4}$ Consider the CTL formula $\phi = \mathsf{E}[p\, \mathsf{U}\, \mathsf{EX}\, q]$ and the model $\mathcal{M}$:



In this exercise we use the symbolic model checking algorithm to determine in which states ϕ holds. We adopt the following binary encoding of states:

state	x	y
0	0	0
1	0	1
2	1	0
3	1	1

For simplicity, you *do not have to* construct the BDDs, instead all operations should be performed on boolean formulas, as in the lecture.

[5] (a) Encode the transition relation $\to$ as a boolean formula $\phi_{\to}$.
Hint: Simplifying the resulting formula may be helpful for parts (b) and (c).

[6] (b) Encode the set of states in which the CTL formulas p, q and $\mathsf{EX}\, q$ hold as boolean formulas. For $\mathsf{EX}\, q$, employ the formula $\phi_{\to}$ from part (a).

[9] (c) Complete the following algorithm for constructing the formula representing the set of states where ϕ holds.

```
W := ⟦p⟧;
X := ∅;
Y := ⟦EX q⟧;
repeat until X = Y
     X := Y;
     Y := Y □ (W □ pre_□(Y));
```

Use this algorithm to determine in which states ϕ holds. Give the formulas representing the intermediate assignments to Y after each iteration.

[20] $\boxed{5}$ Determine whether the following statements are true or false. Every correct answer is worth 2 points. For every wrong answer 1 point is subtracted, provided the total number of points is non-negative.

statement
$[\![\, \mathsf{E}[\phi \,\mathsf{U}\, \psi] \,]\!]$ is the least fixed point of the function $F\colon \mathcal{P}(S) \to \mathcal{P}(S)$ that maps X to $[\![\, \psi \,]\!] \cap ([\![\, \phi \,]\!] \cup \mathsf{pre}_{\exists}(X))$.
The instance $\{(01, 0), (011, 1), (10, 1)\}$ of Post correspondence problem has a solution.
Every adequate set of temporal CTL connectives contains EF.
The term $f(y, z)$ is free for y in $\forall x\, ((\forall z\, (P(z) \wedge Q(y))) \to \neg P(x) \vee Q(z))$.
$\exists x\, \phi \wedge \exists x\, \psi \dashv\vdash \exists x\, (\phi \wedge \psi)$
The algebraic normal form of the boolean function $f(x, y) = x \cdot \overline{y}$ is $x \oplus xy$.
The proof rules LEM, MT and $\neg\neg$e are inter-derivable with respect to the other basic proof rules of natural deduction.
The terms $p(X, X, Y)$ and $p(g(Y), g(Z), a)$ are unifiable.
A unary boolean function f is self-dual if and only if $f(0) \neq f(1)$.
Executing the Prolog query `?- [A,B] = [A|C].` produces the answer `C = [B]`.