Informatics and Mathematical Modelling
DTU

02280 Data-Logic Spring term 2003

Solutions to most exercises in *Data Logic* chapters 14 - 16

Exercise 14.2 Logic program parser with derive

$$
\begin{array}{l}
derive(S_1, S_2) \leftarrow \\
\quad append(L, Bl, S_1), append(Al, [A], L), \\
\quad rule(A, Ga), \\
\quad append(Al, Ga, L'), append(L', Bl, S_2).
\end{array}
$$

$$
\begin{array}{l}
deriveclosure(S_1, S_1). \\
deriveclosure(S_1, S_3) \leftarrow derive(S_1, S_2), deriveclosure(S_2, S_3).
\end{array}
$$

Exercise 14.3 Logic program for parsing BNF

In the below predicate the first argument is a sentential form and the two last arguments form the string to be analysed as a difference list.

The first clause covers the case of a terminal symbol in front; the third clause covers the case of a nonterminal in front of the sentential form.

$$
\begin{array}{l}
parseBNF([T|Sl], [T|Tl1], Tl2) \leftarrow parseBNF(Sl, Tl1, Tl2). \\
parseBNF([], Tl, Tl). \\
parseBNF([S|Sl], Tl1, Tl3) \leftarrow \\
\quad rule(S, Rhs), \\
\quad parseBNF(Rhs, Tl1, Tl2), parseBNF(Sl, Tl2, Tl3).
\end{array}
$$

Exercise 14.5 BNF grammar for propositional logic

$\langle prop \rangle ::= \langle prop0 \rangle \ \langle prop0list \rangle$

$\langle prop0 \rangle ::= \langle prop1 \rangle \ \rightarrow \ \langle prop0 \rangle \mid \langle prop1 \rangle$

$\langle prop0list \rangle ::= \epsilon \mid \ \leftarrow \ \langle prop0 \rangle \ \langle prop0list \rangle$

$\langle prop1 \rangle ::= \langle prop2 \rangle \ \vee \ \langle prop1 \rangle \mid \langle prop2 \rangle$

$\langle prop2 \rangle ::= \langle prop3 \rangle \ \wedge \ \langle prop2 \rangle \mid \langle prop3 \rangle$

$\langle prop3 \rangle ::= \neg \langle prop3 \rangle \mid (\ \langle prop \rangle \ ) \mid \langle ident \rangle$

Exercise 15.1 Reversal of list using difference list

$$
\begin{array}{l}
reverse(L, Lr) \leftarrow rev(L, Lr, []) \\
rev([], L1, L1). \\
rev([X|T], L1, L2) \leftarrow rev(T, L1, [X|L2]).
\end{array}
$$

Exercise 15.3 Clauses for grammar (see 14.5)

$prop(S_1, S_3, T) \leftarrow prop0(S_1, S_2, T1), prop0list(S_2, S_3, T1, T).$

$prop0list(S, S, T, T).$
$prop0list([\text{“}\leftarrow'' \,|S_1], S_2, T1, T) \leftarrow prop0(S_1, S_2, T2), prop0list(S_2, S_3, if(T1, T2), T).$

$prop0(S_1, S_3, imp(T1, T2)) \leftarrow prop1(S_1, [\text{“}\rightarrow'' \,|S_2], T1), prop0(S_2, S_3, T2).$
$prop0(S_1, S_2, T) \leftarrow prop1(S_1, S_2, T).$

$prop1(S_1, S_3, or(T1, T2)) \leftarrow prop2(S_1, [\text{“}\vee'' \,|S_2], T1), prop1(S_2, S_3, T2).$
$prop1(S_1, S_2, T) \leftarrow prop2(S_1, S_2, T).$

$prop2(S_1, S_3, and(T1, T2)) \leftarrow prop3(S_1, [\text{“}\wedge'' \,|S_2], T1), prop2(S_2, S_3, T2).$
$prop2(S_1, S_2, T) \leftarrow prop3(S_1, S_2, T).$

$prop3([\text{“}\neg''|S_1], S_2, non(T)) \leftarrow prop3(S_1, S_2, T).$
$prop3([\text{“}(''|S_1], S_2, T) \leftarrow prop3(S_1, [\text{“})''|S_2], T).$
$prop3(S_1, S_2, T) \leftarrow ident(S_1, S_2, T).$

Exercise 16.1 Clauses for truth values of propositional sentences

$istrue(and(P, Q), I) \leftarrow istrue(P, I) \wedge istrue(Q, I).$

$istrue(or(P, Q), I) \leftarrow istrue(P, I).$
$istrue(or(P, Q), I) \leftarrow istrue(Q, I).$

$istrue(non(P), I) \leftarrow isfalse(P, I).$

$istrue(imp(P, Q), I) \leftarrow istrue(or(non(P), Q), I).$

$istrue(atm(P), I) \leftarrow member(e(P, t), I).$

and for the complementary predicate:

$isfalse(and(P, Q), I) \leftarrow isfalse(P, I).$
$isfalse(and(P, Q), I) \leftarrow isfalse(Q, I).$

$isfalse(or(P, Q), I) \leftarrow isfalse(P, I) \wedge isfalse(Q, I).$

$isfalse(non(P), I) \leftarrow istrue(P, I).$

$isfalse(imp(P, Q), I) \leftarrow isfalse(or(non(P), Q), I).$

$isfalse(atm(P), I) \leftarrow member(e(P, f), I).$

Exercise 16.3 Clauses for satisfiability

$satisfiable(P, A) \leftarrow inter(I, A) \wedge istrue(P, I).$

$inter([], []).$
$inter([(S, f)|Il], [S|Slist]) \leftarrow inter(Il, Slist).$
$inter([(S, t)|Il], [S|Slist]) \leftarrow inter(Il, Slist).$

Exercise 16.4 Clauses for validity

In predicate logic:

$$\forall\ valid(P, A) \leftarrow \forall I(inter(I, A) \rightarrow istrue(P, I)).$$

This sentence cannot be transformed into definite clause form in a simple way. Formalisation of validity can however be done by introducing a predicate for calculating a list of all the possible interpretations.

Formalisation of logical consequence can be expressed by way of validity of an implication sentence.

Exercise 16.6 Reformulation of clauses for *solve*

$$solve(P) \leftarrow solve(if(P, B)) \ \wedge \ solvelist(B)$$

$$solvelist([P|Pl]) \leftarrow solve(P) \ \wedge \ solvelist(Pl)$$

$$solvelist([])$$