

Unit 4 Seminar exercises

1. Prolog can be used to test the questions included in Unit 2. For example, to test exercise 1 carry out the following steps.

- Surf to <https://swi-prolog.org>
- Click on “try swi-prolog online”.
- On the SWISH page click on notebook.
- Click on Query.
- In the ‘query’ box enter “member(c, [a,b,c,2,3,4])”.
- Click the go (>) button – it should give the answer ‘true’ (I.e., c is a member of the set).
- How many of the questions in exercise 1 can you check in this way?

Answer:

The screenshot shows the SWISH web interface. On the left, there's a sidebar with navigation links like 'examples', 'movies', 'prolog_tutorials', and 'dict'. The main area displays a 'Welcome to SWISH' message and a list of example programs. On the right, a query box contains the Prolog query `member(c, [a,b,c,2,3,4])`. Below the query box, the execution results are shown, including a list of messages and a final output of `true`. The interface also includes a search bar and a 'table results' button.

Negation: $\sim P$

`not_p(P) :- \+ P.`

Conjunction: $P \wedge Q$

`and_p_q(P, Q) :- P, Q.`

Disjunction: $P \vee Q$

`or_p_q(P, Q) :- P; Q.`

Implication: $P \rightarrow Q$

`implies_p_q(P, Q) :- \+ P; Q.`

Biconditional: $P \leftrightarrow Q$

`biconditional_p_q(P, Q) :- implies_p_q(P, Q), implies_p_q(Q, P).`

Complex: $P \rightarrow (\sim Q)$

`implies_p_not_q(P, Q) :- \+ P; \+ Q.`

Negation of Implication: $(\sim Q) \rightarrow (\sim P)$

`neg_implies_q_p(Q, P) :- Q, \+ P.`

Exclusive OR: $P \text{ XOR } Q$

`xor_p_q(P, Q) :- (P, \+ Q); (\+ P, Q).`

Negation of Conjunction: $\sim(P \wedge Q)$

`neg_and_p_q(P, Q) :- \+ (P, Q).`

Disjunction with Conjunction: $P \vee (Q \wedge R)$

`or_p_and_q_r(P, Q, R) :- P; (Q, R).`

Nested Disjunction: $P \vee (Q \vee R)$

`or_p_or_q_r(P, Q, R) :- P; Q; R.`

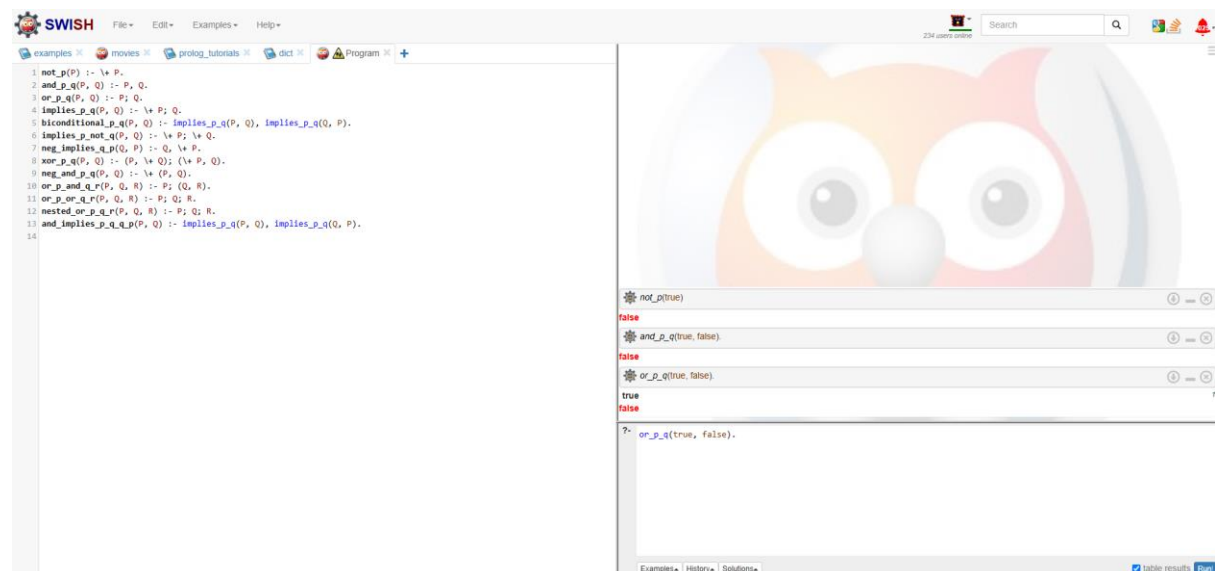
Comparison of Disjunctions: $(P \vee Q) \vee R$

`nested_or_p_q_r(P, Q, R) :- P; Q; R.`

Biconditional: $(P \rightarrow Q) \wedge (Q \rightarrow P)$

`and_implies_p_q_q_p(P, Q) :- implies_p_q(P, Q), implies_p_q(Q, P).`

Testing:



The screenshot shows the SWISH Prolog IDE interface. On the left, a list of 14 logic rules is displayed, numbered 1 through 14. The rules define various logical operations like not, and, or, implies, xor, biconditional, neg_implies, neg_and, or_and, or_or, nested_or, and and_implies. On the right, the test results for these rules are shown. The first four rules (1-4) are marked as 'false', while the fifth rule (5) is marked as 'true'. The sixth rule (6) is also marked as 'false'. The seventh rule (7) is marked as 'true'. The eighth rule (8) is marked as 'false'. The ninth rule (9) is marked as 'true'. The tenth rule (10) is marked as 'false'. The eleventh rule (11) is marked as 'true'. The twelfth rule (12) is marked as 'false'. The thirteenth rule (13) is marked as 'true'. The fourteenth rule (14) is marked as 'false'. The results are displayed in a table format with columns for the rule number, the rule name, and the result.

Rule	Rule Name	Result
1	<code>not_p(P) :- \+ P.</code>	false
2	<code>and_p_q(P, Q) :- P, Q.</code>	false
3	<code>or_p_q(P, Q) :- P; Q.</code>	false
4	<code>implies_p_q(P, Q) :- \+ P; Q.</code>	false
5	<code>biconditional_p_q(P, Q) :- implies_p_q(P, Q), implies_p_q(Q, P).</code>	true
6	<code>implies_p_not_q(P, Q) :- \+ P; \+ Q.</code>	false
7	<code>neg_implies_q_p(Q, P) :- Q, \+ P.</code>	true
8	<code>xor_p_q(P, Q) :- (P, \+ Q); (\+ P, Q).</code>	false
9	<code>neg_and_p_q(P, Q) :- \+ (P, Q).</code>	true
10	<code>or_p_and_q_r(P, Q, R) :- P; (Q, R).</code>	false
11	<code>or_p_or_q_r(P, Q, R) :- P; Q; R.</code>	true
12	<code>nested_or_p_q_r(P, Q, R) :- P; Q; R.</code>	false
13	<code>and_implies_p_q_q_p(P, Q) :- implies_p_q(P, Q), implies_p_q(Q, P).</code>	true
14		false

2. Read Ritchie (2002) section 8.2 (starting on pg 12). Input the facts into the SWI-SWISH page and run the queries. To do this:

- Click the + sign next to the word notebook.
- Choose program.
- Enter the facts (printed in the Ritchie book) into the large window.
- On the right hand side of the screen you will see a smaller window, with a “?- “ at the top corner – this is the query box.
- Enter your queries into the box then click the run button.
- Try all the queries presented in sections 8.2 and 8.3 of the Ritchie book.

Answer:

The screenshot shows the SWISH Prolog environment. On the left, a program is loaded with the following facts:

```

1 man(paul).
2 man(david).
3 man(peter).
4 woman(louise).
5 woman(helen).
6 woman(mandy).
7 wifeof(paul, louise).
8 wifeof(peter, helen).
9 sonof(paul, peter).
10 daughterof(peter, mandy).

```

On the right, several queries are being executed:

- `man(peter)` returns `true`.
- `man(louise)` returns `false`.
- `woman(Someone)` returns a table with 3 rows: `louise`, `helen`, and `mandy`.
- `wifeof(paul, Hiswife)` returns a table with 1 row: `Hiswife` is `louise`.
- `wifeof(Herhusband, louise)` returns a table with 1 row: `Herhusband` is `paul`.
- `daughterof(Father, mandy)` returns a table with 1 row: `Father` is `peter`.
- `sonof(david, Son)` returns `false`.
- `cousin(david, Son)` returns `false`.
- `procedure "cousin(A,B)" does not exist` is shown.
- A query `?- cousin(david, Son).` is entered in the query box.

The screenshot shows the SWISH Prolog environment. On the left, a program is loaded with the following facts:

```

1 man(paul).
2 man(david).
3 man(peter).
4 woman(louise).
5 woman(helen).
6 woman(mandy).
7 wifeof(paul, louise).
8 wifeof(peter, helen).
9 sonof(paul, peter).
10 daughterof(peter, mandy).
11 parentof(Person1, Person2):-daughterof(Person1, Person2).
12 parentof(Person1, Person2):-sonof(Person1, Person2).

```

On the right, a query `?- parentof(Parent, Child).` is entered in the query box. The results show a table with 2 rows: `Parent` is `paul` and `Child` is `peter`; `Parent` is `louise` and `Child` is `peter`.

3. Enter the Prolog version of the “crossing problem” into the SWISH program window and run it. What is the result?

Unable to answer as the Prolog answer is not provided in the learning materials