

Week 12 Tutorial Solutions

Miscellaneous topics

Question 1

(1) Consider the following Prolog program:

```
alts([A|Xs],[A,B|Ys]) :- alts(Xs,Ys).      % (A1)
alts([B|Xs],[A,B|Ys]) :- alts(Xs,Ys).     % (A2)
alts([],[]).                               % (A3)
alts([A],[A]).                             % (A4)
```

(a) For the query `alts(Ls,[1,2,3,4,5]).`, what values of `Ls` result, and in what order?

Solution `[1,3,5] [1,4,5] [2,3,5] [2,4,5]`

(b) One of its solutions is `Ls = [1,4,5]`.

Explain in detail the sequence of steps which leads to the the solution `Ls = [1,4,5]`.

For each of these steps include the following information:

- The subgoal being attempted (such as, eg, “solve `alts(Ls,[1,3,4])`”)
- Which rule (of (A1) to (A4)) is matched to that subgoal, in the course of producing the solution
- What solution is immediately and ultimately obtained for the variable (eg, `Ls`) in that subgoal

Solution

Goal	Rule	Immediate value	Eventual value
<code>alts(Ls,[1,2,3,4,5]).</code>	(A1)	<code>Ls = [1 Ms]</code>	<code>Ls = [1,4,5]</code>
<code>alts(Ms,[3,4,5]).</code>	(A2)	<code>Ms = [4 Ns]</code>	<code>Ms = [4,5]</code>
<code>alts(Ns,[5]).</code>	(A4)	<code>Ns = [5]</code>	<code>Ns = [5]</code>

(c) How would the answer to (a) change if we changed the order of the clauses?

Solution

If you change the order of the first two clauses, the solutions (in order) become `[2,4,5] [2,3,5] [1,4,5] [1,3,5]`

Other changes don't matter, since when the second argument is a concrete list (as here) the clauses require the number of elements to be ≥ 2 , ≥ 2 , $= 1$, $= 0$ respectively.

- (d) What happens if we try the goal `alts(As, Bs).`? How does this change if we reorder the clauses?

Solution Prolog can instantiate the goal variables `As` and `Bs` to `As = [A|Xs]` and `Bs = [A,B|Ys]` (with its own choice of variable names), and then try to solve `alts(Xs, Ys)`. This goes on forever, with no result.

But if you put the clause `alts([A],[A]).` first, then you get

```
| ?- alts(X,Y).
```

```
X = [A]
```

```
Y = [A] ? ;
```

```
X = [A,B]
```

```
Y = [A,_,B] ? ;
```

```
X = [A,B,C]
```

```
Y = [A,_,B,_,C] ? ;
```

```
X = [A,B,C,D]
```

```
Y = [A,_,B,_,C,_,D] ? ;
```

Further variations will depend on which of the two rules appears before the other, and whether one or both facts appear before the first of the rules, and in which order.

- (2) In the course of executing a Prolog program, the Prolog engine needs to unify the following two terms:

```
arr(A, p(arr(B, D), G))
arr(arr(int, G), p(A, bool))
```

Unify these two terms, so as to obtain the most general unifier. Show clearly the steps in the algorithm you use.

Solution

For each step which obtains a new substitution, you apply that substitution to the rhs of previous substitutions and to the remaining pairs of terms to be unified. At each step we either decompose a pair of terms in the list of pairs to be unified or we choose a pair of which one term is a variable, and make that the next new substitution. The next new substitution is underlined.

Pairs to be unified	Substitutions
<code>arr(A, p(arr(B, D), G)) = arr(arr(int, G), p(A, bool))</code>	
<u><code>A = arr(int, G)</code></u> ; <code>p(arr(B, D), G) = p(A, bool)</code>	
<code>p(arr(B, D), G) = p(arr(int, G), bool)</code>	<u><code>A = arr(int, G)</code></u>
<code>arr(B, D) = arr(int, G)</code> ; <u><code>G = bool</code></u>	<code>A = arr(int, G)</code>
<code>arr(B, D) = arr(int, bool)</code>	<code>A = arr(int, bool)</code> ; <code>G = bool</code>
<u><code>B = int</code></u> ; <code>D = bool</code>	<code>A = arr(int, bool)</code> ; <code>G = bool</code>
<u><code>D = bool</code></u>	<code>A = arr(int, bool)</code> ; <code>G = bool</code> ; <code>B = int</code>
	<code>A = arr(int, bool)</code> ; <code>G = bool</code> ; <code>B = int</code> ; <code>D = bool</code>

Question 2

We want to infer types for each of the following λ -calculus terms. For each of them

- draw the parse tree, indicating, with type variables, types for each subterm
- generate the constraints relating these types
- solve all the constraints to find the most general type of the whole term

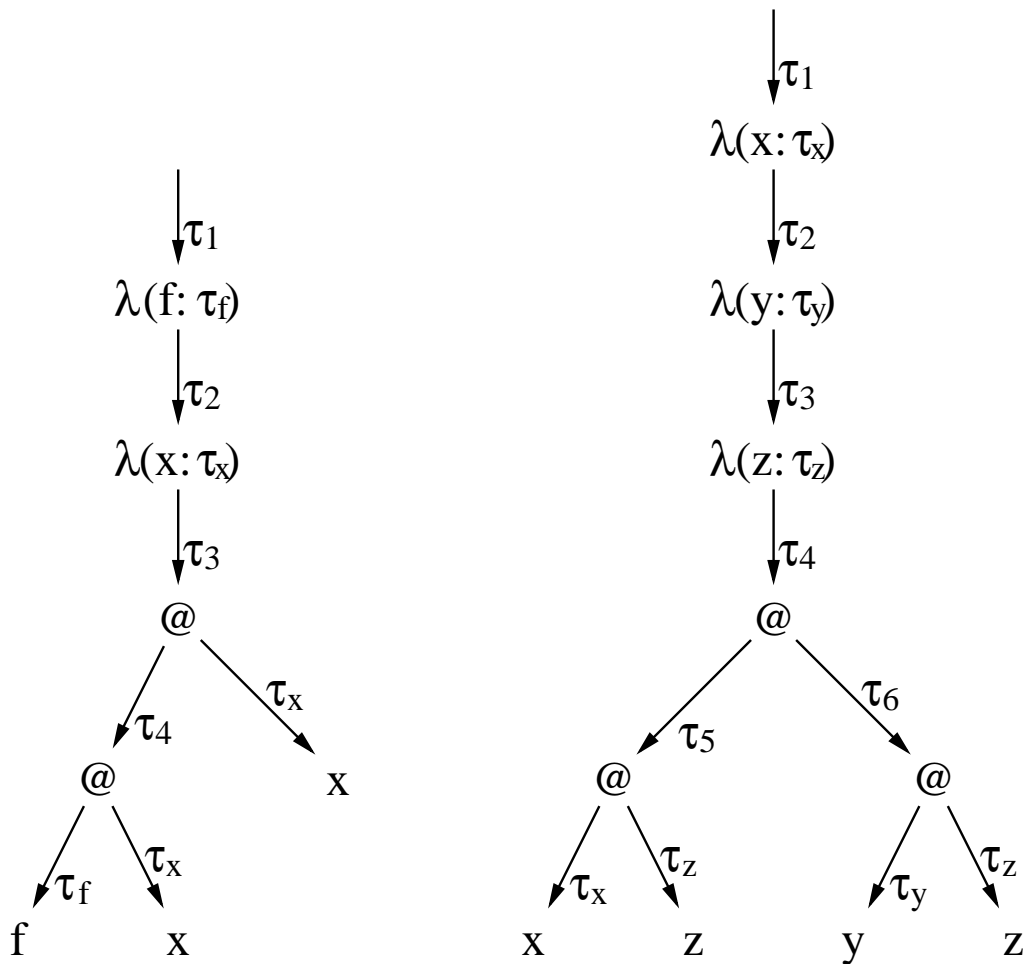
(1) $\lambda f. \lambda x. f x x$

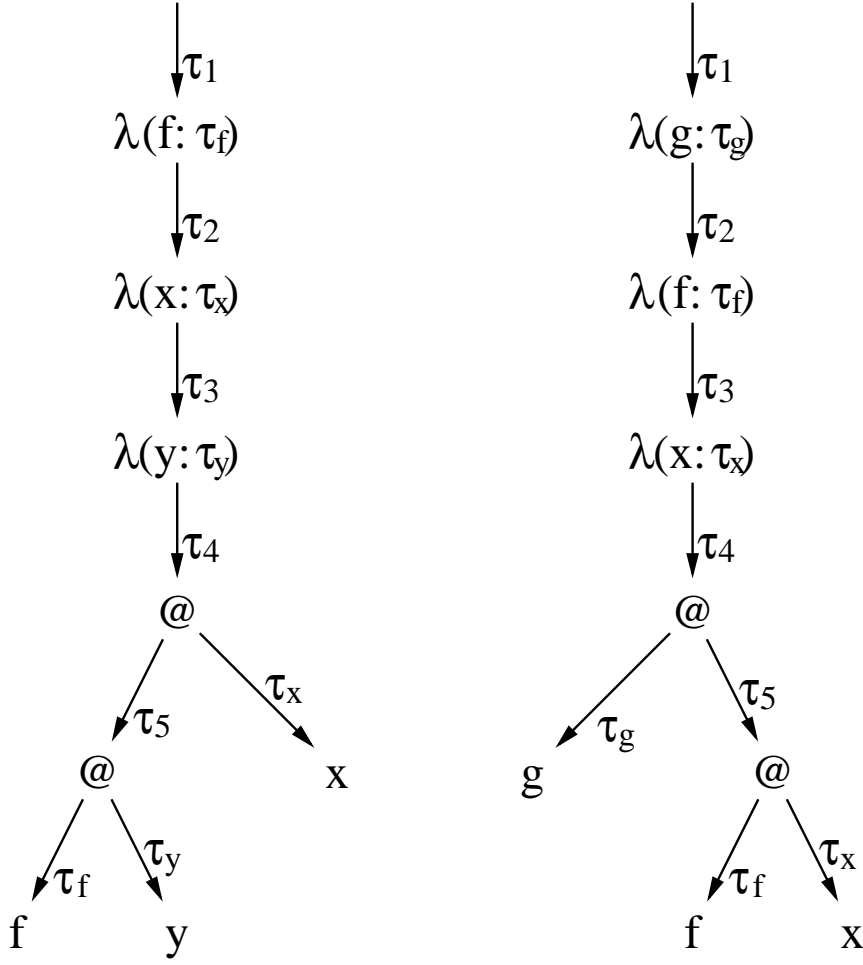
(2) $\lambda x. \lambda y. \lambda z. x z (y z)$

(3) $\lambda f. \lambda x. \lambda y. f y x$

(4) $\lambda g. \lambda f. \lambda x. g (f x)$

Solution





Constraints:

	$\tau_1 = \tau_x \rightarrow \tau_2$		$\tau_1 = \tau_g \rightarrow \tau_2$
$\tau_1 = \tau_f \rightarrow \tau_2$	$\tau_2 = \tau_y \rightarrow \tau_3$	$\tau_1 = \tau_f \rightarrow \tau_2$	$\tau_1 = \tau_g \rightarrow \tau_2$
$\tau_2 = \tau_x \rightarrow \tau_3$	$\tau_3 = \tau_z \rightarrow \tau_4$	$\tau_2 = \tau_x \rightarrow \tau_3$	$\tau_2 = \tau_f \rightarrow \tau_3$
$\tau_4 = \tau_x \rightarrow \tau_3$	$\tau_5 = \tau_6 \rightarrow \tau_4$	$\tau_3 = \tau_y \rightarrow \tau_4$	$\tau_3 = \tau_x \rightarrow \tau_4$
$\tau_f = \tau_x \rightarrow \tau_4$	$\tau_x = \tau_z \rightarrow \tau_5$	$\tau_5 = \tau_x \rightarrow \tau_4$	$\tau_g = \tau_5 \rightarrow \tau_4$
	$\tau_y = \tau_z \rightarrow \tau_6$	$\tau_f = \tau_y \rightarrow \tau_5$	$\tau_f = \tau_x \rightarrow \tau_5$

Solutions to the constraints:

- (1) $(\tau_x \rightarrow \tau_x \rightarrow \tau_3) \rightarrow \tau_x \rightarrow \tau_3$
- (2) $(\tau_z \rightarrow \tau_6 \rightarrow \tau_4) \rightarrow (\tau_z \rightarrow \tau_6) \rightarrow \tau_z \rightarrow \tau_4$
- (3) $(\tau_y \rightarrow \tau_x \rightarrow \tau_4) \rightarrow \tau_x \rightarrow \tau_y \rightarrow \tau_4$
- (4) $(\tau_5 \rightarrow \tau_4) \rightarrow (\tau_x \rightarrow \tau_5) \rightarrow \tau_x \rightarrow \tau_4$

Question 3

In lectures (the lecture on Soundness and Completeness, slides 12 and 13) we have tables giving the relevant rules which correspond to the calculations done in constructing a row of the truth-table, for the \wedge and \rightarrow boolean operators. These tables are also shown below.

p	q	$p \rightarrow q$	corresponding “rule”	p	q	$p \wedge q$	corresponding “rule”
T	T	T	$\frac{p \quad q}{p \rightarrow q}$	T	T	T	$\frac{p \quad q}{p \wedge q}$
T	F	F	$\frac{p \quad \neg q}{\neg(p \rightarrow q)}$	T	F	F	$\frac{p \quad \neg q}{\neg(p \wedge q)}$
F	T	T	$\frac{\neg p \quad q}{p \rightarrow q}$	F	T	F	$\frac{\neg p \quad q}{\neg(p \wedge q)}$
F	F	T	$\frac{\neg p \quad \neg q}{p \rightarrow q}$	F	F	F	$\frac{\neg p \quad \neg q}{\neg(p \wedge q)}$

Construct similar tables for the \vee and \neg operators. Make sure you know how to prove the corresponding rules.

Solution

p	q	$p \vee q$	corresponding “rule”	p	$\neg p$	corresponding “rule”
T	T	T	$\frac{p \quad q}{p \vee q}$	T	F	$\frac{p}{\neg \neg p}$
T	F	T	$\frac{p \quad \neg q}{p \vee q}$	F	T	$\frac{\neg p}{\neg p}$
F	T	T	$\frac{\neg p \quad q}{p \vee q}$			
F	F	F	$\frac{\neg p \quad \neg q}{\neg(p \vee q)}$			

The proof for the fourth \vee rules was given in lectures, but here it is again. The Lemma is that from a contradiction, you can derive anything

1	$\neg p$					
2	$\neg q$					
3	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">$p \vee q$</td> <td></td> </tr> </table>	$p \vee q$				
$p \vee q$						
4	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;"> <table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">p</td> <td></td> </tr> </table> </td> <td></td> </tr> </table>	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">p</td> <td></td> </tr> </table>	p			
<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">p</td> <td></td> </tr> </table>	p					
p						
5	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">$p \wedge \neg p$</td> <td style="padding-left: 10px;">\wedge-I, 4, 1</td> </tr> </table>	$p \wedge \neg p$	\wedge -I, 4, 1			
$p \wedge \neg p$	\wedge -I, 4, 1					
6	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">q</td> <td></td> </tr> </table>	q				
q						
7	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">$q \wedge \neg q$</td> <td style="padding-left: 10px;">\wedge-I, 6, 2</td> </tr> </table>	$q \wedge \neg q$	\wedge -I, 6, 2			
$q \wedge \neg q$	\wedge -I, 6, 2					
8	<table style="border-collapse: collapse;"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;">$p \wedge \neg p$</td> <td style="padding-left: 10px;">Lemma, 7</td> </tr> </table>	$p \wedge \neg p$	Lemma, 7			
$p \wedge \neg p$	Lemma, 7					
9	$p \wedge \neg p$	\vee -E, 3, 4–5, 6–8				
10	$\neg(p \vee q)$	\neg -I, 3–9				