

1 Formatting and Beautifying (ISO) Prolog

1.1 Test environment

The test environment is used as starting point to specify the different possible formattings for Prolog code. Based on this decisions, the test cases were build to guarantee the correctness of the formatter and as consequence the correctness of the produced code. Of course, the code has to be semantically and syntactic correct. Otherwise a formatter would be useless. Next, different scenarios are illustrated on the left-hand side and their corresponding formatted code on the right-hand side. On every example code of formation on the right-hand side, every formation in use is used.

Starting with an empty set of code, the formatter naturally should not add anything and left the blank line.

Figure 1: Formatting empty lines

For better readability there should be some white spaces around an operator, as you can see around `+` and `=` in the following figure 2.

<code>X is 3+A.</code>	<code>X is 3 + A.</code>
<code>X=d(2,3).</code>	<code>X = d(2,3).</code>

Figure 2: Formatting white spaces around operators

But if a comma is used for a set of arguments, different variations of white spaces could be confusing. We reduce needed white spaces to a minimum and clear all useless ones, as illustrated in 3. Of course an already optimal code should not become worse.

<code>A = d(a ,b).</code>	<code>A = d(a,b).</code>
<code>A = d(a , b).</code>	<code>A = d(a,b).</code>
<code>A = d(a , b).</code>	<code>A = d(a,b).</code>
<code>A = d(a,b).</code>	<code>A = d(a,b).</code>

Figure 3: Formatting whitespaces in complex terms

Lists are more a complex topic and is handled different among programmers. Figure 4 shows a set of formations for lists. They always result in a style like $[a,b|rest]$. If the rest is empty, we can drop it.

$[]$.	$[]$.
$[a]$.	$[a]$.
$[a, []]$.	$[a]$.
$[a []]$.	$[a]$.
$[a, b]$.	$[a, b]$.
$[a b]$.	$[a b]$.
$[a, [b, c]]$.	$[a, [b, c]]$.
$[a [b, c]]$.	$[a, b, c]$.
$[a, b c]$.	$[a, b c]$.
$[a, b, c]$.	$[a, b, c]$.
$[[a, b], c]$.	$[[a, b], c]$.
$[[a, b] c]$.	$[[a, b] c]$.
$\text{one_to_3}(L) :- L = .(1, .(2, .(3, [])))$.	$\text{one_to_3}(L) :- L = [1, 2, 3]$.
$\text{one_to_2s}(L) :- L = .(1, .(2, -))$.	
	$\text{one_to_2s}(L) :- L = [1, 2 -]$.

Figure 4: Formatting lists

Also white spaces in lists can occur but of course reduce readability as well. Figure 5 shows the corresponding scenario and the formatted solution.

$[]$.	$[]$.
$[a]$.	$[a]$.
$[a]$.	$[a]$.
$[a]$.	$[a]$.
$[a , b]$.	$[a, b]$.
$[a b]$.	$[a b]$.
$[a , [b , c]]$.	$[a, [b, c]]$.
$[a , b , c]$.	$[a, b, c]$.
$[a , b c]$.	$[a, b c]$.
$[a , b , c]$.	$[a, b, c]$.
$[[a , b] , c]$.	$[[a, b], c]$.
$[[a , b] c]$.	$[[a, b] c]$.

Figure 5: Formatting white spaces in lists

A very important topic are brackets in Prolog to see which operator is at first and has a higher priority. The priority defines how strong surrounding parameters are bound to the operator. The next scenario illustrates some examples based on math. Naturally, multiplication has a higher priority than addition in mathematics, if the addition is not in brackets. On the other hand if addition comes first, it has to be in brackets to represent a higher priority. On the other hand useless brackets can be dropped. Both test cases are illustrated in figure 10.

<pre> a + b. (a + b). a + b * c. a + (b * c). (a + b) * c. (a + b) * (a + b). a:-b,c ; d,(X is 3 * 3),!. </pre>	<pre> a + b. a + b. a + b * c. a + b * c. (a + b) * c. (a + b) * (a + b). a :- b, c ; d, X is 3 * 3, !. </pre>
---	--

Figure 10: Formatting needed brackets around operators

<pre> (a = 1 ; b = 2). (a;b;c). (a;b),(c;d). </pre>	<pre> a = 1 ; b = 2. a ; b ; c. (a ; b), (c ; d). </pre>
---	---

Figure 11: Formatting or statements

The next topic focuses on anonymous parameters in a parameter list. If an AND expression is a parameter and in brackets, it has to be in brackets after formatting the clause. For example in the first case the complete expression (b, c) is one parameter. ??.

<pre> d (a,(b,c)). b(-,[a,b],[a,b]),c. b(-,[a,(u,v),b],[a,b]),c. </pre>	<pre> d (a,(b, c)). b(-,[a,b],[a, b]), c. b(-,[a,(u, v),b],[a, b]), c. </pre>
---	--

Figure 12: Formatting or statements

The last topic is line break. When a line reaches the default or user-set linewidth the clause has to be split to several lines. Three cases are possible. First, the line fits into the line length. Second the line will be split after the `:-` and it will be checked if clause body and head are both fit in their lines. Last the clause will be split after the `:-` and at every AND (`,`) in the body 13.

```

    test (a,b,c).
test(a,b,c) :- a = b.
test(a,b,c) :- a = b, b = c, c = a, c is a.
test(a,b,c) :- a = b, b = c, c = a, a is b, b is c, c is a, a is 'very_complicated'.
test(a,b,c) :- a = b, b = c, c = a; a is b; b is c, c is a, a is 'very_complicated'.

    test (a,b,c).
test(a,b,c) :- a = b.
test(a,b,c) :-
    a = b, b = c, c = a, c is a.
test(a,b,c) :-
    a = b,
    b = c,
    c = a,
    a is b,
    b is c,
    c is a,
    a is very_complicated.
test(a,b,c) :-
    a = b,
    b = c,
    c = a ; a is b ; b is c,
    c is a,
    a is very_complicated.

```

Figure 13: Formatting needed brackets around operators