Lists are also supported in Prolog.

Up to now, we had to represent lists by suitable terms: \( \text{nil} \in \Sigma_0 \quad \text{cons} \in \Sigma_2 \)

\[
\text{len} (\text{nil}, 0). \\
\text{len} (\text{cons} (X, Xs), Y) :\neg \text{len} (Xs, Y1), Y \text{ is } Y1 + 1.
\]

stands for the list with first element \( X \) and \( Xs \) is the rest of the list.

\(?- \text{len} (\text{cons} (3, \text{cons} (7, \text{nil})), Y). \quad Y = 2\)

Instead of \( \text{nil} \) and \( \text{cons} \), Prolog has pre-defined lists built with \( [\ ] \in \Sigma_0 \), \( . \in \Sigma_2 \)

\[
\text{len} ([], 0). \\
\text{len} (. (X, Xs), Y) :\neg \text{len} (Xs, Y1), Y \text{ is } Y1 + 1.
\]

Prolog offers the following alternative ways to write lists built with \( [\ ] \) and \( . \).

\[
= \quad . (t_1, t_2) = [ t_1 \mid t_2 ] \\
= \quad . (t_1, [\ ]) = [ t_1 ] \\
= \quad . (t_1, . (t_2, . (t_3, t))) = [ t_1, t_2, t_3 \mid t ]
\]
\[( t_1 \cdot (t_2 \cdot (t_3, [3]))) = [t_1, t_2, t_3] = [t_1, t_2 \mid [t_3 \mid [3]]] \]

The shorthand notations are considered to be syntactically identical to the corresponding term built with \[\cdot\] and \[\mid\].

?- \[(1,2) \equiv [1 \mid [2])\].
\[\text{true}\]

?- \[(1,X) = [1,2,3]\].
\[X = [2,3]\]

?- \[[X, [1 \mid X3]] = [2,3] \mid Y\].
\[X = [2], \ Y = [1,2]\]

Ex: List Concatenation (append)

\[\text{app}([3, Ys, Ys]).\]
\[\text{app}([X|Xs], Ys, [X|ts]) :- \text{app}(Xs, Ys, ts).\]

?- \text{app}([1,2], [3,4,5], Xs).
\[Xs = [1,2,3,4,5]\]
\( \text{app}(X, Y, [1,2,3]) \).

\( X = [1], \ Y = [1,2,3] \);

\( X = [1,2], \ Y = [2,3] \);

...