Formal translation of generalized path expressions

This page contains the translation for rules that contains a path expression with either the "star" (*) or the "cross" (+) operators.

Consider the following abstract rule with the "star" (*) operator:

```
if C@Class(RestCondition,Path_A.Class_A.*.Class_B.Path_B Predicate)
then Conclusion
```

which is translated into the following set of production and deductive rules.

```
if Class_C@xml_seq(elem_order ∋ Class_D) and
   not Class_D@xml_seq(elem_order ∋ Class_D)
then tmp_elem1(cnd_elem:Class_D,path_string:'Class_D.Path_B',recursive:no)

if Class_C@xml_seq(elem_order ∋ Class_D) and
   Class_D@xml_seq(elem_order ∋ Class_D)
then tmp_elem1(cnd_elem:Class_D,path_string:'Class_D*.Path_B',recursive:yes)

if XX1@tmp_elem1(cnd_elem:XX2 TermCond_1,path_string:XX3,recursive:XX4) and
   XX2@xml_seq(elem_order ∋ XX5 TermCond_1) and
   not XX5@xml_seq(elem_order ∋ XX5)
then tmp_elem1(cnd_elem:XX5,path_string:'XX5.XX3',recursive:XX4)

if XX1@tmp_elem1(cnd_elem:XX2 TermCond_1,path_string:XX3,recursive:no) and
   XX2@xml_seq(elem_order ∋ XX4 TermCond_1) and
   XX4@xml_seq(elem_order ∋ XX4)
then tmp_elem1(cnd_elem:XX4,path_string:'XX4*.XX3',recursive:yes)

if XX1@tmp_elem1(cnd_elem:XX2 TermCond_1,path_string:XX3,recursive:XX4) and
   TermCond_2
then tmp_elem2(path_string:'Class_A.XX3',recursive:XX4)

if XX1@tmp_elem2(path_string:XX2,recursive:no)
then new_rule('if C@Class(RestCondition,Path_A.XX2 Predicate)
               then Conclusion')
     => deductive_rule

if XX4@tmp_elem2(path_string:XX5,recursive:yes) and
   prolog{split_path(XX5,XX6,XX7,XX8)}
then new_rule('if C@Class(RestCondition,Path_A.XX6.XX7.XX8 Predicate)
               then Conclusion')
     => deductive_rule,
     new_rule('if C@Class(RestCondition,XX7.XX8 ∋ XX1)
               then tmp_elem3(cnd_obj:XX1)')
     => deductive_rule,
     new_rule('if XX2@tmp_elem3(cnd_obj:XX1) and
                  XX1@XX7(XX7 ∋ XX3)
               then tmp_elem3(cnd_obj:XX3)')
     => deductive_rule,
     new_rule('if XX1@tmp_elem3(Path_A.XX6.cnd_obj Predicate)
               then Conclusion')
     => deductive_rule
```

In the above set of rules, the following macro-symbols have been used:

$$Class_C = \begin{cases} Path_B[1], & if \;\; Path_B \neq \varnothing \\ Class, & if \;\; Path_B = \varnothing \end{cases}$$

$$Class_D = \begin{cases} Class_B, & if \;\; Class_B \neq \varnothing \\ < \text{var} >, & if \;\; Class_B = \varnothing \end{cases}$$

$$TermCond_1 = \begin{cases} '\backslash = Class_A', & if \;\; Class_A \neq \varnothing \\ \varnothing, & if \;\; Class_A = \varnothing \end{cases}$$

$$TermCond_2 = \begin{cases} 'XX2@xml\_seq(elem\_order \ni Class_A)', & if \;\; Class_A \neq \varnothing \\ 'not \;\; XX2@xml\_seq', & if \;\; Class_A = \varnothing \end{cases}$$

Consider the following abstract rule with the "cross" (+) operator:

```
if C@Class(RestPreds,Path.+ Preds)
then Conclusion
```

which is translated into the following two X-DEVICE rules:

```
if C@Class(RestPreds,Path.* Preds)
then tmp_elem1(TmpVars,tmp_obj:C)

if XX1@tmp_elem1(TmpVars,tmp_obj:XX2) and
   not( XX3@tmp_elem1(tmp_obj:XX4\=XX2) and
        XX2@Class(* ∍ XX4) )
then Conclusion
```