Formal translation of ordering expressions

This page contains the general procedure for translating X-DEVICE rules that contain ordering expressions.

Complex ordering expressions

Consider the following abstract X-DEVICE rule:

```
if Cond1 and
   C@class(Preds,PathExpr ϶{rel_op, abs_op} Var) and
   Cond2
then Conclusion
```

which is translated into the following two rules:

```
if Cond1 and
   C@class(Preds,PathExpr ϶rel_op XX1)
then tmp_elem1(tmp_obj:list(XX1))

if XX1@tmp_elem1(tmp_obj ϶abs_op Var) and
   Cond2
then Conclusion
```

Relative ordering expressions

Consider the following abstract X-DEVICE rule:

```
if Cond1 and
   C1@class1(Preds1,alt_elem1.PathExpr1 Op1 Var1) and
   C2@class2(Preds2,alt_elem2.PathExpr2 ϶rel_op(Var1) Var2) and
   Cond2
then Conclusion
```

which is translated into the following rule:

```
if Cond1 and
   C1@class1(Preds1,xml_alt_class1.PathExpr1 Op1 XX1) and
   XX1@xml_alt_class1(alt_elem1 ϶ Var1) and
   C2@class2(Preds2,xml_alt_class2.PathExpr2 ϶rel_op|XX1| XX2) and
   XX2@xml_alt_class2(alt_elem2 ϶ Var2) and
   Cond2
then Conclusion
```

Multiple ordering expressions

Consider the following abstract X-DEVICE rule:

```
if Cond0 and
   C1@class1(PathExpr1 ListOp1 Var1) and
   Cond1(Var1) and
   C2@class2(PathExpr2 ListOp2 Var2) and
   Cond2(Var2) and
   ...
   Cn@classn(PathExprn ListOpn Varn) and
   Condn(Varn) and
```

```
    Cond_rest
then Conclusion
```

which is translated into the following set of rules:

```
if Cond_0 and
   C_1@class_1(PathExpr_1 ListOp_1 Var_1) and
   Cond_1(Var_1)
then tmp_elem_1(Tmp_vars_1,tmp_obj_1:list(Var_1))

if XX1@tmp_elem_1(Tmp_vars_1,tmp_obj_1:Var_1) and
   C_2@class_2(PathExpr_2 ListOp_2 Var_2) and
   Cond_2(Var_2)
then tmp_elem_2(Tmp_vars_2,tmp_obj_1:Var_1,tmp_obj_2:list(Var_2))

...

if XX1@tmp_elem_n-1(Tmp_vars_n-1,tmp_obj_1 ∋ Var_1,tmp_obj_2 ∋ Var_2,...,tmp_obj_n-1 ∋ Var_n-1) and
   C_n@class_n(PathExpr_n ListOp_n Var_n) and
   Cond_n(Var_n) and
   Cond_rest
then Conclusion
```

Single ordering expression

Consider the following abstract X-DEVICE rule:

```
if Cond_1 and
    C@class(Preds,PathExpr ∋_ListOp Var) and
    Cond_2
then Conclusion
```

which is translated into the three following rules:

```
if Cond_1 and
   C@class(Preds,PathExpr ∋ XX1)
then tmp_elem1(Tmp_vars,tmp_obj:list(XX1))

if XX3@tmp_elem1(Tmp_vars,tmp_obj:XX1) and
   prolog{select_sub_list(ListOp,XX1,XX2)}
then tmp_elem2(Tmp_vars,tmp_obj:XX2)

if XX1@tmp_elem2(Tmp_vars,tmp_obj ∋ Var) and
   Cond_2
then Conclusion
```

The Prolog predicate `select_sub_list` implements the different ordering expressions on a list of objects. For example, the absolute ordering expression `=<N` is implemented as follows:

```
select_sub_list('=<'(N),ListIn,ListOut) :-
   first_n(1,N,ListIn,ListOut).
first_n(X,N,_,[]) :-
   X > N, !.
first_n(X,N,[H|T],[H|T1]) :-
   X1 is X + 1,
   first_n(X1,N,T,T1).
first_n(_,_,[],[]).
```

The following is an example of the implementation of a relative ordering expression, namely the expression `between(E1,E2)`:

```
select_sub_list(between(E1,E2),ListIn,ListOut) :-
   append(_,[E1|L],ListIn),
   append(ListOut,[E2|_],L), !.
select_sub_list(between(_,_),[],[]).
```

Similar implementations exist for all the other types of ordering expressions.