grammar org.xtext.example.mydsl.TR with org.eclipse.xtext.common.Terminals

generate tR "http://www.xtext.org/example/mydsl/TR"

Model:

 greetings+=TRprogram\*;

TRprogram:

 ('enums'

   (enums+=Enums)\*

 'end')? &

 'percepts'

   (percepts+=Percepts)+

 'end' &

 ('durative'

   (durative+=Durative)\*

 'end')? &

 ('discrete'

   (discrete+=Discrete)\*

 'end')? &

 ('beliefs'

   (beliefs+=Beliefs)\*

 'end')? &

 ('vars'

   (vars+=Vars)\*

 'end')? &

 'goals'

 (goals+=Goals)+

 'end' &

 ('messages'

   (msgs+=Messages)\*

   'end')?

   &

 ('procedures'

   (proc+=Procedures)\*

   'end')?

 /\*  &

 ('relations'

   (rel+=Relations)\*

   'end')? \*/

 ;

Enums:

 name=ID '::=' (r+=EnumElem)+

;

EnumElem:

 name=ID

;

FQN: ID ("." ID)\* ;

Condition:

 (sc=Simple\_condition (('and'|'or') conds+=Simple\_condition)\*)

;

Simple\_condition: Predication | ('not' Predication);

PerBel: Percepts | Beliefs;

VarPar:  Vars | Param ;

SuperVarPar:

 v=[VarPar] e=[EnumElem|FQN]

;

PerVarParam: p=[PerBel] '(' (p2=Value)? (',' p3+=Value)\*  ')' | v=SuperVarPar

;

Predication:

 pv=PerVarParam | left=Arith\_term ('<'|'>'|'='|'<='|'>='|'!=') right=Arith\_term | 'true'

;

Arith\_term: {Arith\_term}

 (INT | DOUBLE | pv=PerVarParam)

 (('+'|'-'|'\*'|'/'|'rem'|'div') arit=Arith\_term)?

;

/\* end conditions for Relations

Relations:

 name=ID ':''('((type=Type)(',' types+=Type)\*)?')'

 name2=ID '('(p=[VarPar] (';' params+=[VarPar])\*)?')' '<=' r=RelCondition

;

\*/

Procedures:

 name=ID ':''('((type=Type)(',' types+=Type)\*)?')'

 name2=ID '('(p=Param (';' params+=Param)\*)?')' '->>'

 p2=Plusplus (';' p3+=Plusplus)\*

;

RemFor: {RemFor}

 ('remember' | 'forget') b=[Beliefs] '(' (p2=Value)? (',' p3+=Value)\* ')' | 'forgetAll'

;

Plusplus: {Plusplus}

 rf+=RemFor | (v+=[Vars]':=' at=Arith\_term) | STRING 'to' Agent | (v+=[Vars]'+:=' at=Arith\_term) | (v+=[Vars]'-:=' at=Arith\_term)

;

Agent: {Agent}

 STRING | p=[Param]

;

Rule:

 Condition (('while' (cw=Condition | Number))? & ('until' (cu=Condition | Number))?) '->' ar=ActionRule (ts=TimedSeq)? ('++' p=Plusplus (';' p2+=Plusplus)\*)?

;

TimedSeq:

 ('for' Number ';' ar3+=DurActionRule 'for' Number)+

;

Actions:

 Durative | Discrete | Goals

;

ActionRule: {ActionRule}

(mya=[Actions] '(' (p=Arith\_term (',' params+=Arith\_term)\*)? ')') | '('')'

;

DurActionRule:

mya=[Durative] '(' (p=Arith\_term (',' params+=Arith\_term)\*)? ')'

;

Goals:

 name=ID ':''('((type=Type)(',' types+=Type)\*)?')'

 name2=ID '('(p=Param (',' params+=Param)\*)?')'

 '{'

 rule+=Rule\*

 '}'

;

Param: name=ID

;

Percepts:

 name=ID ':' '('((type=Type)(',' types+=Type)\*)?')'

;

Beliefs:

 name=ID ':' '('((type=Type)(',' types+=Type)\*)?')'

;

Durative:

 name=ID ':' '('((type=Type)(',' types+=Type)\*)?')'

;

Discrete:

 name=ID ':' '('((type=Type)(',' types+=Type)\*)?')'

 ;

Vars:

 Type name=ID (':=' value=Primitive\_Value)?

;

Messages:

 // from, msg, params

 "handle" '('p1=MsgParam ',' p2=MsgParam ',' p3=MsgParam2')' ('when' c=Condition)? ('->' (p+=Plusplus (',' p4+=Plusplus)\*))?

;

MsgParam: {MsgParam}

 '\_' | STRING

;

MsgParam2: {MsgParam2}

 m=Param | '\_'

;

Primitive\_Type: {Primitive\_Type}

 ('int' | 'double' | 'string')

;

Type:

 Primitive\_Type | enumerate=[Enums]

;

Value: {Value}

 Primitive\_Value | value=[EnumElem|FQN]

;

Primitive\_Value:

 INT | DOUBLE |STRING |'\_'

;

terminal DOUBLE:

    INT '.' INT

;

Number:

 DOUBLE | INT

;