```
grammar it.unibo.Qactork with org.eclipse.xtext.common.Terminals
generate qactork "http://www.unibo.it/Qactork"

QActorSystem: "System" ( trace ?= "-trace" )? (  logmsg ?= "-msglog" )? spec=QActorSystemSpec ;

terminal VARID  :  ('A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;
terminal KCODE  : '#' ( . )* '#'  ;
terminal PCOLOR : 'blue' | 'red' | 'green' | 'black' | 'yellow' | 'magenta' | 'cyan' | 'gray';

QualifiedName : ID ('.' ID)* ;

QActorSystemSpec:
         name=ID
        ( mqttBroker = BrokerSpec)?
        ( libs       = UserLibs  )?
        ( message   += Message   )*
        ( context   += Context   )*
        ( actor     += QActorDeclaration    )*
        (display    = DisplayDecl)?
        (facade     = FacadeDecl)?
;
BrokerSpec : "mqttBroker" brokerHost=STRING ":" brokerPort=INT "eventTopic" eventtopic=STRING;

/*
 * ------------------------------------------
 * UserLib JAN 24
* ------------------------------------------
 */
UserLibs : "UserLibs" (lib += UserLib)+ ;
UserLib  : "-f"  file=STRING  ;

/*
 * ------------------------------------------
 * MESSAGE
* ------------------------------------------
 */
//Message :                OutOnlyMessage | OutInMessage | BasicMessage ;
//OutOnlyMessage :      Dispatch |  Event | Signal | Token ;
//OutInMessage:                 Request  | Reply | Invitation ;
Message :           BasicMessage | Event | OtherMsg;
BasicMessage:       Dispatch | Request;
OtherMsg    :       Reply | Invitation | Signal | Token ;

Event:          "Event"      name=ID  ":" msg = PHead  (cmt=STRING)?;
Signal:         "Signal"     name=ID  ":" msg = PHead  (cmt=STRING)?;
Token:          "Token"      name=ID  ":" msg = PHead  (cmt=STRING)?;
Dispatch:       "Dispatch"   name=ID  ":" msg = PHead  ( cmt=STRING)?;
Request:        "Request"    name=ID  ":" msg = PHead  (cmt=STRING)?;
Reply:      "Reply"      name=ID  ":" msg = PHead ( "for" reqqq = [Request] )? (cmt=STRING)?;
Invitation:     "Invitation" name=ID  ":" msg = PHead  (cmt=STRING)? ;

/*
 * Context
 */
Context : "Context"  name=ID "ip"  ip = ComponentIP   ( "commonObj" commonObj = STRING)? ( mqtt ?=
"+mqtt" )?;
ComponentIP : {ComponentIP} "[" "host=" host=STRING "port=" port=INT "]" ;

/*
 * QActor
 */
QActorDeclaration : QActorInternal | QActorExternal  ;
QActorInternal:  QActor | QActorCoded;
QActorExternal    : "ExternalQActor" name=ID "context" context = [ Context ] ;
QActorCoded       : "CodedQActor"    name=ID "context" context = [ Context ] "className" className
= STRING ( dynamic ?= "dynamicOnly")?;
QActor            : "QActor"         name=ID "context" context = [ Context ]  ("withobj" withobj =
WithObject)? ( dynamic ?= "dynamicOnly")?
    "{"
```

```
        ( imports += UserImport )* //JAN24
        ( start   = AnyAction )?
                ( states += State )*
        "}"
;

WithObject : name=ID  "using" method = STRING;


/*
 * UserImport  JAN24
 */
UserImport : "import" file = STRING ;


/*
 * State
 */
State :
        "State" name=ID  ( normal ?= "initial" )?
        //actionseq = ActionSequence
        "{" ( actions += StateAction )*  "}"
        ( transition = Transition )?
;

/*
 * StateAction
 */
StateAction:
/*1*/ AnyAction |
/*2*/ Forward|Demand|Answer|ReplyReq|AutoMsg|AutoRequest|
/*3*/ MsgCond | GuardedStateAction |    IfSolvedAction |
/*4*/ MqttConnect | Publish | Subscribe | SubscribeTopic|
/*5*/ Emit |  EmitLocal |  EmitLocalStream |
/*6*/ UpdateResource |  ObserveResource  |
/*7*/ Delegate |  DelegateCurrent       |
/*8*/ SolveGoal |
/*9*/ CreateQActor | ExecResult |
/*10*/ ReturnFromInterrupt |
/*11*/ CodeRunSimple | CodeRunActor | MachineExec |
/*12*/ Print | PrintCurMsg | DiscardMsg |
/*13*/  DelayInt  | MemoTime | Duration |
/*14*/ EndActor |


;
IfSolvedAction     : {IfSolvedAction} "ifSolved" "{" ( solvedactions += StateAction )*  "}"
//action=ActionSequence
                                        ("else"  "{" ( notsolvedactions += StateAction )*  "}")?
;
GuardedStateAction : {GuardedStateAction} "if" guard = AnyAction "{" ( okactions += StateAction )*
"}"  //action=ActionSequence
                                        ("else"  "{" ( koactions += StateAction )*  "}")?
;


PrintCurMsg    :  {PrintCurMsg} "printCurrentMessage"  ("color" color=PCOLOR )? ;
Print          :  {Print} "println" "(" args=PHead ")" ("color" color=PCOLOR )?   ;
//Printcolored   :  {Printcolored} "printlncolor" "(" args=PHead ")" "color"color=PCOLOR     ;

SolveGoal      :  {SolveGoal} "solve" "(" goal=PHead ("," resVar=Variable)? ")";
DiscardMsg     :  {DiscardMsg} "discardMsg" (discard?='On' | 'Off') ;
MemoTime       :  {MemoTime} "memoCurrentTime"  store=VARID ;
Duration       :  {Duration} "setDuration" store=VARID "from" start=VARID;

Forward   : "forward" dest=[QActorDeclaration] "-m" msgref=[Dispatch] ":" val = PHead ;
//ExecutorForward   : "forwardToExecutor" dest=STRING "-m" msgref=[Dispatch] ":" val = PHead ;
//lo fa il generatore
Emit      : "emit" msgref=[Event] ":" val = PHead       ;
```

```
EmitLocal : "emitlocal" msgref=[Event] ":" val = PHead  ;
EmitLocalStream : "emitlocalstream" msgref=[Event] ":" val = PHead      ;
Demand    : "request" dest=[QActorDeclaration] "-m" msgref=[Request]  ":" val = PHead ;
Answer    : "replyTo" reqref=[Request] ("ofsender" sender=VarRef)?  "with"  msgref=[Reply]   ":"
val = PHead ( "caller==" dest=[QActorDeclaration])?;
ReplyReq  : "ask"  reqref=[Request] ":" val = PHead  "forrequest" msgref=[Request] ( "caller=="
dest=[QActorDeclaration])?;

AutoMsg         : "autodispatch" msgref=[Dispatch]  ":" val = PHead ;
AutoRequest     : "autorequest"  msgref=[Request]   ":" val = PHead ;

//Feb2024
MqttConnect     : "connectToMqttBroker" brokerAddr=STRING;
Publish         : "publish"  topic=STRING "-m" msgref=[Event] ":" val = PHead ;
SubscribeTopic  : "subscribe" topic=STRING;

Delay     : DelayInt | DelayVar | DelayVref | DelaySol ;
DelayInt  : "delay" time=INT  ;
DelayVar  : "delayVar"    refvar     = Variable ;
DelayVref : "delayVarRef" reftime    = VarRef ;
DelaySol  : "delaySol"    refsoltime = VarSolRef ;
MsgCond   :     "onMsg" "(" message=[Message] ":" msg = PHead ")" "{" ( condactions += StateAction
)*  "}"
                         ("else" ifnot = NoMsgCond )? ;
EndActor  : "terminate" arg=INT;

ReturnFromInterrupt :  {ReturnFromInterrupt}  "returnFromInterrupt" memo=STRING?;
UpdateResource      :  {UpdateResource}  "updateResource"  val=AnyAction     ;
//ObserveResource     :  {ObserveResource} "observeResource" resource=[QActor] ;
ObserveResource     :  {ObserveResource} "observeResource" resource=[QActorDeclaration] ("_"
suffix=STRING)? ("msgid" msgid=[Dispatch] )?;
//ObserveDynamicActor :  {ObserveDynamicActor} "observeDynamicActor" resource=[QActorDeclaration]
("_" suffix=STRING)?;

Subscribe           : "subscribeTo" localactor=[QActor] ("_" suffix=STRING)?;
//Delegate            : "delegate" msg= STRING "to" localactor=[QActor];
Delegate            : "delegate" msg=[BasicMessage] "to" localactor=[QActor];  //JAN24




NoMsgCond :     {NoMsgCond}  "{" (  notcondactions += StateAction )*  "}"  ;
AnyAction : {AnyAction}   "[" body=KCODE "]"  ;
//"[" body=STRING "]";

CodeRun        : CodeRunActor | CodeRunSimple  ;
CodeRunActor   : "qrun"   aitem=QualifiedName "(" "myself" ( "," args+=PHead ("," args+=PHead)* )?
")" ;
CodeRunSimple  : "run"    bitem=QualifiedName "(" (args+=PHead ("," args+=PHead)* )? ")";

MachineExec    : "machineExec" action=STRING ;


CreateQActor  : "create"  executor=[QActorDeclaration]
        ("_" suffix=STRING)? (confined="confined")?  (outinforeply=OutInforReply)?
(outinfoevent=OutInfoevent)? ;

OutInforReply : "requestbycreator"  msgref=[Request]  ":" val = PHead ;
OutInfoevent  : "emitforcreator"    msgref=[Event] ;//":" val = PHead   ;
ExecResult    : "execresultReplyTo" reqref=[Request]  "with"  msgref=[Reply]   ":" val = PHead ;

//JAN24
DelegateCurrent : "delegateCurrentMsgTo" localactor=[QActor];


//CreateObject  : "object" name=ID  "using" method = STRING; //promoted factory and singleton

//OutInfoEvent: "emit" event=EmitLocalStream ;
```

```
/*
 * Transition
 */
Transition         :  EmptyTransition | NonEmptyTransition  ;
EmptyTransition    : "Goto" targetState=[State]  ("if" eguard=AnyAction "else" othertargetState=
[State] )?  ;

NonEmptyTransition :  "Transition" name=ID  (duration=Timeout)?   ( trans += InputTransition)*
("else" elseempty=EmptyTransition)?;
Timeout            : TimeoutInt | TimeoutVar | TimeoutSol | TimeoutVarRef; //| InterruptMsg;
TimeoutInt         : "whenTime"    msec=INT                    "->" targetState = [State]  ;
TimeoutVar         : "whenTimeVar"    variable   = Variable    "->" targetState = [State]  ;
TimeoutVarRef      : "whenTimeVarRef" refvar     = VarRef      "->" targetState = [State]  ;
TimeoutSol         : "whenTimeSol"    refsoltime = VarSolRef   "->" targetState = [State]  ;


InputTransition    : EventTransSwitch | MsgTransSwitch | RequestTransSwitch |  ReplyTransSwitch |
InterruptTranSwitch | InterruptEvent ;
InterruptTranSwitch: "whenInterrupt"   message=[Dispatch] ("and"  guard=AnyAction  )?  "->"
targetState=[State]  ;
InterruptEvent     : "whenInterruptEvent"   message=[Event] ("and"  guard=AnyAction  )?  "->"
targetState=[State]  ;
EventTransSwitch   : "whenEvent"   message=[Event]    ("and"  guard=AnyAction  )?  "->"
targetState=[State]  ;
MsgTransSwitch     : "whenMsg"     message=[Dispatch] ("and"  guard=AnyAction  )?  "->"
targetState=[State]  ;
RequestTransSwitch : "whenRequest" message=[Request]  ("and"  guard=AnyAction  )?  "->"
targetState=[State]  ;
ReplyTransSwitch   : "whenReply"   message=[Reply]    ("and"  guard=AnyAction  )?  "->"
targetState=[State]  ;

/*
 * PROLOG like
 */
PHead : PAtom | PStruct | PStructRef ;
PAtom : PAtomString | Variable | PAtomNum | PAtomic | VarRef | VarSolRef | VarRefInStr;
PAtomString : val = STRING ;
PAtomic     : val = ID ;
PAtomNum    : val = INT ;
PStructRef  : "$" struct = PStruct;  //
PStruct     : functor=ID "(" (msgArg += PHead) ("," msgArg += PHead)* ")" ;  //At least one arg is
required
Variable    : {Variable} varName= VARID ;
//USING vars (from solve or from code)
VarRef      : "$" varName= VARID ;      //in msg payload  e.g. modelChange(robot,$Curmove) =>
$Curmove
VarRefInStr : "#" varName= VARID ;      //in msg payload. e.g. modelChange(robot,#M)       =>
${getCurSol("M").toString()}
VarSolRef   : "@" varName= VARID ;      //in run          e.g. run itunibo....doMove( @M ) =>
getCurSol("V").toString()
```