Rapport IDM
Modélisation, Vérification et Génération de Jeux

2ème Année, Département des Sciences Numériques - Parcours Logiciel

Oussama Echcherqaoui
Mehdi Sensali
Reda El Jai
Younes Saoudi

2020 - 2021

Contents

Introduction

Ce projet consiste à produire une chaîne de modélisation, de vérification et de génération de code pour des jeux de parcours/découverte.

La modélisation consiste à concevoir un langage dédié pour décrire le jeu sous la forme d'un modèle et à implanter les outils d'édition, vérification et génération associés.

La vérification permet d'assurer qu'il existe une solution pour le jeu décrit par un modèle. Pour répondre à cette question, nous utilisons les outils de model-checking définis sur les réseaux de Petri au travers de la boîte à outils TINA. Il nous faudra donc traduire un modèle de jeu en un réseau de Petri.

Tâches Réalisées

2.1 Modèle `XText`

Voir script `XPDL.xtext`

2.2 Métamodèle Généré

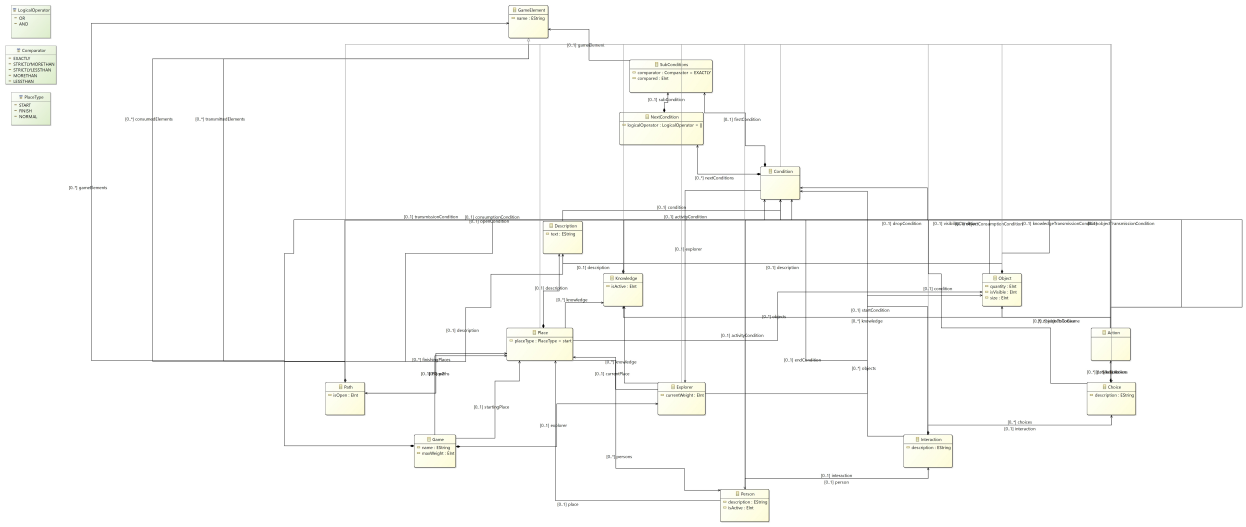

Figure 2.1: Métamodèle Game

2.3 Modèle de Jeux : Enigme

2.3.1 `XText`

```
game EnigmaGame is {
  explorer Hamid {
      has objects (Attempt)
      with weight 3
      is in Enigma
  } with maxWeight 3

  contains {
    place Enigma  is start "Starting Place" {
```

```
      persons (Phoenix)
      paths (EnigmaToSuccess, EnigmaToFailure)
    },
    place Success is finish "Success!" {
      paths (EnigmaToSuccess)
    },
    place Failure is finish "Failure!" {
      paths (EnigmaToFailure)
    },

    path EnigmaToSuccess is "PATH TO SUCCESS" {
      open 0 if hasAnsweredCorrectly
      from Enigma to Success
    },
    path EnigmaToFailure is "PATH TO FAILURE" {
      open 0 if hasFailedEnigma
      from Enigma to Failure
    },

    object Attempt : 3 is "ATTEMPTS TO ANSWER ENIGMA" {
      visible 1
      size 1
    },
    knowledge Victory is "VICTORY IS REQUIRED FOR SUCCESS (lol)" {
      active 1
    },

    condition hasFailedEnigma on Hamid {
      possesses EXACTLY 0 Attempt
    },
    condition hasAnsweredCorrectly on Hamid {
      possesses EXACTLY 1 Victory
    },
    condition canAnswer on Hamid{
      possesses EXACTLY 0 Victory AND possesses MORETHAN 1 Attempt
    },

    person Phoenix is "Phoenix's Famous Question" {
      active 0 if canAnswer
      in Enigma
      interactable with AnswerPhoenix
    },

    interaction AnswerPhoenix is "Interact with Phoenix" with Phoenix {
        choose from (choice Answer of AnswerPhoenix is "Answer Enigma"{
                (action wrongAnswer of Answer  {consumes objects (Attempt)},
                action rightAnswer of Answer {gives knowledge (Victory)})
              })
    }
  }
  from Enigma to (Success, Failure)
}
```

2.3.2 Transformation en réseau de Petri

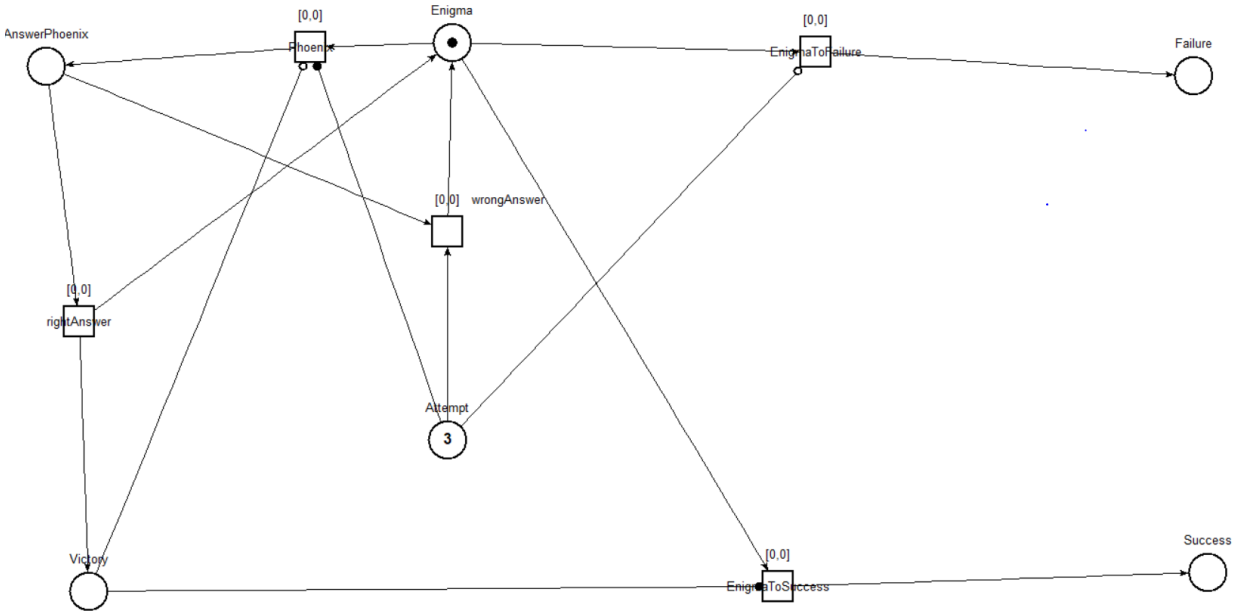

Figure 2.2: Transformation du jeu Enigma en Réseau de Pétri

Voir script `Game2PetriNet.atl`

2.3.3 Propriétés LTL

```
# Game End Assertion
op finished = Success \/ Failure;
op started = Enigma;

# A finished game doesn't evolve anymore
[] (finished => dead);
# A game will always eventually stop evolving
[] <> dead;
# A game not evolving is a finished one
[] (dead => finished);
# A game never finishes
- <> finished;
# A finished game will not restart
[] (finished => - <> started);
```

2.3.4 Infrastructure Java

Lors de l'implémentation en Java, nous avons essayé d'être le plus fidèle possible au métamodèle. Nous retrouvons ainsi une classe correspondante à chaque eClass dans le package My_Game. Voici une vue d'ensemble des sous packages composants GAME.

- `Display` : Contient les classes qui permettent d'afficher le jeu
- `GameController` : Contient les classes qui permettent de contrôler la dynamique du jeu

  - `Controller` : Classe abstraite contenant la methode play() qui fait jouer l'utilisateur
  - `ControllerImpl` : Implémentation du Controller

- `My_Game` : Contient une classe pour toute eClass du métamodèle, à savoir : Action, Interaction, Path, Place, Choice, ConditionKnowledge, Object, Person (qui héritent tous de GameElement), puis Game, Explorer, LogicalOperator, SubConditions, Comparator, Description.
- `test` : contient l'implémentation en java des exemples étudiés
  - Enigme.java
  - Exams.java

2.4 Autres Exemple

2.4.1 Exemple de Passation d'Examens

Le concept de ce jeu est simple: Khalid est un étudiant (explorateur) en pleine période d'examens. Il doit alors se rendre au lieu FirstSession pour intéragir avec la "personne" FirstSessionExam. Il peut alors soit réussir ses examens en obtenant la connaissance PassedExams le permettant de valider son année en allant au lieu Succès, soit les rater et devoir participer à SecondSession ce qui lui fait perdre son objet FirstSessionAttempt en échange de l'objet SecondSessionAttempt.

Dans le lieu SecondSession se trouve la "personne" SecondSessionExam avec laquelle Khalid peut intéragir afin de sauver sa peau: Il peut ainsi soit réussir soit rater ses examens encore une fois. Dans le dernier cas, il perd SecondSessionAttempt et est dirigé vers le bureau du directeur (lieu nommé ici RetakeYear) qu'il doit supplier afin de pouvoir redoubler son année. Le directeur l'avertit que c'est la dernière fois et lui prend son objet YearAttempt mais lui donne une seconde chance: un autre objet FirstSessionAttempt.

Khalid va alors repasser la première session de l'année prochaine puis soit réussir soit passer à la deuxième session encore une fois, mais cette fois-ci il n'y a plus de redoublement: s'il ne réussit pas cette seconde session, il ira directement à Failure.

Ceci donne le script suivant:

```
game Exams is {
  explorer Khalid {
    has objects(FirstSessionAttempt, YearAttempt)
    with weight 2
    is in FirstSession
  } with maxWeight 3

  contains {
    person FirstSessionExam is "Answer Exam!" {
      active 0 if canAttendFirstSession
      in FirstSession
      interactable with TakeFirstSession
    },
    interaction TakeFirstSession is "Take the First Session Exams!" with FirstSessionExam {
      choose from (choice FirstSessionAnswer of TakeFirstSession is "Answer Exam" {
        (action failFirstSessionExam of FirstSessionAnswer{gives objects (
    SecondSessionAttempt) consumes objects (FirstSessionAttempt)}, action
    passFirstSessionExam of FirstSessionAnswer {gives knowledge (PassedExams)})
      })
    },
    person SecondSessionExam is "Answer Exam!" {
      active 0 if canAttendSecondSession
      in SecondSession
      interactable with TakeSecondSession
    },
    interaction TakeSecondSession is "Take the Second Session Exams!" with SecondSessionExam
     {
```

```
      choose from (choice SecondSessionAnswer of TakeSecondSession is "Answer Exam" {
        (action failSecondSessionExam of SecondSessionAnswer{ consumes objects (
    SecondSessionAttempt)}, action passSecondSessionExam of SecondSessionAnswer {gives
    knowledge (PassedExams)})
      })
    },
    person Director is "Beg to retake year" {
      active 0 if canRetakeYear
      in RetakeYear
      interactable with BegToRetakeYear
    },
    interaction BegToRetakeYear is "Beg me!" with Director {
      choose from (choice BegDirector of BegToRetakeYear is "BEEEG" {
        (action Beg of BegDirector{gives objects (FirstSessionAttempt) consumes objects (
    YearAttempt)})
      })
    },
    place FirstSession is start "Exams Room!" {
      persons (FirstSessionExam)
      paths (FirstSessionToSuccess, FirstSessionToSecondSession, RetakeYearToFirstSession)
    },
    place SecondSession is normal "Second Sessions Exams!"{
      persons (SecondSessionExam)
      paths (SecondSessionToSuccess, SecondSessionToFailure, FirstSessionToSecondSession,
    SecondSessionToRetakeYear)
    },
    place Success is finish "Successfully Passed Exams" {
      paths (FirstSessionToSuccess, SecondSessionToSuccess)
    },
    place Failure is finish "Failed Exams" {
      paths (SecondSessionToFailure)
    },
    place RetakeYear is normal "Retake Year" {
      persons (Director)
      paths (SecondSessionToRetakeYear, RetakeYearToFirstSession)
    },

    path FirstSessionToSuccess is "PATH TO SUCCESS" {
      open 0 if hasPassedExam
      from FirstSession to Success
    },
    path FirstSessionToSecondSession is "ANOTHER PATH TO SUCCESS" {
      open 0 if canAttendSecondSession
      from FirstSession to SecondSession
    },

    path SecondSessionToSuccess is "PATH TO SUCCESS" {
      open 0 if hasPassedExam
      from SecondSession to Success
    },
    path SecondSessionToRetakeYear is "Another Year Another Me!" {
      open 0 if canRetakeYear
      from SecondSession to RetakeYear
    },
    path SecondSessionToFailure is "PATH TO FAILURE" {
      open 0 if hasFailedExams
      from SecondSession to Failure
    },
    path RetakeYearToFirstSession is "PATH TO SUCCESS 2.0" {
      open 0 if canAttendFirstSession
      from RetakeYear to FirstSession
    },

    object FirstSessionAttempt:1 is "Attempts at First Session" {
      visible 0
```

```
      size 1
    },
    object SecondSessionAttempt:1 is "Attempts at First Session" {
      visible 0
      size 1
    },

    knowledge PassedExams is "Finally passed exams!" {
      active 0
    },
    object YearAttempt:1 is "Finally can go home" {
      visible 0
      size 1
    },

    condition hasPassedExam on Khalid {
      possesses MORETHAN 1 PassedExams
    },
    condition canAttendFirstSession on Khalid {
      possesses MORETHAN 1 FirstSessionAttempt
    },
    condition canAttendSecondSession on Khalid {
      possesses MORETHAN 1 SecondSessionAttempt
    },
    condition hasFailedExams on Khalid {
      possesses EXACTLY 0 SecondSessionAttempt AND possesses EXACTLY 0 FirstSessionAttempt
    AND possesses EXACTLY 0 YearAttempt
    },
    condition canRetakeYear on Khalid {
      possesses EXACTLY 0 SecondSessionAttempt AND possesses EXACTLY 0 FirstSessionAttempt
    AND possesses EXACTLY 1 YearAttempt
    }
  }
  from FirstSession to (Success, Failure)
}
```

2.5 Contraintes `OCL`

Les contraintes que chaque métamodèle Game doit vérifier sont les suivantes:

- Les noms des éléments doivent être valides
- Il doiy y avoir un unique lieu de départ valide
- Il doit y avoir au moins un lieu d'arrivée
- La taille d'un objet doit être positive
- La quantité d'un objet dans l'inventaire de l'explorateur doit être positive
- L'attribut `currentWeight` de l'explorateur ne doit pas dépasser la taille maximale
- Un chemin doit avoir des extrémités différentes
- Les attributs `transmittedElements` et `consumedElements` des `Path` doivent être de type `Knowledge` ou `Object`
- Deux lieux différents ont des noms différents
- Deux personnes différentes ont des noms différents
- Les conditions sur l'explorateur portent sur des éléments de type `Knowledge` ou `Object`

Voir script `GAME.ocl`

2.6 Transformation en Réseau de Pétri

La transformation vers Petrinet s'est faite via l'outil ATL. Pour contourner un problème d'instanciations non-maitrisées (par exemple, des objets présents dans des conditions se retrouvaient instanciés comme étant des objets réels), nous avons utilisé les called rules et les sections impératives do. Ainsi, le modèle est parcouru manuellement et les éléments sont instanciés de façon maitrisée.

Au début, on a pu transformer tout les éléments du modèles de façon maitrisée, sauf les conditions où on n'a pas pu trouver une transformation généralisables, surtout les conditions où on doit avoir 0 occurence d'un objet ou d'une connaissance, alors on a choisi d'utiliser les read-arcs et les arcs de type inhibitor qui on faciliter le travail, et qu'on a pu généraliser pour tout les modèles GAME.

Voir script `Game2PetriNet.atl`

2.7 Transformations `Accéléo`

2.7.1 Propriétés `LTL`

Les invariants à vérifier sont:

- Un jeu fini n'évolue plus:

$$\square \texttt{ finished} \Rightarrow \texttt{dead}$$

- Un jeu finit toujours par cesser d'évoluer:

$$\square \Diamond \texttt{ dead}$$

- Un jeu n'évoluant plus est terminé: (doit être faux)

$$\square \texttt{ (dead} \Rightarrow \texttt{finished)}$$

- Un jeu fini ne peut pas redémarrer:

$$\square \texttt{ (finished} \Rightarrow \neg \Diamond \texttt{ started)}$$

Voir script `toLTL.mtl`

2.7.2 Transformation vers `Java`

La transformation vers java se fait selon les étapes suivantes :

- Nous construisons tout d'abord l'ensemble des éléments du jeu avec des constructeurs minimaux prenant en paramètre , soit l'identifiant de l'element de jeu si la eClass contient l'attribut name, soit aucun paramètre si la eClass ne contient pas l'attribut. Pour ce faire, on génère les instructions faisant appel à ces constructeurs grâce aux templates qui s'écrivent sous la forme `(ELEMENT)`.
- Nous récupérons les éléments nécessaires du fichier.game à l'aide des querys get(ELEMENT), puis on passe le résultat comme paramètre à nos templates `(init(ELEMENT)`.
- Nous remplissons au fur et à mesure tous les objets et établissons les dépendances entre eux, grâce aux templates du type `fill(ELEMENT)` : ces derniers permettent de de générer les appels Java des setter et des adders pour mettre à jour les attributs des objets.
- Nous lançons la méthode play du controler.

Voir script `toJava.mtl`

Conclusion

Dans ce projet ont été testées nos facultés à construire une infrastructure basée modèle autour d'une problématique concrète. Cette approche a été effectuée en deux axes : une partie conception et une partie réalisation en utilisant les outils du projet Eclipse Modelling pris en main pendant les cours, les TPs et le mini-projet SimplePDL.

Annexe

4.1 GAME.xtext

```
grammar fr.n7.idm.projet.GAME with org.eclipse.xtext.common.Terminals

generate game "http://www.n7.fr/idm/projet/GAME"

Game:
  'game' name=ID 'is' '{'
    explorer = Explorer 'with' 'maxWeight' maxWeight = INT
    ('contains' '{' gameElements += GameElement ( "," gameElements += GameElement)* '}' )?
    'from' startingPlace = [Place] 'to' '(' finishingPlaces += [Place] ( "," finishingPlaces
     += [Place])*')'
   '}'
;

Explorer :
  'explorer' name = ID '{'
    ('has' 'objects' '(' objects+=[Object] ( "," objects+=[Object])* ')' )?
    ('has' 'knowledge' '(' knowledge+=[Knowledge] ( "," knowledge+=[Knowledge])* ')' )?
    'with' 'weight' currentWeight = INT
    'is' 'in' currentPlace = [Place]
  '}' ;


GameElement returns GameElement: Person | Object | Knowledge | Place | Path | Interaction |
    Action | Choice | Explorer| Condition ;

Person:
  'person' name=ID 'is' description=STRING '{'
  'active' isActive = INT ('if' activityCondition = [Condition])?
  'in' place = [Place]
  'interactable' 'with' interaction = [Interaction]
  '}'
;


Description : text = STRING ('depends' 'on' condition = [Condition])?;

Object:'object' name=ID (':' quantity = INT)? 'is' description = Description '{'
    'visible' isVisible = INT ('if' visibilityCondition = [Condition])?
    'size' size = INT
    ('droppable' 'if' dropCondition = [Condition])?
  '}';


Knowledge:
  'knowledge' name=ID 'is' description = Description '{'
    'active' isActive = INT ('if' activityCondition = [Condition])?
  '}'
```

```
;

Place:'place' name=ID 'is' placeType = PlaceType description = Description '{'
    ('knowledge' '(' knowledge+=[Knowledge] ( "," knowledge+=[Knowledge])* ')' )?
    ('objects' '(' objects+=[Object] ( "," objects+=[Object])* ')' )?
    ('persons' '(' persons+=[Person] ( "," persons+=[Person])* ')' )?
    ('paths' '(' paths += [Path] ( "," paths +=[Path])*')')
  '}'
;

enum PlaceType : START = 'start' | FINISH = 'finish' | NORMAL = 'normal';

Path:'path' name=ID 'is' description = Description '{'
    'open' isOpen = INT ('if' openCondition = [Condition])?
    'from' Place1 = [Place] 'to' Place2 = [Place]
    ('transmits' '(' transmittedElements += [GameElement] ("," transmittedElements+=[
    GameElement])* ')' 'if' transmissionCondition = [Condition])?
    ('consumes' '(' consumedElements += [GameElement] ("," consumedElements+=[GameElement])*
     ')' 'if' consumptionCondition = [Condition])?
  '}'
;


// condition on Hamid {
//    possesses moreThan 5 Attempt
//  }

// condition on Hamid {
//    possesses moreThan 5 Attempt or possesses lessThan 1 Victory and possesses exactly 5
    Attempt
//  }


Condition : 'condition' name=ID 'on' explorer = [Explorer] '{'
    firstCondition = SubConditions (nextConditions += NextCondition)*
  '}'
;

SubConditions :'possesses' comparator = Comparator compared = INT gameElement = [GameElement
    ];

NextCondition : logicalOperator = LogicalOperator subCondition = SubConditions;

enum Comparator : EXACTLY = 'EXACTLY' | STRICTLYMORETHAN = 'STRICTLYMORETHAN' |
    STRICTLYLESSTHAN = 'STRICTLYLESSTHAN' | MORETHAN = 'MORETHAN' | LESSTHAN = 'LESSTHAN';

enum LogicalOperator : OR = 'OR' | AND = 'AND';

Action : {Action} 'action' name = ID 'of' choice = [Choice] ('{'
    ('gives' 'objects' '(' objectsToGive+=[Object] ( "," objectsToGive+=[Object])* ')' ('if'
     objectTransmissionCondition = [Condition] )?)?
    ('gives' 'knowledge' '(' knowledge+=[Knowledge] ( "," knowledge+=[Knowledge])* ')'  ('if
    ' knowledgeTransmissionCondition = [Condition])?)?
    ('consumes' 'objects' '(' objectsToConsume+=[Object] ( "," objectsToConsume+=[Object])*
    ')' ('if' objectConsumptionCondition = [Condition])?)?
  '}')?;

Choice : 'choice' name = ID 'of' interaction = [Interaction] 'is' description = STRING '{'
  ('(' actions+=Action ( "," actions+=Action)* ')' )?
  ('depends' 'on' '(' previousActions+=[Action] ( "," previousActions+=[Action])* ')' 'and'
    condition = [Condition])?
  '}'
;

Interaction : 'interaction' name = ID 'is' description = STRING 'with' person = [Person] '{'
```

```
102   ('starts' 'if' startCondition = [Condition])?
103   'choose' 'from' '(' choices+=Choice ( "," choices+=Choice)* ')'
104   ('ends' 'if' endCondition = [Condition])?
105   '}';
```

4.2 GAME.ocl

```
import 'GAME.ecore'

package game

context Game
inv hasCorrectName('Invalid name : ' + self.name) :
  self.name.size() >= 2 and self.name.matches('[A-Za-z_][A-Za-z0-9_]*')
inv isStartingPlaceDefined :
   (not startingPlace.oclIsUndefined()) and startingPlace->asSequence()->size() = 1
inv doFinishingPlacesExist :
  finishingPlaces->asSequence()->size() >= 1

context GameElement
def: Game() : Game = Game.allInstances()->select(g | g.gameElements->includes(self))->
    asSequence()->first()
inv hasCorrectName('Invalid name : ' + self.name) :
  self.name.size() >= 2 and self.name.matches('[A-Za-z_][A-Za-z0-9_]*')

context Object
inv hasValidWeight: self.size >= 0

context Knowledge
inv KnowledgeHaveDifferentNames :
  Knowledge.allInstances()
  ->forAll(k1, k2 | k1 <> k2 implies k1.name <> k2.name)

context Explorer
def: Game() : Game = Game.allInstances()->select(g | g.explorer->includes(self))->asSequence
    ()->first()
inv respectsMaxWeight : currentWeight <= Game().maxWeight

context Path
inv hasCorrectTransmittedElements('Incorrect Transmission Elements Type ' +
    transmittedElements->oclType().toString()):
  transmittedElements->oclIsUndefined() or transmittedElements->forAll(e : GameElement |
              e.oclType() = Object
              or e.oclType() = Knowledge
            )
inv hasCorrectConsumedElements('Incorrect Consumed Elements Type ' + consumedElements->
    oclType().toString()):
  consumedElements->oclIsUndefined() or consumedElements->forAll(e : GameElement |
              e.oclType() = Object
              or e.oclType() = Knowledge
            )
inv hasDifferentExtremities:
  Place1 <> Place2

context Place
inv twoPlacesHaveDifferentNames :
  Place.allInstances()
  ->forAll(p1, p2 | p1 <> p2 implies p1.name <> p2.name)
inv twoPeopleHaveDifferentNames :
  Person.allInstances()
  ->forAll(p1, p2 | p1 <> p2 implies p1.name <> p2.name)

context SubConditions
inv hasCorrectConsumedElements('SubCondition on Incorrect Type or GameElement SubType ' +
    gameElement.name + '::' + gameElement.oclType().toString() ):
  self.gameElement.oclType() = Object or self.gameElement.oclType() = Knowledge

endpackage
```

4.3 Game2PetriNet.atl

```
module Game2PetriNet;
create OUT: petrinet from  IN: game;

-- Obtenir le processus qui contient ce process element.
-- Remarque: Ce helper ne serait pas utile si une rfrence opposite
-- avait t place entre Process et ProcessElement
helper context game!GameElement
def: getGame(): game!Game =
  game!Game.allInstances()
    ->select(p | p.gameElements->includes(self))
    ->asSequence()->first();


-- Traduire un Game en un PetriNet de mme nom
rule Game2PetriNet {
  from g: game!Game
  to pn: petrinet!PetriNet (name <- g.name)
}


-- Traduire un Object en un motif sur le rseau de Petri
rule Object2PetriNet {
  from o: game!Object
  to
    p_object :  petrinet!Places(
              name <- o.name,
              NombreTokens <- o.quantity,
              noeudNet <- o.getGame()
          )
}


-- Traduire un Knowledge en un motif sur le rseau de Petri
rule Knowledge2PetriNet {
  from k: game!Knowledge
  to
    p_knowledge :  petrinet!Places(
              name <- k.name,
              NombreTokens <- 0,
              noeudNet <- k.getGame()
          )
}


-- Traduire un Place en un motif sur le rseau de Petri
rule Place2PetriNet {
  from p: game!Place
  to
    p_place :  petrinet!Places(
              name <- p.name,
              NombreTokens <- if p.placeType = #start then 1 else 0 endif,
              noeudNet <- p.getGame()
          )
}


-- Traduire un Person en un motif sur le rseau de Petri
rule Person2PetriNet {
  from p: game!Person
  to
    t_person :  petrinet!Transitions(
        name <- p.name,
```

```
        noeudNet <- p.getGame()
        )

    , p_interaction : petrinet!Places(
        name <- p.interaction.name,
        NombreTokens <- 0,
        noeudNet <- p.getGame()
        )

    , arc_person_inter : petrinet!Arcs(
        poids_arc <- 1,
        type <- #normal,
        source <- t_person,
        destination <- p_interaction,
        arcNet <- p.getGame()
      )

    , arc_place_person : petrinet!Arcs(
        poids_arc <- 1,
        type <- #normal,
        source <- thisModule.resolveTemp(p.place, 'p_place'),
        destination <- t_person,
        arcNet <- p.getGame()
      )

    do {
      if (p.isActive = 0){
        thisModule.subCondition2PetriNet(p.activityCondition.firstCondition, t_person, p.
    getGame());
        for (subCond in p.activityCondition.nextConditions) {
          thisModule.subCondition2PetriNet(subCond.subCondition, t_person, p.getGame());
        }
      }

      for(c in p.interaction.choices) {
        for(a in c.actions) {
          thisModule.Action2PetriNet(a, p_interaction, p.getGame());
        }
      }
    }
}

-- Traduire un Path en un motif sur le rseau de Petri
rule Path2PetriNet {
  from p: game!Path
  to
    t_path :  petrinet!Transitions(
        name <- p.name,
        noeudNet <- p.getGame()
        )

    ,  arc_fromPlace :  petrinet!Arcs(
        poids_arc <- 1,
        type <- #normal,
        source <- thisModule.resolveTemp(p.Place1,'p_place'),
        destination <- t_path,
        arcNet <- p.getGame()
      )

    ,  arc_toPlace :  petrinet!Arcs(
        poids_arc <- 1,
        type <- #normal,
        source <- t_path,
        destination <- thisModule.resolveTemp(p.Place2, 'p_place'),
        arcNet <- p.getGame()
```

```
127       )
128
129
130     do {
131       if (p.isOpen = 0){
132         thisModule.subCondition2PetriNet(p.openCondition.firstCondition, t_path, p.getGame()
    );
133         for (subCond in p.openCondition.nextConditions) {
134           thisModule.subCondition2PetriNet(subCond.subCondition, t_path, p.getGame());
135         }
136       }
137
138       if (not p.transmittedElements -> isEmpty()){
139         for (o in p.transmittedElements) {
140           thisModule.Path2Objettransmitted(t_path,o,p.getGame());
141         }
142       }
143
144       if (not p.consumedElements -> isEmpty()){
145         for (o in p.consumedElements) {
146           thisModule.Path2ObjetConsumed(t_path,o,p.getGame());
147         }
148       }
149     }
150 }
151
152
153 -- Traduire un Interaction en un motif sur le rseau de Petri
154 rule Action2PetriNet(action: game!Action, p_interaction : petrinet!Places, game : game!Game)
     {
155   to
156     t_action :  petrinet!Transitions(
157         name <- action.name,
158         noeudNet <- game
159         )
160
161     , arc_interaction_action :  petrinet!Arcs(
162         poids_arc <- 1,
163         type <- #normal,
164         source <- p_interaction,
165         destination <- t_action,
166         arcNet <- game
167       )
168
169     , arc_action_place :  petrinet!Arcs(
170         poids_arc <- 1,
171         type <- #normal,
172         source <- t_action,
173         destination <- thisModule.resolveTemp(action.choice.interaction.person.place, '
    p_place'),
174         arcNet <- game
175       )
176
177     do {
178
179       if (not action.objectsToGive -> isEmpty()){
180         for (o in action.objectsToGive) {
181           thisModule.Action2ObjetGive(t_action,game,o);
182         }
183       }
184       if (not action.objectsToConsume -> isEmpty()){
185         for (o in action.objectsToConsume) {
186           thisModule.Action2ObjetConsume(t_action,game,o);
187         }
188       }
```

```
      if (not action.knowledge -> isEmpty()){
        for (k in action.knowledge) {
          thisModule.Action2Knowledge(t_action,game,k);
        }
      }
    }

}

rule Action2ObjetGive (t_action: petrinet!Transitions, game : game!Game, o : game!Object) {
  to
    arc_action_objectGive :  petrinet!Arcs(
        poids_arc <- 1,
        type <- #normal,
        source <- t_action,
        destination <- thisModule.resolveTemp(o, 'p_object'),
        arcNet <- game
      )
}


rule Action2ObjetConsume (t_action: petrinet!Transitions, game : game!Game, o : game!Object)
    {
  to
    arc_objectCons_action :  petrinet!Arcs(
      poids_arc <- 1,
      type <- #normal,
      source <- thisModule.resolveTemp(o, 'p_object'),
      destination <- t_action,
      arcNet <- game
    )
}

rule Action2Knowledge (t_action: petrinet!Transitions, game : game!Game, k : game!Knowledge)
    {
  to
     arc_action_Knowledge :  petrinet!Arcs(
        poids_arc <- 1,
        type <- #normal,
        source <- t_action,
        destination <- thisModule.resolveTemp(k, 'p_knowledge'),
        arcNet <- game
      )
}


rule subCondition2PetriNet (subCondition : game!SubConditions, trans : petriNet!Transitions,
    g : game!Game) {
  to
    arc_contition_element : petrinet!Arcs(
        poids_arc <- if subCondition.compared = 0 then 1 else subCondition.compared endif,
        type <- if subCondition.compared = 0 then #inhibitor else #read_arc endif,
        source <- if subCondition.gameElement -> oclIsTypeOf(game!Object) then
              thisModule.resolveTemp(subCondition.gameElement, 'p_object')
            else
              thisModule.resolveTemp(subCondition.gameElement, 'p_knowledge')
            endif,
        destination <- trans,
        arcNet <- g
      )
}

rule Path2Objettransmitted(t_path : petrinet!Transitions, elem : game!GameElement, g : game!
    Game) {
  to
```

```
250     arc_path_objecttransmitted : petrinet!Arcs(
251         poids_arc <- 1,
252         type <- #normal,
253         source <- t_path,
254         destination <- if elem -> oclIsTypeOf(game!Object) then
255               thisModule.resolveTemp(elem, 'p_object')
256             else
257               thisModule.resolveTemp(elem, 'p_knowledge')
258             endif,
259         arcNet <- g
260       )
261 }
262 
263 
264 rule Path2ObjetConsumed(t_path : petrinet!Transitions, elem : game!GameElement, g : game!
        Game) {
265   to
266     arc_path_objecttransmitted : petrinet!Arcs(
267         poids_arc <- 1,
268         type <- #normal,
269         source <- if elem -> oclIsTypeOf(game!Object) then
270               thisModule.resolveTemp(elem, 'p_object')
271             else
272               thisModule.resolveTemp(elem, 'p_knowledge')
273             endif,
274         destination <- t_path,
275         arcNet <- g
276       )
277 }
```

4.4 toLTL.mtl

```
[comment encoding = UTF-8 /]
[module toLTL('http://www.n7.fr/idm/projet/GAME')]
[template public gameToLTL(aGame : Game)]
[comment @main/]

[file (aGame.name.concat('_GameAssertions.ltl'), false, 'UTF-8')]
 _____________________________________________
|                                             |
                [aGame.name/] Game
|_____________________________________________|


# Game End Assertion
[for (finishingPlace : Place | aGame.finishingPlaces) before (' op finished = ') separator (
    '\\/') after (';')] [finishingPlace.name/][/for]
op started = [aGame.startingPlace.name/];

# A finished game doesn't evolve anymore
['[]'/] (finished => dead);
# A game will always eventually stop evolving
['[] <>'/] dead;
# A game not evolving is a finished one
['[]'/] (dead => finished);
# A game never finishes : Should be false!
['- <>'/] finished;
# A finished game will not restart
['[]'/] (finished => ['- <>'/] started);
# A finished game means it had started beforehand : Should be false!
['[]'/] (finished => started);

[/file]
[/template]
```

4.5 toJAVA.mtl

```
[comment encoding = UTF-8 /]
[module toJAVA('http://www.n7.fr/idm/projet/GAME')]
[template public toJAVA( game : Game )]
[comment @main/]
[file (game.name.concat('.java'), false, 'UTF-8')]

package test;
import Game_Controller.*;
import Display.Display;
import Display.DisplayImpl;
import My_game.*;
import My_game.Object;

public class [game.name.toUpperFirst()/] {
public static void main(String['[]'/] args) {
    /**
     * Init all elements
     */
    [game.initGame()/]
    [game.getGameElements() -> initGameElements()/]
        [game.getActions() -> initActions()/]
        [game.getChoices() -> initChoices()/]
        [game.initExplorateur()/]
    [game.fillConditions()/]
    [game.FillElements()/]

    /**
     * Launch the game
     */
        Display display = new DisplayImpl();
    Controller controleur = new ControllerImpl(display, [game.name.toLowerFirst()/]);
    controleur.play();
  }
}
[/file]
[/template]


[template private initGame(game : Game)]
[game.eClass().name/] [game.name.toLowerFirst()/] = new [game.eClass().name/]("[game.name.
    toLowerFirst()/]");
[/template]

[template private initGameElements(gameElements : OrderedSet(GameElement))]
[for (ge : GameElement | gameElements) ]
[ge.eClass().name/] [ge.name.toLowerFirst()/] = new [ge.eClass().name/]("[ge.name/]");
[/for]
[/template]

[template private initActions(actions : OrderedSet(Action))]
[for (a : Action | actions)]
[a.eClass().name/] [a.name.toLowerFirst()/] = new [a.eClass().name/]("[a.name.toLowerFirst()
    /]");
[/for]
[/template]

[template private initChoices(choices : OrderedSet(Choice))]
[for (c : Choice | choices)]
[c.eClass().name/] [c.name.toLowerFirst()/] = new [c.eClass().name/]("[c.name.toLowerFirst()
    /]");
[/for]
[/template]
```

```
60
61 [template private initExplorateur(game : Game)]
62 [let e : Explorer = game.explorer]
63 [e.eClass().name/] [e.name.toLowerFirst()/] = new [e.eClass().name/]("[e.name/]");
64 [/let]
65 [/template]
66 [template private fillConditions(game : Game)]
67 [for (c : Condition | game.getConditions())]
68 [let fc : SubConditions = c.firstCondition]
69 [if  not fc.oclIsUndefined()]
70 [fc.eClass().name/] subcondition[fc.comparator/][fc.compared/][c.name/][fc.gameElement.name/
    ] = new [fc.eClass().name/]([fc.gameElement.name.toLowerFirst()/], [fc.compared/],
    Comparator.[fc.comparator/]);
71 [c.name/].addSubCond(subcondition[fc.comparator/][fc.compared/][c.name/][fc.gameElement.name
    /]);
72 [/if]
73 [/let]
74
75 [for ( nc : NextCondition | c.nextConditions)]
76 [if not nc.oclIsUndefined()]
77 [let sc : SubConditions = nc.subCondition]
78 [sc.eClass().name/] subcondition[sc.comparator/][sc.compared/][c.name/][sc.gameElement.name/
    ] = new [sc.eClass().name/]([sc.gameElement.name.toLowerFirst()/], [sc.compared/],
    Comparator.[sc.comparator/]);
79 [c.name.toLowerFirst()/].addSubCond(subcondition[sc.comparator/][sc.compared/][c.name/][sc.
    gameElement.name/]);
80 [c.name.toLowerFirst()/].addLogicalOp(LogicalOperator.[nc.logicalOperator/]);
81 [/let]
82 [/if]
83 [/for]
84 [c.name.toLowerFirst()/].setExplorer([game.explorer.name.toLowerFirst()/]);
85 [/for]
86 [/template]
87
88
89
90 [template private fillObjects(objects : OrderedSet(Object))]
91 [for( o : Object | objects)]
92 [o.name.toLowerFirst()/].setCount([o.quantity/]);
93 [o.name.toLowerFirst()/].setSize([o.size/]);
94 [/for]
95 [/template]
96
97
98 [template private fillKnowledge(knowledge : OrderedSet(Knowledge))]
99 [for( n : Knowledge | knowledge)]
100 [n.name.toLowerFirst()/].setCount(1);
101 [/for]
102 [/template]
103
104
105 [template private fillExplorateur(explorer : Explorer, game : Game)]
106 [explorer.name.toLowerFirst()/].setCurrentPlace([game.startingPlace.name.toLowerFirst()/]);
107 [explorer.name.toLowerFirst()/].setCurrentWeight([game.explorer.currentWeight/]);
108 [for( o : Object | explorer.objects)]
109 [if not o.oclIsUndefined()]
110 [explorer.name.toLowerFirst()/].addObject([o.name.toLowerFirst()/]);
111 [/if]
112 [/for]
113 [for( n : Knowledge | explorer.knowledge)]
114 [if not n.oclIsUndefined()]
115 [explorer.name/].addKnowledge([n.name.toLowerFirst()/]);
116 [/if]
117 [/for]
118 [/template]
```

```
119
120
121 [template private fillPersons(persons : OrderedSet(Person))]
122 [for( p : Person | persons)]
123 [p.name.toLowerFirst()/].setDescription("[p.description/]");
124 [p.name.toLowerFirst()/].setActivityCondition([p.activityCondition.name.toLowerFirst()/]);
125 [p.name.toLowerFirst()/].setInteraction([p.interaction.name.toLowerFirst()/]);
126 [p.name.toLowerFirst()/].setIsActive([tobolean(p.isActive)/]);
127 [/for]
128 [/template]
129
130 [template private fillPaths(paths : OrderedSet(Path))]
131 [for( p : Path | paths)]
132 [p.name.toLowerFirst()/].setPlace1([p.Place1.name.toLowerFirst()/]);
133 [p.name.toLowerFirst()/].setPlace2([p.Place2.name.toLowerFirst()/]);
134 [if not p.openCondition.oclIsUndefined()]
135 [p.name.toLowerFirst()/].setOpenCondition([p.openCondition.name.toLowerFirst()/]);
136 [p.name.toLowerFirst()/].setOpen([tobolean(p.isOpen)/]);
137 [/if]
138 [/for]
139 [/template]
140
141
142 [template private fillPlaces(places : OrderedSet(Place))]
143 [for (p : Place | places)]
144 [for(prs : Person| p.persons)]
145 [p.name.toLowerFirst()/].addPersons([prs.name.toLowerFirst()/]);
146 [/for]
147
148 [for(pa : Path| p.paths)]
149 [p.name.toLowerFirst()/].addPaths([pa.name.toLowerFirst()/]);
150 [/for]
151
152 [for(o : Object| p.objects)]
153 [p.name.toLowerFirst()/].addObject([o.name.toLowerFirst()/]);
154 [/for]
155
156 [for(n : Knowledge| p.knowledge)]
157 [p.name.toLowerFirst()/].addKnowledge([n.name.toLowerFirst()/]);
158 [/for]
159 [/for]
160 [/template]
161
162
163 [template private fillActions(actions : OrderedSet(Action))]
164 [for (a : Action | actions)]
165 [for(o : Object | a.objectsToConsume)]
166 [a.name.toLowerFirst()/].addObjectsToConsume([o.name.toLowerFirst()/]);
167 [/for]
168
169 [for(o : Object | a.objectsToGive)]
170 [a.name.toLowerFirst()/].addObjectsTogive([o.name.toLowerFirst()/]);
171 [/for]
172
173 [for(n : Knowledge| a.knowledge)]
174 [a.name.toLowerFirst()/].addKnowledge([n.name.toLowerFirst()/]);
175 [/for]
176 [/for]
177 [/template]
178
179
180 [template private fillChoice( choices : OrderedSet(Choice))]
181 [for (ch : Choice | choices)]
182  [if not ch.description.oclIsUndefined()]
183 [ch.name.toLowerFirst()/].setDescription("[ch.description/]");
```

```
[/if]
[for(a : Action | ch.actions)]
[ch.name.toLowerFirst()/].addActions([a.name.toLowerFirst()/]);
[/for]
[/for]
[/template]

[template private fillInteraction( interactions : OrderedSet(Interaction))]
[for (it : Interaction | interactions)]
[if not it.startCondition.oclIsUndefined()]
[it.name.toLowerFirst()/].setStartCondition([it.startCondition.name.toLowerFirst()/]);
[/if]
[if not it.endCondition.oclIsUndefined()]
[it.name.toLowerFirst()/].setEndCondition([it.endCondition.name.toLowerFirst()/]);
[/if]
[for(ch : Choice | it.choices)]
[it.name.toLowerFirst()/].addChoices([ch.name.toLowerFirst()/]);
[/for]
[/for]
[/template]

[template private fillGame(game : Game)]
[game.name.toLowerFirst()/].setExplorer([game.explorer.name.toLowerFirst()/]);
[game.name.toLowerFirst()/].setStartingPlace([game.startingPlace.name.toLowerFirst()/]);
[game.name.toLowerFirst()/].setMaxWeight([game.maxWeight/]);
[for(fp : Place | game.finishingPlaces)]
[game.name.toLowerFirst()/].addFinishingPlace([fp.name.toLowerFirst()/]);
[/for]
[/template]



[template private FillElements(game : Game)]
[game.getPlaces() -> fillPlaces()/];
[game.getPaths() -> fillPaths()/];
[game.getObjects() -> fillObjects()/];
[game.getKnowledges() -> fillKnowledge()/];
[game.getPersons() -> fillPersons()/];
[game.getActions() -> fillActions()/];
[game.getChoices() -> fillChoice()/];
[game.getInteractions() -> fillInteraction()/];
[game.fillGame()/];
[game.getExplorateur().fillExplorateur(game)/];
[/template]



[query private getConditions(game : Game) : OrderedSet(Condition) = game.gameElements->
    select(c  | c.oclIsTypeOf(Condition)) -> collect(c | c.oclAsType(Condition)) ->
    asOrderedSet() /]
[query private getPlaces(game : Game) : OrderedSet(Place) = game.gameElements-> select(p  |
    p.oclIsTypeOf(Place)) -> collect(p | p.oclAsType(Place)) -> asOrderedSet() /]
[query private getPaths(game : Game) : OrderedSet(Path) =game.gameElements-> select(p  | p.
    oclIsTypeOf(Path)) -> collect(p | p.oclAsType(Path)) -> asOrderedSet() /]
[query private getObjects(game : Game) : OrderedSet(Object) = game.gameElements -> select(g
     | g.oclIsTypeOf(Object)) -> collect(g | g.oclAsType(Object)) -> asOrderedSet() /]
[query private getKnowledges(game : Game) : OrderedSet(Knowledge) = game.gameElements->
    select(n  | n.oclIsTypeOf(Knowledge)) -> collect(n | n.oclAsType(Knowledge)) ->
    asOrderedSet() /]
[query private getPersons(game : Game) : OrderedSet(Person) = game.gameElements-> select(p
    | p.oclIsTypeOf(Person)) -> collect(p | p.oclAsType(Person)) -> asOrderedSet() /]
[query private getActions(game : Game) : OrderedSet(Action) = game.getChoices()->collect(c |
     c.actions)-> select(a  | a.oclIsTypeOf(Action)) -> collect(a | a.oclAsType(Action)) ->
    asOrderedSet() /]
[query private getChoices(game : Game) : OrderedSet(Choice) = game.getInteractions()->
```

```
    collect(i |i.choices)-> select(c  | c.oclIsTypeOf(Choice)) -> collect(c | c.oclAsType(
    Choice)) -> asOrderedSet() /]
239 [query private getInteractions(game : Game) : OrderedSet(Interaction) = game.gameElements->
    select(i  | i.oclIsTypeOf(Interaction)) -> collect(i | i.oclAsType(Interaction)) ->
    asOrderedSet() /]
240 [query private getExplorateur(game : Game) : Explorer = game.explorer /]
241 [query private getGameElements(game : Game) : OrderedSet(GameElement) = gameElements ->
    asOrderedSet()/]
242 [query private tobolean(n : Integer) : Boolean = if (n = 0) then true else false endif /]
```