Appendix A – Sample rules

External rules file generated from the syntactic editor used as input to the application

- **data/F02133221.ilr**

```
//=========================================== -*- Java -*- ========================
// Prototype for the student course advice system:  Sample rules that should be maintained
//=============================================================================
import course.advice.*;

/**Degree CS determines:  Course credits = 225;
                  max 110 credits at 100 level;
                  min 56 at 300/400 level;
                  max 22 credits from other departments;
                  max 56 credits per year  */
rule DegreeComputerScienceDetermines
{
   priority = maximum;
   when
     {
 ?x:CurrDegree(currDegreeCode.equals("02130001");currStudyProgram.equals("02133221"));
     }
   then
     {
  assert CourseDetails(225,70,56,22,56);
     }
};
/**Faculty Requirement WTW114 */
rule FacultyRequirementWTW114
{
  priority = maximum;
   when
     {
        not
StudSubj(subjName.equals("WTW114");(passLevel==PASSED)||(passLevel==REGISTERED)
            ||(passLevel==OPTIONAL));
     }
   then
     {
        assert StudSubj("WTW114",LEVEL100,11,REGISTERED,BOTH_SEMESTERS);
```

```
    }
};
/**FacultyForseRequirement WTW114 */
rule FacultyRequirementForseWTW114
{
  priority = maximum;
   when
    {
       not
StudSubj(subjName.equals("WTW114");(passLevel==PASSED)||(passLevel==REGISTERED));
       ?x: StudSubj(subjName.equals("WTW114");(passLevel==OPTIONAL));
    }
   then
    {
       retract ?x;
       assert StudSubj("WTW114",LEVEL100,11,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement COS110 */
rule ITRequirementCOS110
{
 priority = 10000;
   when
    {
       not
StudSubj(subjName.equals("COS110");(passLevel==PASSED)||(passLevel==REGISTERED)
       ||(passLevel==OPTIONAL));
    }
   then
    {
 assert StudSubj("COS110",LEVEL100,11,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement Forse COS110 */
rule ITRequirementForseCOS110
{
 priority = 10000;
   when
    {
```

```
        not
StudSubj(subjName.equals("COS110");(passLevel==PASSED)||(passLevel==REGISTERED));
        ?x: StudSubj(subjName.equals("COS110");(passLevel==OPTIONAL));
    }
  then
    {
        retract ?x;
 assert StudSubj("COS110",LEVEL100,11,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Optional COS120 */
rule OptionalCOS120
{
 priority = 1000;
  when
    {
        not
StudSubj(subjName.equals("COS120");(passLevel==PASSED)||(passLevel==REGISTERED)||(passLe
vel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS120",LEVEL100,11,OPTIONAL,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement COS212:  Prerequisite COS110 passed */
rule ITRequirementCOS212
{
 priority = 900;
  when
    {
        ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
        not
StudSubj(subjName.equals("COS212");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS212",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
```

```
};
/**Information Technology Requirement COS213:  Prerequisite COS110  passed*/
rule ITRequirementCOS213
{
  priority = 900;
   when
    {
       ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
       not
StudSubj(subjName.equals("COS213");(passLevel==PASSED)||(passLevel==REGISTERED)||
              (passLevel==OPTIONAL));
    }
   then
    {
 assert StudSubj("COS213",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement COS221:  Prerequisite COS110 passed */
rule ITRequirementCOS221
{
  priority = 900;
   when
    {
       ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
       not
StudSubj(subjName.equals("COS221");(passLevel==PASSED)||(passLevel==REGISTERED)||
              (passLevel==OPTIONAL));
    }
   then
    {
 assert StudSubj("COS221",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement COS222:  Prerequisite COS110 passed */
rule ITRequirementCOS222
{
  priority = 900;
   when
    {
       ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
```

```
        not
StudSubj(subjName.equals("COS222");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS222",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement COS283:  Prerequisite COS110 passed */
rule ITRequirementCOS283
{
  priority = 900;
   when
    {
       ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
       not
StudSubj(subjName.equals("COS283");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS283",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Information Technology Requirement COS284:  Prerequisite COS110  passed */
rule ITRequirementCOS284
{
  priority = 900;
  when
    {
       ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
       not
StudSubj(subjName.equals("COS284");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS284",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
```

```
};
/**Information Technology Requirement COS301 : Prerequisite COS120 passed*/
rule ITRequirementCOS301
{
   priority = 500;
   when
     {
        ?x: StudSubj(subjName.equals("COS120");(passLevel==PASSED));
        not
StudSubj(subjName.equals("COS301");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
     }
   then
     {
  assert StudSubj("COS301",LEVEL300,7,REGISTERED,BOTH_SEMESTERS);
     }
};
/**Compulsory Course PHY171:  Prerequisite WTW114  Passed or Registered*/
rule CompulsoryCoursePHY171
{
  priority = high;
   when
     {
        ?x:
StudSubj(subjName.equals("WTW114");(passLevel==PASSED)||(passLevel==REGISTERED));
        not
StudSubj(subjName.equals("PHY171");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
     }
   then
     {
  assert StudSubj("PHY171",LEVEL100,22,REGISTERED,BOTH_SEMESTERS);
     }
};
/**Compulsory Course ERS220:  No Prerequisites*/
rule CompulsoryCourseERS220
{
  priority = 900;
   when
     {
```

```
        not
StudSubj(subjName.equals("ERS220");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("ERS220",LEVEL200,8,REGISTERED,BOTH_SEMESTERS);
    }
};
/**Compulsory Course ERS320:  Prerequisite ERS220  Passed*/
rule CompulsoryCourseERS320
{
  priority = 500;
  when
    {
      ?x: StudSubj(subjName.equals("ERS220");(passLevel==PASSED));
      not
StudSubj(subjName.equals("ERS320");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("ERS320",LEVEL300,8,REGISTERED,BOTH_SEMESTERS);
    }
};
/**AtLeast4Courses COS314:  One of at least 4 - Prerequisite COS110  Passed*/
rule AtLeast4CoursesCOS314
{
  priority = 500;
  when
    {
      ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
      not
StudSubj(subjName.equals("COS314");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS314",LEVEL300,7,REGISTERED,BOTH_SEMESTERS);
    }
```

```
};
/**AtLeast4Courses COS324:  One of at least 4 Prerequisite COS222*/
rule AtLeast4CoursesCOS324
{
   priority = 500;
   when
    {
       ?x: StudSubj(subjName.equals("COS222");(passLevel==PASSED));
       not
StudSubj(subjName.equals("COS324");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
   then
    {
 assert StudSubj("COS324",LEVEL300,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**AtLeast4Courses COS332:  One of at least 4 - Prerequisite COS283  Passed */
rule AtLeast4CoursesCOS332
{
   priority = 500;
   when
    {
       ?x: StudSubj(subjName.equals("COS283");(passLevel==PASSED));
       not
StudSubj(subjName.equals("COS332");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
   then
    {
 assert StudSubj("COS332",LEVEL200,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**AtLeast4Courses COS333:  One of at least 4 - Prerequisite COS110  Passed */
rule AtLeast4CoursesCOS333
{
   priority = 500;
   when
    {
       ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
```

```
        not
StudSubj(subjName.equals("COS333");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS333",LEVEL300,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**AtLeast4Courses COS341:  One of at least 4 - Prerequisite COS212  Passed */
rule AtLeast4CoursesCOS341
{
  priority = 500;
  when
    {
      ?x: StudSubj(subjName.equals("COS212");(passLevel==PASSED));
      not
StudSubj(subjName.equals("COS341");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS341",LEVEL300,7,REGISTERED,BOTH_SEMESTERS);
    }
};
/**AtLeast4Courses COS343:  One of at least 4 - Prerequisite COS110  Passed */
rule AtLeast4CoursesCOS343
{
  priority = 500;
  when
    {
      ?x: StudSubj(subjName.equals("COS110");(passLevel==PASSED));
      not
StudSubj(subjName.equals("COS343");(passLevel==PASSED)||(passLevel==REGISTERED)||
                (passLevel==OPTIONAL));
    }
  then
    {
 assert StudSubj("COS343",LEVEL300,7,REGISTERED,BOTH_SEMESTERS);
    }
```

```
};
/**CourseApproved1:  Head of Department/lecturer approval*/
rule CourseApproved1
{
   priority = 1000;
   when
     {
        ?a: StudSubj(?x:
subjName;(passLevel==TD)||(passLevel==TDH);?y:level;?z:noOfUnits;?s:semester);
        not StudSubj(subjName.equals(?x);(passLevel==PASSED)||(passLevel==REGISTERED)||
              (passLevel==OPTIONAL));
     }
   then
     {
        retract ?a;
  assert StudSubj(?x,?y,?z,REGISTERED,?s);
     }
};
rule OnlyOnePassLevel1
{
   priority = low;
   when
     {
?x: StudSubj(?y: subjName; passLevel==PASSED);
       ?z: StudSubj(subjName.equals(?y);(passLevel==REGISTERED)||(passLevel==OPTIONAL));
       }
   then
     {
       retract ?z;
     }
};
rule OnlyOnePassLevel2
{
   priority = low;
   when
     {
?x: StudSubj(?y: subjName; passLevel==REGISTERED);
       ?z: StudSubj(subjName.equals(?y);passLevel==OPTIONAL);
     }
   then
```

```
    {
     retract ?z;
    }
};
/**
Max 110 level 100 credits allowed */
rule FacultyRequirementTooManyLevel100Credits
{
  priority = maximum -5;
   when
    {
      CourseDetails(?a:level100Max);
      ?c: collect ( new TestLevelSubj(LEVEL100))

StudSubj(level==LEVEL100;(passLevel==PASSED)||(passLevel==REGISTERED);?y:noOfUnits)
      where (totalNoOfUnits() > ?a);
    }
   then
    {

     assert(?c);
    }
};
/**
Max 56 Credits per year allowed */
rule FacultyRequirementTooManySELECTEDSubjects
{
  priority = maximum -10;
   when
    {
      CourseDetails(?a:creditsPerYear);
      ?c: collect ( new TestLevelSubj(REGISTERED))
         StudSubj(?x:level;(passLevel==REGISTERED);?y:noOfUnits)
      where (totalCreditsPerYear() > ?a);
    }
   then
    {
     assert(?c);
    }
};
```

```
/*** Credits SELECTED reduced - one high level course moved to OPTIONAL */
rule TooManyCreditsPerYearMovedHighestLevelToOptional
{
 priority = maximum -10;
  when
   {
     ?g: StudSubj(?b: subjName;?c: level; ?d: noOfUnits;passLevel==REGISTERED; ?e: semester; ?f:
itemNo);
     not StudSubj(passLevel==REGISTERED; ?c < level);
     CourseDetails(?a: creditsPerYear);
     ?x: TestLevelSubj(total > ?a;collectionType==TestLevelSubj.REGISTERED);
   }
  then
   {
     retract ?g;
     assert StudSubj(?b,?c,?d,OPTIONAL,?e);
     update refresh ?x;
   }
};
/*** Level 100 credits selected reduced - one moved to OPTIONAL */
rule TooManyLevel100CreditsMoveLastOneAdded
{
 priority = maximum - 5;
  when
   {
     ?g: StudSubj(?b: subjName;?c:level;level==LEVEL100; ?d:
noOfUnits;passLevel==REGISTERED; ?e: semester; ?f: itemNo);
     not StudSubj(passLevel==REGISTERED;level==LEVEL100; ?f < itemNo);
     CourseDetails(?a:level100Max);
     ?x: TestLevelSubj(total > ?a;collectionType==TestLevelSubj.LEVEL100);
   }
  then
   {
     retract ?g;
     assert StudSubj(?b,?c,?d,OPTIONAL,?e);
     update refresh ?x;
   }
};
```

The syntactical mode of the syntactic editor: rules from data/F02133221.ilr

Rule DegreeDetermines

WHEN
there is a **CurrDegree** called ?x
such that currDegreeCode.equals("02130001")
and currStudyProgram.equals("02133221")
THEN
retract ?x
assert CourseDetails (225, 70, 56, 22, 56)

Rule FacultyRequirement

WHEN
there is no **StudSubj**
such that subjName.equals("WTW114")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED)
there is a **StudSubj** called ?x
such that subjName.equals("WTW114")
and passLevel = StudSubj.OPTIONAL
THEN
retract ?x
assert StudSubj ("WTW114", StudSubj.LEVEL100, 11, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement1

WHEN
there is no **StudSubj**
such that subjName.equals("COS110")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED)
there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.OPTIONAL
THEN
retract ?x
assert StudSubj ("COS110", StudSubj.LEVEL100, 11, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule Optional1

WHEN
there is no **StudSubj**
such that subjName.equals("COS120")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)
THEN
assert StudSubj ("COS120", StudSubj.LEVEL100, 11, StudSubj.OPTIONAL, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement2

WHEN
there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS212")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)
THEN
assert StudSubj ("COS212", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement3

WHEN
there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS213")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)
THEN
assert StudSubj ("COS213", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement4

WHEN
there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS221")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)
THEN
assert StudSubj ("COS221", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement5

WHEN
there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS222")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)
THEN
assert StudSubj ("COS222", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement6

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS283")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS283", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement7

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS284")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS284", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule ITRequirement8

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS120")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS301")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS301", StudSubj.LEVEL300, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule CompulsoryCourse1

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("WTW114")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED)
there is no **StudSubj**
such that subjName.equals("PHY171")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED

```
                or passLevel = StudSubj.OPTIONAL )
THEN
    assert StudSubj ( "PHY171", StudSubj.LEVEL100, 22, StudSubj.REGISTERED,
StudSubj.BOTH_SEMESTERS )
```

Rule CompulsoryCourse2

```
WHEN
    there is no StudSubj
        such that subjName.equals("ERS220")
            and ( passLevel = StudSubj.PASSED
                or passLevel = StudSubj.REGISTERED
                or passLevel = StudSubj.OPTIONAL )
THEN
    assert StudSubj ( "ERS220", StudSubj.LEVEL200, 8, StudSubj.REGISTERED,
StudSubj.BOTH_SEMESTERS )
```

Rule CompulsoryCourse3

```
WHEN
    there is a StudSubj called ?x
        such that subjName.equals("ERS220")
            and passLevel = StudSubj.PASSED
    there is no StudSubj
        such that subjName.equals("ERS320")
            and ( passLevel = StudSubj.PASSED
                or passLevel = StudSubj.REGISTERED
                or passLevel = StudSubj.OPTIONAL )
THEN
    assert StudSubj ( "ERS320", StudSubj.LEVEL300, 8, StudSubj.REGISTERED,
StudSubj.BOTH_SEMESTERS )
```

Rule AtLeast4Courses1

```
WHEN
    there is a StudSubj called ?x
        such that subjName.equals("COS110")
            and passLevel = StudSubj.PASSED
    there is no StudSubj
        such that subjName.equals("COS314")
            and ( passLevel = StudSubj.PASSED
                or passLevel = StudSubj.REGISTERED
                or passLevel = StudSubj.OPTIONAL )
THEN
    assert StudSubj ( "COS314", StudSubj.LEVEL300, 7, StudSubj.REGISTERED,
StudSubj.BOTH_SEMESTERS )
```

Rule AtLeast4Courses3

```
WHEN
    there is a StudSubj called ?x
        such that subjName.equals("COS222")
            and passLevel = StudSubj.PASSED
    there is no StudSubj
        such that subjName.equals("COS324")
            and ( passLevel = StudSubj.PASSED
                or passLevel = StudSubj.REGISTERED
```

or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS324", StudSubj.LEVEL300, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule AtLeast4Courses5

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS283")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS332")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS332", StudSubj.LEVEL200, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule AtLeast4Courses7

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS333")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS333", StudSubj.LEVEL300, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule AtLeast4Courses9

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS212")
and passLevel = StudSubj.PASSED
there is no **StudSubj**
such that subjName.equals("COS341")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS341", StudSubj.LEVEL300, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule AtLeast4Courses11

WHEN

there is a **StudSubj** called ?x
such that subjName.equals("COS110")
and passLevel = StudSubj.PASSED
there is no **StudSubj**

such that subjName.equals("COS343")
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

assert StudSubj ("COS343", StudSubj.LEVEL300, 7, StudSubj.REGISTERED, StudSubj.BOTH_SEMESTERS)

Rule CourseApproved1

WHEN

there is a **StudSubj** called ?a
where subjName is called ?x
and level is called ?y
and noOfUnits is called ?z
and semester is called ?s
such that (passLevel = StudSubj.TD
or passLevel = StudSubj.TDH)
there is no **StudSubj**
such that subjName.equals(?x)
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

retract ?a
assert StudSubj (?x, ?y, ?z, StudSubj.REGISTERED, ?s)

Rule OnlyOnePassLevel1

WHEN

there is a **StudSubj** called ?x
where subjName is called ?y
such that passLevel = StudSubj.PASSED
there is a **StudSubj** called ?z
such that subjName.equals(?y)
and (passLevel = StudSubj.REGISTERED
or passLevel = StudSubj.OPTIONAL)

THEN

retract ?z

Rule OnlyOnePassLevel2

WHEN

there is a **StudSubj** called ?x
where subjName is called ?y
such that passLevel = StudSubj.REGISTERED
there is a **StudSubj** called ?z
such that subjName.equals(?y)
and passLevel = StudSubj.OPTIONAL

THEN

retract ?z

Rule FacultyRequirement1

WHEN

there is a **CourseDetails**
where level100Max is called ?a

the collection of **StudSubj** called ?c stored in (new TestLevelSubj(TestLevelSubj.LEVEL100))
where noOfUnits is called ?y
such that level = StudSubj.LEVEL100
and (passLevel = StudSubj.PASSED
or passLevel = StudSubj.REGISTERED)
verifies totalNoOfUnits() > ?a

THEN

execute
so that System.out.println((" totalNoOfUnits = " + ?c.totalNoOfUnits()) + " > ") + ?a)
and ?context.assert(?c)

Rule FacultyRequirement2

WHEN

there is a **CourseDetails**
where creditsPerSemester is called ?a
the collection of **StudSubj** called ?c stored in (new TestLevelSubj(TestLevelSubj.REGISTERED))
where level is called ?x
and noOfUnits is called ?y
such that passLevel = StudSubj.REGISTERED
verifies totalCreditsPerYear() > ?a

THEN

execute
so that ?context.assert(?c)
and System.out.println((" totalCreditsPerYear = " + ?c.totalCreditsPerYear()) + " > ") + ?a)

Rule TooManyCreditsPerYear

WHEN

there is a **StudSubj** called ?g
where subjName is called ?b
and level is called ?c
and noOfUnits is called ?d
and semester is called ?e
and itemNo is called ?f
such that passLevel = StudSubj.REGISTERED
there is no **StudSubj**
such that passLevel = StudSubj.REGISTERED
and ?c < level
there is a **CourseDetails**
where creditsPerSemester is called ?a
there is a **TestLevelSubj** called ?x
such that total > ?a
and collectionType = TestLevelSubj.REGISTERED

THEN

retract ?g
assert StudSubj (?b, ?c, ?d, StudSubj.OPTIONAL, ?e)
update refresh ?x

Rule TooManyLevel100Credits

WHEN

there is a **StudSubj** called ?g
where subjName is called ?b
and level is called ?c
and noOfUnits is called ?d

```
            and semester is called ?e
            and itemNo is called ?f
        such that level = StudSubj.LEVEL100
            and passLevel = StudSubj.REGISTERED
    there is no StudSubj
        such that passLevel = StudSubj.REGISTERED
            and level = StudSubj.LEVEL100
            and ?f < itemNo
    there is a CourseDetails
        where level100Max is called ?a
    there is a TestLevelSubj called ?x
        such that total > ?a
            and collectionType = TestLevelSubj.LEVEL100
THEN
    retract ?
    assert StudSubj ( ?b, ?c, ?d, StudSubj.OPTIONAL, ?e )
    update refresh ?x
```