**Observaciones:**

**Constraints**:

**1. Un profesor no puede ser alumno de la asignatura que imparte**

Context Persona inv notProfesorAndAlumno:

Persona.allInstances -> forAll(p | p.asignaturaProfesor and p.asignaturaAlumno -> isEmpty() )

**2. La nota debe estar entre 0 y 10.**

Context Matricula inv rangoNota:

1) Matricula.allInstances -> forAll(m | m.nota not null implies m.nota >= 0 and m.nota <= 10)

2) nota not null implies self.nota >= 0 and self.nota <= 10

**3. Un alumno solo puede aprobar una asignatura una vez, no pudiendo matricularse de asignaturas que ya ha aprobado en cursos anteriores.**

Context Matricula inv asignaturaAprobada:

Matricula.allInstances -> forAll(m1,m2 | (m1.alumno = m2.alumno) and (m1.asignatura = m2. asignatura) and (m1.fechaMatriculacion < m2.fechaMatriculacion) implies m1.nota < 5)

**6. Asignatura.titulación > 0 (no ponemos composición porque una asignatura puede estar en varias titulaciones)**

Context Asignatura inv alMenosUnaTitulacion:

a.titulacion -> size() > 0

**8. Finalmente, para poder matricularse de una asignatura de posgrado, el alumno ha de estar en posesión de un título de grado (de esa universidad o de otra) -> Usar kindof o typeof para saber si los títulos que tiene son grados o postgrados. (Es decir títulos de un alumno > 0)**

Context Matricula inv GradoParaPostgrado:

Matricula.allInstances -> forAll(m | m.asignatura.titulacion.oclIsTypeOf(Posgrado) implies m.alumno.titulo ->select(titulo.titulacion.oclIsTypeOf(Grado)) -> size() > 0)

Matricula.allInstances -> forAll(m | m.asignatura.titulacion.oclIsTypeOf(Posgrado) implies m.alumno.titulo -> exists(titulo.titulacion.oclIsTypeOf(Grado)))

**10. Asignaturas con mismo id deben ser de distintas titulaciones**

Context Asignatura inv mismoID:

Asignatura.allInstances -> forAll(a1,a2 | a1.id = a2.id => a1.titulacion <> a2.titulacion)

**12. A lo largo de su vida, un alumno puede obtener tantos títulos como titulaciones oferta la**

(NO SE DA NUNCA EXPLICARLO)

Context Persona inv cantTitulos:

Universidad.allInstances -> forAll(u | u.titulacion -> size() >=

(Persona.allInstances -> select(p | p.titulo.titulo.nombreUniversidad

-> includes(u.nombre))) -> size())

**13. Para obtener un título es preciso tener aprobados al menos 240 créditos de las asignaturas que componen una titulación de grado, o los 60 que componen una de posgrado -> restricción comprobar si sumamos 240 créditos de asignaturas de la titulación correspondiente al título que asociamos a la persona (con un forall de las asignaturas cuya titulación.nombre sea = titulo.titulacion.nombre)**

Context Persona inv tituloValido:

Persona.allInstances -> forAll (p | p.titulo ->

forAll(t | if t.titulo.titulacion.oclIsTypeOf(Grado) then

(240 <= (t.estudiante.asignaturaMatricula ->

select(m|(m.asignatura.titulacion =t.titulo.titulacion) and

not m.fechaAprobada.oclIsUndefined()).asignatura.creditos -> sum()

)

)

else

(60 <= (t.estudiante.asignaturaMatricula ->

select(m|(m.asignatura.titulacion = t.titulo.titulacion) and

not m.fechaAprobada.oclIsUndefined()).asignatura.creditos -> sum()

)

)

endif

)

)

**14. En “Titulación” el atributo “numCursos” tiene que ser mayor que 0.**

Context Titulacion inv rangoNumCursos:

Titulacion.AllInstances -> forAll(t |t.numCursos > 0)

**15. En “Asignatura” el atributo “créditos” tiene que ser mayor que 0.**

Context Asignatura inv notNullCreditos

Asignatura.allInstances -> forAll(a | a.creditos > 0)

**17**. **ID asignatura entre 100 y numCursos\*100**

Context Asignatura inv rangoCodigo

(self.codigo >= 100 and self.codigo < (self.titulacion.numCursos + 1) \* 100)

**18 FechaMatriculacion < FechaAprobada en Matriculacion**

Context Matricula inv fechas:

Self.FechaMatriculacion < self.fechaAprobada

**\*Clase de asociación Alumno entre Persona y Asignatura, guardamos nota**. (Si es null estará impartiendo la asignatura en este momento)´

**\*Asociación Profesor entre Persona y Asignatura**

**\*Clase título composición de titulación, nombre universidad atributo derivado por titulación, esta asociado con Persona**