www.miso.es

TOWARDS AUTOMATING THE **ANALYSIS OF INTEGRITY CONSTRAINTS** IN MULTI-LEVEL MODELS

Esther Guerra and Juan de Lara

{esther.guerra, juan.delara}@uam.es

Modelling and Software Engineering Group (miso)

Computer Science Department

Universidad Autónoma de Madrid

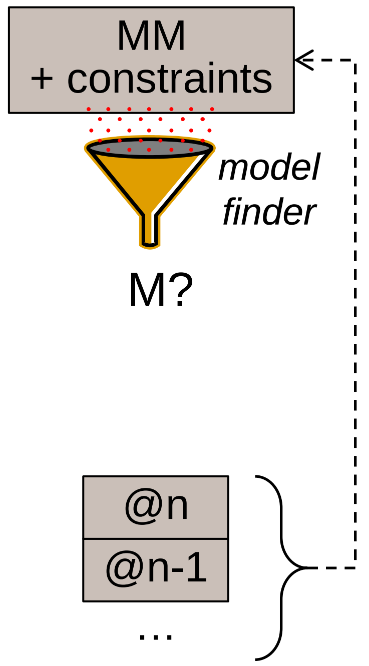
GOAL

- analysis of correctness properties in multi-level models
- starting with a basic correctness property: satisfiability

  "given a meta-model with (ocl) integrity constraints,
  is there a valid model that satisfies all constraints?"

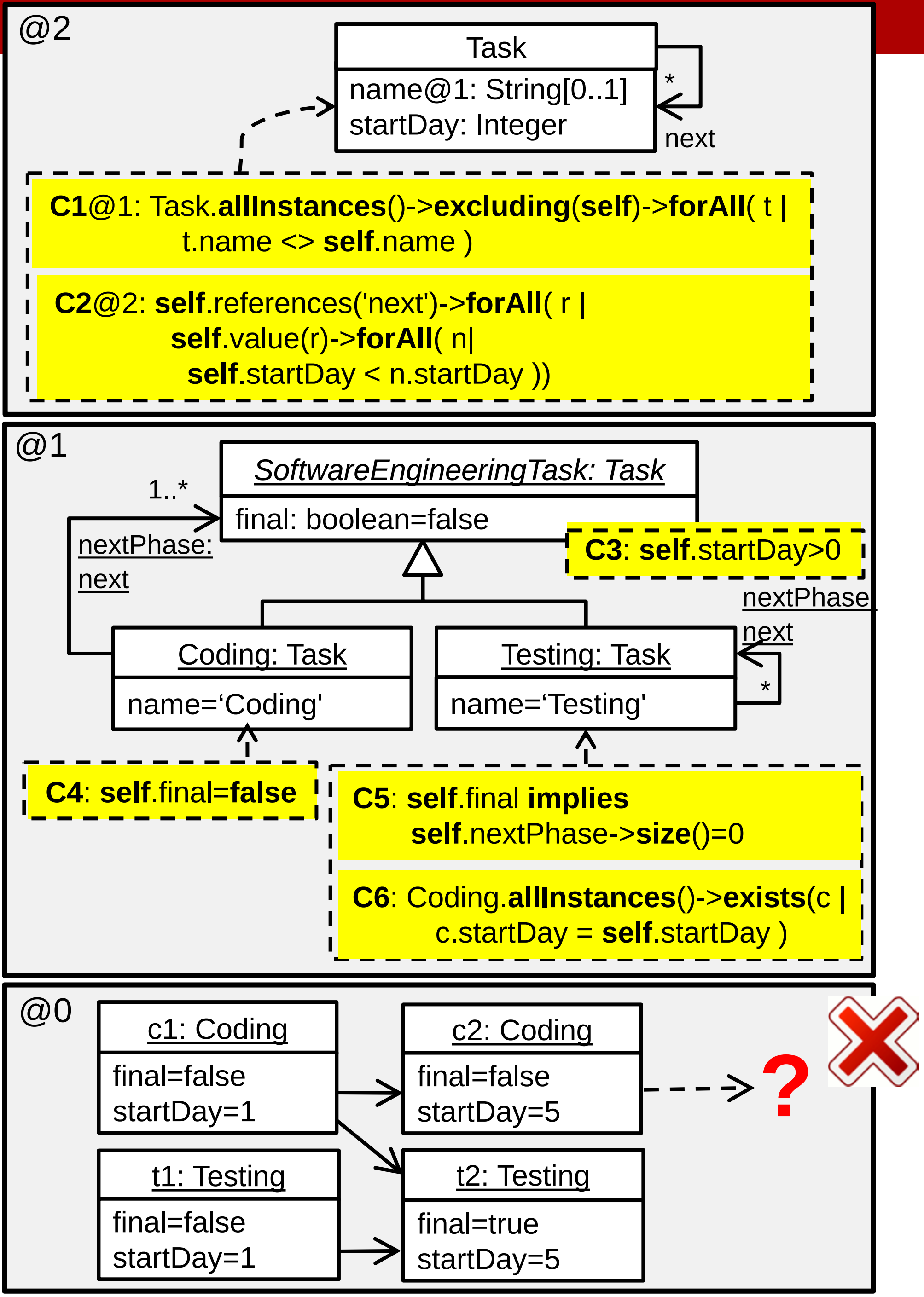
2-LEVELS

- constraints defined in MM, and evaluated in M
- analysis by means of *off-the-shelf* model finders



MULTI-LEVEL MODELLING

- constraints defined at any meta-level
- constraints evaluated *n* meta-levels below
- contribution: how to use standard model finders to analyse multi-level models



miso
www.miso.es

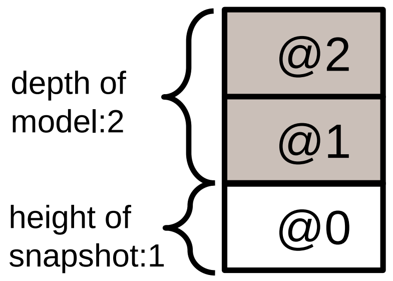
metaDepth

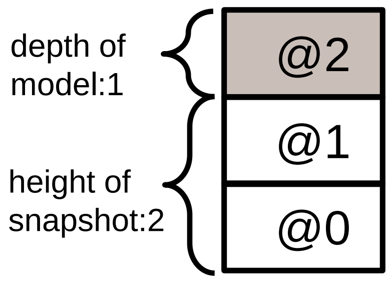
http://astreo.ii.uam.es/~jlara/metaDepth/

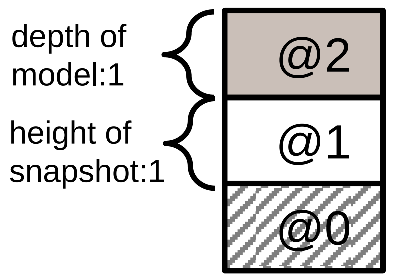
- multi-level textual framework
- constraints defined in any meta-level
- **potency**: meta-level of instantiation (or constraint evaluation)
- **reflection**: useful for constraints to be evaluated >1 meta-levels below
  - *references(r)*: name of references that instantiate reference *r*
  - *value(r)*: content of reference with name *r*

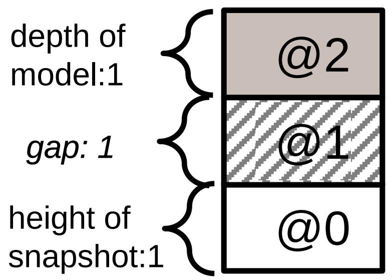
can this model be completed?

SCENARIOS in the analysis of multi-level models









APPLICATIONS

model completion, satisfiability of new constraints at level 1	satisfiability at any meta-level, of a language definition	standard two-level scenario	existence of models@0 with certain features, assess potencies

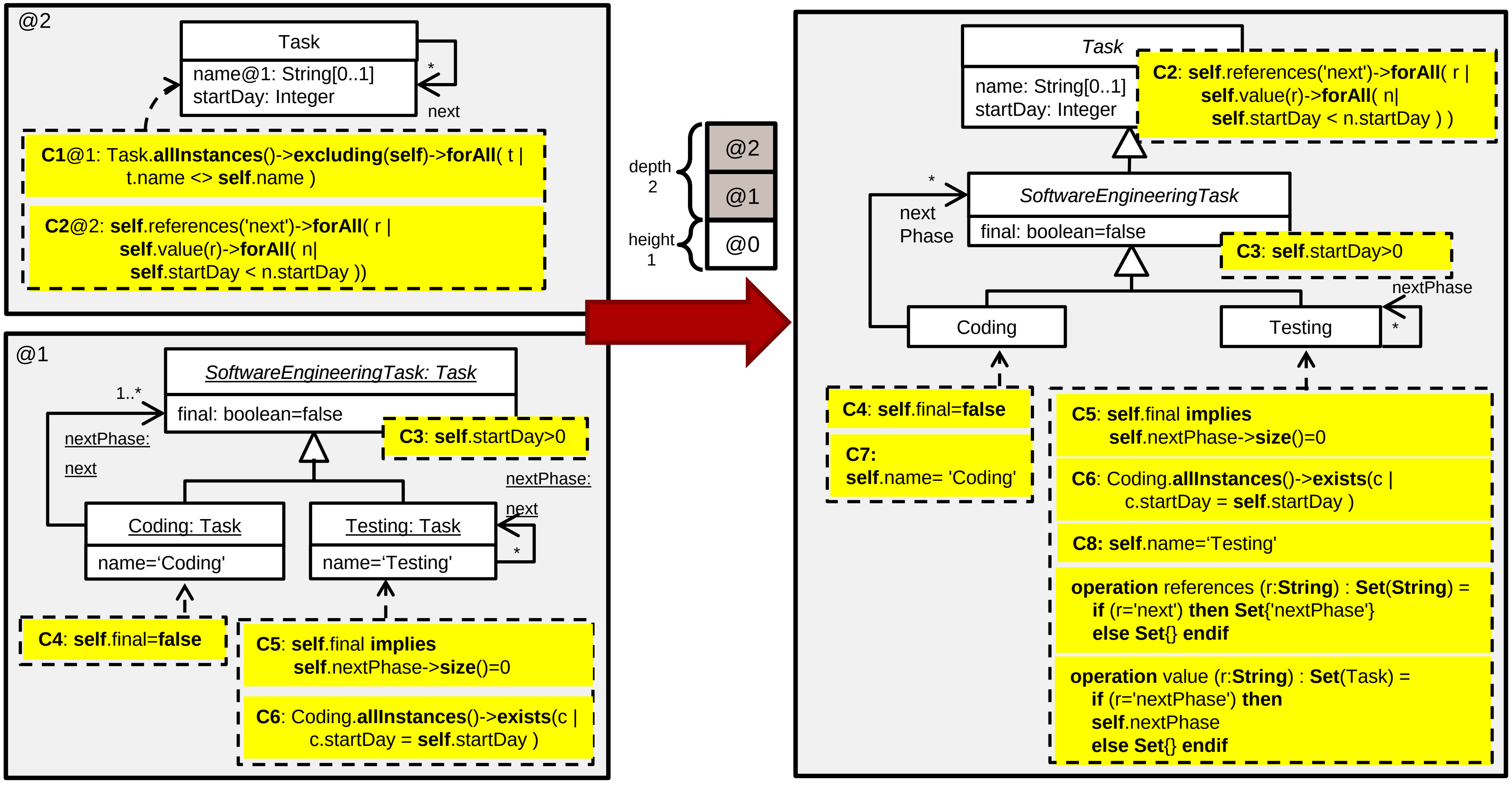
CHALLENGES

flatten two meta-levels into one, which will be input to the finder	emulate the generation of several meta-levels within one	remove constraints with potency > 1	particular case of second scenario

► Flattened model (@2+@1) is used to search models at level 0

www.miso.es

1. Keep all clabjects; top clabjects are set to abstract
2. Keep references at level 1
3. Keep constraints evaluated at level 0 (*C2* to *C6*)
4. Instantiation is replaced by inheritance: attributes with potency 2 become inherited
5. Attribute slots at level 1 (*name*) are removed, their value is given as constraints
6. Emulate built-in operations (*reference*, *value*)



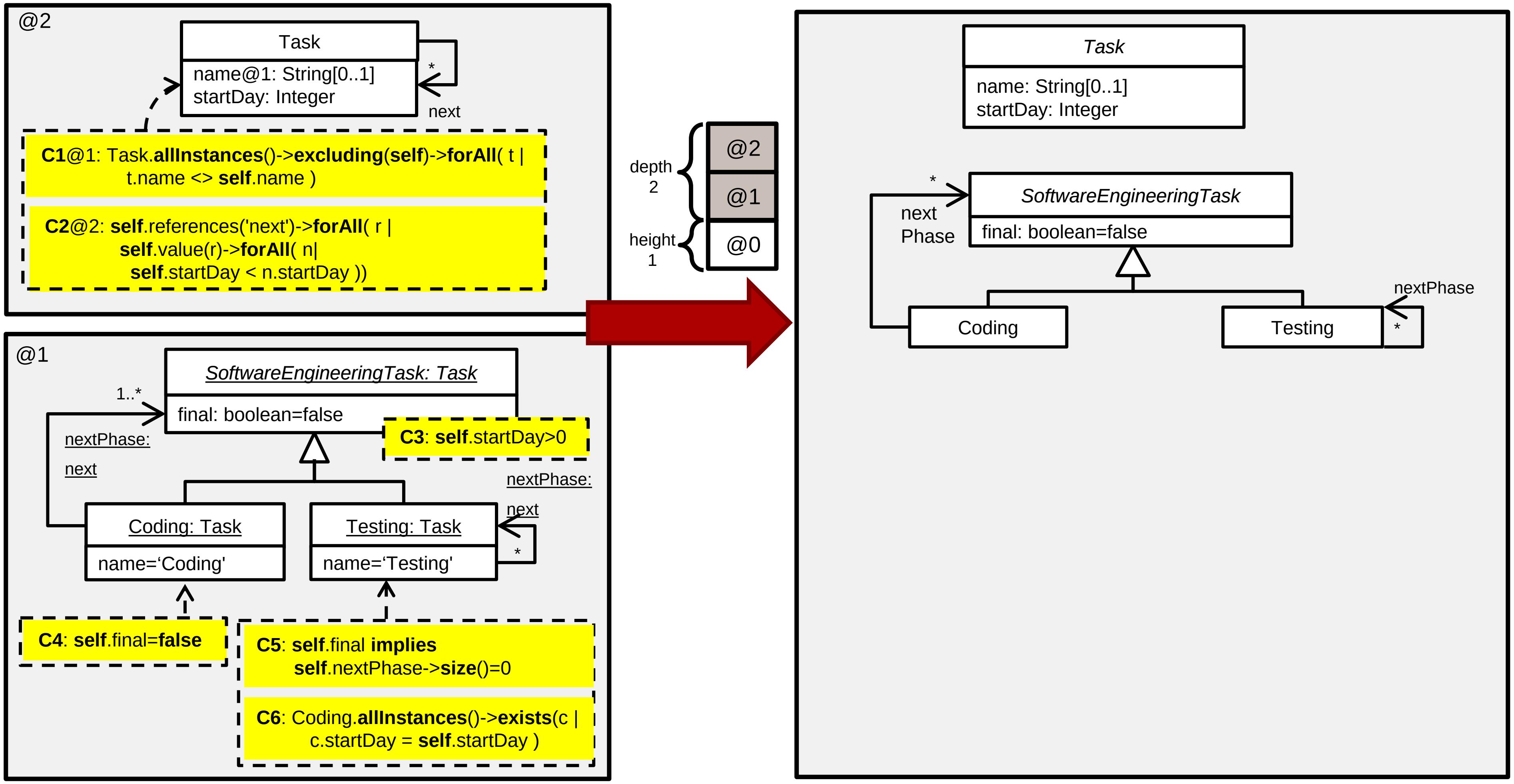
```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
          self.startDay < n.startDay ))

C3: self.startDay>0

C4: self.final=false

C5: self.final implies
        self.nextPhase->size()=0

C6: Coding.allInstances()->exists(c |
        c.startDay = self.startDay )
```

```
C2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
          self.startDay < n.startDay ) )

C3: self.startDay>0

C4: self.final=false

C7:
self.name= 'Coding'

C5: self.final implies
        self.nextPhase->size()=0

C6: Coding.allInstances()->exists(c |
        c.startDay = self.startDay )

C8: self.name='Testing'

operation references (r:String) : Set(String) =
    if (r='next') then Set{'nextPhase'}
    else Set{} endif

operation value (r:String) : Set(Task) =
    if (r='nextPhase') then
    self.nextPhase
    else Set{} endif
```

► Flattened model (@2+@1) is used to search models at level 0

www.miso.es

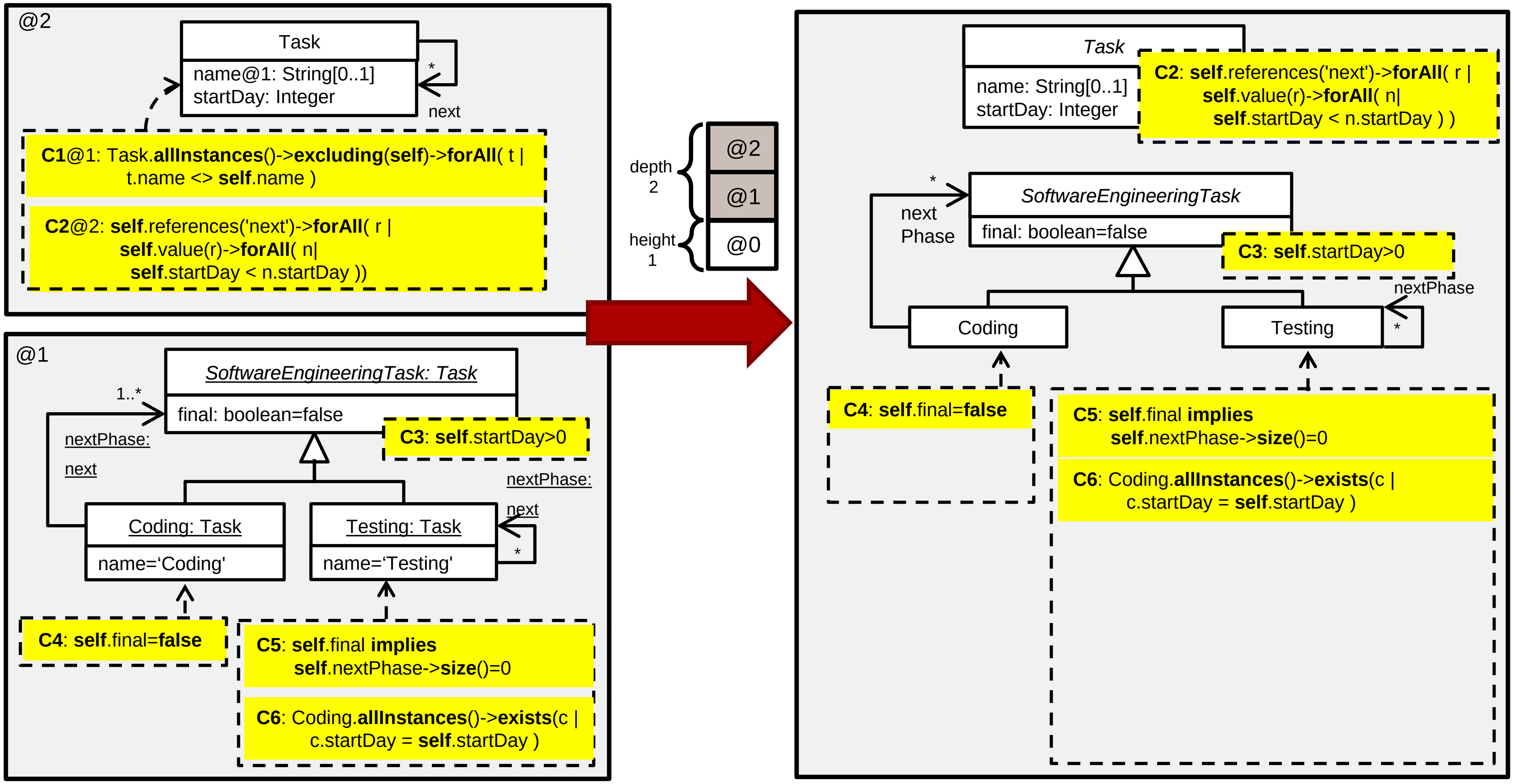
1. Keep all clabjects; top clabjects are set to abstract
2. Keep references at level 1
3. Keep constraints evaluated at level 0 (*C2* to *C6*)
4. Instantiation is replaced by inheritance: attributes with potency 2 become inherited
5. Attribute slots at level 1 (*name*) are removed, their value is given as constraints
6. Emulate built-in operations (*reference*, *value*)



► Flattened model (@2+@1) is used to search models at level 0

www.miso.es

1. Keep all clabjects; top clabjects are set to abstract
2. Keep references at level 1
3. Keep constraints evaluated at level 0 (*C2* to *C6*)
4. Instantiation is replaced by inheritance: attributes with potency 2 become inherited
5. Attribute slots at level 1 (*name*) are removed, their value is given as constraints
6. Emulate built-in operations (*reference*, *value*)

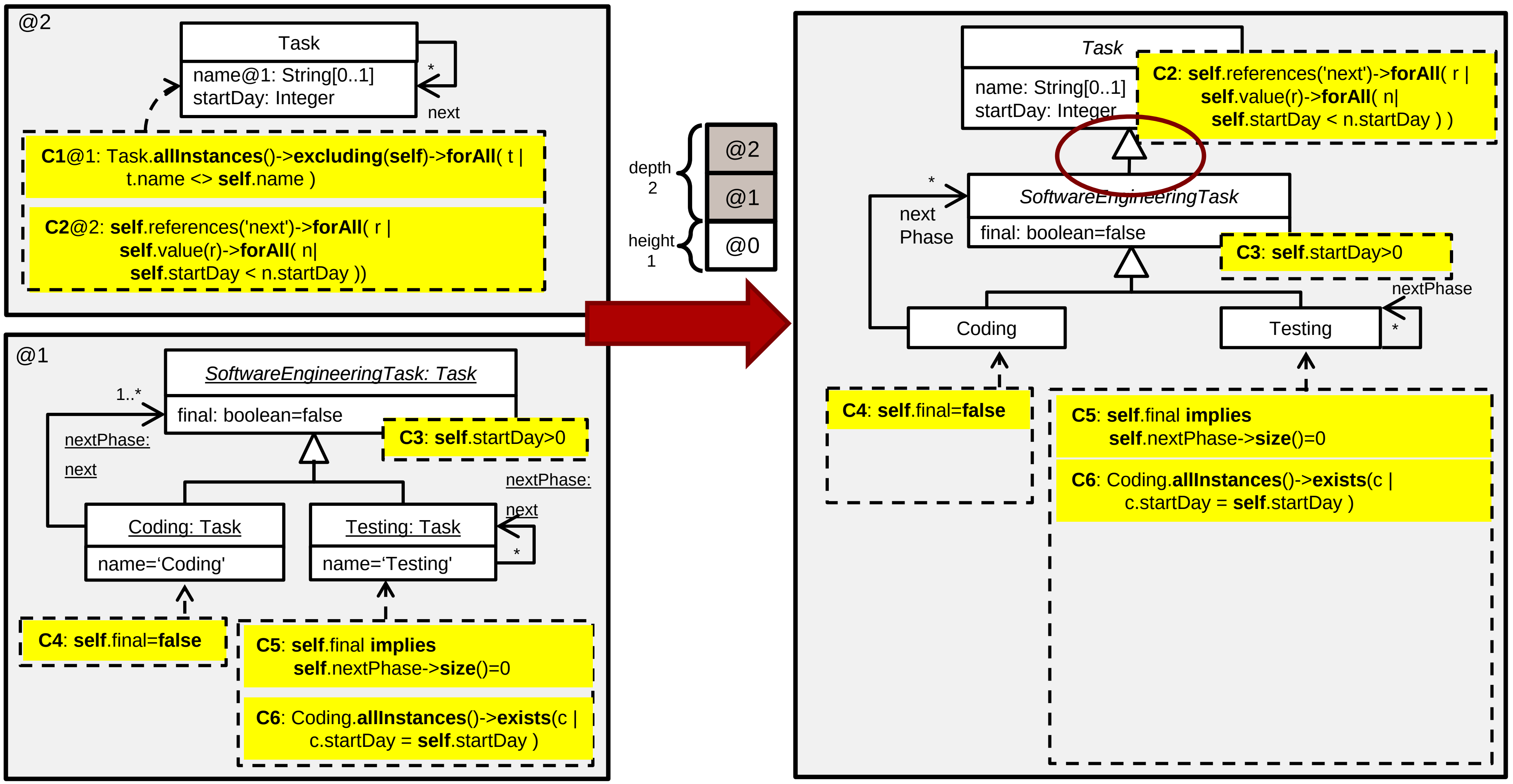


@2
Task
name@1: String[0..1]
startDay: Integer
*
next
C1@1: Task.allInstances()->excluding(self)->forAll(t | t.name <> self.name)
C2@2: self.references('next')->forAll(r | self.value(r)->forAll(n| self.startDay < n.startDay))
@1
SoftwareEngineeringTask: Task
final: boolean=false
1..*
nextPhase: next
C3: self.startDay>0
nextPhase: next
*
Coding: Task
name='Coding'
Testing: Task
name='Testing'
C4: self.final=false
C5: self.final implies self.nextPhase->size()=0
C6: Coding.allInstances()->exists(c | c.startDay = self.startDay)
depth 2
height 1
@2
@1
@0
Task
name: String[0..1]
startDay: Integer
C2: self.references('next')->forAll(r | self.value(r)->forAll(n| self.startDay < n.startDay))
*
next Phase
SoftwareEngineeringTask
final: boolean=false
C3: self.startDay>0
nextPhase
Coding
Testing
*
C4: self.final=false
C5: self.final implies self.nextPhase->size()=0
C6: Coding.allInstances()->exists(c | c.startDay = self.startDay)

► Flattened model (@2+@1) is used to search models at level 0

www.miso.es

1. Keep all clabjects; top clabjects are set to abstract
2. Keep references at level 1
3. Keep constraints evaluated at level 0 (*C2* to *C6*)
4. Instantiation is replaced by inheritance: attributes with potency 2 become inherited
5. Attribute slots at level 1 (*name*) are removed, their value is given as constraints
6. Emulate built-in operations (*reference*, *value*)

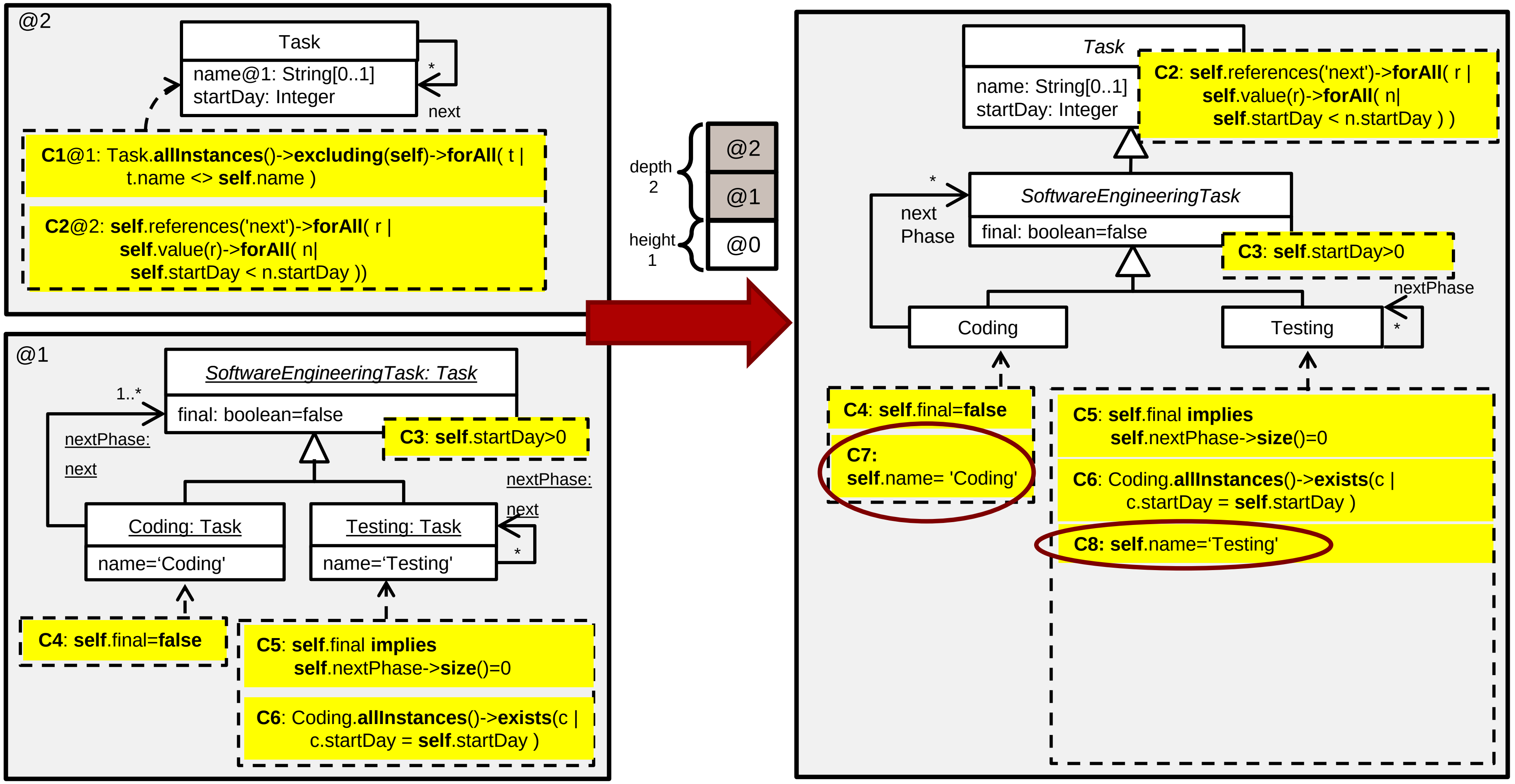


@2
Task
name@1: String[0..1]
startDay: Integer
*
next
C1@1: Task.**allInstances**()->**excluding**(**self**)->**forAll**(t | t.name <> **self**.name)
C2@2: **self**.references('next')->**forAll**(r | **self**.value(r)->**forAll**(n| **self**.startDay < n.startDay))
@1
SoftwareEngineeringTask: Task
final: boolean=false
1..*
nextPhase: next
C3: **self**.startDay>0
Coding: Task
name='Coding'
Testing: Task
name='Testing'
nextPhase: next
*
C4: **self**.final=**false**
C5: **self**.final **implies** **self**.nextPhase->**size**()=0
C6: Coding.**allInstances**()->**exists**(c | c.startDay = **self**.startDay)
depth 2
height 1
@2
@1
@0
Task
name: String[0..1]
startDay: Integer
C2: **self**.references('next')->**forAll**(r | **self**.value(r)->**forAll**(n| **self**.startDay < n.startDay))
SoftwareEngineeringTask
final: boolean=false
*
next Phase
C3: **self**.startDay>0
nextPhase
Coding
Testing
*
C4: **self**.final=**false**
C5: **self**.final **implies** **self**.nextPhase->**size**()=0
C6: Coding.**allInstances**()->**exists**(c | c.startDay = **self**.startDay)

► Flattened model (@2+@1) is used to search models at level 0

www.miso.es

1. Keep all clabjects; top clabjects are set to abstract
2. Keep references at level 1
3. Keep constraints evaluated at level 0 (*C2* to *C6*)
4. Instantiation is replaced by inheritance: attributes with potency 2 become inherited
5. Attribute slots at level 1 (*name*) are removed, their value is given as constraints
6. Emulate built-in operations (*reference*, *value*)



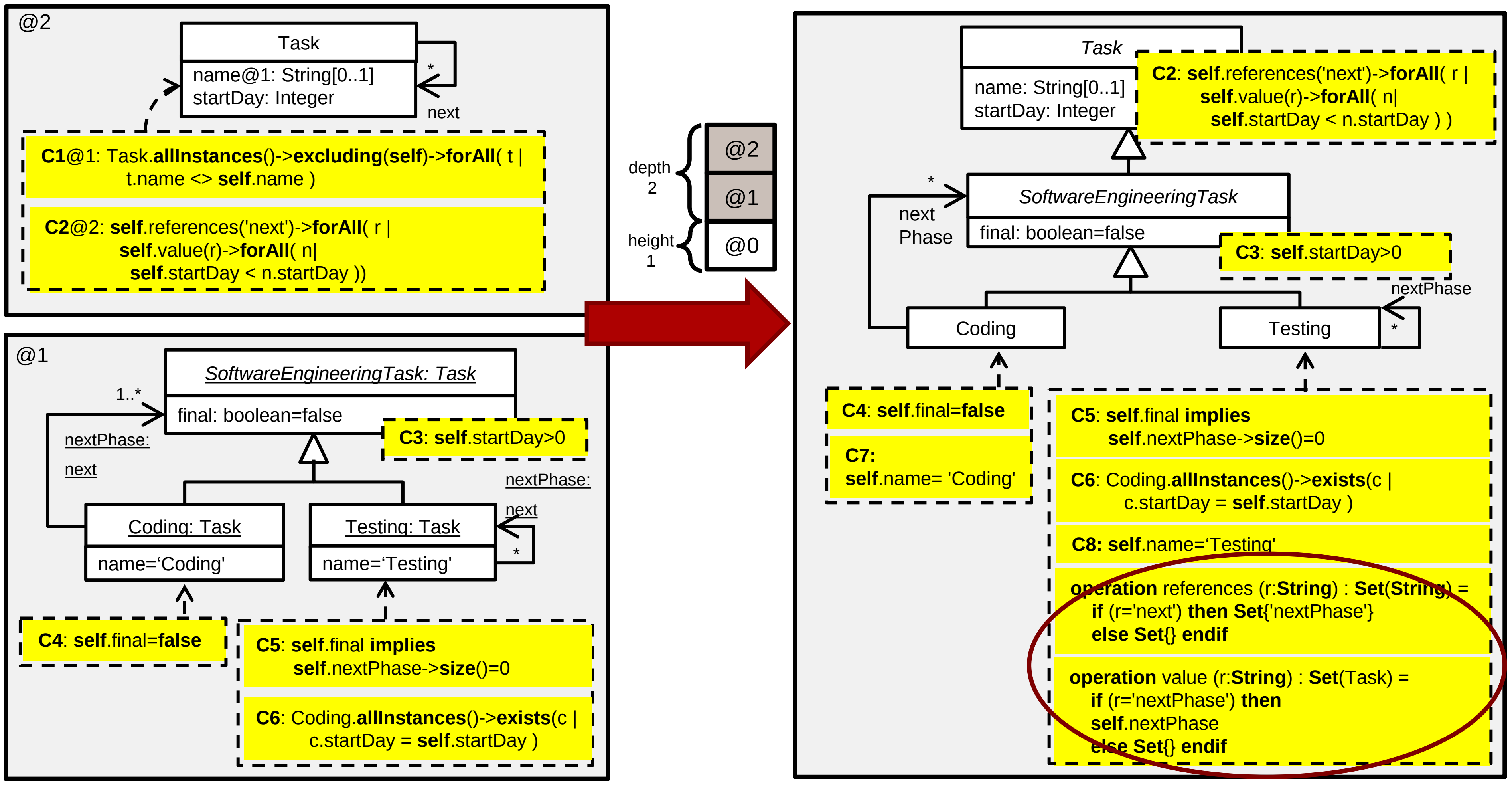
```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
         self.startDay < n.startDay ))

C3: self.startDay>0

C4: self.final=false

C5: self.final implies
       self.nextPhase->size()=0

C6: Coding.allInstances()->exists(c |
       c.startDay = self.startDay )
```

```
C2: self.references('next')->forAll( r |
       self.value(r)->forAll( n|
        self.startDay < n.startDay ) )

C3: self.startDay>0

C4: self.final=false

C7:
self.name= 'Coding'

C5: self.final implies
       self.nextPhase->size()=0

C6: Coding.allInstances()->exists(c |
       c.startDay = self.startDay )

C8: self.name='Testing'
```

► Flattened model (@2+@1) is used to search models at level 0

www.miso.es

1. Keep all clabjects; top clabjects are set to abstract
2. Keep references at level 1
3. Keep constraints evaluated at level 0 (*C2* to *C6*)
4. Instantiation is replaced by inheritance: attributes with potency 2 become inherited
5. Attribute slots at level 1 (*name*) are removed, their value is given as constraints
6. Emulate MetaDepth built-in operations (*reference*, *value*)

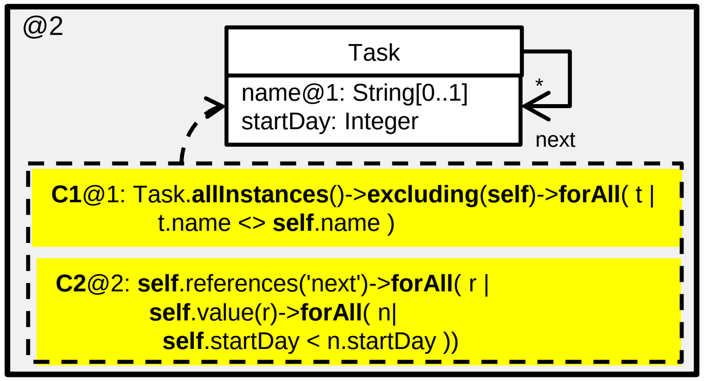


```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
          self.startDay < n.startDay ))

C3: self.startDay>0

C4: self.final=false

C5: self.final implies
      self.nextPhase->size()=0

C6: Coding.allInstances()->exists(c |
      c.startDay = self.startDay )
```

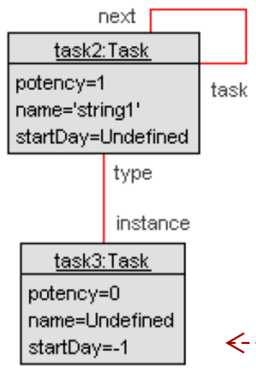
```
C2: self.references('next')->forAll( r |
      self.value(r)->forAll( n|
        self.startDay < n.startDay ) )

C3: self.startDay>0

C4: self.final=false

C7:
self.name= 'Coding'

C5: self.final implies
      self.nextPhase->size()=0

C6: Coding.allInstances()->exists(c |
      c.startDay = self.startDay )

C8: self.name='Testing'

operation references (r:String) : Set(String) =
  if (r='next') then Set{'nextPhase'}
  else Set{} endif

operation value (r:String) : Set(Task) =
  if (r='nextPhase') then
  self.nextPhase
  else Set{} endif
```

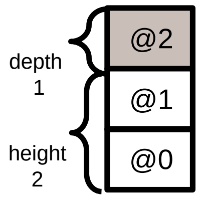
► Extended model is used to search models at levels 1 and 0

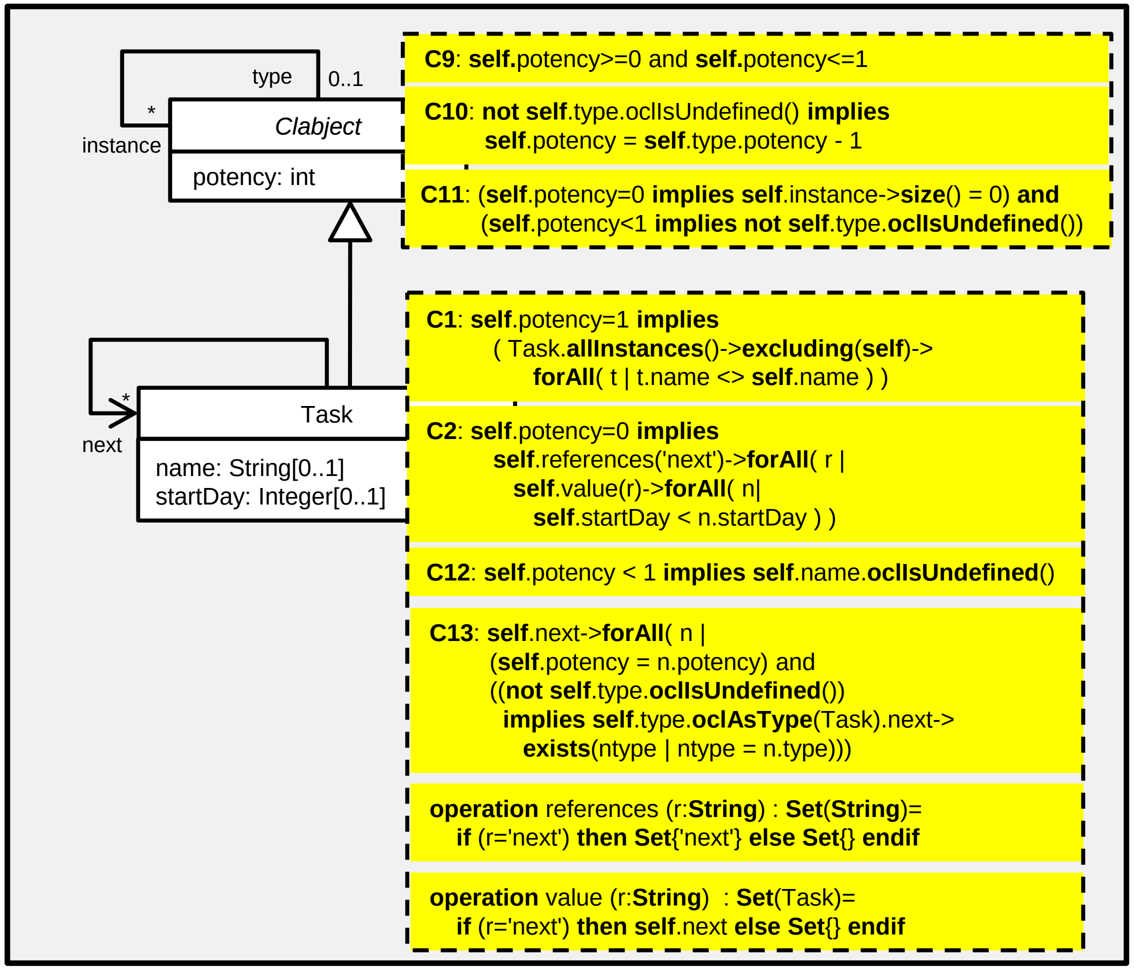
1. Keep all clabjects and references
2. Make explicit clabject features (clabjects inherit from *Clabject*)
3. All constraints are kept, modified to take into account its potency
4. All attributes are set to optional, and set undefined depending on their potency
5. Emulate MetaDepth built-in operations (reference, value)



```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
          self.value(r)->forAll( n|
            self.startDay < n.startDay ))
```



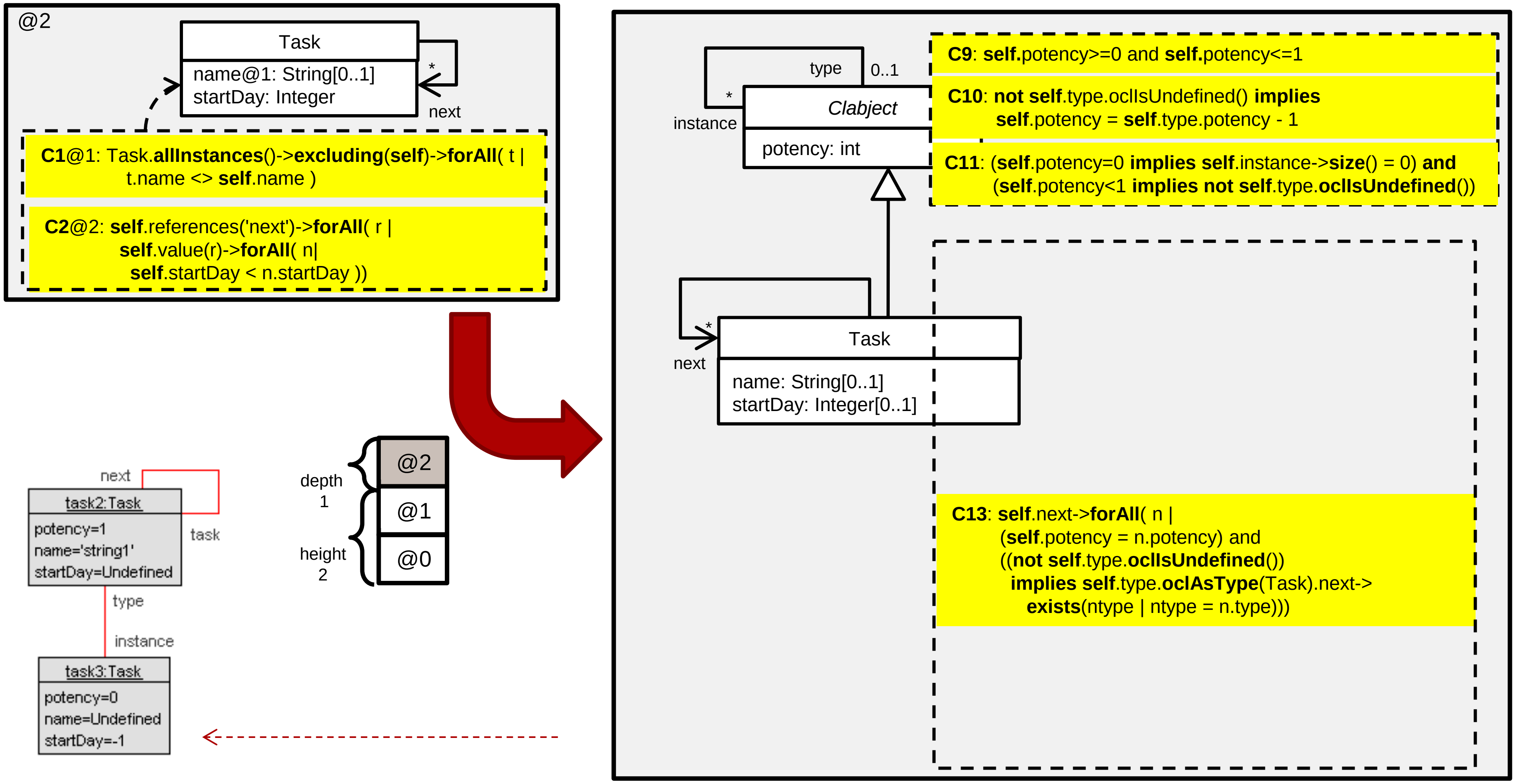




```
C9: self.potency>=0 and self.potency<=1

C10: not self.type.oclIsUndefined() implies
       self.potency = self.type.potency - 1

C11: (self.potency=0 implies self.instance->size() = 0) and
       (self.potency<1 implies not self.type.oclIsUndefined())

C1: self.potency=1 implies
       ( Task.allInstances()->excluding(self)->
           forAll( t | t.name <> self.name ) )

C2: self.potency=0 implies
       self.references('next')->forAll( r |
         self.value(r)->forAll( n|
           self.startDay < n.startDay ) )

C12: self.potency < 1 implies self.name.oclIsUndefined()

C13: self.next->forAll( n |
       (self.potency = n.potency) and
       ((not self.type.oclIsUndefined())
        implies self.type.oclAsType(Task).next->
          exists(ntype | ntype = n.type)))

operation references (r:String) : Set(String)=
   if (r='next') then Set{'next'} else Set{} endif

operation value (r:String)  : Set(Task)=
   if (r='next') then self.next else Set{} endif
```

► Extended model is used to search models at levels 1 and 0

1. Keep all clabjects and references
2. Make explicit clabject features (clabjects inherit from *Clabject*)
3. All constraints are kept, modified to take into account its potency
4. All attributes are set to optional, and set undefined depending on their potency
5. Emulate MetaDepth built-in operations (reference, value)

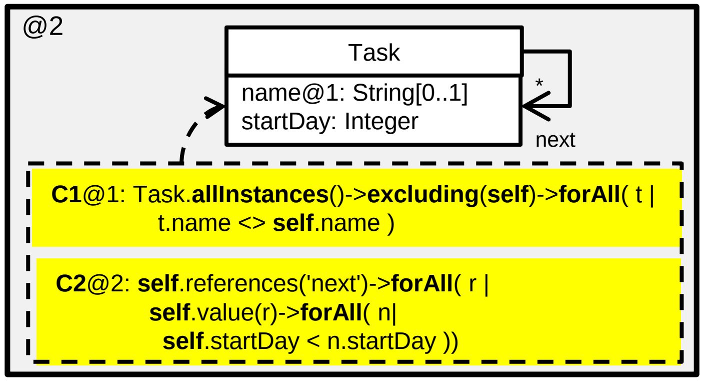


```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
         self.startDay < n.startDay ))

C9: self.potency>=0 and self.potency<=1

C10: not self.type.oclIsUndefined() implies
       self.potency = self.type.potency - 1

C11: (self.potency=0 implies self.instance->size() = 0) and
       (self.potency<1 implies not self.type.oclIsUndefined())

C13: self.next->forAll( n |
       (self.potency = n.potency) and
       ((not self.type.oclIsUndefined())
        implies self.type.oclAsType(Task).next->
          exists(ntype | ntype = n.type)))
```

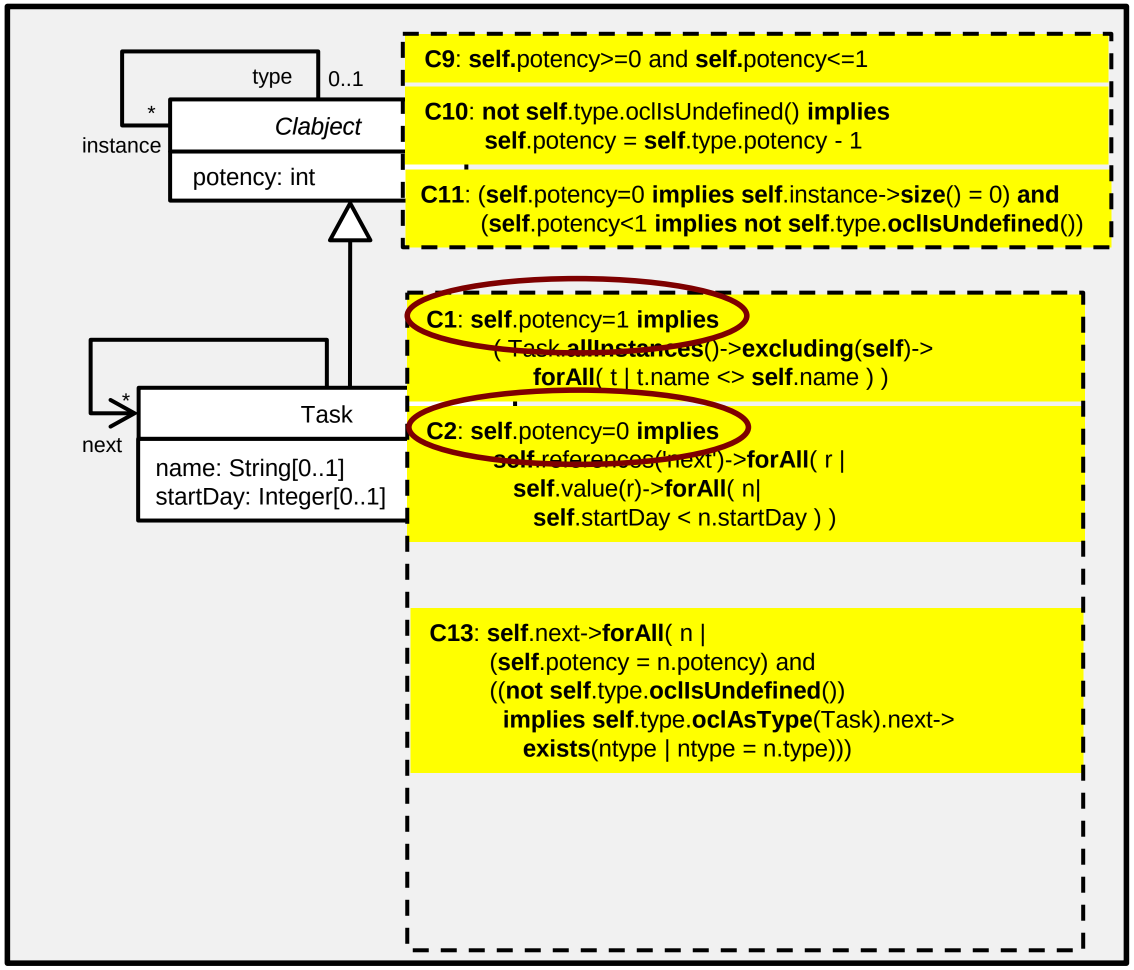
► Extended model is used to search models at levels 1 and 0

www.miso.es

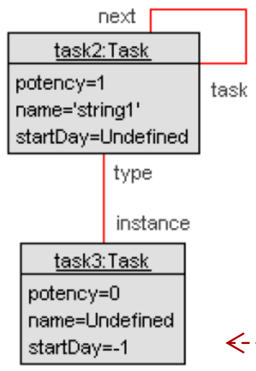
1. Keep all clabjects and references
2. Make explicit clabject features (clabjects inherit from *Clabject*)
3. All constraints are kept, modified to take into account its potency
4. All attributes are set to optional, and set undefined depending on their potency
5. Emulate MetaDepth built-in operations (reference, value)

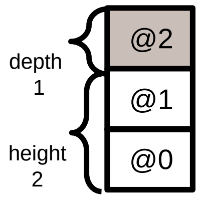


```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
          self.startDay < n.startDay ))
```



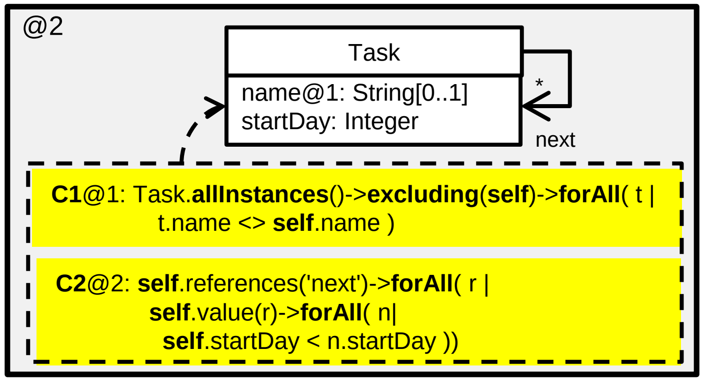
```
C9: self.potency>=0 and self.potency<=1

C10: not self.type.oclIsUndefined() implies
       self.potency = self.type.potency - 1

C11: (self.potency=0 implies self.instance->size() = 0) and
       (self.potency<1 implies not self.type.oclIsUndefined())

C1: self.potency=1 implies
      ( Task.allInstances()->excluding(self)->
          forAll( t | t.name <> self.name ) )

C2: self.potency=0 implies
      self.references('next')->forAll( r |
        self.value(r)->forAll( n|
          self.startDay < n.startDay ) )

C13: self.next->forAll( n |
        (self.potency = n.potency) and
        ((not self.type.oclIsUndefined())
          implies self.type.oclAsType(Task).next->
            exists(ntype | ntype = n.type)))
```



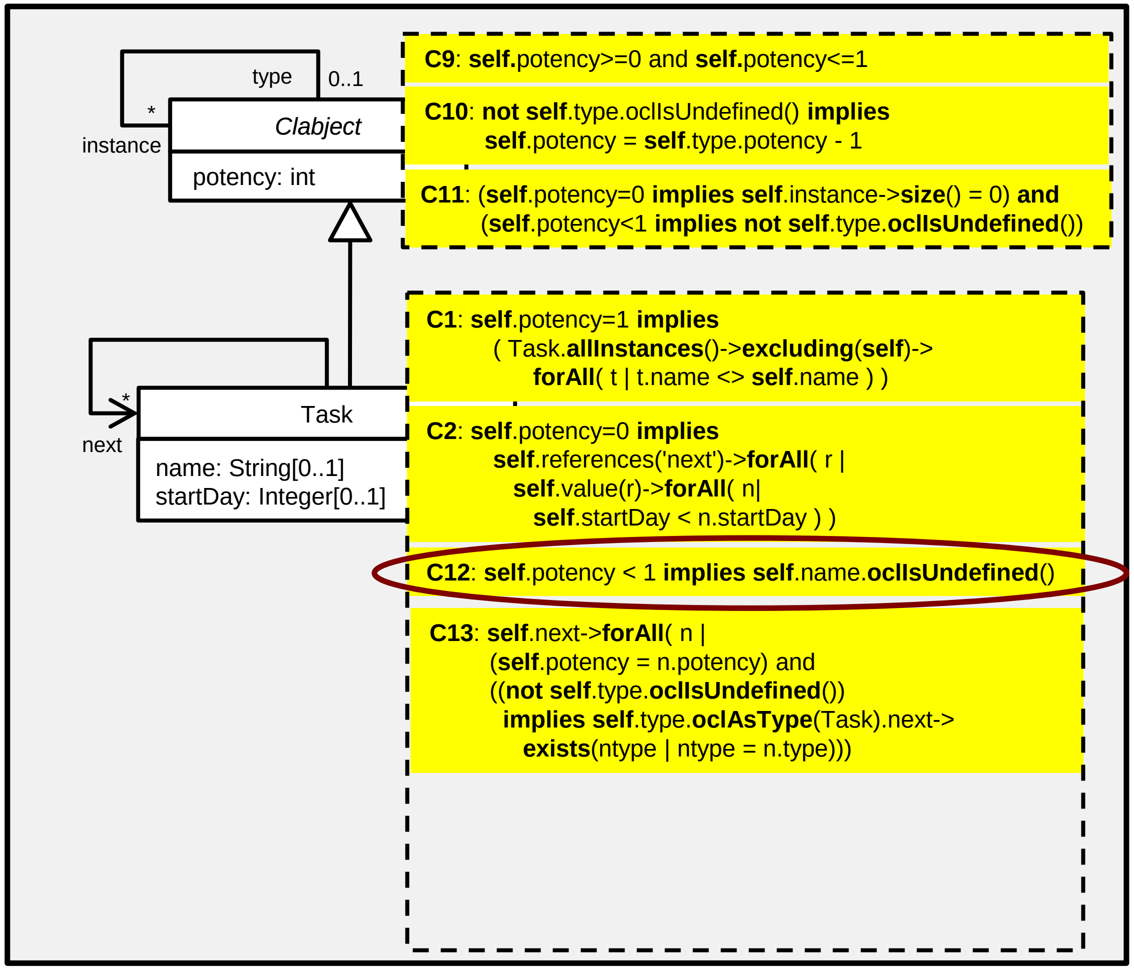


► Extended model is used to search models at levels 1 and 0

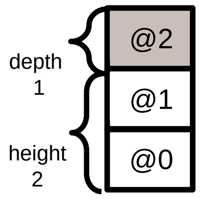
1. Keep all clabjects and references
2. Make explicit clabject features (clabjects inherit from *Clabject*)
3. All constraints are kept, modified to take into account its potency
4. All attributes are set to optional, and set undefined depending on their potency
5. Emulate MetaDepth built-in operations (reference, value)

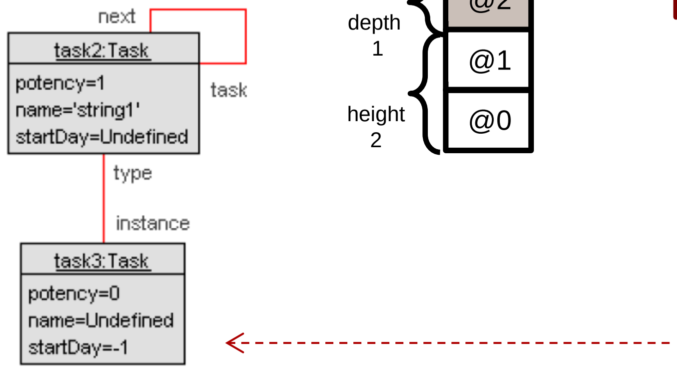


```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
        t.name <> self.name )

C2@2: self.references('next')->forAll( r |
        self.value(r)->forAll( n|
         self.startDay < n.startDay ))
```



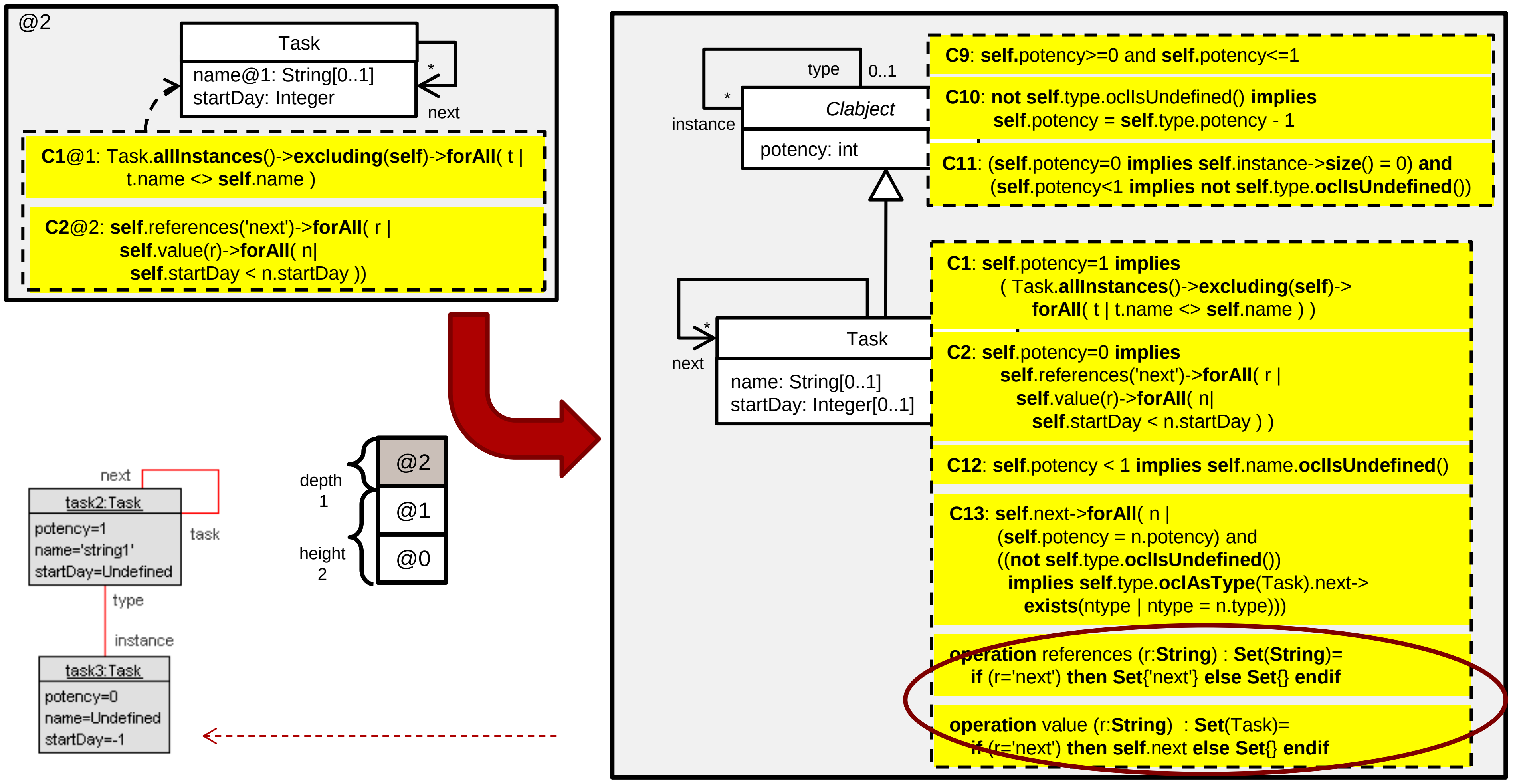
```
C9: self.potency>=0 and self.potency<=1

C10: not self.type.oclIsUndefined() implies
       self.potency = self.type.potency - 1

C11: (self.potency=0 implies self.instance->size() = 0) and
       (self.potency<1 implies not self.type.oclIsUndefined())

C1: self.potency=1 implies
       ( Task.allInstances()->excluding(self)->
          forAll( t | t.name <> self.name ) )

C2: self.potency=0 implies
       self.references('next')->forAll( r |
          self.value(r)->forAll( n|
            self.startDay < n.startDay ) )

C12: self.potency < 1 implies self.name.oclIsUndefined()

C13: self.next->forAll( n |
       (self.potency = n.potency) and
       ((not self.type.oclIsUndefined())
        implies self.type.oclAsType(Task).next->
          exists(ntype | ntype = n.type)))
```





► Extended model is used to search models at levels 1 and 0

1. Keep all clabjects and references
2. Make explicit clabject features (clabjects inherit from *Clabject*)
3. All constraints are kept, modified to take into account its potency
4. All attributes are set to optional, and set undefined depending on their potency
5. Emulate MetaDepth built-in operations (reference, value)

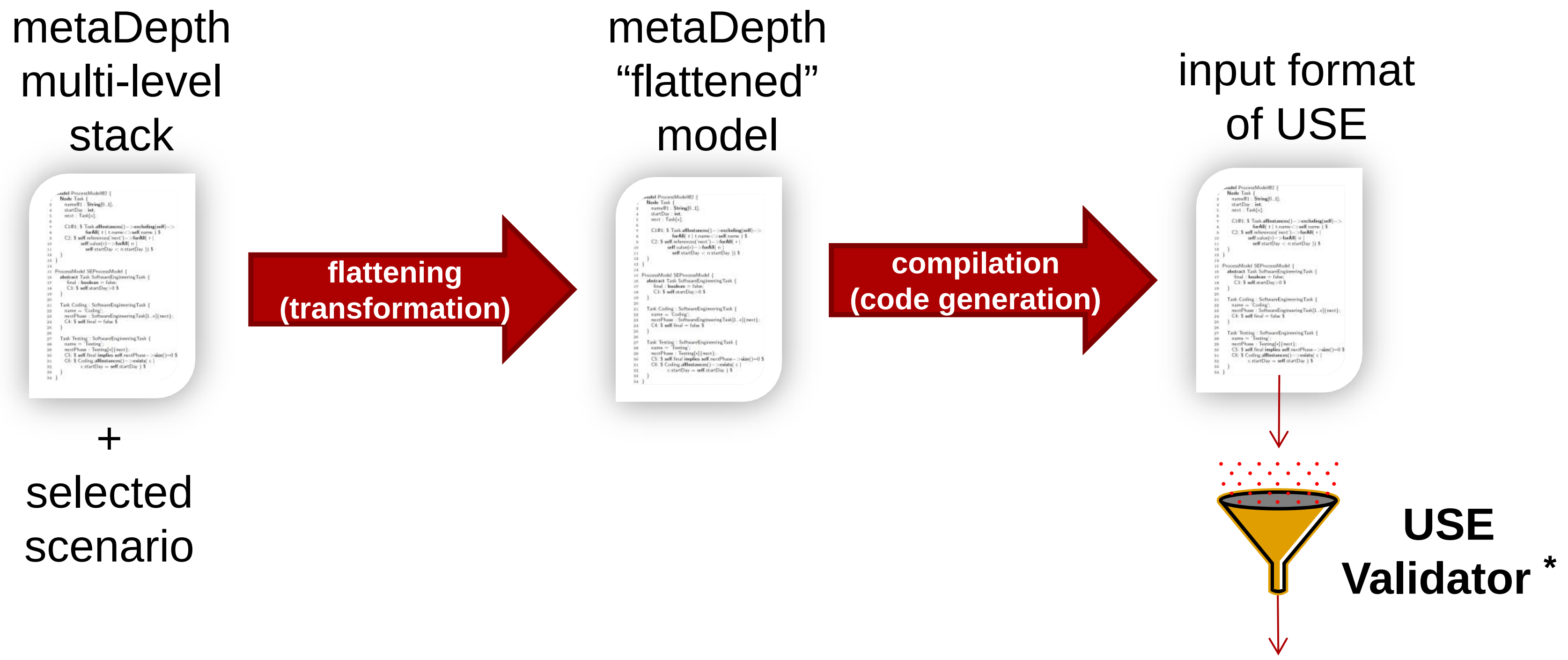


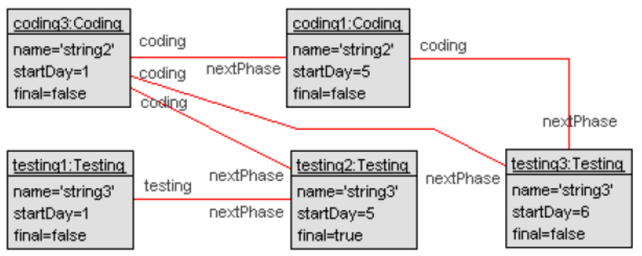
```
C1@1: Task.allInstances()->excluding(self)->forAll( t |
         t.name <> self.name )

C2@2: self.references('next')->forAll( r |
         self.value(r)->forAll( n|
          self.startDay < n.startDay ))
```

```
C9: self.potency>=0 and self.potency<=1

C10: not self.type.oclIsUndefined() implies
        self.potency = self.type.potency - 1

C11: (self.potency=0 implies self.instance->size() = 0) and
        (self.potency<1 implies not self.type.oclIsUndefined())
```

```
C1: self.potency=1 implies
       ( Task.allInstances()->excluding(self)->
          forAll( t | t.name <> self.name ) )

C2: self.potency=0 implies
       self.references('next')->forAll( r |
         self.value(r)->forAll( n|
          self.startDay < n.startDay ) )

C12: self.potency < 1 implies self.name.oclIsUndefined()

C13: self.next->forAll( n |
       (self.potency = n.potency) and
       ((not self.type.oclIsUndefined())
        implies self.type.oclAsType(Task).next->
          exists(ntype | ntype = n.type)))

operation references (r:String) : Set(String)=
  if (r='next') then Set{'next'} else Set{} endif

operation value (r:String)  : Set(Task)=
  if (r='next') then self.next else Set{} endif
```

Tool Support


metaDepth multi-level stack
+
selected scenario
flattening (transformation)
metaDepth "flattened" model
compilation (code generation)
input format of USE
USE Validator *



* M. Kuhlmann, L. Hamann, M. Gogolla. Extensive validation of OCL models by integrating SAT solving into USE. In TOOLS (49), LNCS 6705, pp.: 290-306.

SUMMARY

- Analysis of integrity constraints in multi-level models
  - flattening of multi-model according to analysis scenario
  - use of standard model finders to check satisfiability

FUTURE WORK

- Analysis of other correctness properties
- Tighter integration of MetaDepth and USE Validator
  - translate USE results back to MetaDepth
  - commands to e.g. complete a model
  - …

www.miso.es

TOWARDS AUTOMATING THE ANALYSIS OF INTEGRITY CONSTRAINTS IN MULTI-LEVEL MODELS

Esther Guerra and Juan de Lara

{esther.guerra, juan.delara}@uam.es

Modelling and Software Engineering Group (miso)

Computer Science Department

Universidad Autónoma de Madrid