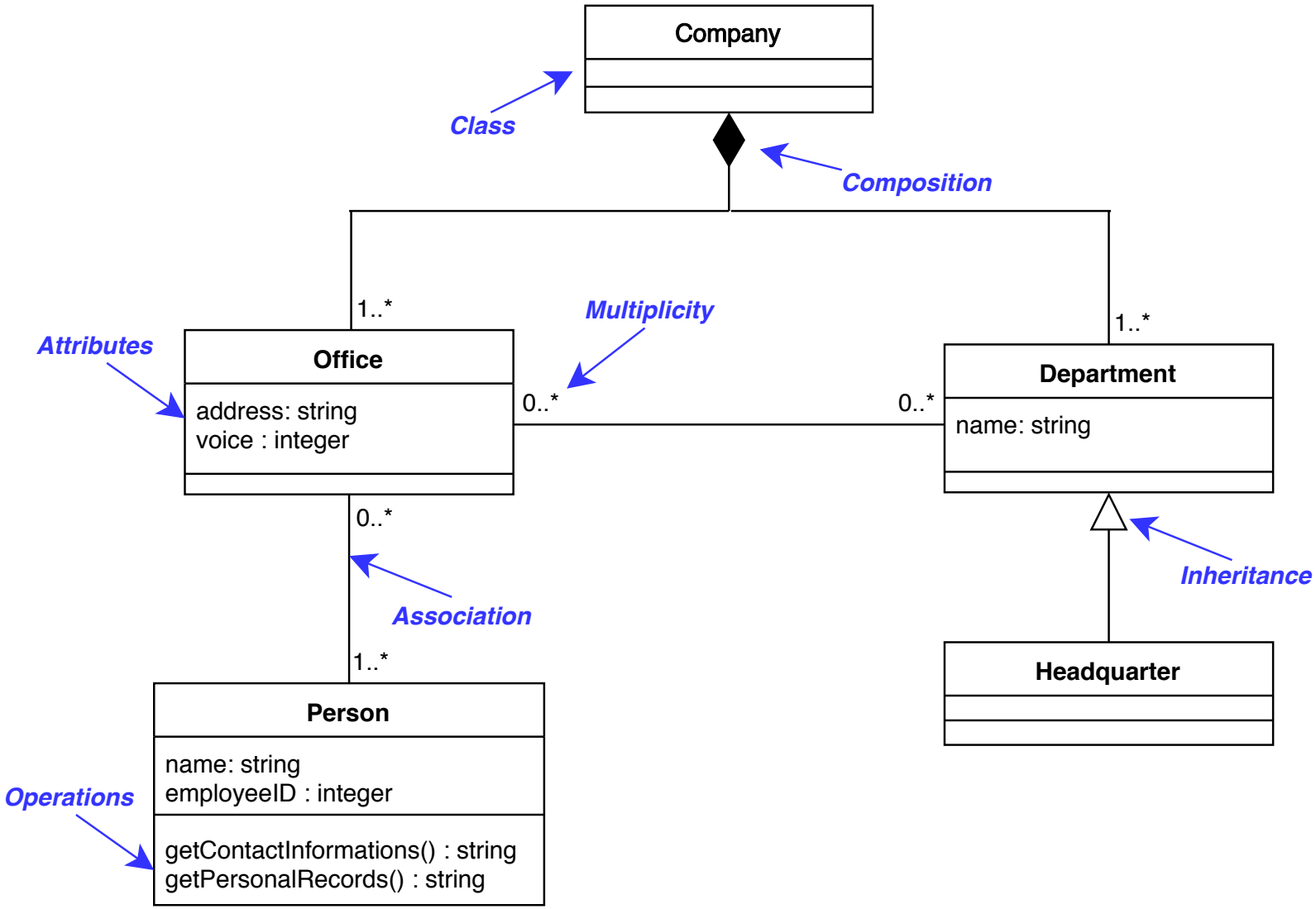
UNIVERSITÄT
DUISBURG
ESSEN

Open-Minded

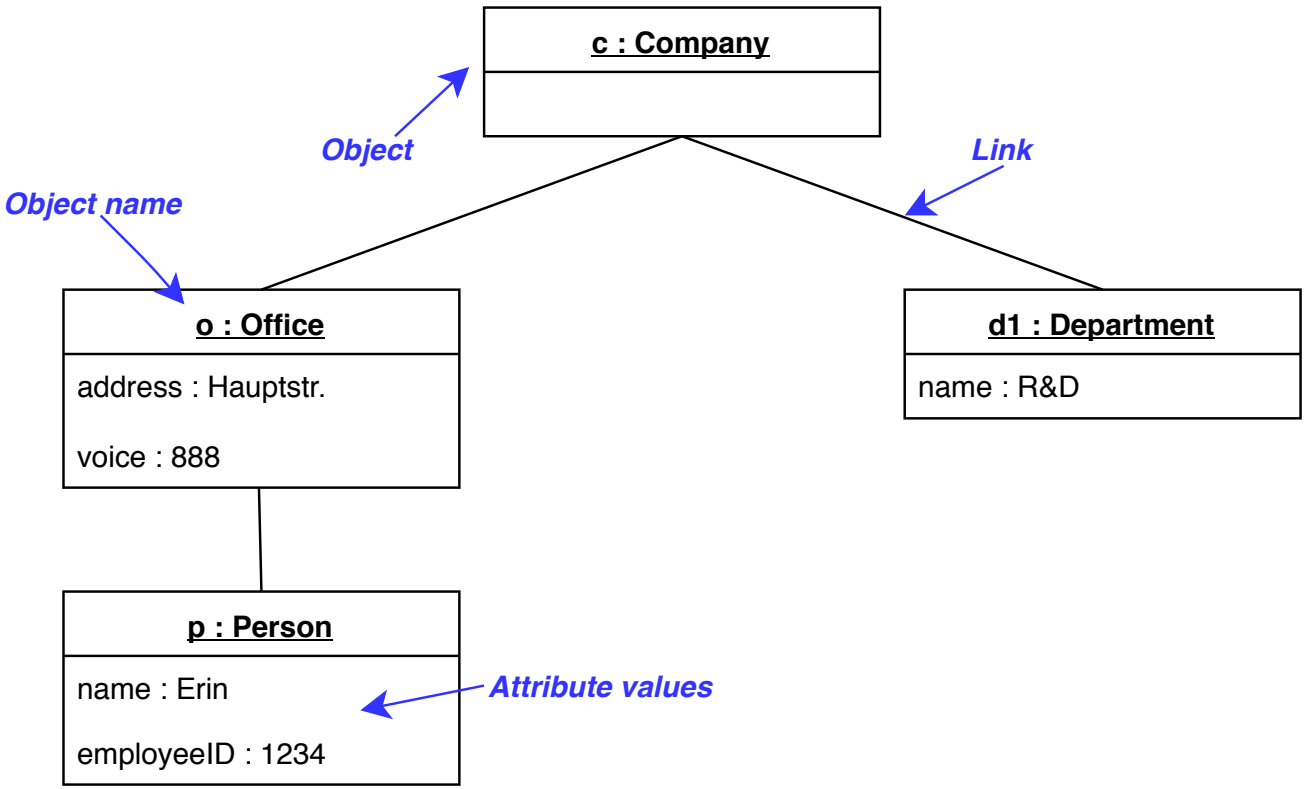
Exercise Task Generation for UML Class/Object Diagrams, via Alloy Model Instance Finding

Violet Kafa, Marcellus Siegburg, Janis Voigtländer ■ 16 July 2019

Motivation: UML (class diagram example)



Motivation: UML (object diagram example)



- This object diagram (OD) conforms to the class diagram (CD) from the previous slide.
- Such *conformance* is an important concept for students to understand, so we want to teach it effectively.

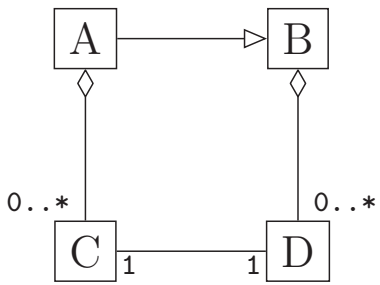
In exercises: examples and counterexamples

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

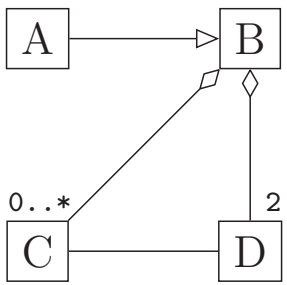
- A typical exercise task, deliberately using artificial names for classes and objects, and a reduced feature set:

Let the following class diagrams be given, each of which shows connections between the classes A, B, C and D.

(1.)

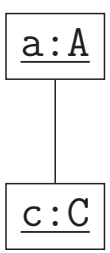


(2.)

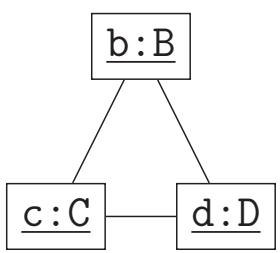


Indicate, for each of the following object diagrams, whether it is valid for the above class diagrams (ten answers altogether). Where that is not the case according to you, explain why and give all reasons.

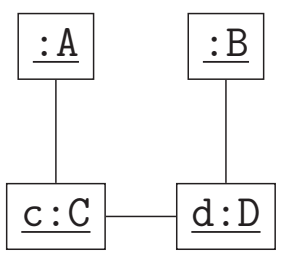
(a)



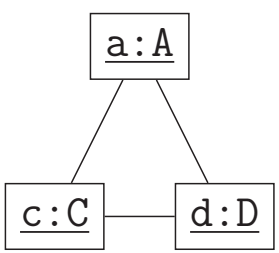
(b)



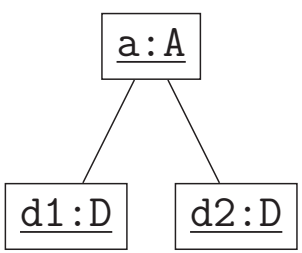
(c)



(d)



(e)



In exercises: examples and counterexamples

What we want of our exercise tasks:

- Diverse examples, in particular variation along the axes of difficulty as well as of specific relationships (and their aspects) covered.
- Verified correctness (respecting all subtleties of the UML standard), to enable reliable grading and feedback to students.
- Targeted attribution of errors/deviations in counterexamples.

Our aim: Automatic generation of many different, but suitably analogous, tasks, ideally per student, along with tailoring for difficulty etc.

Automatic exercise task generation

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

The general idea:

1. Randomly generate CDs respecting configuration options set by the lecturer (how many classes and objects, which types of relationships to involve, additional complexity constraints).
2. *Randomly generate ODs.*

Automatic exercise task generation

The general idea:

1. Randomly generate CDs respecting configuration options set by the lecturer (how many classes and objects, which types of relationships to involve, additional complexity constraints).
2. Translate each CD to a logical specification in the Alloy language, which allows model instance finding via SAT solving.
3. Interpret found solution instances as (positive) OD examples.

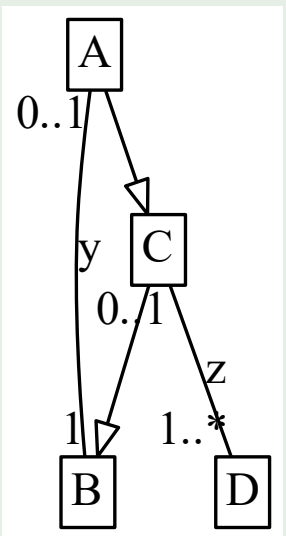
Crucially, devise a strategy for generating negative OD examples as well!

On using Alloy, specifically CD2Alloy

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

CD2Alloy, Maoz et al. (2011):

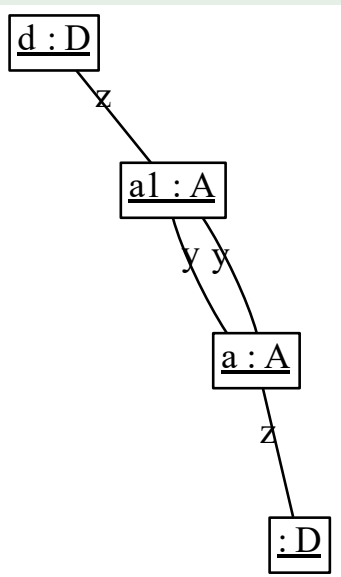
- translating CDs to Alloy specifications
- interpreting found instances as ODs



⇒

```
...
one sig y extends FName {}
...
fun ASubsCD : set Obj { A }
fun BSubsCD : set Obj { B + CSubsCD }
fun CSubsCD : set Obj { C + ASubsCD }
...
ObjLUAttrib[ASubsCD, y, BSubsCD, 1, 1]
ObjLU[BSubsCD, y, ASubsCD, 0, 1]
...
```

⇒



Originally for software developers (Eclipse plugin), here repurposed by us.

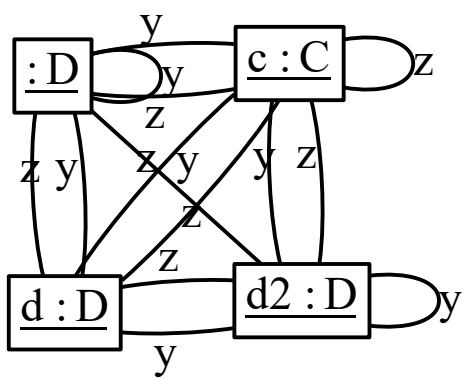
But what about counterexample generation?

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

Unfortunately, this approach does not lend itself immediately to generating useful negative OD examples as well.

The problem is that, continuing the case from the previous slide,

- while Alloy command **run** {cd} for 4 Obj gave plenty useful positive examples (such as the one just seen),
- a naive attempt at counterexample generation, by using command **run** {**not** cd} for 4 Obj, gives not so useful results, such as this one:



But what about counterexample generation?

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

Additional idea:

1. Mutate a given CD slightly, for example by removing a relationship, changing a multiplicity, etc.
2. Translate both original CD and mutated CD to Alloy specifications.
3. Use conjunctive expressions, e.g. **run** {(**not** cd1) **and** cd2}, to search for instances in the difference of the sets of respective models.

What this achieves:

- More interesting/challenging exercise tasks, because counterexamples are not “obviously off” at first sight.
- Precise targeting of deviations, focusing on specific aspects.
- Potential for fine control over the task generation.

Our task generation strategy in more detail

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

Arrived at after quite some experimentation and engineering:

Concrete algorithm (if failing at any step, start over)

1. Generate a random CD0 (respecting the configuration options).
2. Mutate CD0 randomly three times, creating CD1, CD2, and CD3 (but such that at least one of CD1/2 does not contain only inheritances).
3. Find OD instances by running Alloy with each of:
   - (**not** cd1) **and** cd2
   - cd1 **and** (**not** cd2)
   - cd1 **and** cd2
   - (**not** cd1) **and** (**not** cd2) **and** cd3

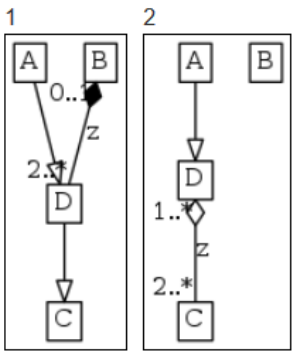
   while imposing additional criteria like “not too many isolated objects”.
4. Select randomly 5 of the found ODs (but at most 2 from each case).
5. Put CD1, CD2, and those 5 ODs into an exercise task.

Integration into an e-learning system

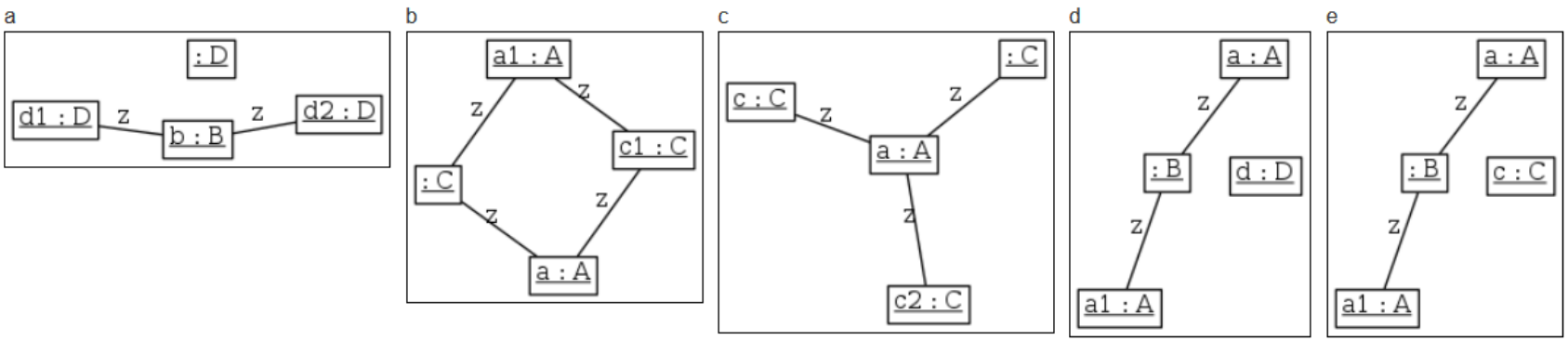
UNIVERSITÄT DUISBURG ESSEN
Open-Minded

https://autotool.fmi.iw.uni-due.de/alloy-cd-od:

Consider the following two class diagrams.



Which of the following five object diagrams conforms to which class diagrams?
An object diagram can conform to neither, either, or both class diagrams.



Please state your answer by giving a list of pairs, each comprising of a class diagram number and a string of object diagram letters.
Each pair specifies that the mentioned object diagrams conform to the respective class diagram.

Conclusion and outlook

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

The presented task generator . . .

- creates CD/OD exercise tasks along with their solutions
- generates individual tasks for every student
- allows for task customisation by the lecturer

Next steps (possible)

- empirical evaluation with student cohorts
- other exercise task types, e.g., letting students identify deviations
- generating (constrained) CDs via Alloy as well, from meta-model
- handling further CD/OD features, and other UML diagrams
- using real world entity names (for classes, objects, . . .)

References

Oliver Kautz, Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CD2Alloy: A translation of class diagrams to Alloy. Techn. Rep. AIB-2017-06, RWTH Aachen University, 2017.

Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CD2Alloy: Class diagrams analysis using Alloy revisited. In *Model Driven Engineering Languages and Systems, Proceedings*, volume 6981 of *LNCS*, pages 592–607. Springer, 2011.

Johannes Waldmann. Generating and grading exercises on algorithms and data structures automatically. In *Automatische Bewertung von Programmieraufgaben, Proceedings*, volume 2015 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.