Dr. Vadim Zaytsev, Universiteit van Amsterdam @ EduSymp'14

@grammarware

Model-based Student Admission

We should test people

the same way

we test software.

Topics edit

- **Test Process:** Behavior driven development, ISO 9000, ISO 9126, CMM, Static code analysis, Lightweight Software Test Automation, Debugging, Mutation analysis, Equivalence Partitioning, Quality control, Software quality, Software testing, Performance engineering, Formal verification, Risk-based Testing, Fault injection, Fagan inspection, Reliability engineering, Software Quality Assurance, Software inspection, Dynamic program analysis, Symbolic computation, Extreme quality assurance, Test automation, Computerized system validation, Testing Web Sites, Quality audit
- **Test levels:** Component or Unit testing, Integration testing, Component integration testing, Acceptance testing, System testing
- **Test types:** Ad hoc testing, Alpha Testing, All-pairs testing, Beta Testing, Black box testing, Boundary testing, Boundary Value Analysis, Build Verification Test, Code coverage, Compatibility testing, Conformance testing, Combinadic, Exploratory testing, Fuzz testing, GUI software testing, Game testing, Hallway testing, Installation testing, Keyword-driven testing, Load testing, Mobile Device Testing, Monkey test, Manual testing, Model-based testing, Playtest, Pseudolocalization, QuickCheck, Regression testing, Recovery testing, Sanity testing, Scenario testing, Soak testing, Software performance testing, Software verification, Smoke testing, Stress testing, Static testing, Session-based testing, Usability testing, White box testing
- **Famous bugs:** List of software bugs
- **People:** Charles E. Brady, Jr., Kenneth D. Cameron, Patrick G. Forrester, Erich Gamma, Charles D. Gemar, Brent Hailpern, Steven Hawley, Cem Kaner, Adam Kolawa, James D. McCaffrey, Brian Marick, Harlan Mills, Stephen S. Oswald, Gene Spafford
- **Companies:** AutomatedQA, Borland, CTG, Compuware, IBM, Lionbridge, Hewlett Packard HP Software Division, Micro Focus, Microsoft, National Software Testing Laboratories, Segue Software, uTest, Telerik
- **Test management:** Test strategy, Test Plan, Test effort, Test Data Generation
- **Tools (commercial):** AdaTEST95, Automation Anywhere, Cantata++, CAST tool, Coverity, ECLAIR, Goanna, IBM OLIVER (CICS interactive test/debug), Insure++, Jinx, Jtest, LDRA Testbed, HP LoadRunner, HP Quality Center, Microsoft Test Professional, Microsoft Visual Studio Ultimate, QF-Test, Polyspace, Ranorex, Silk Performer, SilkTest, SIMMON, TestComplete, TestPartner, Testware, Time Partition Testing, TOSCA, HP WinRunner, Test Studio
- **Tools (free/open source):** AutoIt, CFUnit, CAMV XML, Check, CPPUnit, Curl-loader, DUnit, Fastest, FindBugs, FitNesse, Framework for Integrated Test, FUnit, HttpUnit, Apache JMeter, JUnit, PHPUnit, Litmus (Mozilla), Mauve (test suite), NUnit, PyUnit, RSpec, Selenium, SimpleTest, soapUI, Splint, STAF, TestNG, Watir, WET Web Tester, xUnit
- **Tools (other)** Category:Emulation software, LURCH, Test Automation Framework, Virtual appliance
- **Certification:** British Computer Society, National Software Testing Laboratories, ISTQB, CSTE
- **Membership associations:** Software Engineering Institute, Association for Software Testing, American Society for Quality
- **Software standards:** IEEE 829, TTCN
- **Terminology:** Software bug, Anomaly in software, Test case, Test suite, Test script, Unusual software bug, System under test, Mock object, Test harness, Test data, Testbed, Test bench, Debugger, Boundary case, Verification and Validation, test plan, Test Anything Protocol, Zarro boogs, Thrash (computer science), Memory debugger, Xqa,
- **Miscellaneous:** Software testing outsourcing, Software metric, List of unit testing frameworks

https://en.wikipedia.org/wiki/Portal:Software_testing

Unit testing

Crystal Clear, LGPL

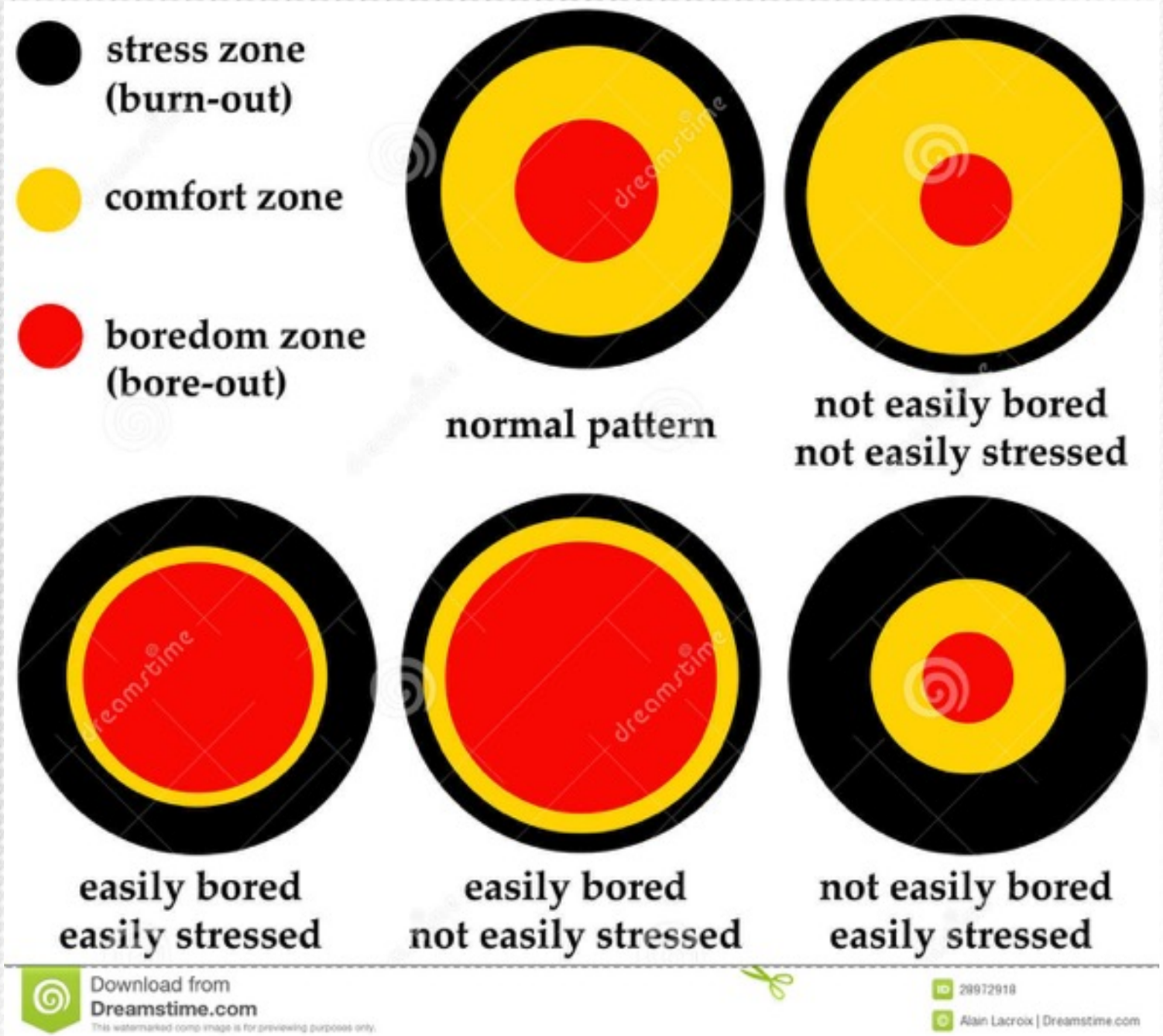
- individual units or modules are tested to determine if they are fit for use

- regular exercises

https://en.wikipedia.org/wiki/Unit_testing

Stress testing

- determines the robustness of software by testing beyond the limits of normal operation
- comfort zone theory



http://www.dreamstime.com/royalty-free-stock-photos-comfort-zone-image28972918

https://en.wikipedia.org/wiki/Stress_testing_(software)

Compatibility testing

- conducted to evaluate one's compatibility with the computing environment
- group assignments

https://en.wikipedia.org/wiki/Compatibility_testing

Installation testing

- focuses on what customers will need to do to install and set up the new software successfully

- dry run = internship

IntelFreePress, Internship and Career Fair, CC-BY.

https://en.wikipedia.org/wiki/Installation_testing

Case study:

UvA MSc SE

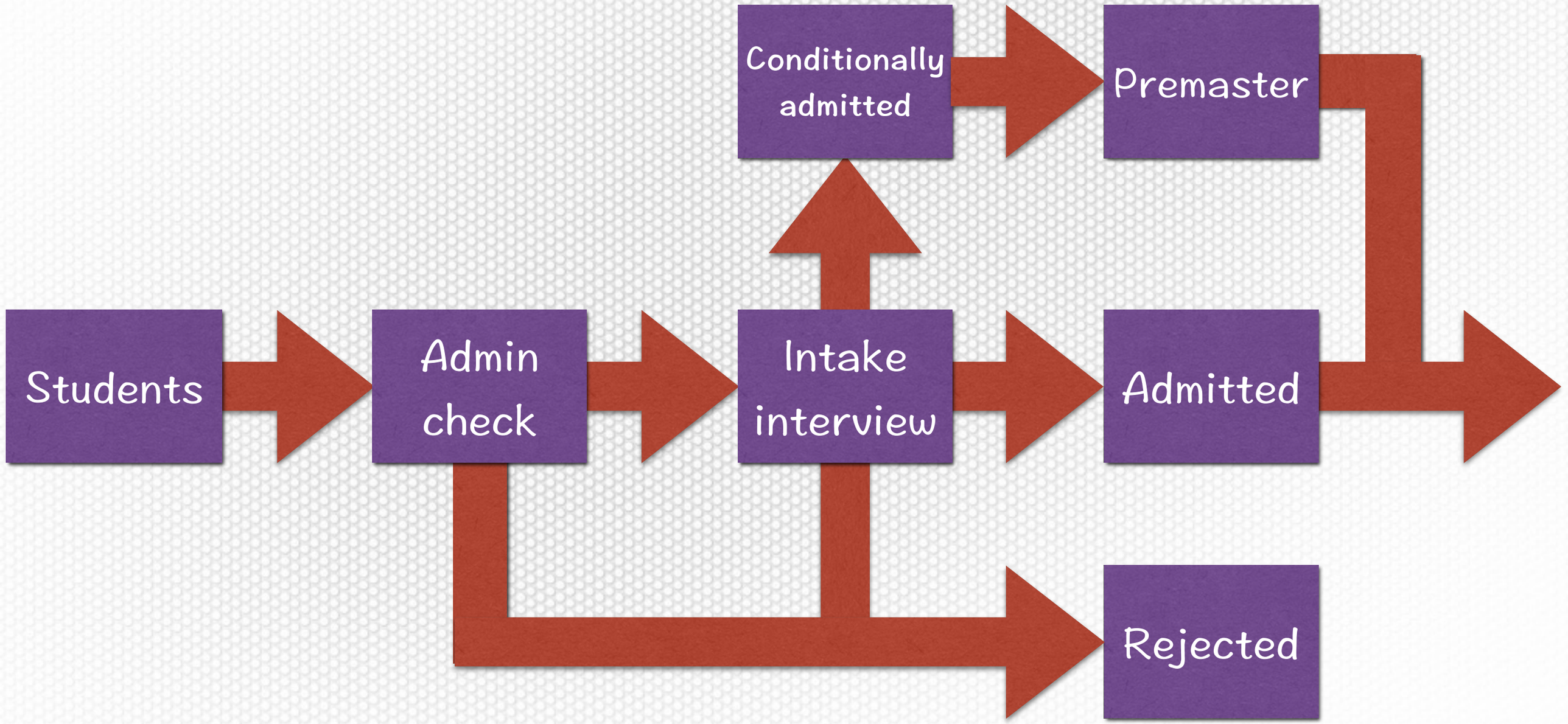
UvA MSc SE

- Software Architecture
- Software Specification and Testing
- Requirements Engineering
- Software Evolution
- Software Process
- Software Construction
- Preparation Master Project

one year

http://www.software-engineering-amsterdam.nl/

Student admission @ MSc SE



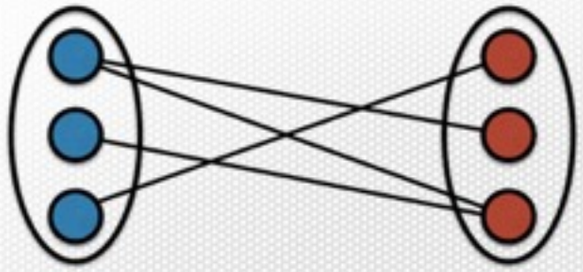
Specification-based testing

- The "read the book, talk to me" paradigm
  - nonfunctional ⇒ hard to test [in a general way]
- Emerging infrastructure
- Is SUT familiar with a notation?
- Can SUT recognise equivalent representations?

Also for flipped classroom

Sets

Vadim Zaytsev aka @grammarware,
Universiteit van Amsterdam, 2014

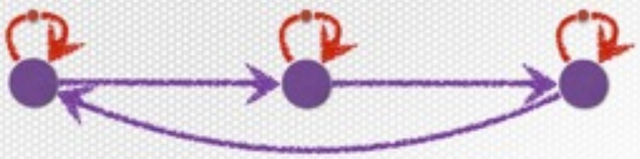


Relations and functions

Vadim Zaytsev aka @grammarware,
Universiteit van Amsterdam, 2014

Also for flipped classroom

Properties of relations



Vadim Zaytsev aka @grammarware,
Universiteit van Amsterdam, 2014

Liskov Substitution Principle

Vadim Zaytsev aka @grammarware,
Universiteit van Amsterdam, 2014

Prolog encoding

equiv.pl

Raw

```
eq('⊇'(A,B), Q) :- eq('⊆'(B,A), Q).
eq('⊆'(A,B),'∀'(X,'→'(LH,RH))) :- xina(X,A,LH), xina(X,B,RH); xnina(X,B,LH), xnina(X,A,RH).
eq('⊆'(A,B),'∀'(X,A,E)) :- xina(X,B,E).

xina(X, A, '∈'(X,A)).
% xina(X, A, '¬'(XA)) :- xnina(X,A,XA).
xina(X, '∪'(A,B), '∨'(XA,XB)) :- xina(X,A,XA), xina(X,B,XB).
xina(X, '∩'(A,B), '∧'(XA,XB)) :- xina(X,A,XA), xina(X,B,XB).
xina(X, ' '(A,B), '∧'(XA,XB)) :- xina(X,A,XA), xnina(X,B,XB).
xina(X, 'C'(A), XA) :- xnina(X,A,XA).
xnina(X, A, '∉'(X,A)).
% xnina(X, A, '¬'(XA)) :- xina(X,A,XA). %, XA =/= '¬'(_).
xnina(X, '∪'(A,B), '∧'(XA,XB)) :- xnina(X,A,XA), xnina(X,B,XB).
xnina(X, '∩'(A,B), '∨'(XA,XB)) :- xnina(X,A,XA), xnina(X,B,XB).
xnina(X, ' '(A,B), '∨'(XA,XB)) :- xnina(X,A,XA), xina(X,B,XB).
xnina(X, 'C'(A), XA) :- xina(X,A,XA).
```

https://gist.github.com/grammarware/813374043858030b2059

?- **eq('⊆'(a,b),'∀'(x,a,'∈'(x,b))).**

true .

?- **eq('⊆'(a,b), E).**

E = '∀'(_G276, '→'('∈'(_G276, a), '∈'(_G276, b))) ;

E = '∀'(_G12, '→'('∉'(_G12, b), '∉'(_G12, a))) ;

E = '∀'(_G12, a, '∈'(_G12, b)) ;

false.

?- **eq('⊇'('∪'(a,b),c), E).**

E = '∀'(_G285, '→'('∈'(_G285, c), '∈'(_G285, '∪'(a, b)))) ;

E = '∀'(_G15, '→'('∈'(_G15, c), 'v'('∈'(_G15, a), '∈'(_G15, b)))) ;

E = '∀'(_G15, '→'('∉'(_G15, '∪'(a, b)), '∉'(_G15, c))) ;

E = '∀'(_G15, '→'('∧'('∉'(_G15, a), '∉'(_G15, b)), '∉'(_G15, c))) ;

E = '∀'(_G15, c, '∈'(_G15, '∪'(a, b))) ;

E = '∀'(_G15, c, 'v'('∈'(_G15, a), '∈'(_G15, b))) ;

false.

Real result:

Question 7. Rephrase the following independent statements with relations on sets:

1. $x \in A \to x \notin B$
2. $x \in A \leftrightarrow x \in B$
3. $x \in A \lor x \in B \to x \in C$

Question 8. Rephrase the following independent statements with quantifiers:

1. $A \setminus B = \varnothing$
2. $2^A \subseteq 2^B$
3. $A \neq \varnothing$

Another result:

Question 9. If c_t^v is a substitution of v with t in c, then what are the results of the following substitutions?

1. $(\forall x Rxy)_z^y =$
2. $(\forall x \forall y (Rxy \wedge Ryz))_y^x =$
3. $(\exists y \neg Rxy)_y^z =$
4. $((\exists x Rxy) \wedge (\forall y Rxy))_y^x =$

A cooler* result:

Question 11. Consider the following grammar:

$$V ::= a \mid b \mid \cdots \mid y \mid z \tag{1}$$
$$C ::= C \div C \mid \Cap C \mid C \bowtie C \mid \blacktriangleright V \blacktriangleleft \tag{2}$$

Recognise correct C terms according to the grammar and draw parse trees of correct ones:

1. $a \div b$
2. $\blacktriangleright\ p\ \blacktriangleleft\bowtie\blacktriangleright\ q\ \blacktriangleleft$
3. $\blacktriangleright\Cap\ o\ \blacktriangleleft$
4. $\Cap\blacktriangleright\ o\ \blacktriangleleft$

* cooler = with grammars

Concluding statements

- We **know** about **software** testing
- Leverage this for **assessment / admission**
- **Installation testing** vs. **Acceptance testing**
- **Exception handling & Destructive testing**
- **Coverage** criteria
- Many more issues to be **explored in the future**