Westfälische Wilhelms-Universität Münster

E-Assessment of Creative Tasks

Herbert Kuchen

joint work with: Dominik Böhm, Susanne Gruttmann, Tim Majchrzak,

Claus Usener, Christoph Lembeck, Roger Müller, Marko Ernsting

European Research Center for Information Systems

wissen.leben
WWU Münster

ERCIS

> Introduction

- regular assessment of the learning progress required
- identify and evaluate teaching success
- mass lectures, decreasing resources and low personnel capacities
- computer-supported assessments help to reduce time and costs

> Formative Assessments

- regular exercises as complementary service for lectures:

> Students	> Lecturers / Tutors
• get an active part in learning process • reflect lecture content • practice and consolidate newly learned knowledge • generate skills while solving corresponding tasks	• get an overview on learning progress of students • detect whether educational objectives are achieved

> Formative assessments can enhance quality of education <

> Processes in Traditional Formative Assessments

Preparation	Execution	Follow-up
• planning assessment • defining topics and goals • creating exercises • generating sample solutions	• collecting exercise sheets • solving exercises • generating submissions • submitting work	• collecting solutions • evaluating and marking solutions • providing feedback • compiling statistics

> E-Assessment Systems Today

- provide multiple choice questions, insertion of short text, and other simple forms of questions

- existing systems check knowledge rather than skills

> How to check creative skills? <

Creative Forms of E-Assessment

- Mathematical proofs
- Programming
- …

> Agenda

> Introduction

> Assessments in Mathematics

> The E-Assessment System EASy

> E-Assessment of Java Programs

> Conclusion

> Formative Assessment of Mathematical Proofs

- proof exercises indispensable in math-related lectures
- complexity proofs is often high
- evaluating and grading proofs is complex and time-consuming

> Electronic support for assessment of mathematical proofs? <

> Shortcomings of Existing Systems for Mathematics

- only few e-assessment systems focus on mathematics
- simple knowledge checks with multiple choice etc.
- some provide support for numeric and symbolic calculations
- some systems support propositional logic
- no literature on processes for e-assessment of proofs in general

> Electronic assessment of proofs is not provided <

> Agenda

> Introduction

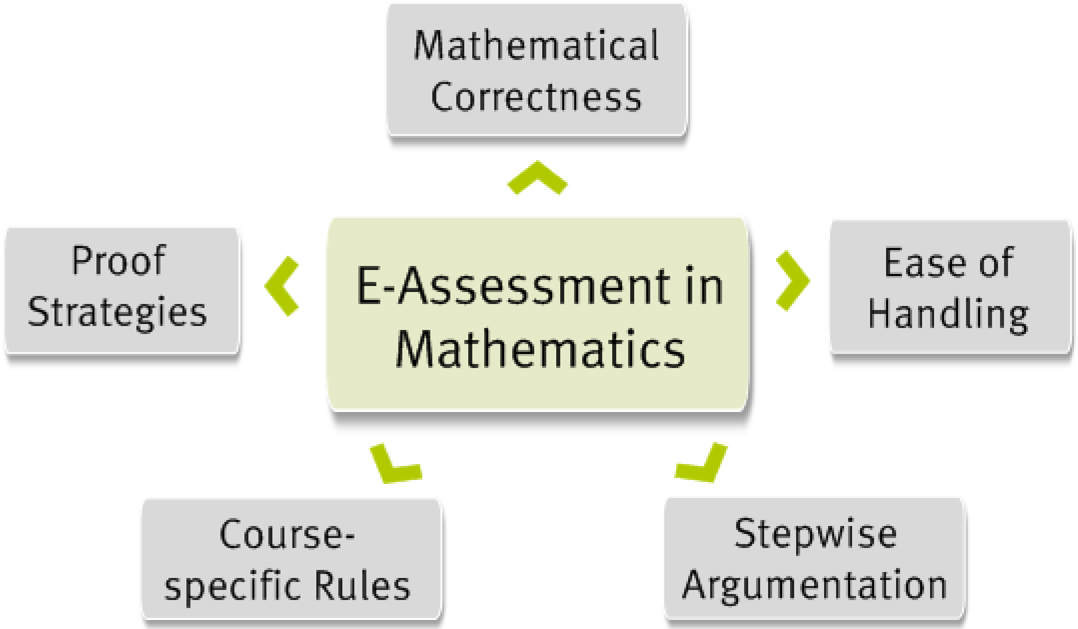
> Assessment in Mathematics

> The E-Assessment System EASy

> E-Assessment of Java Programs

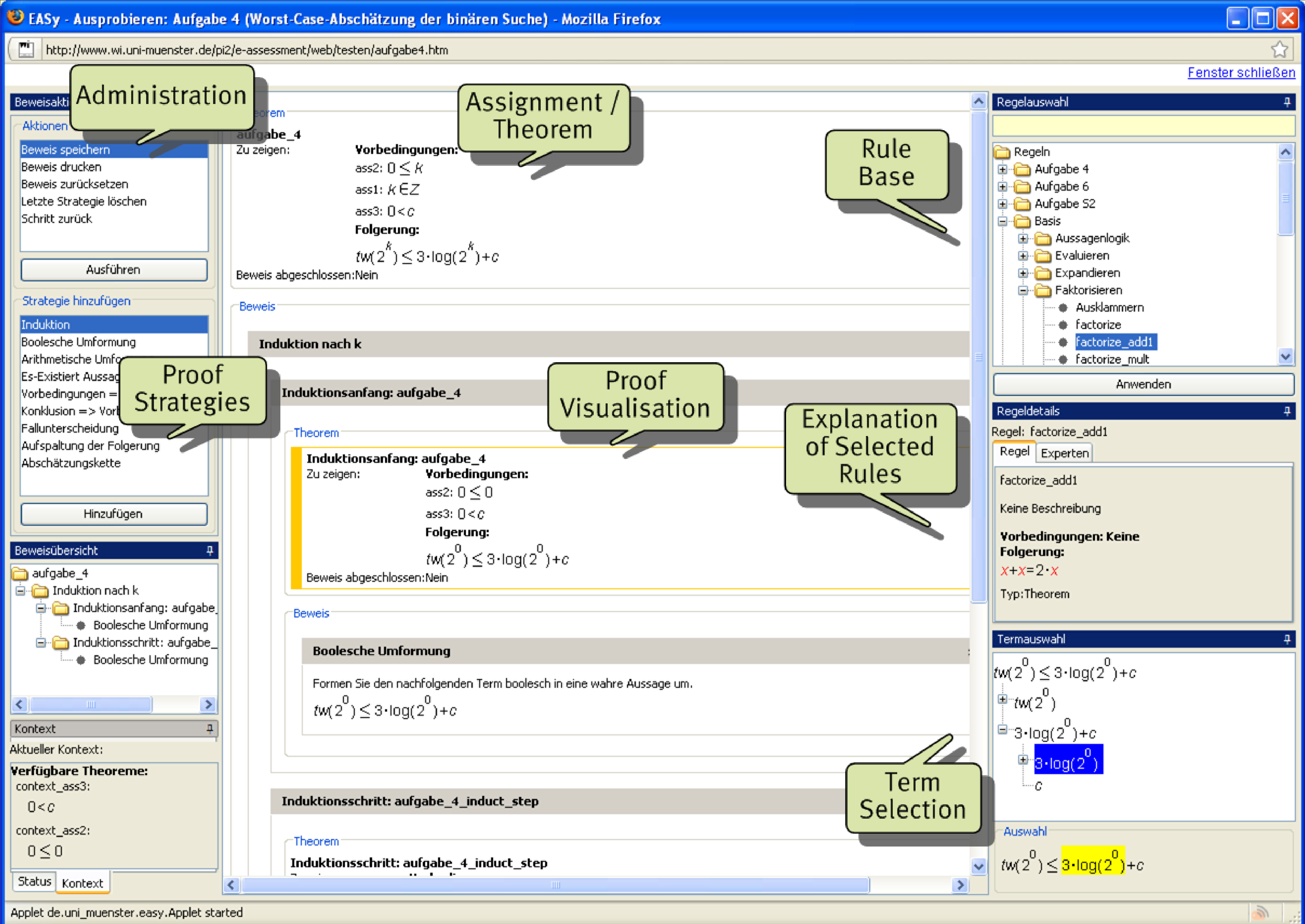
> Conclusion

> Requirements

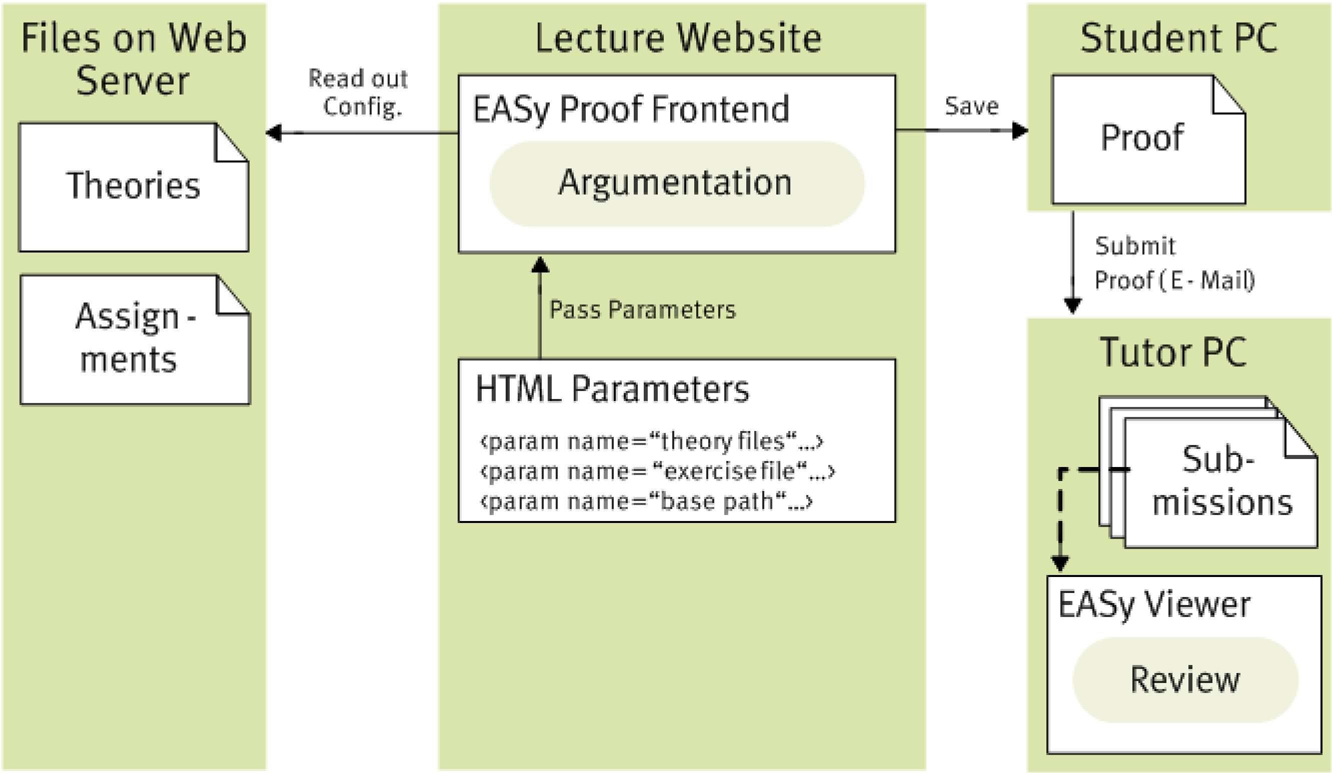


> The E-Assessment System EASy

- allows to establish a mathematical proof
  - step by step
  - by applying predefined rules and strategies
- a student cannot proof wrong theorems (but can get stuck)
- EASy implemented as Java applet
- based on term rewriting internally


EASy - Ausprobieren: Aufgabe 4 (Worst-Case-Abschätzung der binären Suche) - Mozilla Firefox
http://www.wi.uni-muenster.de/pi2/e-assessment/web/testen/aufgabe4.htm
Fenster schließen
Administration
Assignment / Theorem
Rule Base
Proof Strategies
Proof Visualisation
Explanation of Selected Rules
Term Selection
Aktionen
Beweis speichern
Beweis drucken
Beweis zurücksetzen
Letzte Strategie löschen
Schritt zurück
Ausführen
Strategie hinzufügen
Induktion
Boolesche Umformung
Es-Existiert Aussag
Fallunterscheidung
Aufspaltung der Folgerung
Abschätzungskette
Hinzufügen
Beweisübersicht
aufgabe_4
Induktion nach k
Boolesche Umformung
Kontext
Aktueller Kontext:
Verfügbare Theoreme:
context_ass3:
0<c
context_ass2:
0≤0
Status
Kontext
aufgabe_4
Zu zeigen:
Vorbedingungen:
ass2: 0≤k
ass1: k∈Z
ass3: 0<c
Folgerung:
tw(2^k)≤3·log(2^k)+c
Beweis abgeschlossen:Nein
Beweis
Induktion nach k
Induktionsanfang: aufgabe_4
Theorem
Induktionsanfang: aufgabe_4
Zu zeigen:
Vorbedingungen:
ass2: 0≤0
ass3: 0<c
Folgerung:
tw(2^0)≤3·log(2^0)+c
Beweis abgeschlossen:Nein
Beweis
Boolesche Umformung
Formen Sie den nachfolgenden Term boolesch in eine wahre Aussage um.
tw(2^0)≤3·log(2^0)+c
Induktionsschritt: aufgabe_4_induct_step
Theorem
Induktionsschritt: aufgabe_4_induct_step
Regelauswahl
Regeln
Aufgabe 4
Aufgabe 6
Aufgabe S2
Basis
Aussagenlogik
Evaluieren
Expandieren
Faktorisieren
Ausklammern
factorize
factorize_add1
factorize_mult
Anwenden
Regeldetails
Regel: factorize_add1
Regel
Experten
factorize_add1
Keine Beschreibung
Vorbedingungen: Keine
Folgerung:
x+x=2·x
Typ:Theorem
Termauswahl
tw(2^0)≤3·log(2^0)+c
tw(2^0)
3·log(2^0)+c
3·log(2^0)
c
Auswahl
tw(2^0)≤3·log(2^0)+c
Applet de.uni_muenster.easy.Applet started

> Using EASy for Proof Exercises

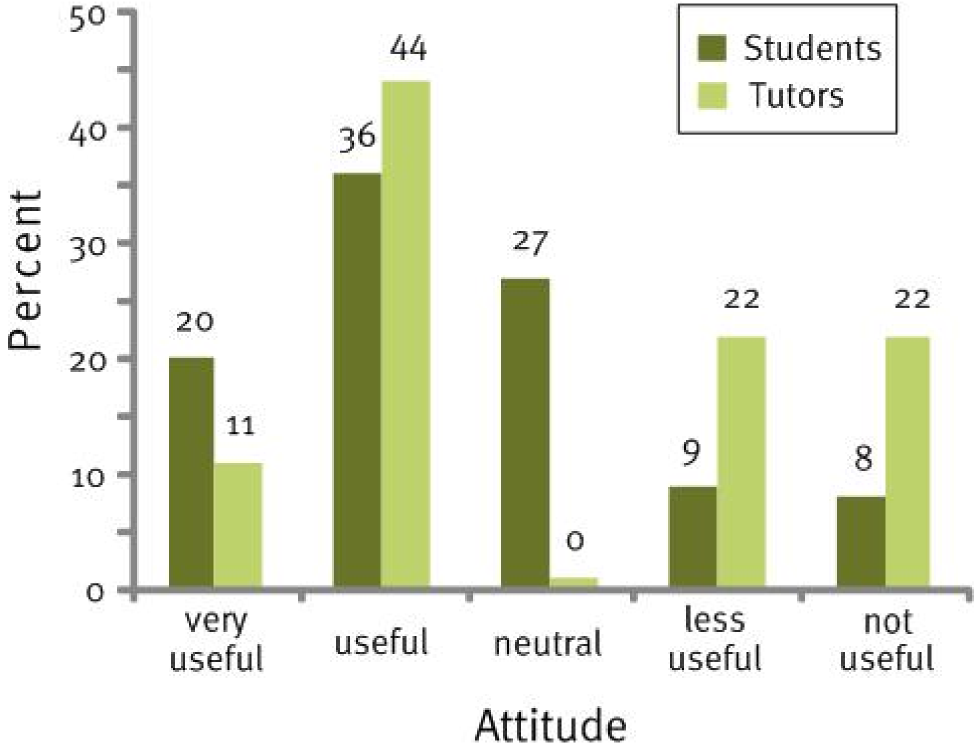


> Evaluation of EASy

- Application of EASy in „Data Structures and Algorithms“ (250 students):

	> Advantages	> Disadvantages
> Students	• provides proof strategies and applicable rules • controls correct use of rules • supports to learn proof structure	• familiarisation complex • first proofs time consuming • small steps: proof of obvious term conversions necessary • electronic proving vs. manual skills
> Tutors	• reduces correction effort significantly • easier to read and well-structured • enhances quality of submissions • facilitates distribution and collection of solutions	• none

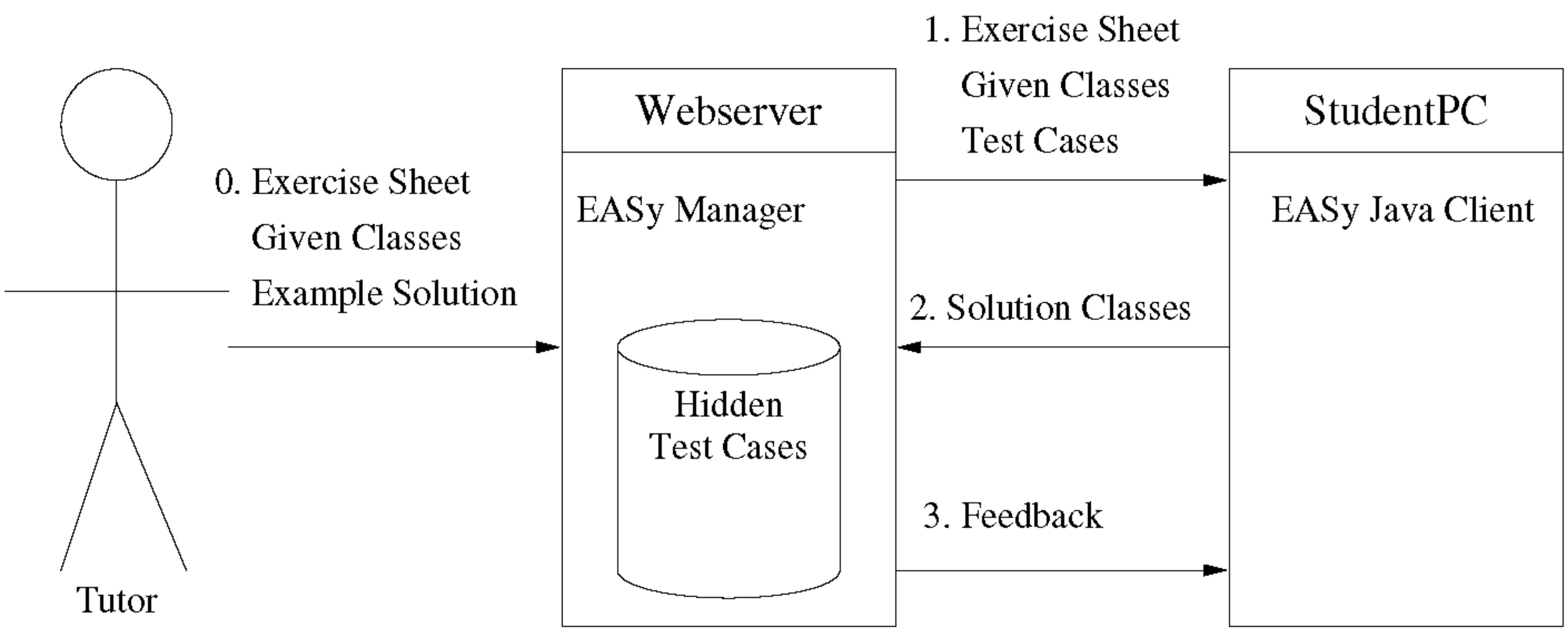
> Evaluation of EASy



> Agenda

- > Introduction
- > Assessment in Mathematics
- > The E-Assessment System EASy
- > **E-Assessment of Java Programs**
- > Conclusion

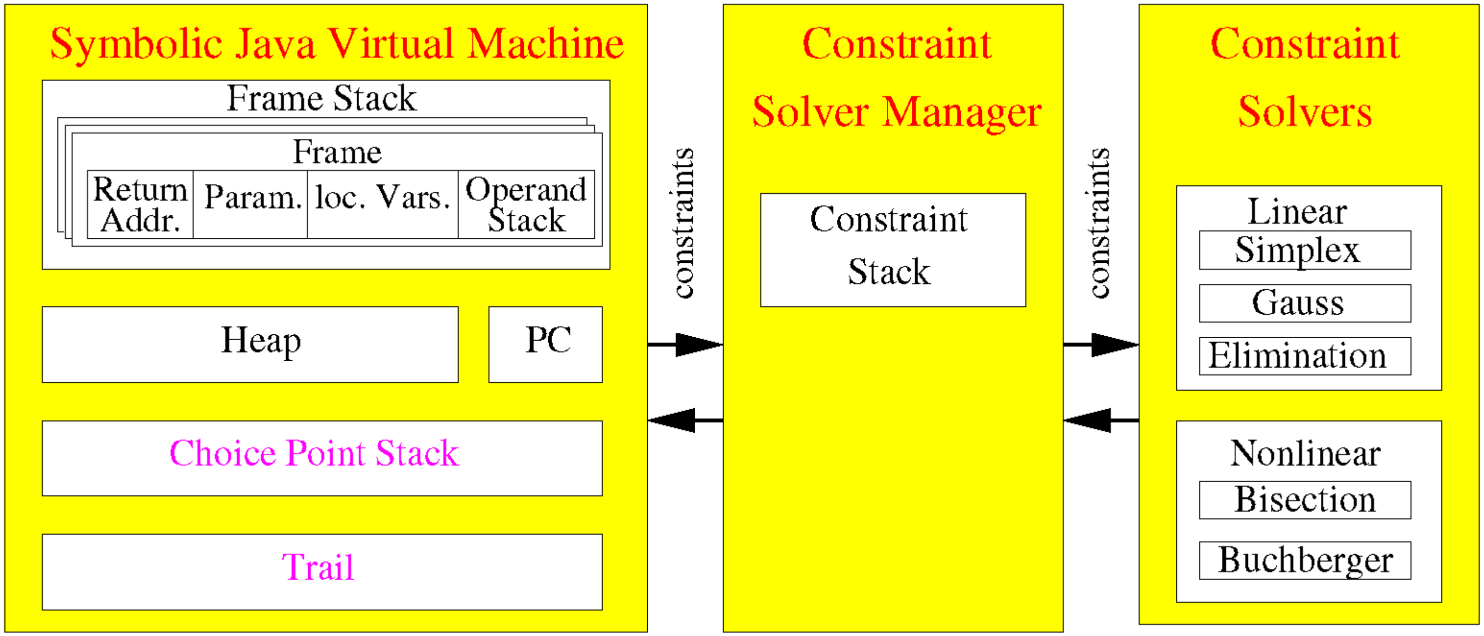
E-Assessment of Java Programs



E-Assessment of Java Programs

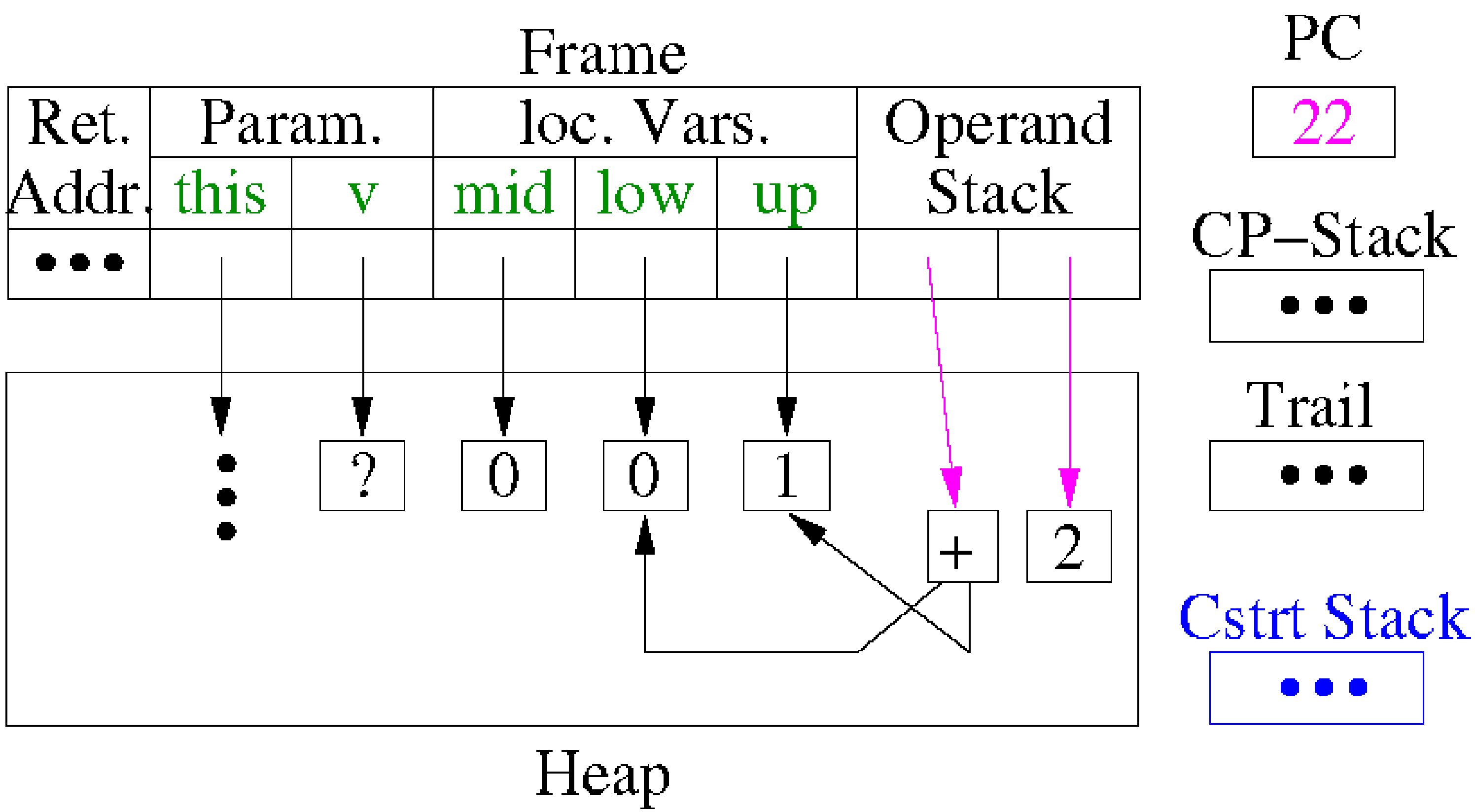
- the EASy Manager generates a set of test cases from the example solution
- it checks the uploaded classes of the student
  - syntactically
  - w.r.t. programming style
  - using the given and hidden test cases
- it gives feedback to the student

Automatic Generation of Test Cases

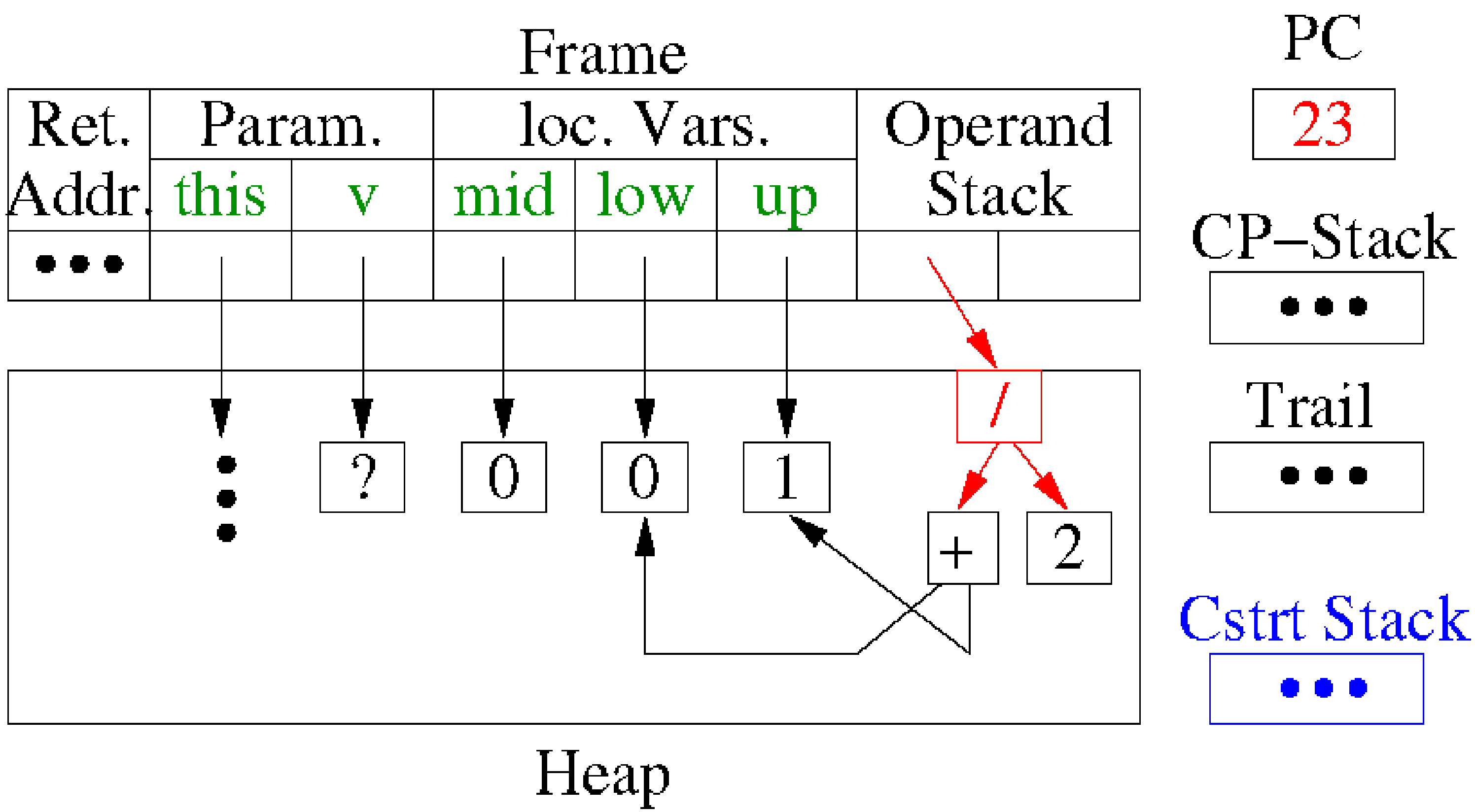


- Symbolic Execution of Java-Byte-Code by SJVM
- On branching instructions (e.g. if_cmpgt, invokevirtual):
  - Constraint solver determines remaining alternatives
  - Alternatives are handled by successively by backtracking mechanism

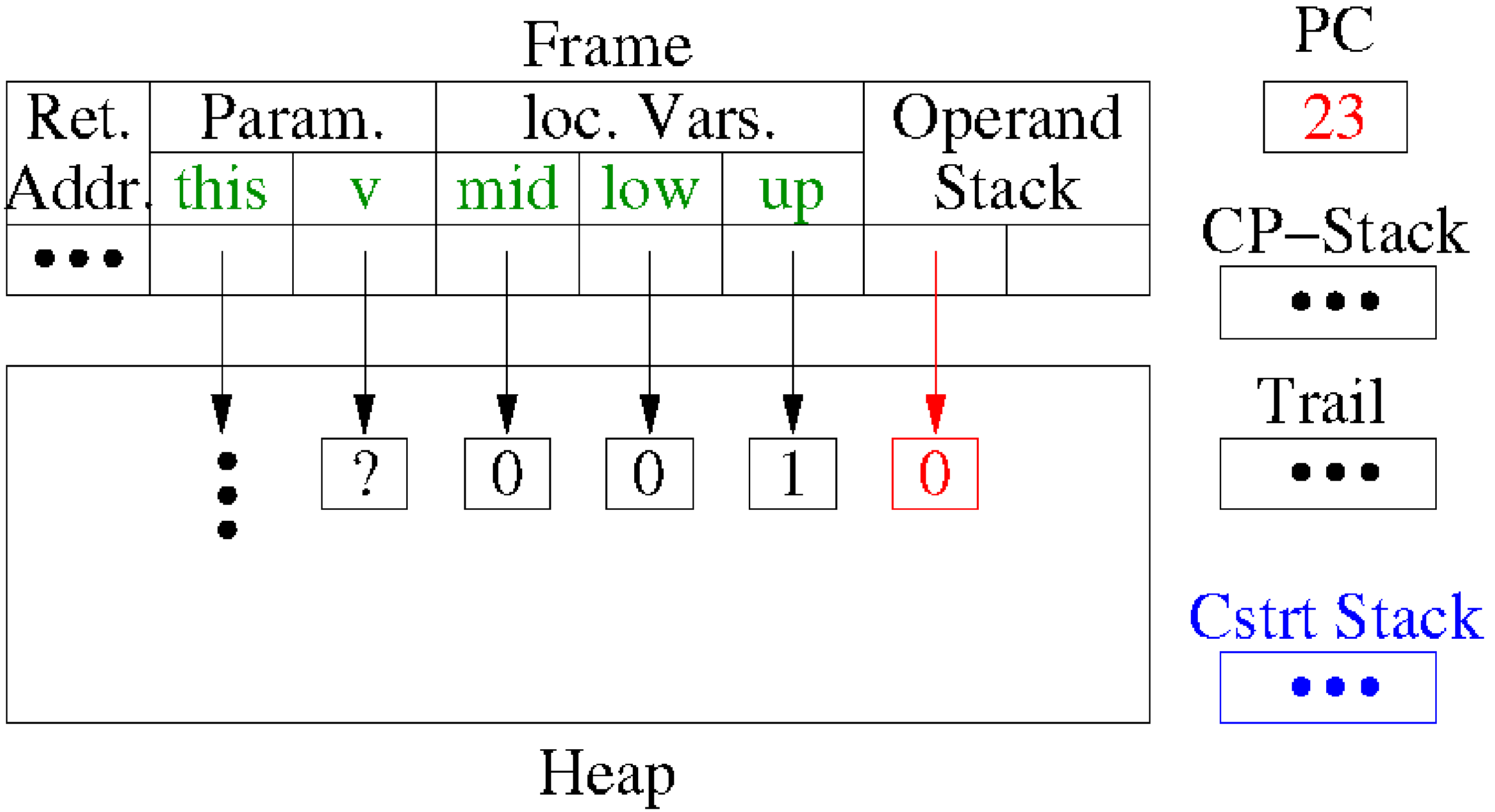
Symbolic Evaluation: idiv



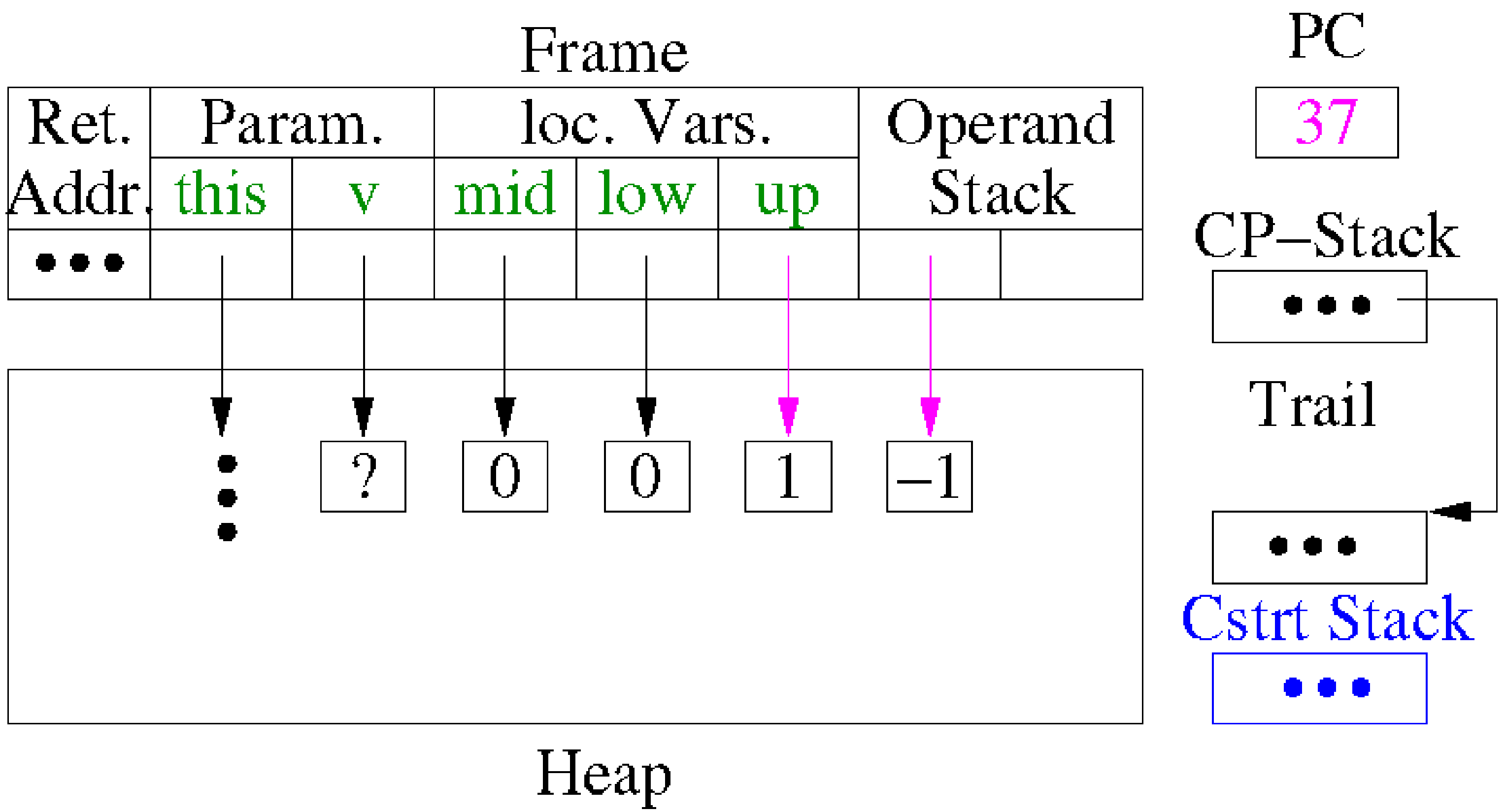
Symbolic Evaluation: idiv



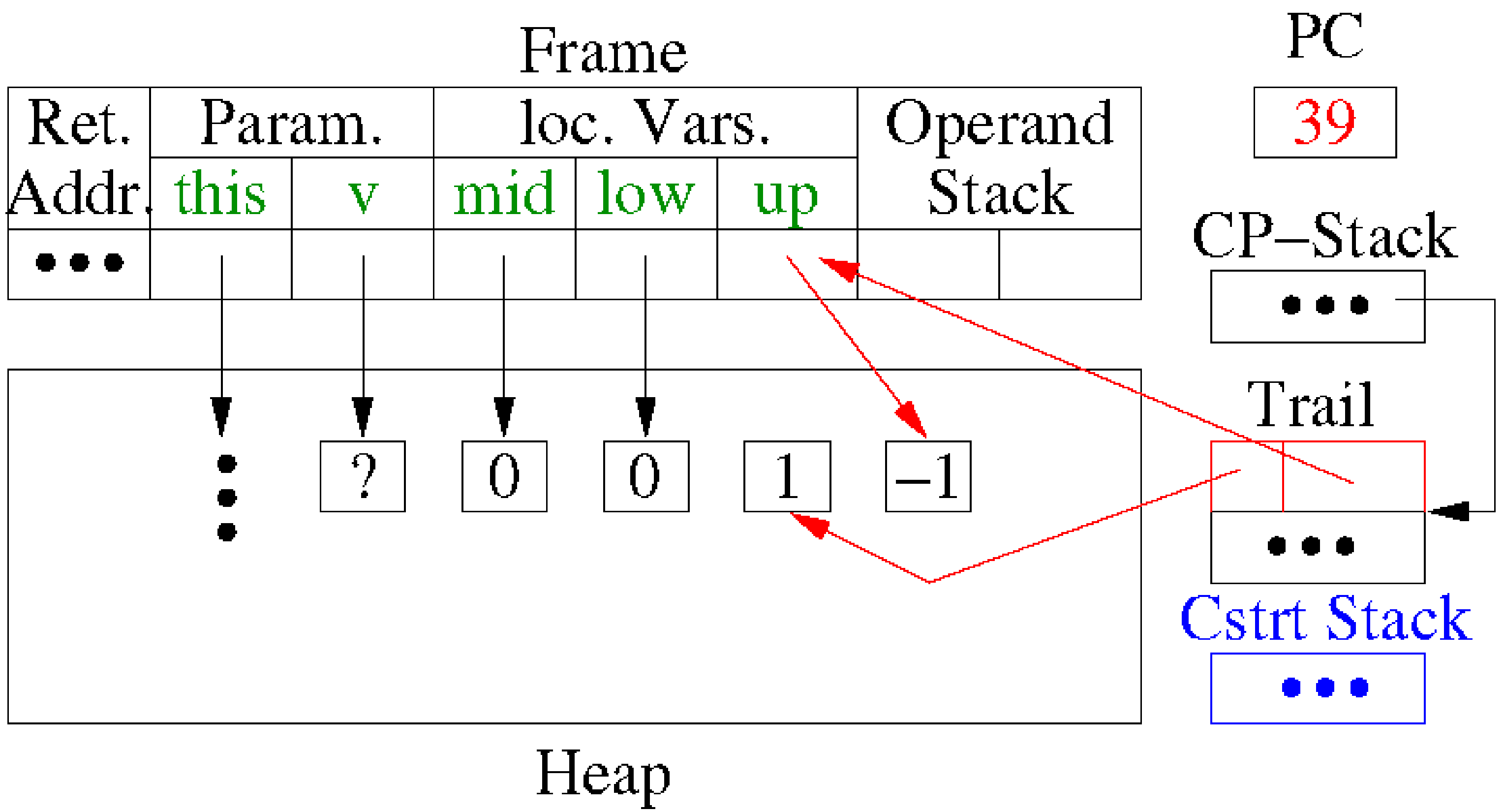
Symbolic Evaluation: idiv



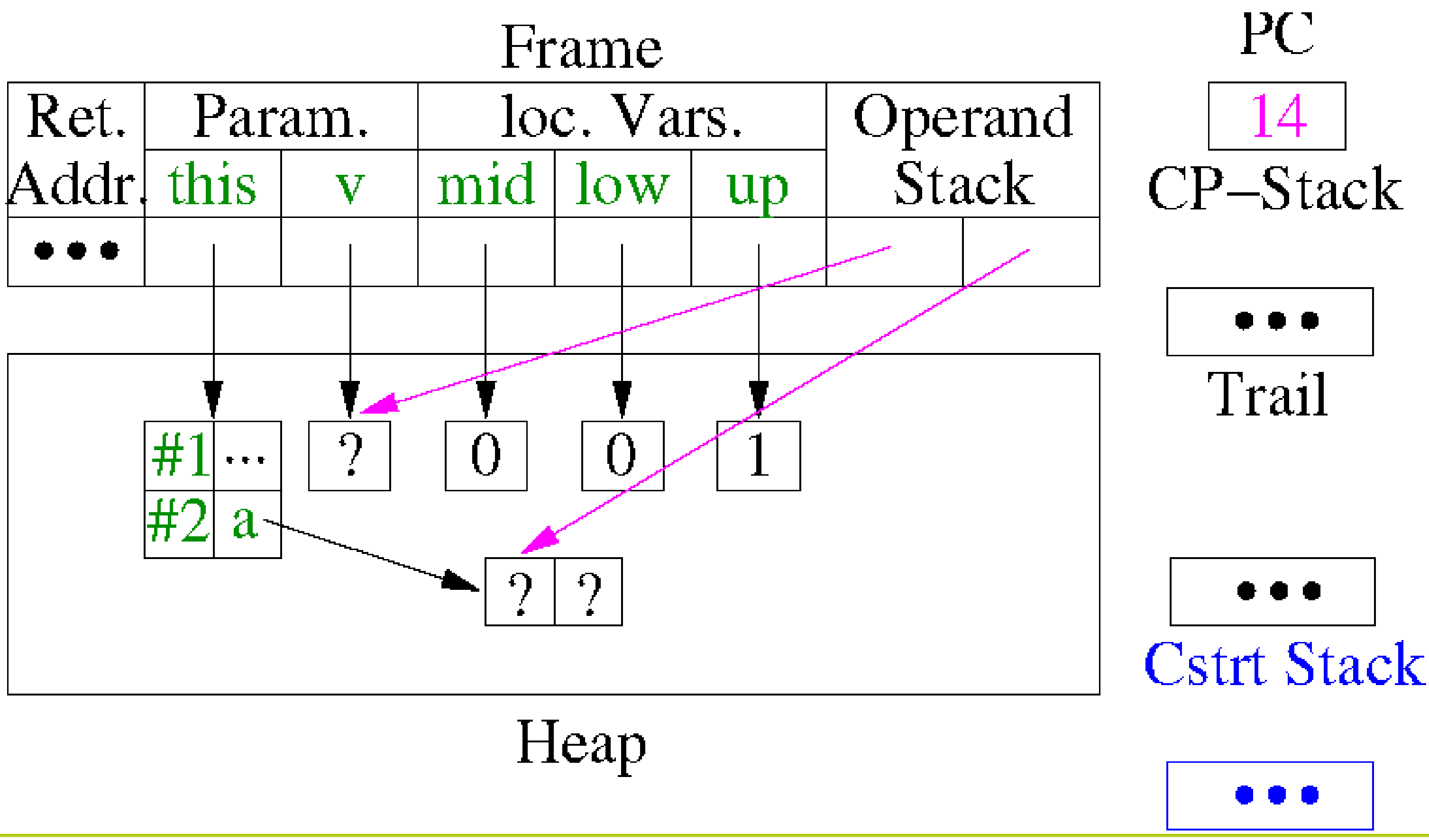
Saving Previous Values: istore 4



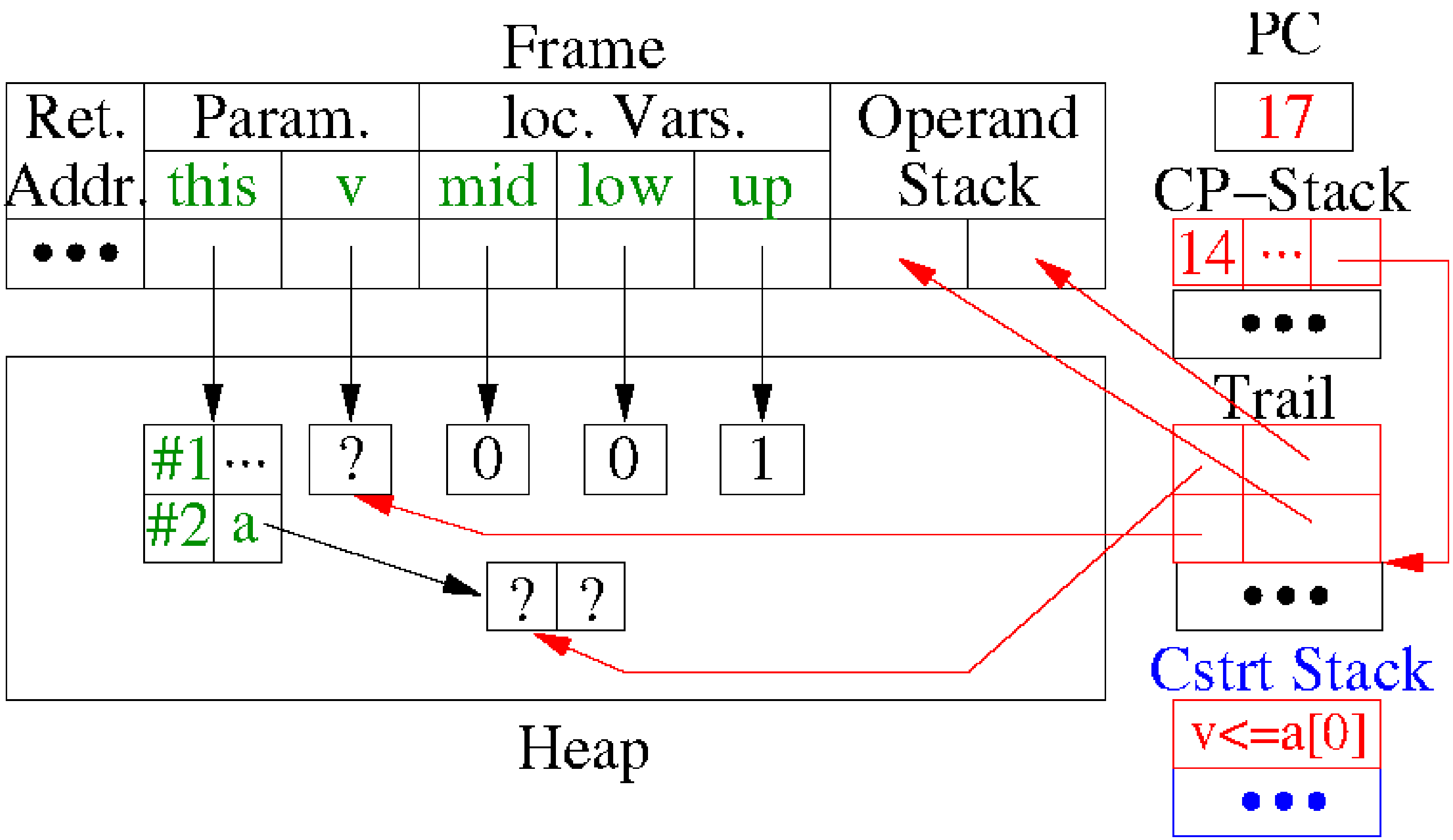
Saving Previous Values: istore 4



Branching: if_icmpgt 61



Branching: if_icmpgt 61



Constraint Solvers

- General requirements:
  - Incremental
  - Support backtracking
  - Allow to compute a concrete solution → test case
- Dual simplex algorithm + branch & cut algorithm for linear mixed-integer constraints
  - Uses special interval arithmetic to avoid rounding errors
- Numeric non-linear (bisection) solver
- Future work: add SMT solver

Bisection Solver

- Numerical approach to solution of polynomial equations
- Successive decomposition of solution space
- Each subspace R is checked for roots:
  - min { p(x) | x Є R} < 0   and   max{ p(x) | x Є R} > 0 ?
  - Yes: decompose R
  - No: discard R

Soundness and Completeness?

- Soundness due to interval arithmetic
- Completeness:
  - Theoretically impossible (→ halting problem)
  - No serious problem in practice

Which Constraints Appear?

Example	Type of Constraints			
	linear	non-linear	double	integer
Ackermann	√			√
Binary search	√			√
bubblesort	√			√
Bresenham	√			√
factorial	√			√
Gaussian eliminanation	√			√
GCD	√	√		√
histogram	√			√
Dijkstra	√			√
Matrix multiplication	√			√
Text search	√			√
log		√	√	
sin		√	√	
sqrt		√	√	

> Agenda

- > Introduction
- > Assessment in Mathematics
- > The E-Assessment System EASy
- > E-Assessment of Java Programs
- > Conclusion

> Conclusion

- development, application and evaluation of the EASy prototype
- e-assessments of mathematical proofs
- e-assessment of Java programs based on test cases generated from an example solution
- design, application and evaluation of a process model for formative e-assessment of proofs and Java programs