Komplexitätsanalyse mit Typen
Vor-/Nachwort

- Dieser Vortrag war an Schüler gerichtet, die sich für das Informatikstudium interessieren, und sollte einen groben Einblick in meine Arbeit geben.
- Folien 25–33 "Warteschlange" waren durch eine Demonstration begleitet, in der die Speicherungen von Objekten in Listen durch bunte Bauklötze in durchsichtigen, gestapelten Plastikbechern visualisiert wurde. Die Inferenz eines Systems linearer Gleichungen wurde mithilfe von farbigen Schüsseln (für Variablen) und Münzen darin (Wert der Variablen) dargestellt.
- Der gezeigte Programmcode ist in der funktionalen und frei verfügbaren Sprache Haskell verfasst, da dieser ohne besondere Vorkenntnisse leicht verständlich ist. Näheres siehe `http://www.Haskell.org`.
- Gegen Ende des Vortrages wurden noch folgende URLs zum Ausprobieren der Automatisierten Amortisierten Analyse genannt:

  `http://www.embounded.org/software/cost/cost.cgi`
  : Diese Implementation der Analyse kann nur lineare Schranken inferieren und scheitert sonst, kann aber mit Typen höherer Ordnung umgehen.

  `http://raml.tcs.ifi.lmu.de/prototype`
  : Diese besser poliert Implementation meiner Kollegen kann sehr gut polynomielle Schranken inferieren, ist aber auf Typen erster Ordnung beschränkt.

Komplexitätsanalyse mit Typen

Dr Steffen Jost

München

3. April 2012

Was kommt heraus?

$$2 + 3 =$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} =$$

$$0 == 1 \wedge 4 + 5 == 9 =$$

$$\text{"}\mathit{Hello}\text{"} \mathbin{++} \text{"}\mathit{World}!\text{"} =$$

$$3 * 4^{2012} =$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if 42 then } \text{"}\mathit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\mathit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} =$$

$$0 == 1 \wedge 4 + 5 == 9 =$$

$$\text{"}\mathit{Hello}\text{"} \mathbin{++} \text{"}\mathit{World}\text{!"} =$$

$$3 * 4^{2012} =$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if 42 then } \text{"}\mathit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\mathit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True}$$

$$0 == 1 \wedge 4 + 5 == 9 =$$

$$\text{"}\mathit{Hello}\text{"} \mathbin{++} \text{"}\mathit{World}\text{!"} =$$

$$3 * 4^{2012} =$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if 42 then "}\mathit{Richtig}\texttt{" else } 5^6 =$$

$$\text{"}\mathit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} =$$

$$3 * 4^{2012} =$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False}$$

$$\text{"}\mathit{Hello}\text{"} \mathbin{++} \text{"}\mathit{World}\text{!"} = \text{"}\mathit{Hello\ World}\text{!"}$$

$$3 * 4^{2012} =$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\mathit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\mathit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False}$$

$$\text{"}\mathit{Hello}\text{"} \mathbin{++} \text{"}\mathit{World}\text{!"} = \text{"}\mathit{Hello\ World}\text{!"}$$

$$3 * 4^{2012} = ?$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\mathit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\mathit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False}$$

$$\text{"}\mathit{Hello}\text{"} \mathbin{++} \text{"}\mathit{World}\text{!"} = \text{"}\mathit{Hello\ World}\text{!"}$$

$$3 * 4^{2012} = ? \qquad :: \mathit{Int}$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\mathit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\mathit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \mathit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False}$$

$$"\mathit{Hello}" \mathbin{++} "\mathit{World}!" = "\mathit{Hello\ World}!"$$

$$3 * 4^{2012} = ? \quad :: \mathit{Int}$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } "\mathit{Richtig}" \texttt{ else } 5^6 =$$

$$"\mathit{Hello}" * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}!\text{"} = \text{"}\textit{Hello World}!\text{"}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}!\text{"} = \text{"}\textit{Hello World}!\text{"}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} =$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if } 42 \texttt{ then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? \quad :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 =$$

$$\texttt{if 42 then "}\textit{Richtig}\texttt{" else } 5^6 =$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? \quad :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 = 12 \quad :: \textit{Int}$$

$$\texttt{if } 42 \texttt{ then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 =$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? \quad :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 = 12 \quad :: \textit{Int}$$

$$\texttt{if } 42 \texttt{ then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 = ?$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? \quad :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 = 12 \quad :: \textit{Int}$$

$$\texttt{if 42 then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 = ? \quad \lightning$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? \quad :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 = 12 \quad :: \textit{Int}$$

$$\texttt{if 42 then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 = ? \quad \lightning$$

$$\text{"}\textit{Hello}\text{"} * 3 =$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} +\!+ \text{"}\textit{World}!\text{"} = \text{"}\textit{Hello World}!\text{"} :: \text{String}$$

$$3 * 4^{2012} = ? :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 = 12 :: \textit{Int}$$

$$\texttt{if 42 then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 = ? \quad \lightning$$

$$\text{"}\textit{Hello}\text{"} * 3 = ? \quad \lightning$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was kommt heraus?

$$2 + 3 = 5 \quad :: \textit{Int}$$

$$\texttt{True oder False} = \texttt{True} \vee \texttt{False} = \texttt{True} \quad :: \textit{Bool}$$

$$0 == 1 \wedge 4 + 5 == 9 = \texttt{False} \quad :: \text{Bool}$$

$$\text{"}\textit{Hello}\text{"} \mathbin{++} \text{"}\textit{World}\text{!"} = \text{"}\textit{Hello World}\text{!"} \quad :: \text{String}$$

$$3 * 4^{2012} = ? \quad :: \textit{Int}$$

$$\sqrt{2012 * \pi} = ? \quad :: \textit{Float}$$

$$\texttt{if } 0 == 1 \texttt{ then } 1 + 2 \texttt{ else } 3 * 4 = 12 \quad :: \textit{Int}$$

$$\texttt{if 42 then } \text{"}\textit{Richtig}\text{"} \texttt{ else } 5^6 = ? \quad \lightning$$

$$\text{"}\textit{Hello}\text{"} * 3 = ? \quad \lightning$$

$\Rightarrow$ Kenn wir den Typ, so kennen wir schon recht viel!

Was sind Typen?

- Ein Typ ist eine Menge von Werten

  z.B. $\mathsf{Bool} = \{\texttt{True}, \texttt{False}\}, \quad \mathsf{Int} = \{1, 2, \dots\}$

- Ein Datenstruktur ist ein Typ und Funktionen darauf

  z.B. Listen mit Cons-Operation

- Viele Programmiersprachen erlauben eigene Typ-Definitionen

  z.B. Aufzählungen, Tupel, etc.

Wozu sind Typen gut?

Robin Milner (1934-2010):

"Well-typed programs do not go wrong."

Arten der Typprüfung:

Keine Schnell, aber Fehler werden nicht erkannt, was zu schweren Systemabstürzen führen kann

Assembler

Dynamisch Prüfung bei Ausführung — verlangsamt das Programm, aber Abbruch sobald Fehler passiert

Basic, Python, JavaScript

Statisch Verlangsamt nur Kompilieren, nicht Ausführung
Viele Fehler werden bereits vom Kompiler erkannt

Haskell, Scala, Java (teilw. dynamisch), C (unsicher)

- Typinferenz in vielen Sprachen möglich
- Typen dokumentieren

Wozu sind Typen gut?

Typurteil allgemein:

$$\Gamma \vdash e : A$$

"Programmausdruck e hat den Typ A in Kontext Γ"

Ein Kontext merkt sich die Typen von Variablen

$$\Gamma = \{i : \mathit{Int}, x : \mathit{Float}, s : \mathit{String}\}$$

Typregel-Beispiel:

$$\frac{\Gamma \vdash e_b : \mathit{Bool} \qquad \Gamma \vdash e_t : A \qquad \Gamma \vdash e_f : A}{\Gamma \vdash \texttt{if } e_b \texttt{ then } e_t \texttt{ else } e_f : A}\ (\textsc{Konditional})$$

Erkennen des Fehlers:

```
if 42 then "Richtig" else 5^6
```

Probleme: Int $\neq$ Bool, String $\neq$ Int

Wozu sind Typen gut?

Typurteil allgemein:

$$\Gamma \vdash e : A$$

"Programmausdruck e hat den Typ A in Kontext Γ"

Ein Kontext merkt sich die Typen von Variablen

$$\Gamma = \{i : \mathit{Int}, x : \mathit{Float}, s : \mathit{String}\}$$

Typregel-Beispiel:

$$\frac{\Gamma \vdash e_b : \mathit{Bool} \qquad \Gamma \vdash e_t : A \qquad \Gamma \vdash e_f : A}{\Gamma \vdash \texttt{if } e_b \texttt{ then } e_t \texttt{ else } e_f : A}\ (\textsc{Konditional})$$

Erkennen des Fehlers:

$$\texttt{if } \underbrace{42}_{\text{Int}} \texttt{ then } \underbrace{\text{"}\mathit{Richtig}\text{"}}_{\text{String}} \texttt{ else } \underbrace{5^6}_{\text{Int}}$$

Probleme: Int $\neq$ Bool, String $\neq$ Int

Wozu sind Typen gut?

Typurteil allgemein:

$$\Gamma \vdash e : A$$

"Programmausdruck e hat den Typ A in Kontext Γ"

Ein Kontext merkt sich die Typen von Variablen

$$\Gamma = \{i : \mathit{Int}, x : \mathit{Float}, s : \mathit{String}\}$$

Typregel-Beispiel:

$$\frac{\Gamma \vdash e_b : \mathit{Bool} \qquad \Gamma \vdash e_t : A \qquad \Gamma \vdash e_f : A}{\Gamma \vdash \texttt{if } e_b \texttt{ then } e_t \texttt{ else } e_f : A}\ (\textsc{Konditional})$$

Erkennen des Fehlers:

$$\texttt{if } \underbrace{42}_{\text{Int}} \texttt{ then } \underbrace{\text{"}\mathit{Richtig}\text{"}}_{\text{String}} \texttt{ else } \underbrace{5^6}_{\text{Int}}$$

Probleme: Int $\neq$ Bool, String $\neq$ Int

Listen

- geordnete Folge von Dingen gleichen Typs

  $[1,2,3], ['H','i','!'], [True, False, False], [[1,2],[3,4,5]]$
- Listen haben beliebige (endliche) Längen
- Erlauben Verarbeitung unbekannt vieler Daten

  z.B. Klausuranmeldungen, Kontaktdatenbank im Handy
- Nur der Kopf der Liste kann bearbeitet werden
  - Neues Element vorne hinzufügen
  - Kopf abtrennen

Notation

- Leere Liste [] (oder auch `Nil`)
- "Cons"-Operation : hat den Typ $A \to [A] \to [A]$

$$[1,2,3] = 1 : [2,3] == 1 : (2 : [3]) = 1 : 2 : 3 : []$$

Beispiel: Listen-Länge berechnen

```
length :: [a] -> Int
length  []    = 0
length (h:t) = 1 + length t
```

www.Haskell.org

Warteschlange

- Kann beliebig viele Objekte speichern (wie Liste)
- Zuerst herein — zuerst heraus (FIFO)
- Puffer für Objekte

Zwei Operationen:

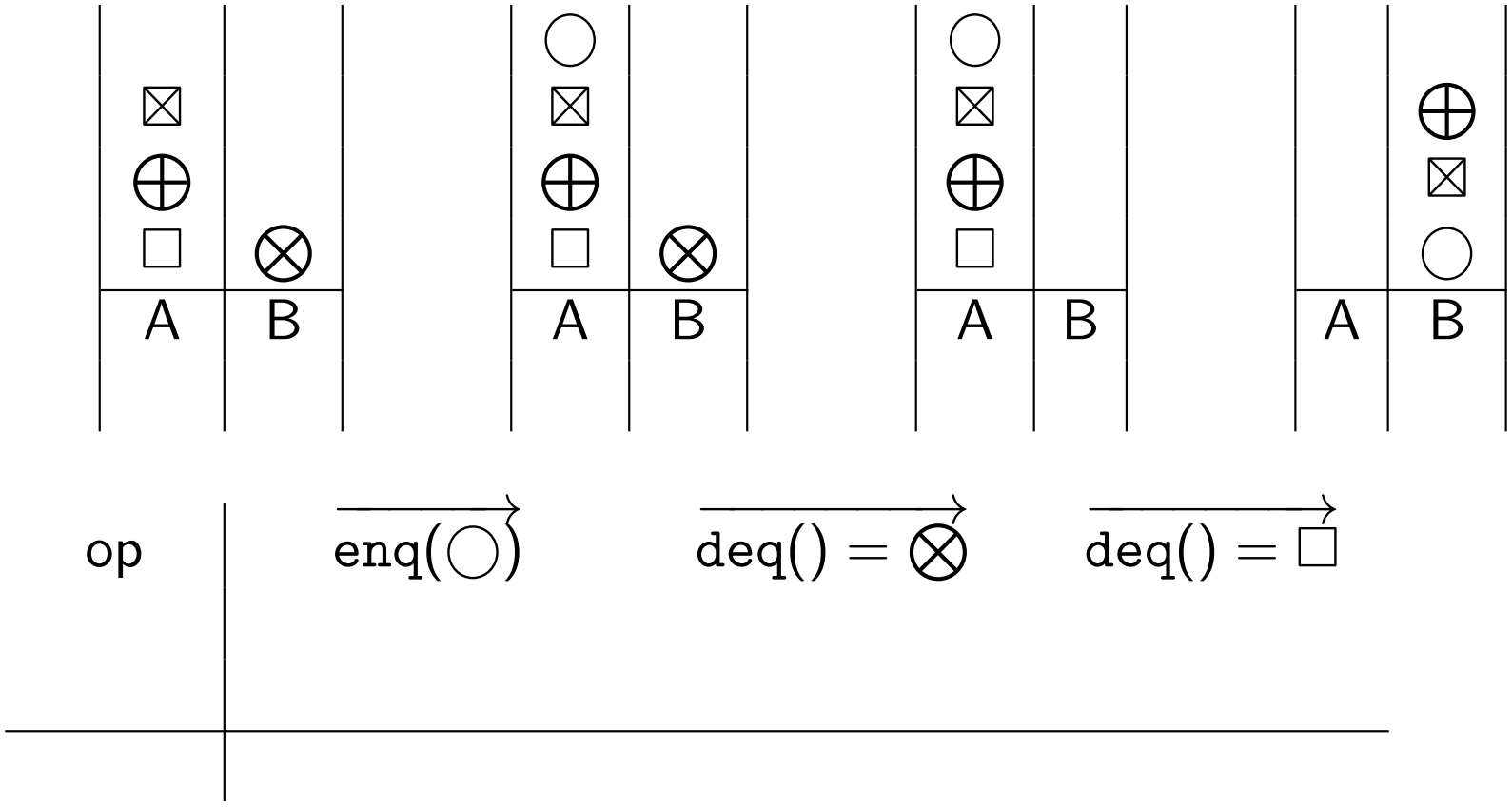
```
enq :: a -> Queue a -> Queue a

deq :: Queue a -> (a, Queue a)
```

Kann mithilfe von Listen implementiert werden.

Wie?

Warteschlange



Warteschlange

```
data Queue a = Queue [a] [a]

enq :: a -> Queue a -> Queue a
enq x (Queue ins outs)   = Queue (x:ins) outs

deq :: Queue a -> (a, Queue a)
deq (Queue ins (o:outs)) = (o, Queue ins outs )
deq (Queue ins      [] ) = (h, Queue [] hs)
  where
    h:hs = umdrehen ins

umdrehen :: [a] -> [a]
umdrehen l =  rev l []
  where
    rev []     a = a
    rev (x:xs) a = rev xs (x:a)
```

Begriffsklärung

Komplexitätstheorie

Wie schwierig ist es, ein konkretes Problem zu Lösen?
(Wieviel Zeit oder Speicherplatz wird zur Lösung benötigt?)

- Konstante Faktoren irrelevant

Komplexitätsanalyse eines konkreten Programmes

Wie gut/schnell löst unser Programm ein Problem?

- Konstante Faktoren relevant

Lineares Gleichungssytem

e – Kosten von `enq`
d – Kosten von `deq`

a – Münzen in der A-Liste
b – Münzen in der B-Liste

$$e = 1 + a$$
$$d + b = 1$$
$$a = 1 + b$$

und $e \geq 0$, $d \geq 0$, $a \geq 0$, $b \geq 0$

Lineares Gleichungssytem

e – Kosten von `enq`
d – Kosten von `deq`

a – Münzen in der A-Liste
b – Münzen in der B-Liste

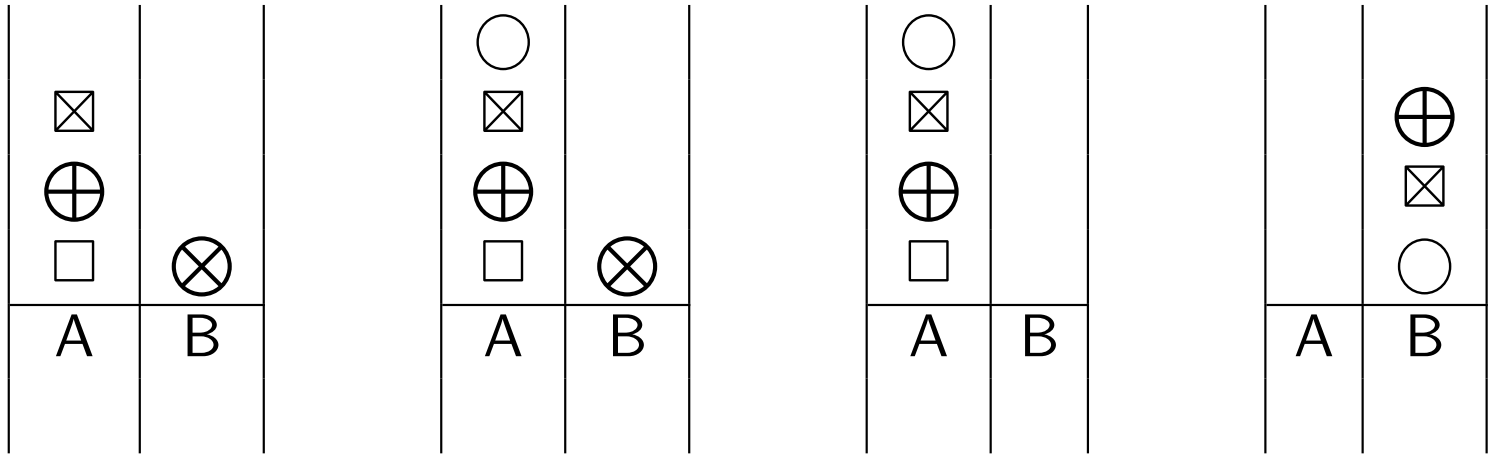
$$e = 1 + a \qquad d + b = 1 \qquad a = 1 + b$$

$$3 = 1 + 2 \qquad 0 + 1 = 1 \qquad 2 = 1 + 1$$

$$2 = 1 + 1 \qquad 1 + 0 = 1 \qquad 1 = 1 + 0$$

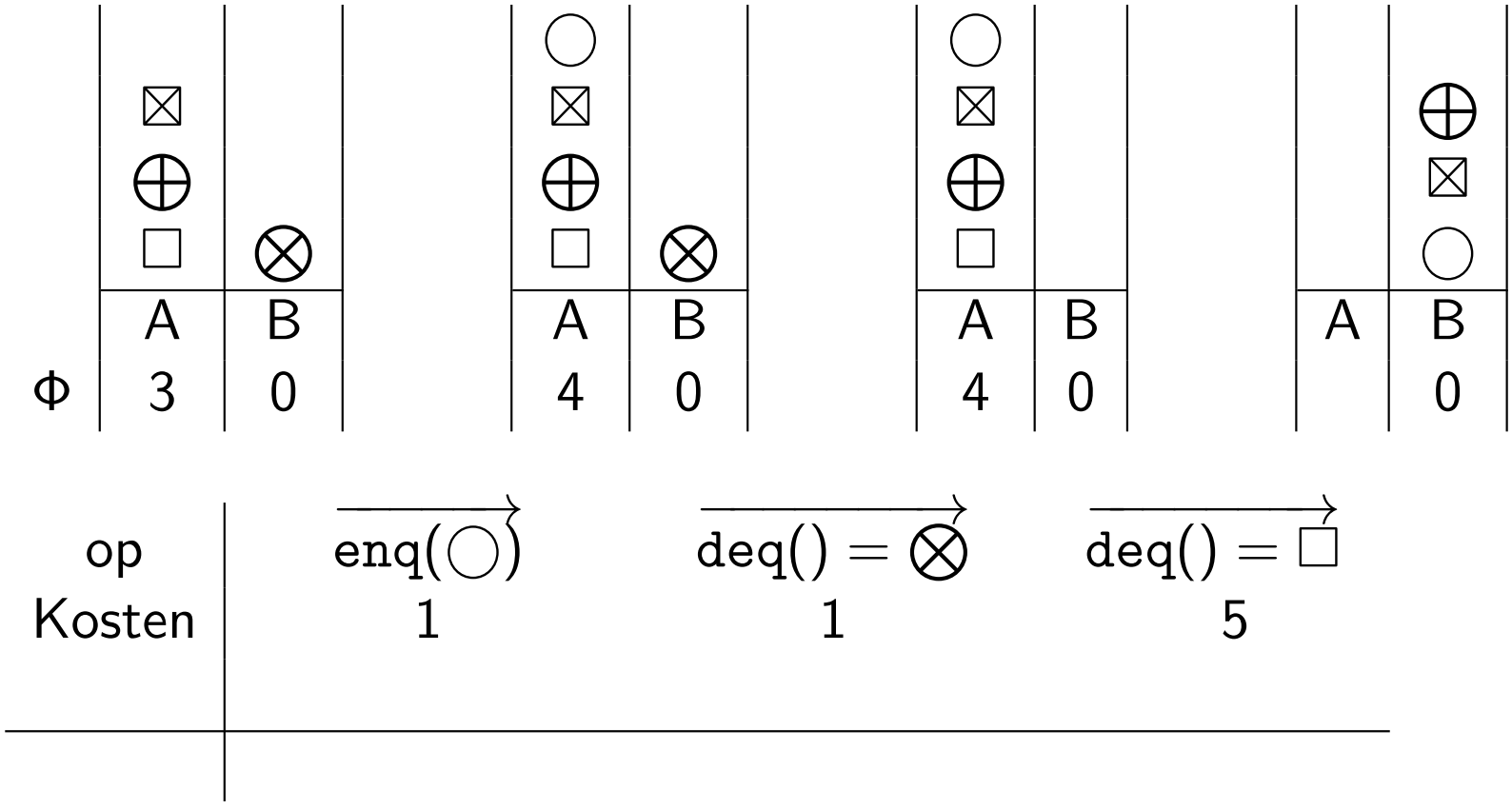
und $e \geq 0$, $d \geq 0$, $a \geq 0$, $b \geq 0$

Warteschlange



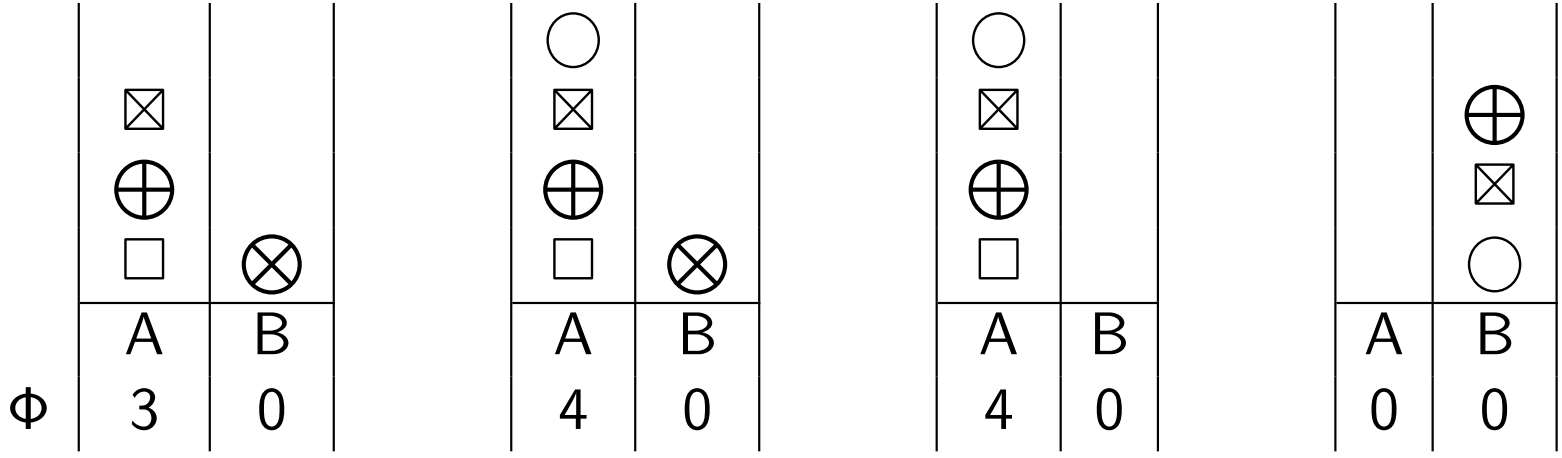
op	$\overrightarrow{\texttt{enq}(\bigcirc)}$	$\overrightarrow{\texttt{deq}() = \otimes}$	$\overrightarrow{\texttt{deq}() = \square}$
Kosten	1	1	5

Warteschlange



- Φ = Summe der "Münzen" in dieser Liste
- Konstante Kosten — macht Berechnen der Gesamtkosten einer Sequenz von Warteschlangen-Operationen viel leichter

Warteschlange



op	$\overrightarrow{\texttt{enq}(\bigcirc)}$	$\overrightarrow{\texttt{deq}() = \otimes}$	$\overrightarrow{\texttt{deq}() = \square}$
Kosten	1	1	5
$\Delta\Phi$	1	0	-4
Σ	2	1	1

- Φ = Summe der "Münzen" in dieser Liste
- Konstante Kosten — macht Berechnen der Gesamtkosten einer Sequenz von Warteschlangen-Operationen viel leichter

Prinzip: Amortisierte Analyse

- Zustand zu Zahl reduzieren – *Potential* $\Phi(\text{Zustand})$
- Tatsächliche Kosten mit Potential bezahlen

  $$\textit{Kosten der Zustandsänderung} \leq \Phi(\textit{vorher}) - \Phi(\textit{nacher})$$
- Gesamtkosten = Potential der Eingabe $\Phi(\textit{Anfang})$

Kosten werden *früher* berechnet als sie anfallen

Grundidee 1985 gefunden von R.E. Tarjan,
erweitert 1998 von C. Okasaki

Erweiterten durch uns um

- Automatisierung vorher nur manuelle Anwendung
- Potential pro Referenz ohne Referenzen-Zählen

Erweiterte Typisierung + Lineare Gleichnungen = Automatische Analyse

Prinzip: Amortisierte Analyse

- Zustand zu Zahl reduzieren – *Potential* $\Phi(\text{Zustand})$
- Tatsächliche Kosten mit Potential bezahlen

  $$\textit{Kosten der Zustandsänderung} \leq \Phi(\textit{vorher}) - \Phi(\textit{nacher})$$
- Gesamtkosten = Potential der Eingabe $\Phi(\textit{Anfang})$

Kosten werden *früher* berechnet als sie anfallen

Grundidee 1985 gefunden von R.E. Tarjan,
erweitert 1998 von C. Okasaki

Erweiterten durch uns um

- Automatisierung vorher nur manuelle Anwendung
- Potential pro Referenz ohne Referenzen-Zählen

Erweiterte Typisierung + Lineare Gleichnungen = Automatische Analyse

Prinzip: Amortisierte Analyse

- Zustand zu Zahl reduzieren – *Potential* $\Phi(\text{Zustand})$
- Tatsächliche Kosten mit Potential bezahlen

  $$\textit{Kosten der Zustandsänderung} \leq \Phi(\textit{vorher}) - \Phi(\textit{nacher})$$
- Gesamtkosten = Potential der Eingabe $\Phi(\textit{Anfang})$

Kosten werden *früher* berechnet als sie anfallen

Grundidee 1985 gefunden von R.E. Tarjan,
erweitert 1998 von C. Okasaki

Erweiterten durch uns um

- Automatisierung — vorher nur manuelle Anwendung
- Potential pro Referenz — ohne Referenzen-Zählen

Erweiterte Typisierung + Lineare Gleichnungen = Automatische Analyse

Prinzip: Amortisierte Analyse

- Zustand zu Zahl reduzieren – *Potential* $\Phi(\text{Zustand})$
- Tatsächliche Kosten mit Potential bezahlen

  $$\textit{Kosten der Zustandsänderung} \leq \Phi(\textit{vorher}) - \Phi(\textit{nacher})$$
- Gesamtkosten = Potential der Eingabe $\Phi(\textit{Anfang})$

Kosten werden *früher* berechnet als sie anfallen

Grundidee 1985 gefunden von R.E. Tarjan,
erweitert 1998 von C. Okasaki

Erweiterten durch uns um

- Automatisierung vorher nur manuelle Anwendung
- Potential pro Referenz ohne Referenzen-Zählen

Erweiterte Typisierung + Lineare Gleichnungen = Automatische Analyse

Prinzip: Amortisierte Analyse

- Zustand zu Zahl reduzieren – *Potential* $\Phi(\text{Zustand})$
- Tatsächliche Kosten mit Potential bezahlen

  $$\textit{Kosten der Zustandsänderung} \leq \Phi(\textit{vorher}) - \Phi(\textit{nacher})$$
- Gesamtkosten = Potential der Eingabe $\Phi(\textit{Anfang})$

Kosten werden *früher* berechnet als sie anfallen

Grundidee 1985 gefunden von R.E. Tarjan,
erweitert 1998 von C. Okasaki

Erweiterten durch uns um

- Automatisierung — vorher nur manuelle Anwendung
- Potential pro Referenz — ohne Referenzen-Zählen

Erweiterte Typisierung + Lineare Gleichnungen = Automatische Analyse

Automatisierung wichtig

Anwendung formaler Methoden teuer

— in der Praxis nur selten benutzt

Allerdings:

- Testen wird immer schwieriger

  App-Stores liefern Apps für viele unterschiedliche Geräte
- Kunden werden immer schwieriger

  Handynutzer akzeptieren kein Bluescreen
- (Eingebettete) Software überall

Automatisierung = Einfach/Billig = Weite Verbreitung

Typbasierte Analyse (ohne Bauklötze) Folie aus einem Konferenzvortrag

$$A ::= \quad \mathsf{unit} \mid \mathsf{bool} \mid A \times A \mid (q; A) + (r; A) \mid \mathsf{list}(q; A)$$
$$\mid A^{q \triangleright q'} \& B^{r \triangleright r'} \mid \forall \alpha \in \psi. A \xrightarrow{q \triangleright q'} A$$

with $q, q', r, r' \in \mathrm{RV}$; $\alpha \subset \mathrm{RV}$; ϕ set of linear constraints over RV

$$\frac{\phi = \{p = q + \textsc{TypSize}\big(A \times \mathsf{list}(q; A)\big),\ p' = 0\}}{x_h{:}A,\ x_t{:}\mathsf{list}(q; A) \ \vdash^{p}_{p'}\ \mathtt{Cons}(x_h, x_t) : \mathsf{list}(q; A) \mid \phi}$$
$$(\textsc{Arthur} \vdash \textsc{List-Cons})$$

$$\frac{\begin{array}{c}\Gamma \vdash^{p}_{p'} e_1 : C \mid \phi \qquad \xi = \big\{p_0 = p + q + \textsc{TypSize}(A \times \mathsf{list}(q; A))\big\} \\ \Gamma,\ x_h{:}A,\ x_t{:}\mathsf{list}(q; A) \ \vdash^{p_0}_{p'}\ e_2 : C \mid \psi\end{array}}{\begin{array}{l}\Gamma,\ x{:}\mathsf{list}(q; A) \ \vdash^{p}_{p'}\ \mathtt{match!}\ x\ \mathtt{with} \mid \mathtt{Nil} \to e_1 \\ \qquad\qquad\qquad\qquad\qquad\qquad \mid \mathtt{Cons}(x_h, x_t) \to e_2\end{array} : C \mid \phi \cup \psi \cup \xi}$$
$$(\textsc{Arthur} \vdash \textsc{List-Elim}!)$$

Funktion Abstraktion und Applikation

Folie aus einem Konferenzvortrag

$$\frac{\mathrm{dom}(\Gamma) = \mathrm{FV}(e) \setminus \{x, y\} \qquad \alpha \cap \left(\mathrm{FV}_\diamond(\Gamma) \cup \mathrm{FV}_\diamond(\phi)\right) = \emptyset \qquad \Gamma, x{:}A, y{:}\forall\alpha \in \psi.A \xrightarrow{q \triangleright q'} B \vdash^{q}_{q'} e : B \mid \xi \qquad \phi \cup \psi \models \xi \qquad \phi \models \{p = 1 + \textsc{TypSize}(\Gamma),\ p' = 0\} \cup \bigcup_{D \in \mathrm{ran}(\Gamma)} \curlyvee(D \,|\, D, D)}{\Gamma \vdash^{p}_{p'} \texttt{rec}\ y\ \texttt{=}\ \texttt{fun}\ x \to e : \forall\alpha \in \psi.A \xrightarrow{q \triangleright q'} B \mid \phi}\ (\textsc{Arthur}\vdash\textsc{Rec})$$

- predictable (minimal) closure size
- ensure newly bound variables are independent
- split constraints of ξ to delayed ψ and applied ϕ constraints
- allow repeated function application by zero closure potential

$$\frac{\sigma(B) = A \xrightarrow{r \triangleright r'} C \qquad \sigma : \alpha \to \text{RV a substitution to fresh resource variables}}{x{:}A,\ y{:}\forall\alpha \in \psi.B \vdash^{q}_{q'} y\,x : C \mid \sigma(\psi) \cup \{r = q,\ r' = q'\}}\ (\textsc{Arthur}\vdash\textsc{App})$$

Beweis der Korrektheit (vereinfacht)

WENN term e wohl-typisiert $\quad \Gamma \vdash^{q}_{q'} e : A \mid \psi$

v löst ψ $\quad v \vDash \psi$

und e wertet aus zu ℓ $\quad \mathcal{S}, \mathcal{H} \vdash e \downarrow \ell, \mathcal{H}'$

im konsistenten Anfangszustand $\quad \mathcal{H} \vDash \mathcal{S} :: \Gamma$

DANN

für alle $p \in \mathbb{N}$ mit $\quad p \geq v(q) + \Phi_{\mathcal{H}}\big(\mathcal{S}{:}v(\Gamma)\big)$

gibt es ein $p' \in \mathbb{N}$ mit $\quad p' \geq v(q') + \Phi_{\mathcal{H}'}\big(\ell{:}v(A)\big)$

SO DASS

e mit eingeschränkten Ressourcen auswertet $\quad \mathcal{S}, \mathcal{H} \vdash^{p}_{p'} e \downarrow \ell, \mathcal{H}'$

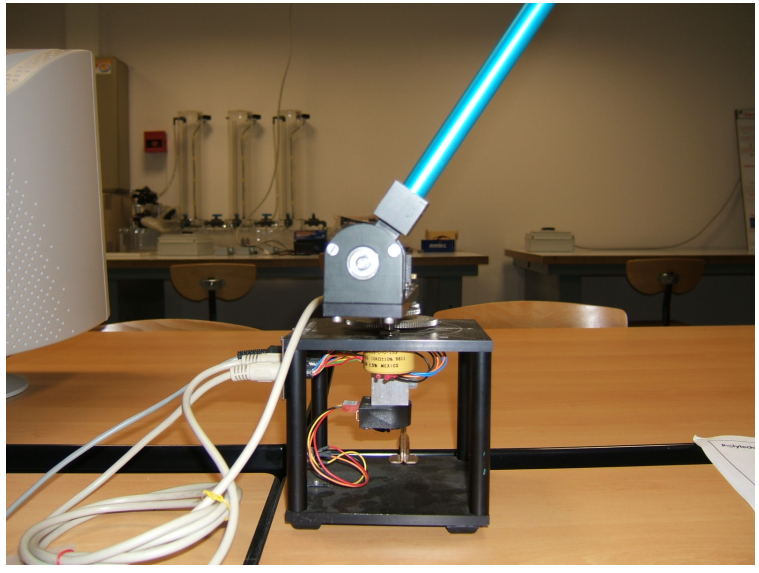
zu einem konsistenten Ergebnis $\quad \mathcal{H}' \vDash \ell :: A$

$p, p' \in \mathbb{N}$, aber $q, q' \in \mathbb{Q}^+$

Beispiel: Umgedrehtes Pendel

www. EmBounded .org

- Echtzeit-Regelung Problem (∼190 lines)
- Durchgeführt und gemessen mit Renesas M32C/85U



- Ergebnis:
  - 36118 to 47635 Taktzyklen pro Schleife gemessen
  - 63678 Taktzyklen als oberere Schranke inferiert 33.7%
  - *Stack Speicher:* Exakte Vorhersage!
  - *Dynamischer Speicher:* Exakte Vorhersage!
- 1115 lineare Gleichungen über 2214 Variablen erzeugt in 0.41s, gelöst in 0.26s mit 1.73Ghz Pentium M, 2MB cache

Verständliche Ergebnisse

Beispiel: `RBinsert: int, rbtree -> rbtree`

Ergebnis der Analyse:

```
ARTHUR3 typing for HumeHeapBoxed:
(int,rbtree[Leaf|Node<10>:colour[Red|Black<18>],#,int,#])
   -(20/0)->   rbtree[Leaf|Node:colour[Red|Black],#,int,#]
```

Automatische Übersetzung in natürliche Sprache:

```
Worst-case Heap-units required to call RBinsert:
  20 + 10*X1 + 18*X2
    where
        X1 = number of "Node" nodes at 1. position
        X2 = number of "Black" nodes at 1. position
```

Bemerke: Daten-Abhängige Schranken berechnet, nicht nur Größen-Abghängig

Forschungsergebnisse

Past

LFPL to malloc-free C	Hofmann, Nordic'00
Heap Usage for First-Order Language	Hofmann & Jost, POPL'03
Java & Storeless Semantics	Hofmann & Jost, ESOP'06
Java Automated Type-Checking	Hofmann & Rodriguez, EACSL'09
Stack Space Usage & Depth	Campbell, ESOP'09
WCET, Algebraic Datatypes & Cost Genericity	Jost et al., FM'09
Higher-order & Polymorphism	Jost et al., POPL'10
Polynomial Bounds	Hofmann & Hoffmann, ESOP'10

Future

Lazy Evaluation	Simões et al.
Combination with Sized Types	
Negative Potential	
Potential for Numeric Types	

Zusammenfassung

- Mathematik/Theoretische Informatik ist toll!
- Typen sind toll!
- Automatische Analysen sind toll
  — wenn deren Korrektheit formell bewiesen wurde