Universität Karlsruhe (TH)
Institut für Programmstrukturen
und Datenorganisation

Lehrstuhl Professor Goos

Erweiterung eines statischen Übersetzers zu einem Laufzeitübersetzungssystem

Marcus Denker

Diplomarbeit

Verantwortlicher Betreuer: Prof. Dr. Gerhard Goos

Betreuender Mitarbeiter: Dipl.-Inform. Florian Liekweg

Aufgabe

Entwicklung eines Laufzeitübersetzers aus einem statischen Übersetzer

Ziele:

- Vollständiger Entwicklungszyklus unterstützt
    - Übersetzung von Quellcode
    - Neuübersetzung von bereits übersetzen Code

- Der Übersetzer erlaubt weitreichende Optimierungen

- Integration in ein vorhandenes System

Squeak3.7a-5526.image

Überblick Vortrag

1. Entwurf
    - SSA Abbau genauer

2. Verwendete Infrastruktur

3. Umsetzung

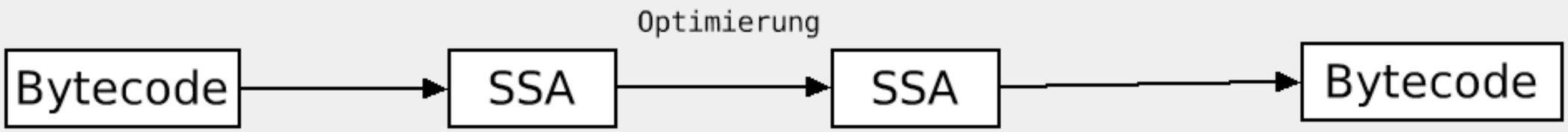
4. Ergebnisse

5. Ausblick

Entwurf

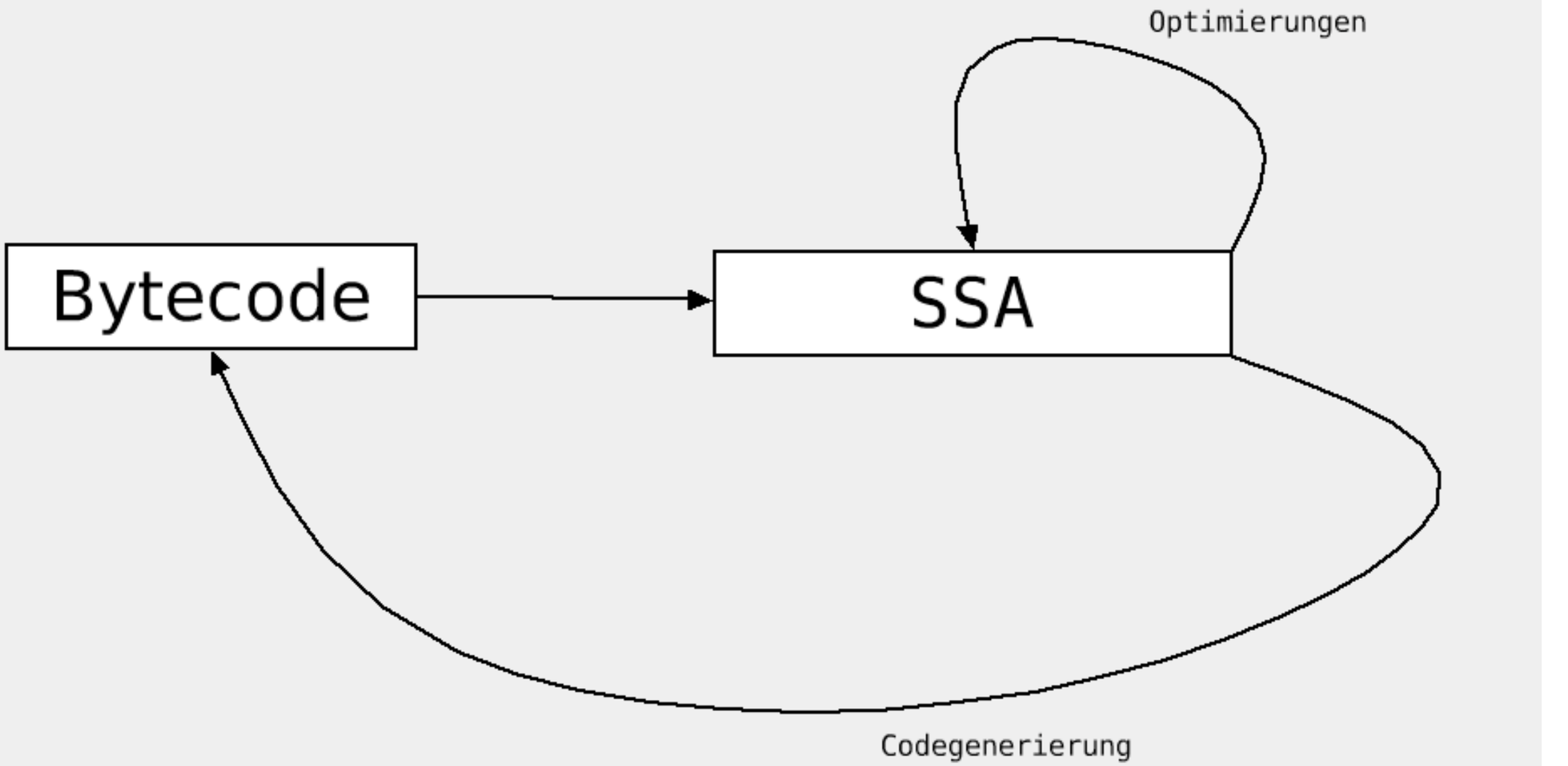
Statischer Übersetzer:

Bytecode als Datenbasis:



Squeak3.7a-5526.image

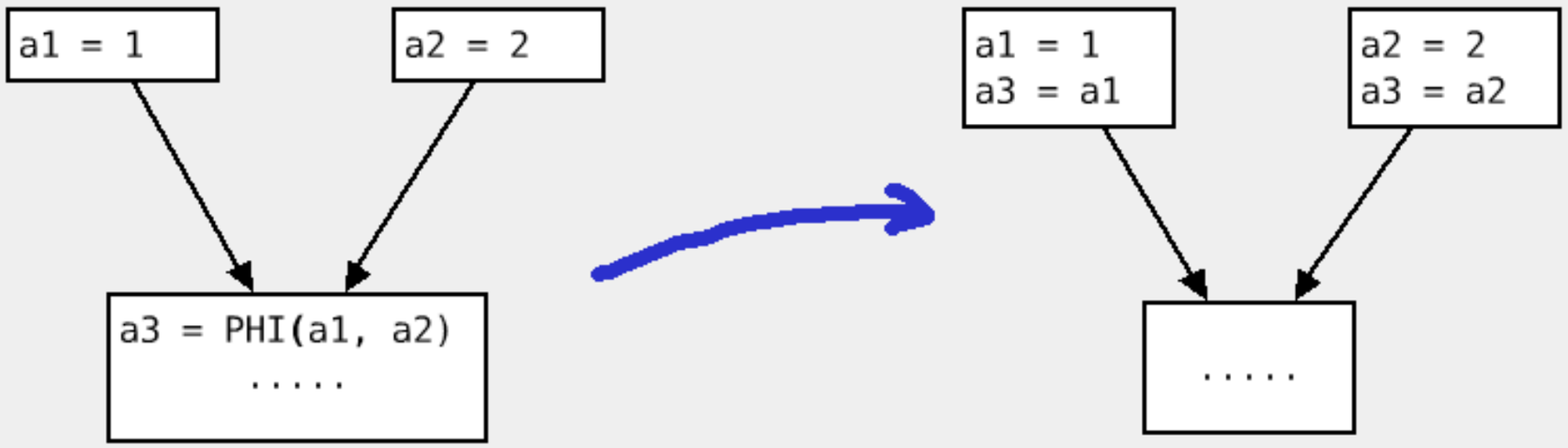
Aufbau



Squeak3.7a-5526.image

SSA Abbau

Klassisches Verfahren:



Probleme:

- Fehlerhaft nach Umordungen

- Kopien müssen entfernt werden

Phi-Kongruenzverfahren

Verfahren von Vugranam C. Sreedhar, Roy Dz-Ching Ju, David M. Gillies, und Vatsa Santhanam.

Idee: Transformiere Programm so, dass alle Variablen in Phi gleich:

```
a1 = PHI(a1,a1)   ⟶   a1 = a1
```

- Einfügen von Kopien

- Umbenennen von Variablen

Phi-Kongruenzverfahren: Definitionen

phi-Verbunden(x):

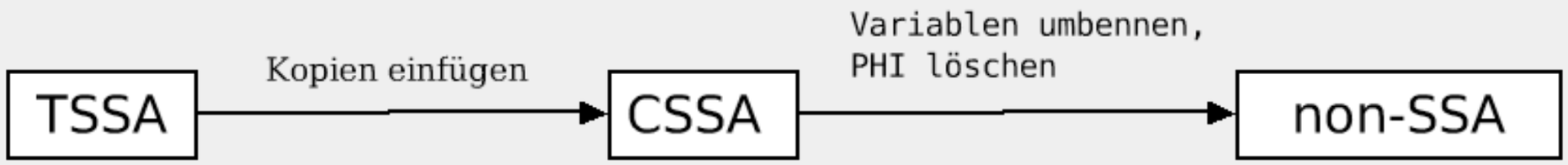
```
a3 = PHI(a1, a2)
a5 = PHI(a3, a4)
```
--> a1, a4 sind verbunden

Phi-Kongruenzklasse:
Reflexive, transitive Hülle von phi-Verbunden(x).

Phi-Kongruenzeigenschaft:
Ersetzen der Kongruenzklassen durch einen Repräsentanten möglich

CSSA: SSA mit Phi-Kongruenzeigenschaft.
TSSA: SSA ohne Phi-Kongruenzeigenschaft.

Phi-Kongruezverfahren: Vorgehen



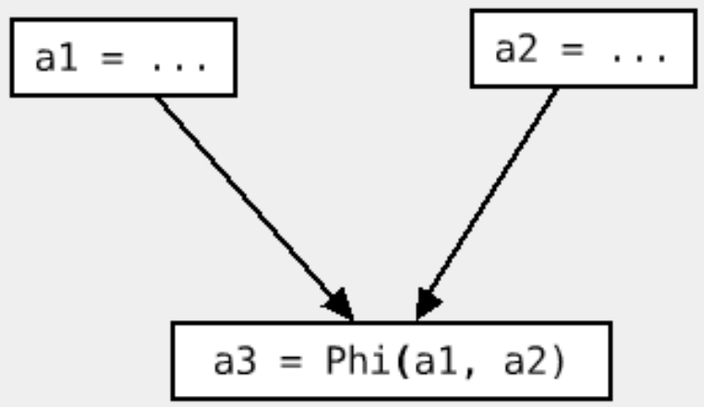
TSSA2CSSA:

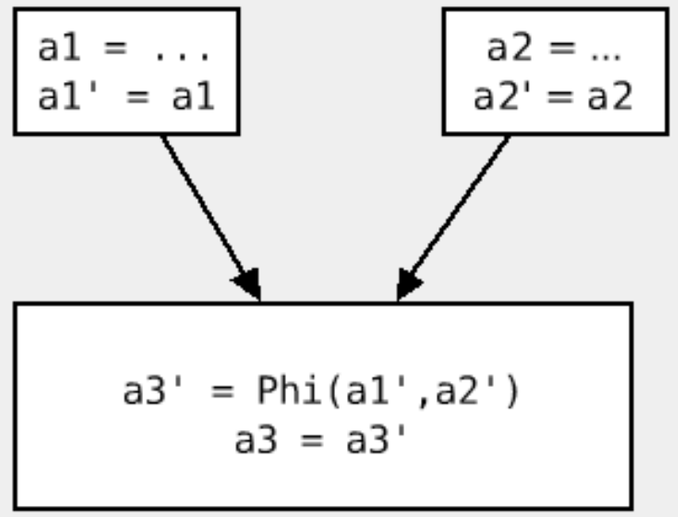
Verfahren I: - Kopien für alle Variablen einfügen
- Kopienfortschaltung nötig

Verfahren II:
- Schrittweise Kongruenzklassen aufbauen
- nur Kopie einfügen, wenn Interferenz
- Anzahl Kopien (1 oder 2) über Datenfluss (LiveIn/LiveOut)

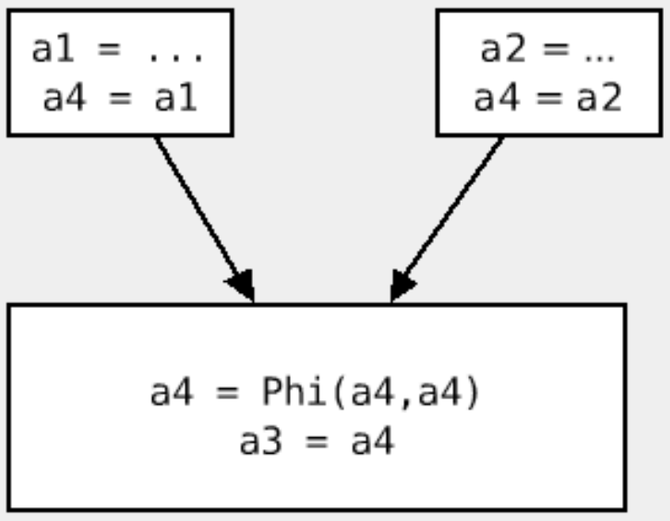
Bespiel SSA Abbau

1) TSSA nach CSSA:

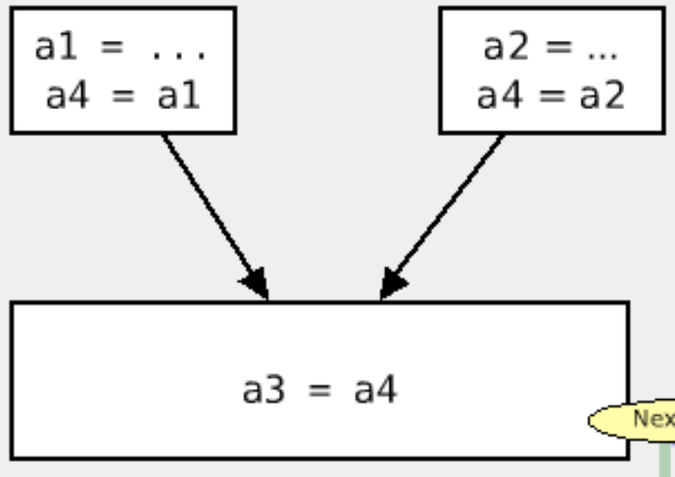




2) Umbennen



3) Phi-Fkt Löschen:



Existierende Infrastruktur

- Squeak 3.7alpha
  Entwicklungsumgebung und virtuelle Maschine

- AOStA (Adaptive Optimization for Smalltalk)
  SSA-Rahmenwerk

- ClosureCompiler
  Smalltalk-nach-Bytecode Übersetzer

- IRBuilder
  Symbolischer Assemblierer für Squeak Bytecodes

AOStA

Ziel: Adaptive Optimierung für Smalltalk

Aufbau:

1. Bytecode -> AST
2. Aufbau SSA-Darstellung
3. Optimierung

Eigenschaften:

- nur SSA für lokale Variablen
- keine Wertnummerierung im Aufbau
- Definitions/Benutzt Informationen
- Transformation / Besucher für Optimierung

IRBuilder

Symbolischer Assemblierer

Beispiel:

```
| ir aCompiledMethod |
ir := IRBuilder new
        rargs: #(self);
        pushLiteral: 1;
        returnTop;
        ir.
aCompiledMethod := ir compiledMethod.
```

Ausführen:

```
aCompiledMethod valueWithReceiver: nil arguments: #()
```

Im System installieren:

```
Float addSelector: #test withMethod: aCompiledMethod.
```

Umsetzung

- AOStA portiert

- Weitere Optimierungen:

    - Kopienweiterschaltung (copy propagation)
    - Entfernung Leerer Grundblöcke

- SSA Abbau mittels Phi-Kongruenzverfahren

    - dazu: Berechnung Lebendigkeit/Interferenz

- Codegenerator

Squeak3.7a-5526.image

Beispiel 1

```
examplePhiSimple
   | a b |
   a := 3.
   (a = 3) ifTrue: [b := 1]
           ifFalse: [b := 2].
   ^b
```

CompiledMethod

ClosureCompiler

AOStA

```
BB9: [
    t1a:= 3.
    if not (t1a = 3) goto BB18]
BB15: [
    t2a := 1.
    goto BB20]
BB18: [
    t2c := 2:]
BB20: [
    t2b := PHI(t2a, t2c).
    s3a := t2b.
    ^s3a]'
```

Optimierung und
SSA Abbau

```
BB9: [
   t1a := 3.
   if not (t1a = 3:) goto
BB18]
BB15: [
   t3a := 1 ).
   goto BB20]
BB18: [
   t3a := 2]
BB20: [
   ^t3a]
```

Codegenerierung

CompiledMethod

self	9 <20> pushConstant: 3
all bytecodes	10 <81 40> storeIntoTemp: 0
header	12 <20> pushConstant: 3
literal1	13 <B6> send: =
9	14 <9A> jumpFalse: 18
10	15 <76> pushConstant: 1
11	16 <69> popIntoTemp: 1
12	17 <91> jumpTo: 20
13	18 <77> pushConstant: 2
14	

```
self valueWithReceiver: nil
      arguments: #()
```

Squeak3.7a-5526.image

Ergebnisse

SSA Abbau: Anzahl Kopien

Verfahren I: 23891
Verfahren II: 1023 (ohne Opt: 0)

Ziele:

- Vollständiger Entwicklungszyklus unterstützt ✔

- Der Übersetzer erlaubt weitreichende Optimierungen ✔

- Integration in ein vorhandenes System ✔

- Codegenerator noch nicht vollständig

Ausblick

- Verbesserungen AOStA
  (Bsp: Kodierung Phi-Funktionen)

- Vereinfachung Smalltalk-Übersetzer

- Laufzeitinformation für Optimierung

- Experimente mit Zwischensprache
  (AST statt Bytecode)

Squeak3.7a-5526.image

Universität Karlsruhe (TH)
Institut für Programmstrukturen
und Datenorganisation

Lehrstuhl Professor Goos

Erweiterung eines statischen Übersetzers zu einem Laufzeitübersetzungssystem

Marcus Denker

Diplomarbeit

Verantwortlicher Betreuer: Prof. Dr. Gerhard Goos

Betreuender Mitarbeiter: Dipl.-Inform. Florian Liekweg