Praktikum Compilerbau

Prof. Dr.-Ing. Gregor Snelting
Matthias Braun und Jürgen Graf

Lehrstuhl für Programmierparadigmen
Universität Karlsruhe (TH)

22. April 2009

Gliederung

Praktikum Compilerbau

Willkommen am Lehrstuhl für Programmierparadigmen

- Die Betreuer:
  - Matthias Braun `braun@ipd.info.uni-karlsruhe.de`
  - Jürgen Graf `grafj@ipd.info.uni-karlsruhe.de`
- Vorraussetzungen:
  - Grundlegende Kenntnisse ueber Parsingtechniken und semantische Analyse (Vorlesung Sprachtechnologie und Compiler 1)
  - Gute Programmierkenntnisse in C/C++ ODER Java.
- Praktikum: 3 SWS, kann in Diplomprüfung eingebracht werden
- Jeden Mittwoch um 14h: Treffen im Raum 207

Gruppeneinteilung & Infrastruktur

- Geplant sind 4 Gruppen a 4-5 Personen.
- Jede Gruppe erhält einen gemeinsamen Account
  - Einige Arbeitsplätze stehen im Erdgeschoß zur Verfügung
  - SSH Zugang ist ohne VPN möglich, AFS nur mit VPN
  - Bei uns läuft Linux (Ubuntu, SuSE)
- Es gibt 2 Wikis und eine normale Webseite
  - Offizielle Seite: `http://pp.info.uni-karlsruhe.de/lehre/SS2009/compprakt/`
  - Praktikum spezifisches Wiki: `http://pp.info.uni-karlsruhe.de/lehre/SS2009/compprakt/wiki/`
  - Lehrstuhl Wiki (Infrastruktur): `http://pp.info.uni-karlsruhe.de/wiki/`

Aufgabe

Erstellen eines Compilers für die Sprache MiniJava

- Zwischencodeerzeugung, Optimierungen, Codegenerierung
- MiniJava $\longrightarrow$ Firm $\longrightarrow$ Java Bytecode
- Gearbeitet wird in Gruppen a 4-5 Personen
- Bei Abschluß einer größeren Phase wird ein Dokument erstellt
- Am Ende des Praktikums stellt jede Gruppe kurz ihr Werk vor

Details

Der Compiler kann wahlweise in Java oder C geschrieben werden.

Die geplanten Phasen bestehen aus:

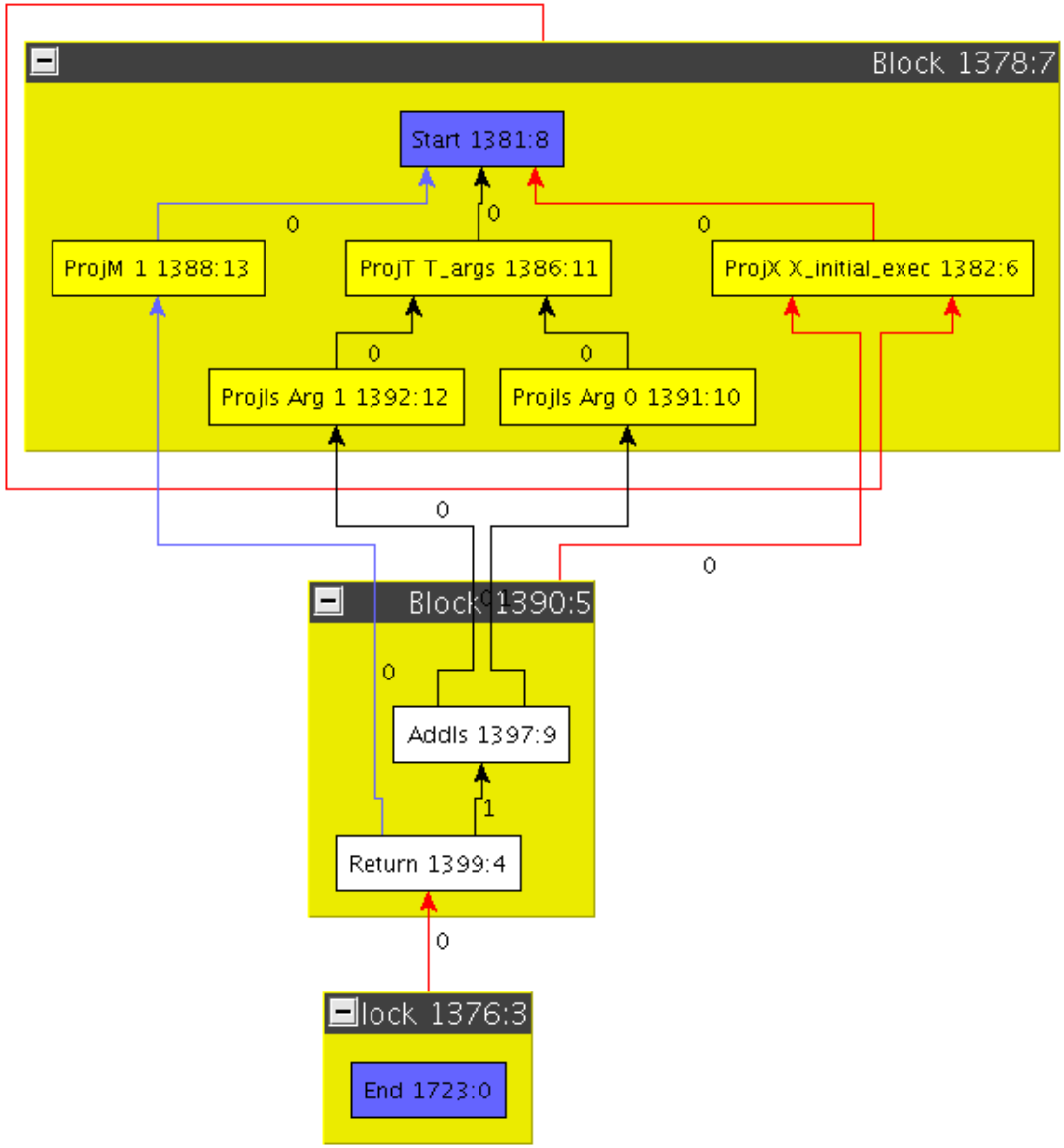
- Scannerspezifikation (Eingabe für Generator)
- Parserspezifikation (Eingabe für Generator)
- abstrakte Syntax/Baumaufbau
- Symboltabelle
- attributierte Grammatik zur Typprüfung
- elementare Programmanalysen/Optimierungen
- Codegenerierung

MiniJava

- Untermenge von Java
  → Kann mit `javac` übersetzt werden → einfaches Testen
- Objektorientiert
- Enthält: Methoden, Attribute, Parameter, lokale Variablen, Rekursion, ...
- Ohne: Vererbung, statische Felder, Threads, ...
- Sprachbericht → Webseite
- Erweiterungen sind erlaubt solange die geforderte Funktionalität enthalten ist (auf eigene Verantwortung)

Firm `libfirm.org`

- SSA basierte Zwischensprache in Graphform
- Statements sind Knoten
- Abhängigkeiten sind Kanten
- Reihenfolge nur noch als Halbordnung
- i386 Backend vorhanden



Java Bytecode

- Platformunabhängig
- Komfort: Laufzeitumgebung, Garbage Collector, ...
- Stackmaschine → Registervergabe entfällt

```
class A { int i; }

class B {
  A a;
  int x;

  static B foo(int x, B b) {
    A a = b.a;
    int i = b.x;
    a.i = i + x;
    return b;
  }
}
```

```
1  static B foo(int, B);
2     0:   aload_1
3     1:   getfield #2; //Field a
4     4:   astore_2
5     5:   aload_1
6     6:   getfield #3; //Field x
7     9:   istore_3
8     10:  aload_2
9     11:  iload_3
10    12:  iload_0
11    13:  iadd
12    14:  putfield #4; //Field A.i
13    17:  aload_1
14    18:  areturn
```

Bis zum nächsten Treffen

- Einlesen
- System einrichten: SVN, eclipse, etc.
- Java oder C?
- Wahl der Tools: Antlr, selbstgeschriebener Parser, lex/yacc
- Aufgabenblatt $\longrightarrow$ Webseite
- MiniJava

Viel Erfolg

Noch Fragen?

Kleine Motivation: Gute Umsetzungen könnten im Firm-Projekt übernommen werden