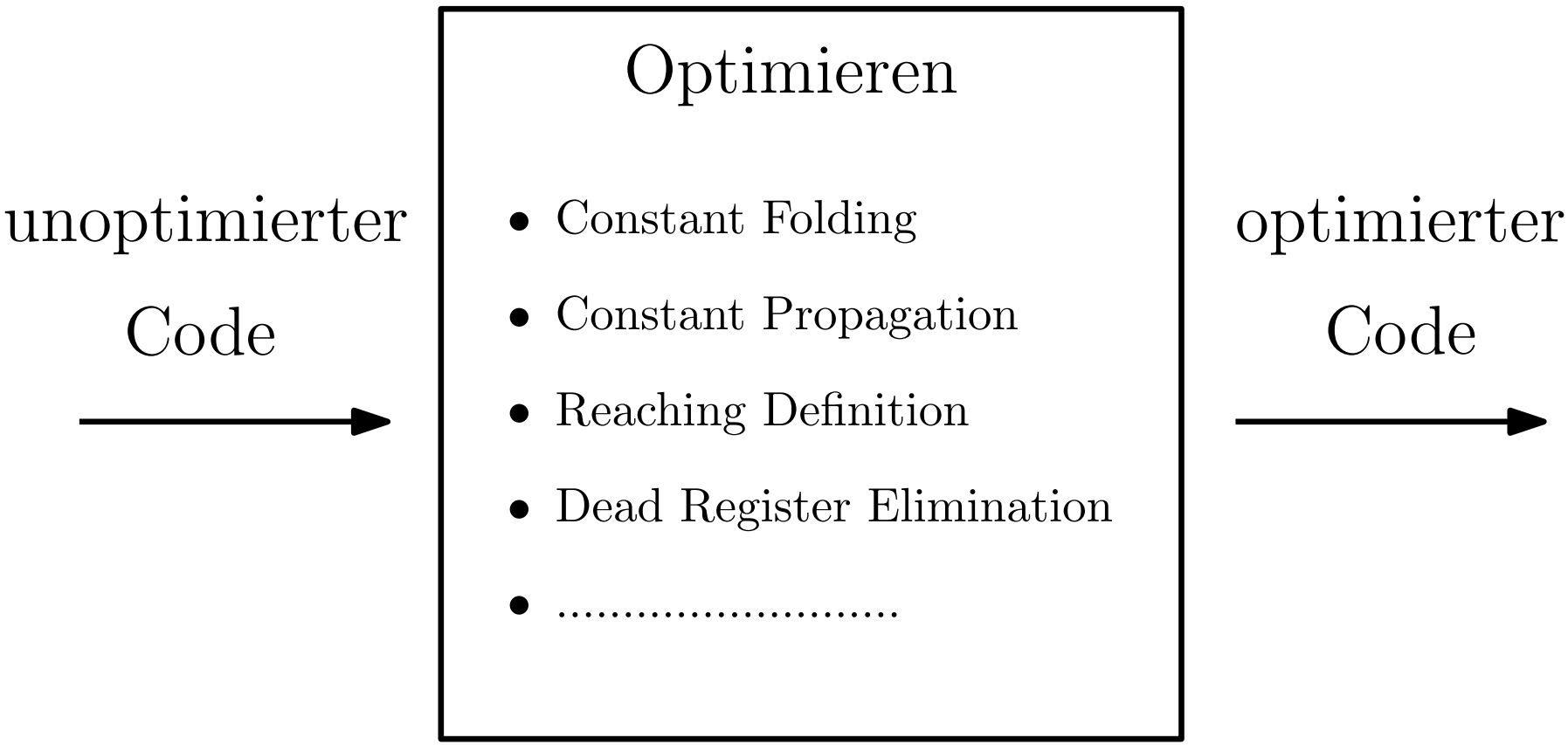
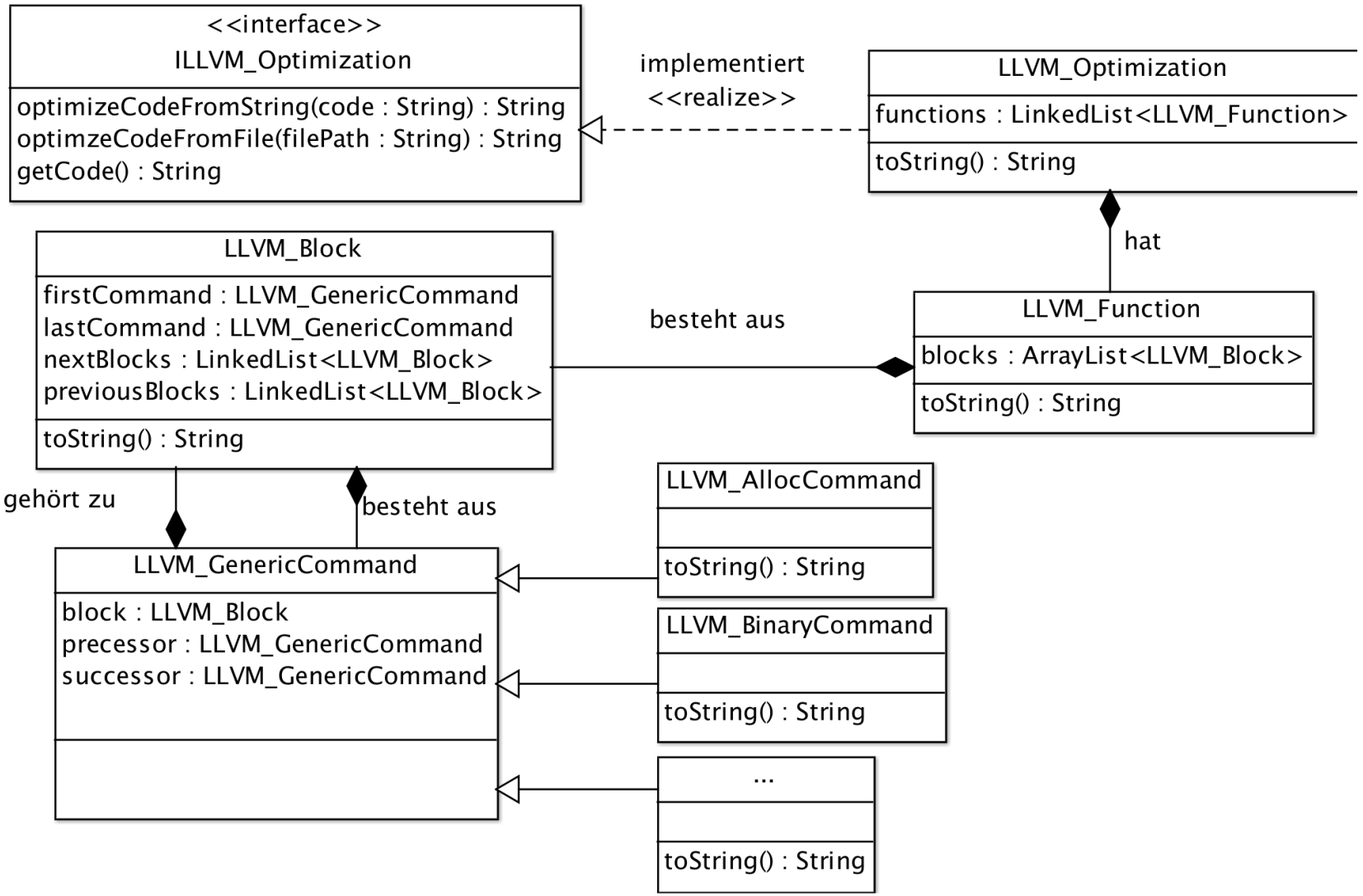
Freie Universität Berlin

Softwareprojekt Übersetzerbau
Optimierungstechniken

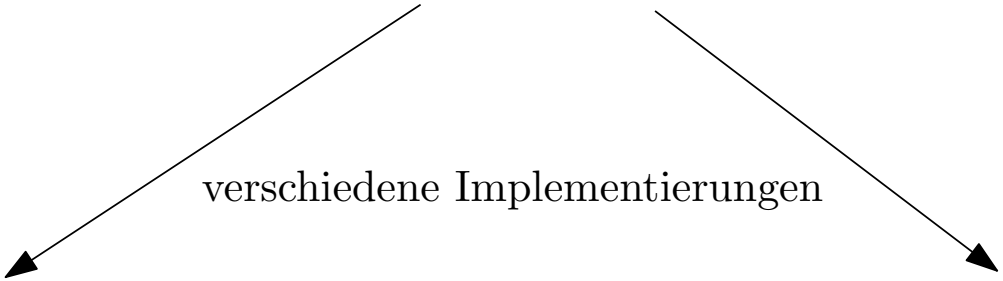
David Knötel, Björn Karger, Daniel Marzin
Freie Universität Berlin, Institut für Informatik


unoptimierter
Code
Optimieren
Constant Folding
Constant Propagation
Reaching Definition
Dead Register Elimination
.....
optimierter
Code

Programmstruktur



LLVM nutzt Static Single Assignment auf Registern



vereinfachte Algorithmen auf Registern

komplexere Algorithmen auf Speicheradressen

Optimierungen & Beispiele

1. Constant folding

```
%a = add i32 1, 7
```

unoptimierter Code

```
%a = add i32 8, 0
(bei uns eine Zuweisung)
```

optimierter Code

Optimierungen & Beispiele

1. Constant folding
2. Constant propagation

```
%a = add i32 8, 0
%b = sub i32 %a, 5
```

unoptimierter Code

```
%b = sub i32 8, 5
```

optimierter Code

Optimierungen & Beispiele

1. Constant folding
2. Constant propagation
3. Reaching definition analysis

```
store i32 5, i32* %a, align 4
%2 = load i32* %a, align 4
```

unoptimierter Code

```
%2 = add i32 5, 0
```

optimierter Code

Optimierungen & Beispiele

1. Constant folding
2. Constant propagation
3. Reaching definition analysis
4. Eliminate dead registers/blocks

```
br i1 1, label %j1, label %j2

j1: %a = add i32 1, 7
j2: %b = add i32 8, 0
```

unoptimierter Code

```
br label %j1

j1: %a = add i32 1, 7
```

optimierter Code

Optimierungen & Beispiele

1. Constant folding
2. Constant propagation
3. Reaching definition analysis
4. Eliminate dead registers/blocks
5. Remove local common subexpressions

```
%q = load i32* %x, align 4
%w = load i32* %x, align 4
```

unoptimierter Code

```
%q = load i32* %x, align 4
%w = add i32 %q, 0
```

optimierter Code

Optimierungen & Beispiele

1. Constant folding
2. Constant propagation
3. Reaching definition analysis
4. Eliminate dead registers/blocks
5. Remove local common subexpressions
6. Global liveness analysis

```
%a = alloca i32, align 4
store i32 1, i32* %a, align 4
ret i32 0
```

unoptimierter Code

```
ret i32 0
```

optimierter Code

Optimierungen & Beispiele

1. Constant folding
2. Constant propagation
3. Reaching definition analysis
4. Eliminate dead registers/blocks
5. Remove local common subexpressions
6. Global liveness analysis
7. Strength reduction

```
%a = mul i32 4, %b
```

unoptimierter Code

```
%a = shl i32 %b, 2
```

optimierter Code

Fazit / Ausblick

- Es wurden mehrere Optimierungsalgorithmen erfolgreich auf einen gegebenen LLVM-Code angewendet. Auch bei natürlichem, aus C-Code über CLANG erzeugten LLVM-Code wurden erfolgreich bedeutende Mengen an Codezeilen entfernt.

- Eine Weiterentwicklung des Programms wäre durch die Vielzahl an potentiellen weiteren Optimierungstechniken problemlos möglich. Priorität hätte hierbei die Anwendung von Schleifenoptimierungen.

Danke

Vielen Dank für Ihre Aufmerksamkeit.