

Mustersuche

Mustersuche

Lernziele

- Varianten der Mustersuche abhängig von der Programmdarstellung
- Formalismen zur Mustersuche

Kontext

- Mustersuche ist Teil von Code-Transformationen

Anwendungen der Mustersuche

- Lokalisierung bei Code-Transformationen
- Programmiermuster (Planerkennung):
 - Jahr-2000-Problematik
 - Suche nach Datumsmanipulationen
 - Suche nach Schaltjahrberechnung
 - Erkennung typischer Datenstrukturen bzw. Algorithmen, wie z.B. Suche nach Listeniteration
 - Code-Anomalien oder Verletzung von Coding-Style-Guides

Statische Suchräume

- Text
- Lexeme
- Syntax
- (statisch-)semantische Information
- globale Daten- und Kontrollfluss-Information

Problem

Die Suche nach Klone hat ergeben, dass es unzählige Tausch-Operationen der Form

```
t = a; a = b; b = t;
```

gibt. Ersetzen Sie sie gegen SWAP (&a, &b) mit:

```
void SWAP (int *x, int *y) {  
    int t = *x;  
    *x = *y;  
    *y = t;  
}
```

Textuelle Ebene

- Suche nach Worten wie „tausch“, „swap“ etc.
- Suche nach regulärem Ausdruck:

grep

```
'[a-zA-Z][a-zA-Z0-9]* *= *[a-zA-Z][a-zA-Z0-9]* *; *  
[a-zA-Z][a-zA-Z0-9]* *= *[a-zA-Z][a-zA-Z0-9]* *; *  
[a-zA-Z][a-zA-Z0-9]* *= *[a-zA-Z][a-zA-Z0-9]* *; *' file.c
```

- Suche nach „kontextsensitivem regulären“ Ausdruck:

grep

```
'\([a-zA-Z][a-zA-Z0-9]*\) *= *\([a-zA-Z][a-zA-Z0-9]*\) *; *  
\2 *= *\([a-zA-Z][a-zA-Z0-9]*\) *; *  
\3 *= *1 *; *' file.c
```

Textuelle Ebene

Gegenbeispiele:

```
t = a; a = b;  
    = t;
```

```
t=a; a=b; /* a comment */ b=t;
```

Lexem-basierte Suche

Tokenstrom:

```
swap-match () =  
id = id ; id = id ; id = id ;
```

Mit zusätzlichen Token-Attributen:

```
id1 = id2 ; id3 = id4 ; id5 = id6 ;
```

```
where id1.text = id6.text  
      and id2.text = id3.text  
      and id4.text = id5.text
```

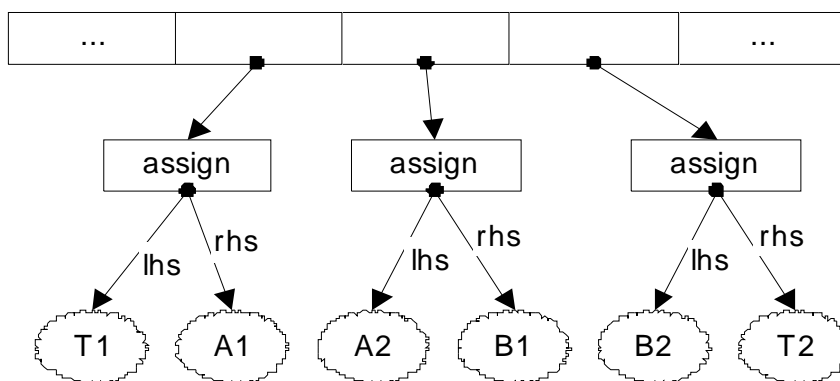
Lexem-basierte Suche

Gegenbeispiele:

`t = a; a = b; ; b = t;`

`t=a[i]; a[i]=b[j]; b[j]=t;`

Syntax-basierte Suche



mit den folgenden Gleichheiten (nicht Identitäten):

$T1 = T2, A1 = A2$ und $B1 = B2$

`seq (_, assign ($T, $A), assign ($A, $B), assign ($B, $T), _)`

Syntax-basierte Suche

Gegenbeispiele:

```
float t, a, b;  
...  
t = a; a = b; b = t;
```

wegen Signatur von SWAP:

```
SWAP (int *x, int *y)
```

Statisch-semantische Information

```
swap-match ($T, $A, $B) =  
  seq (_,  
    assign ($T, $A),  
    assign ($A, $B),  
    assign ($B, $T),  
    _)  
where type ($T)=type($A)=type($B)=int
```

Statisch-semantische Information

Gegenbeispiel:

```
t = a; a = b; x = y; b = t;
```

Erster Lösungsansatz:

```
seq (_, assign ($T, $A), _ ,  
      assign ($A, $B), _ ,  
      assign ($B, $T), _)
```

Aber:

```
t = a; a = b; goto l; b = t;
```

Statisch-semantische Information

Zweiter Lösungsansatz:

```
swap-match ($T, $A, $B) =  
seq (_, assign ($T, $A), $X ,  
      assign ($A, $B), $Y ,  
      assign ($B, $T), _)
```

where not contains (\$X, goto)
and not contains (\$X, label)
and not contains (\$Y, goto)
and not contains (\$Y, label)

Statisch-semantische Information

Gegenbeispiel:

```
t=a; goto l; x=y; l: a=b; b=t;
```

(... wobei `l` von keinem anderen `goto` angesprungen wird.)

Kontrollflussinformation

swap-match (\$T, \$A, \$B) =

S1: assign (\$T, \$A),

S2: assign (\$A, \$B),

S3: assign (\$B, \$T),

where pdom* (S1) = S2

and pdom* (S2) = S3

and dom* (S2) = S1

and dom* (S3) = S2

Kontrollflussinformation

Gegenbeispiele:

`t = a; a = b; t = x; b = t;`

`a = 1; b = 2;`

`t = a; a = b; b = t;`

- o Ergebnis: `a==t==b==2`, wenn `t` und `a` Aliase sind.
- o Ergebnis von SWAP (`&a, &b`):
`a == 2` und `b == 1`

Datenflussinformation

swap-match (`$T, $A0, $B0, $A1, $B1`) =

- S1: assign (`$T, $A0`),
- S2: assign (`$A1, $B0`),
- S3: assign (`$B1, $T`)

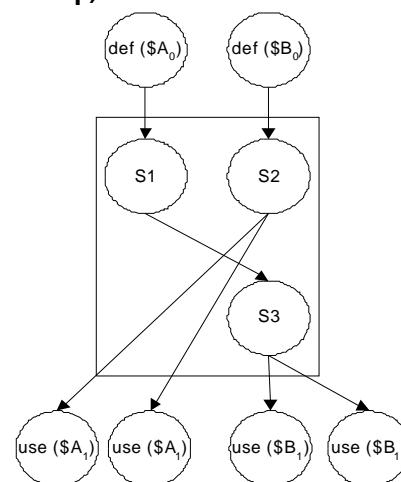
where

S1 verwendet ausschließlich `def(A0)`

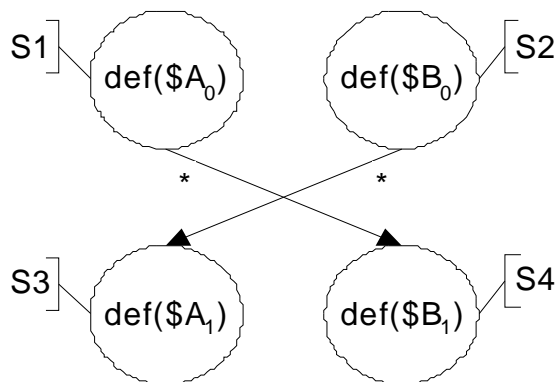
S2 verwendet ausschließlich `def(B0)`

S3 verwendet ausschließlich die
Definition in S1

die Definition in S1 wird nur in S3 verwendet



Datenflussinformation



uses (S3) = {def (\$B₀)}

and uses (S4) = {def (\$A₀)}

Formalismen

- Text
 - reguläre Ausdrücke über Zeichenketten
- Lexeme
 - reguläre Ausdrücke über Tokens mit Bedingungen über Token-Attribute
- Syntax
 - Matching über abstrakten Syntaxbaum

Formalismen

- (statisch–)semantische Information
 - Matching über abstrakten Syntaxbaum mit statisch–semantischen Bedingungen
- globale Daten– und Kontrollfluss–Information
 - Matching über abstrakten Syntaxbaum mit statisch–semantischen Bedingungen und Bedingungen über Kontroll– und Datenfluss