4. Tiefensuche und Breitensuche

Algorithmus zur Bestimmung schwacher Zusammenhangskomponenten

WCOMP

```
          /* Anfangs sind alle Knoten und */
          /* alle Kanten unmarkiert */

1   for (alle Knoten v aus V)    /* Knotenliste */
2       { if (v unmarkiert)
3             { richte neue schwache Zusammenhangskomponente C ein;
4               if (v isolierter Knoten)
5                   { kennzeichne C als uneigentliche
6                         schwache Zusammenhangskomponente;
7                   }
8               else
9                   { kennzeichne C als a-kreisfrei;
10                    WCP(v, C);
11      }   }   }
```

WCP(v,C)

```
1   if (v markiert)
2       { kennzeichne C als nicht a-kreisfrei;
3         return;
4       }
5   markiere v;
6   füge v in die Zusammenhangskomponente C ein;
7   for (alle Kanten, Ausgangsbögen und Eingangsbögen l von v)
8       { if (l unmarkiert)
9             { markiere l;
10              füge l in die Zusammenhangskomponente C ein;
11              WCP(otherend(l, v), C);
12      }   }
13  return;
```

4. Tiefensuche und Breitensuche

Algorithmus zu topologischen Präsortierung

PTOPSORT

```
        /* Anfangs sind alle Knoten unmarkiert. */
        /* Der Keller stack ist anfangs leer. */

1   for (alle Knoten v aus V)    /* Knotenliste */
2       { if (v unmarkiert)
3             PTPS(v);
4       };
```

PTPS(v)

```
1   if (v markiert) return;
2   markiere v;
3   for (alle Kanten und Ausgangsbögen l von v)    /* rekursiver Aufruf */
4       { if (l unmarkiert)
5             { markiere l;
6               PTPS(otherend(l, v);
7       }     };
8   push (stack, v);
```

4. Tiefensuche und Breitensuche

Algorithmus zur Bestimmung der starken Zusammenhangskomponenten, des externen Dags und der Schichtennnummern

STRONGCOMP

```
        /* Anfangs sind alle Knoten unmarkiert und alle Kanten ungefärbt. */
        /* Alle Knoten und alle starken Zusammenhangskomponenten */
        /* haben 0 als Voreinstellung für die Schichtennummer */
        /* Im Keller stack stehen die Knoten in PTOPSORT-Reihenfolge. /*
1   v = pop(stack);
2   while (v ≠ NULL)    /* Keller abarbeiten */
3      { if (v unmarkiert)
4           { if (v hat keine Kanten und keine ungefärbten Eingangsbögen)
5                { kennzeichne v als rückkehrfrei;
6                  füge v in Warteschlange queue ein;
7                  for (alle Eingangsbögen a von v)
8                     { if (lv(otherend(a, v)) >= lv(v)) lv(v) = lv(otherend(a, v)) + 1 ;
9                     }
10                 for (alle Ausgangsbögen a von v)
11                    { a als Bogen des externen Dag kennzeichnen;
12                      färbe a gelb;
13               }    }
14            else
15               { richte neue starke Zusammenhangskomponente SC ein;
16                 STRCP(v, SC);
17                 for (alle Knoten v in SC)
18                    {  if (v hat gelbe oder ungefärbte Bögen)
19                          { kennzeichne v als schwachen Verheftungspunkt von SC;
20                            for (alle gelben Eingangsbögen a von v)
21                               { if (lv(otherend(a, v)) >= lv(v))
22                                  lv(v) = lv(otherend(a, v)) + 1 ;
23                               }
24                            if (lv(v) > lv(SC)) lv(SC) = lv(v);
25                            for (alle ungefärbten Ausgangsbögen a von v)
26                               { a als Bogen des externen Dag kennzeichnen;
27                                 färbe a gelb;
28                    }     }     }
29                 for (alle Knoten v in SC) lv(v) = lv(SC);
30          }    }
31       v = pop(stack);
32    }
```

4. Tiefensuche und Breitensuche

b-Tiefensuche zur Bestimmung der Knoten und Linien einer starken Zusammenhangskomponente

STRCP(v,SC)

```
1   if (v markiert)
2       {kennzeichne SC als f-zyklisch;
3        return;
4       }
5   markiere v;
6   füge v in SC ein;
7   füge v in Warteschlange queue ein;
8   for (alle Kanten und Eingangsbögen l von v)
9       {if (l ungefärbt)
10          {färbe l blau;
11           füge l in die Zusammenhangskomponente SC ein;
12           STRCP(otherend(l, v), SC);
13      }   }
```

4. Tiefensuche und Breitensuche

Breitensuche

Es sei v ein Knoten eines allgemeinen Graphen. Ein Knoten $w \neq v$ gehört *zur f-Schale l von v*, wenn man auf einem f-Weg in l Schritten von v zu w kommen kann, aber nicht in weniger Schritten. Entsprechen sind a-Schalen und b-Schalen definiert.

Satz: a.In einem anfänglich unmarkierten Graph besucht eine von v ausgehende f-Breitensuche (a-Breitensuche, b-Breitensuche) alla von v f-erreichbaren (a-ereichbaren, b-erreichbaren) Knoten und nur diese.

b. Es werden erst alle Knoten der Schale l markiert, eihe ein Knoten der Schla $l+1$ markiert wird.

5. Die Biblockzerlegung

Höherer Zusammenhang

Ein allgemeiner Graph heißt *k-f-zusammenhängend*, wenn zu jedem Knotenpaar (v, w) mit $v \neq w$ k intern disjunkte f-Wege von v nach w existieren.
Ein allgemeiner Graph heißt *k-f-linienzusammenhängend*, wenn zu jedem Knotenpaar (v, w) mit $v \neq w$ k liniendisjunkte f-Wege von v nach w existieren.
Für a-Wege und b-Wege gelten entsprechende Definitionen. Bei a-Zusammenhang brauchen die Wege nur für (v, w) zu existieren. a-Wege in Rückrichtung sind dann auch vorhanden.

Von besonderer Bedeutung ist der 2-a-Zusammenhang, bzw. 2-a-Linienzusammenhang. Man spricht auch von *zweifachem* Zusammenhang.

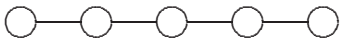
In einem 2-a-zusammenhängenden Graphen liegen je zwei Knoten auf einem a-Kreis.

5. Die Biblockzerlegung

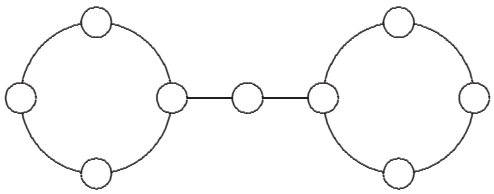
Zweifacher Zusammenhang läßt sich durch *Äquivalenzklassen von Linien* charkterisieren.

Klassen geschlossener a-Wege:

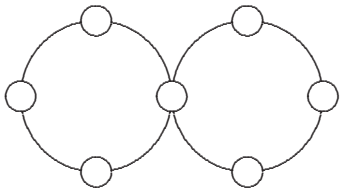
a-Kreise $\subseteq$ linieneinfache geschlossene a-Wege $\subseteq$ stoppfreie Wege



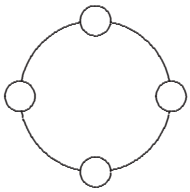
a. Geschlossener a-Weg



b. Stoppfreier Weg



c. Linieneinfacher geschlossener a-Weg



d. a-Kreis

Stoppfreier Weg: Geschlossener a-Weg, bei dem keine Linie zweimal hintereinander durchlaufen wird und die letzte Linie verschieden von der ersten ist.

5. Die Biblockzerlegung

Definition

Zwei Linien e und f sind

1. *stoppfrei verbunden*, wenn es einen stoppfreien Weg gibt, der e und f enthält,
2. *linieneinfach verbunden*, wenn es einen linieneinfachen geschlossenen a-Weg gibt, der e und f enthält,
3. *kreisverbunden*, wenn es einen a-Kreis gibt, der e und f enthält.

Satz

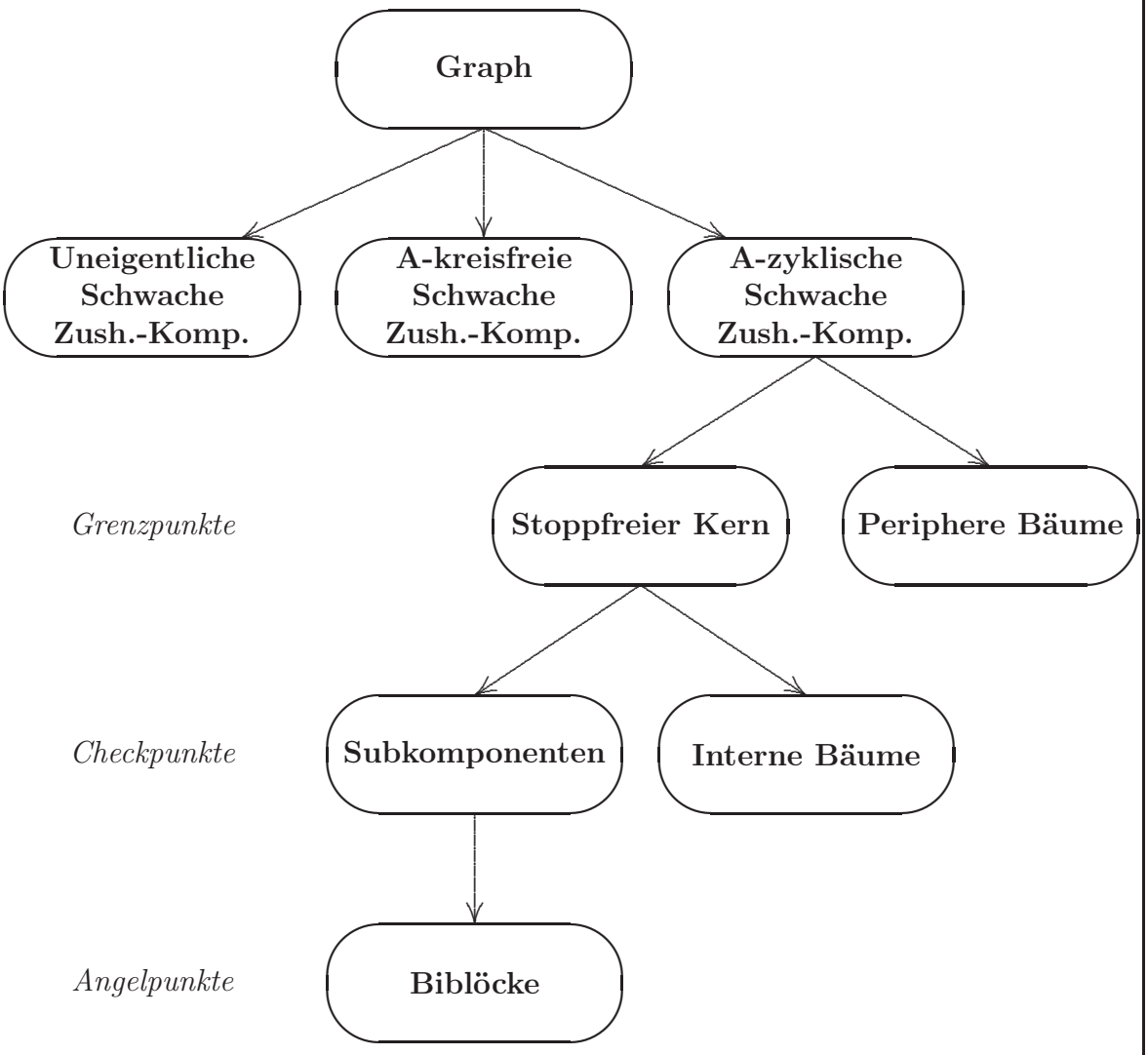
1. Stoppfreie Verbundenheit ist eine Äquivalenzrelation in der Menge der Linien, die auf einem stoppfreien Weg liegen.
2. Kanteneinfache Verbundenheit ist eine Äquivalenzrelation in der Menge der Linien, die auf einem linieneinfachen geschlossenen a-Weg liegen.
3. Kreisverbundenheit ist eine Äquivalenzrelation in der Menge der Linien, die auf einem a-Kreis liegen.

Definition

1. Eine Äquivalenzklasse stoppfrei verbundener Linien heißt *stoppfreier Kern*.
2. Eine Äquivalenzklasse linieneinfach verbundener Linien heißt *Subkomponente*.
3. Eine Äquivalenzklasse kreisverbundener Linien heißt *Biblock*.

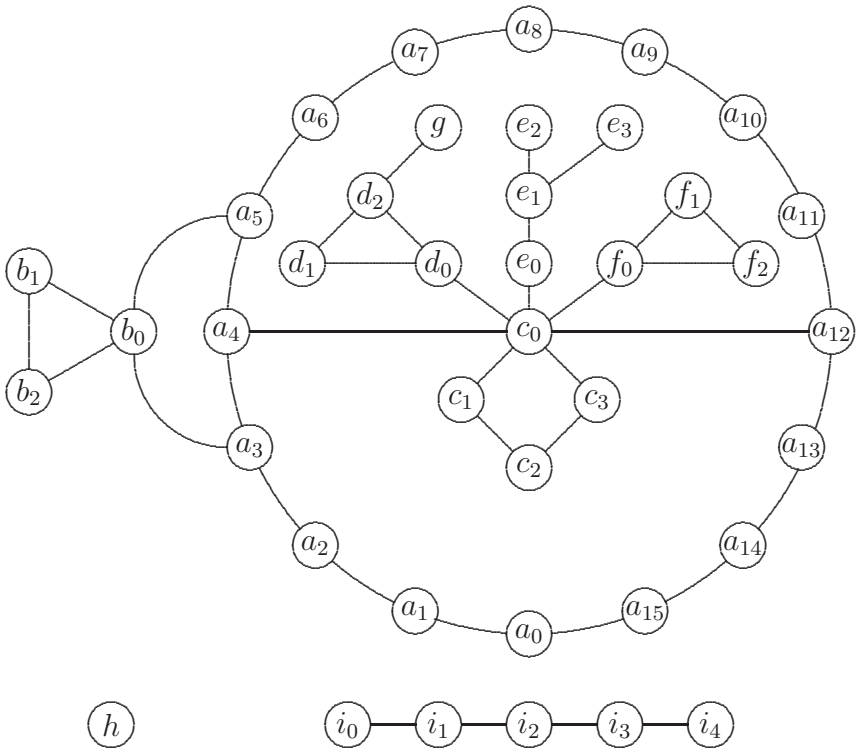
Die von den Äquivalenzklassen erzeugten Untergraphen tragen die gleichen Namen. *Komponenten der Biblockzerlegung*.

5. Die Biblockzerlegung



5. Die Biblockzerlegung

Beispiel Ugraph1



5. Die Biblockzerlegung

Periphere Bäume

Satz: 1. In einer schwachen a-zyklischen Zusammenhangskomponente gibt es genau einen stoppfreien Kern.

2. Eine Brücke mit Inzidenzpunkten a und b liegt genau dann auf einem stoppfreien Weg, wenn sowohl $[a]$ als auch $[b]$ einen a-Kreis enthalten.

3. Jeder periphere Baum besitzt genau einen Grenzpunkt.

Subkomponenten und interne Bäume

Satz: 1. Alle Nichtbrücken, die mit dem gleichen Knoten inzidieren, gehören zur gleichen Subkomponente.

2. Ein interner Baum hat mit einer Subkomponente höchstens einen Knoten gemeinsam. Er besitzt mindesten zwei Checkpunkte.

Biblöcke

Satz: 1. Jeder Biblock einer Subkomponente, die mehrere Biblöcke unfaßt, enthält mindestens einen Angelpunkt.

2. Zwei verschiedene Biblöcke einer Subkomponente haben höchstens einen Angelpunkt gemeinsam.

3. Zwei Angelpunkte einer Subkomponente liegen genau dann auf einem a-Kreis, wenn sie zum selben Biblock gehören.

5. Die Biblockzerlegung

Programm zur Biblockzerlegung von Ugraph1

```
  #include <stdio.h>
  #include <GHSstructure.h>

int main()
{
  GRAPH      *graph, *bgraph ;

  graph = readgraphlist(NULL);
  astd(graph);            // Bestimmung der Biblockzerlegung
  gprstd(graph, STRCS, NULL);
  aprstd(graph, STRR, NULL);
//aprstd(graph, STRA, NULL);
//aprstd(graph, STR, NULL);
  aprstdvt(graph, VPA, NULL);
//
  return 0;
}
```

5. Die Biblockzerlegung

Zerlegungsübersicht für Ugraph1

```
BEGIN CONDENSED STRUCTURE OF GENERAL DECOMPOSITION
$GRAPH Ugraph1
$TYPE  UGSLF
$No_VERTICES                          40
$No_EDGES                             44
$No_ARCS                               0
$No_ISOLATED_VERTICES                  1
$No_WEAK_COMPONENTS_TYPE_1             0  (0 f-trees)
$No_WEAK_COMPONENTS_TYPE_2             1  (1 f-trees)
$No_WEAK_COMPONENTS_TYPE_3             0  (0 rooted)
$No_WEAK_COMPONENTS_TYPE_4             0  (0 rooted)
$No_WEAK_COMPONENTS_TYPE_5             1  (1 rooted)
$No_WEAK_ATTACHMENT_POINTS             0
$No_STRONG_COMPONENTS_(f-ACYCLIC)      1
$No_STRONG_COMPONENTS_(f-CYCLIC)       1
$No_STOPFREE_KERNELS                   1
$No_PERIPHERAL_TREES                   2
$No_SUBCOMPONENTS                      3
$No_BIBLOCKS                           5
$No_INTERNAL_TREES                     1
$No_HINGE_POINTS                       2
$No_CHECK_POINTS                       3
END CONDENSED STRUCTURE OF GENERAL DECOMPOSITION
```

5. Die Biblockzerlegung

Zerlegungsübersicht für Ugraph1

```
BEGIN BIBLOCK DECOMPOSITION
REDUCED STRUCTURE
$GRAPH Ugraph1
$TYPE  UGSLF
$No_VERTICES                40
$No_EDGES                   44
$No_ARCS                     0
$No_ISOLATED_VERTICES        1
$No_A-ACYCLIC_WEAK_COMPONENTS       1     (5V, 4E, 0A)
$No_A-CYCLIC_WEAK_COMPONENTS        1     (34V, 40E, 0A)
   $A-CYCLIC_WEAK_COMPONENT Ugraph1.WCOMP1   (34V, 40E, 0A, 5AP, 2BP, 3CP, 2HP)
   $No_PERIPHERAL_TREES 2
        $PERIPHERAL_TREE Ugraph1.WCOMP1.PT0  (5V, 4E, 0A)
        $PERIPHERAL_TREE Ugraph1.WCOMP1.PT1  (2V, 1E, 0A)
        $STOPFREE_KERNEL Ugraph1.WCOMP1.STP  (29V, 35E, 0A, 5AP, 2BP, 3CP, 2HP)
        $No_SUBCOMPONENTS 3
            $SUBCOMPONENT Ugraph1.WCOMP1.SUB0     (23V, 27E, 0A, 2AP, 1BP, 1CP, 2HP)
            $No_BIBLOCKS 3
                $BIBLOCK Ugraph1.WCOMP1.SUB0.BLB0          (3V, 3E, 0A, 1AP, 0BP, 0CP, 1HP)
                $BIBLOCK Ugraph1.WCOMP1.SUB0.BLB1          (4V, 4E, 0A, 1AP, 1BP, 1CP, 1HP)
                $BIBLOCK Ugraph1.WCOMP1.SUB0.BLB2          (18V, 20E, 0A, 2AP, 1BP, 1CP, 2HP)
            $SUBCOMPONENT Ugraph1.WCOMP1.SUB1     (3V, 3E, 0A, 2AP, 1BP, 1CP, 0HP)
            $No_BIBLOCKS 1
                $BIBLOCK Ugraph1.WCOMP1.SUB1.BLB0          (3V, 3E, 0A, 2AP, 1BP, 1CP, 0HP)
            $SUBCOMPONENT Ugraph1.WCOMP1.SUB2     (3V, 3E, 0A, 1AP, 0BP, 1CP, 0HP)
            $No_BIBLOCKS 1
                $BIBLOCK Ugraph1.WCOMP1.SUB2.BLB0          (3V, 3E, 0A, 1AP, 0BP, 1CP, 0HP)
        $No_INTERNAL_TREES 1
            $INTERNAL_TREE Ugraph1.WCOMP1.IT0    (3V, 2E, 0A, 3AP, 1BP, 3CP, 1HP)
END REDUCED STRUCTURE
```

5. Die Biblockzerlegung

Verheftungspunkte von Ugraph1 (Teil I)

```
BIBLOCK DECOMPOSITION
PRINT OPTION VPA  (Undirected loops are counted twice!)
$GRAPH Ugraph1
$TYPE UGSLF
$No_VERTICES 40
$No_EDGES 44
$No_ARCS 0
VERTICES

d0              $TDG 3    $DG 3    $ODG 0    $IDG 0    Ugraph1.WCOMP1
   Check Point
   $NO_BIBLOCK_EDGES                    2  (Ugraph1.WCOMP1.SUB1.BLB0)
                _d0*d1
                _d0*d2
   $NO_INTERNAL_TREE_EDGES              1  (Ugraph1.WCOMP1.IT0)
                _c0*d0

d2              $TDG 3    $DG 3    $ODG 0    $IDG 0    Ugraph1.WCOMP1
   Border Point
   $NO_PERIPHERAL_TREE_EDGES            1  (Ugraph1.WCOMP1.PT1)
                _d2*g
   $NO_BIBLOCK_EDGES                    2  (Ugraph1.WCOMP1.SUB1.BLB0)
                _d0*d2
                _d1*d2

f0              $TDG 3    $DG 3    $ODG 0    $IDG 0    Ugraph1.WCOMP1
   Check Point
   $NO_BIBLOCK_EDGES                    2  (Ugraph1.WCOMP1.SUB2.BLB0)
                _f0*f1
                _f0*f2
   $NO_INTERNAL_TREE_EDGES              1  (Ugraph1.WCOMP1.IT0)
                _c0*f0

b0              $TDG 4    $DG 4    $ODG 0    $IDG 0    Ugraph1.WCOMP1
   Hinge Point
   $NO_BIBLOCK_EDGES                    2  (Ugraph1.WCOMP1.SUB0.BLB0)
                _b0*b1
                _b0*b2
   $NO_BIBLOCK_EDGES                    2  (Ugraph1.WCOMP1.SUB0.BLB2)
                _b0*a3
                _b0*a5
```

5. Die Biblockzerlegung

Verheftungspunkte von Ugraph1 (Teil II)

```
c0               $TDG 7    $DG 7    $ODG 0    $IDG 0    Ugraph1.WCOMP1
   Border Point  Check Point  Hinge Point
   $NO_PERIPHERAL_TREE_EDGES              1  (Ugraph1.WCOMP1.PT0)
                _c0*e0
   $NO_BIBLOCK_EDGES                      2  (Ugraph1.WCOMP1.SUB0.BLB1)
                _c0*c1
                _c0*c3
   $NO_BIBLOCK_EDGES                      2  (Ugraph1.WCOMP1.SUB0.BLB2)
                _c0*a12
                _c0*a4
   $NO_INTERNAL_TREE_EDGES                2  (Ugraph1.WCOMP1.IT0)
                _c0*d0
                _c0*f0
```

5. Die Biblockzerlegung

Biblockgraph zu Ugraph1