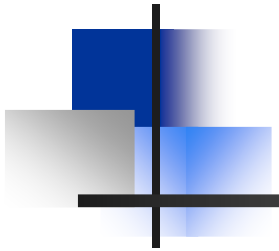


A. Turing, Checking a large routine
Proseminar Assertions SS07

Betreut von Prof. Dr. Heike Wehrheim
Vortragender : Vitali Gripp

Übersicht

- Einführung
- Autoren
- Kleines Beispiel zur Einführung (Addition)
- Größeres Beispiel (Fakultät mit Addition)
- Andeutung eines generellen Verfahrens

Einführung

- Assertions (Zusicherungen)
  - Beweis der Korrektheit des Programms (formale Verifikation)
    - Implementierung entspricht Spezifikation

Einführung

- Assertions (Zusicherungen)
  - Beweis der Korrektheit des Programms (formale Verifikation)
    - Implementierung entspricht Spezifikation
  - boolesche Aussagen, die Anforderungen an Programme beschreiben

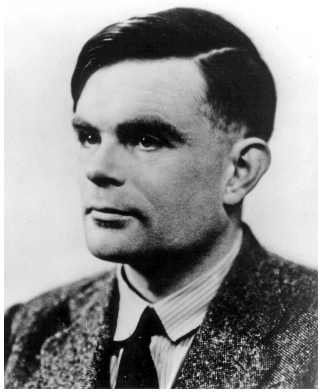
Einführung

- Assertions (Zusicherungen)
  - Beweis der Korrektheit des Programms (formale Verifikation)
    - Implementierung entspricht Spezifikation
  - boolesche Aussagen, die Anforderungen an Programme beschreiben
- Anfänge von Assertions
  - Checking a large routine by A. Turing

Einführung

- Assertions (Zusicherungen)
  - Beweis der Korrektheit des Programms (formale Verifikation)
    - Implementierung entspricht Spezifikation
  - boolesche Aussagen, die Anforderungen an Programme beschreiben
- Anfänge von Assertions
  - Checking a large routine by A. Turing
  - An Early Programm Proof by A. Turing

Autoren



- Alan Mathison Turing
  - Lebte 1912 – 1954
  - Gilt als einer der einflussreichsten Theoretiker in der Informatik
  - Schrieb seit Ende 1945 Computerprogramme
    - Erst am National Physical Laboratory in London
    - Dann an der Universität Manchester
  - Stellte „Checking a large routine“ im Juni 1949 vor

Autoren

- F.L. Morris
  - Lehrte an der Universität Syracuse im Staat New York
- C.B. Jones
  - IBM Product Test Group
  - Professor der Informatik an der Universität Manchester

Autoren

- F.L. Morris
  - Lehrte an der Universität Syracuse im Staat New York
- C.B. Jones
  - IBM Product Test Group
  - Professor der Informatik an der Universität Manchester

- An Early Program Proof by Alan Turing
  - Entstand 1981 während eines Besuchs der Forschungsgruppe für Programmierung in Oxford

Kleines Beispiel zur Einführung (Addition)

- Möchten die Korrektheit dieser Addition zeigen

```
  1 3 7 4
  5 9 0 6
  6 7 1 9
  4 3 3 7
  7 7 6 8
-----------
2 6 1 0 4
```

Kleines Beispiel zur Einführung (Addition)

- Möchten die Korrektheit dieser Addition zeigen

```
  1 3 7 4
  5 9 0 6
  6 7 1 9
  4 3 3 7
    7 7 6 8
-----------
2 6 1 0 4
```

- 1. Möglichkeit:
  - Komplett nachrechnen
  - Bei großen Zahlen sehr aufwendig

Kleines Beispiel zur Einführung (Addition)

- Möchten die Korrektheit dieser Addition zeigen

```
  1 3 7 4
  5 9 0 6
  6 7 1 9
  4 3 3 7
  7 7 6 8
-----------
2 6 1 0 4
```

- 1. Möglichkeit:
  - Komplett nachrechnen
  - Bei großen Zahlen sehr aufwendig
- 2. Möglichkeit:
  - In Abschnitte aufteilen und die Abschnitte auf Korrektheit Prüfen
  - D.h. für jeden Abschnitt eine boolesche Aussage (Assertion)

Kleines Beispiel zur Einführung (Addition)

1374	1	3	7	4
5906	5	9	0	6
6719	6	7	1	9
4337	4	3	3	7
7768	7	7	6	8
26104	23	29	17	34

Kleines Beispiel zur Einführung (Addition)

1374		1	3	7	4
5906		5	9	0	6
6719		6	7	1	9
4337		4	3	3	7
7768		7	7	6	8
26104		23	29	17	34

3974
2213
26104

Kleines Beispiel zur Einführung (Addition)

1 3 7 4		1	3	7	4
5 9 0 6		5	9	0	6
6 7 1 9		6	7	1	9
4 3 3 7		4	3	3	7
7 7 6 8		7	7	6	8
2 6 1 0 4		2 3	2 9	1 7	3 4

3 9 7 4

2 2 1 3

2 6 1 0 4

Kleines Beispiel zur Einführung (Addition)

1 3 7 4		1	3	7	4
5 9 0 6		5	9	0	6
6 7 1 9		6	7	1	9
4 3 3 7		4	3	3	7
7 7 6 8		7	7	6	8
2 6 1 0 4		2 3	2 9	1 7	3 4

3 9 7 4

2 2 1 3

2 6 1 0 4

Kleines Beispiel zur Einführung (Addition)

1 3 7 4		1	3	7	4
5 9 0 6		5	9	0	6
6 7 1 9		6	7	1	9
4 3 3 7		4	3	3	7
7 7 6 8		7	7	6	8
2 6 1 0 4		2 3	2 9	1 7	3 4

3 9 7 4

2 2 1 3

2 6 1 0 4

Kleines Beispiel zur Einführung (Addition)

1 3 7 4	1	3	7	4
5 9 0 6	5	9	0	6
6 7 1 9	6	7	1	9
4 3 3 7	4	3	3	7
7 7 6 8	7	7	6	8
2 6 1 0 4	2 3	2 9	1 7	3 4

3 9 7 4

2 2 1 3

2 6 1 0 4

Kleines Beispiel zur Einführung (Addition)

```
 1 3 7 4
 5 9 0 6
 6 7 1 9
 4 3 3 7
   7 7 6 8
-----------
 2 6 1 0 4
```

```
   1   3   7   4
   5   9   0   6
   6   7   1   9
   4   3   3   7
     7   7   6   8
-----------------
   2 3   2 9   1 7   3 4
```

```
   3 9 7 4
 2 2 1 3
---------
 2 6 1 0 4
```

```
 1 3 7 4
 5 9 0 6
 6 7 1 9
 4 3 3 7
   7 7 6 8
---------
   3 9 7 4
 2 2 1 3
---------
 2 6 1 0 4
```

Beispiel: Fakultät

- Prinzip kann bei großen Programmen eingesetzt werden

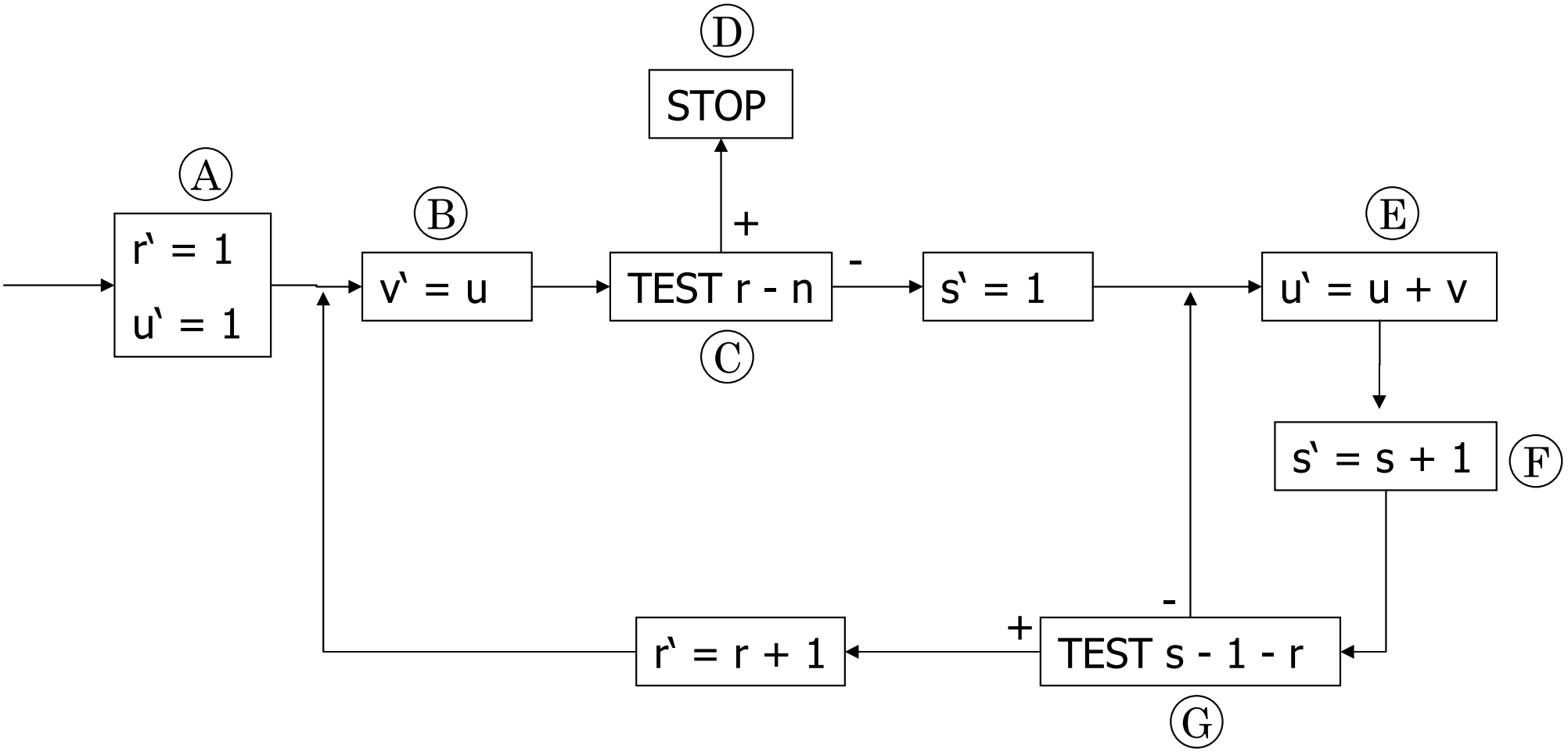
Beispiel: Fakultät

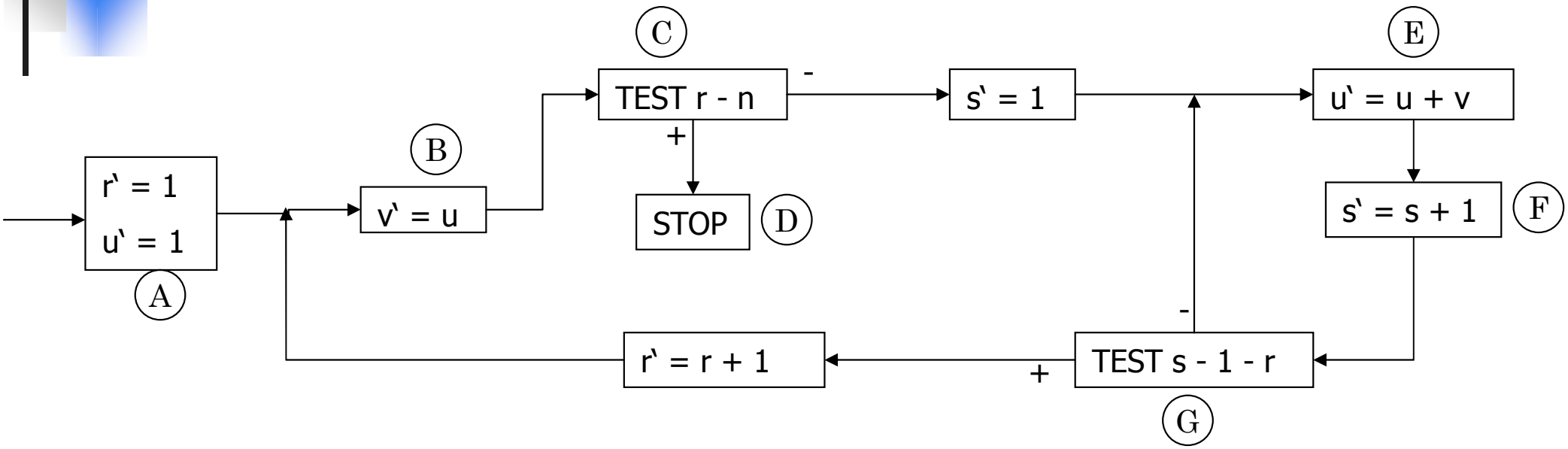
- Prinzip kann bei großen Programmen eingesetzt werden
- Beispiel: Programm zur Berechnung der Fakultät
  - nur mit Addition

Beispiel: Fakultät

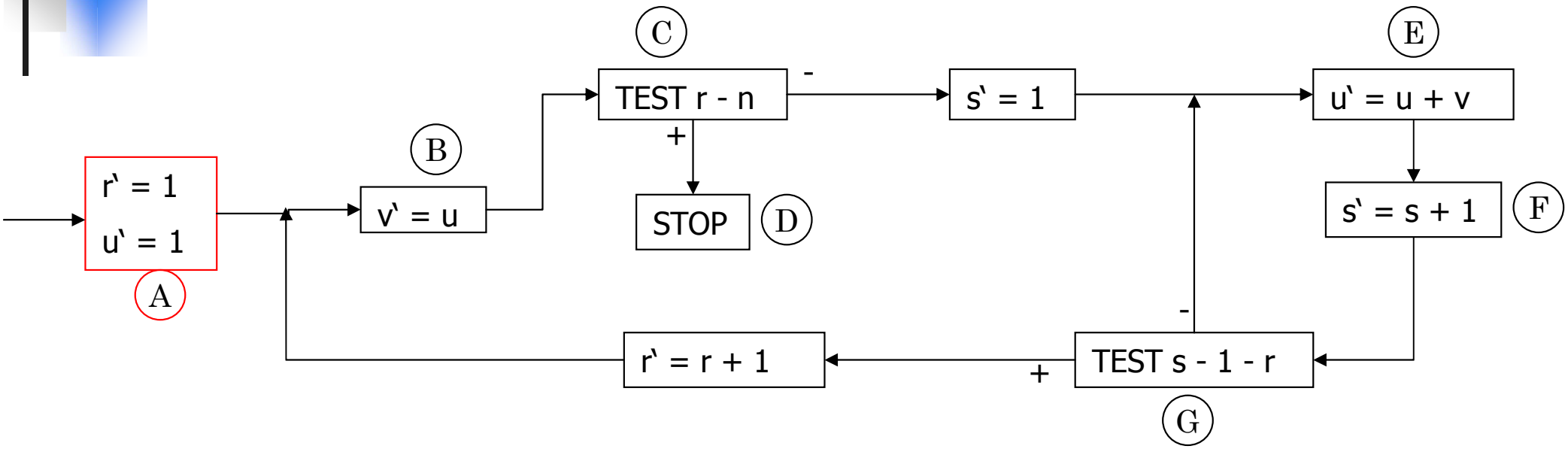
- Prinzip kann bei großen Programmen eingesetzt werden
- Beispiel: Programm zur Berechnung der Fakultät
  - nur mit Addition
- Vorher:
  - s r! + r! = (s+1) r!
  - s = r+1 => s r! = (r+1) r! = (r+1)!
  - Werte von s, r, n, u, v in den Speicherstellen 27 bis 31
  - in u befinden sich Ausdrücke folgender Art
    - s r! (= (r+1)! bei s = r+1)
    - (s+1) r! (= (r+1)! bei s = r)
    - s (r-1)! (= r! bei s = r)

Fakultät mit Addition

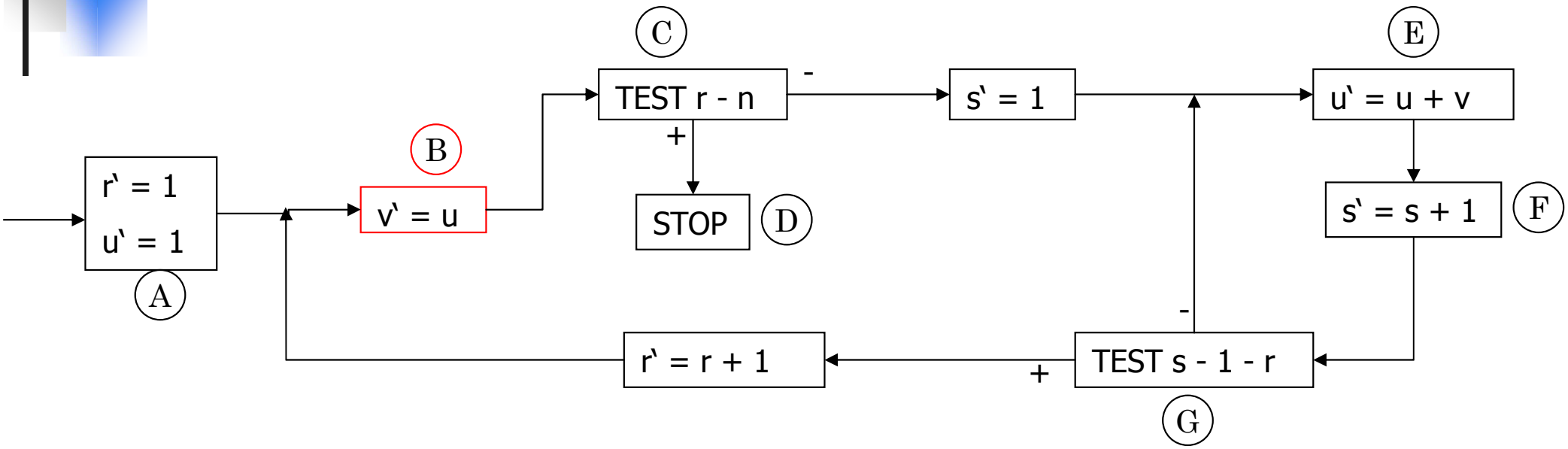




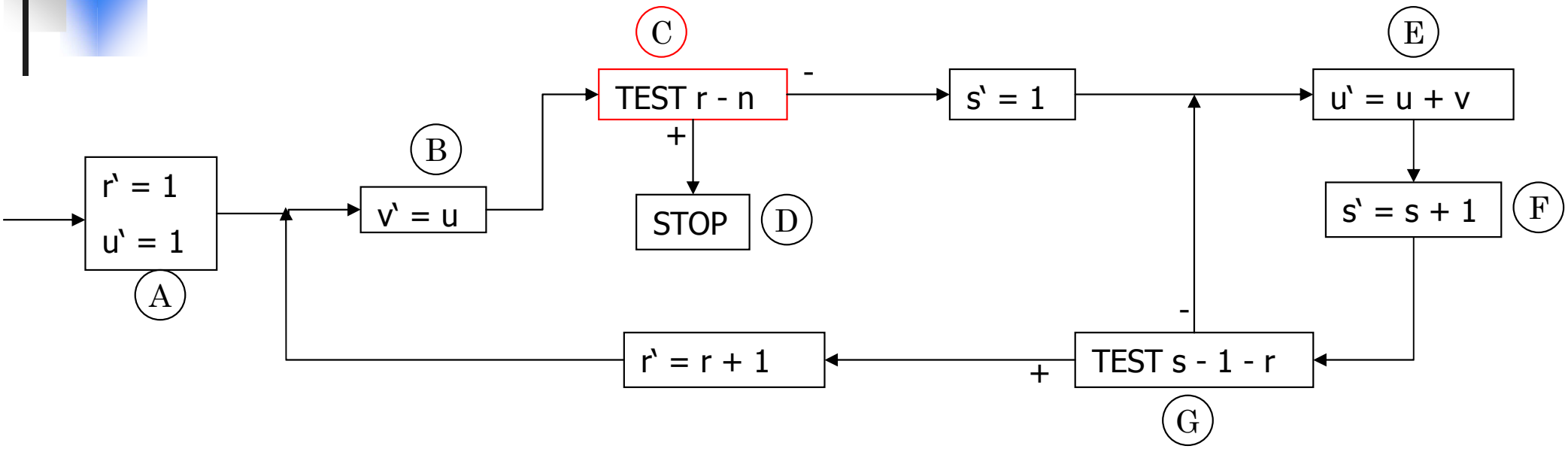
A (Initial)	B	C	D (Stop)	E	F	G



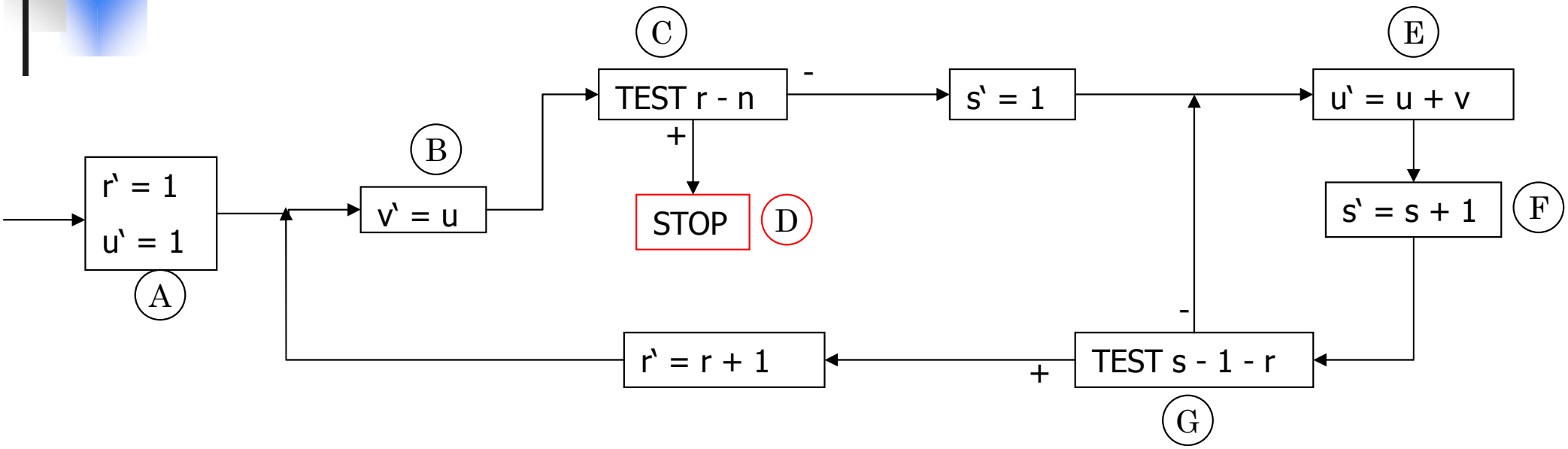
A (Initial)	B	C	D (Stop)	E	F	G
n						
TO B WITH r‘=1 u‘=1						



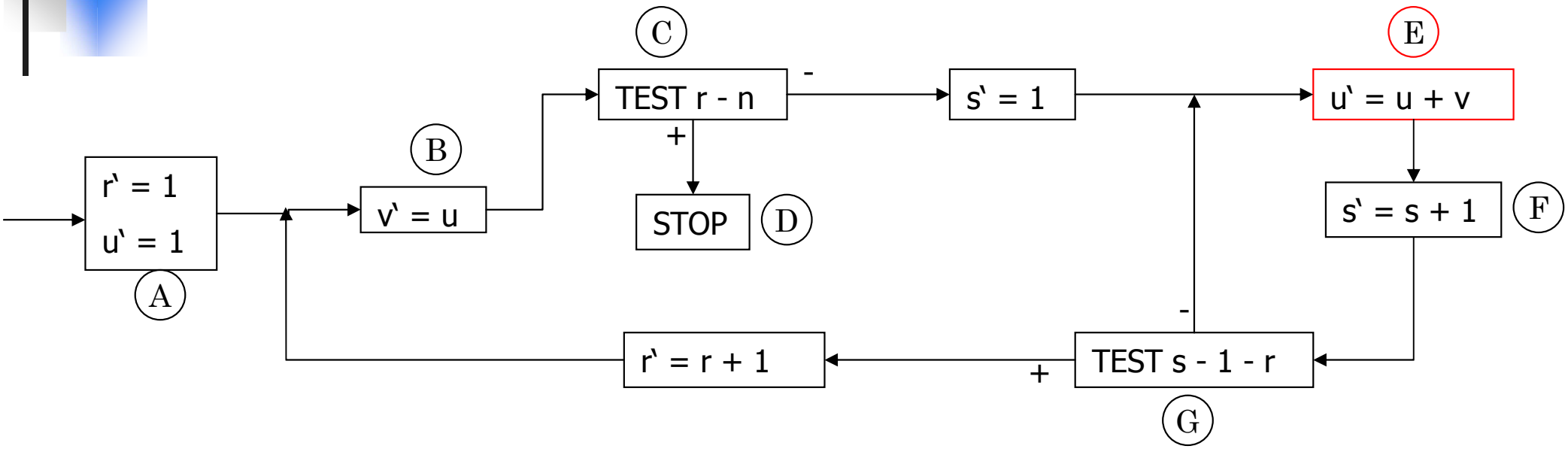
A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!					
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u					



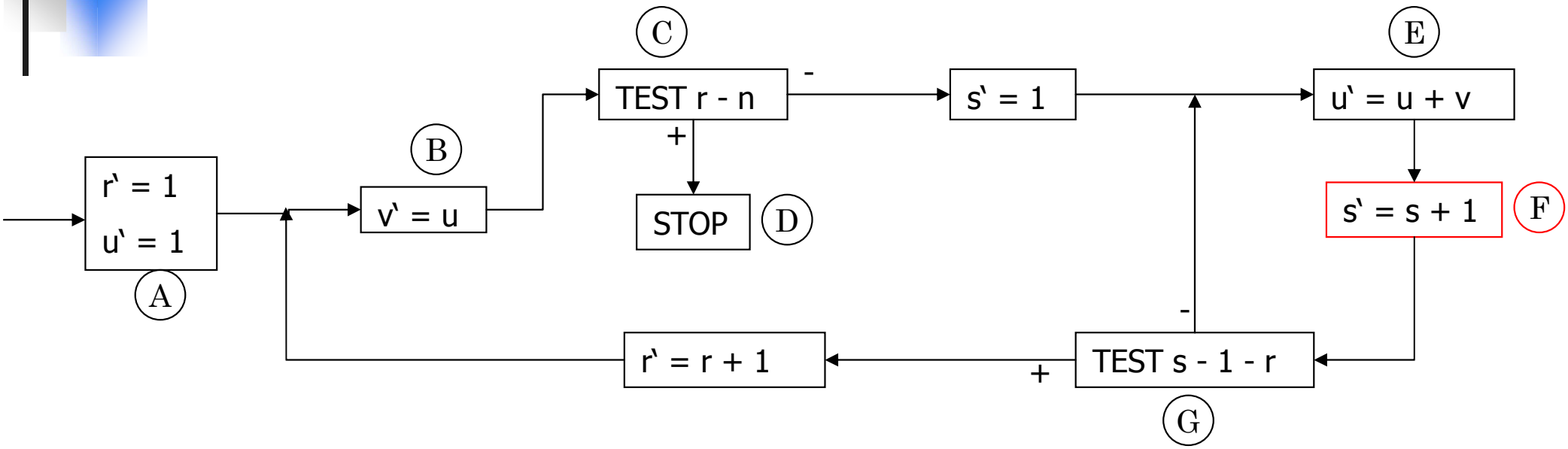
A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!				
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;				



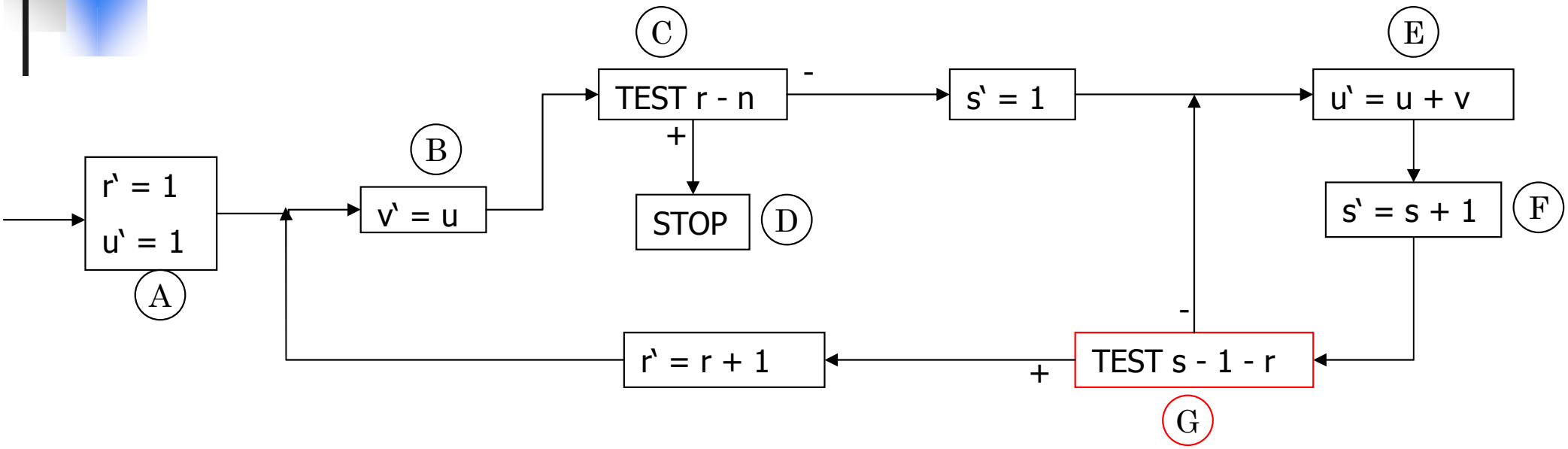
A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!	n v = n!			
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;				



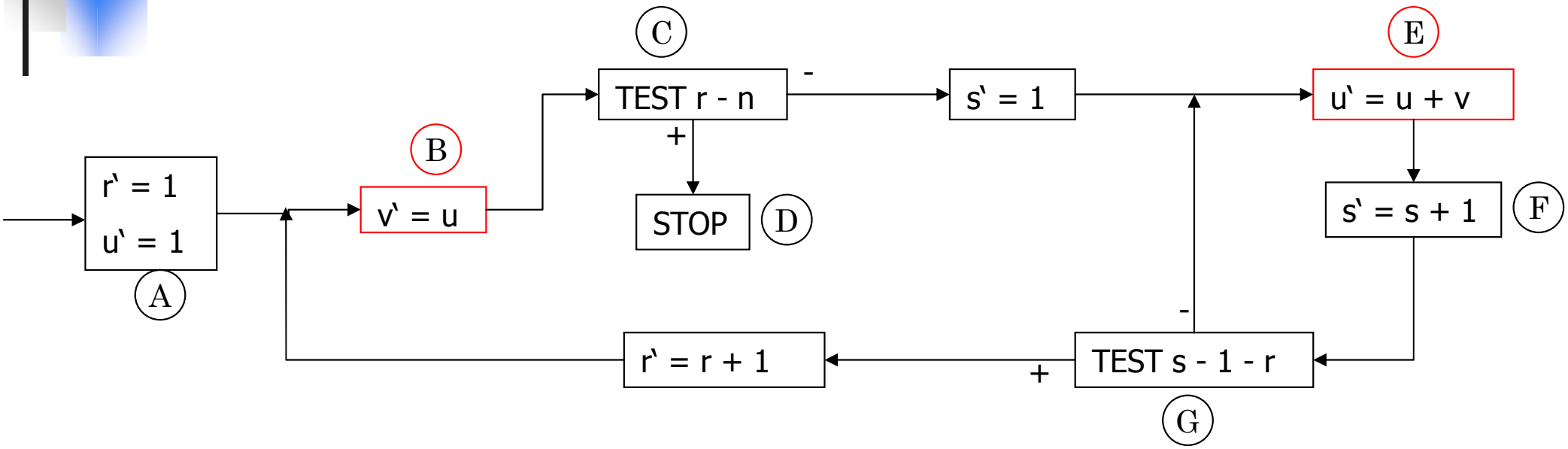
A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!	n v = n!	n r s u = s r! v = r!		
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;		TO F WITH u‘=u+v		



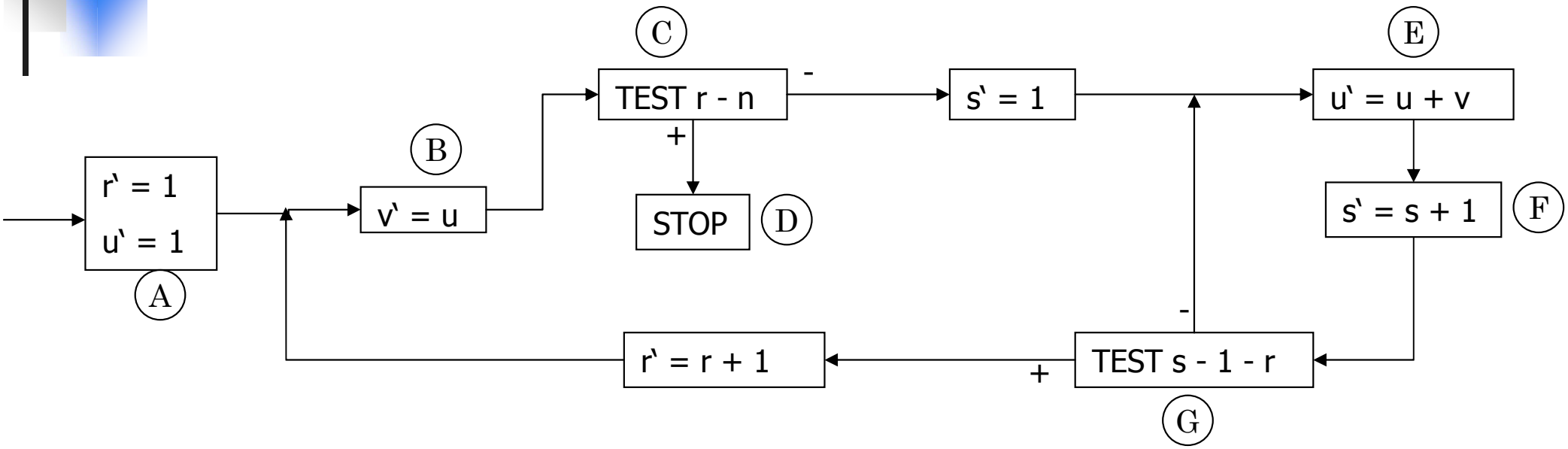
A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!	n v = n!	n r s u = s r! v = r!	n r s u = (s+1)r! v = r!	
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;		TO F WITH u‘=u+v	TO G WITH s‘=s+1	



A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!	n v = n!	n r s u = s r! v = r!	n r s u = (s+1)r! v = r!	n r s u = s r! v = r!
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;		TO F WITH u‘=u+v	TO G WITH s‘=s+1	IF s = = r+1 TO B WITH r‘=r+1 ; IF s < = r TO E

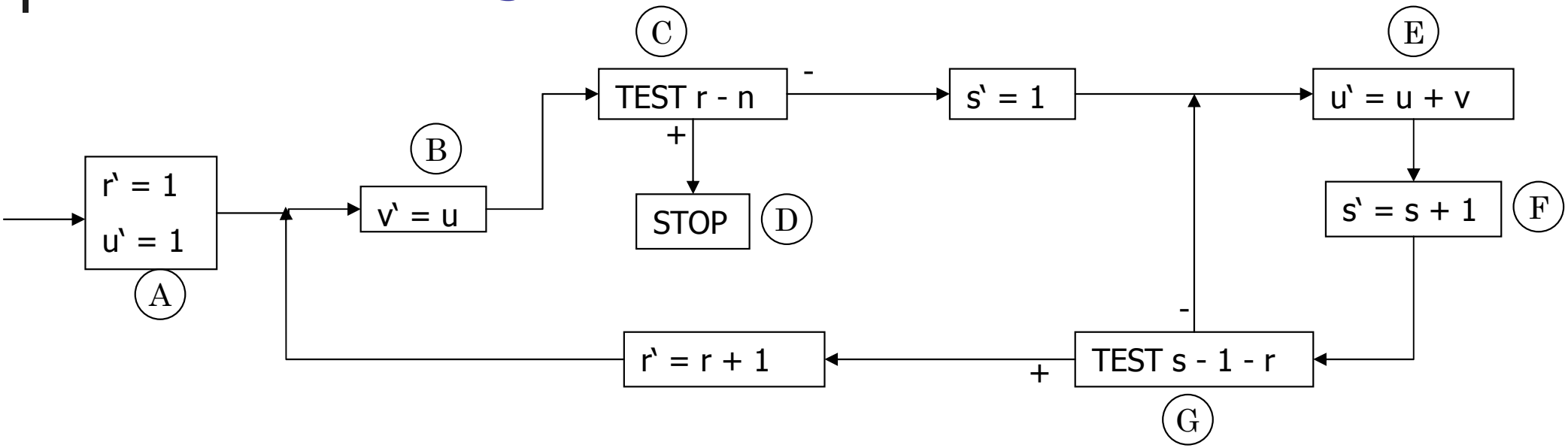


A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!	n v = n!	n r s u = s r! v = r!	n r s u = (s+1)r! v = r!	n r s u = s r! v = r!
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;		TO F WITH u‘=u+v	TO G WITH s‘=s+1	IF s = = r+1 TO B WITH r‘=r+1 ; IF s < = r TO E



A (Initial)	B	C	D (Stop)	E	F	G
n	n r u = r!	n r u = r! v = r!	n v = n!	n r s u = s r! v = r!	n r s u = (s+1)r! v = r!	n r s u = s r! v = r!
TO B WITH r‘=1 u‘=1	TO C WITH v‘=u	IF r = = n TO D ; IF r < n TO E WITH s‘=1;		TO F WITH u‘=u+v	TO G WITH s‘=s+1	IF s = = r+1 TO B WITH r‘=r+1 ; IF s < = r TO E

Terminierung

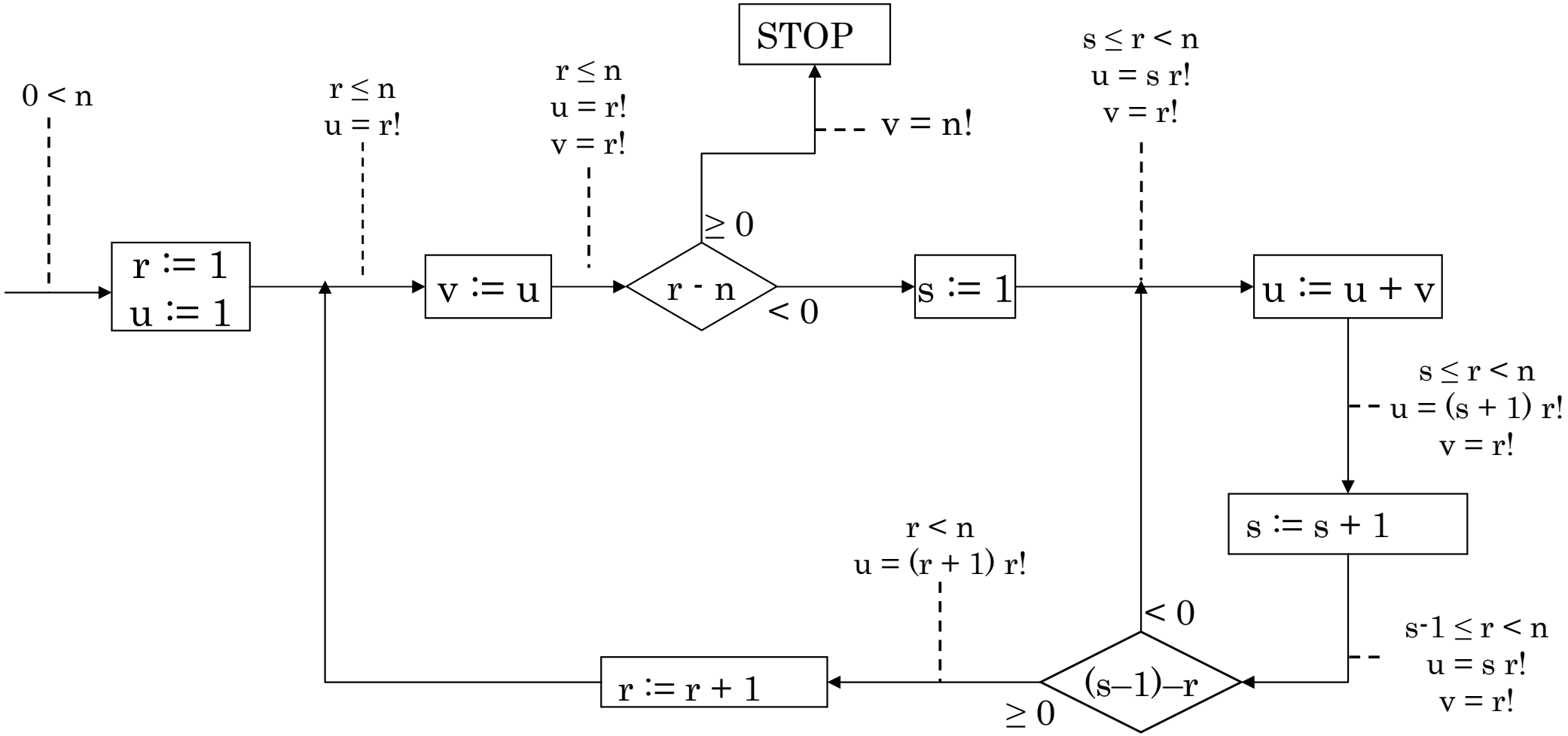


- Äußere Schleife: n Durchläufe
  - Innere Schleife: ≤n Durchläufe
- Laufzeit: $O(n^2)$
- Assertion: Variable wird dekrementiert und wird Null im Stopzustand

Generelles Verfahren

- Flussdiagramm und Tabelle mit Assertions zusammenfassen
- Flussdiagramm nach der Art von Robert Floyd
  - R. Floyd ein Informatiker aus USA
  - Wichtiger Beitrag zum Hoare-Kalkül

Flussdiagramm (Floyd)



Zusammenfassung

- Haben kennen gelernt :
  - Was sind Assertions?

Zusammenfassung

- Haben kennen gelernt :
  - Was sind Assertions?
  - Beweis der Korrektheit mit Hilfe von Assertions
    - Zerlegung des Programms in kleinere Abschnitte
    - Beweis der Korrektheit der einzelnen Abschnitte
    - Folgt Korrektheit des gesamten Programms

Zusammenfassung

- Haben kennen gelernt :
  - Was sind Assertions?
  - Beweis der Korrektheit mit Hilfe von Assertions
    - Zerlegung des Programms in kleinere Abschnitte
    - Beweis der Korrektheit der einzelnen Abschnitte
    - Folgt Korrektheit des gesamten Programms
  - Andeutung eines generellen Verfahrens
    - Assertions in einem Floyd-Diagramm

Zusammenfassung

- Haben kennen gelernt :
  - Was sind Assertions?
  - Beweis der Korrektheit mit Hilfe von Assertions
    - Zerlegung des Programms in kleinere Abschnitte
    - Beweis der Korrektheit der einzelnen Abschnitte
    - Folgt Korrektheit des gesamten Programms
  - Andeutung eines generellen Verfahrens
    - Assertions in einem Floyd-Diagramm
  - Eine 1949 entstandene Arbeit ist ein wichtiger Beitrag zu späteren Programmbeweisen