Einführung in die Informatik I

Kapitel I.5: Verzweigungen

Prof. Dr.-Ing. Marcin Grzegorzek

Juniorprofessur für Mustererkennung im Institut für Bildinformatik

Department Elektrotechnik und Informatik

Fakultät IV der Universität Siegen

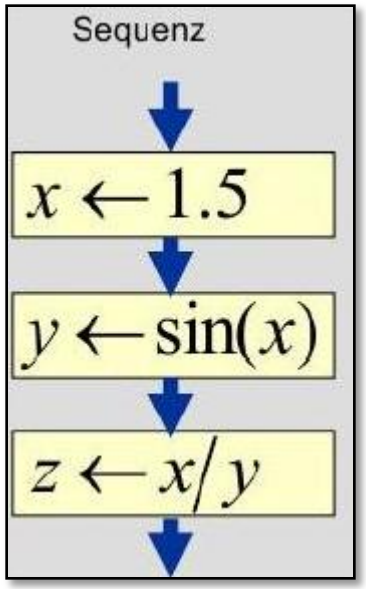
14.11.2012

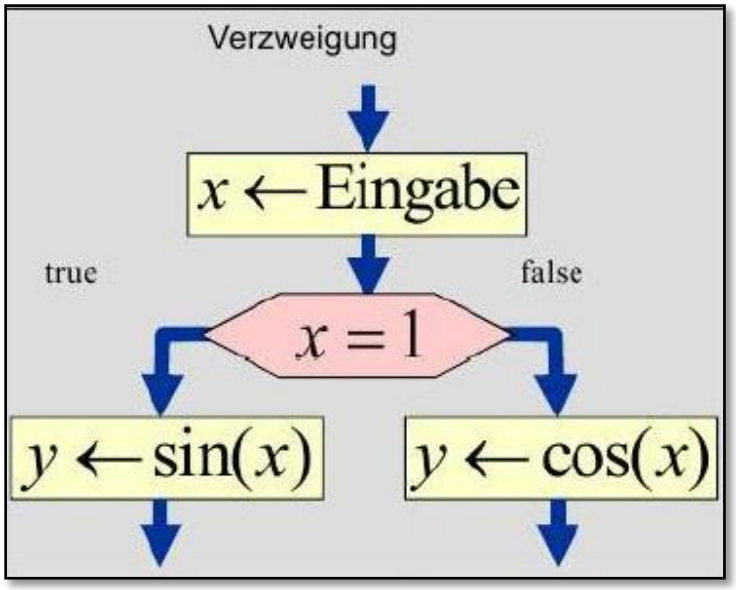
Inhaltsverzeichnis

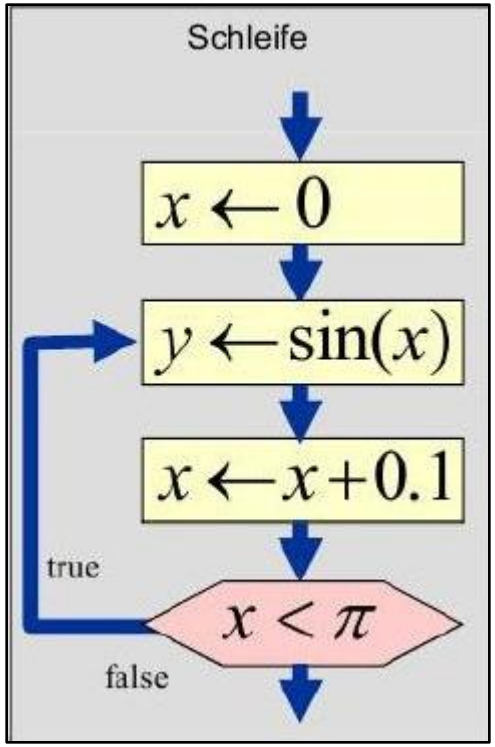
I. MATLAB-Einführung
   1. Voraussetzungen und Konventionen
   2. Variablen und arithmetische Ausdrücke
   3. Automatisierung von Berechnungen
   4. Logische Ausdrücke
   5. Verzweigungen
   6. Schleifen
   7. Fehlersuche in Programmen
   8. Funktionen
   9. Arbeitsweise von Funktionen
   10. Vektoren
   11. Matrizen
II. Algorithmen
III. MATLAB-Fortsetzung
IV. Wissenschaftliche Werkzeuge

Die wichtigsten Kontrollstrukturen

- **Bisher** besteht ein Programm aus einer **festen Sequenz von Anweisungen**: Der Programmablauf ist immer gleich.
- **Kontrollstrukturen** in Programmiersprachen steuern den Programmablauf in Abhängigkeit von aktuellen Variablenwerten.
- Es gibt im wesentlichen **zwei Typen** davon:
  - **Verzweigungen** und
  - **Schleifen**







Verzweigungen

- Eine **Verzweigung** – oder auch bedingter Anweisungsblock – hat in MATLAB die Form

  if *Bedingung*

  Anweisungssequenz

  end

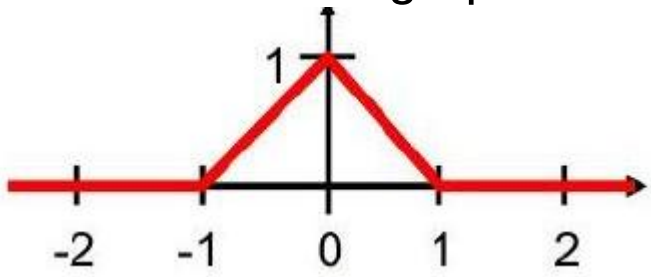
- Die Bedingung muss ein boolescher Ausdruck sein: Ergebnis ist ein Wahrheitswert.
- **Die Anweisungssequenz wird nur dann ausgeführt, wenn die Bedingung wahr (d.h. 1) ist. Falls die Bedingung falsch ist, geht es nach dem *end* weiter.**

Beispiel: Stückweise definierte Funktion

- Mathematische Notation

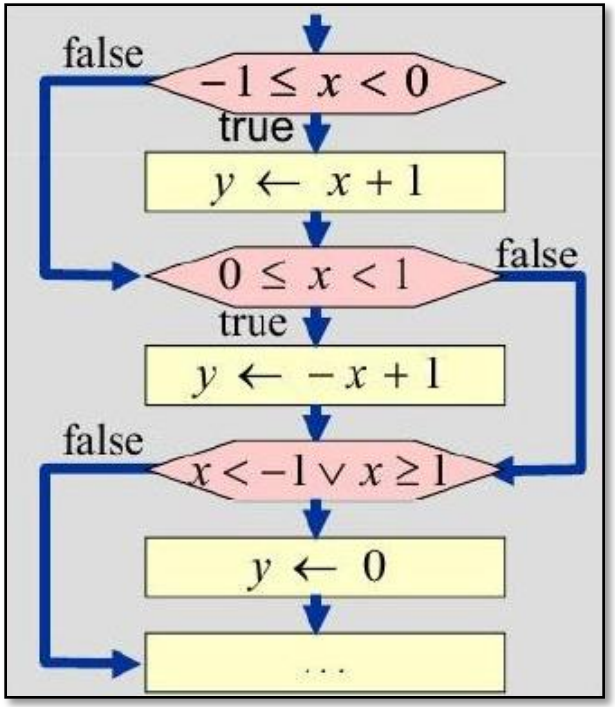
$$f(x)=\begin{cases} x+1, & \text{falls } -1 \le x < 0 \\ -x+1, & \text{falls } 0 \le x < 1 \\ 0 & \text{sonst} \end{cases}$$

Funktionsgraph



- Implementierung mit Hilfe von Verzweigungen

```
if -1<=x & x<0
   y= x+1;
end;
if  0<=x & x<1
   y=-x+1;
end;
if  x<-1 | x>=1
   y=0;
end
```



Verzweigung mit Alternative

- Eine Verzweigung mit Alternative hat in MATLAB die Form

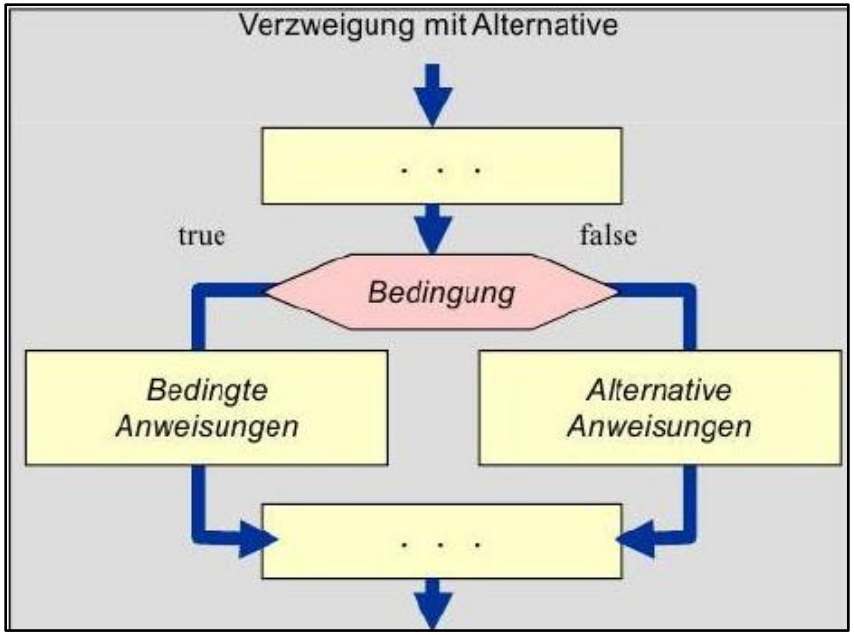
if *Bedingung*

Bedingte Sequenz

else

Alternative Sequenz

end



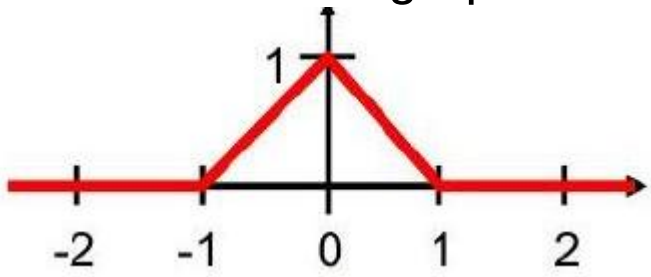
- **Die alternative Sequenz wird genau dann ausgeführt, wenn die Bedingung falsch (d.h. 0) ist. Falls die Bedingung wahr ist, wird die bedingte Sequenz ausgeführt. In beiden Fällen geht es dann nach dem *end* weiter.**

Beispiel: Stückweise definierte Funktion

- Mathematische Notation

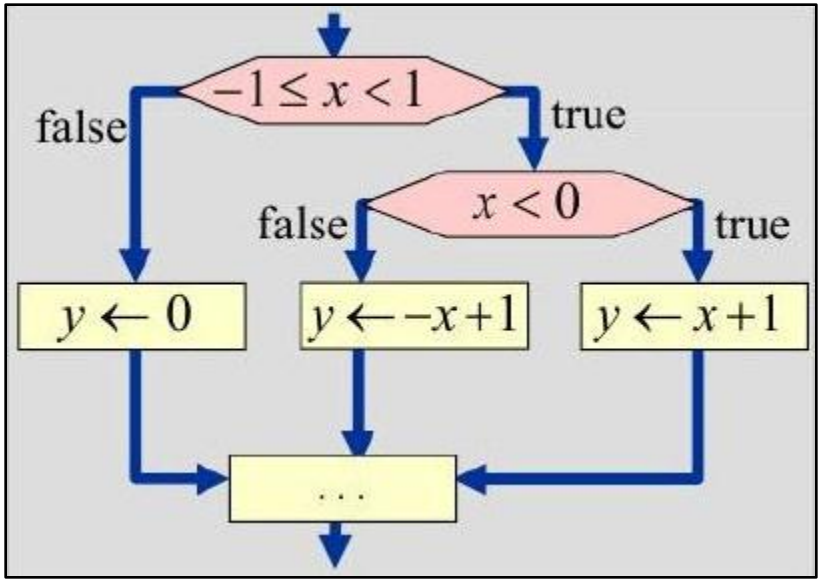
$$f(x)=\begin{cases} x+1, & \text{falls } -1 \le x < 0 \\ -x+1, & \text{falls } 0 \le x < 1 \\ 0 & \text{sonst} \end{cases}$$

Funktionsgraph



- Implementierung mit Hilfe von Verzweigungen

```
if -1<=x & x<1
    if x<0
        y= x+1;
    else
        y=-x+1;
    end
else
    y=0;
end
```



Blockstruktur wird durch Einrücken deutlich

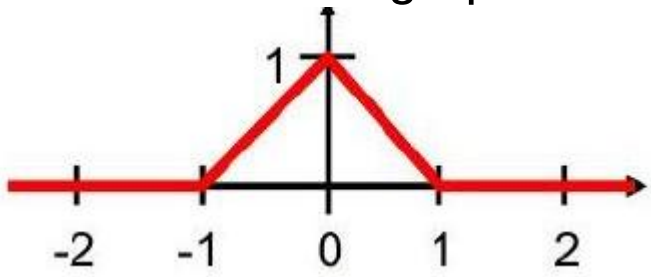
MATLAB unterstützt automatisches Einrücken

Beispiel: Stückweise definierte Funktion

- Mathematische Notation

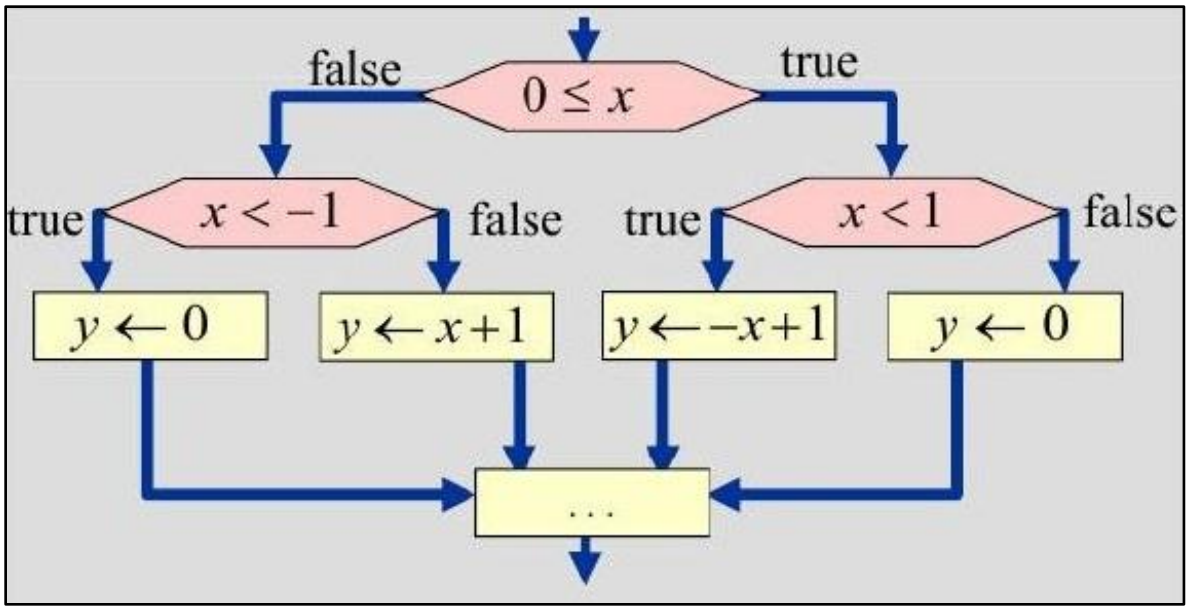
$$f(x)=\begin{cases} x+1, & \text{falls } -1 \le x < 0 \\ -x+1, & \text{falls } 0 \le x < 1 \\ 0 & \text{sonst} \end{cases}$$

Funktionsgraph



- Alternative Implementierung mit Hilfe von Verzweigungen

```
if 0<=x
    if x<1
        y=-x+1;
    else
        y=    0;
    end
else
    if x<-1
        y=    0;
    else
        y= x+1;
    end
end
```



Fallunterscheidungen

- Eine Fallunterscheidung hat in MATLAB die Form:

```
if      Fall 1
        Sequenz 1
elseif  Fall 2
        Sequenz 2
elseif  Fall 3
        Sequenz 3
…
else
        alternative Sequenz
```

- Es darf immer nur einer der Fälle oder die Alternative eintreten.
- Der else-Zweig kann wegfallen.
- Flussdiagramme kennen keine Notation für Fallunterscheidungen.

- Die Fallunterscheidung ist eine Abkürzung für:

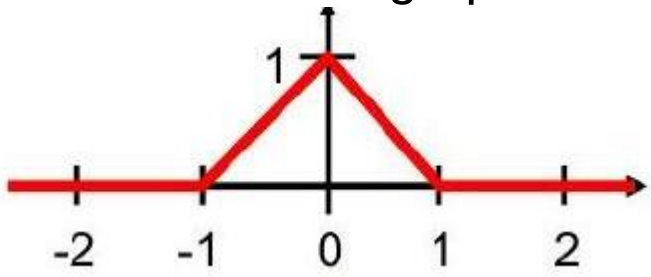
```
if    Fall 1
        Sequenz 1
else
      if   Fall 2
           Sequenz 2
      else
           if  Fall 3
               Sequenz 3
           else
               Alternative Sequenz
           end
      end
end
```

Beispiel: Stückweise definierte Funktion

- Mathematische Notation

$$f(x)=\begin{cases} x+1, & \text{falls } -1 \le x < 0 \\ -x+1, & \text{falls } 0 \le x < 1 \\ 0 & \text{sonst} \end{cases}$$

Funktionsgraph



- Implementierung mit einer Fallunterscheidung

```
if                    x<-1
    y=    0;
elseif -1<=x & x< 0
    y= x+1;
elseif  0<=x & x< 1
    y=-x+1;
else
    y=    0;
end;
```

Programmierstil

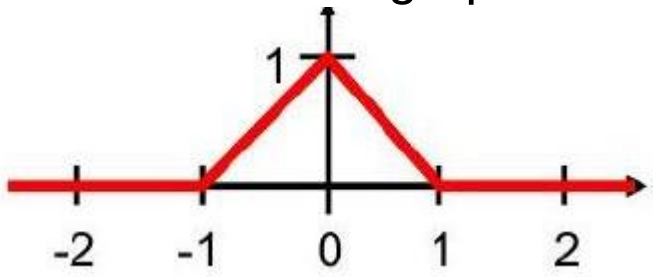
- Viele Programmvarianten können dasselbe Problem lösen.
- Die Problemlösung ist mehr oder weniger übersichtlich.
- Ohne sorgfältiges Einrücken der Blöcke wird ein Programm schnell unlesbar.
- Flussdiagramme werden schnell sehr groß.

Mehrere zutreffende Alternativen

- Mathematische Notation

$$f(x)=\begin{cases} x+1, & \text{falls } -1 \le x < 0 \\ -x+1, & \text{falls } 0 \le x < 1 \\ 0 & \text{sonst} \end{cases}$$

Funktionsgraph



- **Es können mehrere Alternativen zutreffen, trotzdem wird nur eine Alternative ausgeführt.**

  - *x=-1.5* → *y=0.0*
  - *x=-0.5* → *y=0.5*
  - *x=0.5* → *y=0.5*
  - *x=1.5* → *y=0.0*

```
if      x<-1
    y=    0;
elseif  x< 0
    y= x+1;
elseif  x< 1
    y=-x+1;
else
    y=    0;
end;
```

Beispiel EM-Vorrunde 2004

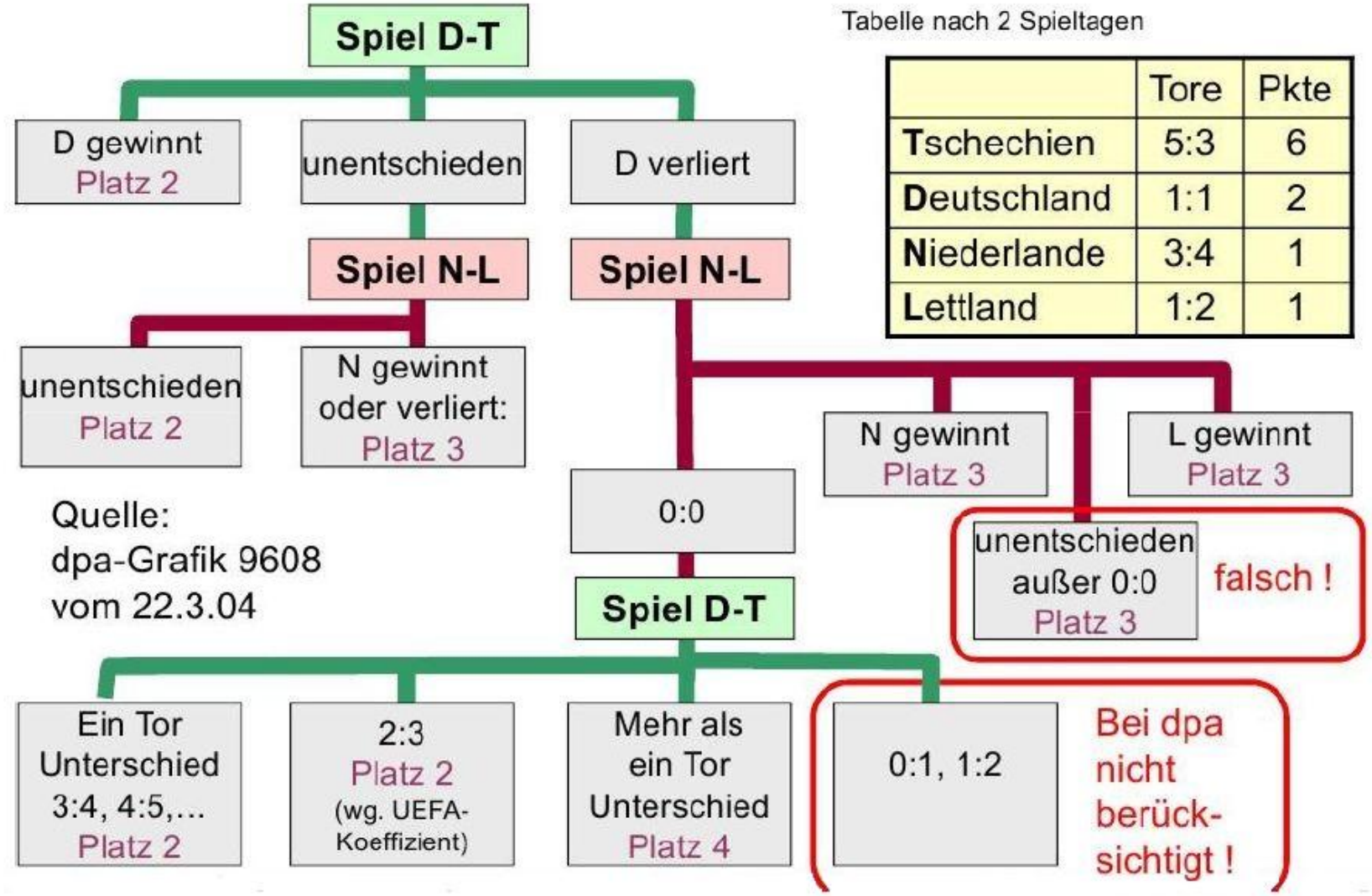


Tabelle nach 2 Spieltagen

	Tore	Pkte
Tschechien	5:3	6
Deutschland	1:1	2
Niederlande	3:4	1
Lettland	1:2	1

Quelle:
dpa-Grafik 9608
vom 22.3.04

Korrekter Entscheidungsbaum

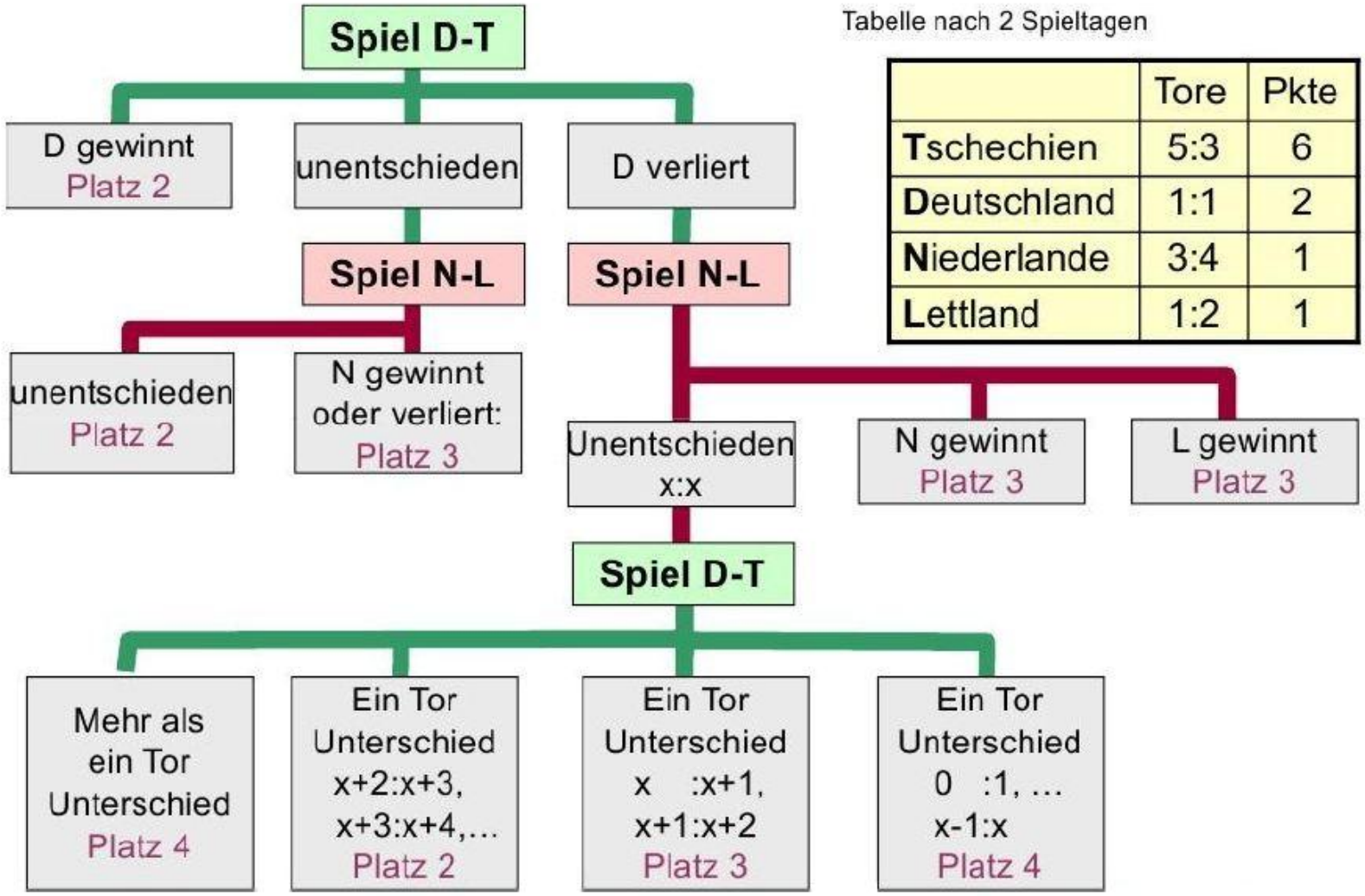
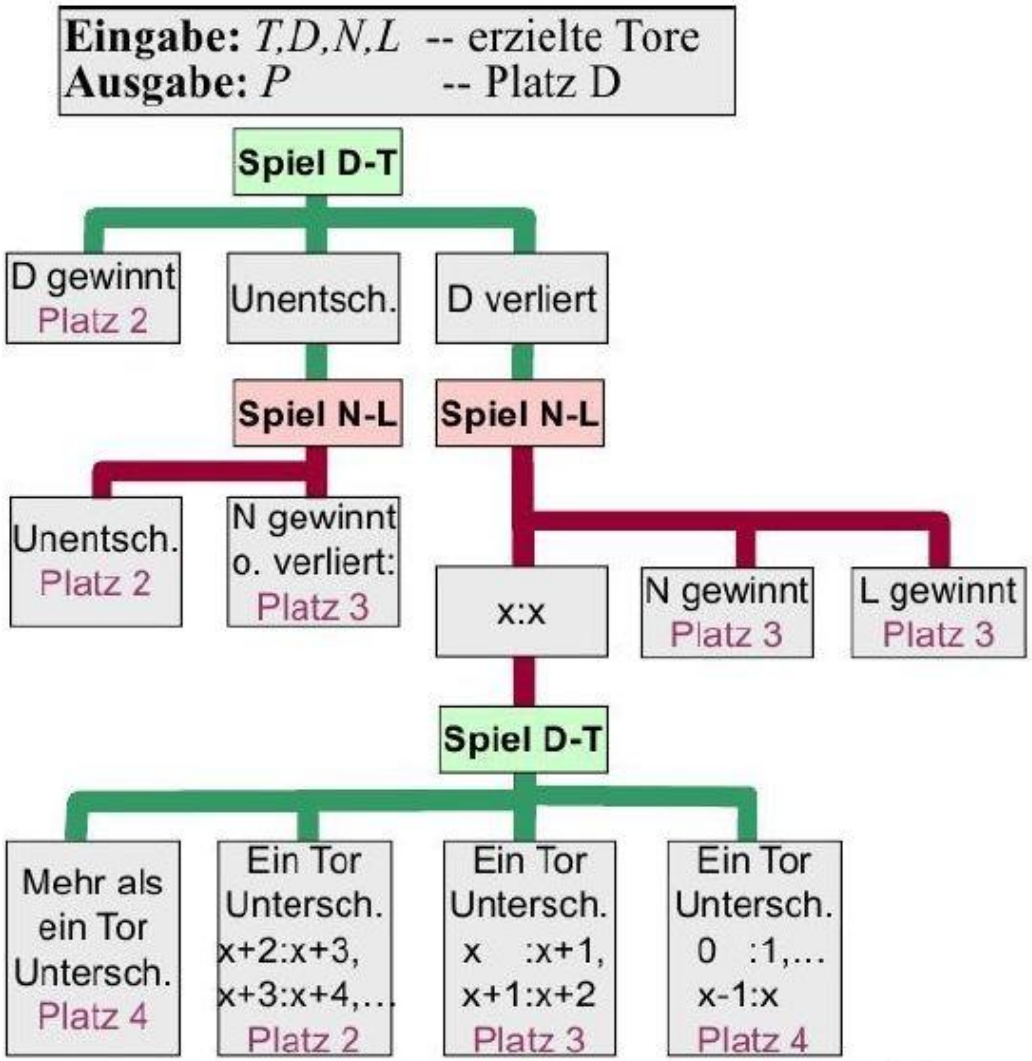


Tabelle nach 2 Spieltagen

	Tore	Pkte
Tschechien	5:3	6
Deutschland	1:1	2
Niederlande	3:4	1
Lettland	1:2	1

Pseudocode für das EM-Beispiel




```
if      D > T   -- D gewinnt
    P ← 2;
elseif  D = T   -- D unentsch.
    if  N = L
        P ← 2;
    else
        P ← 3;
    endif;
else            -- D verliert
    if  N ≠ L   -- N oder L gew.
        P ← 3;
    else        -- N - L unentsch.
        x ← N;
        if      T > D + 1
            P ← 4;
        elseif  D ≥ x + 2
            P ← 2;
        elseif  D ≥ x
            P ← 3
        else
            P ← 4
        endif;
    endif;
endif;
```

Pseudocode im Vergleich zu MATLAB

```
if      D > T   -- D gewinnt
   P ← 2;
elseif  D = T   -- D unentsch.
   if  N = L
      P ← 2;
   else
      P ← 3;
   endif;
else               -- D verliert
   if  N ≠ L       -- N oder L gew.
      P ← 3;
   else            -- N - L unentsch.
      x ← N;
      if      T > D + 1
         P ← 4;
      elseif  D ≥ x + 2
         P ← 2;
      elseif  D ≥ x
         P ← 3
      else
         P ← 4
      endif;
   endif;
endif;
```

```
T = input('Tore Tschechien   : ');
D = input('Tore Deutschland  : ');
N = input('Tore Niederlande  : ');
L = input('Tore Lettland     : ');

if       D>T        % D gewinnt
   P=2;
elseif   D==T       % D unentschieden
   if   N==L
        P=2;
   else
        P=3;
   end;
else                % D verliert
   if      N~=L
      P=3;
   else
      x=N;
      if      T>D+1
         P=4;
      elseif D>=x+2
         P=2;
      elseif D>=x
         P=3
      else
         P=4
      end;
   end;
end;
disp('Platz Deutschland');disp(P);
```