

Mathematical Reasoning (mathematisches Begründen)

Prof. Dr. Josef F. Bürgler

Studiengang Informatik
Hochschule Luzern, Informatik

I.BA_DMATH

Thema: Induktionsbeweis, Rekursion, Inferenzregeln

Ziele: Sie können den Induktionsbeweis auf verschiedene Beweis-Probleme anwenden. Sie verstehen rekursiv definierte Funktionen und rekursive Algorithmen. Sie können die Inferenzregeln praktisch anwenden.

Resultate: Dank der gelernten Methoden können Sie viele weitere mathematische Fragestellungen, wie sie im Informatikalltag auftreten, beweisen.

Vorgehen: Mittels verschiedener Beispiele wird die Theorie vermittelt und vertieft.

- 1 Bereits bekannte Beweismethoden
- 2 Mathematische Induktion
- 3 Rekursiv definierte Funktionen
- 4 Rekursive Algorithmen
- 5 Schlussregeln (Inferenzregeln)
- 6 Korrekte Programme

1 Bereits bekannte Beweismethoden

2 Mathematische Induktion

3 Rekursiv definierte Funktionen

4 Rekursive Algorithmen

5 Schlussregeln (Inferenzregeln)

6 Korrekte Programme

- **Direkter Beweis:** Man zeigt, dass $p \rightarrow q$ wahr ist, indem zeigt, dass aus $p = \text{WAHR}$ sofort $q = \text{WAHR}$ folgt, d.h. die Kombination $p = \text{WAHR}$ und $q = \text{FALSCH}$ kommt nicht vor (siehe Beweis, dass aus “ n ungerade, sofort “ n^2 ungerade” folgt).

- **Direkter Beweis:** Man zeigt, dass $p \rightarrow q$ wahr ist, indem zeigt, dass aus $p = \text{WAHR}$ sofort $q = \text{WAHR}$ folgt, d.h. die Kombination $p = \text{WAHR}$ und $q = \text{FALSCH}$ kommt nicht vor (siehe Beweis, dass aus " n ungerade, sofort " n^2 ungerade" folgt).
- **Beweis durch Kontraposition:** Es handelt sich um einen indirekten Beweis bei welchem man verwendet, dass $p \rightarrow q$ äquivalent zur Kontraposition $\neg q \rightarrow \neg p$ ist (siehe Beweis, dass mit $3n + 2$ ungerade, auch n ungerade ist).

- **Direkter Beweis:** Man zeigt, dass $p \rightarrow q$ wahr ist, indem zeigt, dass aus $p = \text{WAHR}$ sofort $q = \text{WAHR}$ folgt, d.h. die Kombination $p = \text{WAHR}$ und $q = \text{FALSCH}$ kommt nicht vor (siehe Beweis, dass aus “ n ungerade, sofort n^2 ungerade” folgt).
- **Beweis durch Kontraposition:** Es handelt sich um einen indirekten Beweis bei welchem man verwendet, dass $p \rightarrow q$ äquivalent zur Kontraposition $\neg q \rightarrow \neg p$ ist (siehe Beweis, dass mit $3n + 2$ ungerade, auch n ungerade ist).
- **Beweis durch Widerspruch:** Wir möchten zeigen, dass p wahr ist. Nehmen wir an, wir finden einen Widerspruch q so, dass $\neg p \rightarrow q$ wahr ist. Weil q falsch ist, aber $\neg p \rightarrow q$ wahr, muss notwendigerweise $\neg p$ falsch sein (Wahrheitstabelle!), d.h. p muss wahr sein (siehe Beweis, dass $\sqrt{2} \notin \mathbb{Q}$).

P	$\neg p$	q	$\neg p \rightarrow q$
f	w	f	f
w	f	f	w

1 Bereits bekannte Beweismethoden

2 Mathematische Induktion

3 Rekursiv definierte Funktionen

4 Rekursive Algorithmen

5 Schlussregeln (Inferenzregeln)

6 Korrekte Programme

Viele Sätze sagen aus, dass eine bestimmte propositionale Funktion $P(n)$ wahr ist für $n \in \mathbb{N}$, d.h. für alle positiven Zahlen n . Beispielsweise könnte $P(n)$ die folgende Aussage sein

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

letztes Glied *1. Glied*
Anzahl Glieder

Der Beweis, dass $P(k)$ wahr ist $\forall k \in \mathbb{N}$ erfolgt in zwei Schritten:

Viele Sätze sagen aus, dass eine bestimmte propositionale Funktion $P(n)$ wahr ist für $n \in \mathbb{N}$, d.h. für alle positiven Zahlen n . Beispielsweise könnte $P(n)$ die folgende Aussage sein

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

Der Beweis, dass $P(k)$ wahr ist $\forall k \in \mathbb{N}$ erfolgt in zwei Schritten:

Induktionsverankerung: Es wird gezeigt, dass $P(1)$ wahr ist.

Viele Sätze sagen aus, dass eine bestimmte propositionale Funktion $P(n)$ wahr ist für $n \in \mathbb{N}$, d.h. für alle positiven Zahlen n . Beispielsweise könnte $P(n)$ die folgende Aussage sein

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

Der Beweis, dass $P(k)$ wahr ist $\forall k \in \mathbb{N}$ erfolgt in zwei Schritten:

Induktionsverankerung: Es wird gezeigt, dass $P(1)$ wahr ist.

Induktionsschritt: Es wird gezeigt, dass die Implikation $P(k) \rightarrow P(k + 1)$ wahr ist $\forall k \geq 1$.

Viele Sätze sagen aus, dass eine bestimmte propositionale Funktion $P(n)$ wahr ist für $n \in \mathbb{N}$, d.h. für alle positiven Zahlen n . Beispielsweise könnte $P(n)$ die folgende Aussage sein

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

Der Beweis, dass $P(k)$ wahr ist $\forall k \in \mathbb{N}$ erfolgt in zwei Schritten:

Induktionsverankerung: Es wird gezeigt, dass $P(1)$ wahr ist.

Induktionsschritt: Es wird gezeigt, dass die Implikation $P(k) \rightarrow P(k + 1)$ wahr ist $\forall k \geq 1$.

Man nennt $P(k)$ auch **Induktionshypothese**.

Viele Sätze sagen aus, dass eine bestimmte propositionale Funktion $P(n)$ wahr ist für $n \in \mathbb{N}$, d.h. für alle positiven Zahlen n . Beispielsweise könnte $P(n)$ die folgende Aussage sein

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

Der Beweis, dass $P(k)$ wahr ist $\forall k \in \mathbb{N}$ erfolgt in zwei Schritten:

Induktionsverankerung: Es wird gezeigt, dass $P(1)$ wahr ist.

Induktionsschritt: Es wird gezeigt, dass die Implikation $P(k) \rightarrow P(k + 1)$ wahr ist $\forall k \geq 1$.

Man nennt $P(k)$ auch **Induktionshypothese**.

Die Kurzform des Induktionsbeweises heisst:

$$[P(1) \wedge \forall k (P(k) \rightarrow P(k + 1))] \rightarrow \forall n P(n)$$

Example

Beweisen Sie die folgenden Aussagen durch Induktion.

^aEin Triomino (od. Trimino) ist ein L-förmiges Gebiet bestehend aus 3 Quadraten.

Einige Beispiele zum Induktionsbeweis

Example

Beweisen Sie die folgenden Aussagen durch Induktion.

① $\forall n \in \mathbb{N} \left(\underbrace{1 + 2 + 3 + \dots + n}_{P(n)} = \frac{n(n+1)}{2} \right)$

1. induktionsverankerung: Wir zeigen, dass $P(1)$ wahr ist! $P(1)$ ist

$$1 \cancel{\neq} \frac{1(1+1)}{2} = 1$$

2. induktionsvoraussetzung: Wir zeigen jetzt: Wenn $P(k)$ wahr ist, dann ist auch $P(k+1)$ wahr.

$$\underbrace{1 + 2 + 3 + \dots + k}_{k(k+1)/2} + \underline{k+1} = \frac{k(k+1)}{2} + \frac{2(k+1)}{2} = \underline{\frac{(k+1)(k+2)}{2}}$$

^aEin Triomino (od. Trimino) ist ein L-förmiges Gebiet bestehend aus 3 Quadraten.

Einige Beispiele zum Induktionsbeweis

Example

Beweisen Sie die folgenden Aussagen durch Induktion.

$$\textcircled{1} \quad \forall n \in \mathbb{N} \left(1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2} \right)$$

$$\textcircled{2} \quad \forall n \in \mathbb{N} \left(\underbrace{7^0 + 7^1 + \dots + 7^{n-1}}_{P(n)} = \frac{7^n - 1}{7 - 1} \right)$$

1. Induktionsurverankerung: $P(1)$ gilt weil für $n=1$ hat man $7^0 \neq \frac{7^1 - 1}{7 - 1} = 1$

2. Induktionsurzufritt:

$$\underbrace{7^0 + 7^1 + \dots + 7^{k-1} + 7^k}_{P(k) \text{ ist wahr}} = \frac{7^k - 1}{7 - 1} + \frac{(7-1)7^k}{7-1} = \frac{\cancel{7^k-1} + 7^{k+1} - \cancel{7^k}}{7-1} = \frac{7^{k+1} - 1}{7-1}$$

$$\frac{7^k - 1}{7 - 1}$$

Also ist auch
 $P(k+1)$ wahr.

^aEin Triomino (od. Trimino) ist ein L-förmiges Gebiet bestehend aus 3 Quadraten.

Example

Beweisen Sie die folgenden Aussagen durch Induktion.

① $\forall n \in \mathbb{N} \left(1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \right)$

② $\forall n \in \mathbb{N} \left(7^0 + 7^1 + \cdots + 7^{n-1} = \frac{7^n - 1}{7 - 1} \right)$

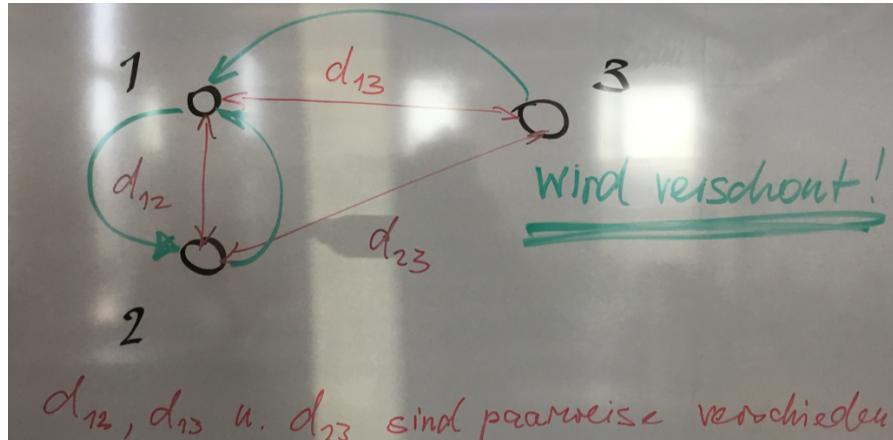
- ③ Eine ungerade Anzahl Personen stehen in jeweils paarweise unterschiedlicher Distanz zueinander. Exakt zum Zeitpunkt $t_0 = 0$ bewirft jeder seinen nächsten Nachbarn mit einer Torte. Zeigen Sie, dass dabei eine Person *verschont* wird. Bei einer geraden Anzahl Personen kann man das nicht zeigen!

Sei die Anzahl Personen $2n+1$. Weiter sei

^aEin Triomino (od. Trimino) ist ein L-förmiges Gebiet bestehend aus 3 Quadraten.

$P(n)$ = "Eine von $2n+1$ Personen wird verschont"

Wir haben bereits die Induktionsverankerung gemacht, d.h. gezeigt, dass $P(1)$ wahr ist.



2. Induktionsabschluß: Dazu müssen wir zeigen, dass wenn $P(n)$ wahr ist, dann ist auch $P(n+1)$ wahr.



A, B : die am nächsten
beieinander stehenden
Personen

Es gibt **zwei Fälle**.

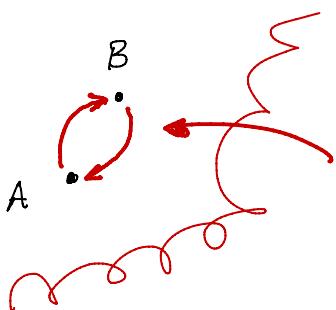
1. Keine der Torten aus der Wolke der $2n+1$ Personen fliegt aus der Wolke.

Da sich A und B gegenüber bewerfen und keine

Torte aus der Wolke raus bzw. reinfliegt, können wir für die "Wolke" die Induktionshypothese anwenden, d.h. eine von den $2n+1$ Personen in der Wolke bleibt verschont.

2. Mind. eine Torte fliegt aus der Wolke raus. Somit

verbleiben in der Wolke $2n$ Torten für $2n+1$ Personen. Also wird eine Person verschont.



q.e.d; w.z.b.w

Example

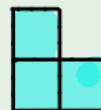
Beweisen Sie die folgenden Aussagen durch Induktion.

① $\forall n \in \mathbb{N} \left(1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \right)$

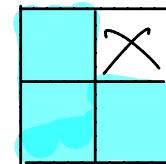
② $\forall n \in \mathbb{N} \left(7^0 + 7^1 + \cdots + 7^{n-1} = \frac{7^n - 1}{7 - 1} \right)$

- ③ Eine ungerade Anzahl Personen stehen in jeweils paarweise unterschiedlicher Distanz zueinander. Exakt zum Zeitpunkt $t_0 = 0$ bewirft jeder seinen nächsten Nachbarn mit einer Torte. Zeigen Sie, dass dabei eine Person *verschont* wird. Bei einer geraden Anzahl Personen kann man das nicht zeigen!
- ④ Entfernt man in einem $2^n \times 2^n$ Schachbrett ein Feld, dann lässt sich der Rest mit (Rechts-) Triominos ^a vollständig zudecken.

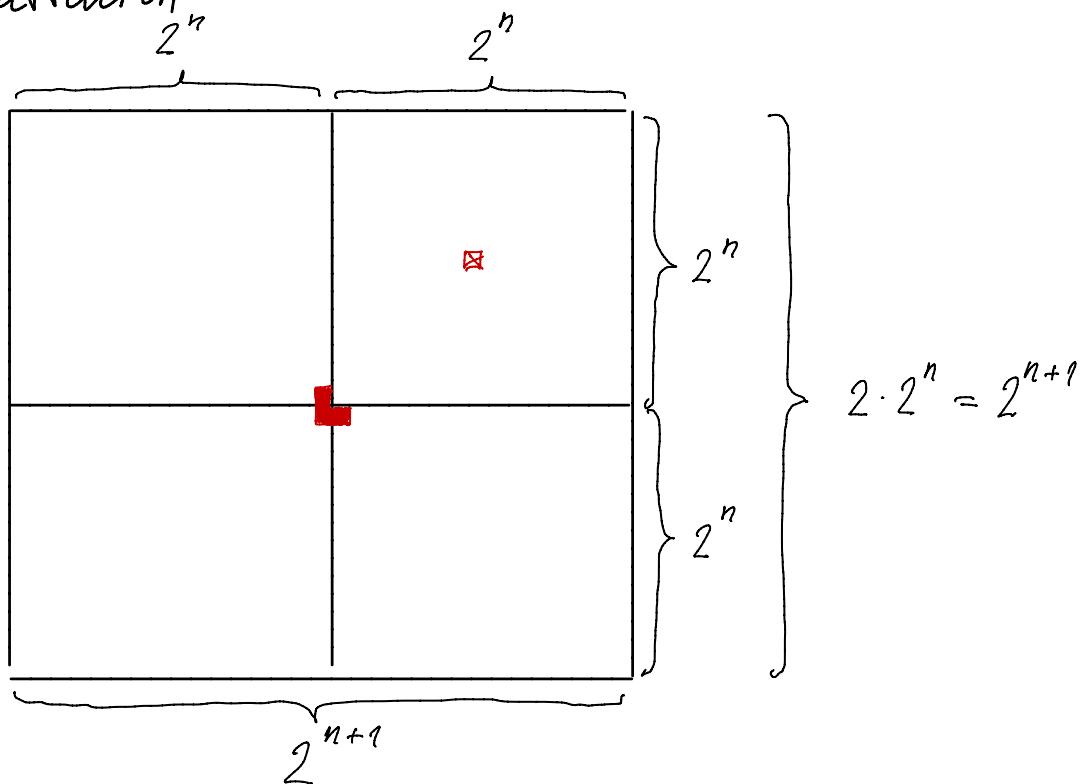
^aEin Triomino (od. Trimino) ist ein L-förmiges Gebiet bestehend aus 3 Quadraten.



1. Induktionsverankerung: Es genügt
ein einziger Triomino



2. Induktionsschritt



1 Bereits bekannte Beweismethoden

2 Mathematische Induktion

3 Rekursiv definierte Funktionen

4 Rekursive Algorithmen

5 Schlussregeln (Inferenzregeln)

6 Korrekte Programme

Definition

Ist f eine Funktion mit Definitionsbereich $D(f) = \mathbb{Z}^+ \cup \{0\}$, für die $f(0)$ definiert ist und für die eine Vorschrift existiert, die den Wert $f(k)$ aus $f(k-1), f(k-2), \dots, f(1), f(0)$ berechnet, dann wird diese Definition **rekursive** oder **induktive Definition** genannt.

Definition

Ist f eine Funktion mit Definitionsbereich $D(f) = \mathbb{Z}^+ \cup \{0\}$, für die $f(0)$ definiert ist und für die eine Vorschrift existiert, die den Wert $f(k)$ aus $f(k-1), f(k-2), \dots, f(1), f(0)$ berechnet, dann wird diese Definition **rekursive** oder **induktive Definition** genannt.

Example (Die Fibonacci Zahlen)

Die Fibonacci Zahlen f_0, f_1, f_2, \dots werden rekursiv definiert durch

$$f_0 = 0 \text{ und } f_1 = 1$$

$$f_k = f_{k-1} + f_{k-2}, \text{ für } k = 2, 3, 4, \dots$$

Man berechne die ersten 6 Glieder dieser Zahlenfolge.

0 1 1 2 3 5 8 13 21 ...

Rekursiv definierte Funktionen (Fort.)

Example

Beweisen Sie die folgende Behauptung über die Fibonacci-Zahlen durch Induktion: Für alle $n \geq 1$ gilt

$$\overbrace{f_1^2 + f_2^2 + \cdots + f_n^2}^{P(n)} = f_n \cdot f_{n+1}$$

Beweis durch Induktion:

1. Induktionsverankerung : $f_1^2 \stackrel{?}{=} f_1 \cdot f_2$ $1^2 \stackrel{?}{=} 1 \cdot 1$

$$\begin{array}{ccc} f_1 & \stackrel{?}{=} & f_1 \cdot f_2 \\ \parallel & & \parallel \\ 1 & & 1 \end{array}$$

2. Induktionswertschritt :

$$\underbrace{f_1^2 + f_2^2 + \cdots + f_n^2}_{\text{Induktions-} \rightarrow \parallel \text{hypothese:}} + f_{n+1}^2 = f_{n+1}^2 + f_n f_{n+1} = f_{n+1} (\underbrace{f_n + f_{n+1}}_{\substack{\parallel \leftarrow \\ \text{Def. der} \\ \text{Fibonacci-} \\ \text{folge}}})$$
$$= f_{n+1} f_{n+2}$$

$P(n)$ wahr !

$P(n+1)$ ist also auch wahr

Also ist $P(n)$ wahr für alle $n \in \mathbb{N}$.

1 Bereits bekannte Beweismethoden

2 Mathematische Induktion

3 Rekursiv definierte Funktionen

4 Rekursive Algorithmen

5 Schlussregeln (Inferenzregeln)

6 Korrekte Programme

Definition

Ein Algorithmus heisst **rekursiv**, wenn er ein Problem löst, indem er dazu ein (oder mehrere) gleiche, aber kleinere Probleme löst.

Wir betrachten folgende rekursive Algorithmen:

Definition

Ein Algorithmus heisst **rekursiv**, wenn er ein Problem löst, indem er dazu ein (oder mehrere) gleiche, aber kleinere Probleme löst.

Wir betrachten folgende rekursive Algorithmen:

- Fakultät $n!$ sowohl rekursiv, wie auch iterativ

Definition

Ein Algorithmus heisst **rekursiv**, wenn er ein Problem löst, indem er dazu ein (oder mehrere) gleiche, aber kleinere Probleme löst.

Wir betrachten folgende rekursive Algorithmen:

- Fakultät $n!$ sowohl rekursiv, wie auch iterativ
- $n.$ Potenz a^n

Definition

Ein Algorithmus heisst **rekursiv**, wenn er ein Problem löst, indem er dazu ein (oder mehrere) gleiche, aber kleinere Probleme löst.

Wir betrachten folgende rekursive Algorithmen:

- Fakultät $n!$ sowohl rekursiv, wie auch iterativ
- $n.$ Potenz a^n
- ggT

Definition

Ein Algorithmus heisst **rekursiv**, wenn er ein Problem löst, indem er dazu ein (oder mehrere) gleiche, aber kleinere Probleme löst.

Wir betrachten folgende rekursive Algorithmen:

- Fakultät $n!$ sowohl rekursiv, wie auch iterativ
- $n.$ Potenz a^n
- ggT
- Fibonacci Zahlen ebenfalls rekursiv und iterativ

Example (Iterative Fakultät)

Wir nehmen an, der Funktion `factorial` werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

Example (Iterative Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
}  
}
```

Example (Iterative Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    int fact = 1;  
  
}  
}
```

Example (Iterative Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    int fact = 1;  
    for (int i = 1; i <= n; i++) {  
  
    }  
  
}
```

Example (Iterative Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    int fact = 1;  
    for (int i = 1; i <= n; i++) {  
        fact = fact*i;  
    }  
}
```

Example (Iterative Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    int fact = 1;  
    for (int i = 1; i <= n; i++) {  
        fact = fact*i;  
    }  
    return fact;  
}
```

Example (Iterative Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    int fact = 1;  
    for (int i = 1; i <= n; i++) {  
        fact = fact*i;  
    }  
    return fact;  
}
```

Beachte: die Funktion ruft sich nicht selber auf. Zudem kommt auch für $n = 0$ das richtige Resultat raus.

Example (Rekursive Fakultät)

Wir nehmen an, der Funktion `factorial` werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

Example (Rekursive Fakultät)

Wir nehmen an, der Funktion `factorial` werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
}  
}
```

Example (Rekursive Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    if ( n == 0 ) {  
  
    } else {  
  
    }  
}
```

Example (Rekursive Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    if ( n == 0 ) {  
        return 1;  
    } else {  
  
    }  
}
```

Example (Rekursive Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    if ( n == 0 ) {  
        return 1;  
    } else {  
        return n*factorial(n-1);  
    }  
}
```

Example (Rekursive Fakultät)

Wir nehmen an, der Funktion factorial werde eine Zahl $n \in \mathbb{N}_0$ übergeben!

```
int factorial(int n) {  
    if ( n == 0 ) {  
        return 1;  
    } else {  
        return n*factorial(n-1);  
    }  
}
```

Beachte: die Funktion ruft sich immer wieder selber auf; allerdings nicht unendlich lange, sondern nur bis das Argument Null wird!

Rekursiver Euklidischer Algorithmus

```
int gcd (int u, int v) {  
    return (v != 0) ? gcd(v, u%v); u;  
}
```

Iterativer Eukl. Algo.:

```
int gcd_iter (int u, int v) {  
    int t;  
    while (v) {  
        t = u;  
        u = v;  
        v = t % v;  
    }  
    return u < 0 ? -u : u; /* abr(u) */  
}
```

$\begin{array}{l} t \\ \parallel \\ u = b \cdot v + t \% v \\ u = t \% v \end{array}$

1 Bereits bekannte Beweismethoden

2 Mathematische Induktion

3 Rekursiv definierte Funktionen

4 Rekursive Algorithmen

5 Schlussregeln (Inferenzregeln)

6 Korrekte Programme

Schlussregeln (Inferenzregeln, rules of inference)

Bei mathematischen Beweisen geht es darum mit gültigen Argumenten die Richtigkeit einer mathematischen Aussage zu begründen.

Ein **Argument** ist eine Folge von Aussagen, die mit einer **Folgerung** (Konklusion) enden.

gültig heisst, dass die Folgerung (Konklusion) aus der Wahrheit der vorhergehenden Aussagen (den **Prämissen**) des Arguments folgt.

Ein Argument ist also dann und nur dann gültig, wenn es unmöglich ist, dass alle Prämissen wahr sind, die Folgerung aber falsch ist.

Frage: Was sind gültige Argumente in der propositionalen Logik?

Example

Betrachte folgendes Argument welches Propositionen einbezieht

Example

Betrachte folgendes Argument welches Propositionen einbezieht

- “Falls Sie ein aktuelles Passwort haben, dann können Sie sich ins Netzwerk einloggen”

Example

Betrachte folgendes Argument welches Propositionen einbezieht

- “Falls Sie ein aktuelles Passwort haben, dann können Sie sich ins Netzwerk einloggen”
- “Sie haben eine aktuelle Passwort”

Example

Betrachte folgendes Argument welches Propositionen einbezieht

- “Falls Sie ein aktuelles Passwort haben, dann können Sie sich ins Netzwerk einloggen”
- “Sie haben eine aktuelle Passwort”
- Demzufolge

Example

Betrachte folgendes Argument welches Propositionen einbezieht

- “Falls Sie ein aktuelles Passwort haben, dann können Sie sich ins Netzwerk einloggen”
- “Sie haben eine aktuelle Passwort”
- Demzufolge
- “Sie können sich ins Netzwerk einloggen”

Example

Betrachte folgendes Argument welches Propositionen einbezieht

- “Falls Sie ein aktuelles Passwort haben, dann können Sie sich ins Netzwerk einloggen”
- “Sie haben eine aktuelle Passwort”
- Demzufolge
- “Sie können sich ins Netzwerk einloggen”

Ist das ein gültiges Argument? Wir möchten bestimmen, ob die Folgerung “Sie können sich ins Netzwerk einloggen” wahr ist, wenn die Prämissen “Falls Sie ein aktuelles Passwort haben, dann können Sie sich ins Netzwerk einloggen” und “Sie haben eine aktuelle Passwort” beide wahr sind!

Bevor wir die Gültigkeit dieses bestimmten Argumentes diskutieren wollen wir eine Kurzschreibweise einführen.

Example (Fort.)

Seien also

Example (Fort.)

Seien also

- p = "Sie haben ein aktuelles Passwort"

Example (Fort.)

Seien also

- p = "Sie haben ein aktuelles Passwort"
- q = "Sie können sich ins Netzwerk einloggen"

Example (Fort.)

Seien also

- p = "Sie haben ein aktuelles Passwort"
- q = "Sie können sich ins Netzwerk einloggen"

Dann kann man das Argument in folgender Form schreiben

$$\begin{array}{c} p \rightarrow q \\ p \\ \hline \therefore q \end{array}$$

Das Symbol \therefore steht für "demzufolge" oder "folglich".

Schlussregeln (Fort.)

Wir wissen, dass $((p \rightarrow q) \wedge p) \rightarrow q$ eine Tautologie ist (überprüfen Sie das). Wenn also $p \rightarrow q$ und p wahr sind, dann muss auch q wahr sein!

Sobald also alle Prämissen (die Aussagen des Arguments ausser dem letzten) wahr sind, dann muss auch die Folgerung bzw. die Konklusion (die letzte Aussage im Argument) wahr sein. Eine solche Form des Arguments nennt man gültig.

Mit den Rechenregeln (Hauptsatz.)

Man hat also folgende Schlussregeln (Inferenzregeln):

- **Modus ponens** (Abtrennungsregel)

$$\begin{array}{c} p \\ p \rightarrow q \\ \hline \therefore q \end{array}$$

Tautologie : $[p \wedge (p \rightarrow q)] \rightarrow q$

P	q	$P \rightarrow q$	$\overbrace{p \wedge (p \rightarrow q)}^A$	$A \rightarrow q$
w	w	w	w	w
w	f	f	f	w
f	w	w	f	w
f	f	w	f	w

Schlussregeln (Fort.)

- **Modus tollens** (Aufhebender Modus)

$$\frac{\neg q \\ p \rightarrow q}{\therefore \neg p}$$

Tautologie : $[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$

Schlussregeln (Fort.)

- **Modus tollens** (Aufhebender Modus)

$$\frac{\begin{array}{c} \neg q \\ p \rightarrow q \end{array}}{\therefore \neg p}$$

Tautologie : $[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$

- **Hypothetischer Syllogismus**

$$\frac{\begin{array}{c} p \rightarrow q \\ q \rightarrow r \end{array}}{\therefore p \rightarrow r}$$

Tautologie : $[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$

Schlussregeln (Fort.)

- **Modus tollens** (Aufhebender Modus)

$$\frac{\begin{array}{c} \neg q \\ p \rightarrow q \end{array}}{\therefore \neg p}$$

Tautologie : $[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$

- **Hypothetischer Syllogismus**

$$\frac{\begin{array}{c} p \rightarrow q \\ q \rightarrow r \end{array}}{\therefore p \rightarrow r}$$

Tautologie : $[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$

- **Disjunktiver Syllogismus**

$$\frac{\begin{array}{c} p \vee q \\ \neg p \end{array}}{\therefore q}$$

Tautologie : $[(p \vee q) \wedge \neg p] \rightarrow q$

- **Addition**

$$\frac{p}{\therefore p \vee q}$$

Tautologie : $p \rightarrow (p \vee q)$

Schlussregeln (Fort.)

- **Addition**

$$\begin{array}{c} p \\ \hline \therefore p \vee q \end{array}$$

Tautologie : $p \rightarrow (p \vee q)$

- **Simplifikation**

$$\begin{array}{c} p \wedge q \\ \hline \therefore p \end{array}$$

Tautologie : $(p \wedge q) \rightarrow p$

Schlussregeln (Fort.)

- **Addition**

$$\begin{array}{c} p \\ \hline \therefore p \vee q \end{array}$$

Tautologie : $p \rightarrow (p \vee q)$

- **Simplifikation**

$$\begin{array}{c} p \wedge q \\ \hline \therefore q \end{array}$$

Tautologie : $(p \wedge q) \rightarrow p$

- **Konjunktion**

$$\begin{array}{c} p \\ q \\ \hline \therefore p \wedge q \end{array}$$

Tautologie : $[(p) \wedge (q)] \rightarrow (p \wedge q)$

• Resolution

$$\begin{array}{c} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$$

Tautologie: $[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$

- Resolution

$$\begin{array}{c} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$$

Tautologie : $[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$

Example (Anwendungsbeispiel)

Zeige dass die Hypothesen "es ist heute Nachmittag nicht sonnig und es ist kälter als gestern", "wir gehen nur schwimmen falls es sonnig ist", "wenn wir nicht schwimmen gehen, machen wir eine Kanufahrt" und "wenn wir eine Kanufahrt machen, werden wir bis Sonnenuntergang zu Hause sein" zum Schluss führen: "wir werden bis Sonnenuntergang zu Hause sein"!

Schlussregeln (Fort.)

Example (Anwendungsbeispiel (Fort.))

Wir verwenden:

$p = \text{"Es ist heute Nachmittag sonnig"}$ $q = \text{"Es ist kälter als gestern"}$
 $r = \text{"Wir gehen schwimmen"}$ $s = \text{"Wir machen eine Kanufahrt"}$
 $t = \text{"Wir sind bis Sonnenuntergang zu Hause"}$

Dann lauten die Hypothesen

$\neg p \wedge q = \text{"es ist heute Nachmittag nicht sonnig und es ist kälter als gestern"}$ (H1)

$\neg r \vee p = r \rightarrow p = \text{"wir gehen nur schwimmen falls es sonnig ist"}$ (H2)

$\neg r \rightarrow s = \text{"wenn wir nicht schwimmen gehen, machen wir eine Kanufahrt"}$ (H3)

$s \rightarrow t = \text{"wenn wir eine Kanufahrt machen, werden wir bis Sonnenuntergang zu Hause sein"}$ (H4)

Example (Anwendungsbeispiel (Fort.))

Wir müssen zeigen, dass aus diesen Hypothesen durch Anwendung der Schlussregeln, der Schluss (bzw. die Konklusion) t folgt.

1. $\neg p \wedge q$ Hypothese **(H1)**
2. $\neg p$ Simplifikation unter Verwendung von (1)
3. $r \rightarrow p$ Hypothese **(H2)**
4. $\neg r$ Modus tollens unter Verwendung von (2) und (3)
5. $\neg r \rightarrow s$ Hypothese **(H3)**
6. s Modus ponens unter Verwendung von (4) und (5)
7. $s \rightarrow t$ Hypothese **(H4)**
8. t Modus ponens unter Verwendung von (6) und (7)

Beachte: Man hätte das Problem auch mit einer Wahrheitstabelle für die fünf Variablen p , q , r , s und t lösen können. Diese Tabelle hätte aber $2^5 = 32$ Zeilen gehabt!

Hypothetischer Syllogismus

$$p \rightarrow q$$

$$q \rightarrow r$$

$$\therefore p \rightarrow r$$

aus der Revolution

$$\begin{array}{c} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array}$$

$$\begin{array}{c} q \vee \neg p \\ \neg q \vee r \\ \hline \therefore \neg p \vee r \end{array}$$

wir verwenden

$$p \rightarrow q \equiv \neg p \vee q,$$

$$q \rightarrow r \equiv \neg q \vee r \quad \text{und}$$

$$p \rightarrow r \equiv \neg p \vee r$$

Setze in der Resolutionsregel $q = r$: Dann hat man

$$\begin{array}{c} p \vee q \\ \neg p \vee q \\ \hline \therefore q \end{array}$$

Falscher dirjunktiver
Syllogismus wenn wir
das zweite q weglassen...
Darf man das? Beweisen!!

Falsch ist $\neg p \vee q \equiv \neg p$!

Die Inferenzregel

$$\begin{array}{c} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$$

stimmt (auch ohne das
 $z q$).

- 1 Bereits bekannte Beweismethoden
- 2 Mathematische Induktion
- 3 Rekursiv definierte Funktionen
- 4 Rekursive Algorithmen
- 5 Schlussregeln (Inferenzregeln)
- 6 Korrekte Programme
(optional – kein Prüfungsvorstoff)

Definition (korrekt)

Ein Programm heisst **korrekt**, falls es den korrekten Output für jeden möglichen Input liefert.

Ein Programm heisst **teilweise korrekt** (engl. **partially correct**), wenn es den korrekten Output liefert, falls es terminiert. Wenn das Programm zudem immer terminiert, dann heisst das Programm **korrekt**.

Definition (korrekt)

Ein Programm heisst **korrekt**, falls es den korrekten Output für jeden möglichen Input liefert.

Ein Programm heisst **teilweise korrekt** (engl. **partially correct**), wenn es den korrekten Output liefert, falls es terminiert. Wenn das Programm zudem immer terminiert, dann heisst das Programm **korrekt**.

Definition (teilweise korrekt)

Ein Programmsegment S heisst **teilweise korrekt** bezüglich der Anfangsbedingung p und der Endbedingung q falls gilt: Ist p wahr für alle Eingaben von S und terminiert S , dann ist q wahr für die Ausgaben von S : man bezeichnet dies mit $p\{S\}q$ (*Hoare Tripel*, nach Tony Hoare).

Example

Zeigen Sie, dass das Programmsegment

$$y := 2, z := x + y$$

bezüglich der Anfangsbedingung $x = 1$ und der Endbedingung $z = 3$ teilweise korrekt ist.

Nehmen wir an p sei wahr, d.h. $x = 1$ zu Beginn. Dann wird y die Zahl 2 zugeteilt und z die Summe von x und y , also 3.

Also ist S (das Programmsegment bestehend aus den zwei Zeilen) korrekt bezüglich der Anfangsbedingung p und der Endbedingung

q. Also ist $p \{S\} q$ korrekt.

Regeln des Schliessens

Hat man zwei Programmsegmente S_1 und S_2 , dann gilt die folgende Regel für das zusammengesetzte Programmsegment $S_1; S_2$:

$$\frac{p \{S_1\} q \\ q \{S_2\} r}{\therefore p \{S_1; S_2\} r}$$

(1) S_1 ist korrekt bezüglich der Aufangsbed. (AB) p und der Endbed. (EB) q .
(2) S_2 ist korrekt bezügl. der AB q und der EB r .

Dann ist $S_1; S_2$ (Zusammensetzung von S_1 und S_2) korrekt bezügl. des AB p und der EB r .

Conditionale Statements

Das Programmsegment S_1 in

```
if (condition) then  
     $S_1$ 
```

wird ausgeführt, falls condition wahr ist. Also hat man:

$$\begin{aligned} & (p \wedge \text{condition}) \{S_1\} q \\ & (p \wedge \neg \text{condition}) \rightarrow q \end{aligned}$$

$$\therefore p \{\text{if condition then } S_1\} q$$

Example

Zeigen Sie, dass das Programmsegment

`if $x > y$ then $y := x$`

bezüglich der Anfangsbedingung T und der Endbedingung $y \geq x$ teilweise korrekt ist.

- Falls die AB True ist und $x \geq y$ dann wird $y := x$ ausgeführt, d.h. die EB $y \geq x$ ist wahr (er ist ja $y = x$ und damit ist $y \geq x$ wahr)
- Falls die AB True ist und $x \not> y$ dann ist ja $x \leq y$ oder eben $y \geq x$, d.h. die EB ist wahr.

Nach obiger Inferenzregel ist dann der Code korrekt bezügl. der gegebenen AB und EB.

Conditionale Statements (Fort.)

Das Programmsegment S_1 im folgenden Code

```
if (condition) then  
    S1  
else  
    S2
```

wird ausgeführt, falls condition wahr ist; ansonsten S_2 . Also gilt:

$$\begin{aligned} &(p \wedge \text{condition})\{S_1\}q \\ &(p \wedge \neg \text{condition})\{S_2\}q \end{aligned}$$

$$\therefore p \{\text{if condition then } S_1 \text{ else } S_2\} q$$

Conditionale Statements (Fort.)

Example

Zeigen Sie, dass das Programmsegment

```
if x < 0 then abs := -x else abs := x
```

bezüglich der Anfangsbedingung T und der Endbedingung $\text{abs} := |x|$ (der Betrag von x) teilweise korrekt ist.

Hier kann man analog zum vorigen Beispiel argumentieren!

Zusammenfassung

- Die StudentIn kennt grundlegende Prinzipien von Beweisen.

Zusammenfassung

- Die StudentIn kennt grundlegende Prinzipien von Beweisen.
- Die StudentIn kann mit Folgen und endlichen Reihen umgehen und einfache, endliche Summen berechnen.

- Die StudentIn kennt grundlegende Prinzipien von Beweisen.
- Die StudentIn kann mit Folgen und endlichen Reihen umgehen und einfache, endliche Summen berechnen.
- Die StudentIn kann einfache Induktionsbeweise führen.

Zusammenfassung

- Die StudentIn kennt grundlegende Prinzipien von Beweisen.
- Die StudentIn kann mit Folgen und endlichen Reihen umgehen und einfache, endliche Summen berechnen.
- Die StudentIn kann einfache Induktionsbeweise führen.
- Die StudentIn versteht rekursiv definierte Funktionen, Mengen, Strukturen und Algorithmen.

- Die StudentIn kennt grundlegende Prinzipien von Beweisen.
- Die StudentIn kann mit Folgen und endlichen Reihen umgehen und einfache, endliche Summen berechnen.
- Die StudentIn kann einfache Induktionsbeweise führen.
- Die StudentIn versteht rekursiv definierte Funktionen, Mengen, Strukturen und Algorithmen.
- Die StudentIn kann einfach Programme auf Korrektheit untersuchen.