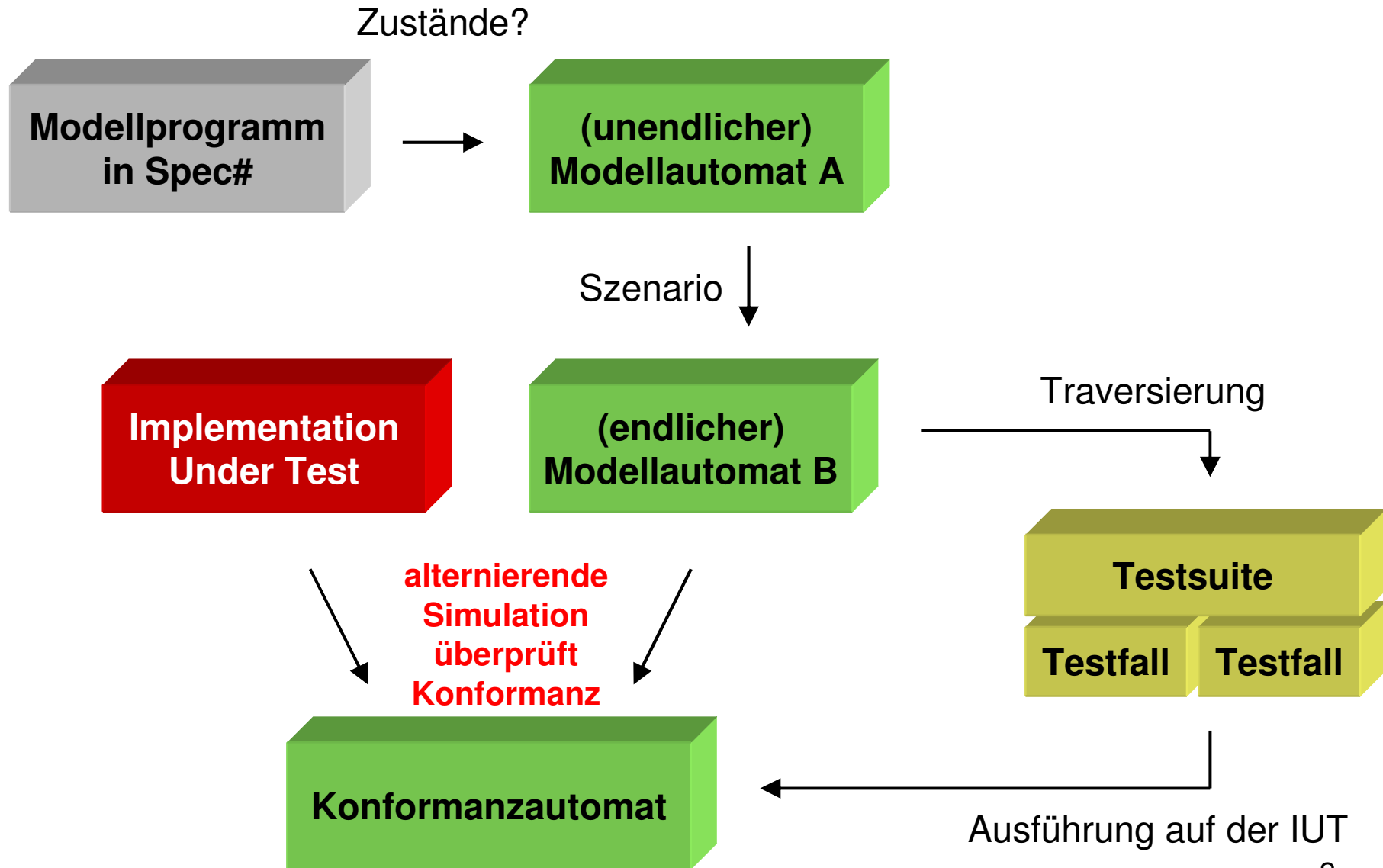


# **Spec# als Testspezifikationsprache**

Andreas Blunk

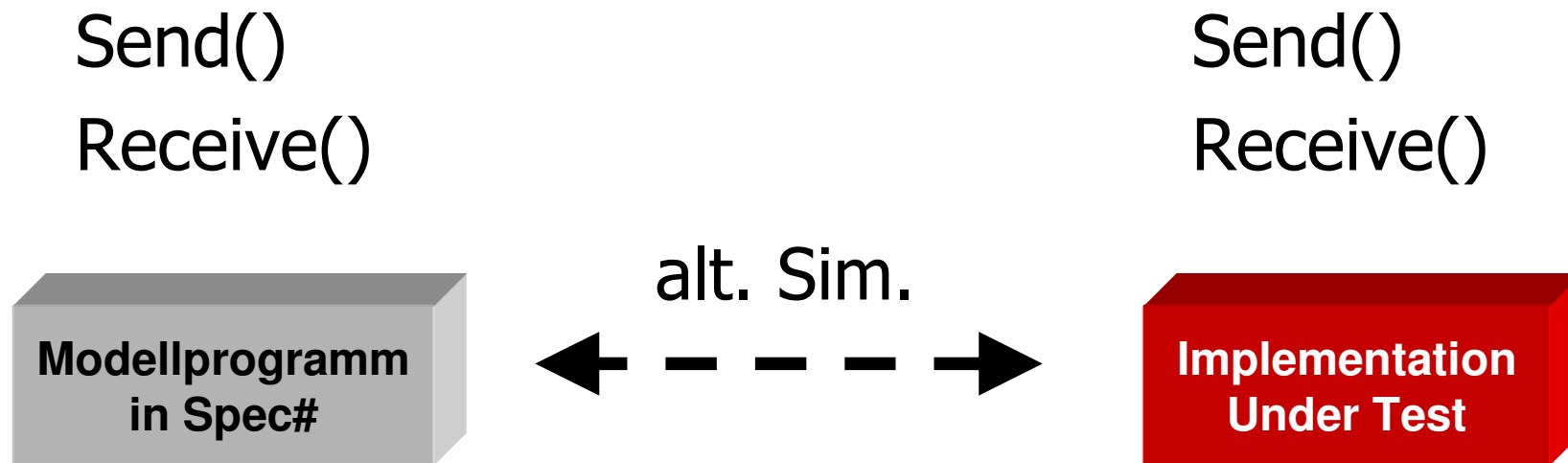
[blunk@informatik.hu-berlin.de](mailto:blunk@informatik.hu-berlin.de)



- **Einfach gesagt:**  
C# angereichert mit Konstrukten zur Spezifikation von Bedingungen (... auf Methoden und Daten)
- **Pre-/Postconditions in Methoden:**  
Kontrakt zwischen dem Rufer einer Methode (pre) und der ausführenden Methode selbst (post)
- **Viele weitere Konstrukte...**

## Beispiel Chat-Client:

- Te
- a)
  - Enter(): Session betreten
  - Send(...): Nachricht an alle Clients senden
  - Receive(...): Nachricht von Client empfangen
- Form: Modellprogramm (in Spec#)
- relevante Zustände identifizieren
- input/output als steuerbare und beobachtbare Aktionen
- welche möglichen Aktionen in best. Zustand (durch Vorbedingungen)
- b) repräsentatives Verhalten extrahieren



## 1

# Beispiel: Modellprogramm Chat-Client

```
class Client {
  bool entered;
  Map<Client,Seq<string>> unreceivedMsgs;

  [Action] Client() {
    this.unreceivedMsgs = Map;
    foreach (Client c in enumof(Client), c != this) {
      c.unreceivedMsgs[this] = Seq{};
      this.unreceivedMsgs[c] = Seq{};
    }
    entered = false;
  }

  [Action] void Enter()
  requires !entered;
  { entered = true; }

  [Action] void Send(string message)
  requires entered;
  {
    foreach (Client c in enumof(Client), c != this, c.entered)
      c.unreceivedMsgs[this] += Seq{message};
  }

  [Action(Kind=ActionAttributeKind.Observable)]
  void Receive(Client sender, string message)
  requires sender != this && unreceivedMsgs[sender].Length > 0 &&
  unreceivedMsgs[sender].Head == message;
  {
    unreceivedMsgs[sender] = unreceivedMsgs[sender].Tail;
  }
}
```

## 1

# Beispiel: Modellprogramm Chat-Client

```
class Client {
  bool entered;
  Map<Client, Seq<string>> unreceivedMsgs;

  [Action] Client() {
    this.unreceivedMsgs = Map();
    foreach (Client c in enumof(Client), c != this) {
      c.unreceivedMsgs[this] = Seq{};
      this.unreceivedMsgs[c] = Seq{};
    }
    entered = false;
  }
}
```

```
class Client {
  bool entered;
  Map<Client, Seq<string>> unreceivedMsgs;
  ...
}
```

```
[Action] void Enter()
```

```
... Client() {
  this.unreceivedMsgs = Map();
  foreach (Client c in enumof(Client), c != this) {
    c.unreceivedMsgs[this] = Seq{};
    this.unreceivedMsgs[c] = Seq{};
  }
  entered = false;
}
```

```
unreceivedMsgs[sender].Head == message;
{
  unreceivedMsgs[sender] = unreceivedMsgs[sender].Tail;
}
}
```

## 1

# Beispiel: Modellprogramm Chat-Client

```

class Client {
  bool entered;
  Map<Client,Seq<string>> unreceivedMsgs;

  [Action] Client() {
    this.unreceivedMsgs = Map;
    foreach (Client c in enumof(Client), c != this) {
      c.unrec
      this.un
    }
    entered =
  }

  [Action] vo
  requires
  { entered =

  [Action] vo
  requires en
  {
    foreach (
      c.unrec

  [Action(Kind=ActionAttributeKind.Observable)]
  void Receive(Client sender, string message)

  [Action(Kind=ActionAttributeKind.Observable)]
  void Receive(Client sender, string message)
  requires sender != this && unreceivedMsgs[sender].Length > 0 &&
  unreceivedMsgs[sender].Head == message;
  {
    unreceivedMsgs[sender] = unreceivedMsgs[sender].Tail;
  }
}
}

```

## 1

# Beispiel: Modellprogramm Chat-Client

```
class Client {
  bool entered;
  Map<Client,Seq<string>> unreceivedMsgs;
```

```
[Action] Client() {
```

```
[Action] void Send(string message)
requires entered;
{
  foreach (Client c in enumof(Client), c != this, c.entered)
    c.unreceivedMsgs[this] += Seq{message};
}
```

```
requires !entered;
```

```
{ entered = true; }
```

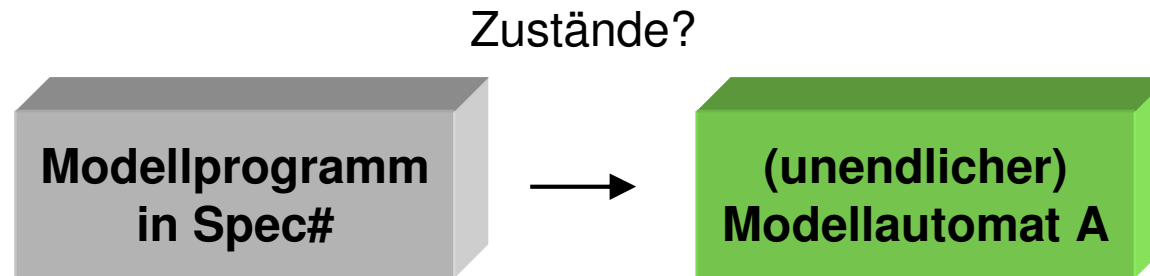
```
[Action(Kind=ActionAttributeKind.Observable)]
void Receive(Client sender, string message)
requires sender != this && unreceivedMsgs[sender].Length > 0
  && unreceivedMsgs[sender].Head == message;
{
  unreceivedMsgs[sender] = unreceivedMsgs[sender].Tail;
}
```

```
unreceivedMsgs[sender].Head == message;
```

```
{
  unreceivedMsgs[sender] = unreceivedMsgs[sender].Tail;
```

```
}
```

```
}
```



- können Modellprogramm als Automaten auffassen
- Zustände (inkl. Start- und Endzustände)
- Transitionen:  
Übergang von einem Zustand in einen Folgezustand durch Aktionen
- u.U. unendliche viele Zustände

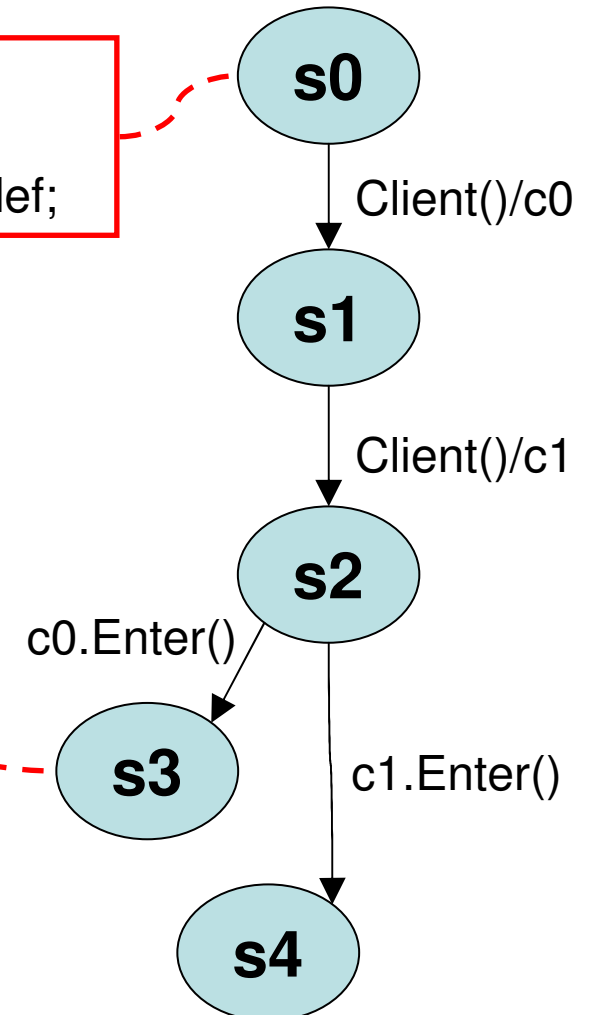
# Beispiel: Zustände

## Zustand s0:

enumof(Client) = Seq{}  
für alle x: entered(x) = undef;

## Zustand s3:

enumof(Client) = Seq{c0, c1}  
entered(c0) = true;  
entered(c1) = false;  
sonst: entered(x) = undef;



**Knoten** = individuelle Zustände

**Pfeile** = Transitionen, die den Zustand ändern

**Ovale** = aktive Zustände

**Diamanten** = passive Zustände

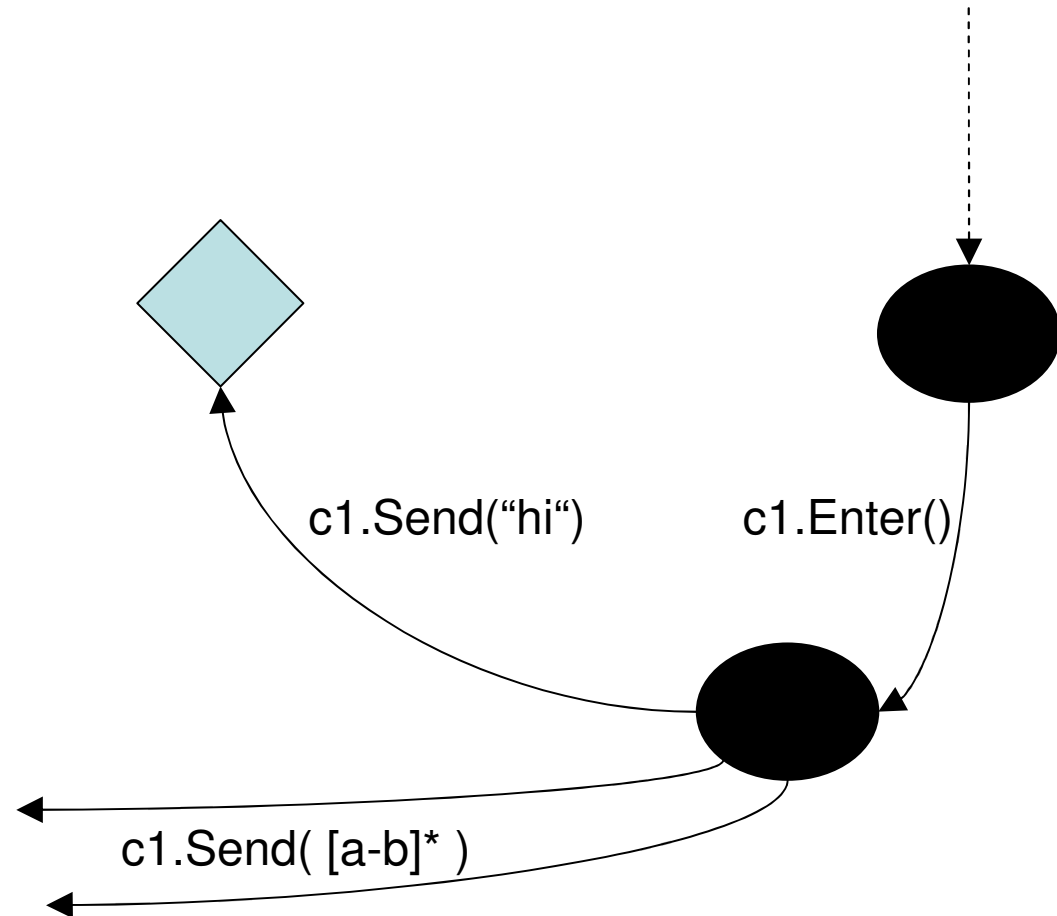
(unendlicher)  
Modellautomat A

Szenario ↓

(endlicher)  
Modellautomat B

### Techniken:

- Parameterauswahl
- Methodenbeschränkung
- Zustandsfilter
- Zustandsgruppierung

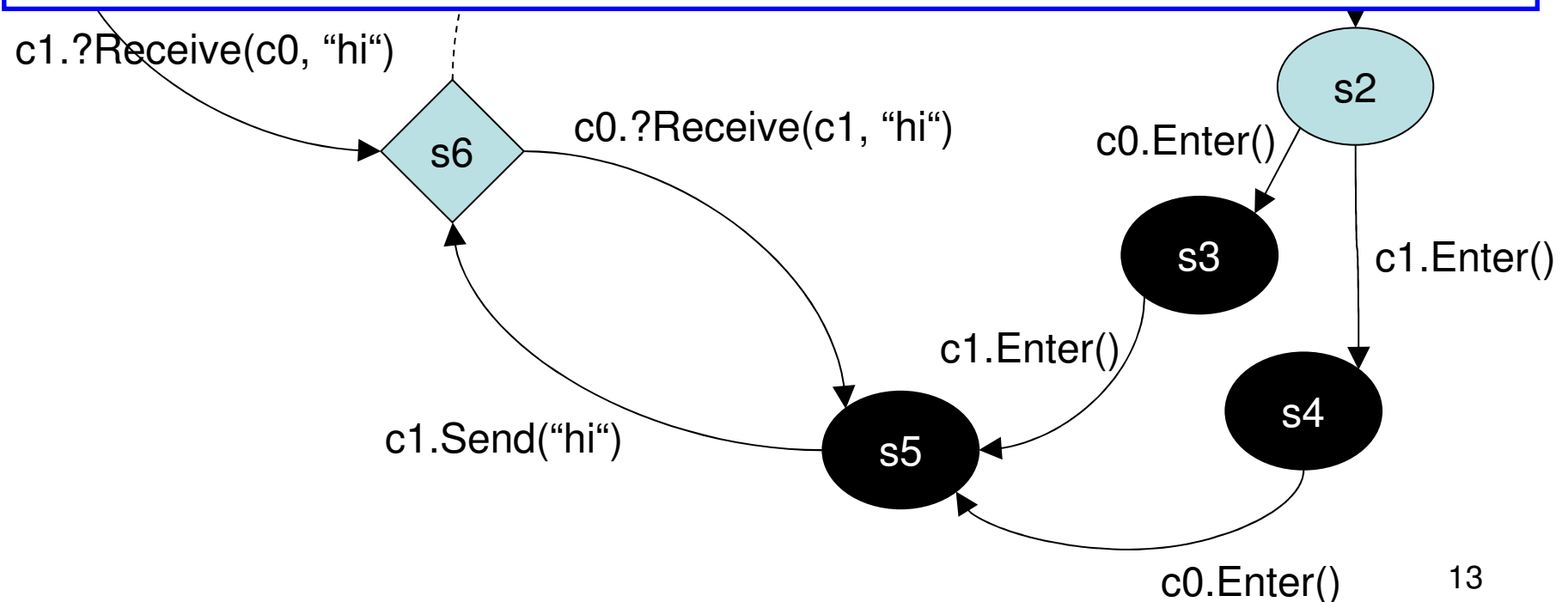


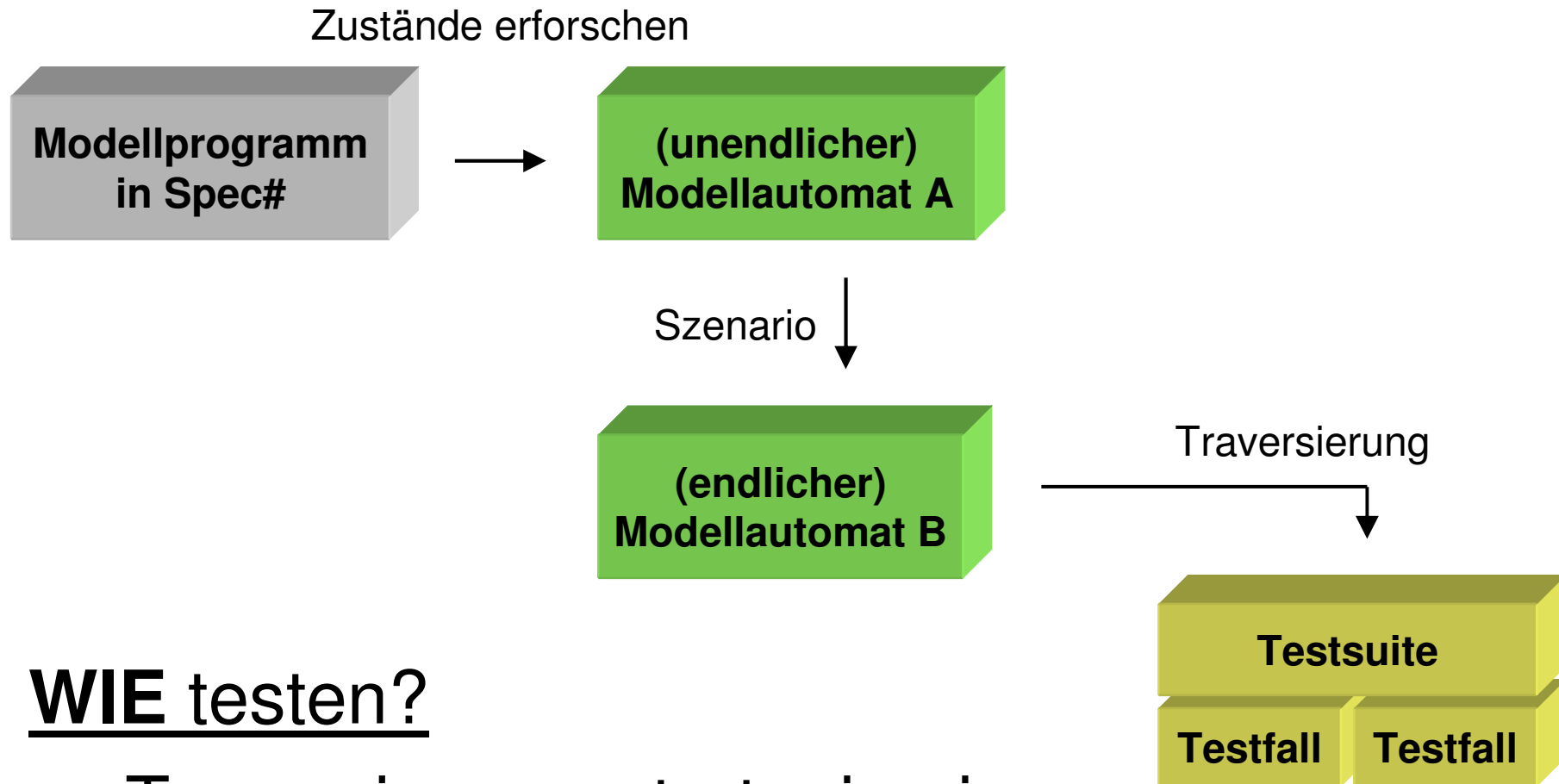
**Bsp.: Endzustände spezifizieren**

```

enumof(Client).size > 0 &&
forall{ c in enumof(Client),
  s in c.unreceivedMsgs.Keys;
  c.unreceivedMsgs[s].Length == 0 }

```





## WIE testen?

- Traversierungsstrategien in Form von Testsuites

## **Definition Testsuite (Auszug) :**

- weiterer Modellautomat  $T$
- erzeugt durch Abgehen von  $M$

## **Bedingungen (Auszug):**

- Endzustand von jedem Zustand aus erreichbar
- keine Schleifen im Graphen (DAG)

## **Testfall:**

- Teilautomat von  $T$
- genau ein Startzustand
- geschlossen unter Transitionsfunktion

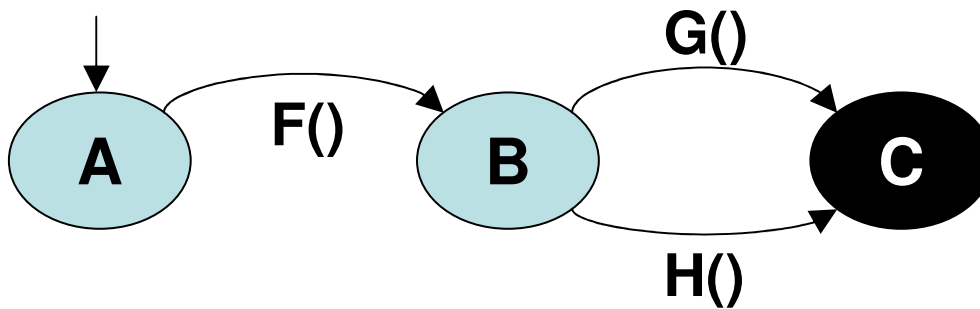
## Traversierungsalgorithmus:

- erzeugt Testsuite mit best. Testziel
- Testziele u.a.:
  - Transitionsabdeckung
  - Zustand erreichen, der best. Bedingung erfüllt
  - Menge zufälliger Wege erzeugen

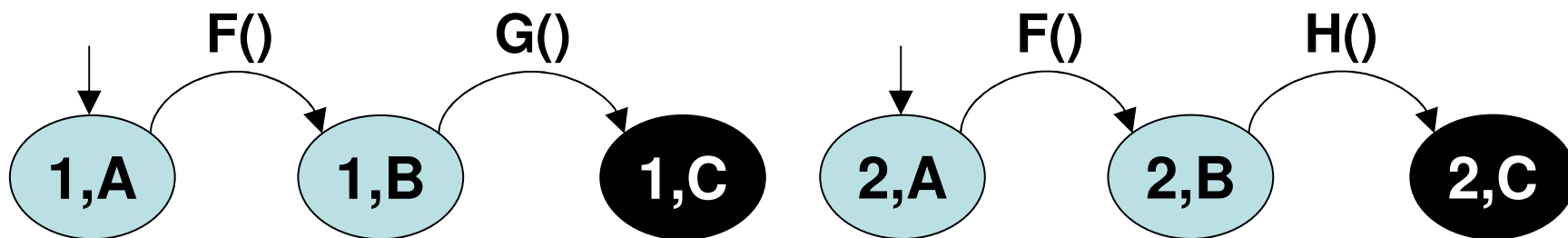
## 4

## Beispiel für Testsuite und Testfälle

**MP:** `enum Mode = {A, B, C}`  
`Mode mode = A;`  
`void F() requires mode == A { mode = B; }`  
`void G() requires mode == B { mode = C; }`  
`void H() requires mode == B { mode = C; }`

**MA:**

**Traversierungs-  
algorithmus:**  
 alle Transitionen  
 von M  
 abdecken



## WAS wollen wir testen?

- eine Konformanzrelation: Verfeinerung

## Konformanz:

- N verhält sich konform zu M, falls eine Verfeinerung von M nach N existiert ( $M \preceq N$ )
- M verfeinert N, falls es eine alternierende Simulation von M nach N gibt

**Definition:**

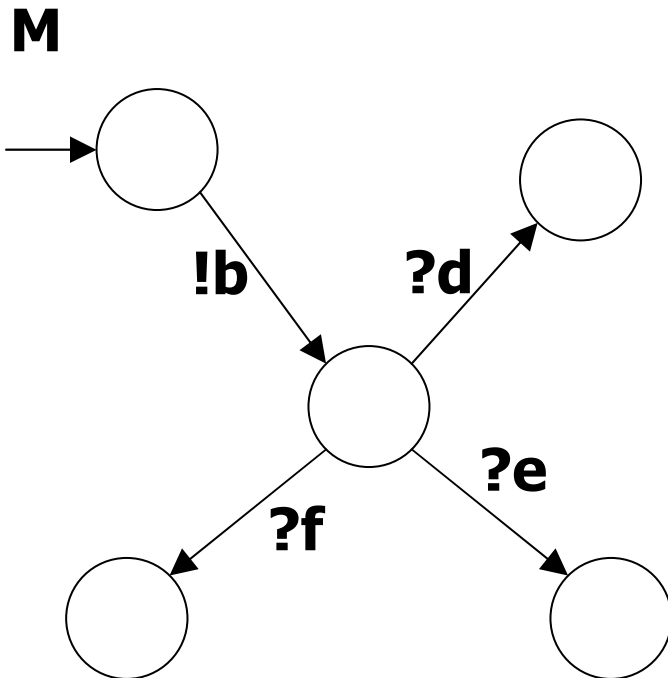
Eine alternierende Simulation von M nach N ist eine Relation  $\rho \subseteq S_M \times S_N$  für die gilt:

Zu Beginn  
enthält  $\rho$  :  
 $S_M^{\text{init}} \times S_N^{\text{init}}$

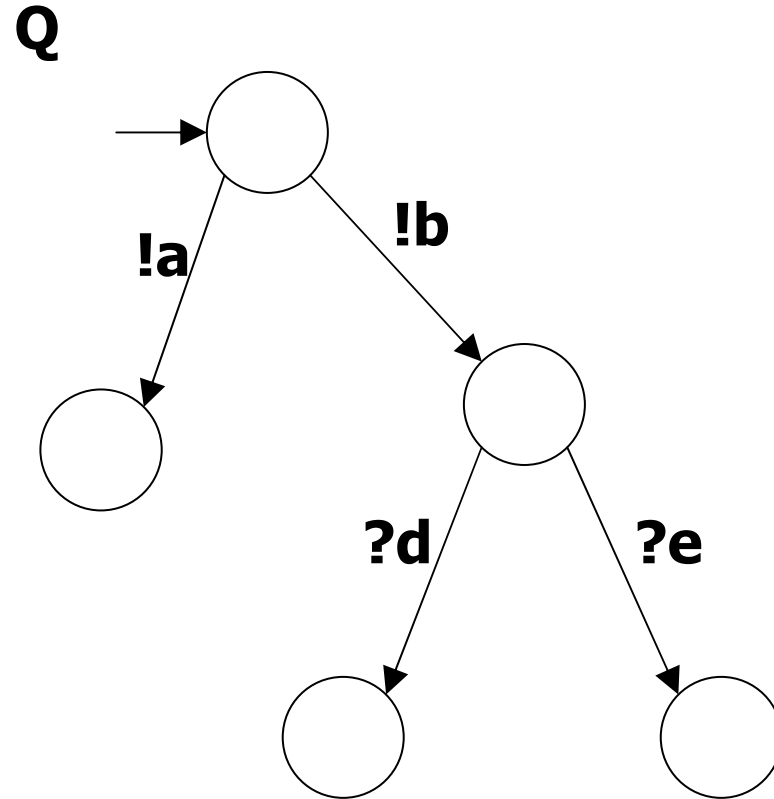
für alle  $a \in \text{Ctrl}_M(s)$  gilt  
es gibt  $t$  mit  $a \in \text{Ctrl}_N(t)$   
und  $(\delta_M(s,a), \delta_N(t,a)) \in \rho$

für alle  $a \in \text{Obs}_N(t)$  gilt  
es gibt  $s$  mit  $a \in \text{Obs}_M(s)$   
und  $(\delta_M(s,a), \delta_N(t,a)) \in \rho$

# Beispiel zur Verfeinerung



$$\mathbf{M} \preceq \mathbf{Q}$$



$$\mathbf{Q} \not\preceq \mathbf{M}$$

