Model Checking mit SPIN

Sabine Bauer

15.08.2005

Gliederung

- 1. Teil: Grundlagen des Model Checking
  - Abgrenzung zur deduktiven Verifikation
  - Das Model Checking-Problem
  - Kripke-Struktur
  - LTL
  - Arbeitsweise eines Model Checkers
- 2. Teil: Der Model Checker „SPIN“
  - Einführung in SPIN
  - Promela
  - Anforderungen in LTL, Promela und als Automat
  - Suchalgorithmus

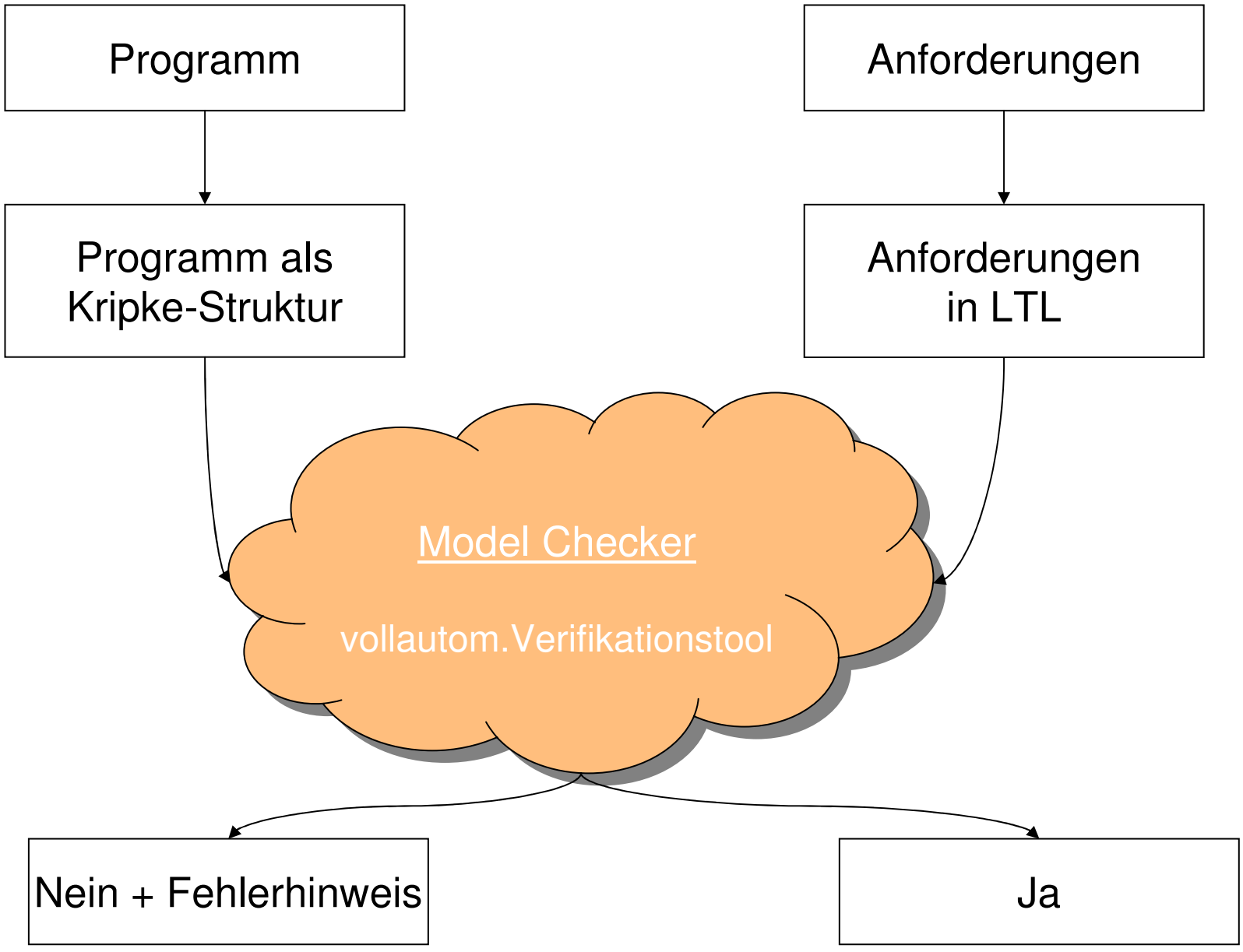
1. Teil:

Grundlagen des Model Checking

Abgrenzung zur deduktiven Verifikation

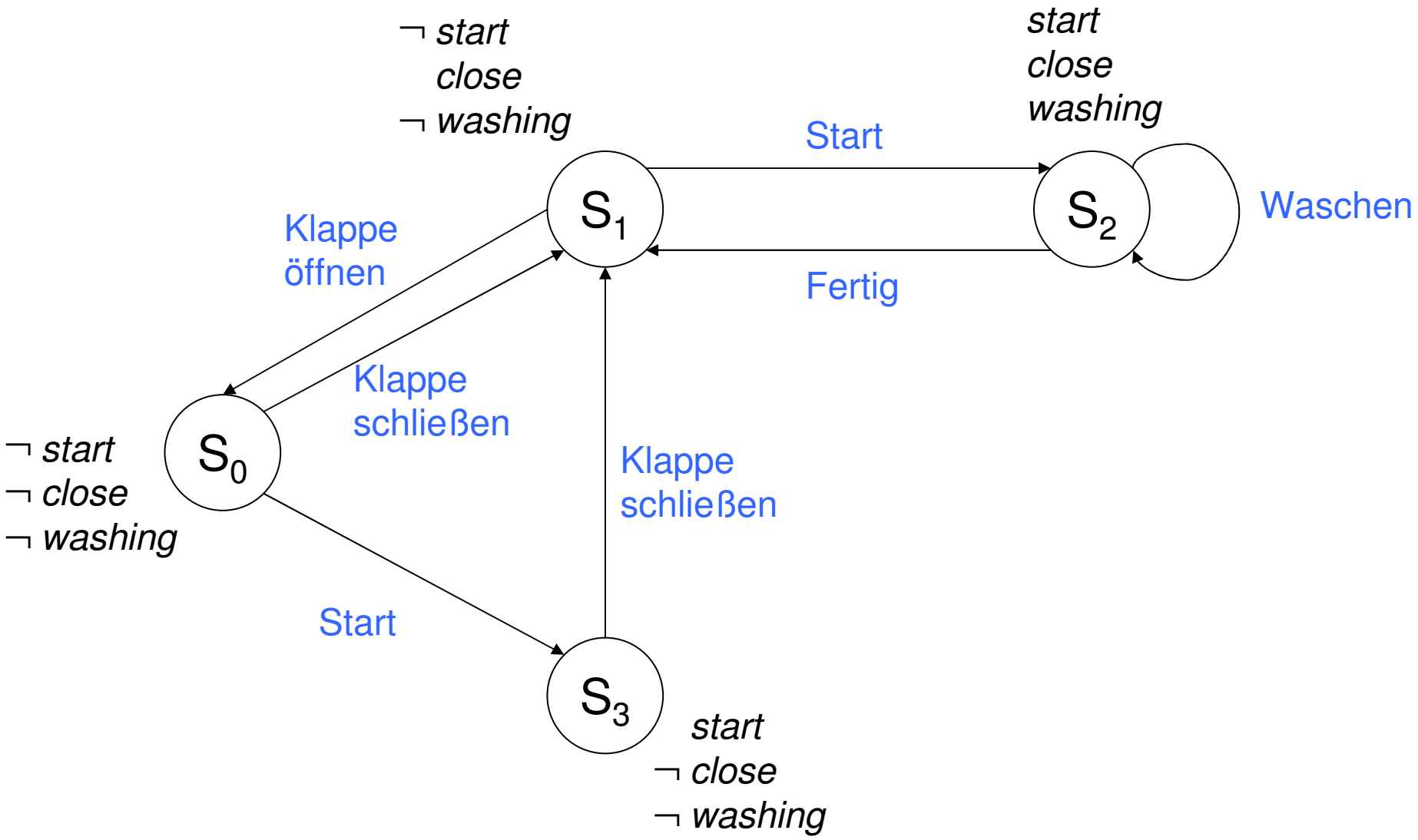
- Deduktive Verifikation
  - Formalisierung eines terminierenden Programms durch Vor- und Nachbedingungen, sowie Zusicherungen
- Model Checking
  - Beschreibung eines für den Endlosbetrieb konzipierten Computersystems durch temporallogische Formeln wie z.B. „Jede Anfrage wird irgendwann beantwortet“ oder „Niemals sind zwei Prozesse gleichzeitig in ihrem kritischen Abschnitt“

Das Model Checking-Problem



Kripke-Struktur

$M = (S, S_0, R, L)$



Anforderungen an das System

Forderungen an alle Pfade:

- Sicherheitsbedingung: Ein unerwünschter Zustand darf nie erreicht werden!
- Lebendigkeitsbedingung: Ein erwünschter Zustand muss irgendwann erreicht werden!

Die Anforderungen werden in LTL formuliert!

Linear Temporal Logic (LTL)

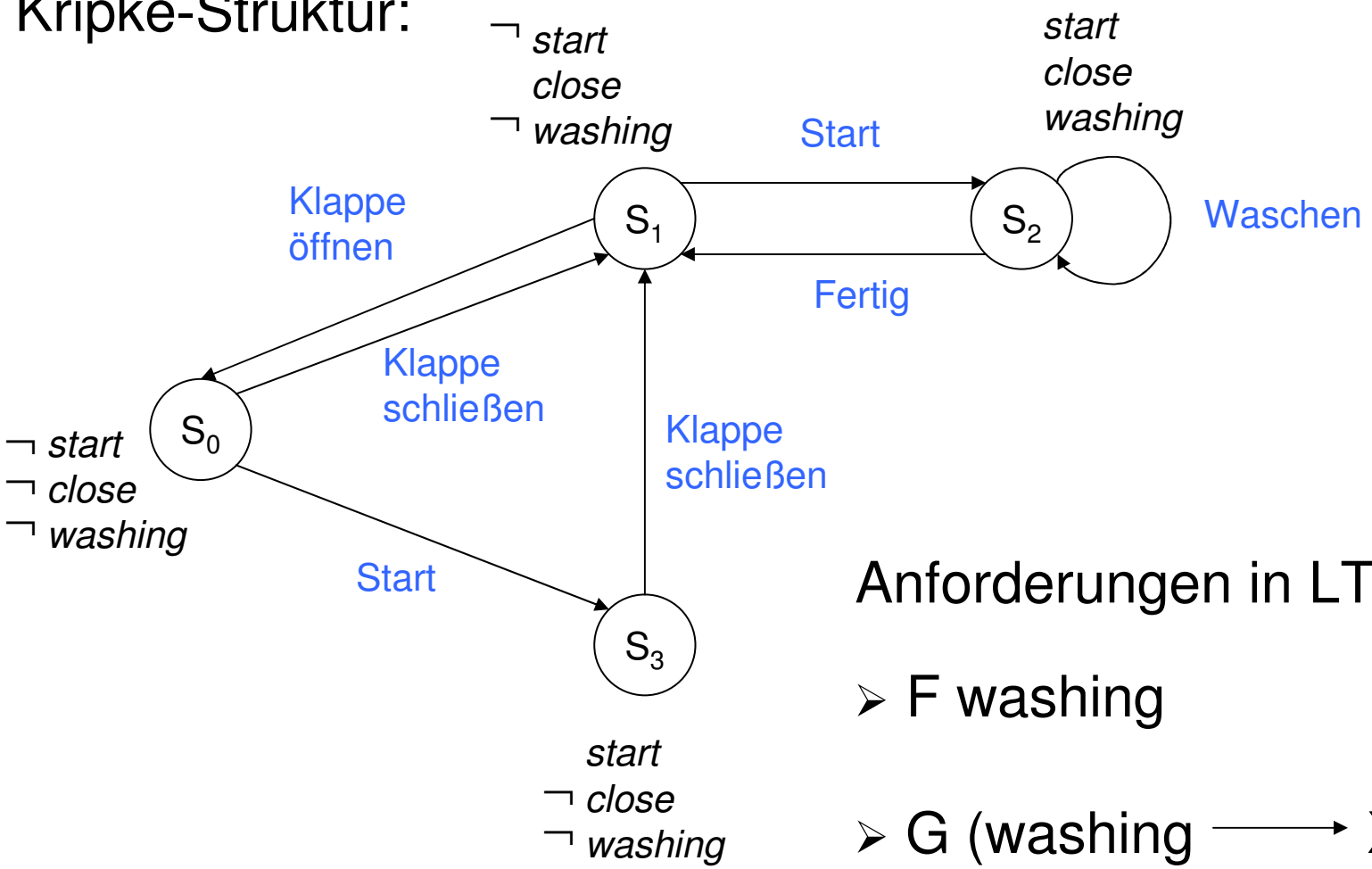
Es gibt die folgenden vier grundlegenden Operatoren:

- **X**p p gilt im nächsten Zustand
- **F**p p gilt irgendwann einmal
- **G**p p gilt immer
- p**U**q Falls es einen Zustand gibt, in dem q gilt, dann muss in jedem vorherigen Zustand p erfüllt sein

LTL baut auf der Aussagenlogik auf!

LTL - Beispiele

Kripke-Struktur:

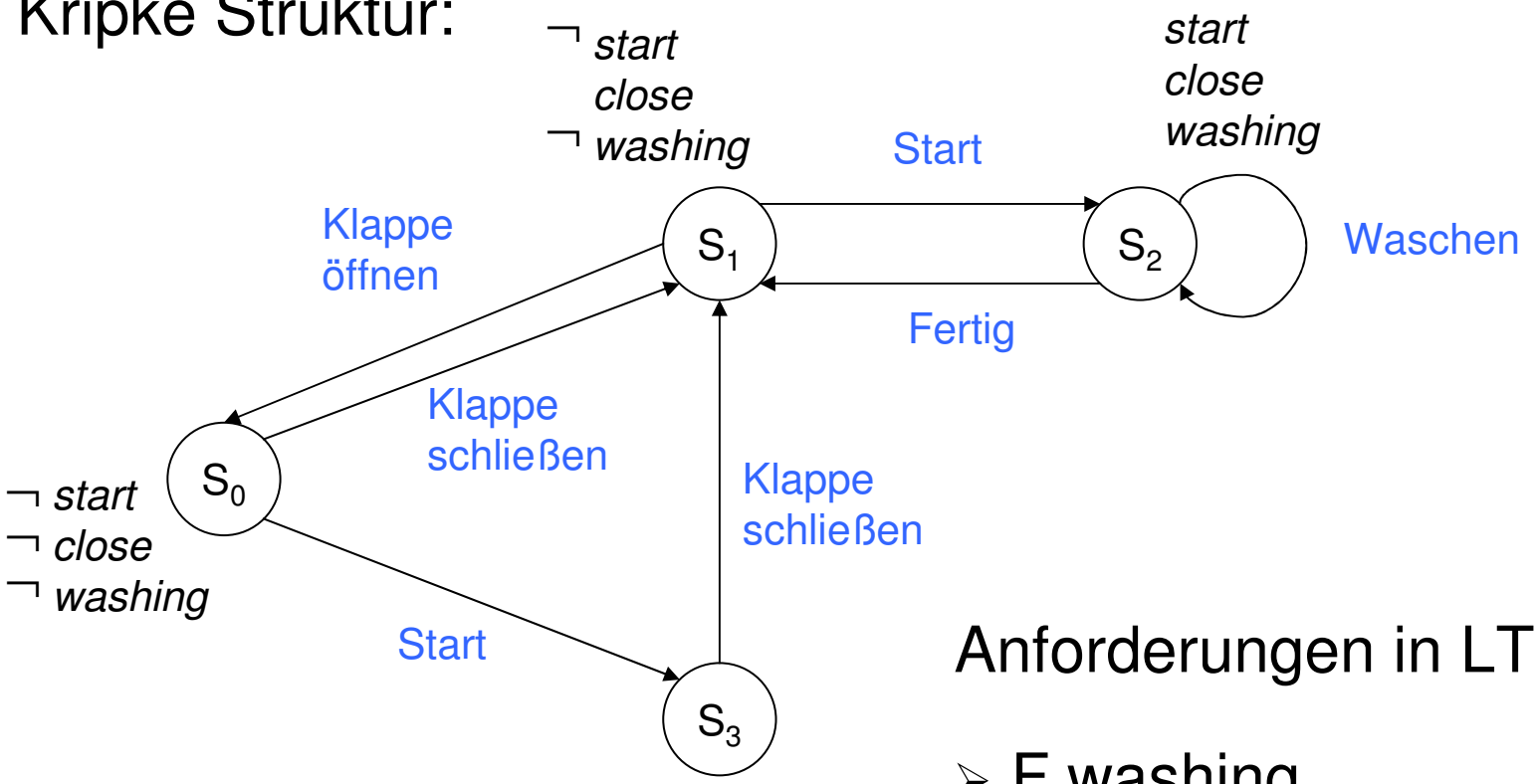


Anforderungen in LTL:

- F washing
- G (washing $\longrightarrow$ X close)
- G ($\neg$(start $\vee$ close $\vee$ washing)) $\longrightarrow$ X (start $\vee$ close)

LTL - Beispiele

Kripke Struktur:



Wie bekommt man die nicht zutreffenden Anforderungen??

Anforderungen in LTL:

- F washing
- G (washing $\longrightarrow$ X close)
- G ($\neg$(start $\vee$ close $\vee$ washing)) $\longrightarrow$ X (start $\vee$ close)

Arbeitsweise eines Model Checkers

Nötig: Automaten auf unendlichen Wörtern (Büchi - Automaten)

(1) Fasse die Kripke-Struktur M als Automaten A_M auf

(2) Wandle die LTL-Formel $\neg\varphi$ in einen Automaten $A_{\neg\varphi}$ um

(3) Bilde den den Durchschnitt der Automaten von A_M und $A_{\neg\varphi}$

(4) Akzeptierte Sprache leer $\longrightarrow$ M erfüllt φ

Akzeptierte Sprache nicht leer $\longrightarrow$ Durchschnitt enthält die Pfade von M, die φ verletzen

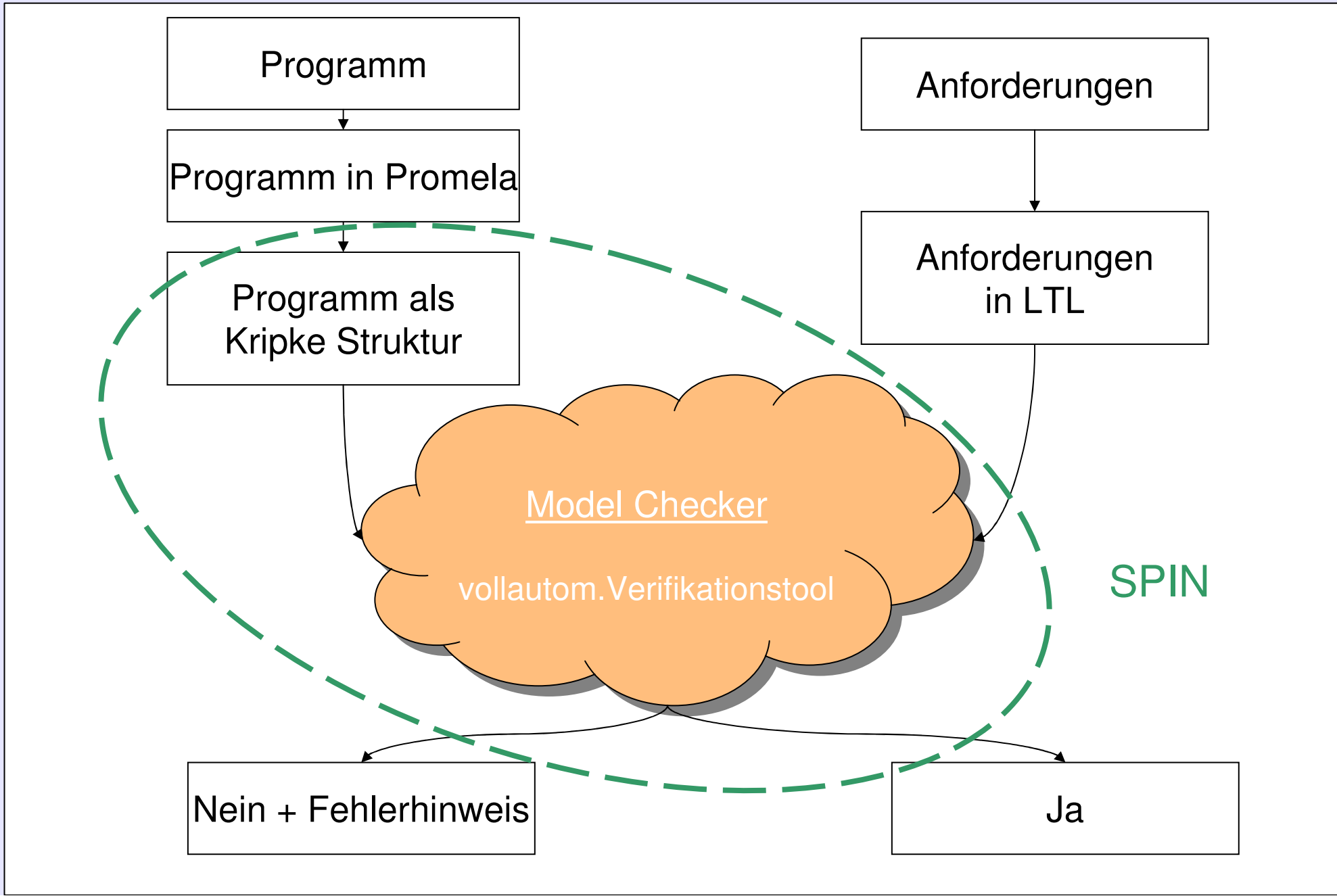
2. Teil:

Der Model Checker „SPIN“

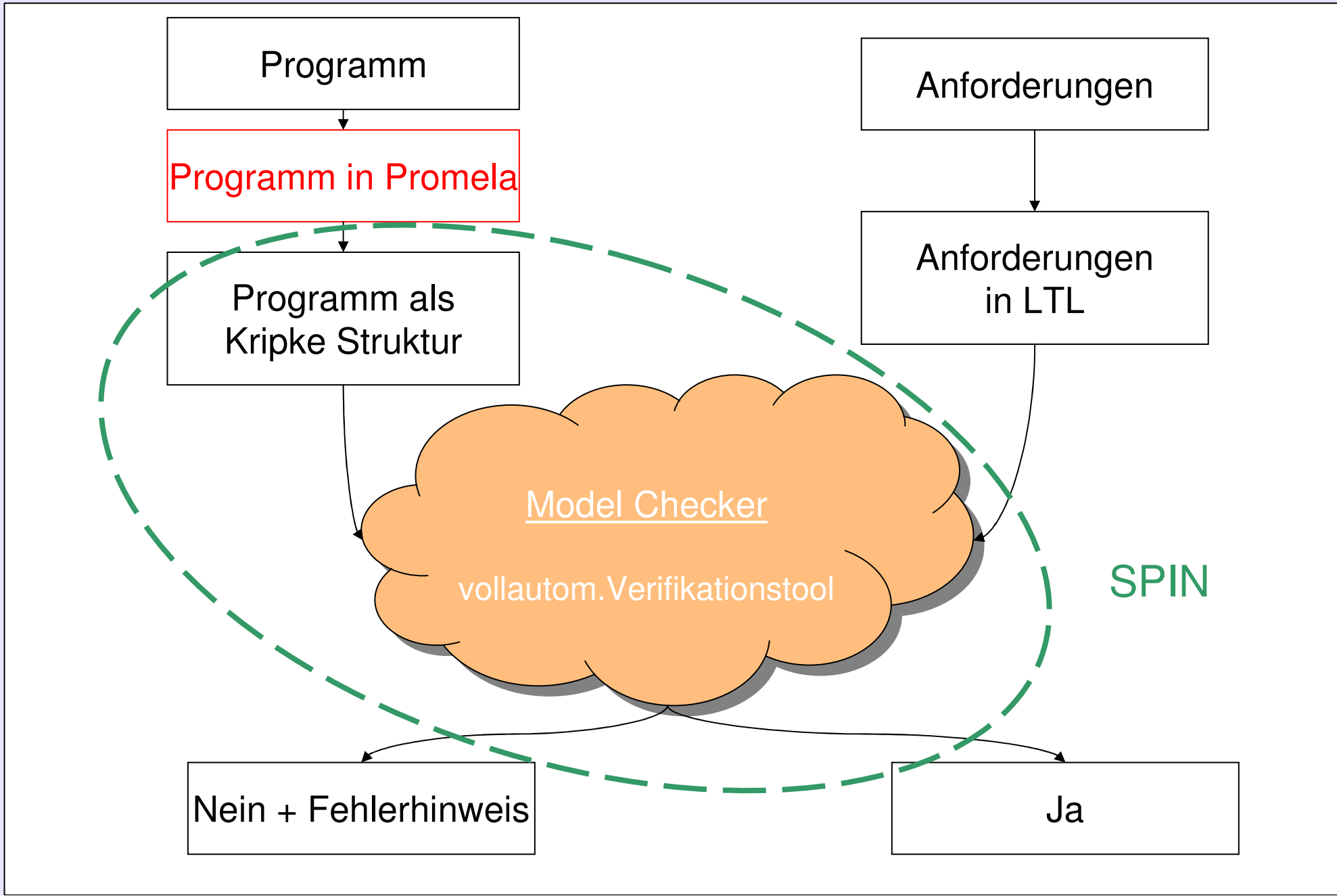
Simple Promela Interpreter (SPIN)

- Validation von Softwaremodellen
- Beschreibung der Softwaremodelle durch Promela (Process/Protocol Meta Language)
  - C ähnliche Syntax
- Angabe der zu erfüllenden Eigenschaften in LTL
- Haupteinsatzgebiet: Analyse von Kommunikations-protokollen
- Entwickelt von Gerard Holzmann, Bell Laboratories

SPIN


Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Model Checker
vollautom.Verifikationstool
SPIN
Nein + Fehlerhinweis
Ja

SPIN


Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Model Checker
vollautom.Verifikationstool
SPIN
Nein + Fehlerhinweis
Ja

Promela - Beispiel

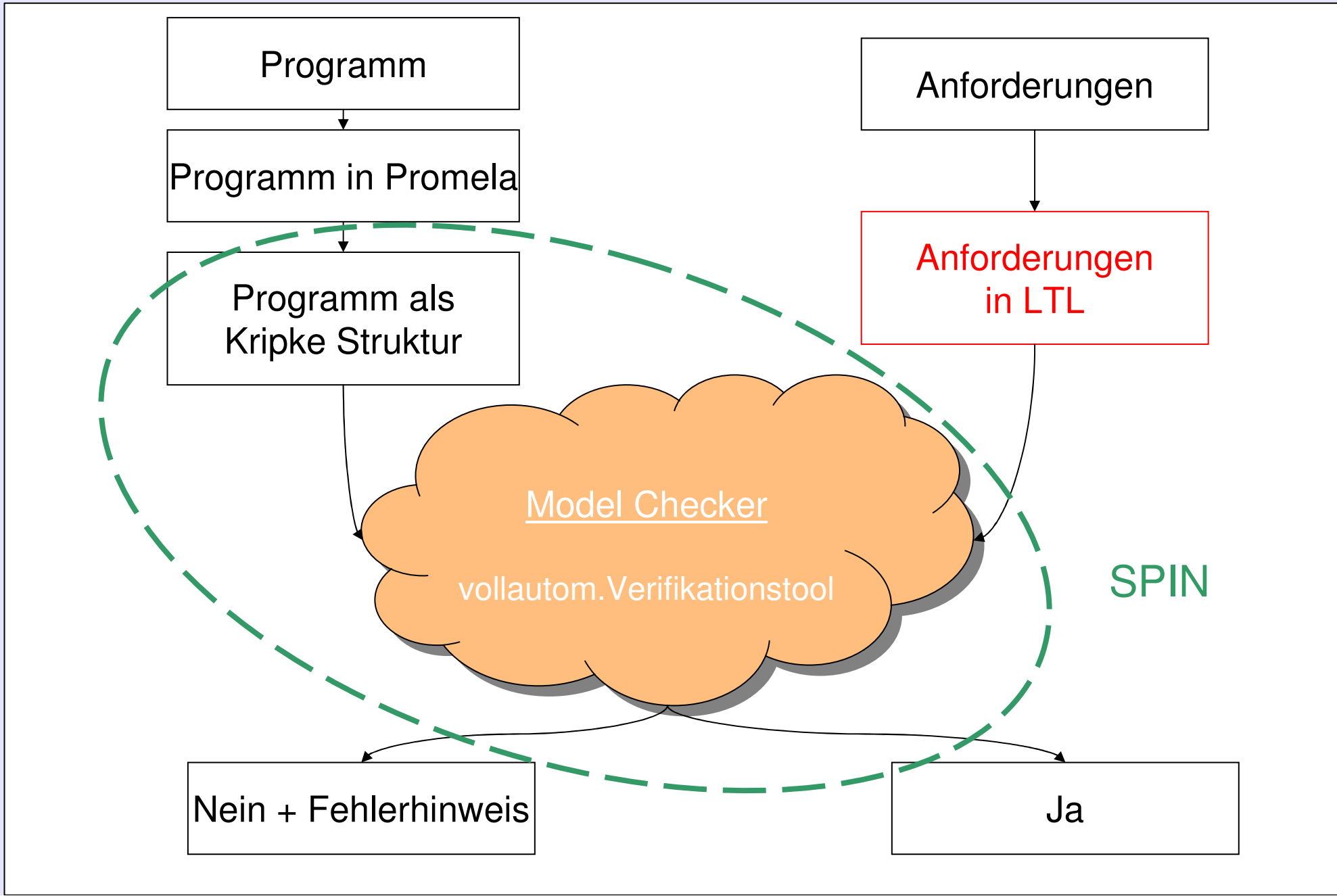
```
mtype = { P, C };

mtype turn = P;

active proctype producer() {
    do
    :: (turn == P) ->
        printf(“Produce\n“);
        turn = C
    od
}

active proctype consumer() {
    do
    :: (turn == C) ->
        printf(“Consume\n“);
        turn = P
    od
}
```

SPIN

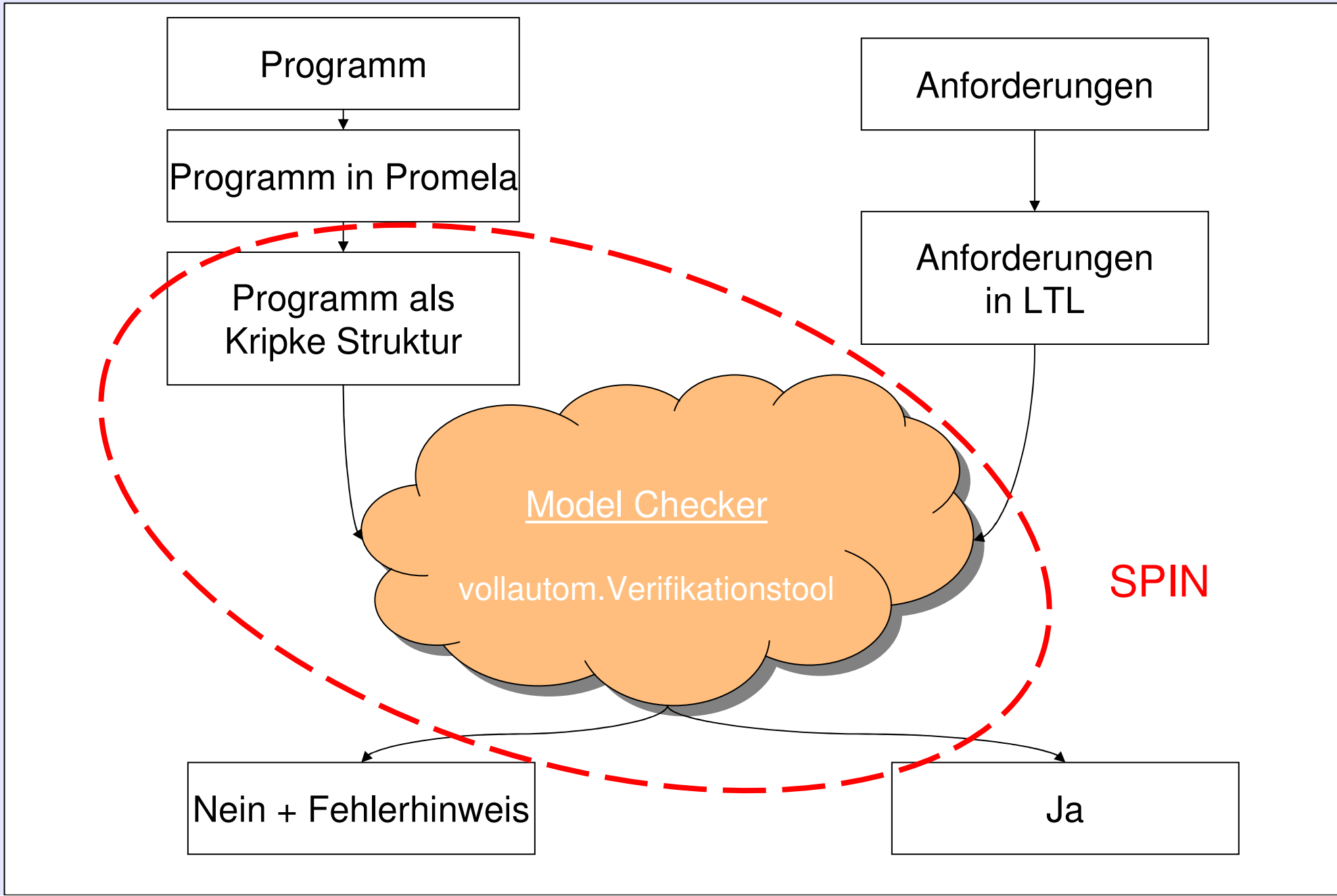

Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Model Checker
vollautom.Verifikationstool
SPIN
Nein + Fehlerhinweis
Ja

Anforderungen nach LTL

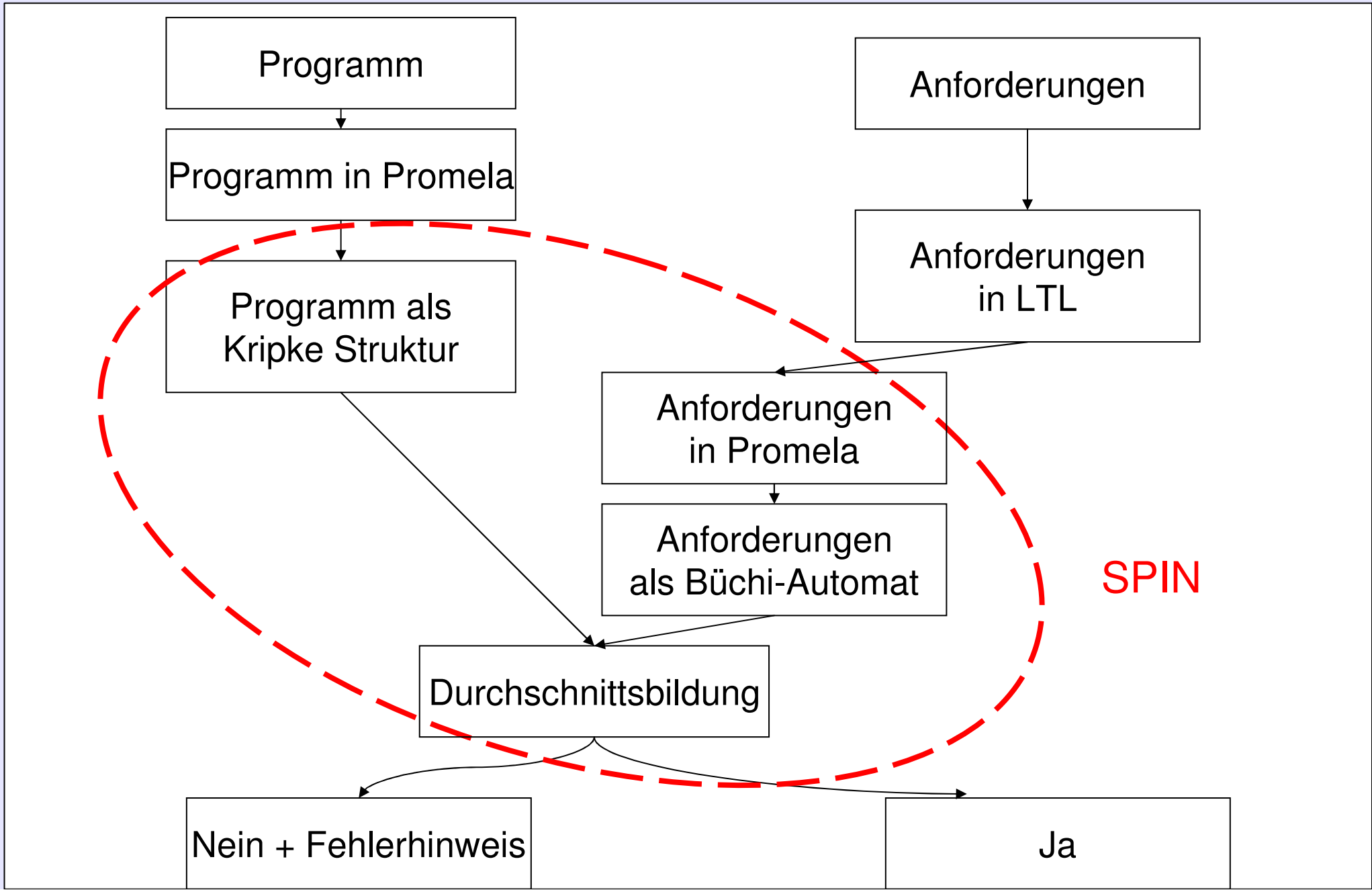
Häufig benutzte LTL-Formeln:

- [] p // immer p
- <> p // irgendwann einmal p
- p ⟶ <> q // wenn p, dann irgendwann q
- p ⟶ q U r // wenn p, dann q bis r auftritt
- [] <> p // immer irgendwann p
- <> [] p // irgendwann immer p

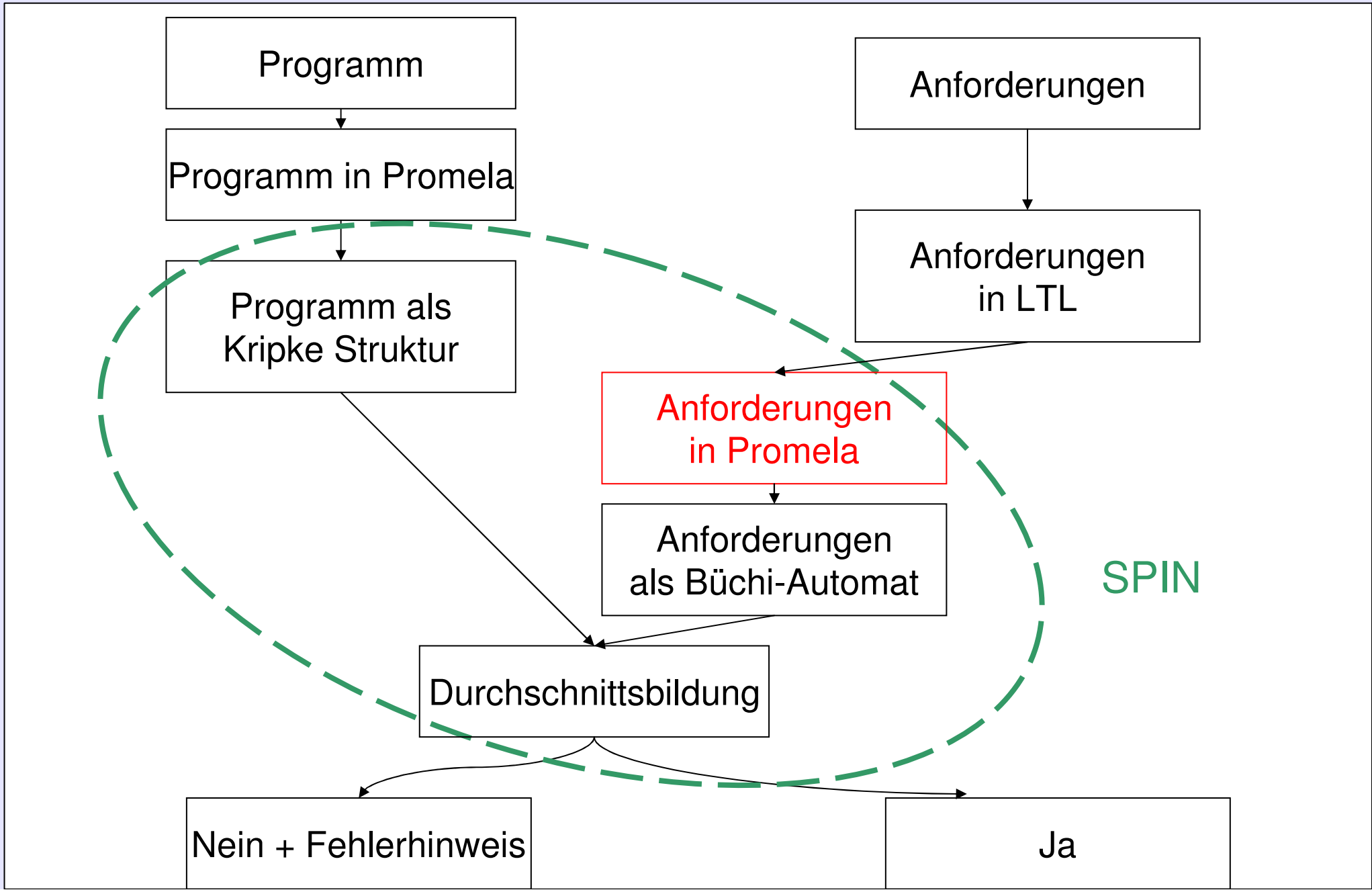
SPIN


Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Model Checker
vollautom.Verifikationstool
SPIN
Nein + Fehlerhinweis
Ja

SPIN


Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Anforderungen in Promela
Anforderungen als Büchi-Automat
SPIN
Durchschnittsbildung
Nein + Fehlerhinweis
Ja

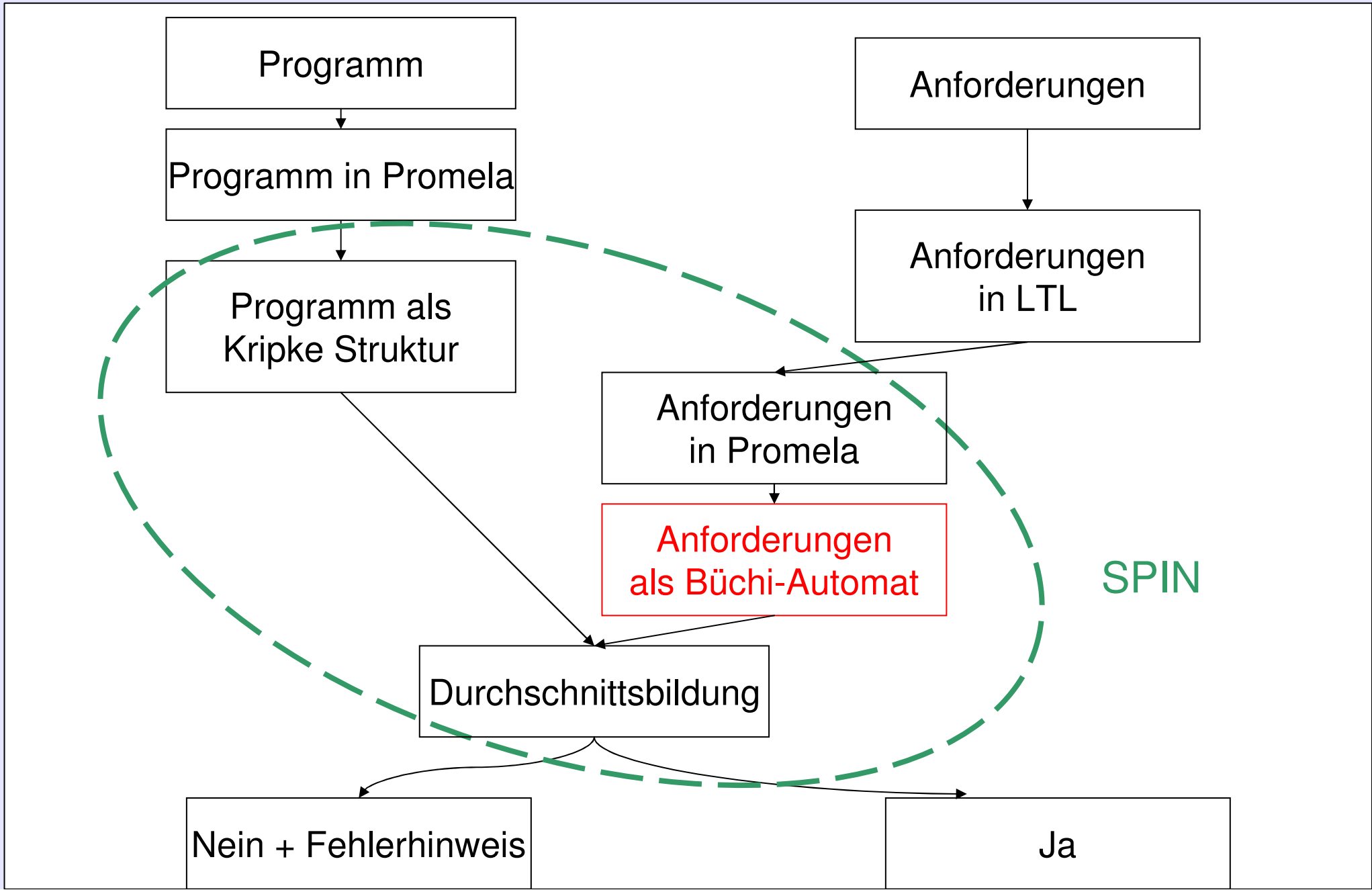
SPIN


Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Anforderungen in Promela
Anforderungen als Büchi-Automat
SPIN
Durchschnittsbildung
Nein + Fehlerhinweis
Ja

LTL nach Promela

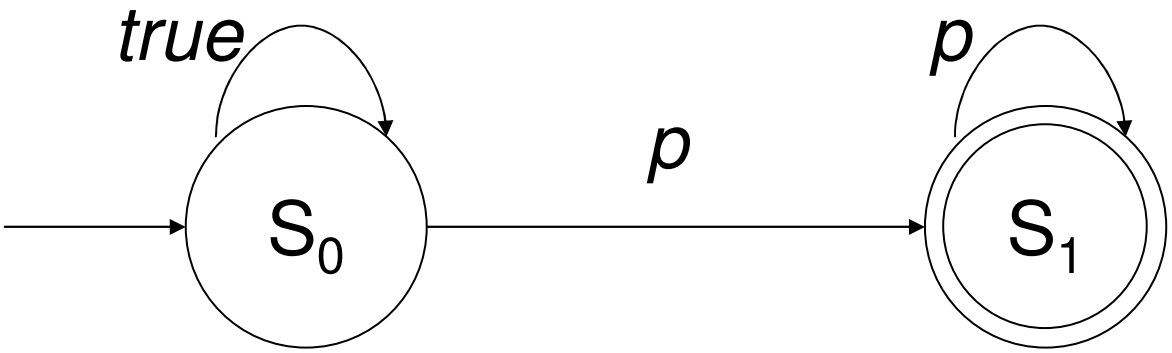
```
never {          /* <> [] p  */
T0_init:
        if
        :: (p) -> goto accept_S4
        :: (1) -> goto T0_init
        fi;
accept_S4:
        if
        :: (p) -> goto accept_S4
        fi
}
```

SPIN

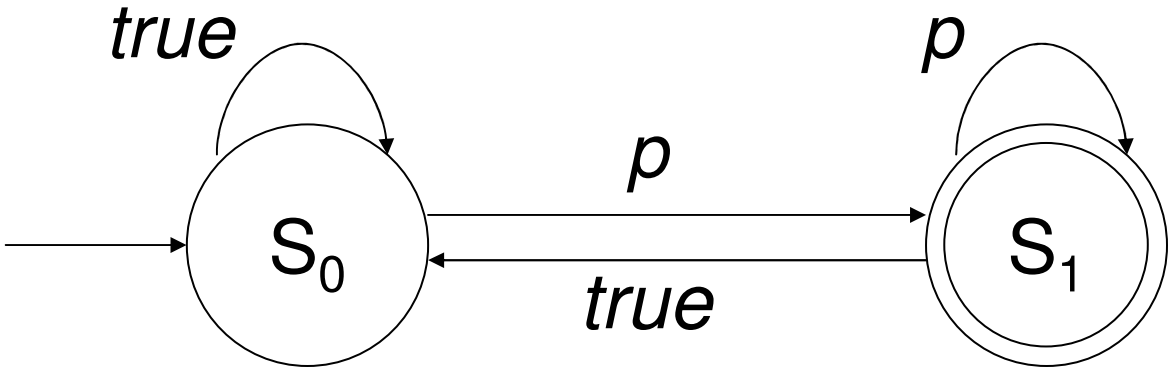

Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Anforderungen in Promela
Anforderungen als Büchi-Automat
SPIN
Durchschnittsbildung
Nein + Fehlerhinweis
Ja

LTL/Promela nach Automat

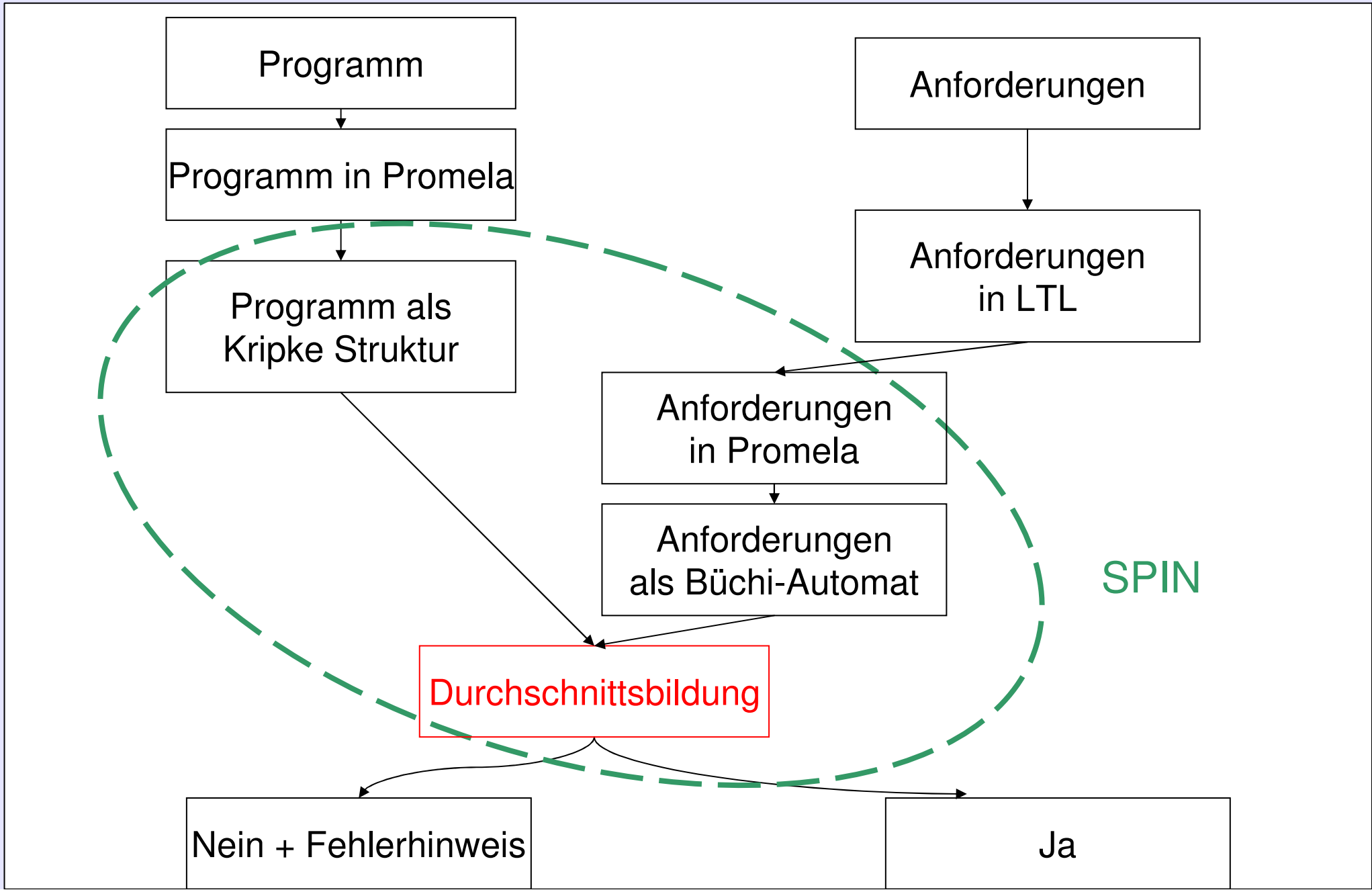
Büchi-Automat für <> [] p:



Büchi-Automat für [] <> p:



SPIN


Programm
Programm in Promela
Programm als Kripke Struktur
Anforderungen
Anforderungen in LTL
Anforderungen in Promela
Anforderungen als Büchi-Automat
SPIN
Durchschnittsbildung
Nein + Fehlerhinweis
Ja

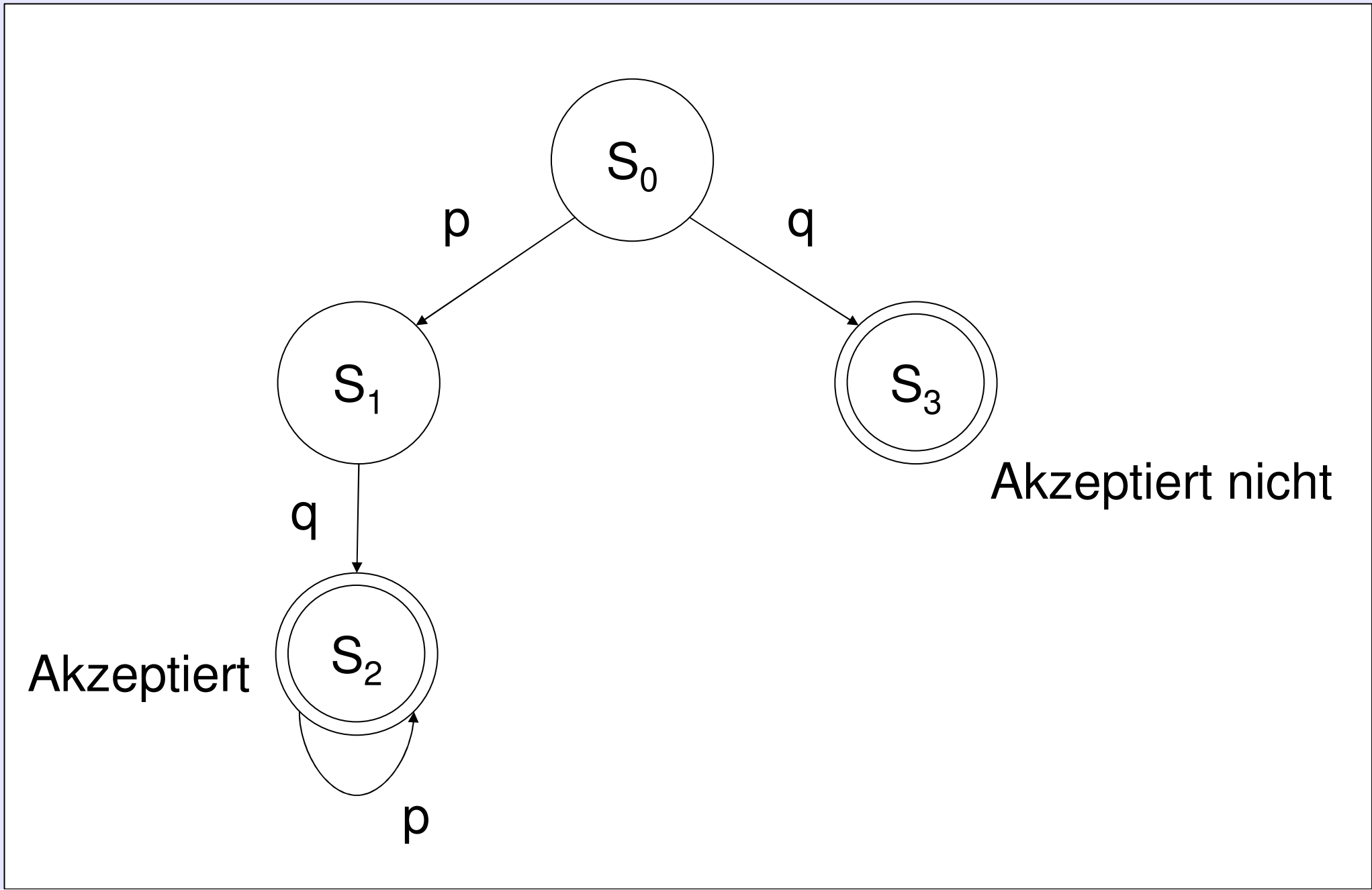
Suchalgorithmus

- Büchi-Automat akzeptiert, wenn ein Endzustand unendlich oft durchlaufen wird

- Daher Suche ob
  1. Endzustand vom Startzustand aus erreicht werden kann und
  2. Endzustand von sich selbst aus erreichbar ist.

- 1. + 2. erfüllt $\longrightarrow$ Automat enthält die Pfade, die φ verletzen
- Andernfalls $\longrightarrow$ M erfüllt φ

Geschachtelte Tiefensuche



Zusammenfassung

Vorteile:

- Automatisierung des Verifikationsprozesses
- Im Fehlerfall werden direkt die verletzten Eigenschaften ermittelt

Nachteile:

- System darf nur eine endliche Menge an Zuständen haben
- Mögliches Auftreten einer Zustandsexplosion

Allgemein:

- SPIN ist auf dem Gebiet des Model Checking Marktführer
- 2001 bekam SPIN den „Software System Award“ verliehen

Quellen

- Holzmann, Gerard J.: The Spin Model Checker, Boston, 2003
- Tretow, Annekatrin: Seminar Logik in der Informatik, Model Checking – Grundlagen, Uni Koblenz, 2004
- Ciesinski, Frank: SPIN / PROMELA, Modelchecker und Spezifikationssprache, Uni Bonn, 2004
- http://www-i7.informatik.rwth-aachen.de/d/teaching/ws0304/modelchk/DivX-PDF.html
- http://www.lib.uni-bonn.de/informatik.html
- http://www.software-kompetenz.de