Theoretische Informatik II

Einheit 6

Berechenbarkeitsmodelle

1. Turingmaschinen
2. Registermaschinen
3. μ-rekursive Funktionen
4. Typ-0 Grammatiken
5. Weitere Berechenbarkeitsmodelle
6. Church'sche These

Berechenbarkeitsmodelle – wozu?

- **Es gibt mehr als nur die Standard PC Architektur**
  - Lisp Maschinen, Parallelrechner, Neuronale Netze
  - Nichtdeterministische Maschinen (Quantencomputer)

Berechenbarkeitsmodelle – wozu?

- **Es gibt mehr als nur die Standard PC Architektur**
  - Lisp Maschinen, Parallelrechner, Neuronale Netze
  - Nichtdeterministische Maschinen (Quantencomputer)
- **Abstrakte Modelle betrachten die wirklichen Fragen zuerst**
  - Was genau ist das Problem?
  - Was charakterisiert eine Lösung des Problems?
  - Wie kann man prinzipiell an das Problem herangehen?
  - Wie kann man über den Stand der Technik hinausgehen?

Berechenbarkeitsmodelle – wozu?

- **Es gibt mehr als nur die Standard PC Architektur**
  - Lisp Maschinen, Parallelrechner, Neuronale Netze
  - Nichtdeterministische Maschinen (Quantencomputer)
- **Abstrakte Modelle betrachten die wirklichen Fragen zuerst**
  - Was genau ist das Problem?
  - Was charakterisiert eine Lösung des Problems?
  - Wie kann man prinzipiell an das Problem herangehen?
  - Wie kann man über den Stand der Technik hinausgehen?
- **Berechenbarkeitsmodelle klären fundamentale Fragen**
  - Was ist überhaupt Berechenbarkeit?
  - Auf welche Arten kann man Berechnungen durchführen?
  - Sind bestimmte Berechnungsmodelle besser als andere?

Berechenbarkeitsmodelle – wozu?

- **Es gibt mehr als nur die Standard PC Architektur**
  - Lisp Maschinen, Parallelrechner, Neuronale Netze
  - Nichtdeterministische Maschinen (Quantencomputer)
- **Abstrakte Modelle betrachten die wirklichen Fragen zuerst**
  - Was genau ist das Problem?
  - Was charakterisiert eine Lösung des Problems?
  - Wie kann man prinzipiell an das Problem herangehen?
  - Wie kann man über den Stand der Technik hinausgehen?
- **Berechenbarkeitsmodelle klären fundamentale Fragen**
  - Was ist überhaupt Berechenbarkeit?
  - Auf welche Arten kann man Berechnungen durchführen?
  - Sind bestimmte Berechnungsmodelle besser als andere?

Berechenbarkeitsmodelle gab es lange vor dem ersten Computer

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159526535897898451991650290437974035739898 68\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.1415926535897898451991650290437974035739898 68\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.1415952653589789845199165029043797403573989868\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159526535897898451991650290437974035739898 68\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.1415952653589789845199165029043797403573989868\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159265358978984519916502904379740357398986 8\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.1415952653589789845199165029043797403573989868\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159265358978984519916502904379740357398 9868\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159265358979845199165029043797403573989868\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159526535897898451991650290437974035739898 68\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14\mathbf{15}9526535897898451991650290437974035739898 68\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159526535897898451991650290437974035739898 68\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.1415{\color{red}9}5265358978984519916502904379740357398986 8\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$h(x) = \begin{cases} 1 & \text{wenn in der Dezimalentwicklung von } \pi \text{ mindestens} \\ & x \text{ aufeinanderfolgende Neunen vorkommen,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.14159265358979898451\mathbf{99}16502904379740357398986 8\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$h(x) = \begin{cases} 1 & \text{wenn in der Dezimalentwicklung von } \pi \text{ mindestens} \\ & x \text{ aufeinanderfolgende Neunen vorkommen,} \\ 0 & \text{sonst} \end{cases}$$

Was ist überhaupt Berechenbarkeit?

$$\pi = 3.141595265358978984519916502904379740357398986\ldots$$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$h(x) = \begin{cases} 1 & \text{wenn in der Dezimalentwicklung von } \pi \text{ mindestens} \\ & x \text{ aufeinanderfolgende Neunen vorkommen,} \\ 0 & \text{sonst} \end{cases}$$

Sind f, g und h berechenbar? Warum?

Was ist überhaupt Berechenbarkeit?

$\pi = 3.141595265358978984519916502904379740357398986 8\ldots$

$$f(x) = \begin{cases} 1 & \text{wenn ein Anfangssegment der Dezimalentwicklung von } \pi \\ & \text{(unter Ignorierung des Punktes) identisch mit } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{wenn ein beliebiges Teilsegment der Dezimalentwicklung} \\ & \text{von } \pi \text{ identisch mit der Zahl } x \text{ ist,} \\ 0 & \text{sonst} \end{cases}$$

$$h(x) = \begin{cases} 1 & \text{wenn in der Dezimalentwicklung von } \pi \text{ mindestens} \\ & x \text{ aufeinanderfolgende Neunen vorkommen,} \\ 0 & \text{sonst} \end{cases}$$

Sind f, g und h berechenbar? Warum?

Der Begriff “berechenbar” muß mathematisch präzisiert werden

Die wichtigsten Berechenbarkeitsmodelle

- **Turingmaschine** (Rechnen mit Papier und Bleistift)
- **Abakus** (Das älteste mechanische Hilfsmittel)
- **Registermaschine** (Assembler / Maschinenprogrammierung)
- **PASCAL-reduziert** (Imperative höhere Sprachen)
- **Nichtdeterministische Turingmaschine** (Parallelismus/Quantenrechner)
- **μ-rekursive Funktionen** (Mathematisches Rechnen)
- **λ-Kalkül** (Funktionale Sprachen, LISP)
- **Logische Repräsentierbarkeit** (Logikprogrammierung, PROLOG)
- **Typ-0 Grammatiken / Markov-Algorithmen** (Regelbasierte Sprachen)

Die wichtigsten Berechenbarkeitsmodelle

- **Turingmaschine** (Rechnen mit Papier und Bleistift)
- **Abakus** (Das älteste mechanische Hilfsmittel)
- **Registermaschine** (Assembler / Maschinenprogrammierung)
- **PASCAL-reduziert** (Imperative höhere Sprachen)
- **Nichtdeterministische Turingmaschine** (Parallelismus/Quantenrechner)
- **μ-rekursive Funktionen** (Mathematisches Rechnen)
- **λ-Kalkül** (Funktionale Sprachen, LISP)
- **Logische Repräsentierbarkeit** (Logikprogrammierung, PROLOG)
- **Typ-0 Grammatiken / Markov-Algorithmen** (Regelbasierte Sprachen)

Alle Modelle führen zu demselben Berechenbarkeitsbegriff

Die wichtigsten Berechenbarkeitsmodelle

- **Turingmaschine** (Rechnen mit Papier und Bleistift)
- **Abakus** (Das älteste mechanische Hilfsmittel)
- **Registermaschine** (Assembler / Maschinenprogrammierung)
- **PASCAL-reduziert** (Imperative höhere Sprachen)
- **Nichtdeterministische Turingmaschine** (Parallelismus/Quantenrechner)
- **μ-rekursive Funktionen** (Mathematisches Rechnen)
- **λ-Kalkül** (Funktionale Sprachen, LISP)
- **Logische Repräsentierbarkeit** (Logikprogrammierung, PROLOG)
- **Typ-0 Grammatiken / Markov-Algorithmen** (Regelbasierte Sprachen)

Alle Modelle führen zu demselben Berechenbarkeitsbegriff

$$\Downarrow$$

Church'sche These:

Intuitive Berechenbarkeit wird durch diese Modelle exakt beschrieben

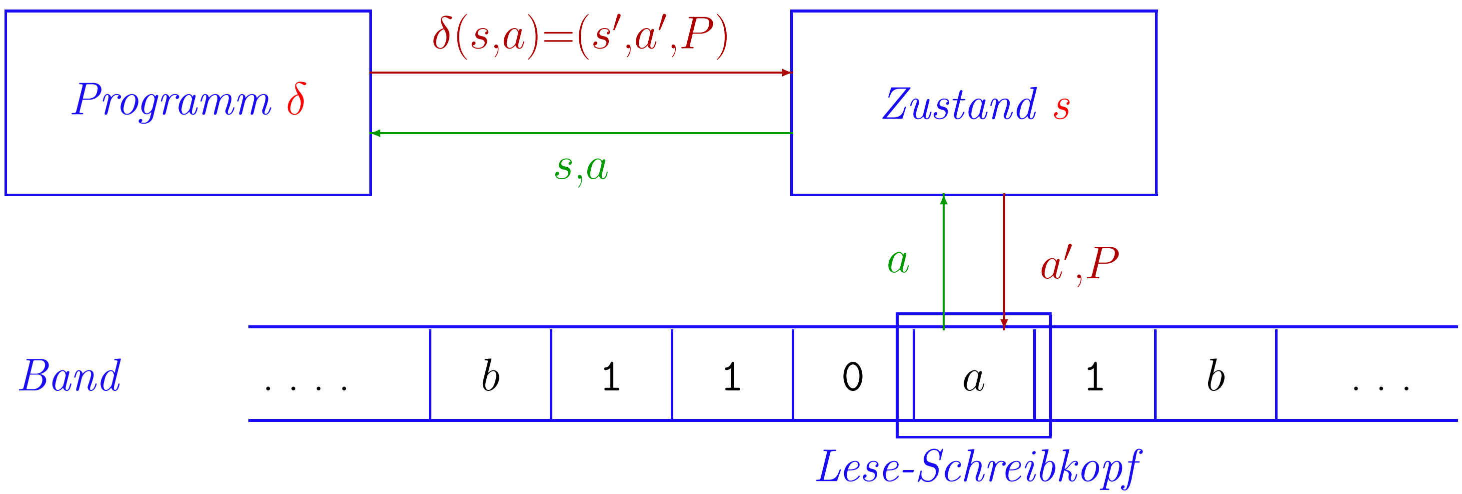
Theoretische Informatik II

Einheit 6.1

Turingmaschinen

1. Arbeitsweise
2. Formale Semantik
3. Turing-Berechenbarkeit
4. Varianten von Turingmaschinen

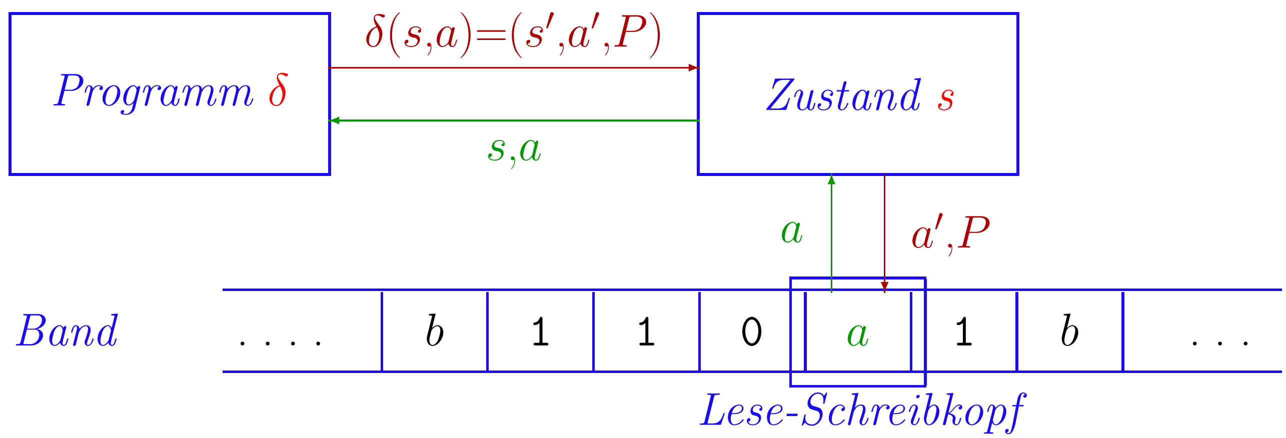
Turingmaschinen



- **Verallgemeinerung von Push-Down Automaten**
  - Stack mit LIFO Zugriff ersetzt durch potentiell unendliches Band
  - Band fast überall unbeschrieben (Leersymbol b ≡ “Blank”)
  - Lese-Schreibkopf kann Symbole lesen, schreiben und bewegt werden

Turingmaschinen – mathematisch

Definition A



Eine **Turingmaschine** ist ein 6-Tupel $\boldsymbol{\tau} = (S, X, \Gamma, \delta, s_0, b)$

- S nichtleere endliche Zustandsmenge
- $s_0 \in S$ Anfangszustand
- Γ nichtleeres endliches Bandalphabet
- $X \subseteq \Gamma$ Eingabealphabet
- $b \in \Gamma \backslash X$ Blanksymbol
- $\delta : S \times \Gamma \to S \times \Gamma \times \{r, l, h\}$ (partielle) Zustandsüberführungsfunktion

Übergangstabelle für δ

Zustand	gelesen	zu schreiben	Folgezustand	Kopfbewegung
s_0	0	s_0	0	r
s_0	1	s_0	1	r
s_0	b	s_1	b	l
s_1	1	s_1	0	l
s_1	0	s_2	1	l
s_1	b	s_3	1	h
s_2	0	s_2	0	l
s_2	1	s_2	1	l
s_2	b	s_3	b	h

Beschreibung von Turingmaschinen

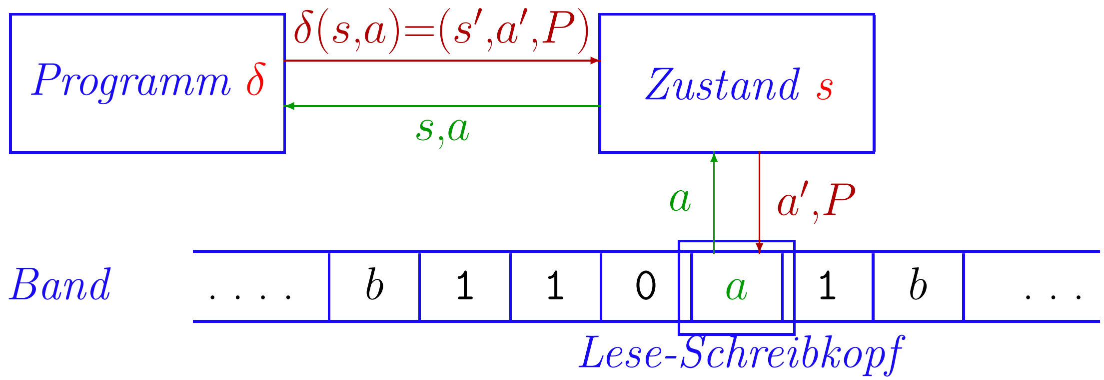
Übergangstabelle für δ

Zustand	gelesen	zu schreiben	Folgezustand	Kopfbewegung
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h

Restliche Komponenten implizit bestimmt

Zustandsmenge	S	$= \{s_0, s_1, s_2, s_3\}$
Anfangszustand	s_0	$= \mathrm{s}_0$
Bandalphabet	Γ	$= \{\mathtt{0}, \mathtt{1}, \mathtt{b}\}$
Eingabealphabet	X	$= \{\mathtt{0}, \mathtt{1}\}$
Blanksymbol	b	$= \mathtt{b}$

Arbeitsweise von Turingmaschinen



- **Anfangssituation**
  - Eingabewort w steht auf dem Band, umgeben von Leerzeichen
  - Kopf über erstem Symbol, Zustand ist s_0
- **Arbeitschritt**
  - Zeichen a lesen, Zustand s und $\delta(s,a)=(s',a',P)$ bestimmen
  - Neuer Zustand s', Zeichen a' schreiben, Kopf gemäß P bewegen
  - Stop wenn $P=h$
- **Ergebnis**
  - Längstes Wort auf Band ohne Leerzeichen am Anfang und Ende

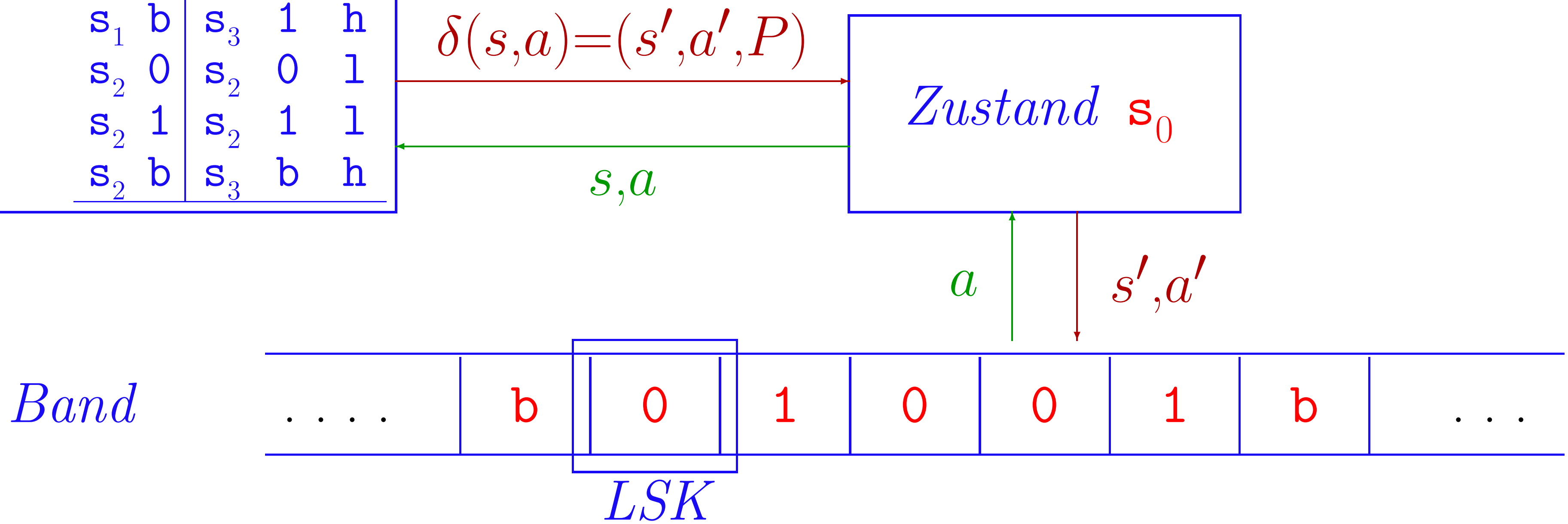
Achtung! Details in Literatur unterschiedlich

Abarbeitung von Turing-Programmen

Eingabewort `01001`

δ

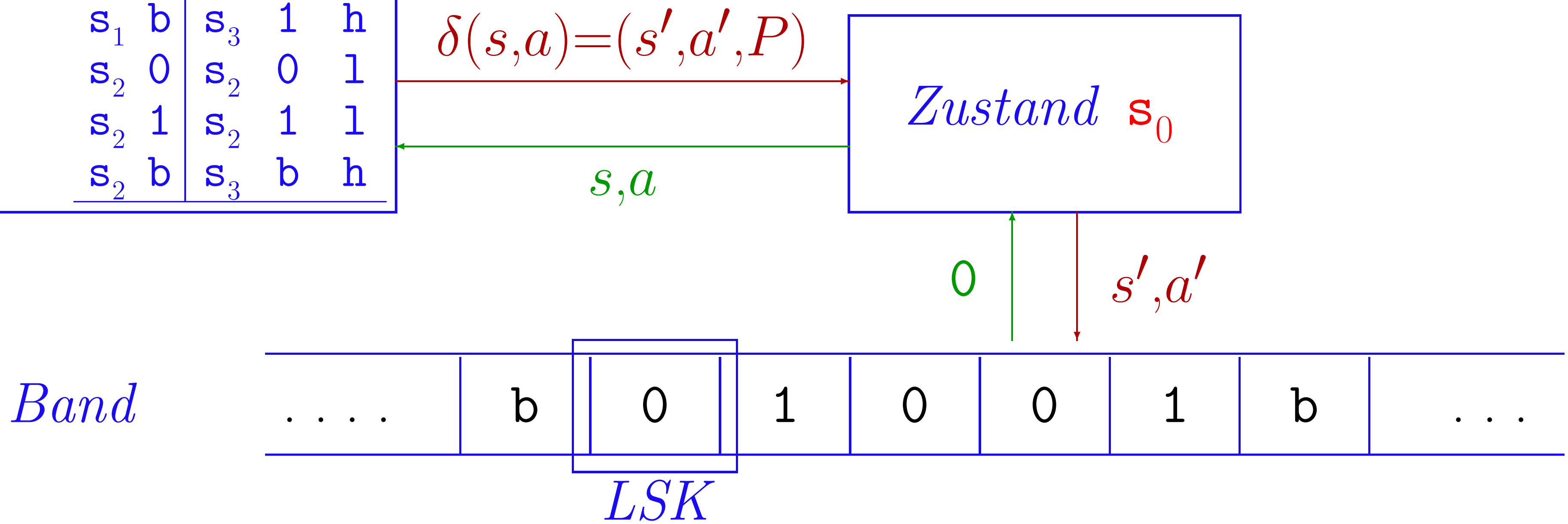
s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

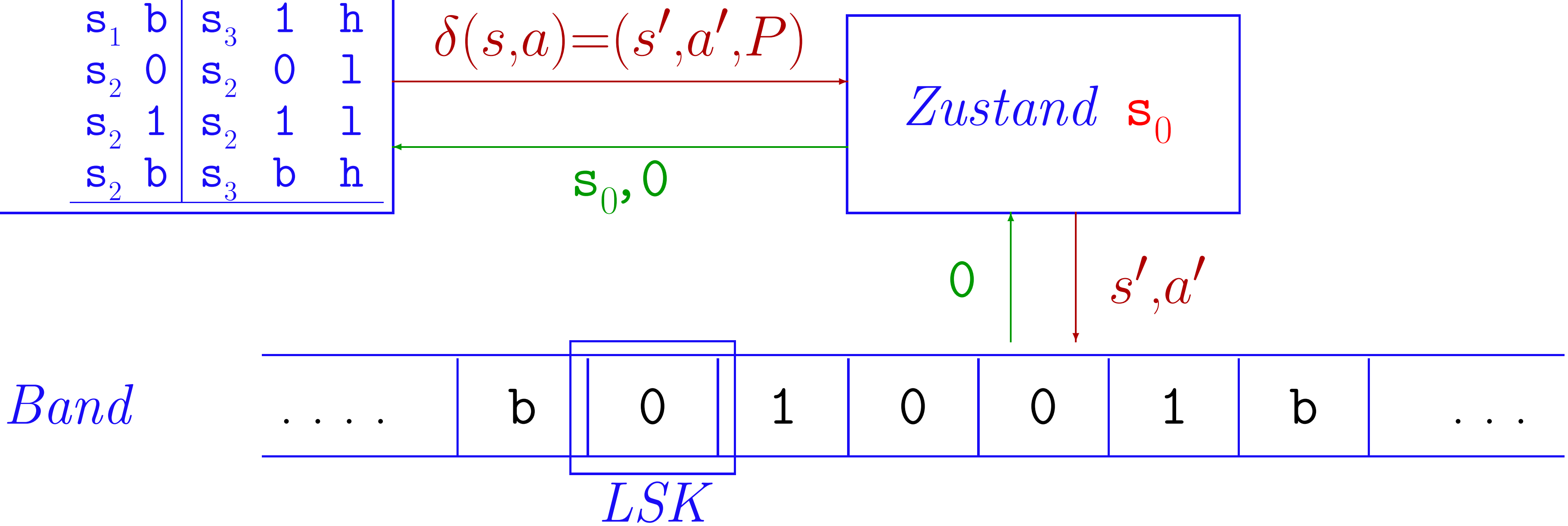
s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

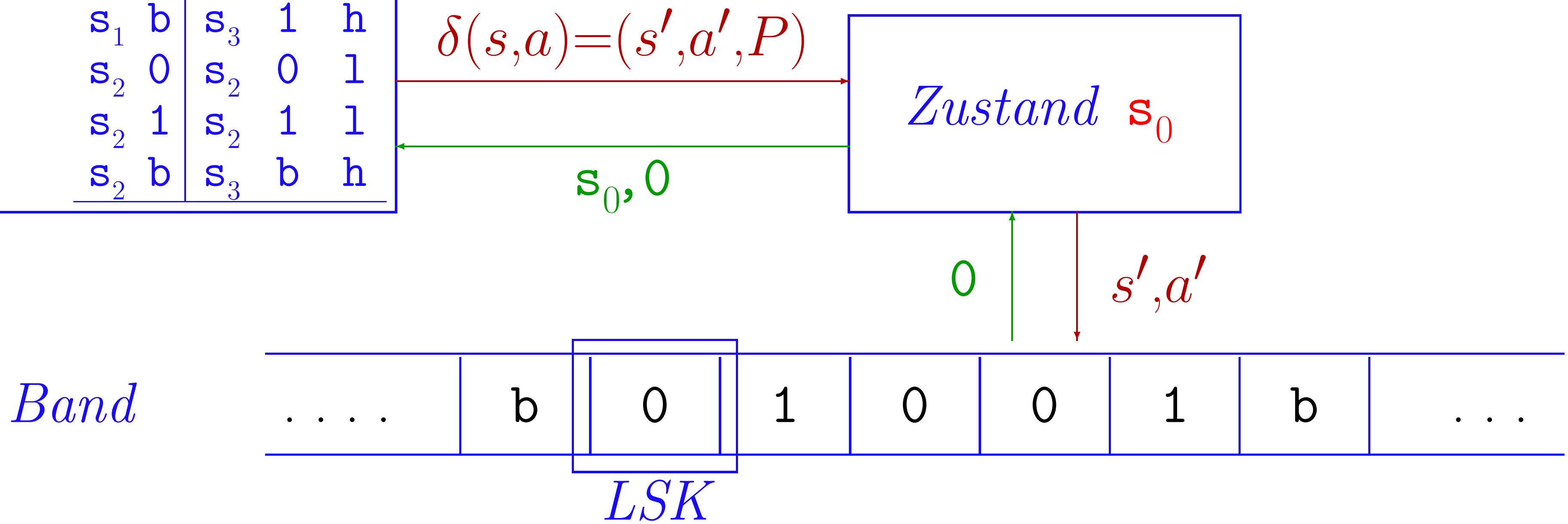
s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

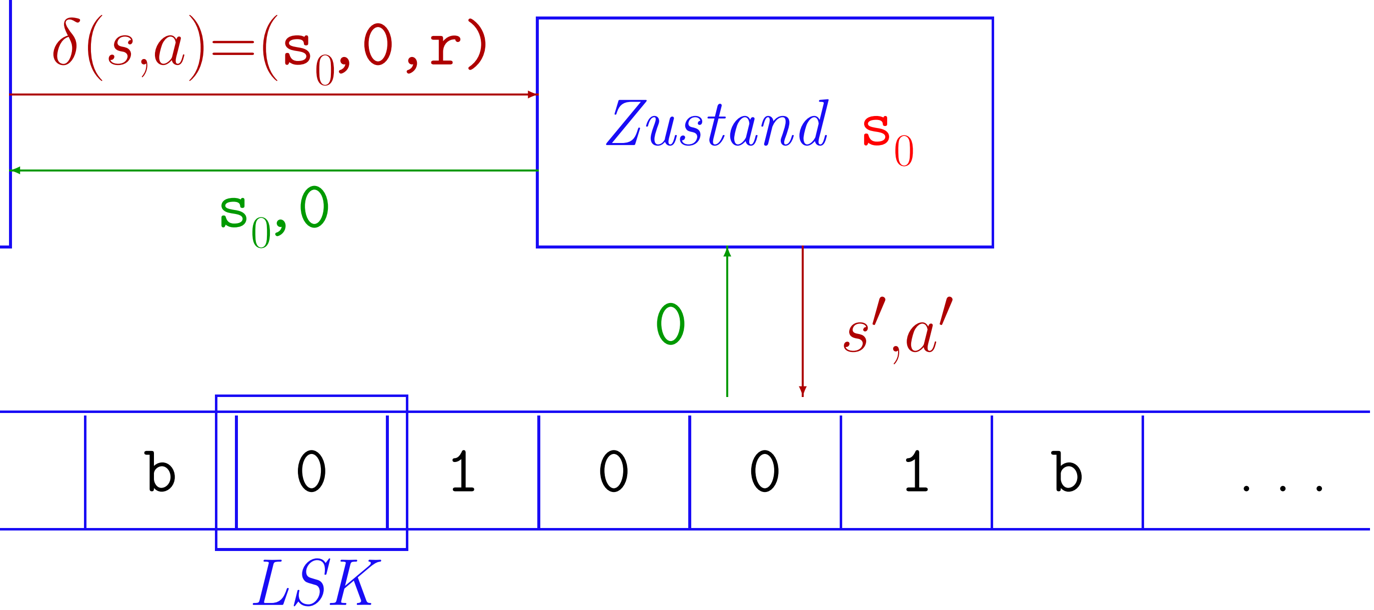
	s	a	s'	a'	P
$\rightarrow$	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

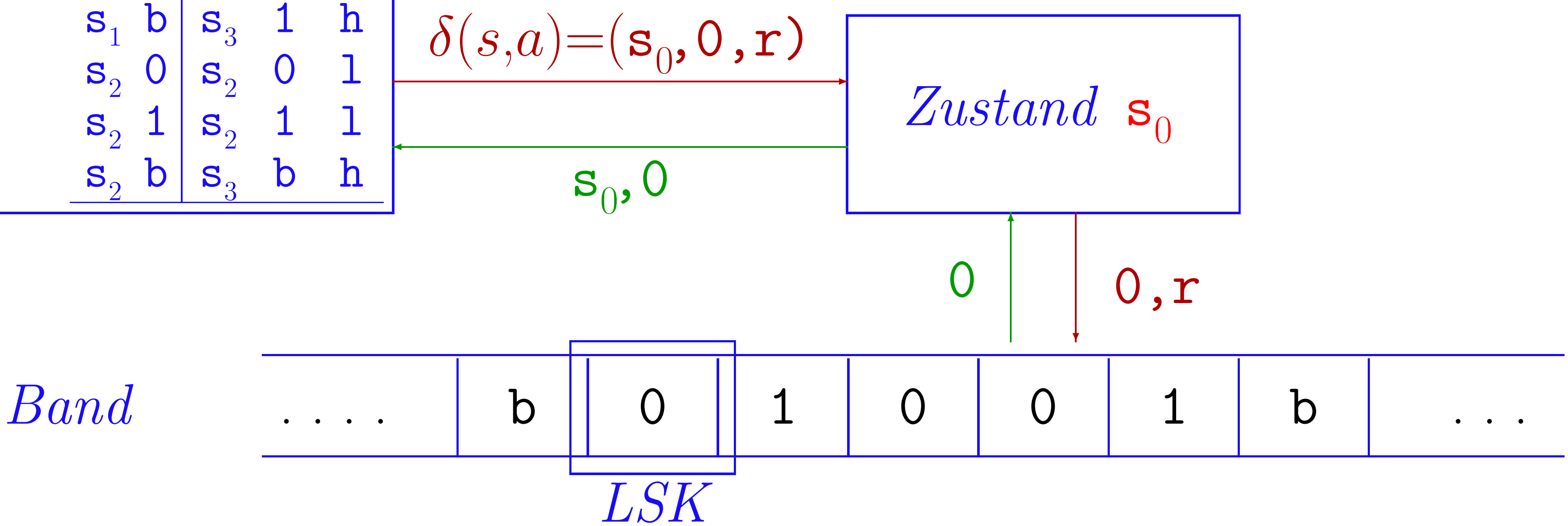
	s	a	s'	a'	P
$\rightarrow$	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

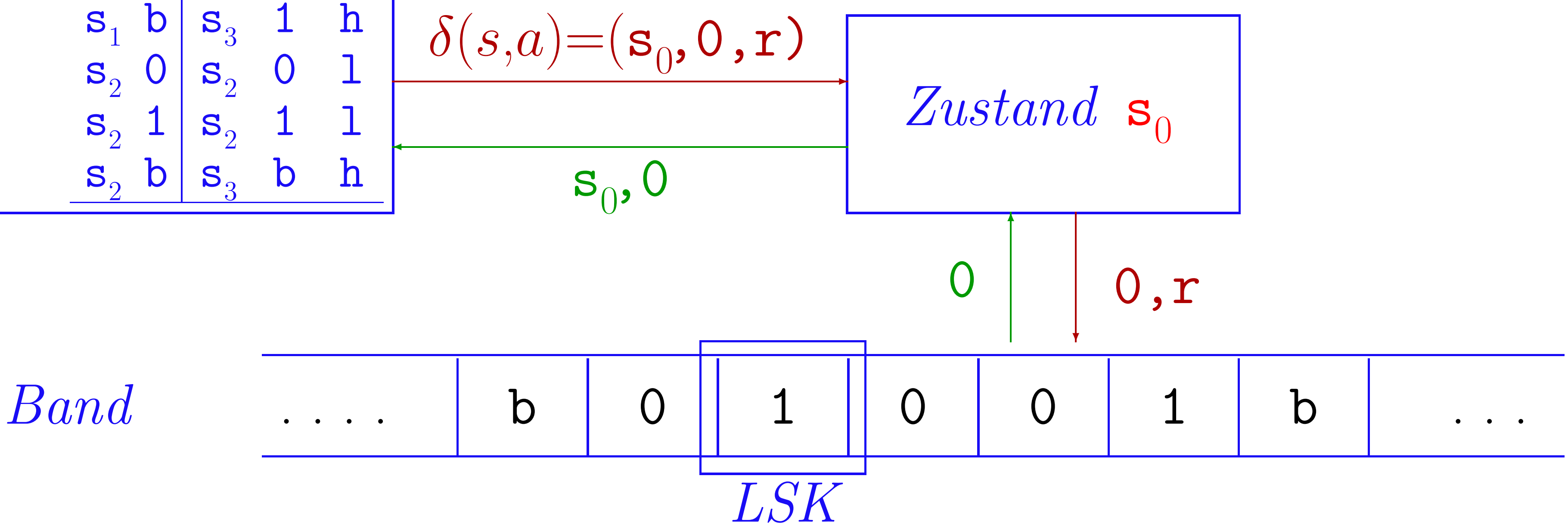
s	a	s'	a'	P
→ s$_0$	0	s$_0$	0	r
s$_0$	1	s$_0$	1	r
s$_0$	b	s$_1$	b	l
s$_1$	1	s$_1$	0	l
s$_1$	0	s$_2$	1	l
s$_1$	b	s$_3$	1	h
s$_2$	0	s$_2$	0	l
s$_2$	1	s$_2$	1	l
s$_2$	b	s$_3$	b	h



Abarbeitung von Turing-Programmen

δ

	s	a	s'	a'	P
$\rightarrow$	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h

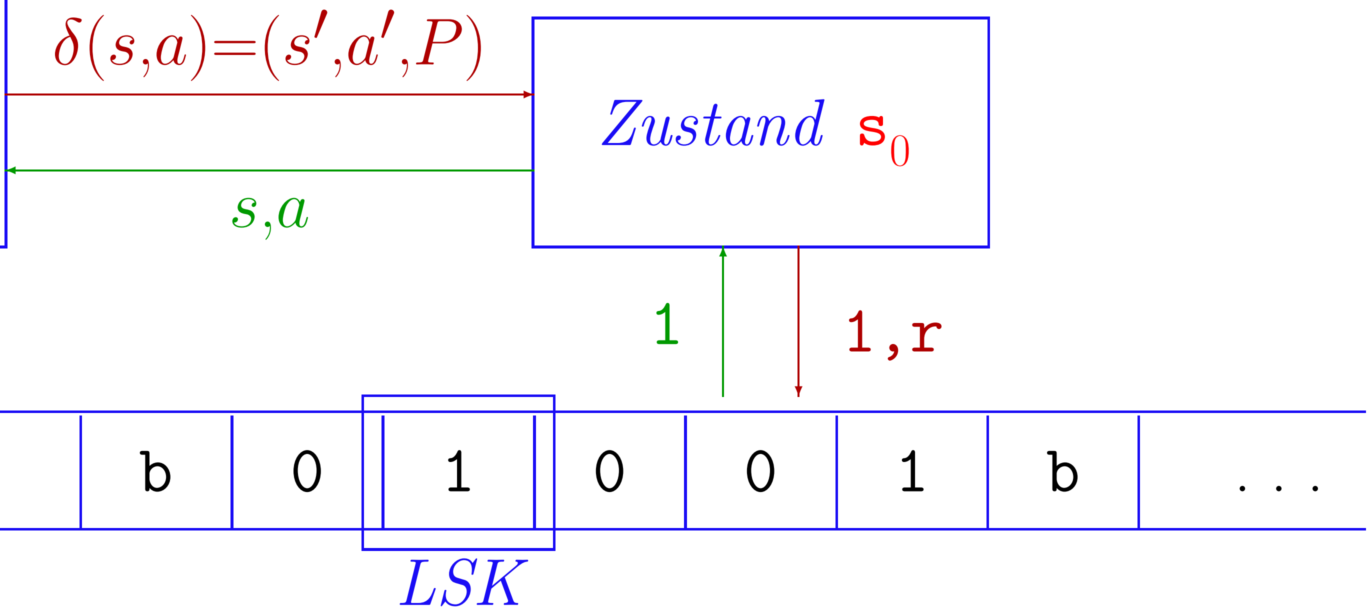


Abarbeitung von Turing-Programmen

δ

	s	a	s'	a'	P
	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\rightarrow$	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h

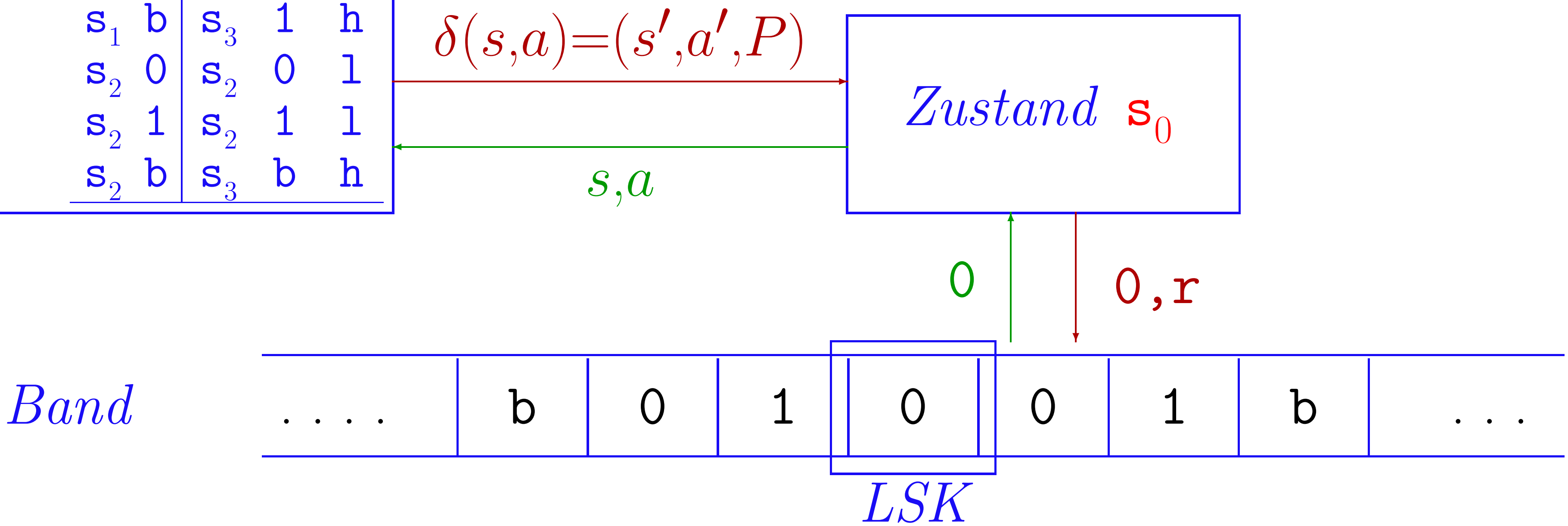
Band



Abarbeitung von Turing-Programmen

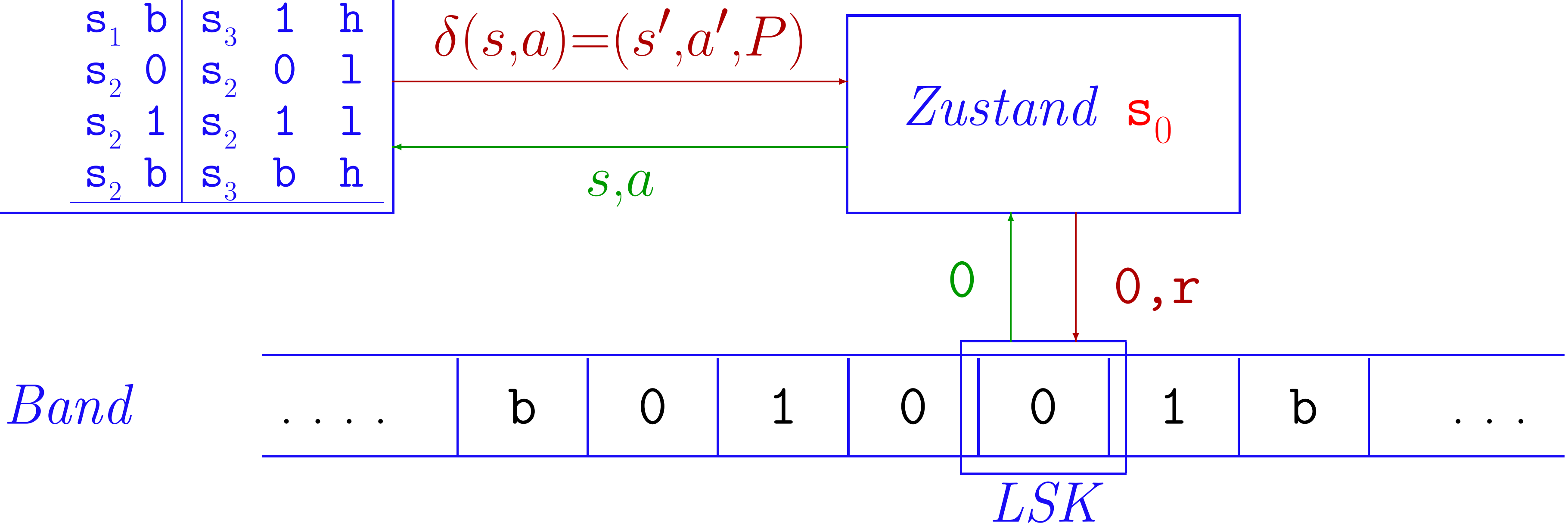
δ

s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h

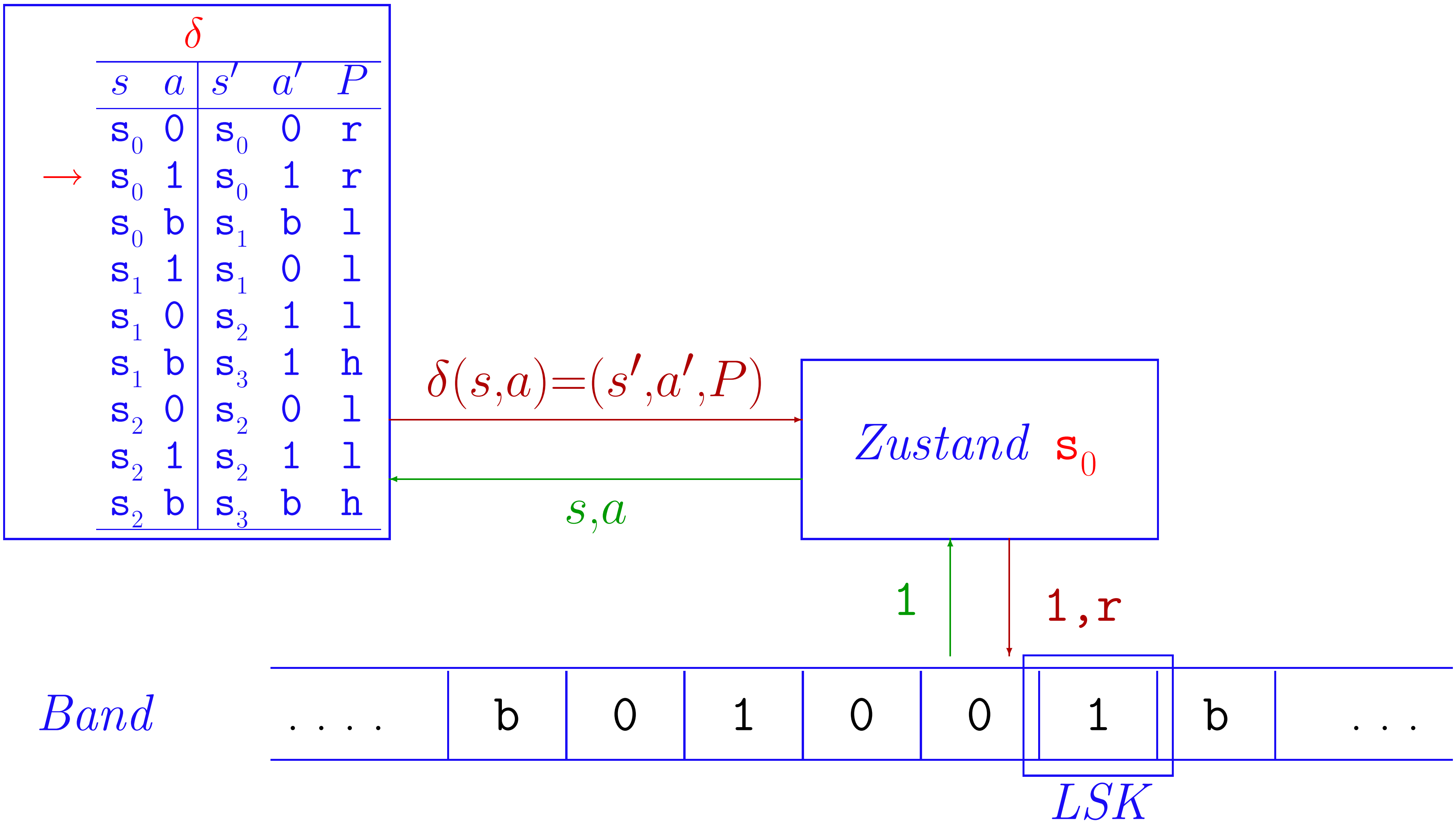


Abarbeitung von Turing-Programmen

	s	a	s'	a'	P
			δ		
$\rightarrow$	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

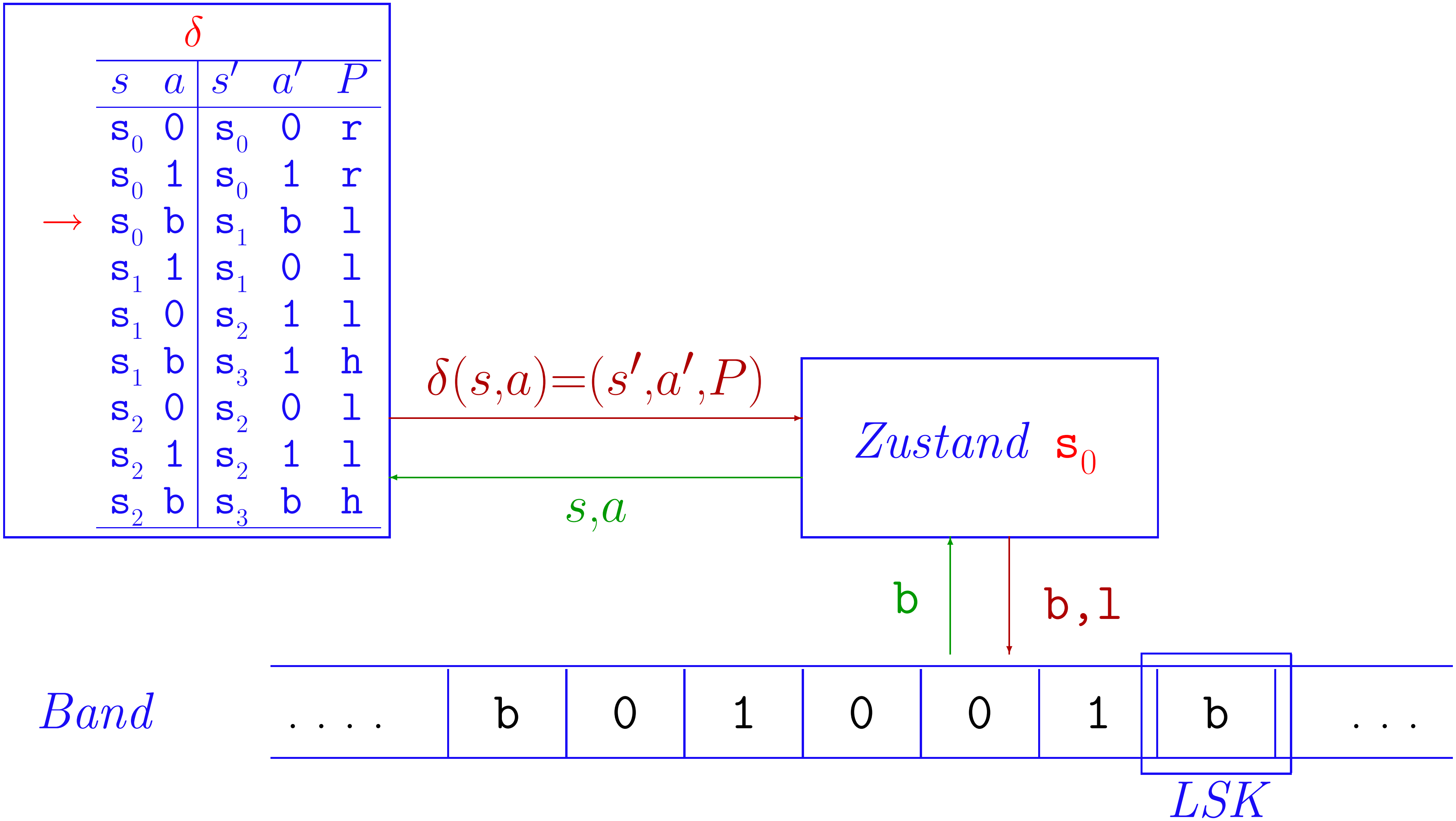


	s	a	s'	a'	P
	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\rightarrow$	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h

Abarbeitung von Turing-Programmen

δ

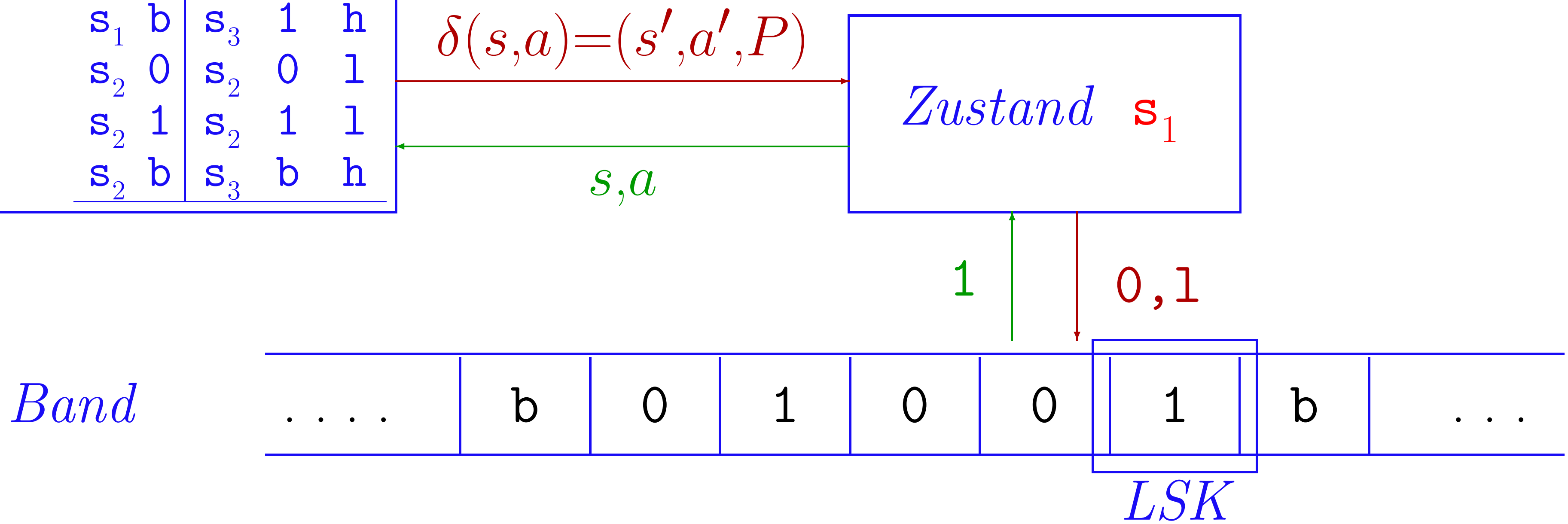
s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
→ $\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

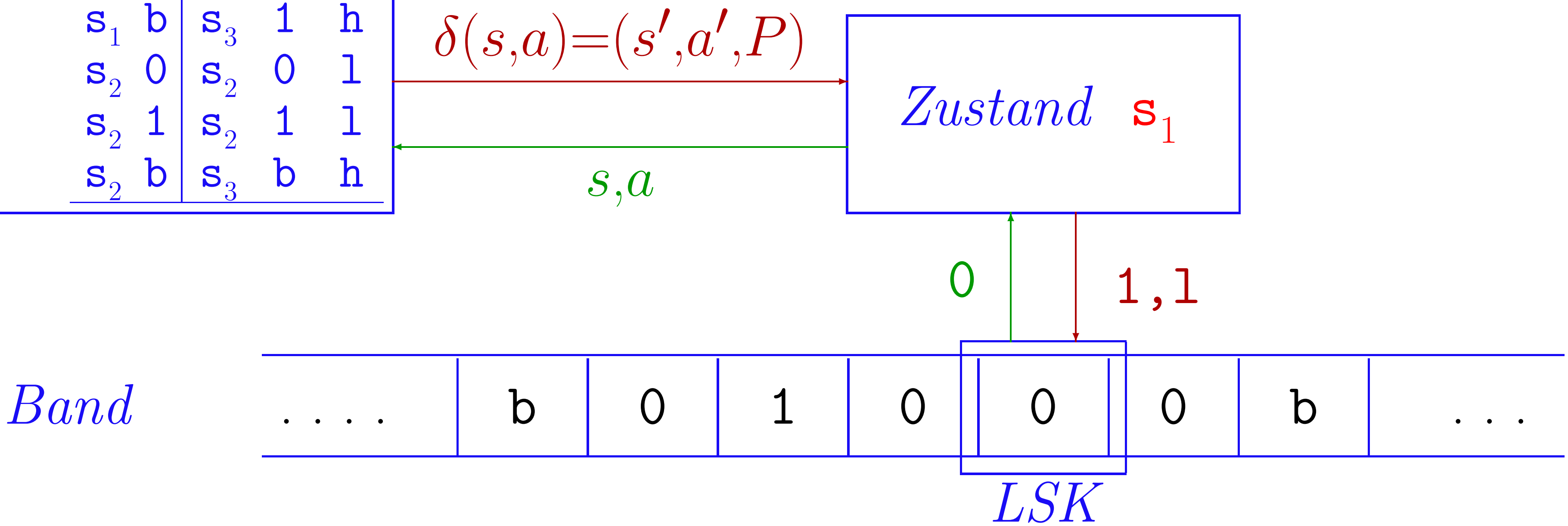
s	a	s'	a'	P
s_0	0	s_0	0	r
s_0	1	s_0	1	r
s_0	b	s_1	b	l
s_1	1	s_1	0	l
s_1	0	s_2	1	l
s_1	b	s_3	1	h
s_2	0	s_2	0	l
s_2	1	s_2	1	l
s_2	b	s_3	b	h



Abarbeitung von Turing-Programmen

δ

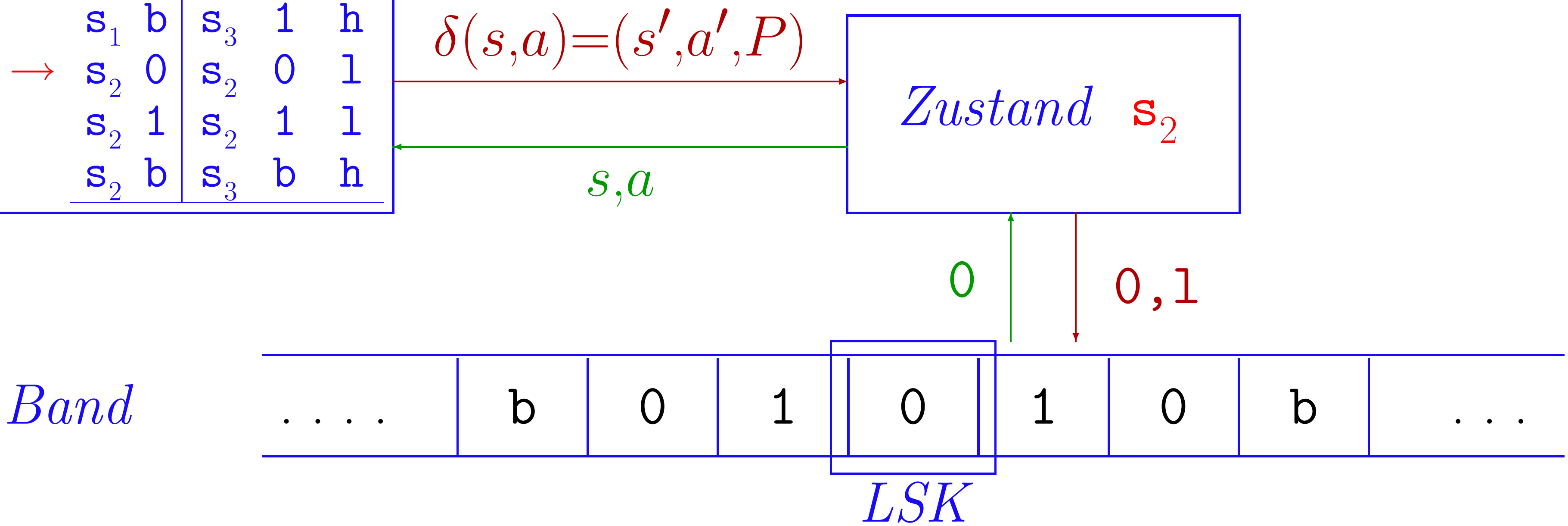
	s	a	s'	a'	P
	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\rightarrow$	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

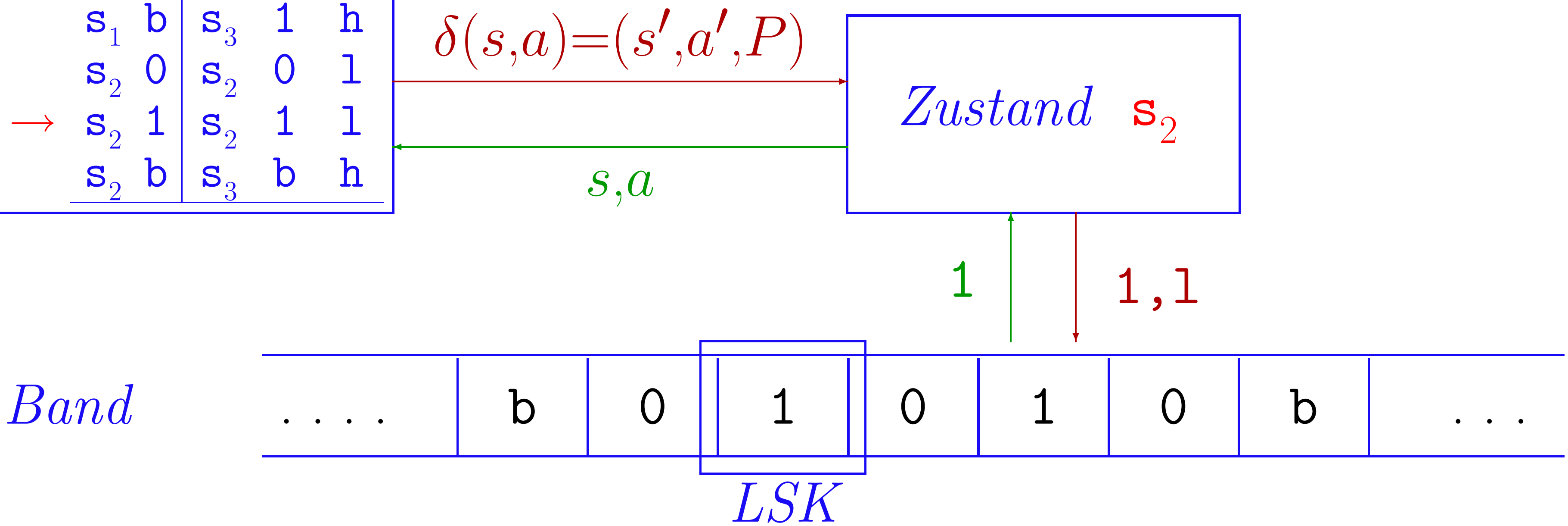
	s	a	s'	a'	P
	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\rightarrow$	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

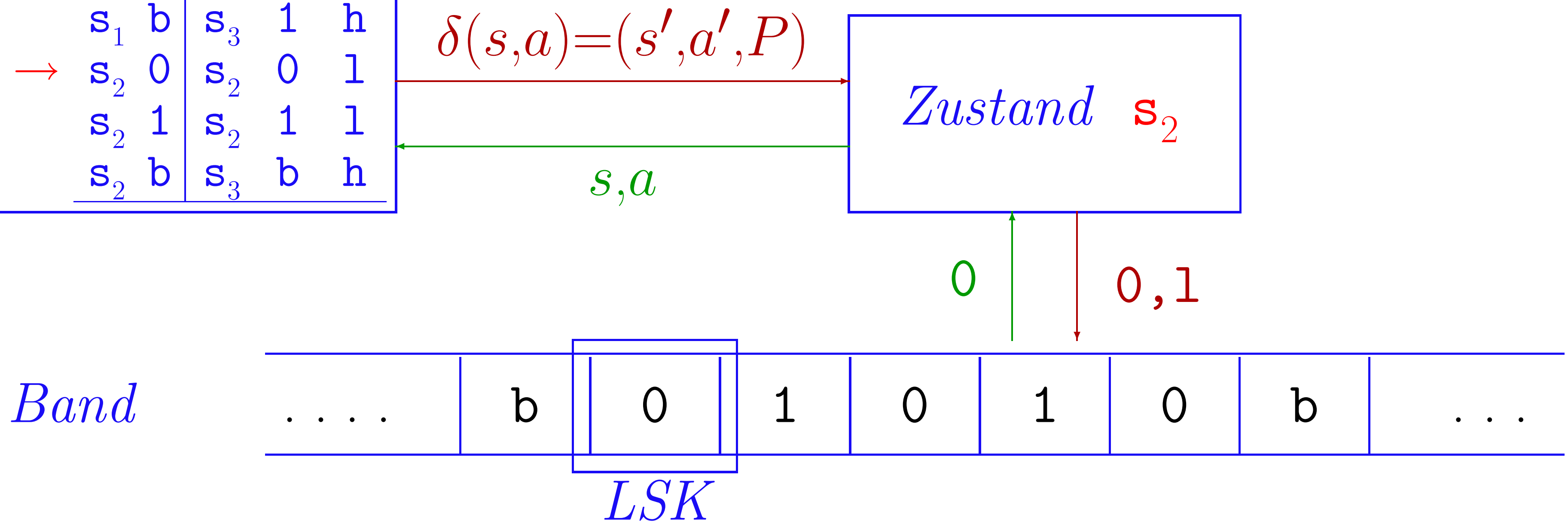
s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
→ $\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Abarbeitung von Turing-Programmen

δ

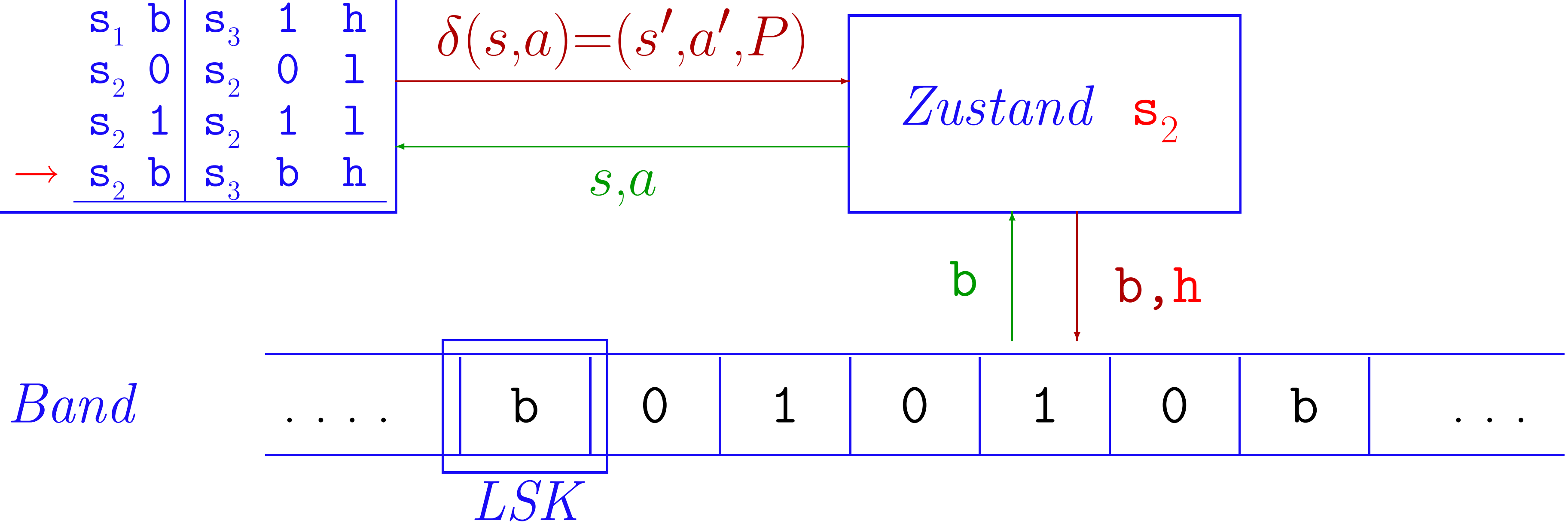
	s	a	s'	a'	P
	s_0	0	s_0	0	r
	s_0	1	s_0	1	r
	s_0	b	s_1	b	l
	s_1	1	s_1	0	l
	s_1	0	s_2	1	l
	s_1	b	s_3	1	h
$\rightarrow$	s_2	0	s_2	0	l
	s_2	1	s_2	1	l
	s_2	b	s_3	b	h



Abarbeitung von Turing-Programmen

δ

	s	a	s'	a'	P
	$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
	$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
	$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
	$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
	$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
	$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
	$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
	$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\longrightarrow$	$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h

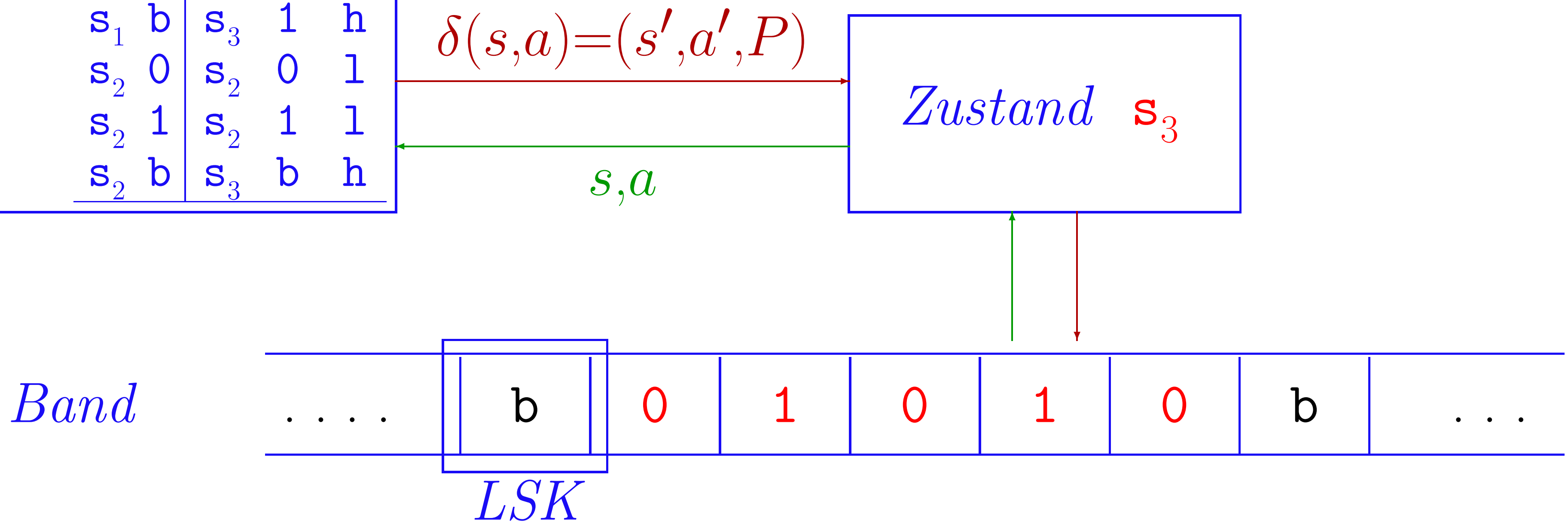


Abarbeitung von Turing-Programmen

Ausgabewort `01010`

δ

s	a	s'	a'	P
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	l
$\mathtt{s}_1$	b	$\mathtt{s}_3$	1	h
$\mathtt{s}_2$	0	$\mathtt{s}_2$	0	l
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	l
$\mathtt{s}_2$	b	$\mathtt{s}_3$	b	h



Verarbeitung von Turing-Programmen – präzisiert

- **Definiere Konfiguration von τ** Definition B
  – Schnapschuß der Turingmaschine τ zu einem gegebenen Zeitpunkt
    · aktueller Zustand + Bandinhalt + Kopfposition
  – K_τ: Menge aller Konfigurationen von τ

Verarbeitung von Turing-Programmen – präzisiert

- **Definiere Konfiguration von τ** Definition B
  - Schnapschuß der Turingmaschine τ zu einem gegebenen Zeitpunkt
    - aktueller Zustand + Bandinhalt + Kopfposition
  - K_τ: Menge aller Konfigurationen von τ
- **Definiere Arbeitsweise von τ** Definition C
  - Anfangskonfiguration $\alpha(w)$ für Eingabeworte $w \in X^*$
  - Nachfolgekonfiguration (Arbeitsschritt) $\hat{\delta}:K_\tau \to K_\tau$
  - Ausgabefunktion (Ergebnis) $\omega:K_\tau \to \Gamma^*$

Verarbeitung von Turing-Programmen – präzisiert

- **Definiere Konfiguration von τ** Definition B
  - Schnapschuß der Turingmaschine τ zu einem gegebenen Zeitpunkt
    - aktueller Zustand + Bandinhalt + Kopfposition
  - K_τ: Menge aller Konfigurationen von τ
- **Definiere Arbeitsweise von τ** Definition C
  - Anfangskonfiguration $\alpha(w)$ für Eingabeworte $w \in X^*$
  - Nachfolgekonfiguration (Arbeitsschritt) $\hat{\delta}:K_\tau \to K_\tau$
  - Ausgabefunktion (Ergebnis) $\omega:K_\tau \to \Gamma^*$
- **Definiere die von τ berechnete Funktion h_τ** Definition D

Konfiguration von Turing-Programmen

Definition B

- **Eine Konfiguration ist ein Tripel $\boldsymbol{\kappa} = (s,f,i)$ mit**
  - $s \in S$ aktueller Zustand
  - $f{:}\mathbb{Z} \to \Gamma$ Bandinhaltsfunktion

    $f(n) \equiv$ Inhalt der n-ten Bandzelle

    $(f(j) = b$ für fast alle $j)$
  - $i \in \mathbb{Z}$ Kopfposition

  Alternative Repräsentation: Tripel (s,u,v) mit
  – s aktueller Zustand,
  – u String links vom Kopf (von rechts nach links),
  – v String rechts vom Kopf

- **$\boldsymbol{K_\tau}$: Menge aller Konfigurationen von τ**

Arbeitsweise von Turingmaschinen

Definition C

- **Anfangskonfiguration** $\boldsymbol{\alpha}$**:**$X^*{\rightarrow}K_\tau$
  - Für ein Eingabewort $w = w_0 w_1 .. w_k$ ist $\alpha(w) = (s_0, f_w, 0)$,
    mit $f_w(j) = \begin{cases} w_j & \text{falls } j \in \{0,..,k\}, \\ b & \text{sonst} \end{cases}$

- **Anfangskonfiguration $\boldsymbol{\alpha}$:** $X^* \to K_\tau$
  - Für ein Eingabewort $w = w_0 w_1 .. w_k$ ist $\alpha(w) = (s_0, f_w, 0)$,

    mit $f_w(j) = \begin{cases} w_j & \text{falls } j \in \{0,..,k\}, \\ b & \text{sonst} \end{cases}$

- **Nachfolgekonfiguration $\hat{\boldsymbol{\delta}}$:** $K_\tau \to K_\tau$
  - Für eine Konfiguration $\kappa = (s, f, i)$ mit $\delta(s, f(i)) = (s', a', P)$ ist $\hat{\delta}(\kappa) = (s', f', i')$

    wobei $f'(j) = \begin{cases} a & \text{falls } j = i, \\ f(j) & \text{sonst} \end{cases}$ und $i' = \begin{cases} i+1 & \text{falls } P = r, \\ i-1 & \text{falls } P = l, \\ i & \text{falls } P = h \end{cases}$

Arbeitsweise von Turingmaschinen

Definition C

- **Anfangskonfiguration $\boldsymbol{\alpha}$:$X^*{\longrightarrow}K_\tau$**
  - Für ein Eingabewort $w = w_0 w_1 ... w_k$ ist $\alpha(w) = (s_0, f_w, 0)$,

    mit $f_w(j) = \begin{cases} w_j & \text{falls } j \in \{0,..,k\}, \\ b & \text{sonst} \end{cases}$

- **Nachfolgekonfiguration $\hat{\boldsymbol{\delta}}$:$K_\tau{\longrightarrow}K_\tau$**
  - Für eine Konfiguration $\kappa{=}(s,f,i)$ mit $\delta(s,f(i)){=}(s',a',P)$ ist $\hat{\delta}(\kappa){=}(s',f',i')$

    wobei $f'(j) = \begin{cases} a & \text{falls } j{=}i, \\ f(j) & \text{sonst} \end{cases}$ und $i' = \begin{cases} i{+}1 & \text{falls } P{=}r, \\ i{-}1 & \text{falls } P{=}l, \\ i & \text{falls } P{=}h \end{cases}$

- **Ausgabefunktion $\boldsymbol{\omega}$:$K_\tau{\longrightarrow}\Gamma^*$**
  - Für eine Konfiguration $\kappa{=}(s,f,i)$ ist

    $\omega(\kappa) = \begin{cases} \epsilon & \text{falls } f(j){=}b \text{ für alle } j, \\ f(k)f(k{+}1)...f(k{+}n) & \text{sonst} \end{cases}$

    wobei $k{=}\max\{i | \forall j{<}i \ f(j){=}b\}$ und $n{=}\min\{i | \forall j{>}k{+}i \ f(j){=}b\}$

Semantik von Turing-Programmen

Definition D

- **Intuitive Beschreibung**
  - Eingabe $\alpha(w)$
  - Wiederholte Anwendung von $\hat{\delta}$
  - Ausgabe $\omega(\kappa)$, wenn Stop-Konfiguration κ erreicht wird.
  - Undefiniert (Endlosschleife), andernfalls

Semantik von Turing-Programmen

Definition D

- **Intuitive Beschreibung**
  - Eingabe $\alpha(w)$
  - Wiederholte Anwendung von $\hat{\delta}$
  - Ausgabe $\omega(\kappa)$, wenn Stop-Konfiguration κ erreicht wird.
  - Undefiniert (Endlosschleife), andernfalls

- **Mathematische Semantik von $\tau = (S, X, \Gamma, \delta, s_0, b)$**
  - Die von τ = **berechnete Funktion h_τ**$:X^*\to\Gamma^*$ ist definiert durch

$$h_\tau(w) = \begin{cases} \omega(\hat{\delta}^{m+1}(\alpha(w))) & \text{falls } m{=}\min\{j \mid \exists s, f, i, s', a' \;\; \hat{\delta}^j(\alpha(w)){=}(s, f, i) \\ & \qquad\qquad\qquad \text{und } \; \delta(s, f(i)){=}(s', a', h)\} \\ & \text{existiert}, \\ \perp & \text{sonst} \end{cases}$$

Semantik von Turing-Programmen

Definition D

- **Intuitive Beschreibung**
  - Eingabe $\alpha(w)$
  - Wiederholte Anwendung von $\hat{\delta}$
  - Ausgabe $\omega(\kappa)$, wenn Stop-Konfiguration κ erreicht wird.
  - Undefiniert (Endlosschleife), andernfalls
- **Mathematische Semantik von $\tau = (S, X, \Gamma, \delta, s_0, b)$**
  - Die von τ = **berechnete Funktion h_τ**:$X^* \to \Gamma^*$ ist definiert durch

$$h_\tau(w) = \begin{cases} \omega(\hat{\delta}^{m+1}(\alpha(w))) & \text{falls } m{=}\min\{j \mid \exists s, f, i, s', a' \;\; \hat{\delta}^j(\alpha(w)){=}(s,f,i) \\ & \qquad\qquad \text{und } \;\delta(s, f(i)){=}(s', a', h)\} \\ & \text{existiert,} \\ \perp & \text{sonst} \end{cases}$$

Definition E

  - **Definitionsbereich** von τ: $\{w \in X^* \mid h_\tau(w) \neq \perp\}$ (Haltebereich, domain)

Semantik von Turing-Programmen

Definition D

- **Intuitive Beschreibung**
  - Eingabe $\alpha(w)$
  - Wiederholte Anwendung von $\hat{\delta}$
  - Ausgabe $\omega(\kappa)$, wenn Stop-Konfiguration κ erreicht wird.
  - Undefiniert (Endlosschleife), andernfalls

- **Mathematische Semantik von $\tau = (S, X, \Gamma, \delta, s_0, b)$**
  - Die von τ = **berechnete Funktion h_τ**$:X^*\to\Gamma^*$ ist definiert durch

$$h_\tau(w) = \begin{cases} \omega(\hat{\delta}^{m+1}(\alpha(w))) & \text{falls } m = \min\{j \mid \exists s, f, i, s', a' \;\; \hat{\delta}^j(\alpha(w)) = (s, f, i) \\ & \qquad \text{und } \; \delta(s, f(i)) = (s', a', h)\} \\ & \text{existiert,} \\ \perp & \text{sonst} \end{cases}$$

Definition E

  - **Definitionsbereich** von τ: $\{w \in X^* \mid h_\tau(w) \neq \perp\}$ (Haltebereich, domain)
  - **Wertebereich** von τ: $\{v \in \Gamma^* \mid \exists w \in X^* \; h_\tau(w) = v\}$ (Ergebnisbereich, range)

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	1	h

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	1	h

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine 1 an ("Bierdeckelmaschine")

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	1	h

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine 1 an ("Bierdeckelmaschine")

- **Mathematische Analyse**:

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	1	h

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine 1 an ("Bierdeckelmaschine")

- **Mathematische Analyse**:
  – Anfangskonfiguration: $\alpha(\mathtt{1}^n) = (\mathtt{s}_0, f_n, 0)$, wobei $f_n(j) = \begin{cases} \mathtt{1} & \text{falls } j \in \{0,..,n-1\}, \\ \mathtt{b} & \text{sonst} \end{cases}$

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{r}$
$\mathtt{s}_0$	$\mathtt{b}$	$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{h}$

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine $\mathtt{1}$ an ("Bierdeckelmaschine")

- **Mathematische Analyse**:
  - Anfangskonfiguration: $\alpha(\mathtt{1}^n) = (\mathtt{s}_0, f_n, 0)$, wobei $f_n(j) = \begin{cases} \mathtt{1} & \text{falls } j \in \{0,..,n-1\}, \\ \mathtt{b} & \text{sonst} \end{cases}$
  - Nachfolgekonfigurationen: $\hat{\delta}(\mathtt{s}_0, f_n, j) = \begin{cases} (\mathtt{s}_0, f_n, j+1) & \text{falls } j \in \{0,..,n-1\}, \\ (\mathtt{s}_0, f_{n+1}, n) & \text{falls } j = n \end{cases}$

Beispiele für Turing-Maschinen

- $\tau_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	1	h

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine $\mathtt{1}$ an ("Bierdeckelmaschine")

- **Mathematische Analyse**:
  - Anfangskonfiguration: $\alpha(\mathtt{1}^n) = (\mathtt{s}_0, f_n, 0)$, wobei $f_n(j) = \begin{cases} \mathtt{1} & \text{falls } j \in \{0,..,n-1\}, \\ \mathtt{b} & \text{sonst} \end{cases}$
  - Nachfolgekonfigurationen: $\hat{\delta}(\mathtt{s}_0, f_n, j) = \begin{cases} (\mathtt{s}_0, f_n, j+1) & \text{falls } j \in \{0,..,n-1\}, \\ (\mathtt{s}_0, f_{n+1}, n) & \text{falls } j=n \end{cases}$
  - Terminierung: $\min\{j \mid \hat{\delta}^j(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_n, j) \wedge \delta(\mathtt{s}_0, f_n(j)) = (\mathtt{s}_0, \mathtt{b}, \mathtt{h})\} = n$

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{r}$
$\mathtt{s}_0$	$\mathtt{b}$	$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{h}$

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine $\mathtt{1}$ an ("Bierdeckelmaschine")

- **Mathematische Analyse**:
  - Anfangskonfiguration: $\alpha(\mathtt{1}^n) = (\mathtt{s}_0, f_n, 0)$, wobei $f_n(j) = \begin{cases} \mathtt{1} & \text{falls } j \in \{0,..,n-1\}, \\ \mathtt{b} & \text{sonst} \end{cases}$
  - Nachfolgekonfigurationen: $\hat{\delta}(\mathtt{s}_0, f_n, j) = \begin{cases} (\mathtt{s}_0, f_n, j+1) & \text{falls } j \in \{0,..,n-1\}, \\ (\mathtt{s}_0, f_{n+1}, n) & \text{falls } j = n \end{cases}$
  - Terminierung: $\min\{j \mid \hat{\delta}^j(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_n, j) \land \delta(\mathtt{s}_0, f_n(j)) = (\mathtt{s}_0, \mathtt{b}, \mathtt{h})\} = n$
  - Ergebnis: $\hat{\delta}^{n+1}(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_{n+1}, n)$

Beispiele für Turing-Maschinen

- $\boldsymbol{\tau}_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{r}$
$\mathtt{s}_0$	$\mathtt{b}$	$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{h}$

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine $\mathtt{1}$ an ("Bierdeckelmaschine")

- **Mathematische Analyse**:
  - Anfangskonfiguration: $\alpha(\mathtt{1}^n) = (\mathtt{s}_0, f_n, 0)$, wobei $f_n(j) = \begin{cases} \mathtt{1} & \text{falls } j \in \{0,..,n-1\}, \\ \mathtt{b} & \text{sonst} \end{cases}$
  - Nachfolgekonfigurationen: $\hat{\delta}(\mathtt{s}_0, f_n, j) = \begin{cases} (\mathtt{s}_0, f_n, j+1) & \text{falls } j \in \{0,..,n-1\}, \\ (\mathtt{s}_0, f_{n+1}, n) & \text{falls } j=n \end{cases}$
  - Terminierung: $\min\{j \mid \hat{\delta}^j(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_n, j) \wedge \delta(\mathtt{s}_0, f_n(j)) = (\mathtt{s}_0, \mathtt{b}, \mathtt{h})\} = n$
  - Ergebnis: $\hat{\delta}^{n+1}(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_{n+1}, n)$
  - Ausgabefunktion: $\omega(\mathtt{s}_0, f_{n+1}, \mathrm{n}) = \mathtt{1}^{n+1}$
    $(\max\{i \mid \forall j{<}i \; f_{n+1}(j){=}b\}{=}0, \;\; \min\{i \mid \forall j{>}i \; f_{n+1}(j){=}b\}{=}n{+}1)$

Beispiele für Turing-Maschinen

- $\tau_1 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_1, \mathtt{s}_0, \mathtt{b})$ mit $\delta_1 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	1	h

  Fügt am Ende eines Wortes $w \in \mathtt{1}^*$ eine $\mathtt{1}$ an ("Bierdeckelmaschine")

- **Mathematische Analyse**:
  - Anfangskonfiguration: $\alpha(\mathtt{1}^n) = (\mathtt{s}_0, f_n, 0)$, wobei $f_n(j) = \begin{cases} \mathtt{1} & \text{falls } j \in \{0,..,n-1\}, \\ \mathtt{b} & \text{sonst} \end{cases}$
  - Nachfolgekonfigurationen: $\hat{\delta}(\mathtt{s}_0, f_n, j) = \begin{cases} (\mathtt{s}_0, f_n, j+1) & \text{falls } j \in \{0,..,n-1\}, \\ (\mathtt{s}_0, f_{n+1}, n) & \text{falls } j=n \end{cases}$
  - Terminierung: $\min\{j \mid \hat{\delta}^j(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_n, j) \wedge \delta(\mathtt{s}_0, f_n(j)) = (\mathtt{s}_0, \mathtt{b}, \mathtt{h})\} = n$
  - Ergebnis: $\hat{\delta}^{n+1}(\mathtt{s}_0, f_n, 0) = (\mathtt{s}_0, f_{n+1}, n)$
  - Ausgabefunktion: $\omega(\mathtt{s}_0, f_{n+1}, \mathrm{n}) = \mathtt{1}^{n+1}$
    $(\max\{i \mid \forall j<i\ f_{n+1}(j)=b\}=0,\ \min\{i \mid \forall j>i\ f_{n+1}(j)=b\}=n+1)$

$\Downarrow$

$$\boxed{h_{\tau_1}(\mathtt{1}^n) = \mathtt{1}^{n+1} \text{ für alle } n, \text{ Definitionsbereich } \{\mathtt{1}\}^*, \text{ Wertebereich } \{\mathtt{1}\}^+}$$

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details
- Präzise genug. um Details rekonstruieren zu können

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details
- Präzise genug. um Details rekonstruieren zu können
- Knapp genug, um übersichtlich und merkbar zu sein

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details
- Präzise genug. um Details rekonstruieren zu können
- Knapp genug, um übersichtlich und merkbar zu sein
- Gedankensprünge sind erlaubt, wenn Sie die Materie gut genug verstehen, daß Sie nichts mehr falsch machen können

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details
- Präzise genug. um Details rekonstruieren zu können
- Knapp genug, um übersichtlich und merkbar zu sein
- Gedankensprünge sind erlaubt, wenn Sie die Materie gut genug verstehen, daß Sie nichts mehr falsch machen können
  ... es reicht nicht, daß Sie es einmal richtig gemacht haben

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details
- Präzise genug. um Details rekonstruieren zu können
- Knapp genug, um übersichtlich und merkbar zu sein
- Gedankensprünge sind erlaubt, wenn Sie die Materie gut genug verstehen, daß Sie nichts mehr falsch machen können
  ... es reicht nicht, daß Sie es einmal richtig gemacht haben
- Tip: ausführliche Lösungen entwickeln, bis Sie genug Erfahrung haben.
  Für Präsentation zentrale Gedanken aus Lösung extrahieren

Exkurs: Wie genau muss man sein?

Ein Beweis ist ein Argument, das den Leser überzeugt

- Nicht notwendig formal oder mit allen Details
- Präzise genug. um Details rekonstruieren zu können
- Knapp genug, um übersichtlich und merkbar zu sein
- Gedankensprünge sind erlaubt, wenn Sie die Materie gut genug verstehen, daß Sie nichts mehr falsch machen können
  … es reicht nicht, daß Sie es einmal richtig gemacht haben
- Tip: ausführliche Lösungen entwickeln, bis Sie genug Erfahrung haben.
  Für Präsentation zentrale Gedanken aus Lösung extrahieren
- Test: verstehen Ihre Kommilitonen Ihre Lösung und warum sie funktioniert?

Beispiele für Turing-Maschinen II

- $\boldsymbol{\tau}_2 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b,1}\}, \delta_2, \mathtt{s}_0, \mathtt{b})$ mit $\delta_2 =$

s	a	s'	a'	P
$\mathtt{s}_0$	$\mathtt{1}$	$\mathtt{s}_0$	$\mathtt{b}$	$\mathtt{r}$
$\mathtt{s}_0$	$\mathtt{b}$	$\mathtt{s}_0$	$\mathtt{b}$	$\mathtt{h}$

Beispiele für Turing-Maschinen II

- $\tau_2 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_2, \mathtt{s}_0, \mathtt{b})$ mit $\delta_2 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	b	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h

Löscht ein Wort $w \in \mathtt{1}^*$:

$h_{\tau_2}(w) = \epsilon$ für alle w, Definitionsbereich $\{\mathtt{1}\}^*$, Wertebereich $\{\epsilon\}$

Beispiele für Turing-Maschinen II

- $\boldsymbol{\tau_2} = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_2, \mathtt{s}_0, \mathtt{b})$ mit $\delta_2 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	b	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h

  Löscht ein Wort $w \in \mathtt{1}^*$:

  $h_{\tau_2}(w) = \epsilon$ für alle w, Definitionsbereich $\{\mathtt{1}\}^*$, Wertebereich $\{\epsilon\}$

- $\boldsymbol{\tau_3} = (\{\mathtt{s}_0,\mathtt{s}_1\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_3, \mathtt{s}_0, \mathtt{b})$ mit $\delta_3 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_1$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	1	h
$\mathtt{s}_1$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_1$	b	$\mathtt{s}_1$	b	r

Beispiele für Turing-Maschinen II

- $\tau_2 = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_2, \mathtt{s}_0, \mathtt{b})$ mit $\delta_2 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	b	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h

  Löscht ein Wort $w \in \mathtt{1}^*$:

  $h_{\tau_2}(w) = \epsilon$ für alle w, Definitionsbereich $\{\mathtt{1}\}^*$, Wertebereich $\{\epsilon\}$

- $\tau_3 = (\{\mathtt{s}_0,\mathtt{s}_1\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_3, \mathtt{s}_0, \mathtt{b})$ mit $\delta_3 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_1$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	1	h
$\mathtt{s}_1$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_1$	b	$\mathtt{s}_1$	b	r

  Testet Anzahl der Einsen in $w \in \mathtt{1}^*$:

$$h_{\tau_3}(\mathtt{1}^n) = \begin{cases} \mathtt{1}^{n+1} & \text{falls } n \text{ gerade,} \\ \perp & \text{sonst} \end{cases}$$

Beispiele für Turing-Maschinen II

- $\boldsymbol{\tau_2} = (\{\mathtt{s}_0\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_2, \mathtt{s}_0, \mathtt{b})$ mit $\delta_2 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_0$	b	r
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h

  Löscht ein Wort $w \in \mathtt{1}^*$:

  $h_{\tau_2}(w) = \epsilon$ für alle w, Definitionsbereich $\{\mathtt{1}\}^*$, Wertebereich $\{\epsilon\}$

- $\boldsymbol{\tau_3} = (\{\mathtt{s}_0,\mathtt{s}_1\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1}\}, \delta_3, \mathtt{s}_0, \mathtt{b})$ mit $\delta_3 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_1$	1	r
$\mathtt{s}_0$	b	$\mathtt{s}_1$	1	h
$\mathtt{s}_1$	1	$\mathtt{s}_0$	1	r
$\mathtt{s}_1$	b	$\mathtt{s}_1$	b	r

  Testet Anzahl der Einsen in $w \in \mathtt{1}^*$:

  $$h_{\tau_3}(\mathtt{1}^n) = \begin{cases} \mathtt{1}^{n+1} & \text{falls } n \text{ gerade,} \\ \perp & \text{sonst} \end{cases}$$

  Definitionsbereich $\{\mathtt{1}^{2k} \mid k \in \mathbb{N}\}$, Wertebereich $\{\mathtt{1}^{2k+1} \mid k \in \mathbb{N}\}$

Beispiele für Turing-Maschinen III

- $\boldsymbol{\tau_4} = (\{\mathtt{s}_0, \mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_3\}, \{\mathtt{1}\}, \{\mathtt{b}, \mathtt{1}, \mathtt{c}\}, \delta_4, \mathtt{s}_0, \mathtt{b})$ mit $\delta_4 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_1$	b	r
$\mathtt{s}_0$	c	$\mathtt{s}_0$	c	h
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h
$\mathtt{s}_1$	1	$\mathtt{s}_1$	1	r
$\mathtt{s}_1$	c	$\mathtt{s}_1$	c	r
$\mathtt{s}_1$	b	$\mathtt{s}_2$	c	r
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	h
$\mathtt{s}_2$	c	$\mathtt{s}_2$	c	h
$\mathtt{s}_2$	b	$\mathtt{s}_3$	c	l
$\mathtt{s}_3$	1	$\mathtt{s}_3$	1	l
$\mathtt{s}_3$	c	$\mathtt{s}_3$	c	l
$\mathtt{s}_3$	b	$\mathtt{s}_0$	b	r

Beispiele für Turing-Maschinen III

- $\tau_4 = (\{\mathtt{s}_0, \mathtt{s}_1, \mathtt{s}_2, \mathtt{s}_3\}, \{\mathtt{1}\}, \{\mathtt{b}, \mathtt{1}, \mathtt{c}\}, \delta_4, \mathtt{s}_0, \mathtt{b})$ mit $\delta_4 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_1$	b	r
$\mathtt{s}_0$	c	$\mathtt{s}_0$	c	h
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h
$\mathtt{s}_1$	1	$\mathtt{s}_1$	1	r
$\mathtt{s}_1$	c	$\mathtt{s}_1$	c	r
$\mathtt{s}_1$	b	$\mathtt{s}_2$	c	r
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	h
$\mathtt{s}_2$	c	$\mathtt{s}_2$	c	h
$\mathtt{s}_2$	b	$\mathtt{s}_3$	c	l
$\mathtt{s}_3$	1	$\mathtt{s}_3$	1	l
$\mathtt{s}_3$	c	$\mathtt{s}_3$	c	l
$\mathtt{s}_3$	b	$\mathtt{s}_0$	b	r

Verdoppelt Anzahl der Einsen

$h_{\tau_4}(\mathtt{1}^n) = \mathtt{c}^{2n}$, Definitionsbereich $\{\mathtt{1}\}^*$, Wertebereich $\{\mathtt{c}^{2k} \mid k \in \mathbb{N}\}$

Beispiele für Turing-Maschinen III

- $\tau_4 = (\{\mathtt{s}_0,\mathtt{s}_1,\mathtt{s}_2,\mathtt{s}_3\}, \{\mathtt{1}\}, \{\mathtt{b},\mathtt{1},\mathtt{c}\}, \delta_4, \mathtt{s}_0, \mathtt{b})$ mit $\delta_4 =$

s	a	s'	a'	P
$\mathtt{s}_0$	1	$\mathtt{s}_1$	b	r
$\mathtt{s}_0$	c	$\mathtt{s}_0$	c	h
$\mathtt{s}_0$	b	$\mathtt{s}_0$	b	h
$\mathtt{s}_1$	1	$\mathtt{s}_1$	1	r
$\mathtt{s}_1$	c	$\mathtt{s}_1$	c	r
$\mathtt{s}_1$	b	$\mathtt{s}_2$	c	r
$\mathtt{s}_2$	1	$\mathtt{s}_2$	1	h
$\mathtt{s}_2$	c	$\mathtt{s}_2$	c	h
$\mathtt{s}_2$	b	$\mathtt{s}_3$	c	l
$\mathtt{s}_3$	1	$\mathtt{s}_3$	1	l
$\mathtt{s}_3$	c	$\mathtt{s}_3$	c	l
$\mathtt{s}_3$	b	$\mathtt{s}_0$	b	r

Verdoppelt Anzahl der Einsen

$h_{\tau_4}(\mathtt{1}^n) = \mathtt{c}^{2n}$, Definitionsbereich $\{\mathtt{1}\}^*$, Wertebereich $\{\mathtt{c}^{2k} \mid k \in \mathbb{N}\}$

Kombinierbar mit isomorpher Variante von τ_3: $h_{\tau_3'} \circ h_{\tau_4}(\mathtt{1}^n) = \mathtt{c}^{2n+1}$

Turing-Berechenbarkeit

Definition E

- **$f{:}X^*{\to}Y^*$ Turing-berechenbar**
  - Es gibt eine Turingmaschine $\tau = (S, X, \Gamma, \delta, s_0, b)$ mit $Y{\subseteq}\Gamma$ und $h_\tau{=}f$

- **$\mathcal{T}$: Menge der Turing-berechenbaren Funktionen**
  - $\mathcal{T}_{X,Y} = \{f{:}X^*{\to}Y^* \mid f \text{ ist Turing-berechenbar}\}$
  - $\mathcal{T} \quad = \bigcup\{\mathcal{T}_{X,Y} \mid X, Y \text{ endliches Alphabet}\}$

Übertragung des Berechenbarkeitbegriffs

- **Berechenbarkeit von Mengen** $M \subseteq X^*$
  - Semi-Entscheidbarkeit: Berechenbarkeit von $\psi_M : X^* \to \{0,1\}^*$,
  - Entscheidbarkeit: Berechenbarkeit von $\chi_M : X^* \to \{0,1\}^*$,

  wobei $\psi_M(w) = \begin{cases} 1 & \text{falls } w \in M, \\ \perp & \text{sonst} \end{cases}$ $\quad \chi_M(w) = \begin{cases} 1 & \text{falls } w \in M, \\ 0 & \text{sonst} \end{cases}$

  (partiell-)charakteristische Funktion

  Definition P

Übertragung des Berechenbarkeitbegriffs

- **Berechenbarkeit von Mengen** $M \subseteq X^*$
  - Semi-Entscheidbarkeit: Berechenbarkeit von $\psi_M : X^* \to \{0,1\}^*$,
  - Entscheidbarkeit: Berechenbarkeit von $\chi_M : X^* \to \{0,1\}^*$,

  wobei $\psi_M(w) = \begin{cases} 1 & \text{falls } w \in M, \\ \perp & \text{sonst} \end{cases}$ $\quad \chi_M(w) = \begin{cases} 1 & \text{falls } w \in M, \\ 0 & \text{sonst} \end{cases}$

  (partiell-)charakteristische Funktion — Definition P

- **Berechenbarkeit auf Zahlen** $f : \mathbb{N} \to \mathbb{N}$

  $\equiv$ Berechenbarkeit der Repräsentation $f_r : X^* \to X^*$,
  wobei $r : \mathbb{N} \to X^*$ bijektiv und $f_r(w) = r(f(\mathrm{r}^{-1}(w)))$

  Standardcodierungen von Zahlen
  - unäre Darstellung $r_u : \mathbb{N} \to \{\mathtt{1}\}^*$ mit $r_u(n) = \mathtt{1}^n$
  - binäre Darstellung $r_b : \mathbb{N} \to \{\mathtt{0},\mathtt{1}\}^*$ (ohne führende Nullen)

Übertragung des Berechenbarkeitbegriffs

- **Berechenbarkeit von Mengen** $M \subseteq X^*$
  - Semi-Entscheidbarkeit: Berechenbarkeit von $\psi_M : X^* \to \{0,1\}^*$,
  - Entscheidbarkeit: Berechenbarkeit von $\chi_M : X^* \to \{0,1\}^*$,

  wobei $\boldsymbol{\psi_M(w)} = \begin{cases} 1 & \text{falls } w \in M, \\ \perp & \text{sonst} \end{cases}$ $\boldsymbol{\chi_M(w)} = \begin{cases} 1 & \text{falls } w \in M, \\ 0 & \text{sonst} \end{cases}$

  (partiell-)charakteristische Funktion

  Definition P

- **Berechenbarkeit auf Zahlen** $f : \mathbb{N} \to \mathbb{N}$

  $\equiv$ Berechenbarkeit der Repräsentation $f_r : X^* \to X^*$,
  wobei $r : \mathbb{N} \to X^*$ bijektiv und $f_r(w) = r(f(\mathrm{r}^{-1}(w)))$

  Standardcodierungen von Zahlen
  - unäre Darstellung $r_u : \mathbb{N} \to \{\mathtt{1}\}^*$ mit $r_u(n) = \mathtt{1}^n$
  - binäre Darstellung $r_b : \mathbb{N} \to \{\mathtt{0},\mathtt{1}\}^*$ (ohne führende Nullen)

- **Berechenbarkeit auf Tupeln** $f : X^* \times X^* \to Y^*$

  $\equiv$ Berechenbarkeit von $f' : (X \cup \{\#\})^* \to Y^*$ mit $f'(v\#w) = f(v,w)$

Berechenbarkeit der Nachfolgerfunktion

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

Berechenbarkeit der Nachfolgerfunktion

Ist s:$\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist s_u:$\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?

Berechenbarkeit der Nachfolgerfunktion

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine `1` anhängen: $s_u = h_{\tau_1}$

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n{+}1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine $\mathtt{1}$ anhängen: $s_u = h_{\tau_1}$

- Bei binärer Codierung
  - Ist $s_b{:}\{\mathtt{0,1}\}^*\to\{\mathtt{0,1}\}^*$ mit $s_b(r_b(n)) = r_b(n{+}1)$ Turing-berechenbar?

Berechenbarkeit der Nachfolgerfunktion

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine `1` anhängen: $s_u = h_{\tau_1}$

- Bei binärer Codierung
  - Ist $s_b{:}\{\mathtt{0,1}\}^*\to\{\mathtt{0,1}\}^*$ mit $s_b(r_b(n)) = r_b(n+1)$ Turing-berechenbar?
  - τ_s muß Ziffern von rechts beginnend umwandeln, ggf. mit Übertrag

BERECHENBARKEIT DER NACHFOLGERFUNKTION

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine $\mathtt{1}$ anhängen: $s_u = h_{\tau_1}$
- Bei binärer Codierung
  - Ist $s_b{:}\{\mathtt{0},\mathtt{1}\}^*\to\{\mathtt{0},\mathtt{1}\}^*$ mit $s_b(r_b(n)) = r_b(n+1)$ Turing-berechenbar?
  - τ_s muß Ziffern von rechts beginnend umwandeln, ggf. mit Übertrag

s	a	s'	a'	P	
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r	*rechtes Ende suchen*
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r	*rechtes Ende suchen*
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l	*rechtes Ende gefunden*

Berechenbarkeit der Nachfolgerfunktion

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine 1 anhängen: $s_u = h_{\tau_1}$

- Bei binärer Codierung
  - Ist $s_b{:}\{\mathtt{0,1}\}^*\to\{\mathtt{0,1}\}^*$ mit $s_b(r_b(n)) = r_b(n+1)$ Turing-berechenbar?
  - τ_s muß Ziffern von rechts beginnend umwandeln, ggf. mit Übertrag

s	a	s'	a'	P	
s_0	0	s_0	0	r	*rechtes Ende suchen*
s_0	1	s_0	1	r	*rechtes Ende suchen*
s_0	b	s_1	b	l	*rechtes Ende gefunden*
s_1	1	s_1	0	l	*Addieren mit Übertrag*

Berechenbarkeit der Nachfolgerfunktion

Ist $s{:}\mathbb{N}\rightarrow\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\rightarrow\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine 1 anhängen: $s_u = h_{\tau_1}$

- Bei binärer Codierung
  - Ist $s_b{:}\{\mathtt{0,1}\}^*\rightarrow\{\mathtt{0,1}\}^*$ mit $s_b(r_b(n)) = r_b(n+1)$ Turing-berechenbar?
  - τ_s muß Ziffern von rechts beginnend umwandeln, ggf. mit Übertrag

s	a	s'	a'	P	
s_0	0	s_0	0	r	*rechtes Ende suchen*
s_0	1	s_0	1	r	*rechtes Ende suchen*
s_0	b	s_1	b	l	*rechtes Ende gefunden*
s_1	1	s_1	0	l	*Addieren mit Übertrag*
s_1	0	s_2	1	h	*Addieren ohne Übertrag*

Berechenbarkeit der Nachfolgerfunktion

Ist $s{:}\mathbb{N}\to\mathbb{N}$ mit $s(n) = n+1$ Turing-berechenbar?

- Bei unärer Codierung:
  - Ist $s_u{:}\{\mathtt{1}\}^*\to\{\mathtt{1}\}^*$ mit $s_u(\mathtt{1}^n) = \mathtt{1}^{n+1}$ Turing-berechenbar?
  - Turingmaschine muß eine $\mathtt{1}$ anhängen: $s_u = h_{\tau_1}$

- Bei binärer Codierung
  - Ist $s_b{:}\{\mathtt{0},\mathtt{1}\}^*\to\{\mathtt{0},\mathtt{1}\}^*$ mit $s_b(r_b(n)) = r_b(n+1)$ Turing-berechenbar?
  - τ_s muß Ziffern von rechts beginnend umwandeln, ggf. mit Übertrag

s	a	s'	a'	P	
$\mathtt{s}_0$	0	$\mathtt{s}_0$	0	r	*rechtes Ende suchen*
$\mathtt{s}_0$	1	$\mathtt{s}_0$	1	r	*rechtes Ende suchen*
$\mathtt{s}_0$	b	$\mathtt{s}_1$	b	l	*rechtes Ende gefunden*
$\mathtt{s}_1$	1	$\mathtt{s}_1$	0	l	*Addieren mit Übertrag*
$\mathtt{s}_1$	0	$\mathtt{s}_2$	1	h	*Addieren ohne Übertrag*
$\mathtt{s}_1$	b	$\mathtt{s}_2$	1	h	*Übertrag am linken Ende*

Berechenbarkeit der Division durch 2

Ist $div_2{:}\mathbb{N}\longrightarrow\mathbb{N}$ mit $div_2(n)=\lfloor n/2\rfloor$ Turing-berechenbar?

Berechenbarkeit der Division durch 2

Ist div_2:$\mathbb{N}\rightarrow\mathbb{N}$ mit $div_2(n)=\lfloor n/2 \rfloor$ Turing-berechenbar?

- Bei unärer Codierung muß τ je zwei Einsen löschen
  und eine neue hinter dem Ende des Wortes schreiben

Berechenbarkeit der Division durch 2

Ist div_2:$\mathbb{N}\longrightarrow\mathbb{N}$ mit $div_2(n)=\lfloor n/2 \rfloor$ Turing-berechenbar?

- Bei unärer Codierung muß τ je zwei Einsen löschen und eine neue hinter dem Ende des Wortes schreiben

s	a	s'	a'	P	
s_0	1	s_1	b	r	*Erste* 1
s_0	b	s_6	b	h	*keine erste* 1
s_1	1	s_2	b	r	*Zweite* 1
s_1	b	s_6	b	h	*keine zweite* 1
s_2	1	s_2	1	r	*Nach rechts zum Eingabeende*
s_2	b	s_3	b	r	*Ende der Eingabe*
s_3	1	s_3	1	r	*Nach rechts zum Ausgabeende*
s_3	b	s_4	1	l	*Ende der Ausgabe, 1 schreiben*
s_4	1	s_4	1	l	*Nach links über Ausgabe*
s_4	b	s_5	b	l	
s_5	1	s_5	1	l	*Nach links über Eingabe*
s_5	b	s_0	b	r	

Berechenbarkeit der Division durch 2

Ist div_2:ℕ→ℕ mit $div_2(n) = \lfloor n/2 \rfloor$ Turing-berechenbar?

- Bei unärer Codierung muß τ je zwei Einsen löschen und eine neue hinter dem Ende des Wortes schreiben

s	a	s'	a'	P	
s_0	1	s_1	b	r	*Erste* 1
s_0	b	s_6	b	h	*keine erste* 1
s_1	1	s_2	b	r	*Zweite* 1
s_1	b	s_6	b	h	*keine zweite* 1
s_2	1	s_2	1	r	*Nach rechts zum Eingabeende*
s_2	b	s_3	b	r	*Ende der Eingabe*
s_3	1	s_3	1	r	*Nach rechts zum Ausgabeende*
s_3	b	s_4	1	l	*Ende der Ausgabe, 1 schreiben*
s_4	1	s_4	1	l	*Nach links über Ausgabe*
s_4	b	s_5	b	l	
s_5	1	s_5	1	l	*Nach links über Eingabe*
s_5	b	s_0	b	r	

- Bei binärer Codierung muß τ die letzte Ziffer löschen

Varianten von Turingmaschinen

- **Vereinfachung für theoretische Analysen**
  - Binäres Bandalphabet $\Gamma = \{\mathtt{1}, \mathtt{b}\}$
  - Halbseitig unendliches Band
  - Restriktivere Ausgabekonvention
  - Endzustand statt Halteinstruktion

- **Erweiterung des Modells für Programmierzwecke**
  - Unvollständige Tabellen für δ
  - Mehrspurmaschinen
  - Mehrkopfmaschinen
  - Mehrbandmaschinen
  - Mehrdimensionale Maschinen
  - Unterprogramme

Alle Varianten führen zum gleichen Berechenbarkeitsbegriff

Einfachere Turingmaschinenmodelle

Kein Verlust der Ausdruckskraft

Simulation normaler Turingmaschinen möglich

Einfachere Turingmaschinenmodelle

Kein Verlust der Ausdruckskraft

Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**

Kein Verlust der Ausdruckskraft

Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  - Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)

    a_l repräsentiert die linke, a_r die rechte Bandhälfte

Kein Verlust der Ausdruckskraft

Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  - Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)
    a_l repräsentiert die linke, a_r die rechte Bandhälfte
- **Binäres Bandalphabet** $\Gamma = \{\mathtt{1}, \mathtt{b}\}$

Kein Verlust der Ausdruckskraft
Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  – Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)
  a_l repräsentiert die linke, a_r die rechte Bandhälfte

- **Binäres Bandalphabet** $\Gamma = \{\texttt{1}, \texttt{b}\}$
  – Binärcodierung beliebiger Alphabete als Strings über $\{\texttt{1b},\texttt{11}\}$

Kein Verlust der Ausdruckskraft
Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  – Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)
  a_l repräsentiert die linke, a_r die rechte Bandhälfte

- **Binäres Bandalphabet** $\Gamma = \{\mathtt{1}, \mathtt{b}\}$
  – Binärcodierung beliebiger Alphabete als Strings über $\{\mathtt{1b}, \mathtt{11}\}$

- **Ausgabewort muß unter dem Kopf beginnen**
  – Ausgabefunktion ist Bandinhalt vom Kopfsymbol bis zum ersten Blank.

Einfachere Turingmaschinenmodelle

Kein Verlust der Ausdruckskraft
Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  – Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)
  a_l repräsentiert die linke, a_r die rechte Bandhälfte
- **Binäres Bandalphabet** $\Gamma = \{\mathtt{1}, \mathtt{b}\}$
  – Binärcodierung beliebiger Alphabete als Strings über $\{\mathtt{1b}, \mathtt{11}\}$
- **Ausgabewort muß unter dem Kopf beginnen**
  – Ausgabefunktion ist Bandinhalt vom Kopfsymbol bis zum ersten Blank.
  – Ergänze Programm für δ um Kopfbewegung zum Wortanfang.

Einfachere Turingmaschinenmodelle

Kein Verlust der Ausdruckskraft
Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  - Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)
    a_l repräsentiert die linke, a_r die rechte Bandhälfte
- **Binäres Bandalphabet** $\Gamma = \{\mathtt{1}, \mathtt{b}\}$
  - Binärcodierung beliebiger Alphabete als Strings über $\{\mathtt{1b},\mathtt{11}\}$
- **Ausgabewort muß unter dem Kopf beginnen**
  - Ausgabefunktion ist Bandinhalt vom Kopfsymbol bis zum ersten Blank.
  - Ergänze Programm für δ um Kopfbewegung zum Wortanfang.
- **Fester Endzustand s_e statt Halteinstruktion**

Kein Verlust der Ausdruckskraft
Simulation normaler Turingmaschinen möglich

- **Halbseitig unendliches Band**
  - Simulation eines beidseitig unendlichen Bands durch Tupelalphabet (a_l, a_r)
    a_l repräsentiert die linke, a_r die rechte Bandhälfte
- **Binäres Bandalphabet** $\Gamma = \{\texttt{1}, \texttt{b}\}$
  - Binärcodierung beliebiger Alphabete als Strings über $\{\texttt{1b}, \texttt{11}\}$
- **Ausgabewort muß unter dem Kopf beginnen**
  - Ausgabefunktion ist Bandinhalt vom Kopfsymbol bis zum ersten Blank.
  - Ergänze Programm für δ um Kopfbewegung zum Wortanfang.
- **Fester Endzustand s_e statt Halteinstruktion**
  - Ändere $\delta(s,a) = (s',a',h)$ in $\delta(s,a) = (s_e,a',l)$

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  - Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
    a_i repräsentiert Spur i

Keine Erweiterung der Ausdruckskraft
Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  - Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
    a_i repräsentiert Spur i
- **Mehrbandmaschinen**

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  - Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
    a_i repräsentiert Spur i
- **Mehrbandmaschinen**
  - Simulation durch Mehrspurmaschine und Marker für Kopfpositionen

Keine Erweiterung der Ausdruckskraft

Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  - Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
    a_i repräsentiert Spur i
- **Mehrbandmaschinen**
  - Simulation durch Mehrspurmaschine und Marker für Kopfpositionen
- **Mehrkopfmaschinen**

Keine Erweiterung der Ausdruckskraft
Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  - Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  - Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
    a_i repräsentiert Spur i
- **Mehrbandmaschinen**
  - Simulation durch Mehrspurmaschine und Marker für Kopfpositionen
- **Mehrkopfmaschinen**
  - Speichere Kopfpositionen auf separatem Band and verarbeite sequentiell

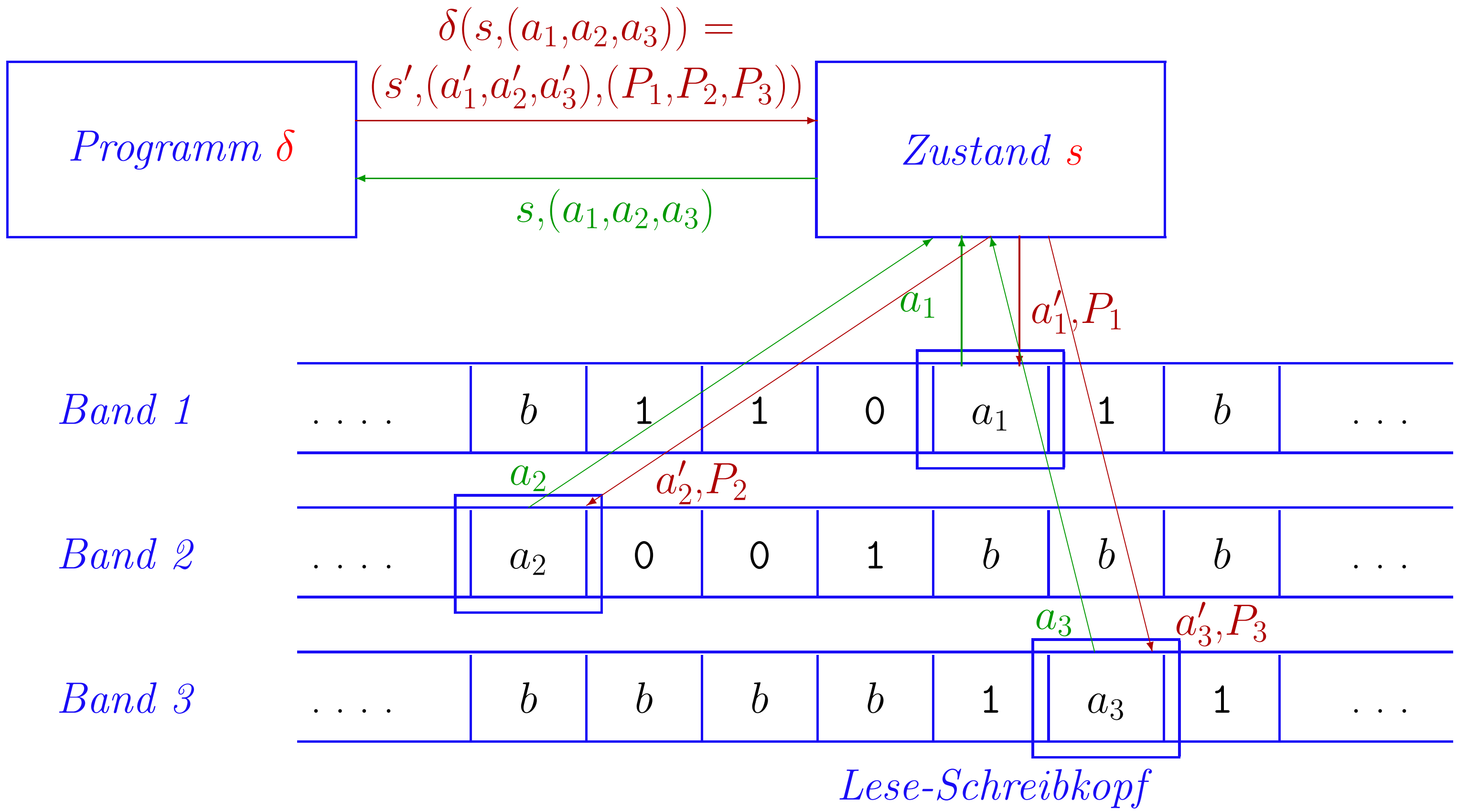
Keine Erweiterung der Ausdruckskraft
Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  – Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  – Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
  a_i repräsentiert Spur i
- **Mehrbandmaschinen**
  – Simulation durch Mehrspurmaschine und Marker für Kopfpositionen
- **Mehrkopfmaschinen**
  – Speichere Kopfpositionen auf separatem Band and verarbeite sequentiell
- **Unterprogramme**

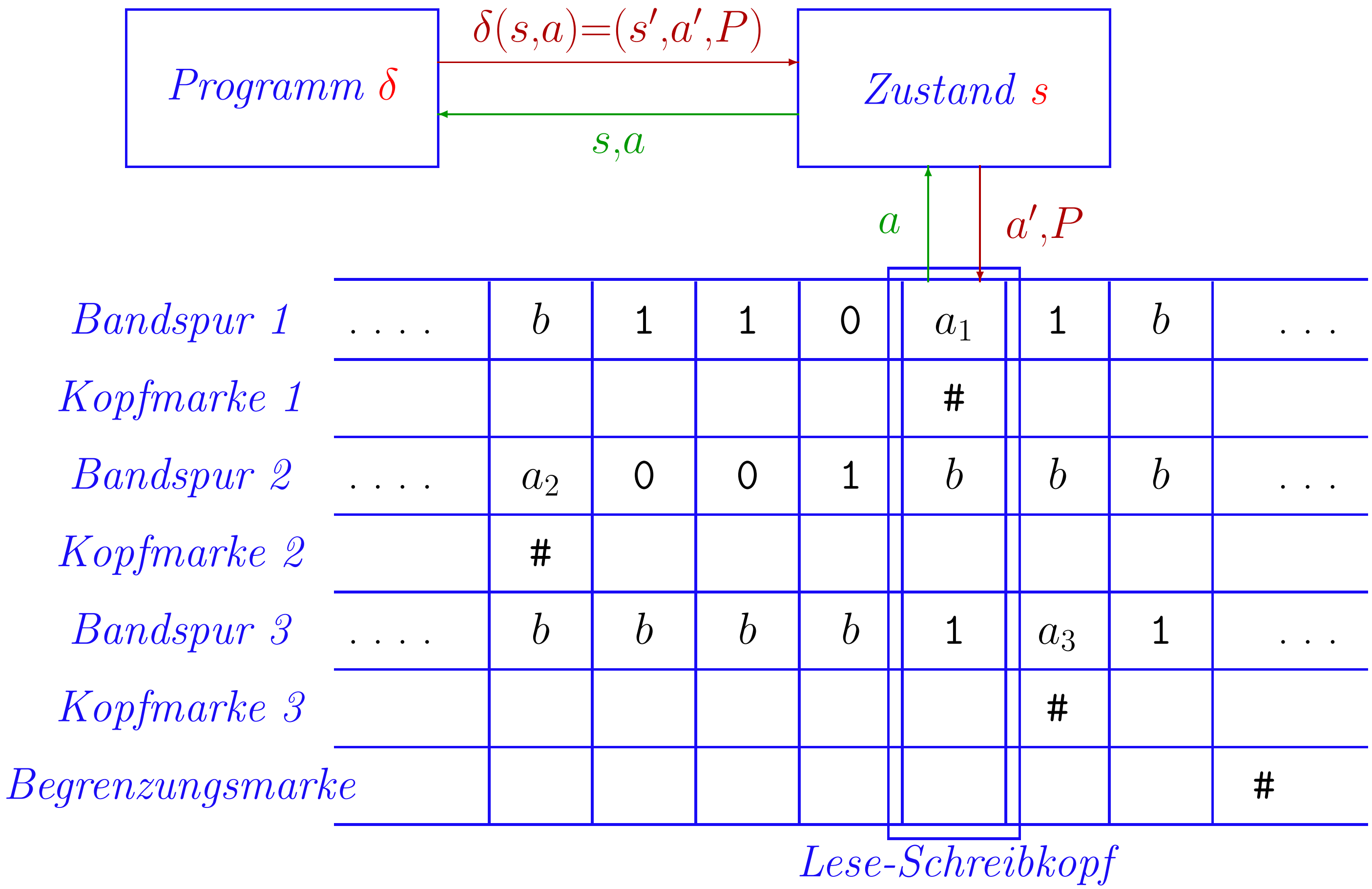
Keine Erweiterung der Ausdruckskraft
Simulation durch normale Turingmaschinen möglich

- **Unvollständige Tabellen für δ**
  – Ergänze nichtgenannte Einträge als $\delta(s,a) = (s,a,h)$
- **Mehrspurmaschinen**
  – Simulation von k Spuren durch Tupelalphabet $(a_1, ..., a_k)$
  a_i repräsentiert Spur i
- **Mehrbandmaschinen**
  – Simulation durch Mehrspurmaschine und Marker für Kopfpositionen
- **Mehrkopfmaschinen**
  – Speichere Kopfpositionen auf separatem Band and verarbeite sequentiell
- **Unterprogramme**
  – Speichere Argumente und Rückgabewerte auf separatem Band.

Mehrbandmaschine



Simulation einer Mehrbandmaschine



- **Verarbeite Bänder sequentiell**
  - Lesen: Suche Begrenzungsmarke, laufe zurück bis zu Kopfmarken
  - Schreiben und Kopfbewegung analog
  - Codiere Symbole und Kopfinstruktionen im Zustand
  - Simulation benötigt quadratische Zeit