Computational Logic

Einführung in die Theoretische Informatik

Woche 10

Harald Zankl

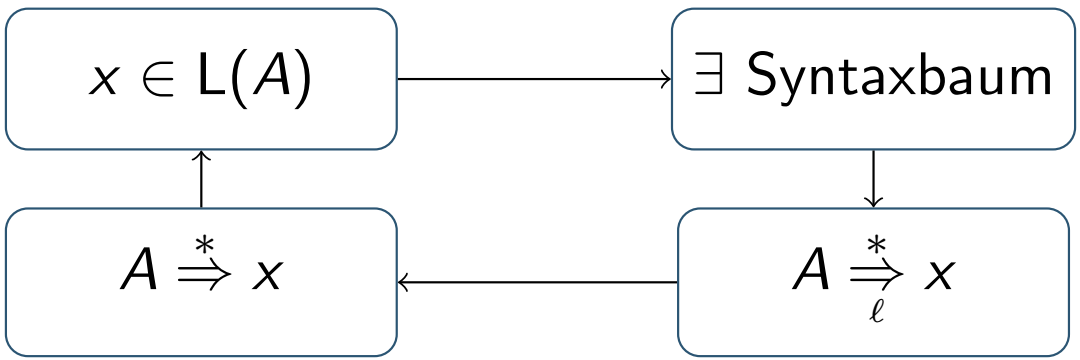
Institut für Informatik @ UIBK
Wintersemester 2014/2015

Zusammenfassung der letzten LV

Satz

Sei $G = (V, \Sigma, R, S)$ eine kontextfreie Grammatik, $A \in V$ und $x \in \Sigma^*$. Die folgenden Aussagen sind äquivalent:

1. $x \in \mathrm{L}(A)$ *nach dem rekursiven Inferenzverfahren*
2. $A \overset{*}{\Rightarrow} x$
3. $A \overset{*}{\underset{\ell}{\Rightarrow}} x$
4. $A \overset{*}{\underset{r}{\Rightarrow}} x$
5. *Es existiert ein Syntaxbaum mit Wurzel A und Ergebnis x*



Inhalte der Lehrveranstaltung

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, Konjunktive und Disjunktive Normalformen

Einführung in die Algebra

Boolesche Algebra, Universelle Algebra, Logische Schaltkreise

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Reguläre Sprachen, Kontextfreie Sprachen

Einführung in die Berechenbarkeitstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare, Verschlüsselung und Sicherheit

Anwendungen von kontextfreien Sprachen

- Parsergeneratoren, beziehungsweise im Compilerbau
  1. Parsergenerator verwandelt die Beschreibung einer Sprache in einen Parser für diese Sprache
  2. Parser werden zur syntaktischen Analyse von Programmen verwendet

Anwendungen von kontextfreien Sprachen

- Parsergeneratoren, beziehungsweise im Compilerbau
  1. Parsergenerator verwandelt die Beschreibung einer Sprache in einen Parser für diese Sprache
  2. Parser werden zur syntaktischen Analyse von Programmen verwendet
- Anwendungen im Bereich der Wissensrepräsentation, etwa in XML-Dokumenten

Anwendungen von kontextfreien Sprachen

- Parsergeneratoren, beziehungsweise im Compilerbau
  1. Parsergenerator verwandelt die Beschreibung einer Sprache in einen Parser für diese Sprache
  2. Parser werden zur syntaktischen Analyse von Programmen verwendet
- Anwendungen im Bereich der Wissensrepräsentation, etwa in XML-Dokumenten

Definition

- XML steht für eXtensible Markup Language
- XML ist eine Markup-Sprache bzw. Kennzeichnungssprache wie HTML

Anwendungen von kontextfreien Sprachen

- Parsergeneratoren, beziehungsweise im Compilerbau
  1. Parsergenerator verwandelt die Beschreibung einer Sprache in einen Parser für diese Sprache
  2. Parser werden zur syntaktischen Analyse von Programmen verwendet
- Anwendungen im Bereich der Wissensrepräsentation, etwa in XML-Dokumenten

Definition

- XML steht für eXtensible Markup Language
- XML ist eine Markup-Sprache bzw. Kennzeichnungssprache wie HTML
- XML steht nicht im Gegensatz zu HTML, sondern soll diese erweitern
- XML beschreibt den Inhalt eines Textes

Definition

- Die *tags* in XML Dokumenten werden vom Benutzer definiert:

  <Element> ``Inhalt'' </Element>

  Solche Tags heißen Namenselemente

Definition

- Die *tags* in XML Dokumenten werden vom Benutzer definiert:

  ```
  <Element> ``Inhalt'' </Element>
  ```

  Solche Tags heißen Namenselemente

- Um den Inhalt zu beschreiben werden
  1. *document type definitions* (DTDs)

     ```
     <!DOCTYPE ``Name der DTD'' [
      ``Liste der Elementbeschreibungen'']>
     ```

Definition

- Die *tags* in XML Dokumenten werden vom Benutzer definiert:

```
<Element> ``Inhalt'' </Element>
```

  Solche Tags heißen Namenselemente

- Um den Inhalt zu beschreiben werden
  1. *document type definitions* (DTDs)

```
<!DOCTYPE ``Name der DTD'' [
``Liste der Elementbeschreibungen'']>
```

  2. oder XML Schema verwendet

Definition

- Die *tags* in XML Dokumenten werden vom Benutzer definiert:

```
<Element> ``Inhalt'' </Element>
```

  Solche Tags heißen Namenselemente

- Um den Inhalt zu beschreiben werden
  1. *document type definitions* (DTDs)

```
<!DOCTYPE ``Name der DTD'' [
``Liste der Elementbeschreibungen'']>
```

  2. oder XML Schema verwendet

Definition

- Die *tags* in XML Dokumenten werden vom Benutzer definiert:

  ```
  <Element> ``Inhalt'' </Element>
  ```

  Solche Tags heißen Namenselemente

- Um den Inhalt zu beschreiben werden
  1. *document type definitions* (DTDs)

     ```
     <!DOCTYPE ``Name der DTD'' [
     ``Liste der Elementbeschreibungen'']>
     ```

  2. oder XML Schema verwendet
- Die Beschreibung der Elemente hat die Form

  ```
  <!ELEMENT ``Name'' (``Elementbeschreibung'')>
  ```

Definition

- Die *tags* in XML Dokumenten werden vom Benutzer definiert:

  ```
  <Element> ``Inhalt'' </Element>
  ```

  Solche Tags heißen Namenselemente
- Um den Inhalt zu beschreiben werden
  1. *document type definitions* (DTDs)

     ```
     <!DOCTYPE ``Name der DTD'' [
     ``Liste der Elementbeschreibungen'']>
     ```

  2. oder XML Schema verwendet
- Die Beschreibung der Elemente hat die Form

  ```
  <!ELEMENT ``Name'' (``Elementbeschreibung'')>
  ```

- Elementbeschreibungen definieren reguläre Sprachen
  dazu verwendet man (erweiterte) reguläre Ausdrücke

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

1. Ein Namenselement ist ein regulärer Ausdruck

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

1. Ein Namenselement ist ein regulärer Ausdruck
2. `#PCDATA` (bezeichnet jedes Wort ohne XML-Tags) ist ein regulärer Ausdruck

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

1. Ein Namenselement ist ein regulärer Ausdruck
2. `#PCDATA` (bezeichnet jedes Wort ohne XML-Tags) ist ein regulärer Ausdruck
3. $E \mid F$ bezeichnet Vereinigung der durch E und F beschriebenen Elemente

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

1. Ein Namenselement ist ein regulärer Ausdruck
2. `#PCDATA` (bezeichnet jedes Wort ohne XML-Tags) ist ein regulärer Ausdruck
3. $E \mid F$ bezeichnet Vereinigung der durch E und F beschriebenen Elemente
4. E, F bezeichnet Konkatenation der durch E und F beschriebenen Elemente

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

1. Ein Namenselement ist ein regulärer Ausdruck
2. `#PCDATA` (bezeichnet jedes Wort ohne XML-Tags) ist ein regulärer Ausdruck
3. $E \mid F$ bezeichnet Vereinigung der durch E und F beschriebenen Elemente
4. E, F bezeichnet Konkatenation der durch E und F beschriebenen Elemente
5. $E\star$ ($E+$) steht für die beliebige (beliebige, aber mindestens einmalige) Wiederholung der durch E beschriebenen Elemente

Elementbeschreibungen

Definition (informell)

Wir definieren (erweiterte) reguläre Ausdrücke induktiv:

1. Ein Namenselement ist ein regulärer Ausdruck
2. `#PCDATA` (bezeichnet jedes Wort ohne XML-Tags) ist ein regulärer Ausdruck
3. $E \mid F$ bezeichnet Vereinigung der durch E und F beschriebenen Elemente
4. E, F bezeichnet Konkatenation der durch E und F beschriebenen Elemente
5. $E\star$ ($E+$) steht für die beliebige (beliebige, aber mindestens einmalige) Wiederholung der durch E beschriebenen Elemente
6. $E?$ steht für die Option E anzugeben

Beispiel

Wir betrachten die folgende DTD:

```
<!DOCTYPE NEWSPAPER [
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>
]>
```

Beispiel

Wir betrachten die folgende DTD:

```
<!DOCTYPE NEWSPAPER [
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>
]>
```

- Name der DTD ist `NEWSPAPER`; `NEWSPAPER` ist eine nichtleere Sequenz von Artikeln

Beispiel

Wir betrachten die folgende DTD:

```
<!DOCTYPE NEWSPAPER [
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>
]>
```

- Name der DTD ist `NEWSPAPER`; `NEWSPAPER` ist eine nichtleere Sequenz von Artikeln
- `ARTICLE` ist die Verknüpfung der folgenden Textelemente:

`HEADLINE`	die Kopfzeile	`BODY`	der eigentliche Artikel
`BYLINE`	der Untertitel	`NOTES`	Anmerkungen
`LEAD`	die Einleitung		

Umwandlung einer DTD in eine Grammatik

Umwandlung einer DTD in eine Grammatik

Beispiel (Fortsetzung)

Wir betrachten

<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>

Umwandlung einer DTD in eine Grammatik

Beispiel (Fortsetzung)

Wir betrachten

<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>

das entspricht der Regel

$$\texttt{ARTICLE} \rightarrow \texttt{HEADLINE BYLINE LEAD BODY NOTES}$$

Umwandlung einer DTD in eine Grammatik

Beispiel (Fortsetzung)

Wir betrachten

<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>

das entspricht der Regel

$$\texttt{ARTICLE} \rightarrow \texttt{HEADLINE BYLINE LEAD BODY NOTES}$$

Nun betrachten wir

<!ELEMENT NEWSPAPER (ARTICLE+)>

Umwandlung einer DTD in eine Grammatik

Beispiel (Fortsetzung)

Wir betrachten

<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>

das entspricht der Regel

$$\texttt{ARTICLE} \rightarrow \texttt{HEADLINE BYLINE LEAD BODY NOTES}$$

Nun betrachten wir

<!ELEMENT NEWSPAPER (ARTICLE+)>

und wir erhalten die folgenden Regeln

$$\texttt{NEWSPAPER} \rightarrow \texttt{ARTICLES}$$

$$\texttt{ARTICLES} \rightarrow \texttt{ARTICLE} \mid \texttt{ARTICLE ARTICLES}$$

Einführung in die Berechenbarkeitstheorie

Frage

Ist jedes Problem algorithmisch lösbar?

Einführung in die Berechenbarkeitstheorie

Frage

Ist jedes Problem algorithmisch lösbar?

Antwort

Nein

Einführung in die Berechenbarkeitstheorie

Frage

Ist jedes Problem algorithmisch lösbar?

Antwort

Nein

Frage

Was meinen wir eigentlich mit Problem?

$\Rightarrow$ eine mit Ja oder Nein beantwortbare Frage

Einführung in die Berechenbarkeitstheorie

Frage

Ist jedes Problem algorithmisch lösbar?

Antwort

Nein

Frage

Was meinen wir eigentlich mit Problem?
$\Rightarrow$ eine mit Ja oder Nein beantwortbare Frage, die prinzipiell algorithmisch lösbar ist

Einführung in die Berechenbarkeitstheorie

Frage

Ist jedes Problem algorithmisch lösbar?

Antwort

Nein

Frage

Was meinen wir eigentlich mit Problem?
$\Rightarrow$ eine mit Ja oder Nein beantwortbare Frage, die prinzipiell algorithmisch lösbar ist
z.B. Gibt Programm P den String "hello, world" aus?

Einführung in die Berechenbarkeitstheorie

Frage

Ist jedes Problem algorithmisch lösbar?

Antwort

Nein

Frage

Was meinen wir eigentlich mit Problem?
$\Rightarrow$ eine mit Ja oder Nein beantwortbare Frage, die prinzipiell algorithmisch lösbar ist
z.B. Gibt Programm P den String "hello, world" aus?

Programm P

```
int main(int argc, char** argv) {
 printf("hello, world");
}
```

Beispiel (Programm F)

```
int main(int argc, char** argv) {
  int n, sum = 3, x, y, z;
  scanf("%d", &n);
  while (1) {
    for (x = 1; x <= sum - 2; x++)
    for (y = 1; y <= sum - x - 1; y++) {
      z = sum - x - y;
      if (pow(x,n) + pow(y,n) == pow(z,n))
      printf("hello, world");
    }
  sum++;
  }
}
```

Beispiel (Programm F)

```
int main(int argc, char** argv) {
  int n, sum = 3, x, y, z;
  scanf("%d", &n);
  while (1) {
    for (x = 1; x <= sum - 2; x++)
    for (y = 1; y <= sum - x - 1; y++) {
      z = sum - x - y;
      if (pow(x,n) + pow(y,n) == pow(z,n))
      printf("hello, world");
    }
  sum++;
  }
}
```

Programm F ist kein „hello, world“-Programm

Beispiel (Programm G)

```
int main(int argc, char** argv) {
  int x, y, test, sum = 4;
  while (1) {
    test = 0;
    for (x = 2; x <= sum; x++) {
      y = sum - x;
      if (is_prime(x) && is_prime(y))
        test = 1;
    }
    if (!test) printf("hello, world");
    sum = sum + 2;
  }
}
```

Beispiel (Programm G)

```
int main(int argc, char** argv) {
  int x, y, test, sum = 4;
  while (1) {
    test = 0;
    for (x = 2; x <= sum; x++) {
      y = sum - x;
      if (is_prime(x) && is_prime(y))
        test = 1;
    }
    if (!test) printf("hello, world");
    sum = sum + 2;
  }
}
```

Program G ist wahrscheinlich kein „hello, world“-Programm

Beispiel (Programm G)

```
int main(int argc, char** argv) {
  int x, y, test, sum = 4;
  while (1) {
    test = 0;
    for (x = 2; x <= sum; x++) {
      y = sum - x;
      if (is_prime(x) && is_prime(y))
        test = 1;
    }
    if (!test) printf("hello, world");
    sum = sum + 2;
  }
}
```

Goldbach'sche Vermutung: *Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen geschrieben werden*

Großer Fermat (16??)

Für $n > 2$ gibt es keine $a, b, c \in \mathbb{N}$, sodass $a^n + b^n = c^n$.

Großer Fermat (16??)

Für $n > 2$ gibt es keine $a, b, c \in \mathbb{N}$, sodass $a^n + b^n = c^n$.

Vermutung von Goldbach (1742)

Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen dargestellt werden.

Großer Fermat (16??)

Für $n > 2$ gibt es keine $a, b, c \in \mathbb{N}$, sodass $a^n + b^n = c^n$.

Vermutung von Goldbach (1742)

Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen dargestellt werden.

Satz

Es kann kein Testprogramm für „hello, world“-Programme geben.

Großer Fermat (16??)

Für $n > 2$ gibt es keine $a, b, c \in \mathbb{N}$, sodass $a^n + b^n = c^n$.

Vermutung von Goldbach (1742)

Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen dargestellt werden.

Satz

Es kann kein Testprogramm für „hello, world“-Programme geben.

Definition (informell)

Ist ein Problem nicht algorithmisch lösbar, nennen wir es unentscheidbar.

Satz

Die folgenden Probleme sind unentscheidbar:

Definition

Als Halteproblem bezeichnen wir das Problem, ob ein beliebiges Programm auf seiner Eingabe hält.

Satz

Die folgenden Probleme sind unentscheidbar:

- *das Halteproblem*

Definition

Als Halteproblem bezeichnen wir das Problem, ob ein beliebiges Programm auf seiner Eingabe hält.

Definition

Postsches Korrespondenzproblem: Gegeben zwei Listen (gleicher Länge) von Strings $x_1, x_2, \ldots, x_n$ und $y_1, y_2, \ldots, y_n$. Gesucht sind $m > 0$ und Indizes $i_1, i_2, \ldots, i_m$, sodass

$$x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$$

Satz

Die folgenden Probleme sind unentscheidbar:

- *das Halteproblem*
- *das Postsche Korrespondenzproblem*

Definition

Als Halteproblem bezeichnen wir das Problem, ob ein beliebiges Programm auf seiner Eingabe hält.

Definition

Postsches Korrespondenzproblem: Gegeben zwei Listen (gleicher Länge) von Strings $x_1, x_2, \ldots, x_n$ und $y_1, y_2, \ldots, y_n$. Gesucht sind $m > 0$ und Indizes $i_1, i_2, \ldots, i_m$, sodass

$$x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$$

Satz

Die folgenden Probleme sind unentscheidbar:

- *das Halteproblem*
- *das Postsche Korrespondenzproblem*
- *Eindeutigkeit einer beliebigen kontextfreien Grammatik*

Definition

Als Halteproblem bezeichnen wir das Problem, ob ein beliebiges Programm auf seiner Eingabe hält.

Definition

Postsches Korrespondenzproblem: Gegeben zwei Listen (gleicher Länge) von Strings $x_1, x_2, \ldots, x_n$ und $y_1, y_2, \ldots, y_n$. Gesucht sind $m > 0$ und Indizes $i_1, i_2, \ldots, i_m$, sodass

$$x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$$

Satz

Die folgenden Probleme sind unentscheidbar:

- *das Halteproblem*
- *das Postsche Korrespondenzproblem*
- *Eindeutigkeit einer beliebigen kontextfreien Grammatik*
- *das Wortproblem in der Gleichungslogik ($E \models s \approx t$)*

Definition

Als Halteproblem bezeichnen wir das Problem, ob ein beliebiges Programm auf seiner Eingabe hält.

Definition

Postsches Korrespondenzproblem: Gegeben zwei Listen (gleicher Länge) von Strings $x_1, x_2, \ldots, x_n$ und $y_1, y_2, \ldots, y_n$. Gesucht sind $m > 0$ und Indizes $i_1, i_2, \ldots, i_m$, sodass

$$x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$$

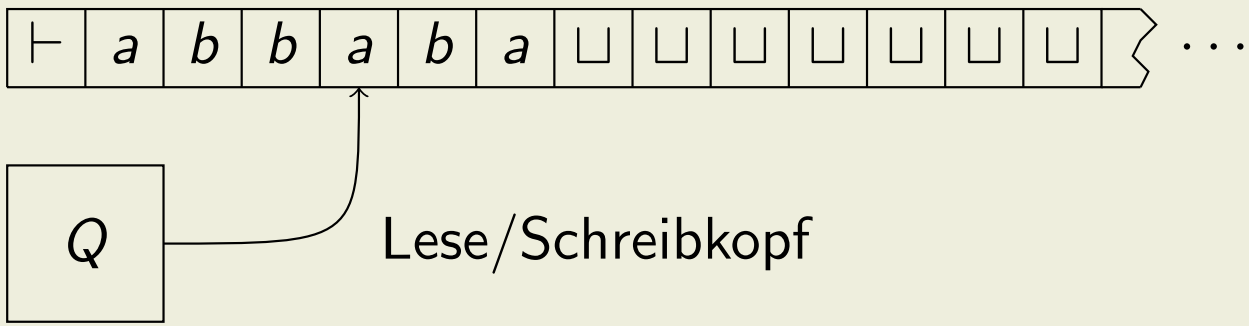
Satz

Die folgenden Probleme sind unentscheidbar:

- *das Halteproblem*
- *das Postsche Korrespondenzproblem*
- *Eindeutigkeit einer beliebigen kontextfreien Grammatik*
- *das Wortproblem in der Gleichungslogik ($E \models s \approx t$)*
- *...*

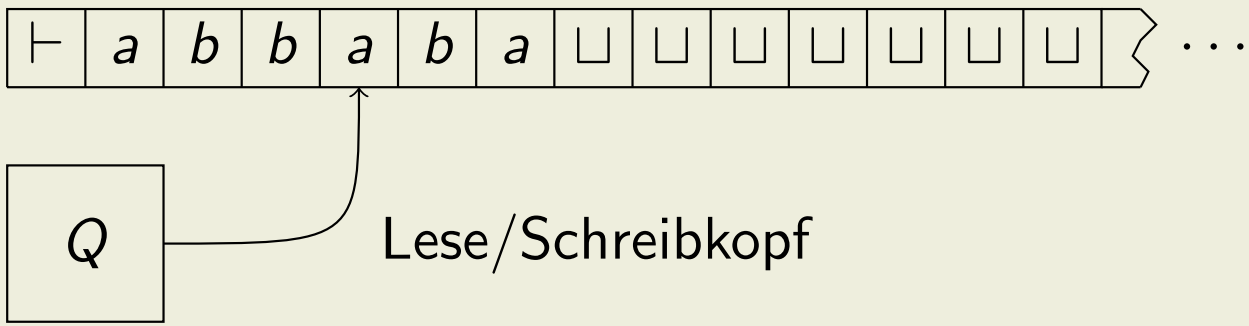
Definition (informell)

deterministische, einbändige Turingmaschine (TM):



Definition (informell)

deterministische, einbändige Turingmaschine (TM):



- Eine TM verwendet ein einseitig unendliches Band als Speicher
- Zu Beginn der Berechnung steht die Eingabe auf dem Band
- Das Band wird mit einem Lese/Schreibkopf gelesen oder beschrieben
- Verhalten der TM wird durch die endliche Kontrolle Q bestimmt

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
5. $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
5. $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,
6. $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die Übergangsfunktion,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
5. $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,
6. $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die Übergangsfunktion,
7. $s \in Q$, der Startzustand,

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
5. $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,
6. $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die Übergangsfunktion,
7. $s \in Q$, der Startzustand,
8. $t \in Q$, der akzeptierende Zustand und

Definition (formal)

Eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
5. $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,
6. $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die Übergangsfunktion,
7. $s \in Q$, der Startzustand,
8. $t \in Q$, der akzeptierende Zustand und
9. $r \in Q$, der verwerfende Zustand mit $t \neq r$.

Übergangsfunktion

$\delta(p, a) = (q, b, d)$ bedeutet: Wenn M im Zustand p das Symbol a liest:

1. wechselt M in den Zustand q,
2. ersetzt M auf dem Band das Zeichen a durch das Zeichen b,
3. bewegt M den Lese/Schreibkopf einen Schritt in Richtung d.

Übergangsfunktion

$\delta(p, a) = (q, b, d)$ bedeutet: Wenn M im Zustand p das Symbol a liest:

1. wechselt M in den Zustand q,
2. ersetzt M auf dem Band das Zeichen a durch das Zeichen b,
3. bewegt M den Lese/Schreibkopf einen Schritt in Richtung d.

Zusatzbedingungen

- Der linke Endmarker darf nicht überschrieben werden

$$\forall\, p \in Q, \exists\, q \in Q \quad \delta(p, \vdash) = (q, \vdash, \mathrm{R})$$

Übergangsfunktion

$\delta(p, a) = (q, b, d)$ bedeutet: Wenn M im Zustand p das Symbol a liest:

1. wechselt M in den Zustand q,
2. ersetzt M auf dem Band das Zeichen a durch das Zeichen b,
3. bewegt M den Lese/Schreibkopf einen Schritt in Richtung d.

Zusatzbedingungen

- Der linke Endmarker darf nicht überschrieben werden

$$\forall\ p \in Q, \exists\ q \in Q \quad \delta(p, \vdash) = (q, \vdash, \mathrm{R})$$

- M verlässt akzeptierenden/verwerfenden Zustand nicht

$$\forall\ b \in \Gamma, \exists\ c, c' \in \Gamma \text{ und } d, d' \in \{\mathrm{L}, \mathrm{R}\}: \quad \delta(t, b) = (t, c, d)$$
$$\delta(r, b) = (r, c', d')$$

Beispiel

Sei $M = (\{s, t, r, q_1, q_2, q_3\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1, \mathrm{X}, \mathrm{Y}\}, \vdash, \sqcup, \delta, s, t, r)$

mit δ wie folgt:

	$\vdash$	$\sqcup$	0	1	X	Y
s	$(s, \vdash, R)$	$(r, \sqcup, R)$	(q_1, X, R)	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)
q_1	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_1, 0, R)$	(q_2, Y, L)	(r, X, R)	(q_1, Y, R)
q_2	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_2, 0, L)$	$(r, 1, R)$	(s, X, R)	(q_2, Y, L)
q_3	$(r, \vdash, R)$	$(t, \sqcup, R)$	$(r, 0, R)$	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)
t	$(t, \vdash, R)$	$(t, \vdash, R)$	$(t, \vdash, R)$	$(t, \vdash, R)$	$(t, \vdash, R)$	$(t, \vdash, R)$
r	$(r, \vdash, R)$	$(r, \vdash, R)$	$(r, \vdash, R)$	$(r, \vdash, R)$	$(r, \vdash, R)$	$(r, \vdash, R)$

Beispiel

Sei $M = (\{s, t, r, q_1, q_2, q_3\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1, \mathrm{X}, \mathrm{Y}\}, \vdash, \sqcup, \delta, s, t, r)$

mit δ wie folgt:

	$\vdash$	$\sqcup$	0	1	X	Y
s	$(s, \vdash, R)$	$(r, \sqcup, R)$	(q_1, X, R)	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)
q_1	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_1, 0, R)$	(q_2, Y, L)	(r, X, R)	(q_1, Y, R)
q_2	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_2, 0, L)$	$(r, 1, R)$	(s, X, R)	(q_2, Y, L)
q_3	$(r, \vdash, R)$	$(t, \sqcup, R)$	$(r, 0, R)$	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)

Beispiel

Sei $M = (\{s, t, r, q_1, q_2, q_3\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1, \mathrm{X}, \mathrm{Y}\}, \vdash, \sqcup, \delta, s, t, r)$

mit δ wie folgt:

	$\vdash$	$\sqcup$	0	1	X	Y
s	$(s, \vdash, R)$	$(r, \sqcup, R)$	(q_1, X, R)	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)
q_1	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_1, 0, R)$	(q_2, Y, L)	(r, X, R)	(q_1, Y, R)
q_2	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_2, 0, L)$	$(r, 1, R)$	(s, X, R)	(q_2, Y, L)
q_3	$(r, \vdash, R)$	$(t, \sqcup, R)$	$(r, 0, R)$	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)

M akzeptiert die Sprache $\{0^n 1^n \mid n \geqslant 1\}$

Beispiel

Sei $M = (\{s, t, r, q_1, q_2, q_3\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1, \mathrm{X}, \mathrm{Y}\}, \vdash, \sqcup, \delta, s, t, r)$

mit δ wie folgt:

	$\vdash$	$\sqcup$	0	1	X	Y
s	$(s, \vdash, R)$	$(r, \sqcup, R)$	(q_1, X, R)	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)
q_1	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_1, 0, R)$	(q_2, Y, L)	(r, X, R)	(q_1, Y, R)
q_2	$(r, \vdash, R)$	$(r, \sqcup, R)$	$(q_2, 0, L)$	$(r, 1, R)$	(s, X, R)	(q_2, Y, L)
q_3	$(r, \vdash, R)$	$(t, \sqcup, R)$	$(r, 0, R)$	$(r, 1, R)$	(r, X, R)	(q_3, Y, R)

M akzeptiert die Sprache $\{0^n 1^n \mid n \geqslant 1\}$

Algorithmus

Schleife:

1. Ersetze erste Null durch X.
2. Ersetze erste Eins durch Y.
3. Wiederhole Schritt 1.

Abbruch:

1. Wenn aktuelles Zeichen $\sqcup$, verwerfe.
2. Wenn rechts am Band $Y^n \sqcup \ldots$ steht, akzeptiere.