Theoretische Informatik I

Einheit 2.4

Reguläre Ausdrücke

1. Anwendungen
2. Syntax und Semantik
3. Vereinfachungsregeln
4. Beziehung zu endlichen Automaten

Eine algebraische Beschreibung für Sprachen

- **Automaten beschreiben Abarbeitung von Sprachen**
  – **Operationale Semantik**: Symbole führen zu Zustandsänderungen
  – Bestimme Worte bzw. Symbolketten werden durch Zustände akzeptiert
  – Für Automaten ist Sprache $\hat{=}$ Menge der akzeptierten Worte

Eine algebraische Beschreibung für Sprachen

- **Automaten beschreiben Abarbeitung von Sprachen**
  - **Operationale Semantik**: Symbole führen zu Zustandsänderungen
  - Bestimme Worte bzw. Symbolketten werden durch Zustände akzeptiert
  - Für Automaten ist Sprache $\hat{=}$ Menge der akzeptierten Worte

- **Wie beschreibt man Eigenschaften von Worten?**
  - **Deklarative Semantik**: äußere Form von Zeichenreihen einer Sprache
    z.B. *Worte haben eine führende Null, dann beliebig viele Einsen*
  - Anwendungen brauchen präzise Beschreibungssprache für Worte
    - Grundeinheiten von Programmiersprachen, Suchmuster für Browser, ...

Eine algebraische Beschreibung für Sprachen

- **Automaten beschreiben Abarbeitung von Sprachen**
  – **Operationale Semantik**: Symbole führen zu Zustandsänderungen
  – Bestimme Worte bzw. Symbolketten werden durch Zustände akzeptiert
  – Für Automaten ist Sprache $\hat{=}$ Menge der akzeptierten Worte

- **Wie beschreibt man Eigenschaften von Worten?**
  – **Deklarative Semantik**: äußere Form von Zeichenreihen einer Sprache
  z.B. *Worte haben eine führende Null, dann beliebig viele Einsen*
  – Anwendungen brauchen präzise Beschreibungssprache für Worte
  · Grundeinheiten von Programmiersprachen, Suchmuster für Browser, ...

- **Reguläre Ausdrücke als formale Syntax**
  – Kurze, prägnante Beschreibung des Aufbaus der Worte einer Sprache
  z.B. `01*`: "Zuerst eine Null, dann beliebig viele Einsen"

Reguläre Ausdrücke beschreiben Sprachen

- **Ausdrücke für einfache Grundmengen**
  - Leere Menge
  - Sprache, die nur das leere Wort enthält
  - Sprache, die nur das Symbol $a \in \Sigma$ enthält

Reguläre Ausdrücke beschreiben Sprachen

- **Ausdrücke für einfache Grundmengen**
  - Leere Menge
  - Sprache, die nur das leere Wort enthält
  - Sprache, die nur das Symbol $a \in \Sigma$ enthält
- **Ausdrücke für Komposition von Mengen**
  - Vereinigung $L \cup M$ von Sprachen

Reguläre Ausdrücke beschreiben Sprachen

- **Ausdrücke für einfache Grundmengen**
  - Leere Menge
  - Sprache, die nur das leere Wort enthält
  - Sprache, die nur das Symbol $a \in \Sigma$ enthält
- **Ausdrücke für Komposition von Mengen**
  - Vereinigung $L \cup M$ von Sprachen
  - Verkettung $L \circ M$ von Sprachen

    $\boldsymbol{L \circ M} = \{w \in \Sigma^* \mid \exists u \in L.\ \exists v \in M.\ w = uv\}$

Reguläre Ausdrücke beschreiben Sprachen

- **Ausdrücke für einfache Grundmengen**
  - Leere Menge
  - Sprache, die nur das leere Wort enthält
  - Sprache, die nur das Symbol $a \in \Sigma$ enthält

- **Ausdrücke für Komposition von Mengen**
  - Vereinigung $L \cup M$ von Sprachen
  - Verkettung $L \circ M$ von Sprachen

    $\boldsymbol{L \circ M} = \{w \in \Sigma^* \mid \exists u \in L.\ \exists v \in M.\ w = uv\}$
  - (Kleene'sche) Hülle L^* von Sprachen

    $\boldsymbol{L^n} = \{w \in \Sigma^* \mid \exists u_1, .., u_n \in L.\ w = u_1..u_n\}$ $(n \in \mathbb{N})$ $\quad (L^0 = \{\epsilon\})$

    $\boldsymbol{L^*} = \bigcup_{n \geq 0} L^n = \{w \in \Sigma^* \mid \exists n \in \mathbb{N}.\ \exists u_1, .., u_n \in L.\ w = u_1..u_n\}$

Reguläre Ausdrücke beschreiben Sprachen

- **Ausdrücke für einfache Grundmengen**
  - Leere Menge
  - Sprache, die nur das leere Wort enthält
  - Sprache, die nur das Symbol $a \in \Sigma$ enthält

- **Ausdrücke für Komposition von Mengen**
  - Vereinigung $L \cup M$ von Sprachen
  - Verkettung $L \circ M$ von Sprachen

    $\boldsymbol{L \circ M} = \{w \in \Sigma^* \mid \exists u \in L.\ \exists v \in M.\ w = uv\}$
  - (Kleene'sche) Hülle L^* von Sprachen

    $\boldsymbol{L^n} = \{w \in \Sigma^* \mid \exists u_1, .., u_n \in L.\ w = u_1..u_n\}$ $(n \in \mathbb{N})$ $\quad (L^0 = \{\epsilon\})$

    $\boldsymbol{L^*} = \bigcup_{n \geq 0} L^n = \{w \in \Sigma^* \mid \exists n \in \mathbb{N}.\ \exists u_1, .., u_n \in L.\ w = u_1..u_n\}$

- **Reguläre Ausdrücke sind nicht selbst Mengen**
  - Nur eine syntaktische Beschreibungsform, die ein Computer versteht

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

– A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

– A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.

– Fundamental building blocks are expressions that match a single character.

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.

Anwendung: Suchmuster in Unix

grep searches files for lines containing a match to a given pattern

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.
- Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters.
  For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`.

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.
- Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters. For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`.
- Certain named classes of characters are predefined within bracket expressions. They are `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`, ...

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.
- Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters. For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`.
- Certain named classes of characters are predefined within bracket expressions. They are `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`, ...
- The period `.` matches any single character.
- The caret `^` and the dollar sign `$` are metacharacters that match the empty string ...

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.
- Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters. For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`.
- Certain named classes of characters are predefined within bracket expressions. They are `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`, ...
- The period `.` matches any single character.
- The caret `^` and the dollar sign `$` are metacharacters that match the empty string ...
- A regular expression may be followed by one of several repetition operators:
  `?`: The preceding item is optional and matched at most once.
  `*`: The preceding item will be matched zero or more times.
  `+` The preceding item will be matched one or more times.

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.
- Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters. For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`.
- Certain named classes of characters are predefined within bracket expressions. They are `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`, ...
- The period `.` matches any single character.
- The caret `^` and the dollar sign `$` are metacharacters that match the empty string ...
- A regular expression may be followed by one of several repetition operators:
  `?`: The preceding item is optional and matched at most once.
  `*`: The preceding item will be matched zero or more times.
  `+` The preceding item will be matched one or more times.
- Two regular expressions may be concatenated; the resulting regular expression matches any string concatenating two substrings that match the subexpressions.

Anwendung: Suchmuster in Unix

grep *searches files for lines containing a match to a given pattern*

- A regular expression is a pattern that describes a set of strings. Regular expressions are constructed by using various operators to combine smaller expressions.
- Fundamental building blocks are expressions that match a single character.
- A bracket expression is a list of characters enclosed by `[` and `]`. It matches any single character in that list. For example, `[0123456789]` matches any single digit.
- Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters. For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`.
- Certain named classes of characters are predefined within bracket expressions. They are `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`, ...
- The period `.` matches any single character.
- The caret `^` and the dollar sign `$` are metacharacters that match the empty string ...
- A regular expression may be followed by one of several repetition operators:
  `?`: The preceding item is optional and matched at most once.
  `*`: The preceding item will be matched zero or more times.
  `+` The preceding item will be matched one or more times.
- Two regular expressions may be concatenated; the resulting regular expression matches any string concatenating two substrings that match the subexpressions.
- Two regular expressions may be joined by the infix operator `|`
  The resulting regular expression matches any string matching either subexpression.

Anwendung: Lexikalische Analyse

Wichtigster Grundbestandteil von Compilern

- **Reguläre Ausdrücke beschreiben Token**
  - Logische Grundeinheiten von Programmiersprachen
  - z.B. Schlüsselwörter, Bezeichner, Dezimalzahlen, ...

Anwendung: Lexikalische Analyse

Wichtigster Grundbestandteil von Compilern

- **Reguläre Ausdrücke beschreiben Token**
  - Logische Grundeinheiten von Programmiersprachen
  - z.B. Schlüsselwörter, Bezeichner, Dezimalzahlen, ...
- **“Lexer” transformieren reguläre Ausdrücke in Analyseprogramme**
  - Analyse kann die Token der Programmiersprache identifizieren

Anwendung: Lexikalische Analyse

Wichtigster Grundbestandteil von Compilern

- **Reguläre Ausdrücke beschreiben Token**
  - Logische Grundeinheiten von Programmiersprachen
  - z.B. Schlüsselwörter, Bezeichner, Dezimalzahlen, ...
- **"Lexer" transformieren reguläre Ausdrücke in Analyseprogramme**
  - Analyse kann die Token der Programmiersprache identifizieren
  - Zugrundeliegende Technik: Umwandlung regulärer Ausdrücke in DEAs

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über** $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über** $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$
  - $\emptyset$, ϵ, und `a` (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $\boldsymbol{E}+\boldsymbol{F}$, $\boldsymbol{E}\circ\boldsymbol{F}$, $\boldsymbol{E}^*$ und $(\boldsymbol{E})$

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$**
  - $\emptyset$, ϵ, und `a` (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $\boldsymbol{E+F}$, $\boldsymbol{E\circ F}$, $\boldsymbol{E^*}$ und $\boldsymbol{(E)}$
- **Semantik: Sprachen über Σ**
  - $\boldsymbol{L(E)}$: Sprache des regulären Ausdrucks E, induktiv definiert

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$**
  - $\emptyset$, ϵ, und $\mathtt{a}$ (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $E+F$, $E\circ F$, E^* und (E)
- **Semantik: Sprachen über Σ**
  - $L(E)$: Sprache des regulären Ausdrucks E, induktiv definiert
  - $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(\mathtt{a}) = \{a\}$ (für alle $a \in \Sigma$)

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$**
  - $\emptyset$, ϵ, und $\mathtt{a}$ (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $\boldsymbol{E}\mathtt{+}\boldsymbol{F}$, $\boldsymbol{E}\circ\boldsymbol{F}$, $\boldsymbol{E}^*$ und $\mathtt{(}\boldsymbol{E}\mathtt{)}$
- **Semantik: Sprachen über Σ**
  - $\boldsymbol{L(E)}$: Sprache des regulären Ausdrucks E, induktiv definiert
  - $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(\mathtt{a}) = \{a\}$ (für alle $a \in \Sigma$)
  - $L(E\mathtt{+}F) = L(E) \cup L(F)$, $L(E\circ F) = L(E) \circ L(F)$, $L(E^*) = (L(E))^*$, $L(\mathtt{(}E\mathtt{)}) = L(E)$

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$**
  - $\emptyset$, ϵ, und $\mathtt{a}$ (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $\boldsymbol{E}\mathtt{+}\boldsymbol{F}$, $\boldsymbol{E}\circ\boldsymbol{F}$, $\boldsymbol{E}^*$ und $\mathtt{(}\boldsymbol{E}\mathtt{)}$
- **Semantik: Sprachen über Σ**
  - $\boldsymbol{L(E)}$: Sprache des regulären Ausdrucks E, induktiv definiert
  - $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(\mathtt{a}) = \{a\}$ (für alle $a \in \Sigma$)
  - $L(E\mathtt{+}F) = L(E) \cup L(F)$, $L(E\circ F) = L(E)\circ L(F)$, $L(E^*) = (L(E))^*$,
    $L(\mathtt{(}E\mathtt{)}) = L(E)$
- **Konventionen**
  - $E\circ F$ wird üblicherweise als $\boldsymbol{EF}$ abgekürzt

Reguläre Ausdrücke präzisiert

- **Syntax:** **Terme über** $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$
  - $\emptyset$, ϵ, und $\mathtt{a}$ (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $\boldsymbol{E}$+$\boldsymbol{F}$, $\boldsymbol{E}\circ\boldsymbol{F}$, $\boldsymbol{E}^*$ und $(\boldsymbol{E})$
- **Semantik:** **Sprachen über** Σ
  - $\boldsymbol{L(E)}$: Sprache des regulären Ausdrucks E, induktiv definiert
  - $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(\mathtt{a}) = \{a\}$ (für alle $a \in \Sigma$)
  - $L(E+F) = L(E) \cup L(F)$, $L(E \circ F) = L(E) \circ L(F)$, $L(E^*) = (L(E))^*$, $L((E)) = L(E)$
- **Konventionen**
  - $E \circ F$ wird üblicherweise als $\boldsymbol{EF}$ abgekürzt
  - Definitorische Abkürzungen: $\boldsymbol{E}^+ \equiv EE^*$, $[\boldsymbol{a_1 \ldots a_n}] \equiv a_1 + \ldots + a_n$

Reguläre Ausdrücke präzisiert

- **Syntax: Terme über $\Sigma \cup \{\emptyset, \epsilon, +, \circ, ^*, (,)\}$**
  - $\emptyset$, ϵ, und a (für alle $a \in \Sigma$) sind reguläre Ausdrücke
  - Sind E und F reguläre Ausdrücke, dann auch $E+F$, $E\circ F$, E^* und (E)
- **Semantik: Sprachen über Σ**
  - $L(E)$: Sprache des regulären Ausdrucks E, induktiv definiert
  - $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$, $L(\mathtt{a}) = \{a\}$ (für alle $a \in \Sigma$)
  - $L(E+F) = L(E) \cup L(F)$, $L(E\circ F) = L(E)\circ L(F)$, $L(E^*) = (L(E))^*$, $L((E)) = L(E)$
- **Konventionen**
  - $E\circ F$ wird üblicherweise als EF abgekürzt
  - Definitorische Abkürzungen: $E^+ \equiv EE^*$, $[a_1 ... a_n] \equiv a_1 + ... + a_n$
  - Prioritätsregelungen ermöglichen es, überflüssige Klammern wegzulassen
    - $*$ ("Sternoperator") bindet stärker als $\circ$, und dies stärker als $+$
    - Verkettung $\circ$ und Vereinigung $+$ sind assoziativ

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  – Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  - Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten
  - Beschreibung von Mengen (auf Folie, Tafel, ...) benötigt textuelle Notation
  - Notation benutzt Kurzschreibweisen wie $\cup$, $\circ$, $*$ für Mengenoperationen

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  - Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten
  - Beschreibung von Mengen (auf Folie, Tafel, ...) benötigt textuelle Notation
  - Notation benutzt Kurzschreibweisen wie ∪, ∘, * für Mengenoperationen

    ... aber ist selbst nur ein Hilfsmittel zur Kommunikation

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  - Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten
  - Beschreibung von Mengen (auf Folie, Tafel, ...) benötigt textuelle Notation
  - Notation benutzt Kurzschreibweisen wie $\cup$, $\circ$, $*$ für Mengenoperationen

    ... aber ist selbst nur ein Hilfsmittel zur Kommunikation

- **Reguläre Ausdrücke sind Terme**
  - Syntaktisches Konstrukt: Struktur, die ein Computer versteht

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  - Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten
  - Beschreibung von Mengen (auf Folie, Tafel, ...) benötigt textuelle Notation
  - Notation benutzt Kurzschreibweisen wie $\cup$, $\circ$, $*$ für Mengenoperationen

    ... aber ist selbst nur ein Hilfsmittel zur Kommunikation
- **Reguläre Ausdrücke sind Terme**
  - Syntaktisches Konstrukt: Struktur, die ein Computer versteht
  - Reguläre Ausdrücke werden zur Beschreibung von Sprachen benutzt

    und sind ähnlich zur Standardnotation von Mengen

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  - Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten
  - Beschreibung von Mengen (auf Folie, Tafel, ...) benötigt textuelle Notation
  - Notation benutzt Kurzschreibweisen wie $\cup$, $\circ$, $*$ für Mengenoperationen

    ... aber ist selbst nur ein Hilfsmittel zur Kommunikation
- **Reguläre Ausdrücke sind Terme**
  - Syntaktisches Konstrukt: Struktur, die ein Computer versteht
  - Reguläre Ausdrücke werden zur Beschreibung von Sprachen benutzt

    und sind ähnlich zur Standardnotation von Mengen
- **Reguläre Ausdrücke sind selbst keine Mengen**
  - Unterscheide Ausdruck E von Sprache des Ausdrucks $L(E)$

Sprachen vs. Ausdrücke

- **Sprachen sind Mengen**
  – Abstraktes Semantisches Konzept: Ungeordnete Kollektion von Worten
  – Beschreibung von Mengen (auf Folie, Tafel, ...) benötigt textuelle Notation
  – Notation benutzt Kurzschreibweisen wie $\cup$, $\circ$, $*$ für Mengenoperationen
  ... aber ist selbst nur ein Hilfsmittel zur Kommunikation

- **Reguläre Ausdrücke sind Terme**
  – Syntaktisches Konstrukt: Struktur, die ein Computer versteht
  – Reguläre Ausdrücke werden zur Beschreibung von Sprachen benutzt
  und sind ähnlich zur Standardnotation von Mengen

- **Reguläre Ausdrücke sind selbst keine Mengen**
  – Unterscheide Ausdruck E von Sprache des Ausdrucks $L(E)$
  – Man verzichtet auf den Unterschied wenn der Kontext eindeutig ist

Entwicklung regulärer Ausdrücke

Beschreibe Menge aller Worte, in denen 0 und 1 abwechseln

1. Regulärer Ausdruck für die Sprache $\{01\}$

– `0` repräsentiert $\{0\}$, `1` repräsentiert $\{1\}$

– Also ist $L(\texttt{01}) \;=\; L(\texttt{0})\circ L(\texttt{1}) \;=\; \{0\}\circ\{1\} \;=\; \{01\}$

Entwicklung regulärer Ausdrücke

Beschreibe Menge aller Worte, in denen 0 und 1 abwechseln

1. Regulärer Ausdruck für die Sprache $\{01\}$

– `0` repräsentiert $\{0\}$, `1` repräsentiert $\{1\}$

– Also ist $L(\texttt{01}) = L(\texttt{0})\circ L(\texttt{1}) = \{0\}\circ\{1\} = \{01\}$

2. Erzeuge $\{01, 0101, 010101, \ldots\}$ durch Sternbildung

– $L(\texttt{(01)*}) = L(\texttt{01})^* = \{01\}^* = \{01, 0101, 010101, \ldots\}$

Entwicklung regulärer Ausdrücke

Beschreibe Menge aller Worte, in denen 0 und 1 abwechseln

1. Regulärer Ausdruck für die Sprache $\{01\}$

– `0` repräsentiert $\{0\}$, `1` repräsentiert $\{1\}$

– Also ist $L(\texttt{01}) = L(\texttt{0})\circ L(\texttt{1}) = \{0\}\circ\{1\} = \{01\}$

2. Erzeuge $\{01, 0101, 010101, \ldots\}$ durch Sternbildung

– $L(\texttt{(01)}^*) = L(\texttt{01})^* = \{01\}^* = \{01, 0101, 010101, \ldots\}$

3. Manche Worte nicht erfaßt

– Start mit Eins statt Null: $\texttt{(10)}^*$

– Start und Ende mit Null: $\texttt{(01)}^*\texttt{0}$

– Start und Ende mit Eins: $\texttt{(10)}^*\texttt{1}$

Vollständiger Ausdruck: $\texttt{(01)}^* + \texttt{(10)}^* + \texttt{(01)}^*\texttt{0} + \texttt{(10)}^*\texttt{1}$

Entwicklung regulärer Ausdrücke

Beschreibe Menge aller Worte, in denen 0 und 1 abwechseln

1. Regulärer Ausdruck für die Sprache $\{01\}$

– `0` repräsentiert $\{0\}$, `1` repräsentiert $\{1\}$

– Also ist $L(\texttt{01}) = L(\texttt{0})\circ L(\texttt{1}) = \{0\}\circ\{1\} = \{01\}$

2. Erzeuge $\{01, 0101, 010101, \ldots\}$ durch Sternbildung

– $L(\texttt{(01)}^*) = L(\texttt{01})^* = \{01\}^* = \{01, 0101, 010101, \ldots\}$

3. Manche Worte nicht erfaßt

– Start mit Eins statt Null: $\texttt{(10)}^*$

– Start und Ende mit Null: $\texttt{(01)}^*\texttt{0}$

– Start und Ende mit Eins: $\texttt{(10)}^*\texttt{1}$

Vollständiger Ausdruck: $\texttt{(01)}^* + \texttt{(10)}^* + \texttt{(01)}^*\texttt{0} + \texttt{(10)}^*\texttt{1}$

4. Es geht auch kürzer

– Optional 1 am Anfang oder 0 am Ende: $\texttt{(}\epsilon\texttt{+1)(01)}^*\texttt{(}\epsilon\texttt{+0)}$

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E} \cong \boldsymbol{F}$, falls $L(E) = L(F)$ "E äquivalent zu F"

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E \cong F}$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist `(01)`*`+(10)`*`+(01)`*`0+(10)`*`1` $\cong$ `(`ϵ`+1)(01)`*`(`ϵ`+0)` ?

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - ***E*** $\cong$ ***F***, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist `(01)`* `+` `(10)`* `+` `(01)`*`0` `+` `(10)`*`1` $\cong$ `(`ϵ`+1)(01)`*`(`ϵ`+0)` ?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E \cong F}$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist `(01)`* + `(10)`* + `(01)`*`0` + `(10)`*`1` $\cong$ `(`ϵ`+1)(01)`*`(`ϵ`+0)` ?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

- **Assoziativität von $\circ$ und +**

  $\boldsymbol{(E \circ F) \circ G \cong E \circ (F \circ G)}$:

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E \cong F}$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist `(01)`*`+(10)`*`+(01)`*`0+(10)`*`1` $\cong$ `(`ϵ`+1)(01)`*`(`ϵ`+0)` ?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

- **Assoziativität von $\circ$ und +**

  $\boldsymbol{(E\circ F)\circ G \cong E\circ(F\circ G)}$:
  - $L((E\circ F)\circ G) = L(E\circ F)\circ L(G) = L(E)\circ L(F)\circ L(G) = L(E)\circ L(F\circ G) = L(E\circ(F\circ G))$

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E \cong F}$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist $\mathtt{(01)}^*\mathtt{+(10)}^*\mathtt{+(01)}^*\mathtt{0+(10)}^*\mathtt{1} \cong \mathtt{(}\epsilon\mathtt{+1)(01)}^*\mathtt{(}\epsilon\mathtt{+0)}$?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

- **Assoziativität von $\circ$ und +**

  $\boldsymbol{(E\circ F)\circ G \cong E\circ(F\circ G)}$:
  - $L((E\circ F)\circ G) = L(E\circ F)\circ L(G) = L(E)\circ L(F)\circ L(G) = L(E)\circ L(F\circ G) = L(E\circ(F\circ G))$

  $\boldsymbol{(E+F)+G \cong E+(F+G)}$:

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $E \cong F$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist $\texttt{(01)}^* + \texttt{(10)}^* + \texttt{(01)}^*\texttt{0} + \texttt{(10)}^*\texttt{1} \cong (\epsilon\texttt{+1})\texttt{(01)}^*(\epsilon\texttt{+0})$?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

- **Assoziativität von $\circ$ und +**

  $(E\circ F)\circ G \cong E\circ(F\circ G)$:
  - $L((E\circ F)\circ G) = L(E\circ F)\circ L(G) = L(E)\circ L(F)\circ L(G) = L(E)\circ L(F\circ G) = L(E\circ(F\circ G))$

  $(E+F)+G \cong E+(F+G)$:
  - $L((E+F)+G) = L(E+F)\cup L(G) = L(E)\cup L(F)\cup L(G) = \ldots = L(E+(F+G))$

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E \cong F}$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist $\texttt{(01)}^* + \texttt{(10)}^* + \texttt{(01)}^*\texttt{0} + \texttt{(10)}^*\texttt{1} \cong \texttt{(}\epsilon\texttt{+1)(01)}^*\texttt{(}\epsilon\texttt{+0)}$?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

- **Assoziativität von $\circ$ und +**

  $\boldsymbol{(E\circ F)\circ G \cong E\circ(F\circ G)}$:
  - $L((E\circ F)\circ G) = L(E\circ F)\circ L(G) = L(E)\circ L(F)\circ L(G) = L(E)\circ L(F\circ G) = L(E\circ(F\circ G))$

  $\boldsymbol{(E+F)+G \cong E+(F+G)}$:
  - $L((E+F)+G) = L(E+F)\cup L(G) = L(E)\cup L(F)\cup L(G) = \ldots = L(E+(F+G))$

- **Kommutativität von +**
  - $\boldsymbol{E+F \cong F+E}$: $L(E+F) = = L(E)\cup L(F) = L(F)\cup L(E) = L(F+E)$

Algebraische Gesetze für regulärer Ausdrücke

Hilfsmittel zur Vereinfachung regulärer Ausdrücke

- **Äquivalenz von Ausdrücken**
  - $\boldsymbol{E \cong F}$, falls $L(E) = L(F)$ "E äquivalent zu F"
  - Was beweist `(01)`*`+(10)`*`+(01)`*`0+(10)`*`1` $\cong$ `(`ϵ`+1)(01)`*`(`ϵ`+0)` ?
  - Algebraische Gesetze erlauben Umschreiben in äquivalente Ausdrücke

- **Assoziativität von $\circ$ und +**

  $\boldsymbol{(E\circ F)\circ G \cong E\circ(F\circ G)}$:
  - $L((E\circ F)\circ G) = L(E\circ F)\circ L(G) = L(E)\circ L(F)\circ L(G) = L(E)\circ L(F\circ G) = L(E\circ(F\circ G))$

  $\boldsymbol{(E+F)+G \cong E+(F+G)}$:
  - $L((E+F)+G) = L(E+F)\cup L(G) = L(E)\cup L(F)\cup L(G) = \ldots = L(E+(F+G))$

- **Kommutativität von +**
  - $\boldsymbol{E+F \cong F+E}$: $L(E+F) == L(E)\cup L(F) = L(F)\cup L(E) = L(F+E)$
  - Kommutativität von $\circ$ gilt nicht: $= L($`01`$) = \{01\} \neq \{10\} = L($`10`$)$

Algebraische Gesetze II

- **Einheiten und Annihilatoren**
  - $\emptyset{+}E \cong E \cong E{+}\emptyset$, $\epsilon{\circ}E \cong E \cong E{\circ}\epsilon$, $\emptyset{\circ}E \cong \emptyset \cong E{\circ}\emptyset$

Algebraische Gesetze II

- **Einheiten und Annihilatoren**
  - $\emptyset{+}E \cong E \cong E{+}\emptyset$, $\quad \epsilon{\circ}E \cong E \cong E{\circ}\epsilon$, $\quad \emptyset{\circ}E \cong \emptyset \cong E{\circ}\emptyset$
- **Distributivgesetze**
  - $(E{+}F){\circ}G \cong E{\circ}G{+}F{\circ}G$:

Algebraische Gesetze II

- **Einheiten und Annihilatoren**
  - $\emptyset{+}E \cong E \cong E{+}\emptyset$, $\epsilon{\circ}E \cong E \cong E{\circ}\epsilon$, $\emptyset{\circ}E \cong \emptyset \cong E{\circ}\emptyset$
- **Distributivgesetze**
  - $(E{+}F){\circ}G \cong E{\circ}G{+}F{\circ}G$:

$$L((E{+}F){\circ}G) = (L(E){\cup}L(F)){\circ}L(G)$$
$$= \{w \in \Sigma^* \mid \exists u \in L(E){\cup}L(F).\exists v \in L(G).w = uv\}$$
$$= \{w \in \Sigma^* \mid \exists u \in L(E).\exists v \in L(G).w = uv \vee \exists u \in L(F).\exists v \in L(G).w = uv\}$$
$$= \{w \in \Sigma^* \mid \exists u \in L(E).\exists v \in L(G).w = uv\} \cup \{w \in \Sigma^* \mid \exists u \in L(F).\exists v \in L(G).w = uv\}$$
$$= L(E){\circ}L(G){\cup}L(F){\circ}L(G) = L(E{\circ}G{+}F{\circ}G)$$

Algebraische Gesetze II

- **Einheiten und Annihilatoren**
  - $\boldsymbol{\emptyset{+}E \cong E \cong E{+}\emptyset}$, $\boldsymbol{\epsilon{\circ}E \cong E \cong E{\circ}\epsilon}$, $\boldsymbol{\emptyset{\circ}E \cong \emptyset \cong E{\circ}\emptyset}$
- **Distributivgesetze**
  - $\boldsymbol{(E{+}F){\circ}G \cong E{\circ}G{+}F{\circ}G}$:

$$
\begin{aligned}
& L((E{+}F){\circ}G) = (L(E){\cup}L(F)){\circ}L(G) \\
& = \{w \in \Sigma^* \mid \exists u \in L(E){\cup}L(F).\exists v \in L(G).w = uv\} \\
& = \{w \in \Sigma^* \mid \exists u \in L(E).\exists v \in L(G).w = uv \vee \exists u \in L(F).\exists v \in L(G).w = uv\} \\
& = \{w \in \Sigma^* \mid \exists u \in L(E).\exists v \in L(G).w = uv\} \cup \{w \in \Sigma^* \mid \exists u \in L(F).\exists v \in L(G).w = uv\} \\
& = L(E){\circ}L(G){\cup}L(F){\circ}L(G) = L(E{\circ}G{+}F{\circ}G)
\end{aligned}
$$

  - $\boldsymbol{G{\circ}(E{+}F) \cong G{\circ}E + G{\circ}F}$

ALGEBRAISCHE GESETZE II

- **Einheiten und Annihilatoren**
  - $\emptyset + E \cong E \cong E + \emptyset$, $\epsilon \circ E \cong E \cong E \circ \epsilon$, $\emptyset \circ E \cong \emptyset \cong E \circ \emptyset$
- **Distributivgesetze**
  - $(E+F)\circ G \cong E\circ G + F\circ G$:

$$\begin{aligned}
&L((E+F)\circ G) = (L(E)\cup L(F))\circ L(G)\\
&= \{w\in\Sigma^* \mid \exists u\in L(E)\cup L(F).\exists v\in L(G).w = uv\}\\
&= \{w\in\Sigma^* \mid \exists u\in L(E).\exists v\in L(G).w = uv \vee \exists u\in L(F).\exists v\in L(G).w = uv\}\\
&= \{w\in\Sigma^* \mid \exists u\in L(E).\exists v\in L(G).w = uv\} \cup \{w\in\Sigma^* \mid \exists u\in L(F).\exists v\in L(G).w = uv\}\\
&= L(E)\circ L(G)\cup L(F)\circ L(G) = L(E\circ G+F\circ G)
\end{aligned}$$

  - $G\circ(E+F) \cong G\circ E + G\circ F$
- **Idempotenz von +**
  - $E+E \cong E$

Algebraische Gesetze II

- **Einheiten und Annihilatoren**
  - $\emptyset{+}E \cong E \cong E{+}\emptyset, \quad \epsilon\circ E \cong E \cong E\circ\epsilon, \quad \emptyset\circ E \cong \emptyset \cong E\circ\emptyset$
- **Distributivgesetze**
  - $(E{+}F)\circ G \cong E\circ G{+}F\circ G$:

    $L((E{+}F)\circ G) = (L(E)\cup L(F))\circ L(G)$

    $= \{w\in\Sigma^* \mid \exists u\in L(E)\cup L(F).\exists v\in L(G).w = uv\}$

    $= \{w\in\Sigma^* \mid \exists u\in L(E).\exists v\in L(G).w = uv \vee \exists u\in L(F).\exists v\in L(G).w = uv\}$

    $= \{w\in\Sigma^* \mid \exists u\in L(E).\exists v\in L(G).w = uv\} \cup \{w\in\Sigma^* \mid \exists u\in L(F).\exists v\in L(G).w = uv\}$

    $= L(E)\circ L(G)\cup L(F)\circ L(G) = L(E\circ G{+}F\circ G)$
  - $G\circ(E{+}F) \cong G\circ E + G\circ F$
- **Idempotenz von +**
  - $E{+}E \cong E$
- **Hüllengesetze**
  - $(E^*)^* \cong E^*, \quad \emptyset^* \cong \epsilon, \quad \epsilon^* \cong \epsilon$
  - $E^+ \cong E\circ E^* \cong E^*\circ E, \quad E^* \cong \epsilon + E^+$

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  – Sei $w \in L((E+F)^*)$

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E{+}F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E{+}F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E+F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  - Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E+F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  - Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i
  - Also $w \in L((E^*F^*)^*)$

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E+F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  - Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i
  - Also $w \in L((E^*F^*)^*)$
- **Beweis benötigt keine Information über E und F**

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E+F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  - Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i
  - Also $w \in L((E^*F^*)^*)$
- **Beweis benötigt keine Information über E und F**
  - Man könnte genausogut konkrete Symbole verwenden

    $(E+F)^* \cong (E^*F^*)^*$ gilt weil $(\mathtt{a}+\mathtt{b})^* \cong (\mathtt{a}^*\mathtt{b}^*)^*$ gilt

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E+F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  - Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i
  - Also $w \in L((E^*F^*)^*)$
- **Beweis benötigt keine Information über E und F**
  - Man könnte genausogut konkrete Symbole verwenden

    $(E+F)^* \cong (E^*F^*)^*$ gilt weil $(\mathtt{a}+\mathtt{b})^* \cong (\mathtt{a}^*\mathtt{b}^*)^*$ gilt
- **Allgemeines Beweisprinzip**
  - E regulärer Ausdruck mit Metavariablen $F_1,..,F_m$ (für Sprachen $L_1,..,L_m$)

    C entsprechender Ausdruck mit Symbolen $\mathtt{a}_1,..\mathtt{a}_m$ statt der F_i

    $w \in L(E)$ ist zerlegbar in $w = w_1..w_k$ mit $w_i \in L_{j_i}$ g.d.w. $v{=}\mathtt{a}_{j_1}..\mathtt{a}_{j_k} \in L(C)$

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  – Sei $w \in L((E+F)^*)$
  – Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  – Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i
  – Also $w \in L((E^*F^*)^*)$

- **Beweis benötigt keine Information über E und F**
  – Man könnte genausogut konkrete Symbole verwenden
  $(E+F)^* \cong (E^*F^*)^*$ gilt weil $(\mathtt{a}+\mathtt{b})^* \cong (\mathtt{a}^*\mathtt{b}^*)^*$ gilt

- **Allgemeines Beweisprinzip**
  – E regulärer Ausdruck mit Metavariablen $F_1,..,F_m$ (für Sprachen $L_1,..,L_m$)
  C entsprechender Ausdruck mit Symbolen $\mathtt{a}_1..\mathtt{a}_m$ statt der F_i
  $w \in L(E)$ ist zerlegbar in $w = w_1..w_k$ mit $w_i \in L_{j_i}$ g.d.w. $v=\mathtt{a}_{j_1}..\mathtt{a}_{j_k} \in L(C)$
  Beweis durch strukturelle Induktion über Aufbau regulärer Ausdrücke

Beweismethodik für algebraische Gesetze

- **Beispiel: Nachweis von $(E+F)^* \cong (E^*F^*)^*$**
  - Sei $w \in L((E+F)^*)$
  - Dann $w = w_1..w_k$ mit $w_i \in L(E)$ oder $w_i \in L(F)$ für alle i
  - Dann $w = w_1..w_k$ mit $w_i \in L(E^*F^*)$ für alle i
  - Also $w \in L((E^*F^*)^*)$
- **Beweis benötigt keine Information über E und F**
  - Man könnte genausogut konkrete Symbole verwenden

    $(E+F)^* \cong (E^*F^*)^*$ gilt weil $(\mathtt{a}+\mathtt{b})^* \cong (\mathtt{a}^*\mathtt{b}^*)^*$ gilt
- **Allgemeines Beweisprinzip**
  - E regulärer Ausdruck mit Metavariablen $F_1,..,F_m$ (für Sprachen $L_1,..,L_m$)

    C entsprechender Ausdruck mit Symbolen $\mathtt{a}_1,..\mathtt{a}_m$ statt der F_i

    $w \in L(E)$ ist zerlegbar in $w = w_1..w_k$ mit $w_i \in L_{j_i}$ g.d.w. $v = \mathtt{a}_{j_1}..\mathtt{a}_{j_k} \in L(C)$

    Beweis durch strukturelle Induktion über Aufbau regulärer Ausdrücke
  - Prüfverfahren für $E \cong F$ ersetzt alle Metavariablen durch Symbole $a \in \Sigma$ und testet dann Gleichheit der konkreten Ausdrücke $\mapsto$ später

Umwandlung regulärer Ausdrücke in Automaten

Sprachen regulärer Ausdrücke sind endlich erkennbar

Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck E gibt es einen ϵ-NEA A mit

– A hat genau einen akzeptierenden Zustand q_f

– Der Startzustand von A ist in keinem $\delta_A(q,a)$ enthalten

– Für alle $a \in \Sigma$ ist $\delta_A(q_f,a) = \emptyset$

– $L(E) = L(A)$

Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck E gibt es einen ϵ-NEA A mit

– A hat genau einen akzeptierenden Zustand q_f

– Der Startzustand von A ist in keinem $\delta_A(q, a)$ enthalten

– Für alle $a \in \Sigma$ ist $\delta_A(q_f, a) = \emptyset$

– $L(E) = L(A)$

Beweis durch strukturelle Induktion über Aufbau der regulären Ausdrücke

Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck E gibt es einen ϵ-NEA A mit

– A hat genau einen akzeptierenden Zustand q_f

– Der Startzustand von A ist in keinem $\delta_A(q, a)$ enthalten

– Für alle $a \in \Sigma$ ist $\delta_A(q_f, a) = \emptyset$

– $L(E) = L(A)$

Beweis durch strukturelle Induktion über Aufbau der regulären Ausdrücke

- **Induktionsanfänge**

Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck E gibt es einen ϵ-NEA A mit

– A hat genau einen akzeptierenden Zustand q_f

– Der Startzustand von A ist in keinem $\delta_A(q,a)$ enthalten

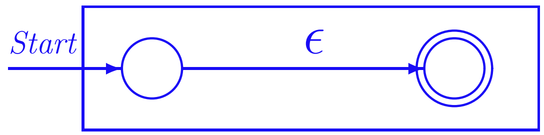
– Für alle $a \in \Sigma$ ist $\delta_A(q_f,a) = \emptyset$

– $L(E) = L(A)$

Beweis durch strukturelle Induktion über Aufbau der regulären Ausdrücke

- **Induktionsanfänge**

  – Für $E = \epsilon$ wähle $A =$



Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck E gibt es einen ϵ-NEA A mit

– A hat genau einen akzeptierenden Zustand q_f

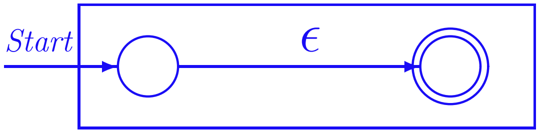
– Der Startzustand von A ist in keinem $\delta_A(q,a)$ enthalten

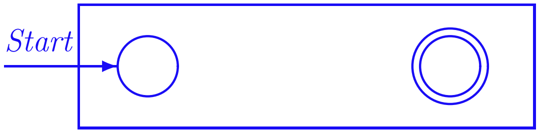
– Für alle $a \in \Sigma$ ist $\delta_A(q_f,a) = \emptyset$

– $L(E) = L(A)$

Beweis durch strukturelle Induktion über Aufbau der regulären Ausdrücke

- **Induktionsanfänge**

  – Für $E = \epsilon$ wähle $A =$ 

  – Für $E = \emptyset$ wähle $A =$ 

Umwandlung regulärer Ausdrücke in Automaten

Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck $\boldsymbol{E}$ gibt es einen $\boldsymbol{\epsilon}$-NEA $\boldsymbol{A}$ mit

– A hat genau einen akzeptierenden Zustand q_f

– Der Startzustand von A ist in keinem $\delta_A(q,a)$ enthalten

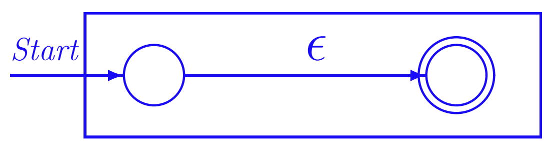
– Für alle $a \in \Sigma$ ist $\delta_A(q_f,a) = \emptyset$

– $L(E) = L(A)$

Beweis durch strukturelle Induktion über Aufbau der regulären Ausdrücke

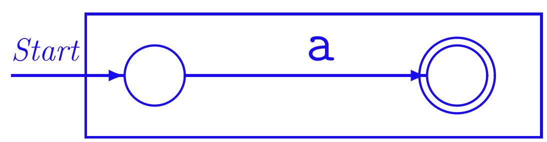
- **Induktionsanfänge**

  – Für $E = \epsilon$ wähle $A =$



  – Für $E = \emptyset$ wähle $A =$

  – Für $E = \mathtt{a}$ wähle $A =$



Sprachen regulärer Ausdrücke sind endlich erkennbar

Für jeden regulären Ausdruck E gibt es einen ϵ-NEA A mit

– A hat genau einen akzeptierenden Zustand q_f

– Der Startzustand von A ist in keinem $\delta_A(q,a)$ enthalten

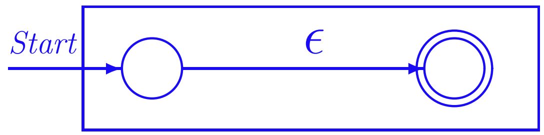
– Für alle $a \in \Sigma$ ist $\delta_A(q_f,a) = \emptyset$

– $L(E) = L(A)$

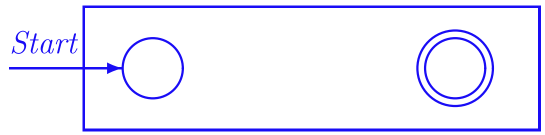
Beweis durch strukturelle Induktion über Aufbau der regulären Ausdrücke

- **Induktionsanfänge**

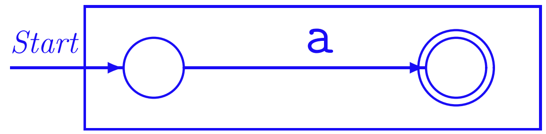
– Für $E = \epsilon$ wähle $A =$



– Für $E = \emptyset$ wähle $A =$



– Für $E =$ `a` wähle $A =$



– **Korrektheit offensichtlich**, da jeweils maximal ein Zustandsübergang

Umwandlung regulärer Ausdrücke in Automaten

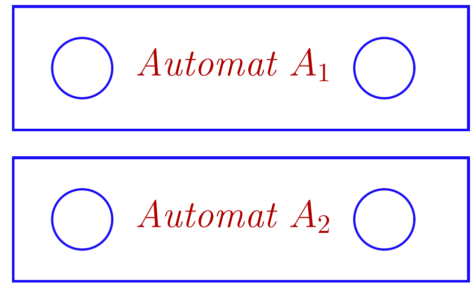
- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**

Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien $\boldsymbol{A_1}$ und $\boldsymbol{A_2}$ $\boldsymbol{\epsilon}$-NEAs für $\boldsymbol{E_1}$ und $\boldsymbol{E_2}$**

- **Induktionsschritt**

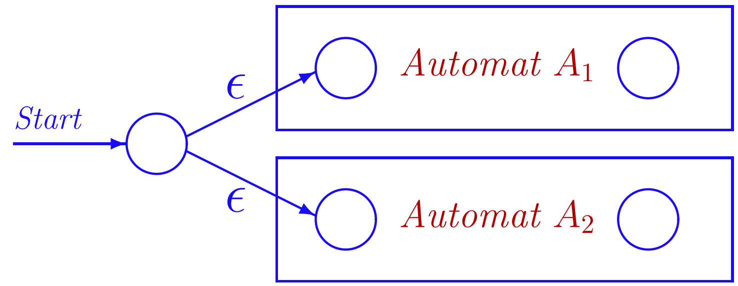
  – Für $E = E_1 + E_2$ wähle $\quad A =$



Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien $\boldsymbol{A_1}$ und $\boldsymbol{A_2}$ $\boldsymbol{\epsilon}$-NEAs für $\boldsymbol{E_1}$ und $\boldsymbol{E_2}$**

- **Induktionsschritt**

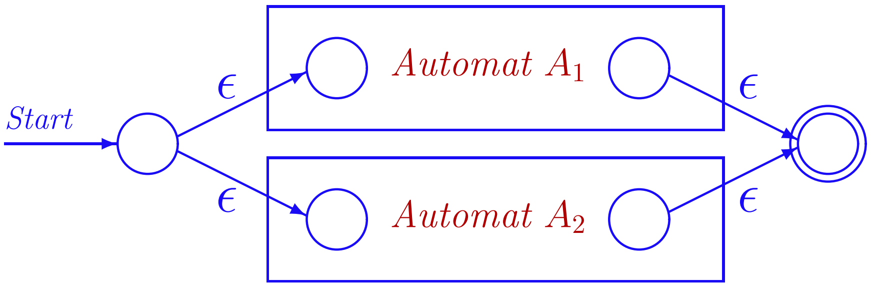
  – Für $E = E_1 + E_2$ wähle $A =$ 

Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**

- **Induktionsschritt**

  – Für $E = E_1 + E_2$ wähle $A =$

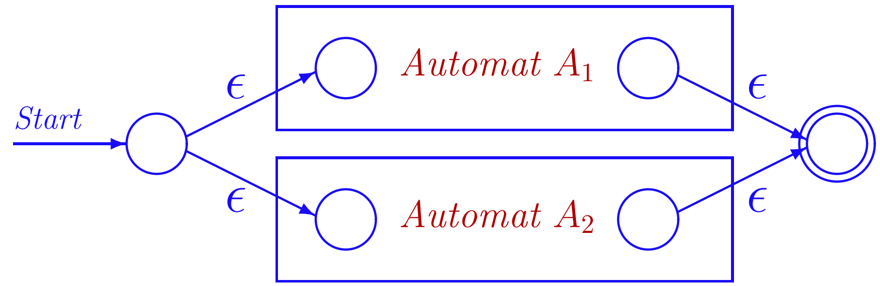


Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**

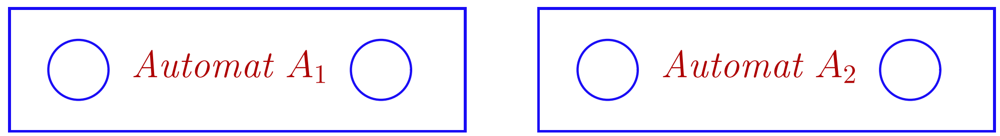
- **Induktionsschritt**

  – Für $E = E_1 + E_2$ wähle $A =$



  – Für $E = E_1 \circ E_2$ wähle

$A =$

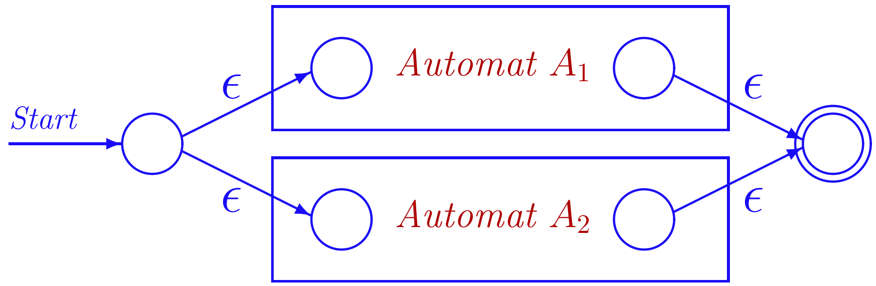


Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**

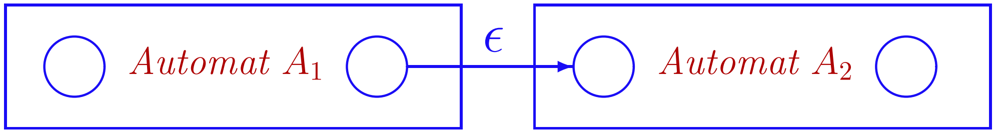
- **Induktionsschritt**

  – Für $E = E_1 + E_2$ wähle $A =$



  – Für $E = E_1 \circ E_2$ wähle

$A =$

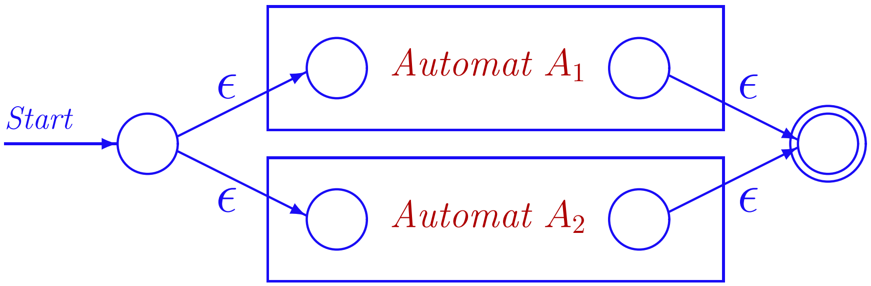


Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien $\boldsymbol{A_1}$ und $\boldsymbol{A_2}$ $\boldsymbol{\epsilon}$-NEAs für $\boldsymbol{E_1}$ und $\boldsymbol{E_2}$**

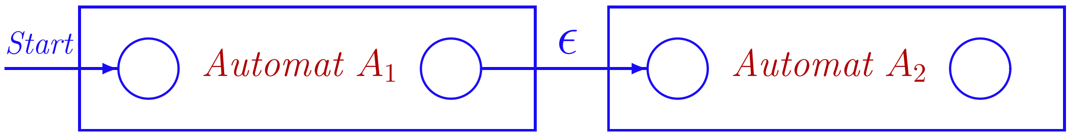
- **Induktionsschritt**

  – Für $E = E_1{+}E_2$ wähle $A =$



  – Für $E = E_1 \circ E_2$ wähle

$A =$

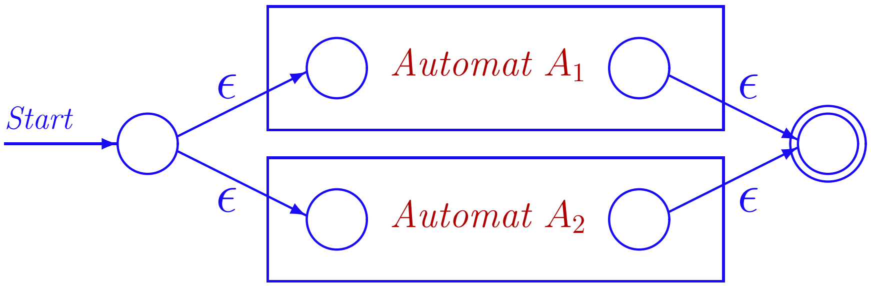


Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**

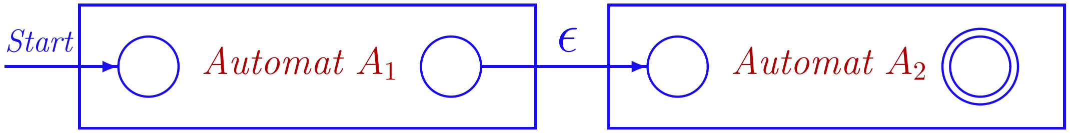
- **Induktionsschritt**

  – Für $E = E_1 + E_2$ wähle $A =$



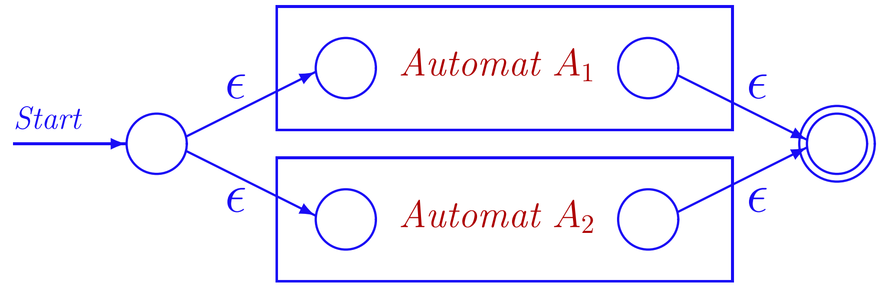
  – Für $E = E_1 \circ E_2$ wähle

$A =$



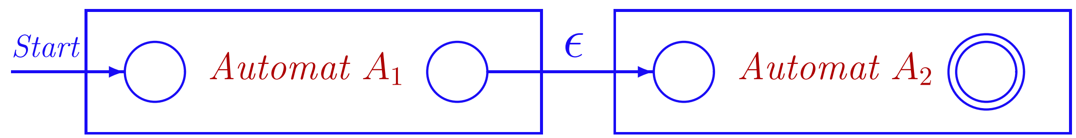
Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**
- **Induktionsschritt**
  - Für $E = E_1 + E_2$ wähle $A =$

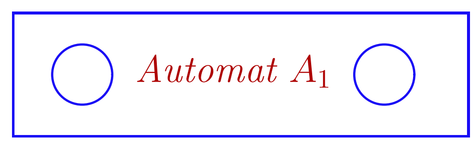


  - Für $E = E_1 \circ E_2$ wähle

$A =$



  - Für $E = E_1^*$ wähle $A =$

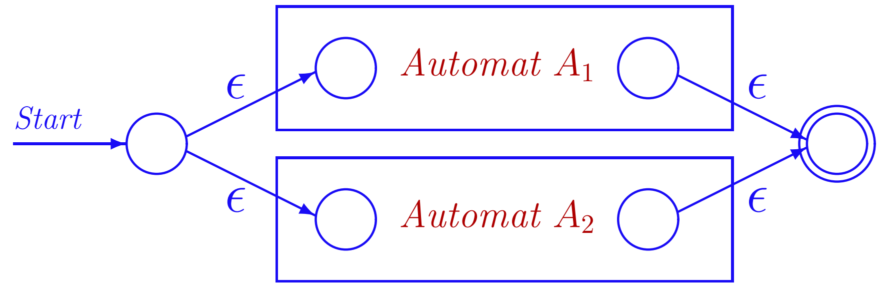


Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien A_1 und A_2 ϵ-NEAs für E_1 und E_2**

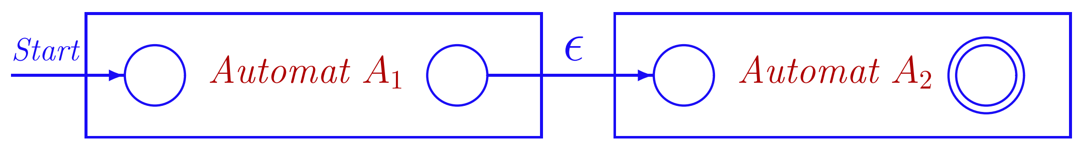
- **Induktionsschritt**

  – Für $E = E_1 + E_2$ wähle $A =$

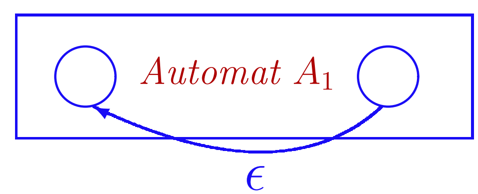


  – Für $E = E_1 \circ E_2$ wähle

$A =$

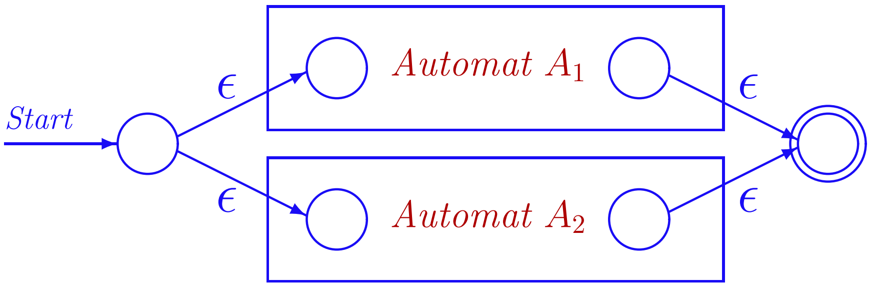


  – Für $E = E_1^*$ wähle $A =$



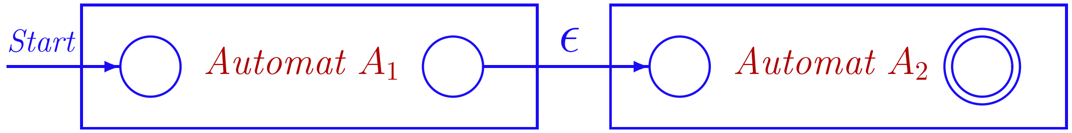
Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien $\boldsymbol{A_1}$ und $\boldsymbol{A_2}$ $\boldsymbol{\epsilon}$-NEAs für $\boldsymbol{E_1}$ und $\boldsymbol{E_2}$**
- **Induktionsschritt**
  - Für $E = E_1 + E_2$ wähle $\quad A =$

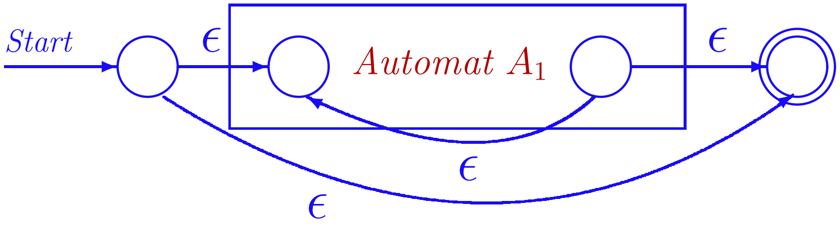


  - Für $E = E_1 \circ E_2$ wähle

$A =$



  - Für $E = E_1^*$ wähle $\quad A =$

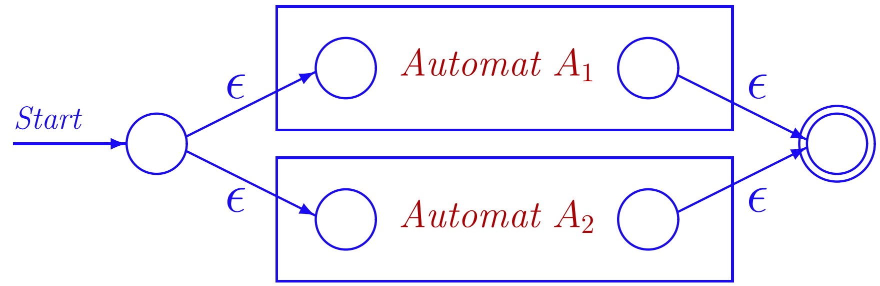


Umwandlung regulärer Ausdrücke in Automaten

- **Induktionsannahme: seien $\boldsymbol{A_1}$ und $\boldsymbol{A_2}$ $\boldsymbol{\epsilon}$-NEAs für $\boldsymbol{E_1}$ und $\boldsymbol{E_2}$**

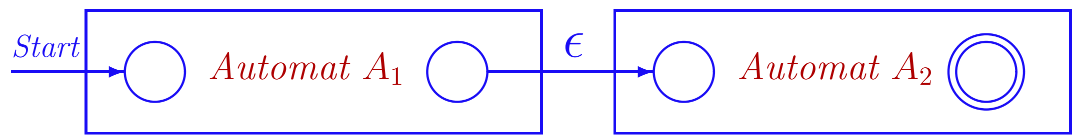
- **Induktionsschritt**

  – Für $E = E_1 + E_2$ wähle $A =$

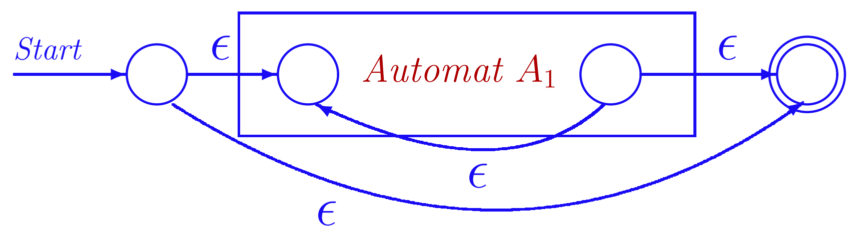


  – Für $E = E_1 \circ E_2$ wähle

$A =$



  – Für $E = E_1^*$ wähle $A =$



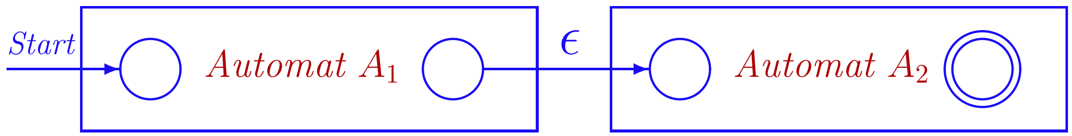
  – Für $E = (E_1)$ wähle $A = A_1$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  - Es ist $L((E_1)) = L(E_1) = L(A_1) = L(A)$

Korrektheit der Umwandlungen

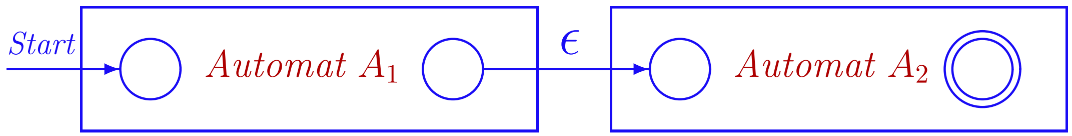
- **Klammern ändern nichts**
  - Es ist $L((E_1)) = L(E_1) = L(A_1) = L(A)$
- **Verkettung ist Verschaltung von Automaten**



  - Es gilt $w \in L(E_1 \circ E_2)$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  - Es ist $L(\mathbf{(}E_1\mathbf{)}) = L(E_1) = L(A_1) = L(A)$
- **Verkettung ist Verschaltung von Automaten**

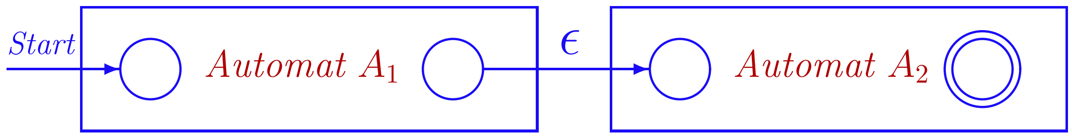


  - Es gilt $w \in L(E_1 \circ E_2)$

    $\Rightarrow\ w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  - Es ist $L(\mathbf{(}E_1\mathbf{)}) = L(E_1) = L(A_1) = L(A)$
- **Verkettung ist Verschaltung von Automaten**



  - Es gilt $w \in L(E_1 \circ E_2)$

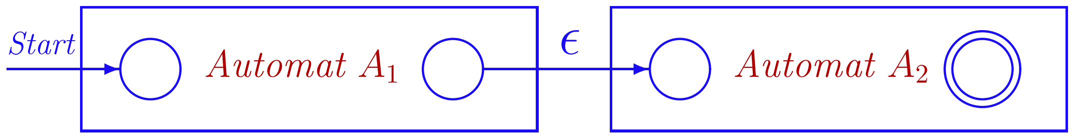
    $\Rightarrow\ w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$

    $\Rightarrow\ \exists u \in L(A_1). \exists v \in L(A_2). w = uv$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  – Es ist $L(\mathbf{(}E_1\mathbf{)}) = L(E_1) = L(A_1) = L(A)$

- **Verkettung ist Verschaltung von Automaten**



  – Es gilt $w \in L(E_1 \circ E_2)$

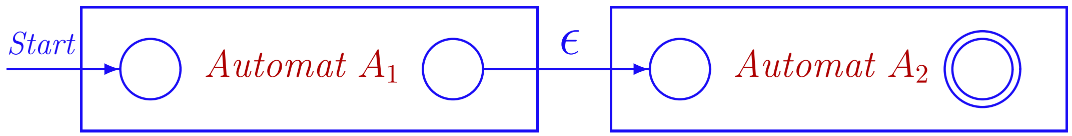
$$\Rightarrow\ w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$$

$$\Rightarrow\ \exists u \in L(A_1).\exists v \in L(A_2).w = uv$$

$$\Rightarrow\ \exists u, v \in \Sigma^*.w = uv \wedge q_{f,1} \in \hat{\delta}_1(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}_2(q_{0,2}, v)$$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  - Es ist $L((E_1)) = L(E_1) = L(A_1) = L(A)$
- **Verkettung ist Verschaltung von Automaten**



  - Es gilt $w \in L(E_1 \circ E_2)$

$$\Rightarrow\; w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$$

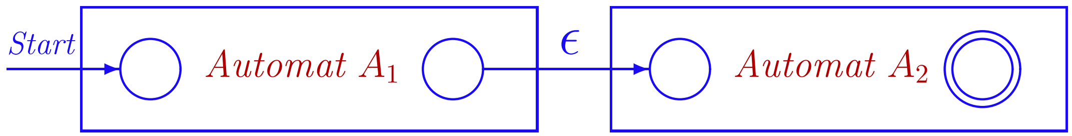
$$\Rightarrow\; \exists u \in L(A_1).\exists v \in L(A_2).w = uv$$

$$\Rightarrow\; \exists u, v \in \Sigma^*.w = uv \wedge q_{f,1} \in \hat{\delta}_1(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}_2(q_{0,2}, v)$$

$$\Rightarrow\; \exists u, v \in \Sigma^*.w = uv \wedge q_{0,2} \in \hat{\delta}(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}(q_{0,2}, v) \qquad (q_{0,2} \in \epsilon\text{-Hülle}(q_{f,1}))$$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  - Es ist $L((E_1)) = L(E_1) = L(A_1) = L(A)$
- **Verkettung ist Verschaltung von Automaten**



  - Es gilt $w \in L(E_1 \circ E_2)$

$\Rightarrow\ w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$

$\Rightarrow\ \exists u \in L(A_1). \exists v \in L(A_2). w = uv$

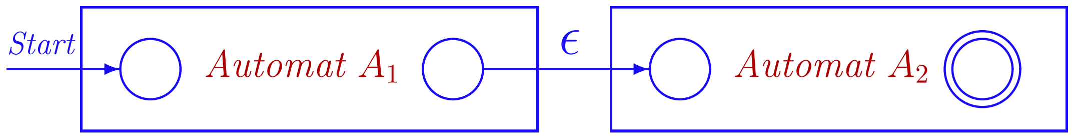
$\Rightarrow\ \exists u, v \in \Sigma^*. w = uv \wedge q_{f,1} \in \hat{\delta}_1(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}_2(q_{0,2}, v)$

$\Rightarrow\ \exists u, v \in \Sigma^*. w = uv \wedge q_{0,2} \in \hat{\delta}(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}(q_{0,2}, v)$ $\quad (q_{0,2} \in \epsilon\text{-Hülle}(q_{f,1}))$

$\Rightarrow\ q_{f,2} \in \hat{\delta}(q_{0,1}, w)$ $\quad$ (Definition $\hat{\delta}$)

- **Klammern ändern nichts**
  – Es ist $L((E_1)) = L(E_1) = L(A_1) = L(A)$

- **Verkettung ist Verschaltung von Automaten**



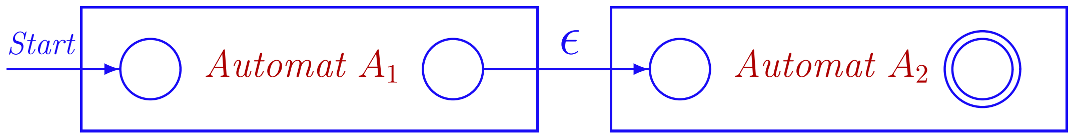
  – Es gilt $w \in L(E_1 \circ E_2)$

$$\Rightarrow\ w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$$
$$\Rightarrow\ \exists u \in L(A_1).\exists v \in L(A_2).w = uv$$
$$\Rightarrow\ \exists u, v \in \Sigma^*.w = uv \wedge q_{f,1} \in \hat{\delta}_1(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}_2(q_{0,2}, v)$$
$$\Rightarrow\ \exists u, v \in \Sigma^*.w = uv \wedge q_{0,2} \in \hat{\delta}(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}(q_{0,2}, v) \qquad (q_{0,2} \in \epsilon\text{-Hülle}(q_{f,1}))$$
$$\Rightarrow\ q_{f,2} \in \hat{\delta}(q_{0,1}, w) \qquad (\text{Definition } \hat{\delta})$$
$$\Rightarrow\ w \in L(A)$$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  - Es ist $L((E_1)) = L(E_1) = L(A_1) = L(A)$

- **Verkettung ist Verschaltung von Automaten**



  - Es gilt $w \in L(E_1 \circ E_2)$

$$\Rightarrow\ w \in L(E_1) \circ L(E_2) = L(A_1) \circ L(A_2)$$

$$\Rightarrow\ \exists u \in L(A_1). \exists v \in L(A_2). w = uv$$

$$\Rightarrow\ \exists u, v \in \Sigma^*. w = uv \wedge q_{f,1} \in \hat{\delta}_1(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}_2(q_{0,2}, v)$$

$$\Rightarrow\ \exists u, v \in \Sigma^*. w = uv \wedge q_{0,2} \in \hat{\delta}(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}(q_{0,2}, v) \qquad (q_{0,2} \in \epsilon\text{-Hülle}(q_{f,1}))$$

$$\Rightarrow\ q_{f,2} \in \hat{\delta}(q_{0,1}, w) \qquad (\text{Definition } \hat{\delta})$$

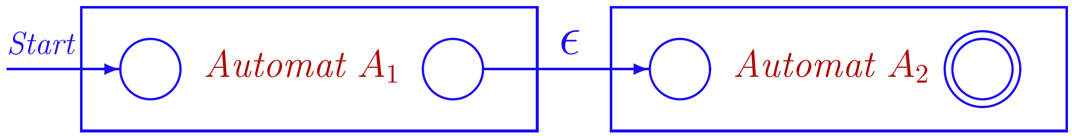
$$\Rightarrow\ w \in L(A)$$

  - Argument ist umkehrbar, also $w \in L(A) \ \Rightarrow\ w \in L(E_1 \circ E_2)$

Korrektheit der Umwandlungen

- **Klammern ändern nichts**
  – Es ist $L(\mathbf{(}E_1\mathbf{)}) = L(E_1) = L(A_1) = L(A)$

- **Verkettung ist Verschaltung von Automaten**



  – Es gilt $w \in L(E_1 \circ E_2)$

$$\Rightarrow\ w \in L(E_1)\circ L(E_2) = L(A_1)\circ L(A_2)$$
$$\Rightarrow\ \exists u \in L(A_1).\exists v \in L(A_2).w = uv$$
$$\Rightarrow\ \exists u, v \in \Sigma^*.w = uv \wedge q_{f,1} \in \hat{\delta}_1(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}_2(q_{0,2}, v)$$
$$\Rightarrow\ \exists u, v \in \Sigma^*.w = uv \wedge q_{0,2} \in \hat{\delta}(q_{0,1}, u) \wedge q_{f,2} \in \hat{\delta}(q_{0,2}, v) \qquad (q_{0,2} \in \epsilon\text{-Hülle}(q_{f,1}))$$
$$\Rightarrow\ q_{f,2} \in \hat{\delta}(q_{0,1}, w) \qquad (\text{Definition } \hat{\delta})$$
$$\Rightarrow\ w \in L(A)$$

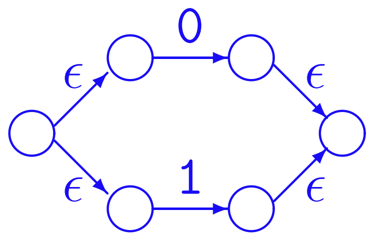
  – Argument ist umkehrbar, also $w \in L(A) \ \Rightarrow\ w \in L(E_1 \circ E_2)$

Sternbildung und Vereinigung in Übungen?

Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für (0+1)*1(0+1)

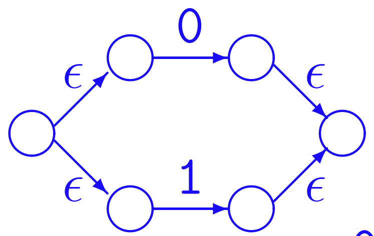
- **Teilautomat für (0+1)**



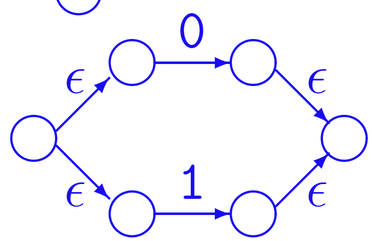
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

- **Teilautomat für** `(0+1)`



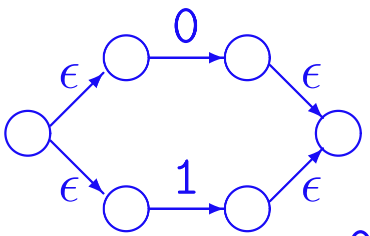
- **Teilautomat für** `(0+1)*`



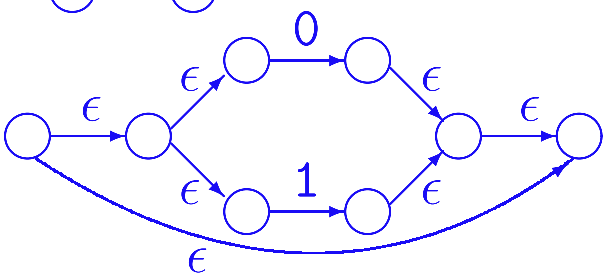
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für (0+1)*1(0+1)

- **Teilautomat für (0+1)**



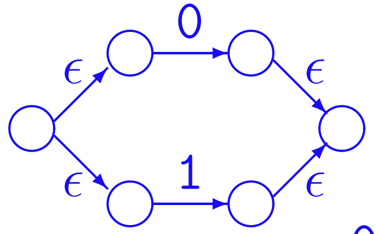
- **Teilautomat für (0+1)***



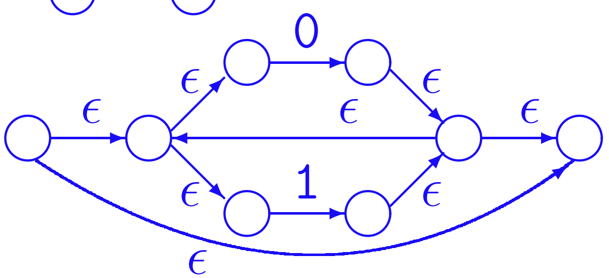
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

- **Teilautomat für** `(0+1)`



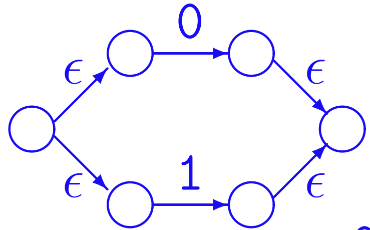
- **Teilautomat für** `(0+1)*`



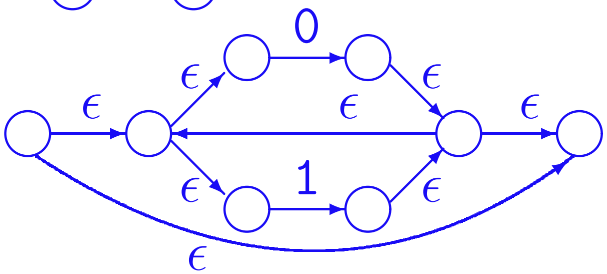
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für (0+1)*1(0+1)

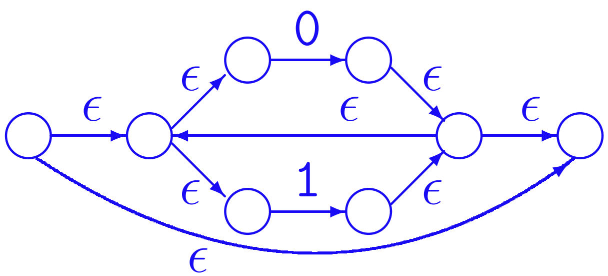
- **Teilautomat für** (0+1)



- **Teilautomat für** (0+1)*



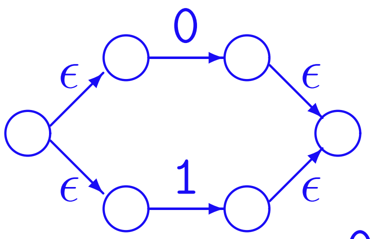
- **Automat für** (0+1)*1(0+1)



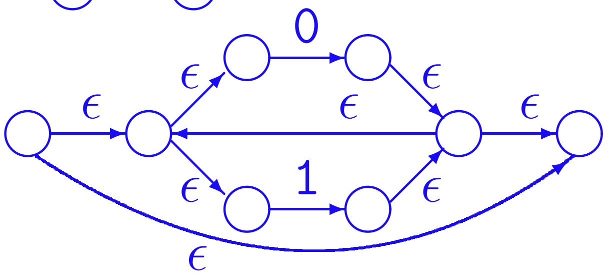
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für (0+1)*1(0+1)

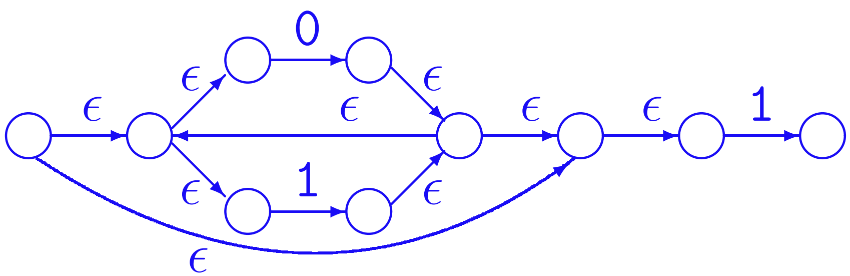
- **Teilautomat für** (0+1)



- **Teilautomat für** (0+1)*



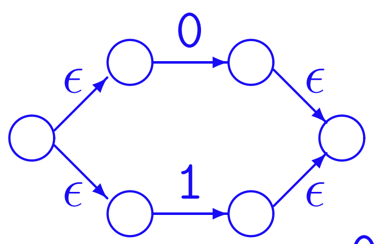
- **Automat für** (0+1)*1(0+1)



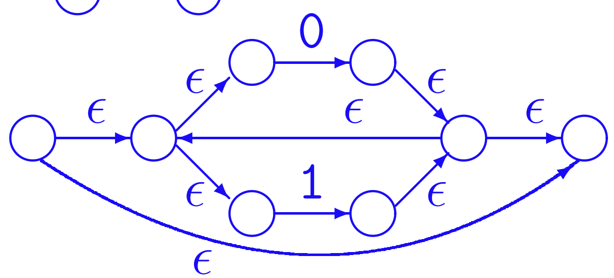
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

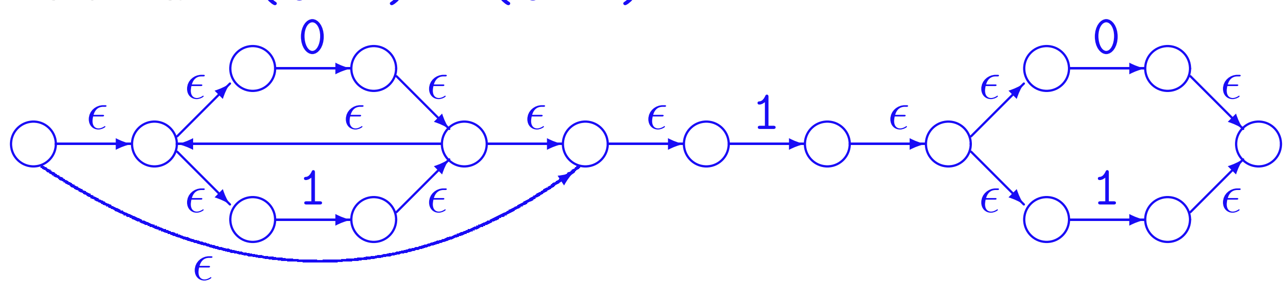
- **Teilautomat für** `(0+1)`



- **Teilautomat für** `(0+1)*`



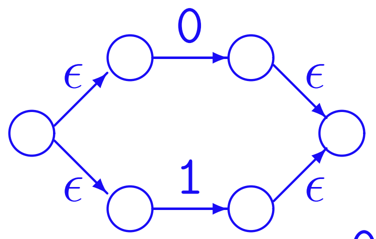
- **Automat für** `(0+1)*1(0+1)`



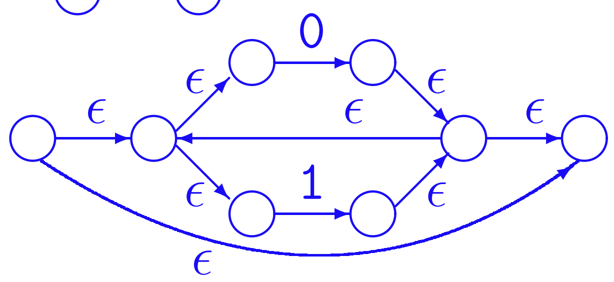
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

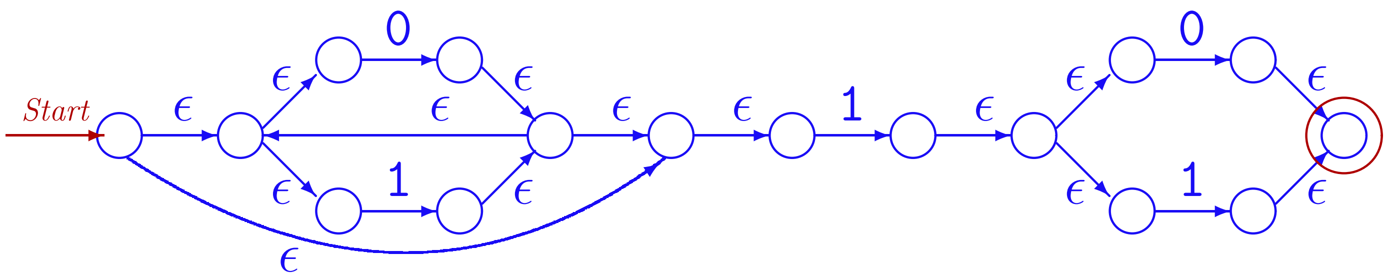
- **Teilautomat für** `(0+1)`



- **Teilautomat für** `(0+1)*`



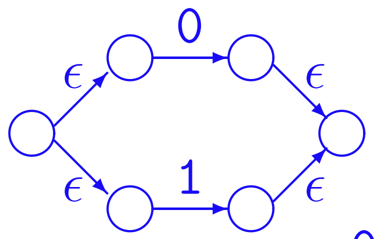
- **Automat für** `(0+1)*1(0+1)`



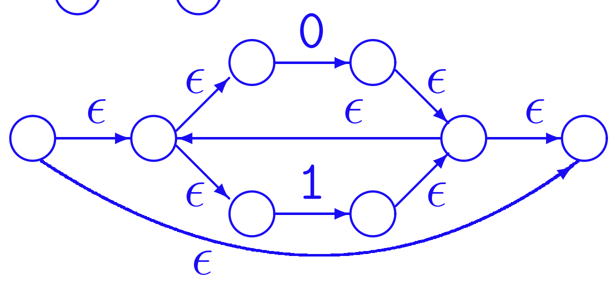
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für (0+1)*1(0+1)

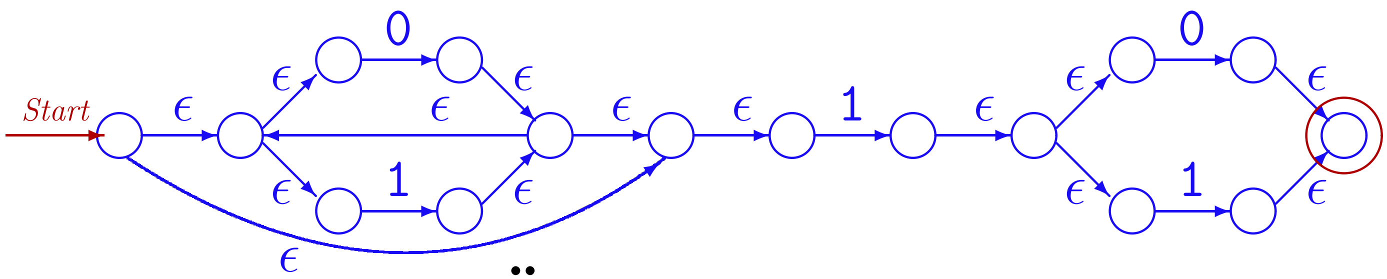
- **Teilautomat für** (0+1)



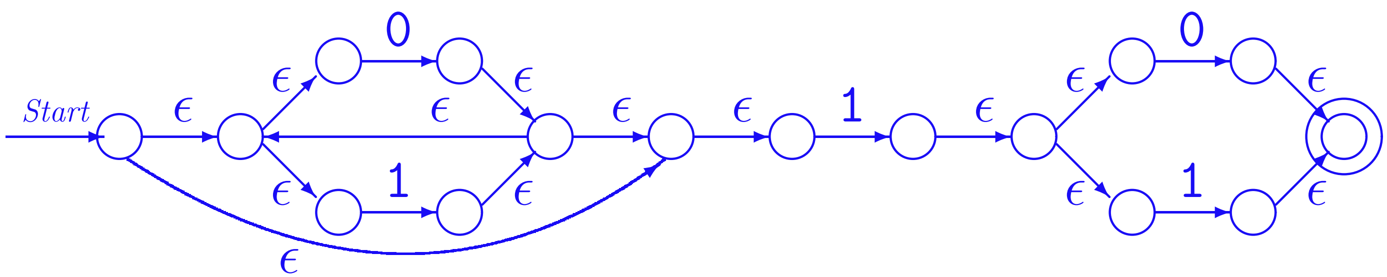
- **Teilautomat für** (0+1)*



- **Automat für** (0+1)*1(0+1)



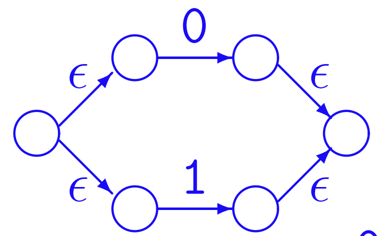
- **Elimination von** ϵ**-Übergängen**



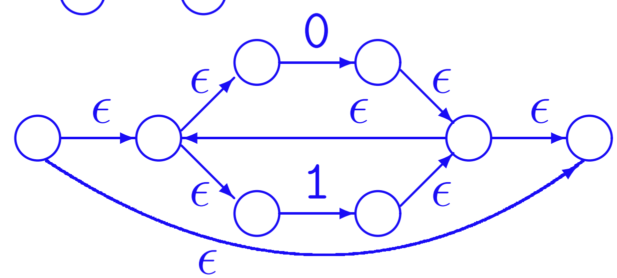
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

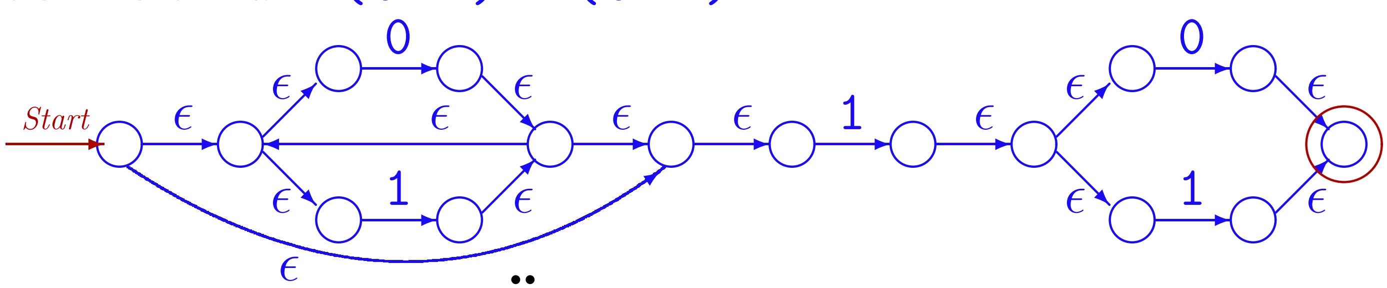
- **Teilautomat für** `(0+1)`



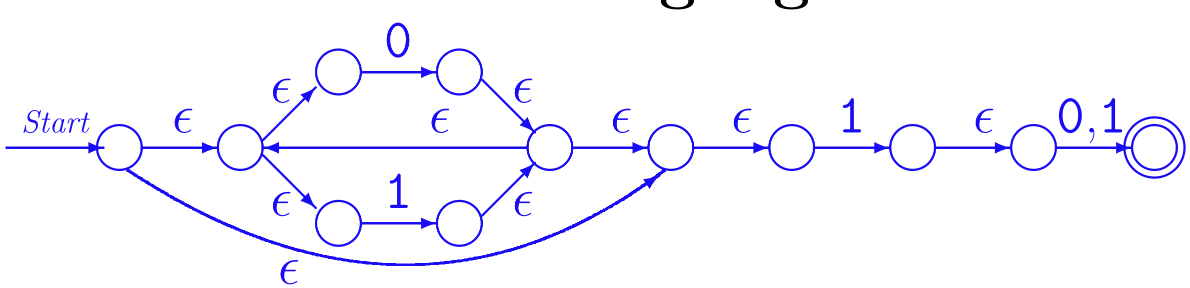
- **Teilautomat für** `(0+1)*`



- **Automat für** `(0+1)*1(0+1)`



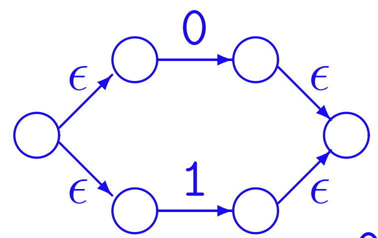
- **Elimination von ϵ-Übergängen**



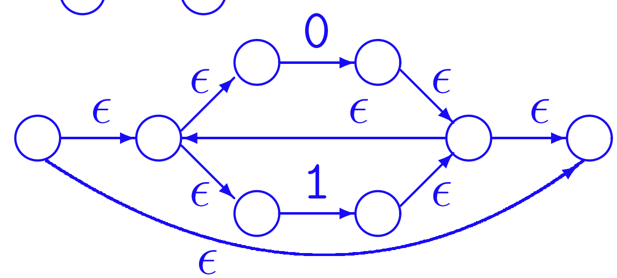
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für (0+1)*1(0+1)

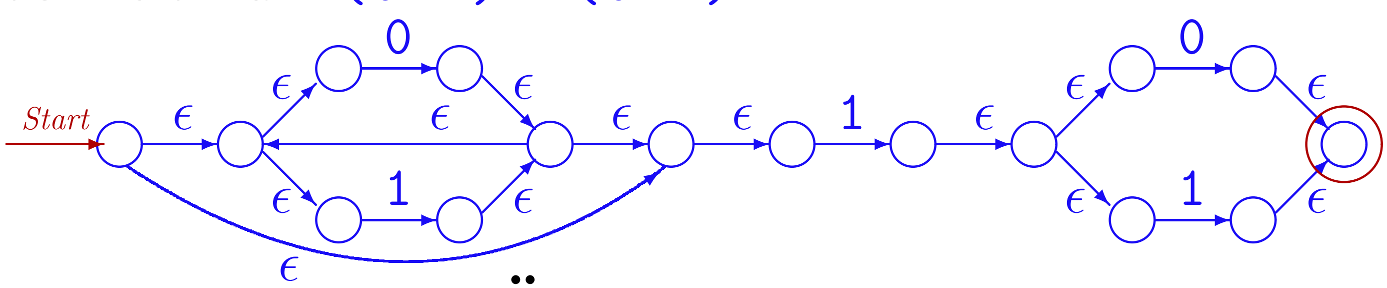
- **Teilautomat für (0+1)**



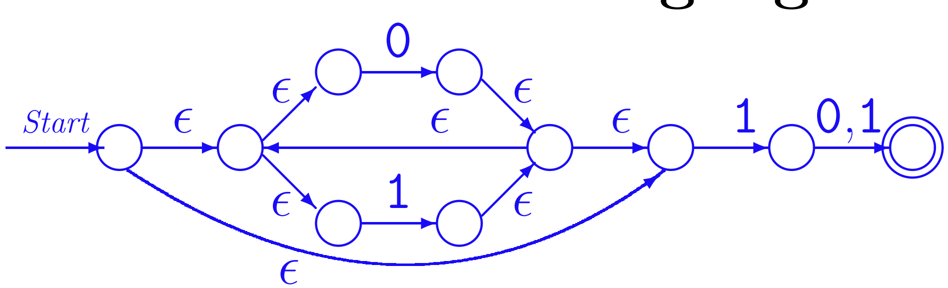
- **Teilautomat für (0+1)***



- **Automat für (0+1)*1(0+1)**



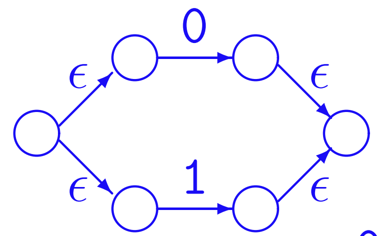
- **Elimination von ϵ-Übergängen**



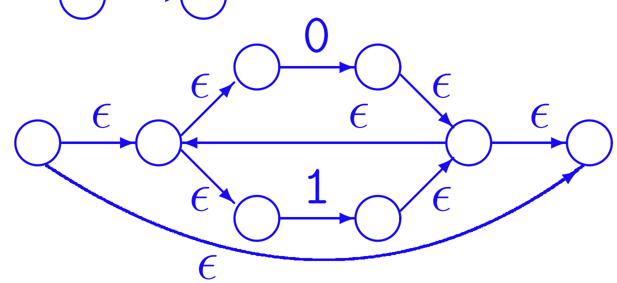
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

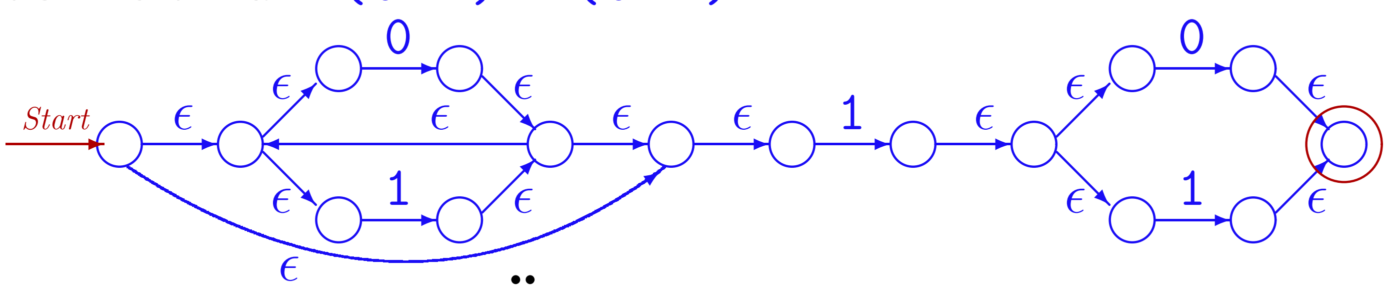
- **Teilautomat für `(0+1)`**



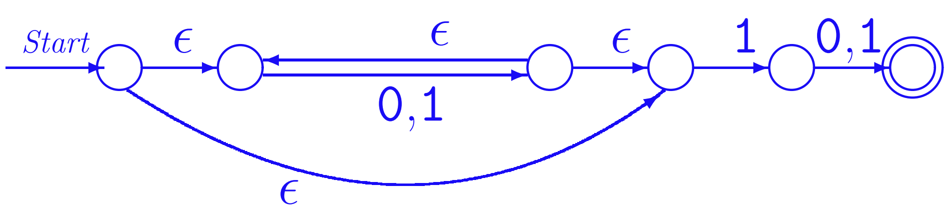
- **Teilautomat für `(0+1)*`**



- **Automat für `(0+1)*1(0+1)`**



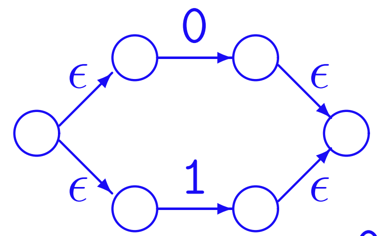
- **Elimination von ϵ-Übergängen**



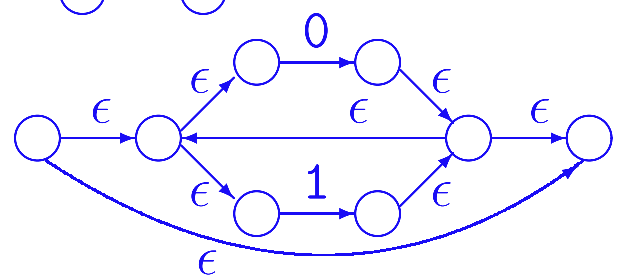
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

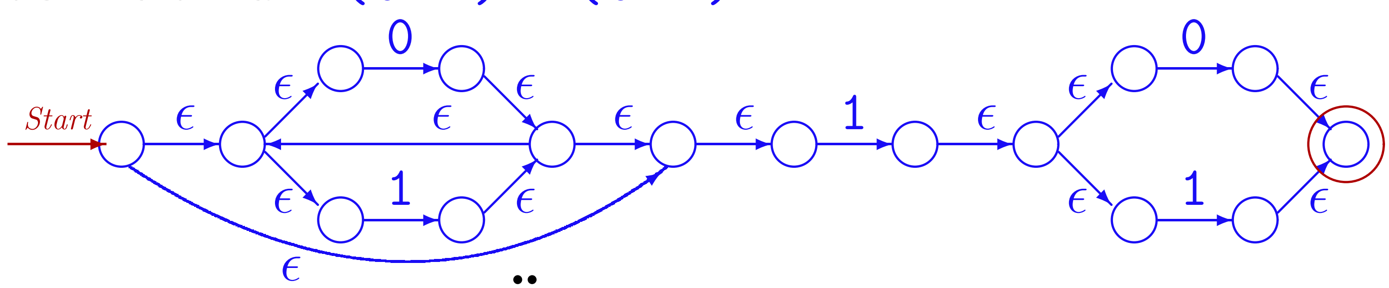
- **Teilautomat für `(0+1)`**



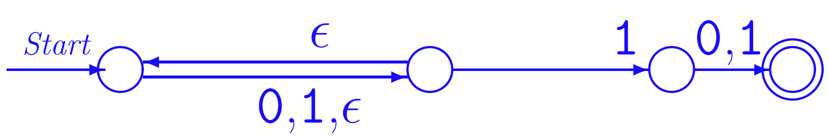
- **Teilautomat für `(0+1)*`**



- **Automat für `(0+1)*1(0+1)`**



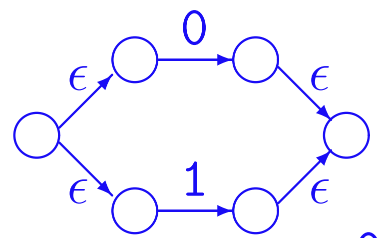
- **Elimination von ϵ-Übergängen**



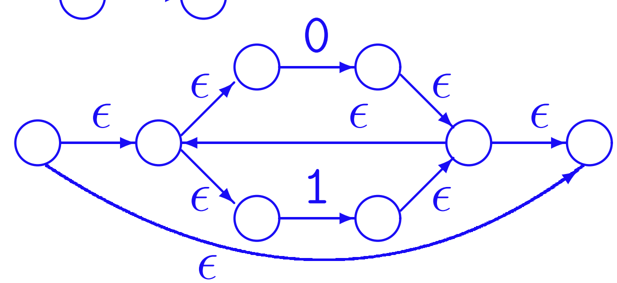
Umwandlung regulärer Ausdrücke am Beispiel

Konstruiere endlichen Automaten für `(0+1)*1(0+1)`

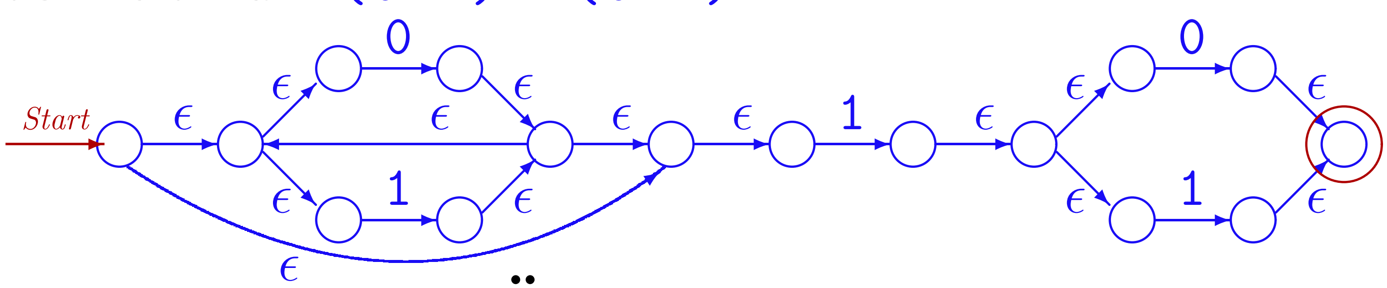
- **Teilautomat für** `(0+1)`



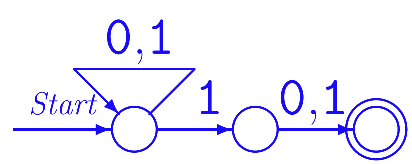
- **Teilautomat für** `(0+1)*`



- **Automat für** `(0+1)*1(0+1)`



- **Elimination von ϵ-Übergängen**



Umwandlung von Automaten in Reguläre Ausdrücke

- **Gegeben** $\boldsymbol{A} = (\{q_1, ..., q_n\}, \Sigma, \delta, q_1, \{q_{f_1}, .., q_{f_m}\})$

Umwandlung von Automaten in Reguläre Ausdrücke

- **Gegeben $A = (\{q_1, ..., q_n\},\ \Sigma,\ \delta,\ q_1,\ \{q_{f_1}, .., q_{f_m}\})$**
- **Definiere Ausdrücke für Pfade durch A**
  - $R_{ij}^{(k)}$: Regulärer Ausdruck für Menge der Worte w mit $\hat{\delta}(q_i, w) = q_j$
    so daß für alle $\epsilon{\neq}v{\sqsubseteq}w$ $(v{\neq}w)$ gilt: $\hat{\delta}(q_i, v){=}q_n \Rightarrow n{\leq}k$
    (Abarbeitung von w berührt keinen Zustand größer als k)

Umwandlung von Automaten in Reguläre Ausdrücke

- **Gegeben $A = (\{q_1, ..., q_n\}, \Sigma, \delta, q_1, \{q_{f_1}, .., q_{f_m}\})$**
- **Definiere Ausdrücke für Pfade durch A**
  - $R_{ij}^{(k)}$: Regulärer Ausdruck für Menge der Worte w mit $\hat{\delta}(q_i, w) = q_j$
    so daß für alle $\epsilon{\neq}v{\sqsubseteq}w$ $(v{\neq}w)$ gilt: $\hat{\delta}(q_i, v){=}q_n \Rightarrow n{\leq}k$
    (Abarbeitung von w berührt keinen Zustand größer als k)
- **Setze die $R_{ij}^{(k)}$ zu Ausdruck für $L(A)$ zusammen**
  - Per Definition ist $R_{ij}^{(n)}$ ein Ausdruck für Worte w mit $\hat{\delta}(q_i, w) = q_j$

Umwandlung von Automaten in Reguläre Ausdrücke

- **Gegeben $A = (\{q_1, ..., q_n\}, \Sigma, \delta, q_1, \{q_{f_1}, .., q_{f_m}\})$**
- **Definiere Ausdrücke für Pfade durch A**
  - $R_{ij}^{(k)}$: Regulärer Ausdruck für Menge der Worte w mit $\hat{\delta}(q_i, w) = q_j$
    so daß für alle $\epsilon{\neq}v{\sqsubseteq}w$ $(v{\neq}w)$ gilt: $\hat{\delta}(q_i, v){=}q_n \;\Rightarrow\; n{\leq}k$
    (Abarbeitung von w berührt keinen Zustand größer als k)
- **Setze die $R_{ij}^{(k)}$ zu Ausdruck für $L(A)$ zusammen**
  - Per Definition ist $R_{ij}^{(n)}$ ein Ausdruck für Worte w mit $\hat{\delta}(q_i, w) = q_j$
  - Setze $R = R_{1f_1}^{(n)} + ... + R_{1f_m}^{(n)}$

Umwandlung von Automaten in Reguläre Ausdrücke

- **Gegeben $A = (\{q_1,...,q_n\},\ \Sigma,\ \delta,\ q_1,\ \{q_{f_1},..,q_{f_m}\})$**
- **Definiere Ausdrücke für Pfade durch A**
  - $R_{ij}^{(k)}$: Regulärer Ausdruck für Menge der Worte w mit $\hat{\delta}(q_i,w) = q_j$
    so daß für alle $\epsilon{\neq}v{\sqsubseteq}w$ $(v{\neq}w)$ gilt: $\hat{\delta}(q_i,v){=}q_n \Rightarrow n{\leq}k$
    (Abarbeitung von w berührt keinen Zustand größer als k)
- **Setze die $R_{ij}^{(k)}$ zu Ausdruck für $L(A)$ zusammen**
  - Per Definition ist $R_{ij}^{(n)}$ ein Ausdruck für Worte w mit $\hat{\delta}(q_i,w) = q_j$
  - Setze $R = R_{1f_1}^{(n)} + ... + R_{1f_m}^{(n)}$
  - Dann gilt $L(R)$
    $= \bigcup_{j=1}^{m}\{w \in \Sigma^* \mid \hat{\delta}(q_1,w) = q_{f_j}\}$

UMWANDLUNG VON AUTOMATEN IN REGULÄRE AUSDRÜCKE

- **Gegeben** $\boldsymbol{A = (\{q_1, ..., q_n\}, \Sigma, \delta, q_1, \{q_{f_1}, .., q_{f_m}\})}$

- **Definiere Ausdrücke für Pfade durch** $\boldsymbol{A}$
  – $\boldsymbol{R_{ij}^{(k)}}$: Regulärer Ausdruck für Menge der Worte w mit $\hat{\delta}(q_i, w) = q_j$
  so daß für alle $\epsilon \neq v \sqsubseteq w$ $(v \neq w)$ gilt: $\hat{\delta}(q_i, v) = q_n \Rightarrow n \leq k$
  (Abarbeitung von w berührt keinen Zustand größer als k)

- **Setze die** $\boldsymbol{R_{ij}^{(k)}}$ **zu Ausdruck für** $\boldsymbol{L(A)}$ **zusammen**
  – Per Definition ist $R_{ij}^{(n)}$ ein Ausdruck für Worte w mit $\hat{\delta}(q_i, w) = q_j$
  – Setze $\boldsymbol{R = R_{1f_1}^{(n)} + ... + R_{1f_m}^{(n)}}$
  – Dann gilt $\boldsymbol{L(R)}$

$$= \bigcup_{j=1}^{m} \{w \in \Sigma^* \mid \hat{\delta}(q_1, w) = q_{f_j}\}$$
$$= \{w \in \Sigma^* \mid \exists q \in \{q_{f_1}, .., q_{f_m}\}.\ \hat{\delta}(q_1, w) = q\}$$

Umwandlung von Automaten in Reguläre Ausdrücke

- **Gegeben $A = (\{q_1, ..., q_n\},\ \Sigma,\ \delta,\ q_1,\ \{q_{f_1}, .., q_{f_m}\})$**
- **Definiere Ausdrücke für Pfade durch A**
  - $R_{ij}^{(k)}$: Regulärer Ausdruck für Menge der Worte w mit $\hat{\delta}(q_i, w) = q_j$

    so daß für alle $\epsilon{\neq}v{\sqsubseteq}w$ $(v{\neq}w)$ gilt: $\hat{\delta}(q_i, v){=}q_n \;\Rightarrow\; n{\leq}k$

    (Abarbeitung von w berührt keinen Zustand größer als k)
- **Setze die $R_{ij}^{(k)}$ zu Ausdruck für $L(A)$ zusammen**
  - Per Definition ist $R_{ij}^{(n)}$ ein Ausdruck für Worte w mit $\hat{\delta}(q_i, w) = q_j$
  - Setze $R = R_{1f_1}^{(n)} + ... + R_{1f_m}^{(n)}$
  - Dann gilt $L(R)$

    $= \bigcup_{j=1}^{m} \{w \in \Sigma^* \mid \hat{\delta}(q_1, w) = q_{f_j}\}$

    $= \{w \in \Sigma^* \mid \exists q \in \{q_{f_1}, .., q_{f_m}\}.\ \hat{\delta}(q_1, w) = q\}$

    $= L(A)$

Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

- **Basisfall** $\boldsymbol{R_{ij}^{0}}$**:** Pfad darf zwischendurch keine Zustände berühren

Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

- **Basisfall $\boldsymbol{R_{ij}^0}$:** Pfad darf zwischendurch keine Zustände berühren
  - Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R_{ii}^0)$

Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

- **Basisfall R_{ij}^0:** Pfad darf zwischendurch keine Zustände berühren
  – Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R_{ii}^0)$
  – Pfadlänge 1: $\{a \in \Sigma \mid \delta(q_i, a) = q_j\} \subseteq L(R_{ij}^0)$

Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

- **Basisfall $\boldsymbol{R_{ij}^0}$:** Pfad darf zwischendurch keine Zustände berühren
  - Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R_{ii}^0)$
  - Pfadlänge 1: $\{a \in \Sigma \mid \delta(q_i, a) = q_j\} \subseteq L(R_{ij}^0)$
  - Ergebnis: $\boldsymbol{R_{ii}^0} = \epsilon + \mathtt{a}_1 + \ldots + \mathtt{a}_k$, $\boldsymbol{R_{ij}^0} = \emptyset + \mathtt{a}_1 + \ldots + \mathtt{a}_k$ $(i{\neq}j)$
    wobei $\{a_1, .., a_k\} = \{a \in \Sigma \mid \delta(q_i, a){=}q_j\}$

Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

- **Basisfall $\boldsymbol{R_{ij}^0}$:** Pfad darf zwischendurch keine Zustände berühren
  - Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R_{ii}^0)$
  - Pfadlänge 1: $\{a \in \Sigma \mid \delta(q_i, a) = q_j\} \subseteq L(R_{ij}^0)$
  - Ergebnis: $\boldsymbol{R_{ii}^0} = \epsilon + \mathtt{a}_1 + \,..\, + \mathtt{a}_k$, $\boldsymbol{R_{ij}^0} = \emptyset + \mathtt{a}_1 + \,..\, + \mathtt{a}_k$ $(i{\neq}j)$
    wobei $\{a_1, .., a_k\} = \{a \in \Sigma \mid \delta(q_i, a){=}q_j\}$

- **Schrittfall $\boldsymbol{R_{ij}^k}$** $(0{<}k{\leq}n)$

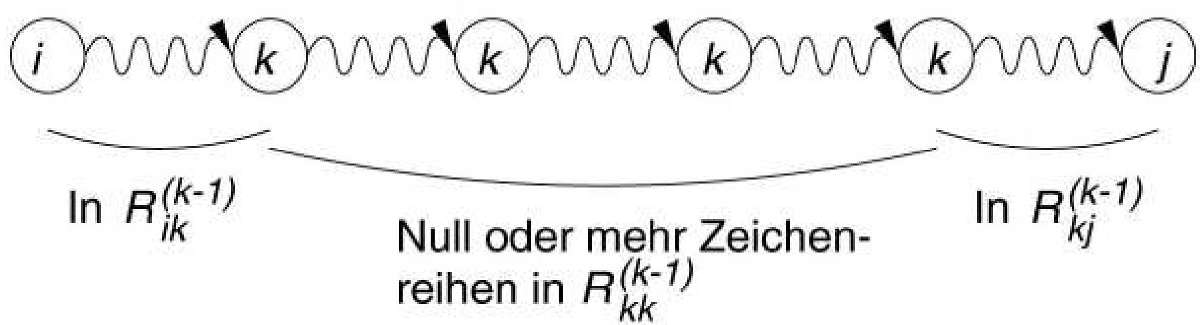
Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

- **Basisfall R_{ij}^{0}:** Pfad darf zwischendurch keine Zustände berühren
  - Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R_{ii}^0)$
  - Pfadlänge 1: $\{a \in \Sigma \mid \delta(q_i, a) = q_j\} \subseteq L(R_{ij}^0)$
  - Ergebnis: $R_{ii}^0 = \epsilon + \mathtt{a}_1 + \ldots + \mathtt{a}_k$, $R_{ij}^0 = \emptyset + \mathtt{a}_1 + \ldots + \mathtt{a}_k$ $(i{\neq}j)$
    wobei $\{a_1, .., a_k\} = \{a \in \Sigma \mid \delta(q_i, a){=}q_j\}$
- **Schrittfall R_{ij}^{k}** $(0{<}k{\leq}n)$
  - Worte $w \in L(R_{ij}^k)$, deren Pfad q_k nicht enthält: $L(R_{ij}^{k-1}) \subseteq L(R_{ij}^k)$

ITERATIVE BESTIMMUNG DER AUSDRÜCKE $R_{ij}^{(k)}$

- **Basisfall R_{ij}^0:** Pfad darf zwischendurch keine Zustände berühren
  - Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R_{ii}^0)$
  - Pfadlänge 1: $\{a \in \Sigma \mid \delta(q_i, a) = q_j\} \subseteq L(R_{ij}^0)$
  - Ergebnis: $R_{ii}^0 = \epsilon + \mathtt{a}_1 + \ldots + \mathtt{a}_k$, $R_{ij}^0 = \emptyset + \mathtt{a}_1 + \ldots + \mathtt{a}_k$ $(i{\neq}j)$
    wobei $\{a_1, .., a_k\} = \{a \in \Sigma \mid \delta(q_i, a){=}q_j\}$

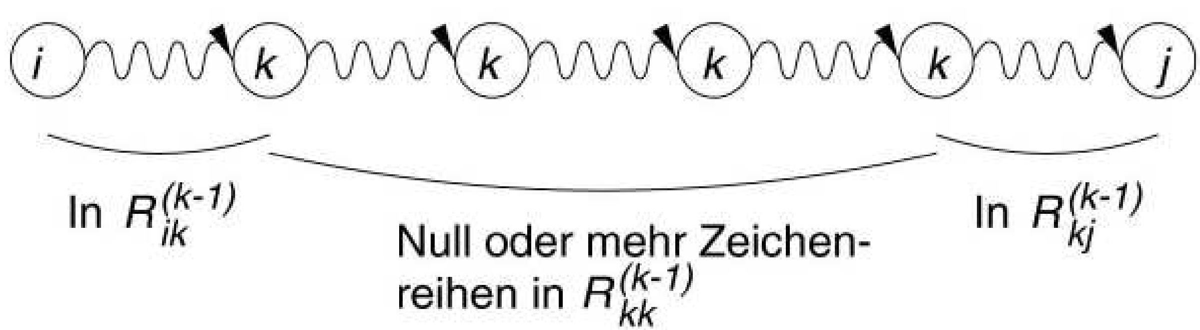
- **Schrittfall R_{ij}^k** $(0{<}k{\leq}n)$
  - Worte $w \in L(R_{ij}^k)$, deren Pfad q_k nicht enthält: $L(R_{ij}^{k-1}) \subseteq L(R_{ij}^k)$
  - Worte $w \in L(R_{ij}^k)$, deren Pfad q_k enthält:
    Zerlege w in $uz_1..z_pv$ mit $\hat{\delta}(q_i, u){=}q_k \wedge \forall l{\leq}p.\hat{\delta}(q_k, z_l){=}q_k \wedge \hat{\delta}(q_k, v){=}q_j$



Iterative Bestimmung der Ausdrücke $R_{ij}^{(k)}$

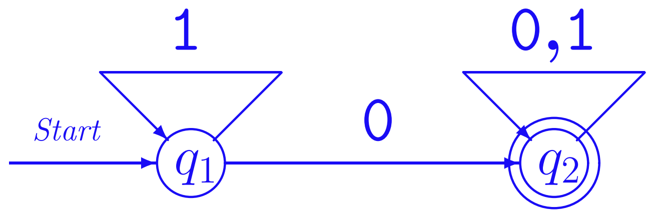
- **Basisfall R^0_{ij}:** Pfad darf zwischendurch keine Zustände berühren
  - Pfadlänge 0 (nur für $i{=}j$): $\epsilon \in L(R^0_{ii})$
  - Pfadlänge 1: $\{a\in\Sigma \mid \delta(q_i,a) = q_j\} \subseteq L(R^0_{ij})$
  - Ergebnis: $R^0_{ii} = \epsilon + \mathtt{a}_1 + .. + \mathtt{a}_k$, $\quad R^0_{ij} = \emptyset + \mathtt{a}_1 + .. + \mathtt{a}_k$ $(i{\neq}j)$
    wobei $\{a_1, .., a_k\} = \{a\in\Sigma \mid \delta(q_i,a){=}q_j\}$

- **Schrittfall R^k_{ij}** $(0{<}k{\leq}n)$
  - Worte $w\in L(R^k_{ij})$, deren Pfad q_k nicht enthält: $L(R^{k-1}_{ij}){\subseteq}L(R^k_{ij})$
  - Worte $w\in L(R^k_{ij})$, deren Pfad q_k enthält:
    Zerlege w in $uz_1..z_pv$ mit $\hat\delta(q_i,u){=}q_k \wedge \forall l{\leq}p.\hat\delta(q_k,z_l){=}q_k \wedge \hat\delta(q_k,v){=}q_j$



  - Ergebnis: $R^k_{ij} = R^{k-1}_{ij} + R^{k-1}_{ik} \circ (R^{k-1}_{kk})^* \circ R^{k-1}_{kj}$

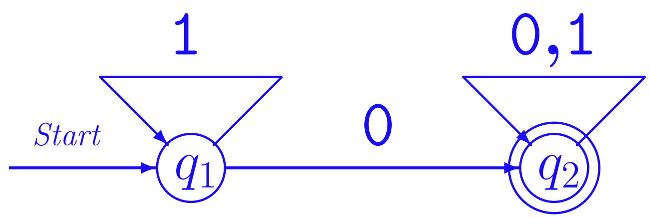
Umwandlung von Automaten am Beispiel



Umwandlung von Automaten am Beispiel

- **Basisfall**

  $R_{11}^{0} = \epsilon$ + 1

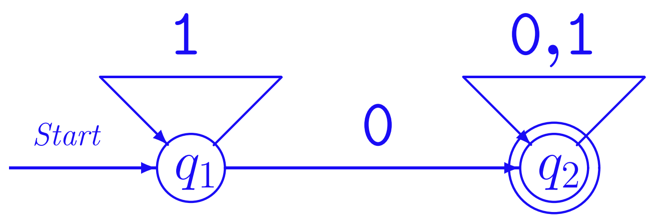


Umwandlung von Automaten am Beispiel

- **Basisfall**

  $R_{11}^{0} = \epsilon$ + 1

  $R_{12}^{0} =$ 0



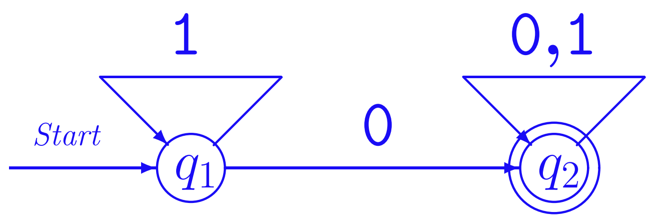
Umwandlung von Automaten am Beispiel

- **Basisfall**

$R_{11}^{0} = \epsilon \texttt{ + 1}$

$R_{12}^{0} = \texttt{0}$

$R_{21}^{0} = \emptyset$



Umwandlung von Automaten am Beispiel

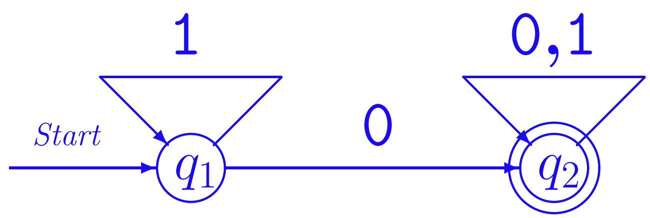
- **Basisfall**

  $R_{11}^{0} = \epsilon \texttt{ + 1}$

  $R_{12}^{0} = \texttt{0}$

  $R_{21}^{0} = \emptyset$

  $R_{22}^{0} = \epsilon \texttt{ + 0 + 1}$



Umwandlung von Automaten am Beispiel

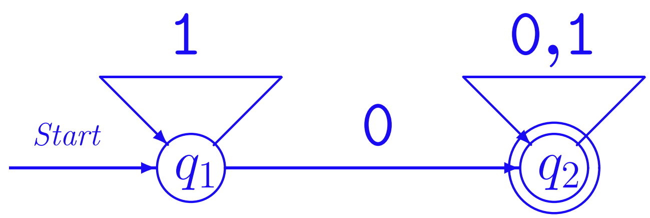
- **Basisfall**

  $R^0_{11} = \epsilon + 1$

  $R^0_{12} = 0$

  $R^0_{21} = \emptyset$

  $R^0_{22} = \epsilon + 0 + 1$



- **Stufe 1**

  $R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1)$

Umwandlung von Automaten am Beispiel

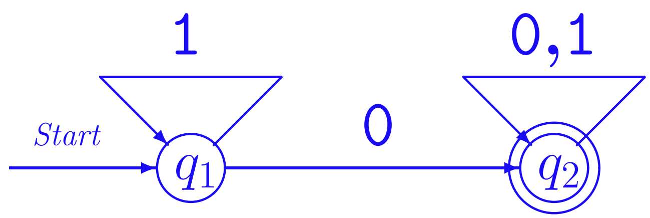
- **Basisfall**

  $R_{11}^0 = \epsilon \texttt{ + 1}$

  $R_{12}^0 = \texttt{0}$

  $R_{21}^0 = \emptyset$

  $R_{22}^0 = \epsilon \texttt{ + 0 + 1}$



- **Stufe 1**

  $R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon\texttt{+1+(}\epsilon\texttt{+1)(}\epsilon\texttt{+1)}^*\texttt{(}\epsilon\texttt{+1)}$

  $R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = \texttt{0+(}\epsilon\texttt{+1)(}\epsilon\texttt{+1)}^*\texttt{0}$

Umwandlung von Automaten am Beispiel

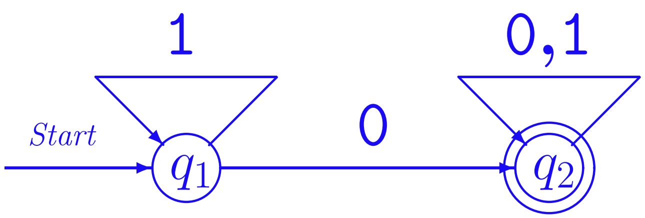
- **Basisfall**

  $R_{11}^{0} = \epsilon + 1$

  $R_{12}^{0} = 0$

  $R_{21}^{0} = \emptyset$

  $R_{22}^{0} = \epsilon + 0 + 1$



- **Stufe 1**

  $R_{11}^{1} = R_{11}^{0} + R_{11}^{0}(R_{11}^{0})^{*}R_{11}^{0} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^{*}(\epsilon + 1)$

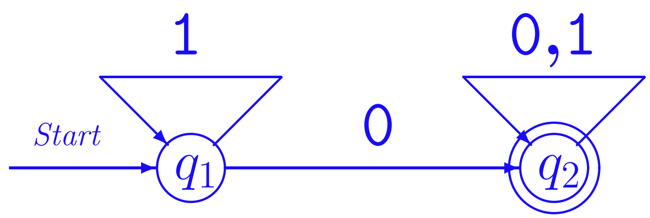
  $R_{12}^{1} = R_{12}^{0} + R_{11}^{0}(R_{11}^{0})^{*}R_{12}^{0} = 0 + (\epsilon + 1)(\epsilon + 1)^{*}0$

  $R_{21}^{1} = R_{21}^{0} + R_{21}^{0}(R_{11}^{0})^{*}R_{11}^{0} = \emptyset + \emptyset(\epsilon + 1)^{*}(\epsilon + 1)$

Umwandlung von Automaten am Beispiel

- **Basisfall**

$R_{11}^0 = \epsilon + 1$
$R_{12}^0 = 0$
$R_{21}^0 = \emptyset$
$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1)$
$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0$
$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1)$
$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0$

Umwandlung von Automaten am Beispiel

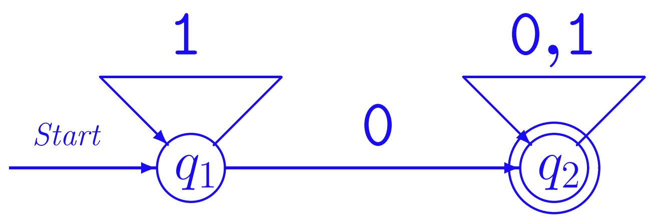
- **Basisfall**

$R^0_{11} = \epsilon + 1$

$R^0_{12} = 0$

$R^0_{21} = \emptyset$

$R^0_{22} = \epsilon + 0 + 1$



- **Stufe 1**

$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$

$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 0 + (\epsilon + 1)(\epsilon + 1)^* 0$

$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1)$

$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0$

Umwandlung von Automaten am Beispiel

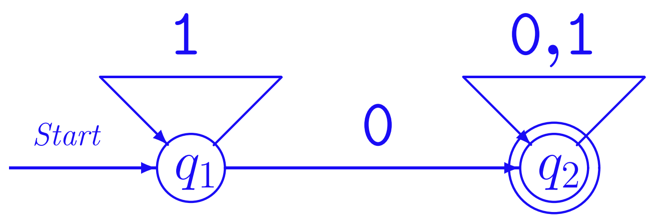
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$$

$$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^* 0$$

$$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1)$$

$$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0$$

Umwandlung von Automaten am Beispiel

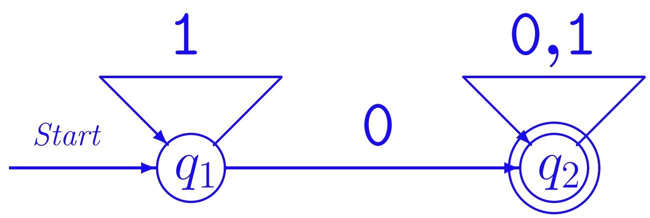
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$

$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^*0$

$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$

$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0$

Umwandlung von Automaten am Beispiel

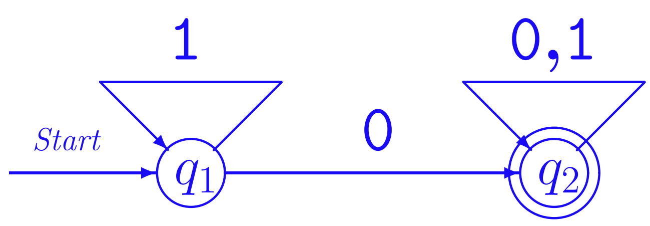
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$$R_{11}^1 = R_{11}^0 + R_{11}^0 (R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$$

$$R_{12}^1 = R_{12}^0 + R_{11}^0 (R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^* 0$$

$$R_{21}^1 = R_{21}^0 + R_{21}^0 (R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$$

$$R_{22}^1 = R_{22}^0 + R_{21}^0 (R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$$

Umwandlung von Automaten am Beispiel

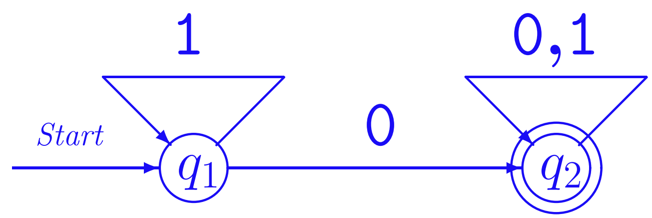
• Basisfall

$R^0_{11} = \epsilon + 1$

$R^0_{12} = 0$

$R^0_{21} = \emptyset$

$R^0_{22} = \epsilon + 0 + 1$



• Stufe 1

$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$

$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^*0$

$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$

$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$

• Stufe 2

$R^2_{11} = R^1_{11} + R^1_{12}(R^1_{22})^* R^1_{21} = 1^* + 1^*0(\epsilon+0+1)^*\emptyset$

Umwandlung von Automaten am Beispiel

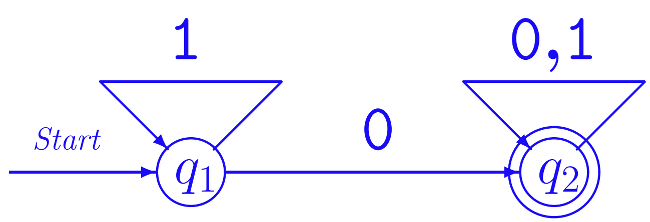
- **Basisfall**

$R^0_{11} = \epsilon + 1$

$R^0_{12} = 0$

$R^0_{21} = \emptyset$

$R^0_{22} = \epsilon + 0 + 1$



- **Stufe 1**

$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \mapsto 1^*$

$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \mapsto 1^* 0$

$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \mapsto \emptyset$

$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \mapsto \epsilon + 0 + 1$

- **Stufe 2**

$R^2_{11} = R^1_{11} + R^1_{12}(R^1_{22})^* R^1_{21} = 1^* + 1^* 0(\epsilon+0+1)^* \emptyset$

$R^2_{12} = R^1_{12} + R^1_{12}(R^1_{22})^* R^1_{22} = 1^* 0 + 1^* 0(\epsilon+0+1)^*(\epsilon+0+1)$

Umwandlung von Automaten am Beispiel

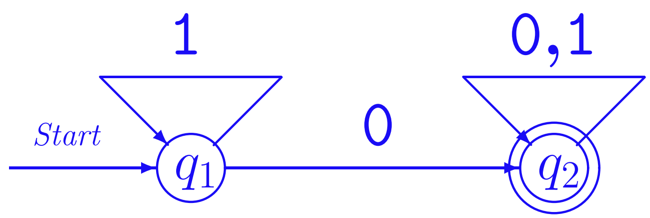
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$$

$$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^*0$$

$$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$$

$$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$$

- **Stufe 2**

$$R_{11}^2 = R_{11}^1 + R_{12}^1(R_{22}^1)^* R_{21}^1 = 1^* + 1^*0(\epsilon{+}0{+}1)^*\emptyset$$

$$R_{12}^2 = R_{12}^1 + R_{12}^1(R_{22}^1)^* R_{22}^1 = 1^*0 + 1^*0(\epsilon{+}0{+}1)^*(\epsilon{+}0{+}1)$$

$$R_{21}^2 = R_{21}^1 + R_{22}^1(R_{22}^1)^* R_{21}^1 = \emptyset + (\epsilon{+}0{+}1)(\epsilon{+}0{+}1)^*\emptyset$$

Umwandlung von Automaten am Beispiel

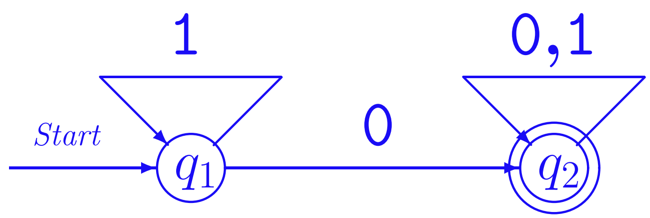
- **Basisfall**

$R^0_{11} = \epsilon + 1$

$R^0_{12} = 0$

$R^0_{21} = \emptyset$

$R^0_{22} = \epsilon + 0 + 1$



- **Stufe 1**

$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$

$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^* 0$

$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$

$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$

- **Stufe 2**

$R^2_{11} = R^1_{11} + R^1_{12}(R^1_{22})^* R^1_{21} = 1^* + 1^* 0(\epsilon+0+1)^* \emptyset$

$R^2_{12} = R^1_{12} + R^1_{12}(R^1_{22})^* R^1_{22} = 1^* 0 + 1^* 0(\epsilon+0+1)^*(\epsilon+0+1)$

$R^2_{21} = R^1_{21} + R^1_{22}(R^1_{22})^* R^1_{21} = \emptyset + (\epsilon+0+1)(\epsilon+0+1)^* \emptyset$

$R^2_{22} = R^1_{22} + R^1_{22}(R^1_{22})^* R^1_{22} = (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1)$

Umwandlung von Automaten am Beispiel

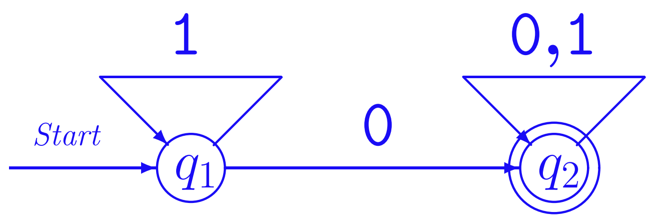
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \mapsto 1^*$

$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \mapsto 1^* 0$

$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \mapsto \emptyset$

$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \mapsto \epsilon + 0 + 1$

- **Stufe 2**

$R_{11}^2 = R_{11}^1 + R_{12}^1(R_{22}^1)^* R_{21}^1 = 1^* + 1^* 0(\epsilon+0+1)^* \emptyset \mapsto 1^*$

$R_{12}^2 = R_{12}^1 + R_{12}^1(R_{22}^1)^* R_{22}^1 = 1^* 0 + 1^* 0(\epsilon+0+1)^*(\epsilon+0+1)$

$R_{21}^2 = R_{21}^1 + R_{22}^1(R_{22}^1)^* R_{21}^1 = \emptyset + (\epsilon+0+1)(\epsilon+0+1)^* \emptyset$

$R_{22}^2 = R_{22}^1 + R_{22}^1(R_{22}^1)^* R_{22}^1 = (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1)$

Umwandlung von Automaten am Beispiel

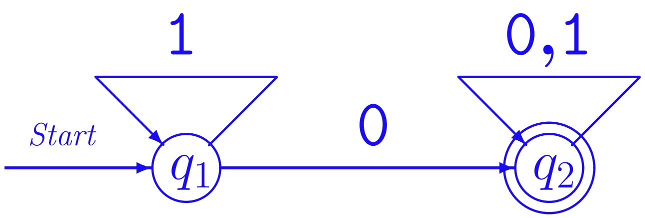
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon+1)(\epsilon+1)^*(\epsilon+1) \quad \mapsto \quad 1^*$

$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon+1)(\epsilon+1)^* 0 \quad \mapsto \quad 1^* 0$

$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon+1)^*(\epsilon+1) \quad \mapsto \quad \emptyset$

$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon+1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$

- **Stufe 2**

$R_{11}^2 = R_{11}^1 + R_{12}^1(R_{22}^1)^* R_{21}^1 = 1^* + 1^* 0(\epsilon+0+1)^* \emptyset \quad \mapsto \quad 1^*$

$R_{12}^2 = R_{12}^1 + R_{12}^1(R_{22}^1)^* R_{22}^1 = 1^* 0 + 1^* 0(\epsilon+0+1)^*(\epsilon+0+1) \quad \mapsto \quad 1^* 0(0+1)^*$

$R_{21}^2 = R_{21}^1 + R_{22}^1(R_{22}^1)^* R_{21}^1 = \emptyset + (\epsilon+0+1)(\epsilon+0+1)^* \emptyset$

$R_{22}^2 = R_{22}^1 + R_{22}^1(R_{22}^1)^* R_{22}^1 = (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1)$

Umwandlung von Automaten am Beispiel

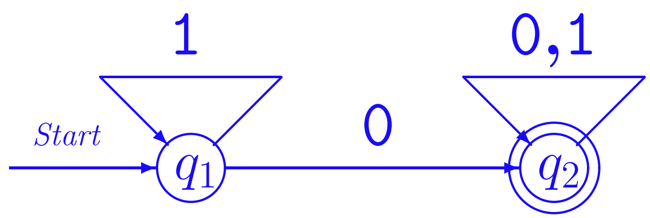
- **Basisfall**

$R^0_{11} = \epsilon + 1$

$R^0_{12} = 0$

$R^0_{21} = \emptyset$

$R^0_{22} = \epsilon + 0 + 1$



- **Stufe 1**

$$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$$

$$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^*0$$

$$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$$

$$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$$

- **Stufe 2**

$$R^2_{11} = R^1_{11} + R^1_{12}(R^1_{22})^* R^1_{21} = 1^* + 1^*0(\epsilon+0+1)^*\emptyset \quad \mapsto \quad 1^*$$

$$R^2_{12} = R^1_{12} + R^1_{12}(R^1_{22})^* R^1_{22} = 1^*0 + 1^*0(\epsilon+0+1)^*(\epsilon+0+1) \quad \mapsto \quad 1^*0(0+1)^*$$

$$R^2_{21} = R^1_{21} + R^1_{22}(R^1_{22})^* R^1_{21} = \emptyset + (\epsilon+0+1)(\epsilon+0+1)^*\emptyset \quad \mapsto \quad \emptyset$$

$$R^2_{22} = R^1_{22} + R^1_{22}(R^1_{22})^* R^1_{22} = (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1)$$

Umwandlung von Automaten am Beispiel

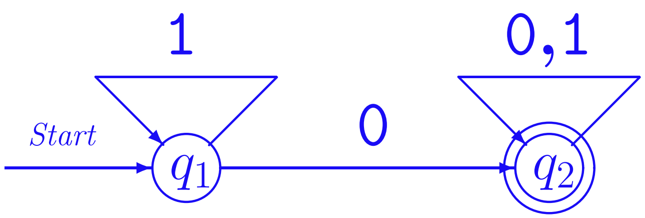
- **Basisfall**

$R_{11}^0 = \epsilon + 1$

$R_{12}^0 = 0$

$R_{21}^0 = \emptyset$

$R_{22}^0 = \epsilon + 0 + 1$



- **Stufe 1**

$$R_{11}^1 = R_{11}^0 + R_{11}^0(R_{11}^0)^* R_{11}^0 = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \mapsto 1^*$$

$$R_{12}^1 = R_{12}^0 + R_{11}^0(R_{11}^0)^* R_{12}^0 = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \mapsto 1^* 0$$

$$R_{21}^1 = R_{21}^0 + R_{21}^0(R_{11}^0)^* R_{11}^0 = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \mapsto \emptyset$$

$$R_{22}^1 = R_{22}^0 + R_{21}^0(R_{11}^0)^* R_{12}^0 = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \mapsto \epsilon + 0 + 1$$

- **Stufe 2**

$$R_{11}^2 = R_{11}^1 + R_{12}^1(R_{22}^1)^* R_{21}^1 = 1^* + 1^* 0(\epsilon+0+1)^* \emptyset \mapsto 1^*$$

$$R_{12}^2 = R_{12}^1 + R_{12}^1(R_{22}^1)^* R_{22}^1 = 1^* 0 + 1^* 0(\epsilon+0+1)^*(\epsilon+0+1) \mapsto 1^* 0(0+1)^*$$

$$R_{21}^2 = R_{21}^1 + R_{22}^1(R_{22}^1)^* R_{21}^1 = \emptyset + (\epsilon+0+1)(\epsilon+0+1)^* \emptyset \mapsto \emptyset$$

$$R_{22}^2 = R_{22}^1 + R_{22}^1(R_{22}^1)^* R_{22}^1 = (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1) \mapsto (0+1)^*$$

Umwandlung von Automaten am Beispiel

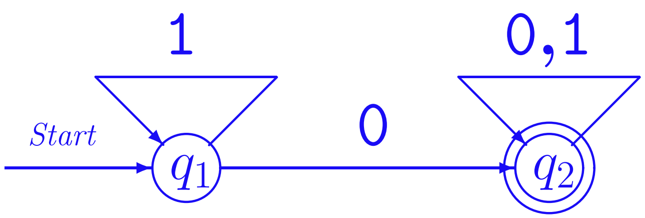
- **Basisfall**

$R^0_{11} = \epsilon + 1$

$R^0_{12} = 0$

$R^0_{21} = \emptyset$

$R^0_{22} = \epsilon + 0 + 1$



- **Stufe 1**

$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad 1^*$

$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 0 + (\epsilon + 1)(\epsilon + 1)^* 0 \quad \mapsto \quad 1^* 0$

$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) \quad \mapsto \quad \emptyset$

$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0 \quad \mapsto \quad \epsilon + 0 + 1$

- **Stufe 2**

$R^2_{11} = R^1_{11} + R^1_{12}(R^1_{22})^* R^1_{21} = 1^* + 1^* 0(\epsilon+0+1)^* \emptyset \quad \mapsto \quad 1^*$

$R^2_{12} = R^1_{12} + R^1_{12}(R^1_{22})^* R^1_{22} = 1^* 0 + 1^* 0(\epsilon+0+1)^*(\epsilon+0+1) \quad \mapsto \quad 1^* 0(0+1)^*$

$R^2_{21} = R^1_{21} + R^1_{22}(R^1_{22})^* R^1_{21} = \emptyset + (\epsilon+0+1)(\epsilon+0+1)^* \emptyset \quad \mapsto \quad \emptyset$

$R^2_{22} = R^1_{22} + R^1_{22}(R^1_{22})^* R^1_{22} = (\epsilon+0+1) + (\epsilon+0+1)(\epsilon+0+1)^*(\epsilon+0+1) \quad \mapsto \quad (0+1)^*$

Regulärer Ausdruck des Automaten: $1^* 0(0+1)^*$

Eine effizientere Umwandlungsmethode

- **Direkte Umwandlung ist sehr aufwendig**
  - Es müssen n^3 Ausdrücke R_{ij}^k erzeugt werden
  - Ausdrücke R_{ij}^k können viermal so groß wie die R_{ij}^{k-1} werden
  - Ohne Vereinfachung der R_{ij}^k sind bis zu $n^3 * 4^n$ Symbole zu erzeugen
  - Optimierung des Verfahrens: vermeide Vielfachkopien der R_{ij}^{k-1}

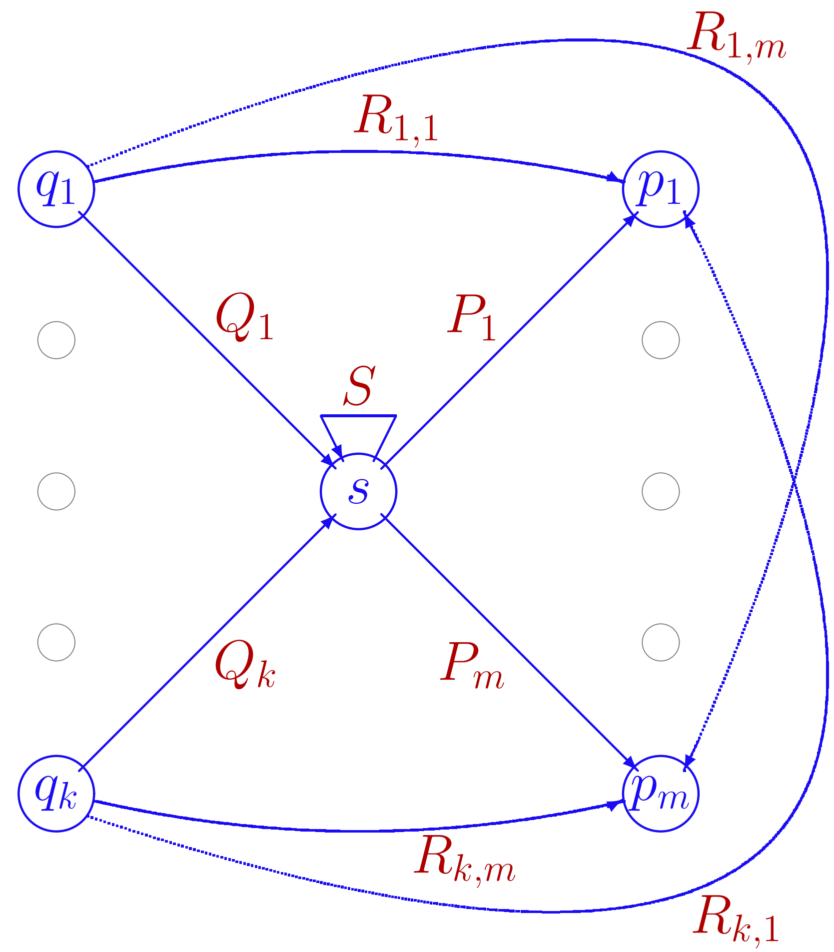
Eine effizientere Umwandlungsmethode

- **Direkte Umwandlung ist sehr aufwendig**
  - Es müssen n^3 Ausdrücke R_{ij}^k erzeugt werden
  - Ausdrücke R_{ij}^k können viermal so groß wie die R_{ij}^{k-1} werden
  - Ohne Vereinfachung der R_{ij}^k sind bis zu $n^3 * 4^n$ Symbole zu erzeugen
  - Optimierung des Verfahrens: vermeide Vielfachkopien der R_{ij}^{k-1}
- **Effizienterer Zugang: Elimination von Zuständen**
  - Statt Pfade zu verlängern, lege Zustände des Automaten zusammen
  - Ersetze Übergänge $q_i \xrightarrow{a\,\in\,\Sigma} q_j$ durch Übergänge mit regulären Ausdrücken
  - Schrittweise Umwandlung erzeugt regulären Ausdruck des Automaten

Eine effizientere Umwandlungsmethode

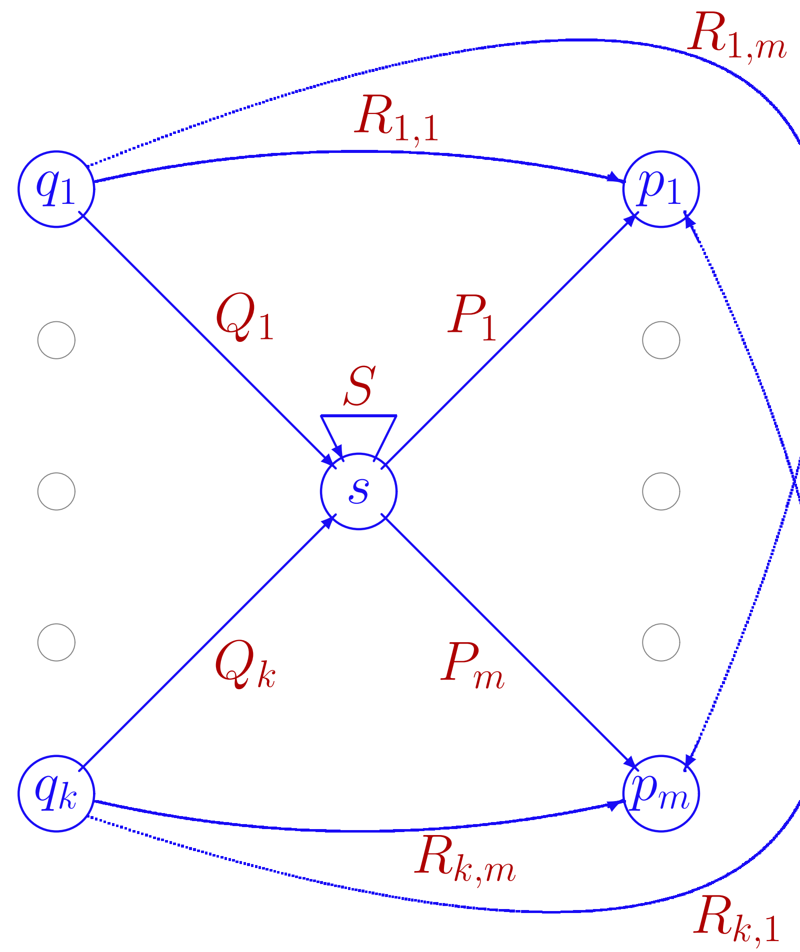
- **Direkte Umwandlung ist sehr aufwendig**
  - Es müssen n^3 Ausdrücke R_{ij}^k erzeugt werden
  - Ausdrücke R_{ij}^k können viermal so groß wie die R_{ij}^{k-1} werden
  - Ohne Vereinfachung der R_{ij}^k sind bis zu $n^3 * 4^n$ Symbole zu erzeugen
  - Optimierung des Verfahrens: vermeide Vielfachkopien der R_{ij}^{k-1}
- **Effizienterer Zugang: Elimination von Zuständen**
  - Statt Pfade zu verlängern, lege Zustände des Automaten zusammen
  - Ersetze Übergänge $q_i \xrightarrow{a\in\Sigma} q_j$ durch Übergänge mit regulären Ausdrücken
  - Schrittweise Umwandlung erzeugt regulären Ausdruck des Automaten
- **Technisches Hilfsmittel: RA-Automaten**
  - Überführungsfunktion δ arbeitet auf regulären Ausdrücken
  - A akzeptiert w, wenn es einen Pfad $w = v_1..v_m$ von q_0 zu einem $q \in F$ gibt und alle v_i in der Sprache des entsprechenden regulären Ausdrucks liegen
  - Konsistente Formalisierung mühsam und ohne Erkenntnisgewinn

Zustandselimination in RA-Automaten



Eliminiere Zustand s
mit Vorgängern $q_1, .., q_k$
und Nachfolgern $p_1, .., p_n$

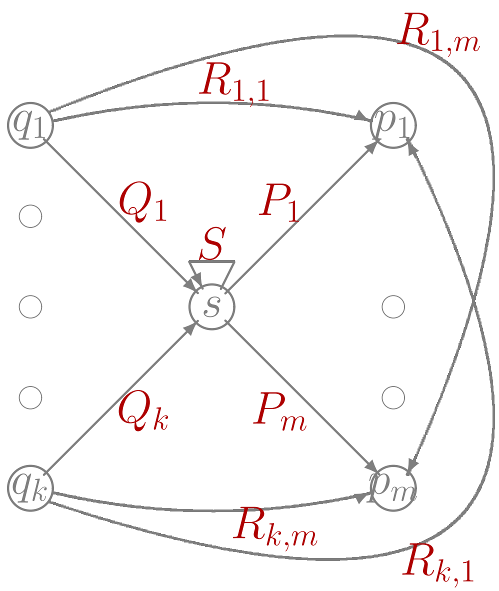
Zustandselimination in RA-Automaten

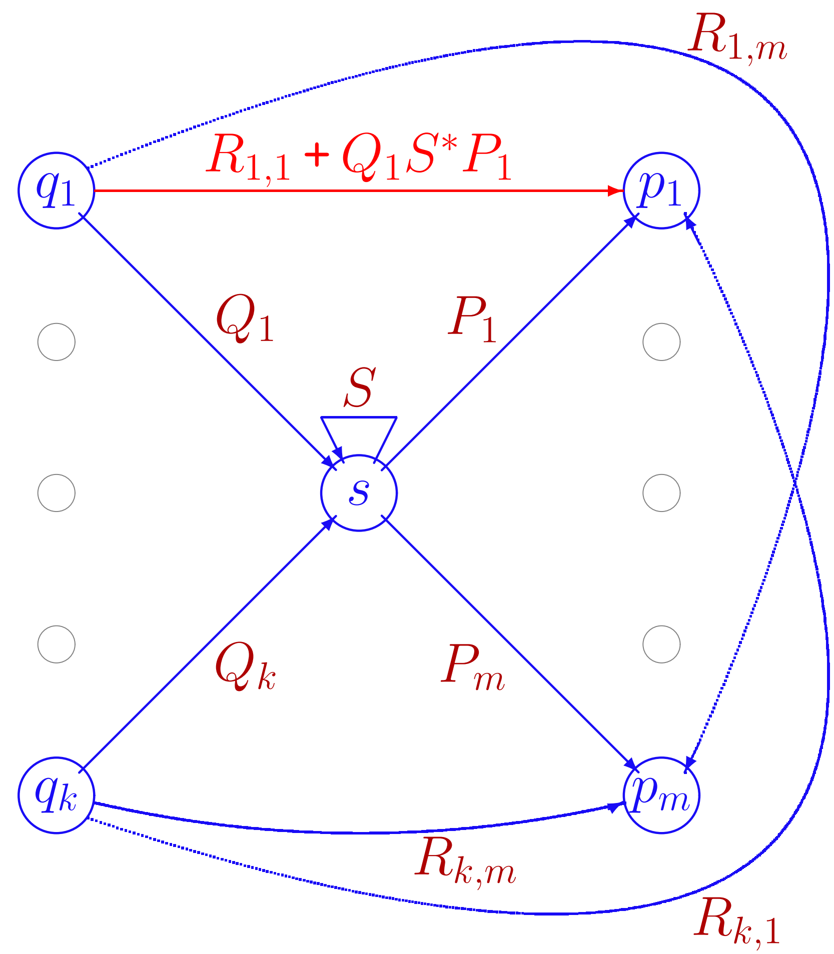


Eliminiere Zustand s
mit Vorgängern $q_1, .., q_k$
und Nachfolgern $p_1, .., p_n$

– Eliminiere Pfad von q_1 nach p_1 über s: $R_{1,1} + Q_1 S^* P_1$

Zustandselimination in RA-Automaten





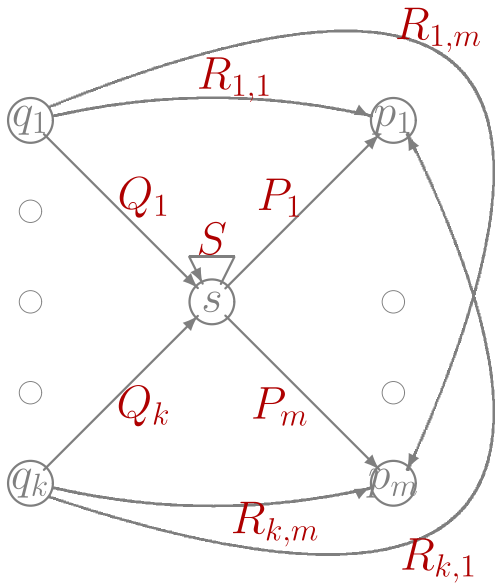
Eliminiere Zustand s
mit Vorgängern $q_1, .., q_k$
und Nachfolgern $p_1, .., p_n$

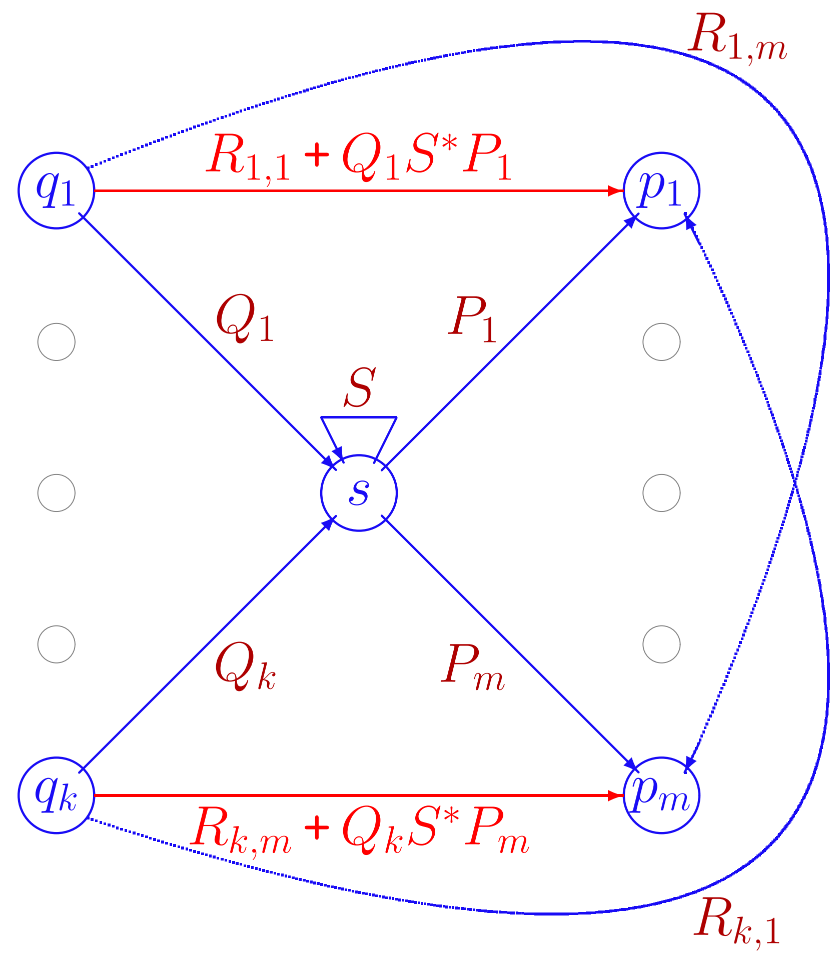
– Eliminiere Pfad von q_1 nach p_1 über s: $R_{1,1} + Q_1 S^* P_1$

$\vdots$

– Eliminiere Pfad von q_1 nach p_m über s: $R_{1,m} + Q_1 S^* P_m$

Zustandselimination in RA-Automaten





Eliminiere Zustand s
mit Vorgängern $q_1, .., q_k$
und Nachfolgern $p_1, .., p_n$

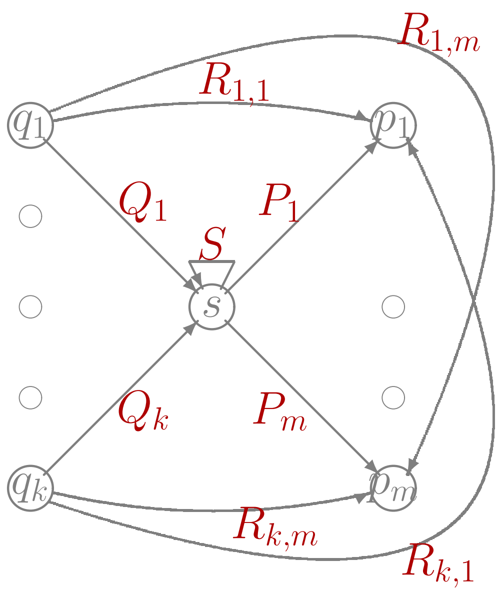
– Eliminiere Pfad von q_1 nach p_1 über s: $R_{1,1} + Q_1S^*P_1$

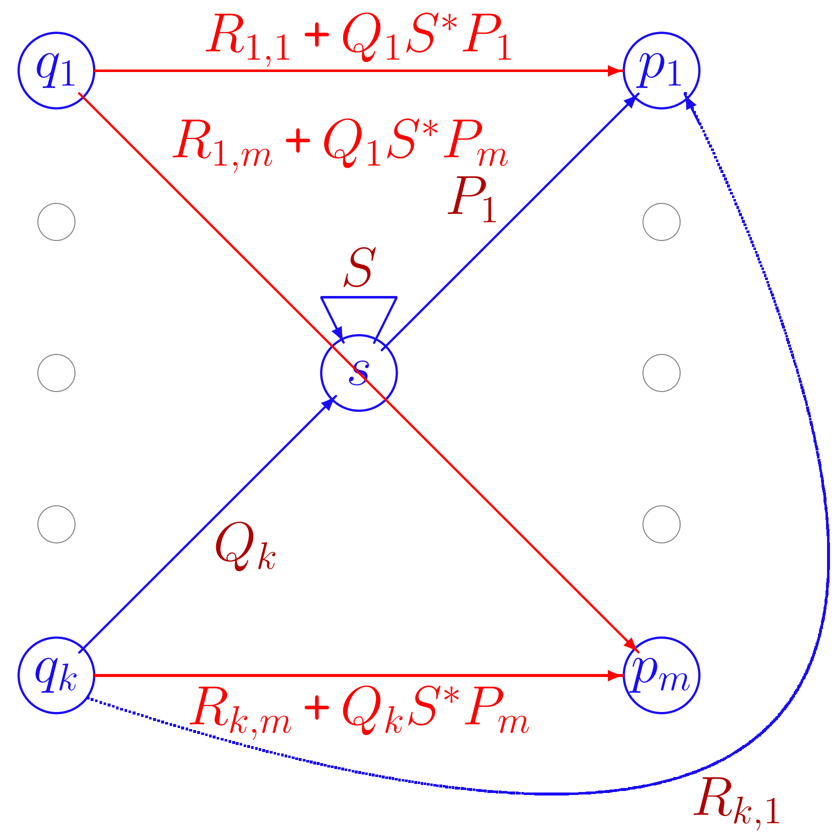
$\vdots$

– Eliminiere Pfad von q_1 nach p_m über s: $R_{1,m} + Q_1S^*P_m$

– Eliminiere Pfad von q_k nach p_1 über s: $R_{k,1} + Q_kS^*P_1$

Zustandselimination in RA-Automaten





Eliminiere Zustand s
mit Vorgängern $q_1, .., q_k$
und Nachfolgern $p_1, .., p_n$

– Eliminiere Pfad von q_1 nach p_1 über s: $R_{1,1} + Q_1 S^* P_1$

$\vdots$

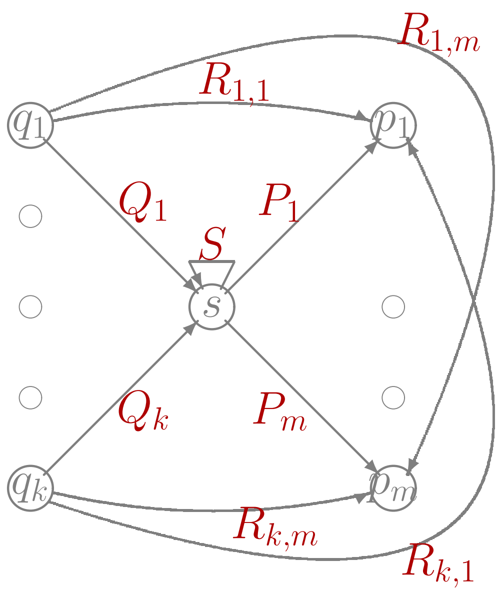
– Eliminiere Pfad von q_1 nach p_m über s: $R_{1,m} + Q_1 S^* P_m$

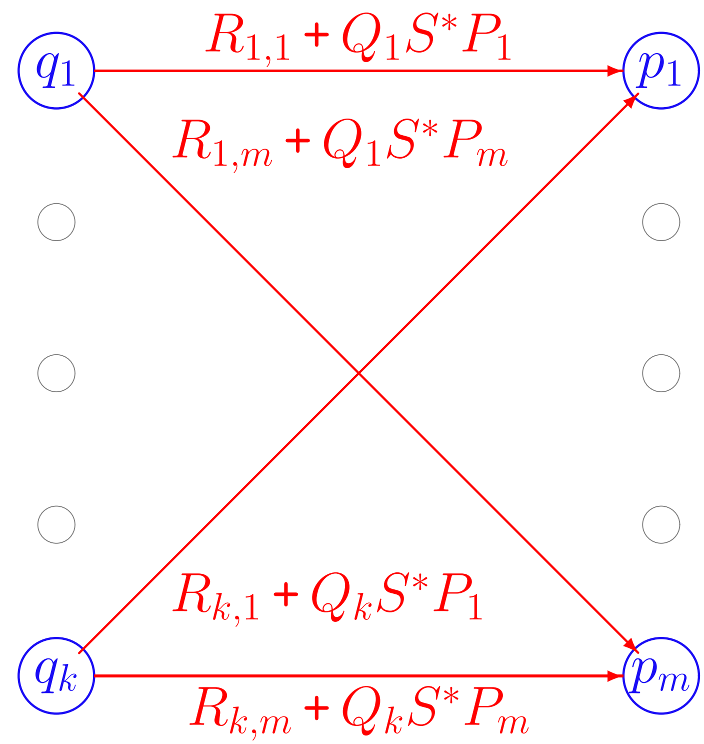
– Eliminiere Pfad von q_k nach p_1 über s: $R_{k,1} + Q_k S^* P_1$

$\vdots$

– Eliminiere Pfad von q_k nach p_m über s: $R_{k,m} + Q_k S^* P_m$

ZUSTANDSELIMINATION IN RA-AUTOMATEN





Eliminiere Zustand s
mit Vorgängern $q_1, .., q_k$
und Nachfolgern $p_1, .., p_n$

– Eliminiere Pfad von q_1 nach p_1 über s: $R_{1,1}+Q_1S^*P_1$

$\vdots$

– Eliminiere Pfad von q_1 nach p_m über s: $R_{1,m}+Q_1S^*P_m$

– Eliminiere Pfad von q_k nach p_1 über s: $R_{k,1}+Q_kS^*P_1$

$\vdots$

– Eliminiere Pfad von q_k nach p_m über s: $R_{k,m}+Q_kS^*P_m$

Umwandlung durch Zustandselimination

1. **Transformiere Automaten in RA-Automaten**
   – Ersetze Beschriftungen mit Symbolen $a \in \Sigma$ durch reguläre Ausdrücke

Umwandlung durch Zustandselimination

1. Transformiere Automaten in RA-Automaten

– Ersetze Beschriftungen mit Symbolen $a \in \Sigma$ durch reguläre Ausdrücke

2. Für $q \in F$ eliminiere alle Zustände außer q_0 und q

– Iterative Anwendung des Eliminationsverfahrens

Umwandlung durch Zustandselimination

1. Transformiere Automaten in RA-Automaten

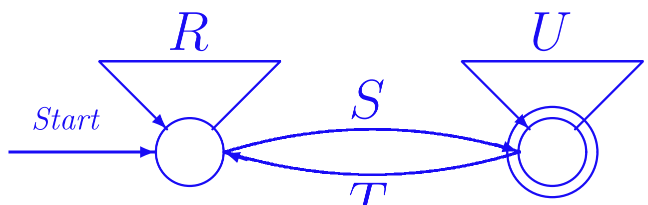
- Ersetze Beschriftungen mit Symbolen $a \in \Sigma$ durch reguläre Ausdrücke

2. Für $q \in F$ eliminiere alle Zustände außer q_0 und q

- Iterative Anwendung des Eliminationsverfahrens

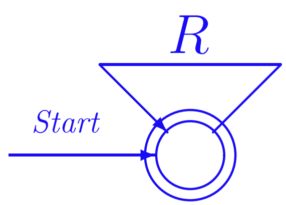
3. Bilde regulären Ausdruck aus finalem Automaten

- $q_0 \neq q$:



$(R + SU^*T)^*SU^*$

- $q_0 = q$:



R^*

1. Transformiere Automaten in RA-Automaten

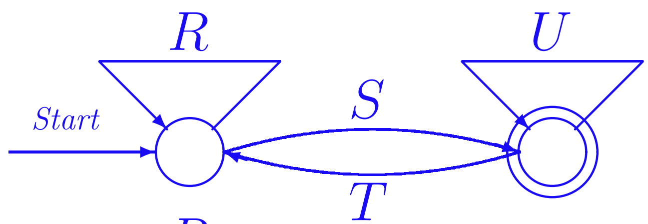
– Ersetze Beschriftungen mit Symbolen $a \in \Sigma$ durch reguläre Ausdrücke

2. Für $q \in F$ eliminiere alle Zustände außer q_0 und q

– Iterative Anwendung des Eliminationsverfahrens

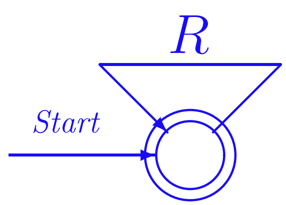
3. Bilde regulären Ausdruck aus finalem Automaten

– $q_0 \neq q$:



$(R + SU^*T)^*SU^*$

– $q_0 = q$:

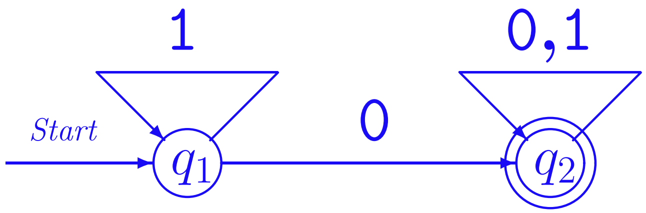


R^*

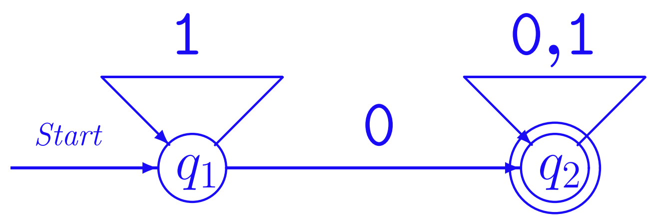
4. Vereinige Ausdrücke aller Endzustände

– Bilde Summe aller entstandenen regulären Ausdrücke

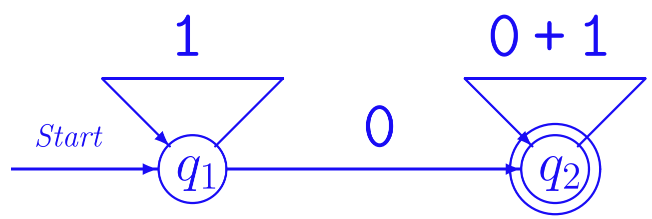
Umwandlung durch Zustandselimination am Beispiel



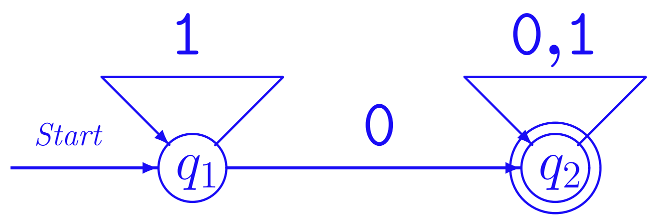
Umwandlung durch Zustandselimination am Beispiel



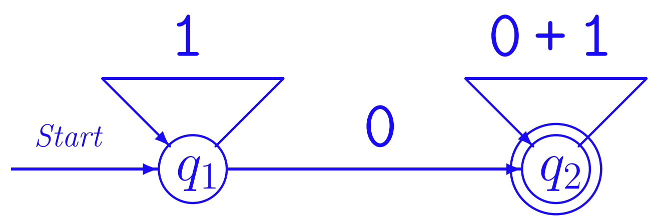
- **Transformiere in RA-Automaten**



Umwandlung durch Zustandselimination am Beispiel

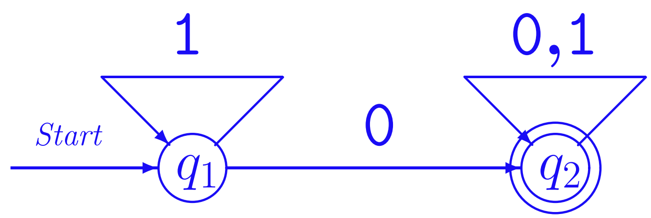


- **Transformiere in RA-Automaten**

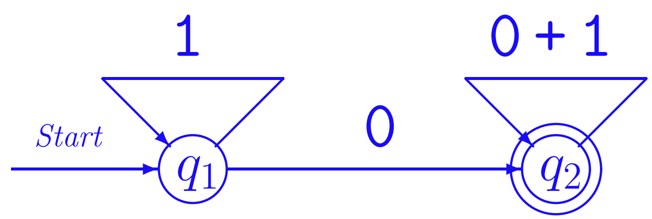


- **Keine Zustände zu eliminieren**

Umwandlung durch Zustandselimination am Beispiel



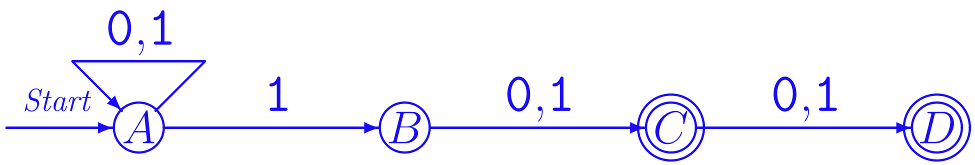
- **Transformiere in RA-Automaten**



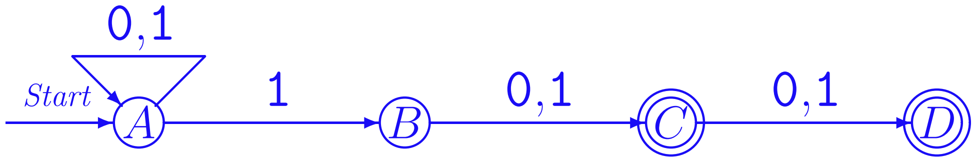
- **Keine Zustände zu eliminieren**

- **Bilde regulären Ausdruck aus finalem Automaten**
  - Extrahierter Ausdruck: $(1 + 0(0+1)^*\emptyset)^*0(0+1)^*$
  - Nach Vereinfachung: $\boxed{1^*0(0+1)^*}$

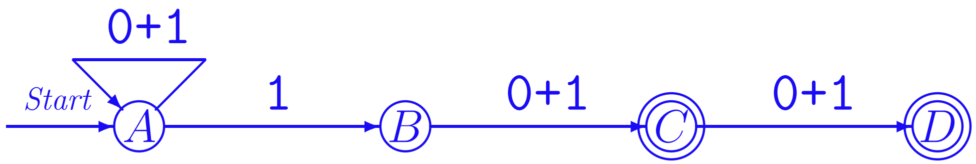
Umwandlung durch Zustandselimination am Beispiel



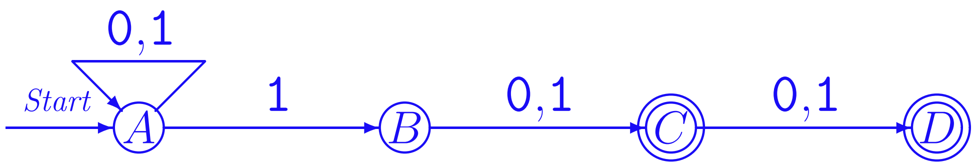
Umwandlung durch Zustandselimination am Beispiel



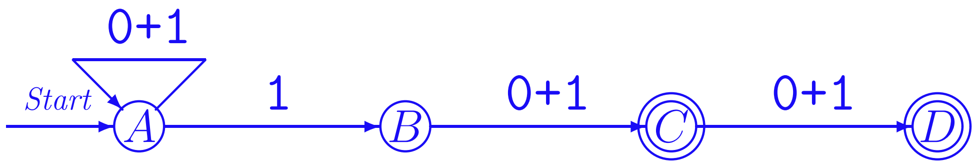
- **Transformiere in RA-Automaten**



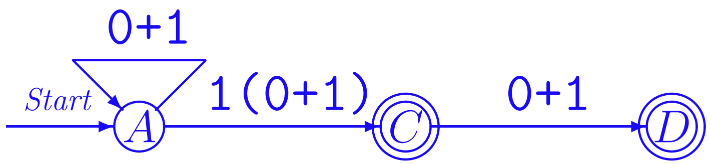
Umwandlung durch Zustandselimination am Beispiel



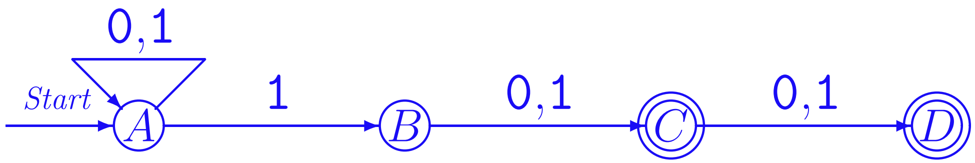
- **Transformiere in RA-Automaten**



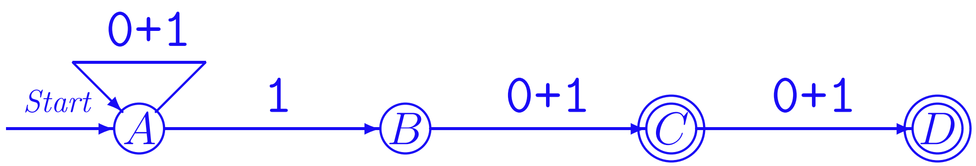
- **Elimination von Zustand *B***



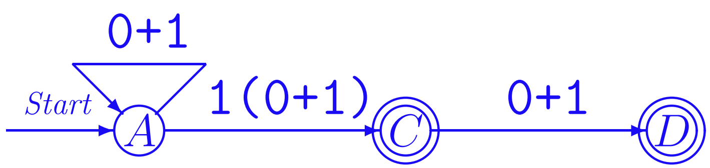
Umwandlung durch Zustandselimination am Beispiel



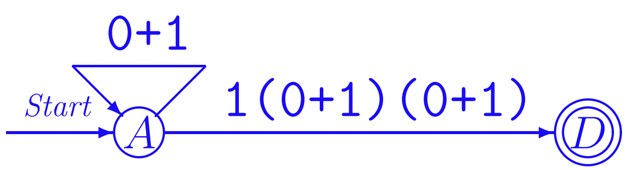
- **Transformiere in RA-Automaten**



- **Elimination von Zustand *B***

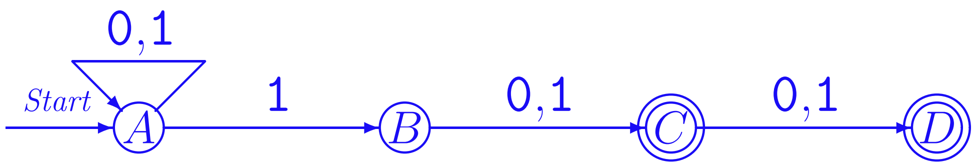


- **Elimination von Zustand *C* für Endzustand *D***

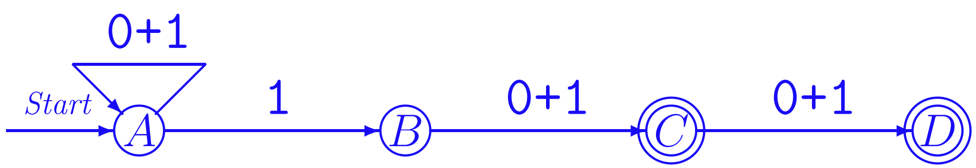


$(0+1)^*1(0+1)(0+1)$

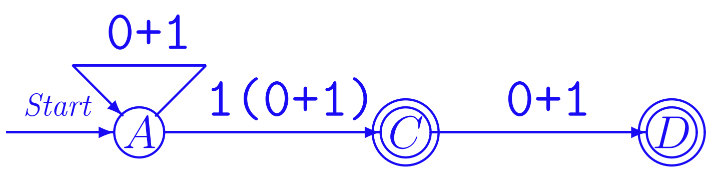
Umwandlung durch Zustandselimination am Beispiel



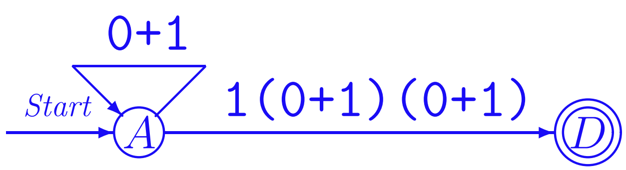
- **Transformiere in RA-Automaten**



- **Elimination von Zustand *B***

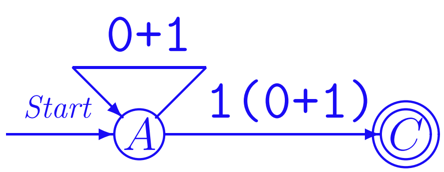


- **Elimination von Zustand *C* für Endzustand *D***



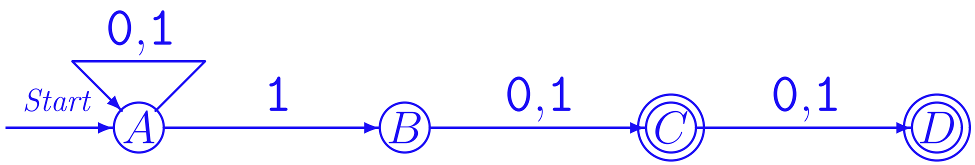
`(0+1)*1(0+1)(0+1)`

- **Elimination von Zustand *D* für Endzustand *C***

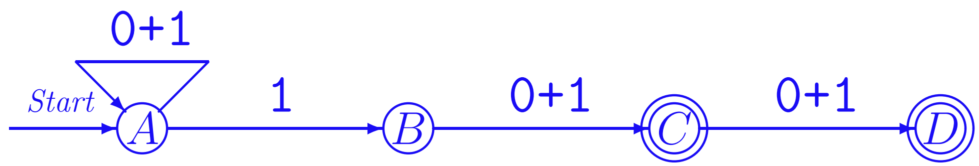


`(0+1)*1(0+1)`

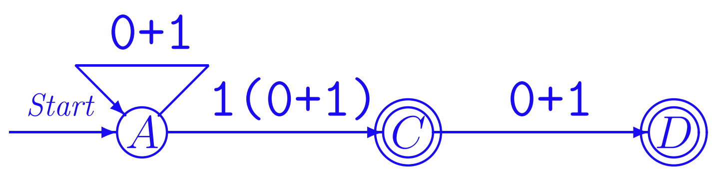
Umwandlung durch Zustandselimination am Beispiel



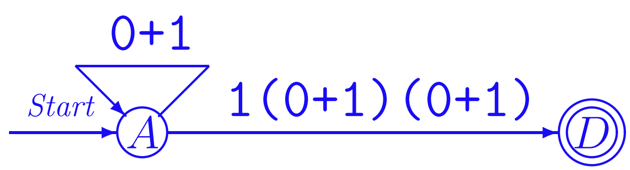
- **Transformiere in RA-Automaten**



- **Elimination von Zustand *B***

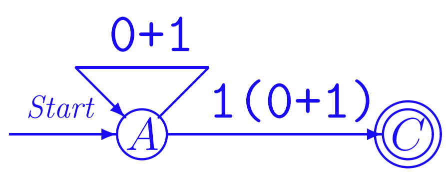


- **Elimination von Zustand *C* für Endzustand *D***



$(0+1)^*1(0+1)(0+1)$

- **Elimination von Zustand *D* für Endzustand *C***



$(0+1)^*1(0+1)$

- **Gesamter Ausdruck:** $\boxed{(0+1)^*1(0+1)\ +\ (0+1)^*1(0+1)(0+1)}$

Reguläre Ausdrücke – Zusammenfassung

- **Algebraische Notation für Sprachen**
  - ϵ, $\emptyset$, Symbole des Alphabets, Vereinigung, Verkettung, Sternoperator
  - Äquivalent zu endlichen Automaten
  - Gut zum Nachweis algebraischer Gesetze von Sprachen
  - Anwendung in Programmiersprachen und Suchmaschinen

Reguläre Ausdrücke – Zusammenfassung

- **Algebraische Notation für Sprachen**
  - ϵ, $\emptyset$, Symbole des Alphabets, Vereinigung, Verkettung, Sternoperator
  - Äquivalent zu endlichen Automaten
  - Gut zum Nachweis algebraischer Gesetze von Sprachen
  - Anwendung in Programmiersprachen und Suchmaschinen
- **Transformation in endliche Automaten**
  - Iterative Konstruktion von ϵ-NEAs
  - Nachträgliche Optimierung durch Elimination von ϵ-Übergängen

Reguläre Ausdrücke – Zusammenfassung

- **Algebraische Notation für Sprachen**
  - ϵ, $\emptyset$, Symbole des Alphabets, Vereinigung, Verkettung, Sternoperator
  - Äquivalent zu endlichen Automaten
  - Gut zum Nachweis algebraischer Gesetze von Sprachen
  - Anwendung in Programmiersprachen und Suchmaschinen

- **Transformation in endliche Automaten**
  - Iterative Konstruktion von ϵ-NEAs
  - Nachträgliche Optimierung durch Elimination von ϵ-Übergängen

- **Transformation von Automaten in Ausdrücke**
  - Konstruktion von Ausdrücken für Verarbeitungspfade im Automaten
  - Konstruktion durch Elimination von Zuständen in RA Automaten
  - Nachträgliche Optimierungen durch Anwendung algebraischer Gesetze