Brzozowski Goes Concurrent

A Kleene Theorem for Pomset Languages

Tobias Kappé[1] Paul Brunet[1] Bas Luttik[2] Alexandra Silva[1] Fabio Zanasi[1]

[1]University College London

[2]Eindhoven University of Technology

CONCUR 2017

Introduction

Kleene Algebra models *program flow*.

- abort (0) and skip (1)
- non-deterministic choice ($+$)
- sequential composition ($\cdot$)
- indefinite repetition (*)

Introduction

Kleene Algebra models *program flow*.

- abort (0) and skip (1)
- non-deterministic choice ($+$)
- sequential composition ($\cdot$)
- indefinite repetition (*)

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_1 \leftarrow y$	$r_2 \leftarrow x$

Introduction

Kleene Algebra models *program flow*.

- abort (0) and skip (1)
- non-deterministic choice ($+$)
- sequential composition ($\cdot$)
- indefinite repetition (*)
- parallel composition ($\|$)

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_1 \leftarrow y$	$r_2 \leftarrow x$

Introduction

Kleene Algebra models *program flow*.

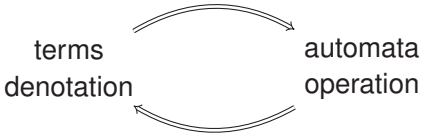
- abort (0) and skip (1)
- non-deterministic choice ($+$)
- sequential composition ($\cdot$)
- indefinite repetition (*)
- parallel composition ($\|$)

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_1 \leftarrow y$	$r_2 \leftarrow x$

$$(x \leftarrow 1; r_1 \leftarrow y) \parallel (y \leftarrow 1; r_2 \leftarrow x)$$

Introduction

Kleene theorem:



Introduction

Existing Kleene Theorems[1] have . . .

- . . . non-deterministic automata
- . . . a *semantic* precondition on automata

[1] Jipsen 2014; Lodaya and Weil 2000.

Introduction

Existing Kleene Theorems[1] have ...

- ... non-deterministic automata
- ... a *semantic* precondition on automata

Our Kleene Theorem has ...

- ... (sequentially) deterministic automata
- ... a *syntactic* precondition on automata

[1] Jipsen 2014; Lodaya and Weil 2000.

Preliminaries

$\mathcal{T}$ given by the grammar

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e \parallel f \mid e^*$$

Preliminaries

$\mathcal{T}$ given by the grammar

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e \parallel f \mid e^*$$

Preliminaries

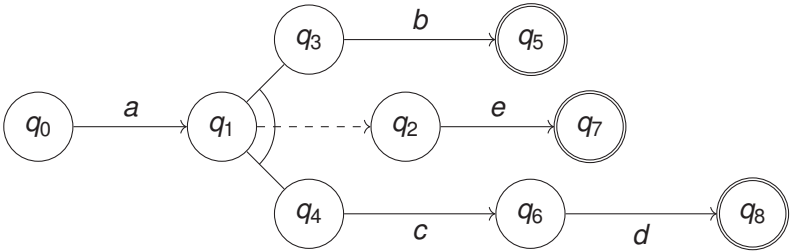
$$a \cdot (b \parallel c) \cdot d = \quad a \;\begin{matrix} \nearrow b \searrow \\ \searrow c \nearrow \end{matrix}\; d$$

Preliminaries

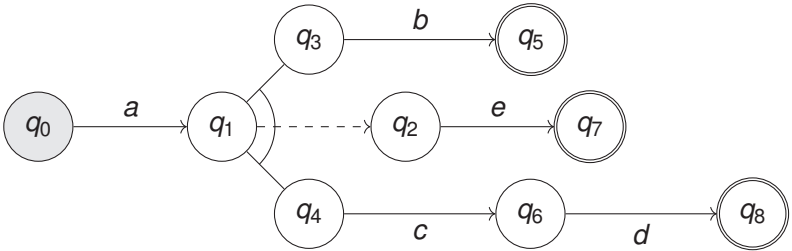
$$[\![-]\!] : \mathcal{T} \to 2^{\mathrm{Pom}_\Sigma}$$

$$\begin{aligned}
[\![0]\!] &= \emptyset \\
[\![1]\!] &= \{1\} \\
[\![a]\!] &= \{a\} \\
[\![e + f]\!] &= [\![e]\!] \cup [\![f]\!] \\
[\![e \cdot f]\!] &= [\![e]\!] \cdot [\![f]\!] \\
[\![e \parallel f]\!] &= [\![e]\!] \parallel [\![f]\!] \\
[\![e^*]\!] &= [\![e]\!]^*
\end{aligned}$$

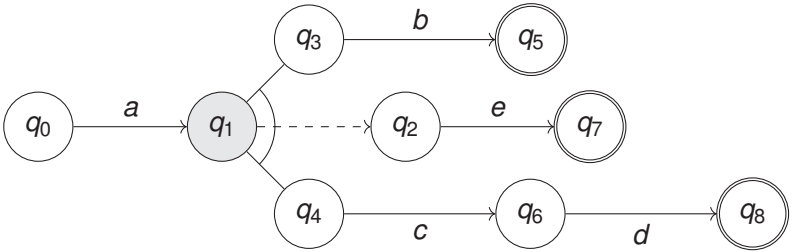
Pomset Automata



Pomset Automata

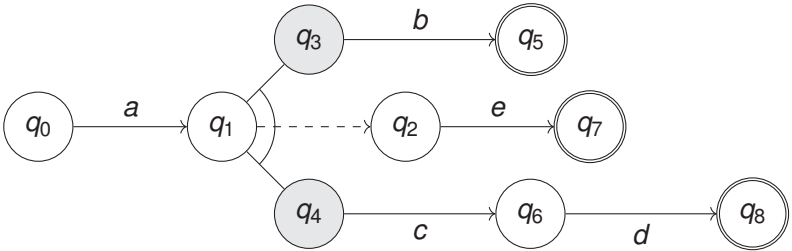


Pomset Automata



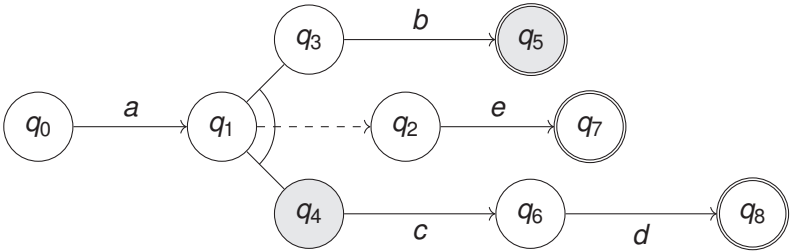
a

Pomset Automata



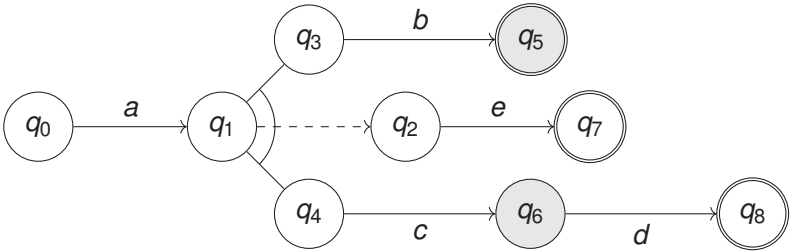
a

Pomset Automata



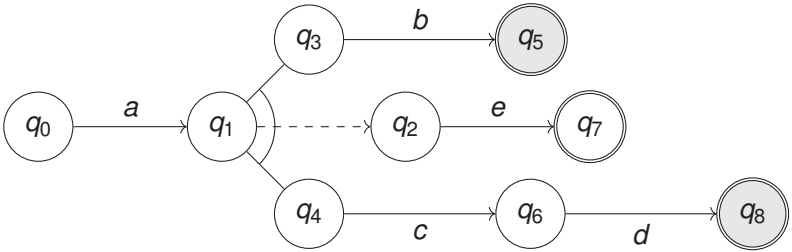
a

Pomset Automata



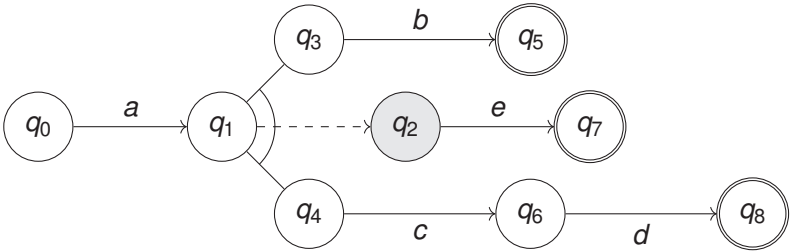
a

Pomset Automata



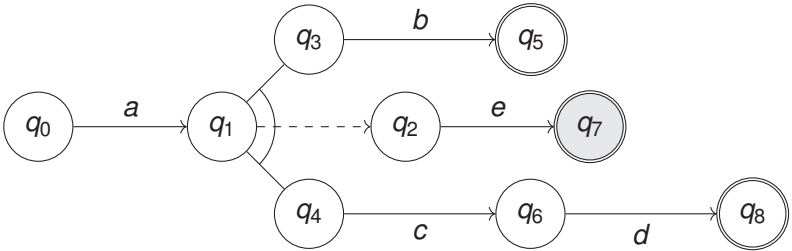
a

Pomset Automata



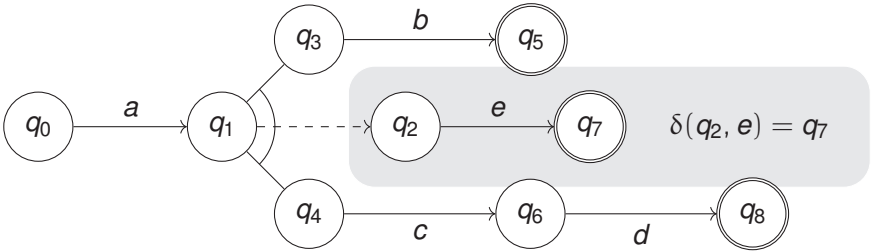
$$a(b \parallel cd)$$

Pomset Automata

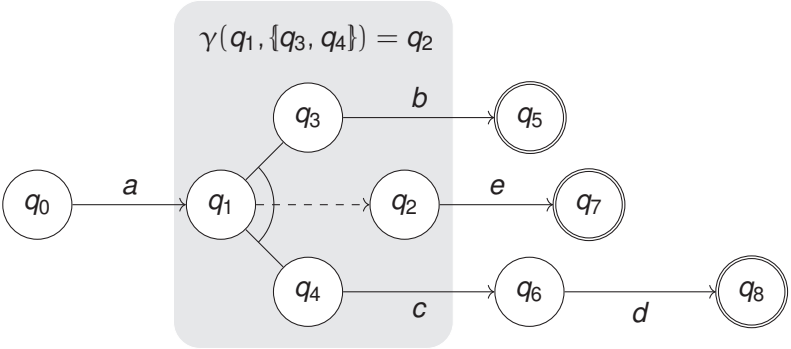


$$a(b \parallel cd)e$$

Pomset Automata



Pomset Automata



Pomset Automata

Definition

A *pomset automaton (PA)* is a tuple $\langle Q, \delta, \gamma, F \rangle$ with

- Q a set of *states*, $F \subseteq Q$ the *accepting states*,
- $\delta : Q \times \Sigma \to Q$, the *sequential transition function*,
- $\gamma : Q \times \mathcal{M}_\omega(Q) \to Q$, a the *parallel transition function*

Pomset Automata

Definition

$\twoheadrightarrow_A \subseteq Q \times \mathrm{Pom}_\Sigma \times Q$ is the smallest relation such that

$$\frac{}{q \xrightarrow{a}\!\!\!\!\twoheadrightarrow_A \delta(q,a)} \qquad \frac{q \xrightarrow{U}\!\!\!\!\twoheadrightarrow_A q'' \qquad q'' \xrightarrow{V}\!\!\!\!\twoheadrightarrow_A q'}{q \xrightarrow{U\cdot V}\!\!\!\!\twoheadrightarrow_A q'} \qquad \frac{r \xrightarrow{U}\!\!\!\!\twoheadrightarrow_A r' \in F \qquad s \xrightarrow{V}\!\!\!\!\twoheadrightarrow_A s' \in F}{q \xrightarrow{U \| V}\!\!\!\!\twoheadrightarrow_A \gamma(q, \{r,s\})}$$

Pomset Automata

Definition

$\twoheadrightarrow_A \subseteq Q \times \mathrm{Pom}_\Sigma \times Q$ is the smallest relation such that

$$\frac{}{q \xrightarrow{a}_{\twoheadrightarrow A} \delta(q,a)} \qquad \frac{q \xrightarrow{U}_{\twoheadrightarrow A} q'' \qquad q'' \xrightarrow{V}_{\twoheadrightarrow A} q'}{q \xrightarrow{U \cdot V}_{\twoheadrightarrow A} q'} \qquad \frac{r \xrightarrow{U}_{\twoheadrightarrow A} r' \in F \qquad s \xrightarrow{V}_{\twoheadrightarrow A} s' \in F}{q \xrightarrow{U \| V}_{\twoheadrightarrow A} \gamma(q, \{r,s\})}$$

$L_A(q)$ is defined by

$$L_A(q) = \{U : q \xrightarrow{U}_{\twoheadrightarrow A} q' \in F\} \cup \{1 : q \in F\}$$

Expressions to Automata

Brzozowski-construction:

1. $\mathcal{T}$ as state space

Expressions to Automata

Brzozowski-construction:

1. $\mathcal{T}$ as state space
2. $F_\Sigma \subseteq \mathcal{T}$: accepting expressions

Expressions to Automata

Brzozowski-construction:

1. $\mathcal{T}$ as state space
2. $F_\Sigma \subseteq \mathcal{T}$: accepting expressions
3. $\delta_\Sigma : \mathcal{T} \times \Sigma \to \mathcal{T}$: sequential derivatives

Expressions to Automata

Brzozowski-construction:

1. $\mathcal{T}$ as state space
2. $F_\Sigma \subseteq \mathcal{T}$: accepting expressions
3. $\delta_\Sigma : \mathcal{T} \times \Sigma \to \mathcal{T}$: sequential derivatives
4. $\gamma_\Sigma : \mathcal{T} \times \mathcal{M}_\omega(\mathcal{T}) \to \mathcal{T}$: parallel derivatives

Expressions to Automata

Brzozowski-construction:

1. $\mathcal{T}$ as state space
2. $F_\Sigma \subseteq \mathcal{T}$: accepting expressions
3. $\delta_\Sigma : \mathcal{T} \times \Sigma \to \mathcal{T}$: sequential derivatives
4. $\gamma_\Sigma : \mathcal{T} \times \mathcal{M}_\omega(\mathcal{T}) \to \mathcal{T}$: parallel derivatives
5. trim to finite automaton as necessary

Expressions to Automata

Brzozowski-construction:

1. $\mathcal{T}$ as state space
2. $F_\Sigma \subseteq \mathcal{T}$: accepting expressions
3. $\delta_\Sigma : \mathcal{T} \times \Sigma \to \mathcal{T}$: sequential derivatives
4. $\gamma_\Sigma : \mathcal{T} \times \mathcal{M}_\omega(\mathcal{T}) \to \mathcal{T}$: parallel derivatives
5. trim to finite automaton as necessary

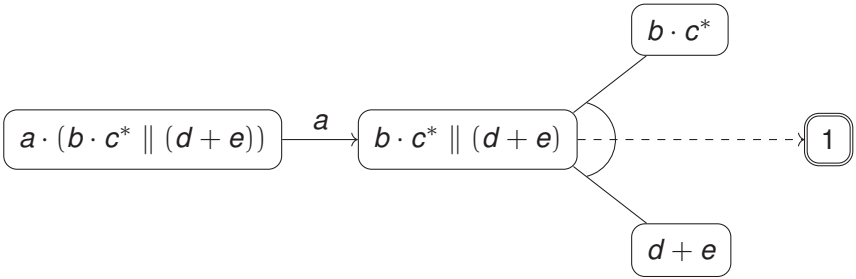
We write $L_\Sigma(e)$ for $L_{A_\Sigma}(e)$.

Expressions to Automata

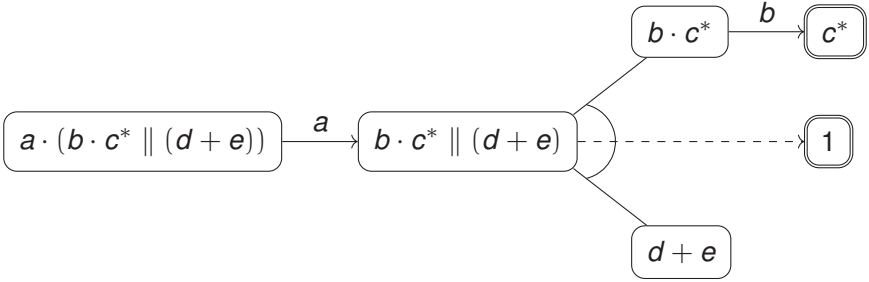
$$a \cdot (b \cdot c^* \parallel (d + e))$$

Expressions to Automata

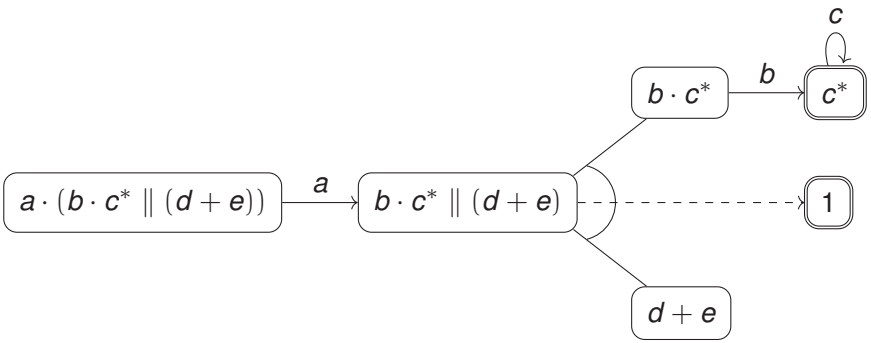
Expressions to Automata



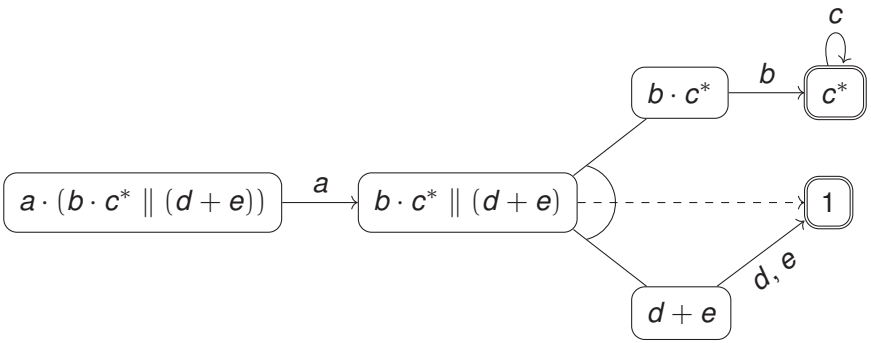
Expressions to Automata



Expressions to Automata



Expressions to Automata



Expressions to Automata

Theorem

Let $e \in \mathcal{T}$; then $L_\Sigma(e) = [\![e]\!]$.

Expressions to Automata

Theorem

Let $e \in \mathcal{T}$; then $L_\Sigma(e) = [\![e]\!]$.

Proof in two parts:

Expressions to Automata

Theorem

Let $e \in \mathcal{T}$; then $L_\Sigma(e) = [\![e]\!]$.

Proof in two parts:

- Deconstruction: if $e \xrightarrow{U}_{\twoheadrightarrow_\Sigma} f$, then ...

Expressions to Automata

Theorem

Let $e \in \mathcal{T}$; then $L_\Sigma(e) = [\![e]\!]$.

Proof in two parts:

- Deconstruction: if $e \xrightarrow{U}\!\!\!\!\rightarrow_\Sigma f$, then ...
- Construction: if ..., then $e \xrightarrow{U}\!\!\!\!\rightarrow_\Sigma f$.

Expressions to Automata

Theorem

Let $e \in \mathfrak{T}$; then $L_\Sigma(e) = [\![e]\!]$.

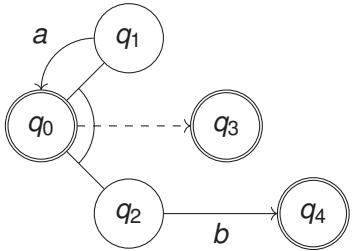
Proof in two parts:

- Deconstruction: if $e \xrightarrow[\twoheadrightarrow_\Sigma]{U} f$, then ...
- Construction: if ..., then $e \xrightarrow[\twoheadrightarrow_\Sigma]{U} f$.

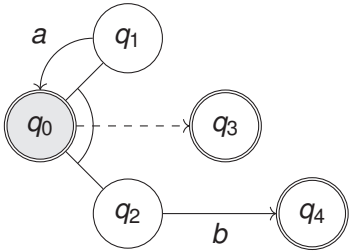
Theorem

Let $e \in \mathfrak{T}$; we find a finite A with a state q such that $L_A(q) = L_\Sigma(e)$.

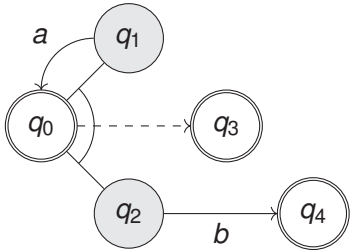
Automata to Expressions



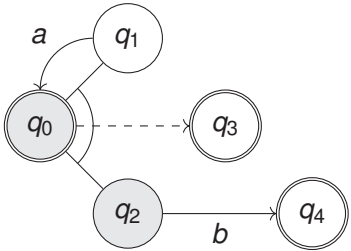
Automata to Expressions



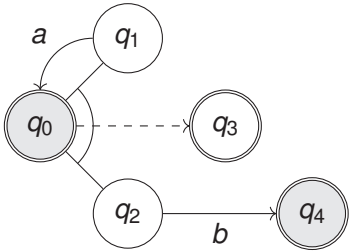
Automata to Expressions



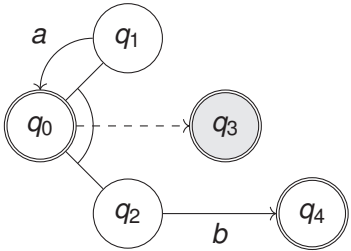
Automata to Expressions



Automata to Expressions

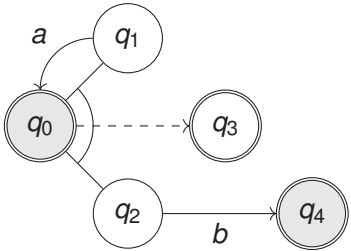


Automata to Expressions

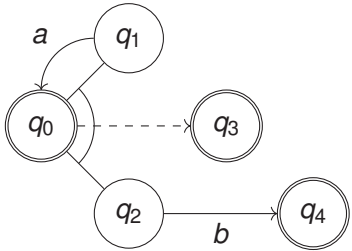


$$a \parallel b$$

Automata to Expressions

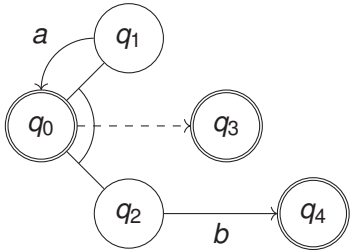


Automata to Expressions



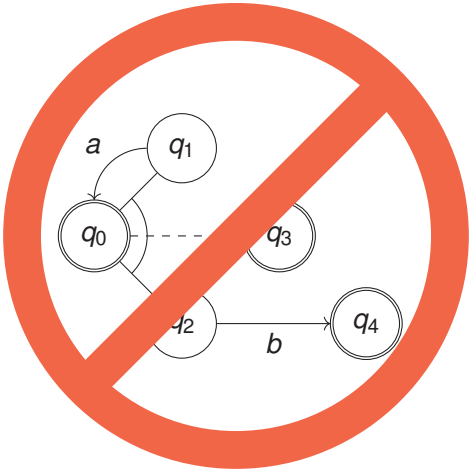
$$a(a \parallel b) \parallel b$$

Automata to Expressions

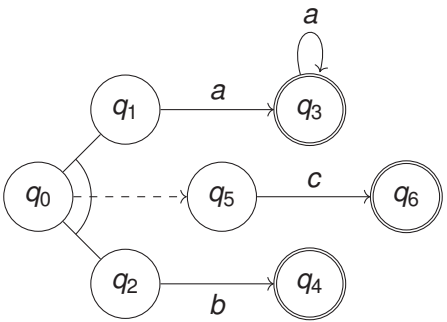


$$a(a(a \parallel b)) \parallel b$$

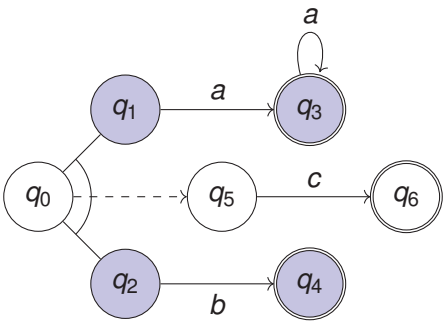
Automata to Expressions



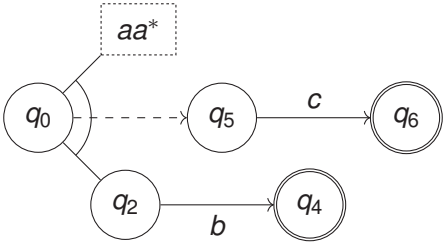
Automata to Expressions



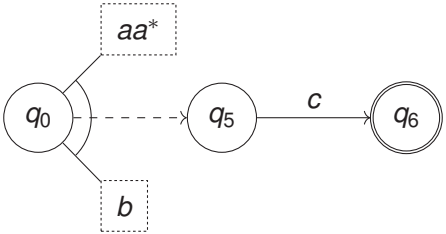
Automata to Expressions



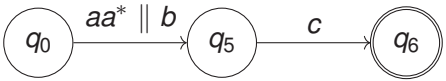
Automata to Expressions



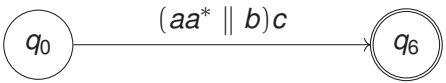
Automata to Expressions



Automata to Expressions



Automata to Expressions



Further Work

Extend to WCKA: exchange law.

[2] Anderson et al. 2014.

Further Work

Extend to WCKA: exchange law.

Possible applications

- Completeness proof based on automata.
- Equivalence checking of WBKA expressions.

[2] Anderson et al. 2014.

Further Work

Extend to WCKA: exchange law.

Possible applications

- Completeness proof based on automata.
- Equivalence checking of WBKA expressions.

Endgame: concurrent extension of NetKAT.[2]

[2] Anderson et al. 2014.