Deterministic Set Automata

Martin Kutrib Andreas Malcher Matthias Wendlandt

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
email: {kutrib,malcher,matthias.wendlandt}@informatik.uni-giessen.de

DLT 2014, Ekaterinburg, Russia

Finite Automata and Storage Media

Deterministic finite automata are the basic and standard model in language theory.

Finite Automata and Storage Media

Deterministic finite automata are the basic and standard model in language theory.

- ➜ Closure under many operations.

Finite Automata and Storage Media

Deterministic finite automata are the basic and standard model in language theory.

- ➜ Closure under many operations.
- ➜ Decidability of almost all decidability questions and, furthermore, often solvable in polynomial time.

Finite Automata and Storage Media

Deterministic finite automata are the basic and standard model in language theory.

- ➜ Closure under many operations.
- ➜ Decidability of almost all decidability questions and, furthermore, often solvable in polynomial time.
- ➜ Efficient minimization algorithms exist.

Finite Automata and Storage Media

Deterministic finite automata are the basic and standard model in language theory.

- ➜ Closure under many operations.
- ➜ Decidability of almost all decidability questions and, furthermore, often solvable in polynomial time.
- ➜ Efficient minimization algorithms exist.
- ➜ The computational power is weak since only regular languages are accepted.

Finite Automata and Storage Media

Study deterministic finite automata extended with some storage medium.

Finite Automata and Storage Media

Study deterministic finite automata extended with some storage medium.

- ➜ Read/write tape: Turing machines and LBA

Finite Automata and Storage Media

Study deterministic finite automata extended with some storage medium.

- ➜ Read/write tape: Turing machines and LBA
- ➜ Pushdown store, LIFO: Pushdown automata/context-free languages

Finite Automata and Storage Media

Study deterministic finite automata extended with some storage medium.

- ➜ Read/write tape: Turing machines and LBA
- ➜ Pushdown store, LIFO: Pushdown automata/context-free languages
- ➜ Pushdown store + pointer: Stack automata

Finite Automata and Storage Media

Study deterministic finite automata extended with some storage medium.

- ➜ Read/write tape: Turing machines and LBA
- ➜ Pushdown store, LIFO: Pushdown automata/context-free languages
- ➜ Pushdown store + pointer: Stack automata
- ➜ Queue store, FIFO: Queue automata (restricted to quasi real-time or finite turn)

Finite Automata and Storage Media

Study deterministic finite automata extended with some storage medium.

- Read/write tape: Turing machines and LBA
- Pushdown store, LIFO: Pushdown automata/context-free languages
- Pushdown store + pointer: Stack automata
- Queue store, FIFO: Queue automata (restricted to quasi real-time or finite turn)
- ...

Finite Automata with Sets

Study finite automata extended with a set.

Finite Automata with Sets

Study finite automata extended with a set.

- ➜ Bag automata [Daley, Eramian, McQuillan 2008]: It is possible to store multisets of symbols.

Finite Automata with Sets

Study finite automata extended with a set.

- ➜ Bag automata [Daley, Eramian, McQuillan 2008]: It is possible to store multisets of symbols.
- ➜ In 1995 [Lange, Reinhardt] introduced a model which allows to store and test words in a set.

Finite Automata with Sets

Study finite automata extended with a set.

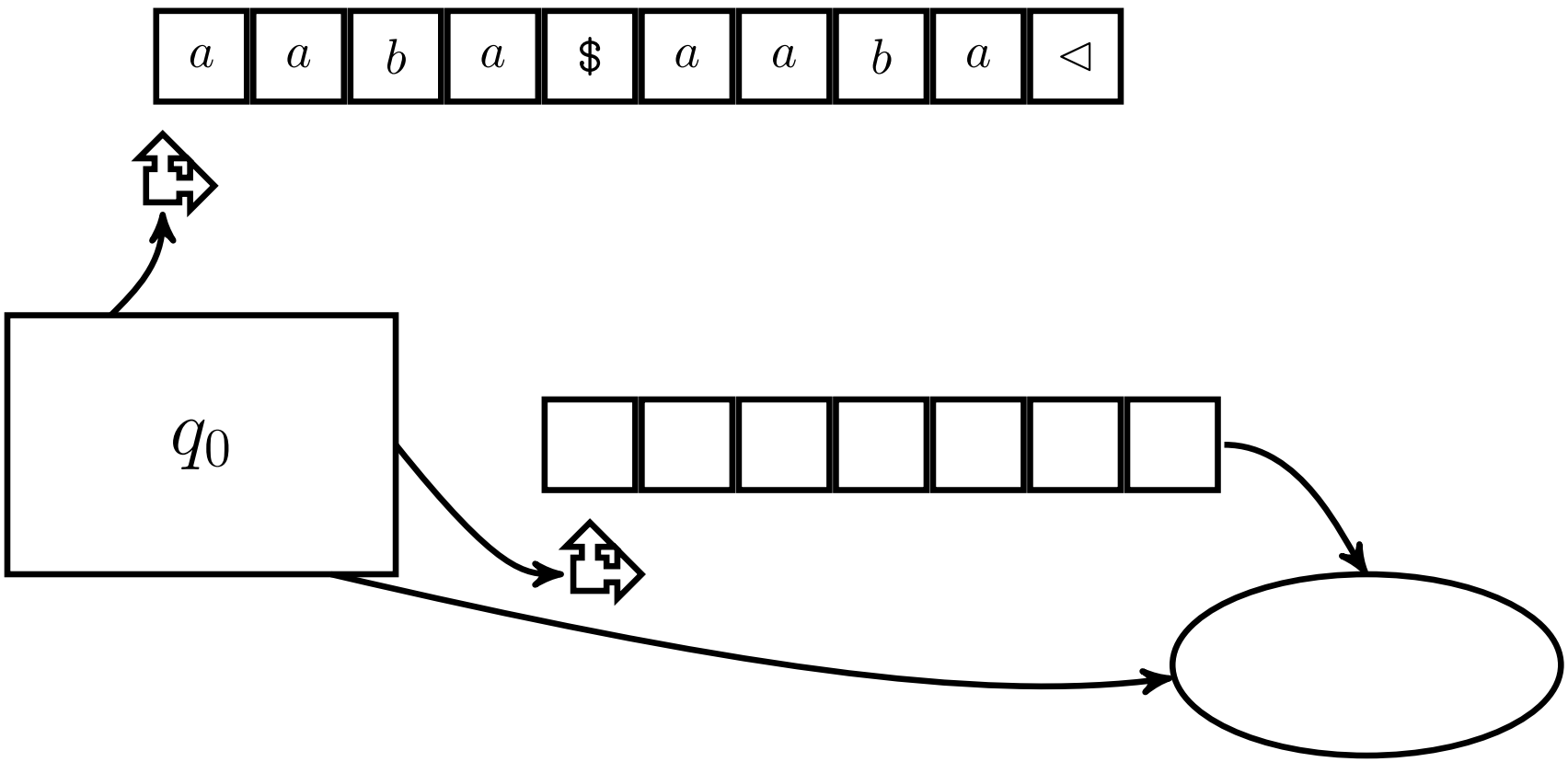
- ➜ Bag automata [Daley, Eramian, McQuillan 2008]: It is possible to store multisets of symbols.
- ➜ In 1995 [Lange, Reinhardt] introduced a model which allows to store and test words in a set.
- ➜ It works nondeterministically, allows no remove operations on the set, and the test operation implicitly adds the word tested to the set.

Finite Automata with Sets

Study finite automata extended with a set.

- ➜ Bag automata [Daley, Eramian, McQuillan 2008]: It is possible to store multisets of symbols.
- ➜ In 1995 [Lange, Reinhardt] introduced a model which allows to store and test words in a set.
- ➜ It works nondeterministically, allows no remove operations on the set, and the test operation implicitly adds the word tested to the set.
- ➜ Closure properties of the model, the emptiness problem, and the word problem are studied.

Deterministic Set Automata – Example



Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (S \times \{\mathtt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (S \times \{\mathtt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (S \times \{\mathtt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (S \times \{\mathtt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (S \times \{\mathtt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\texttt{in}, \texttt{out}\})) \cup (S \times \{\texttt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\texttt{in}, \texttt{out}\})) \cup (S \times \{\texttt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

Definition

A deterministic set automaton (DSA) is a system $M = \langle S, \Sigma, \Gamma, \lhd, \delta, s_0, F \rangle$, where

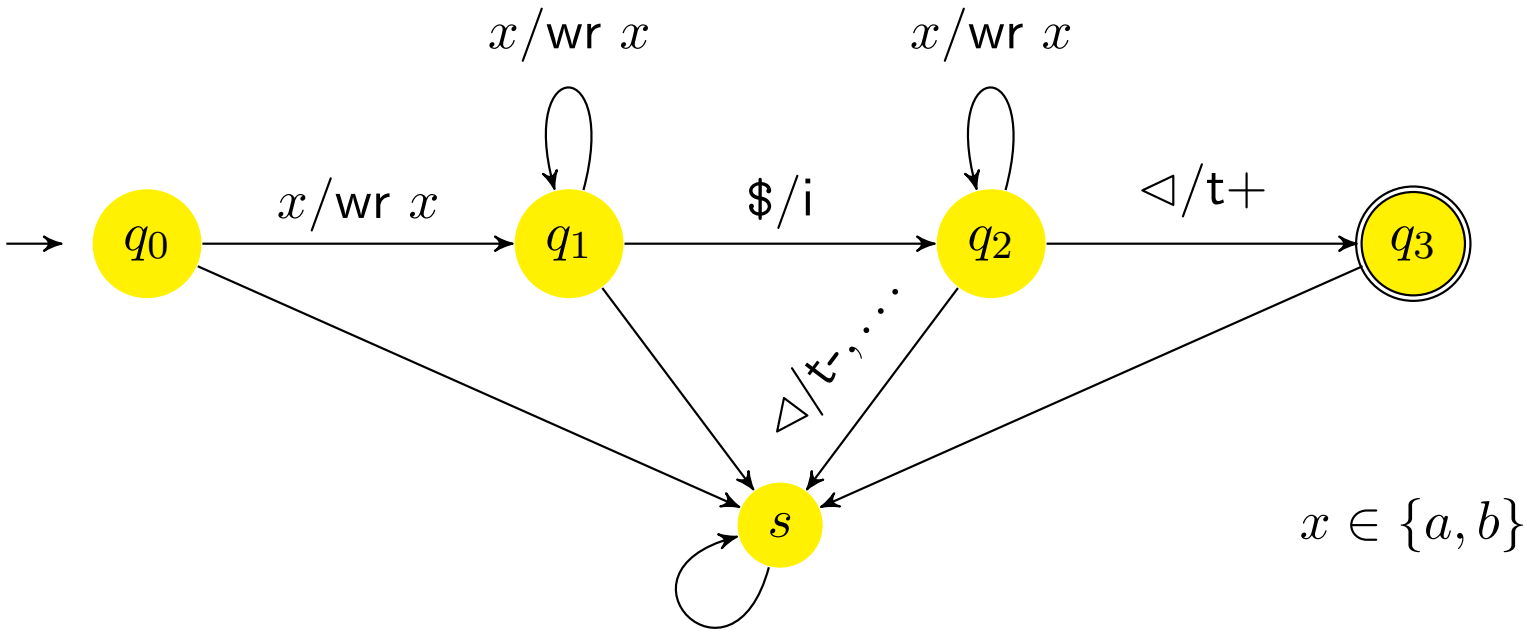
- S is the finite set of internal states,
- Σ is the finite set of input symbols,
- Γ is the finite set of tape symbols,
- $\lhd \notin \Sigma$ is the right endmarker,
- $s_0 \in S$ is the initial state,
- $F \subseteq S$ is the set of accepting states, and
- δ is the partial transition function where:

$$\delta : S \times (\Sigma \cup \{\lambda, \lhd\}) \to (S \times (\Gamma^* \cup \{\mathtt{in}, \mathtt{out}\})) \cup (S \times \{\mathtt{test}\} \times S)$$

- The machine accepts if the whole input is read and an accepting state is entered.

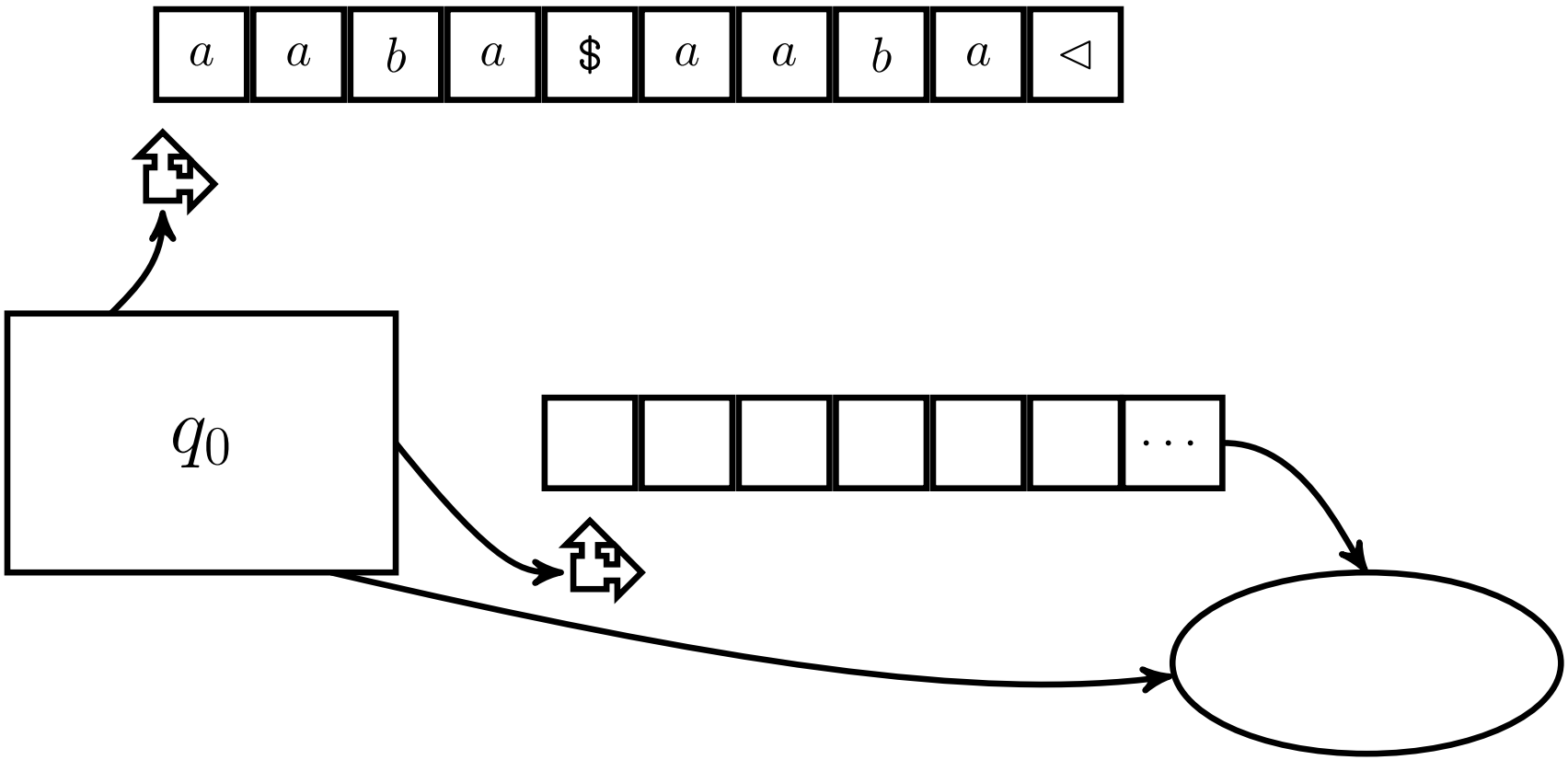
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a,b\}^+ \,\}$$



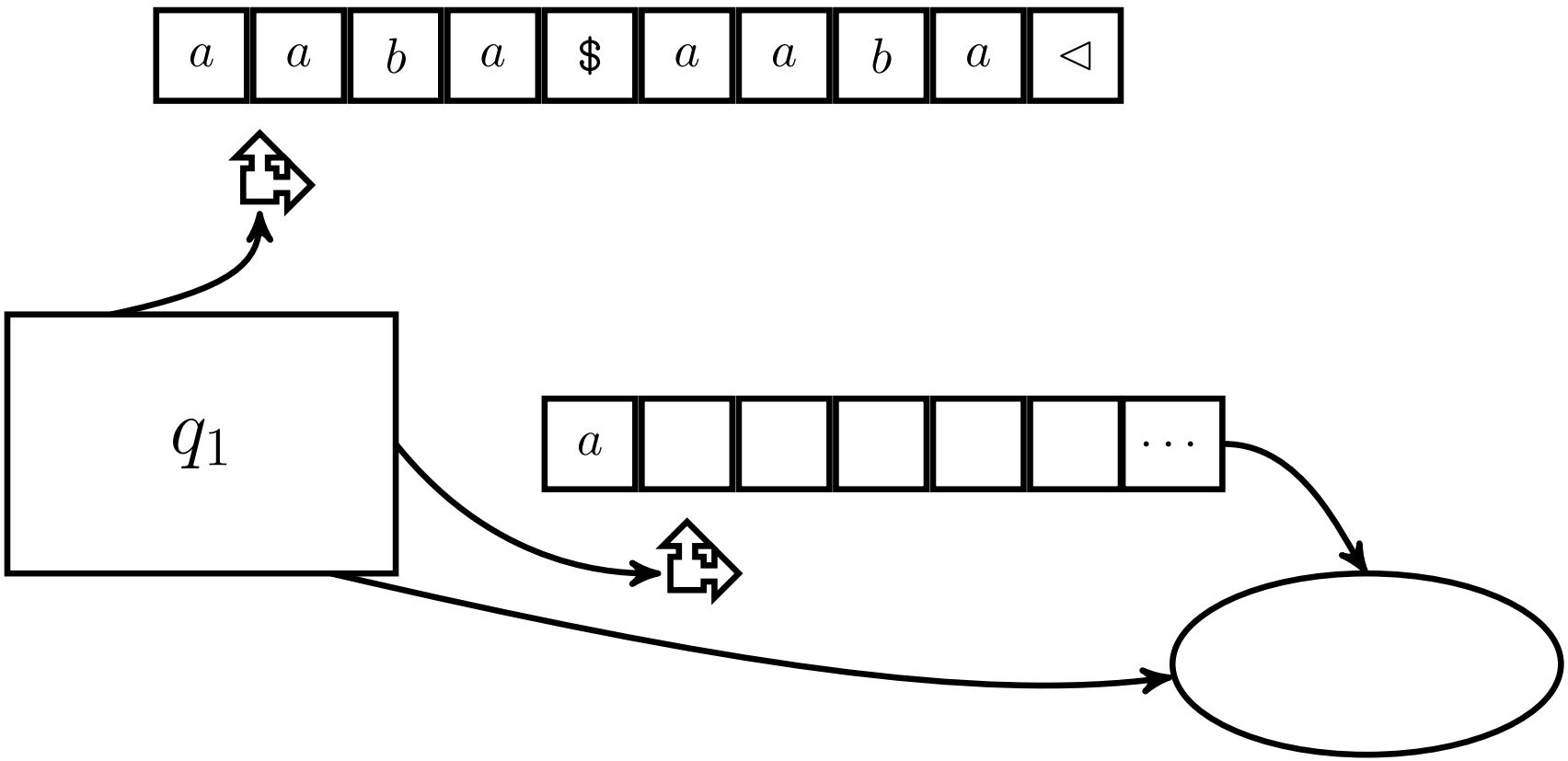
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



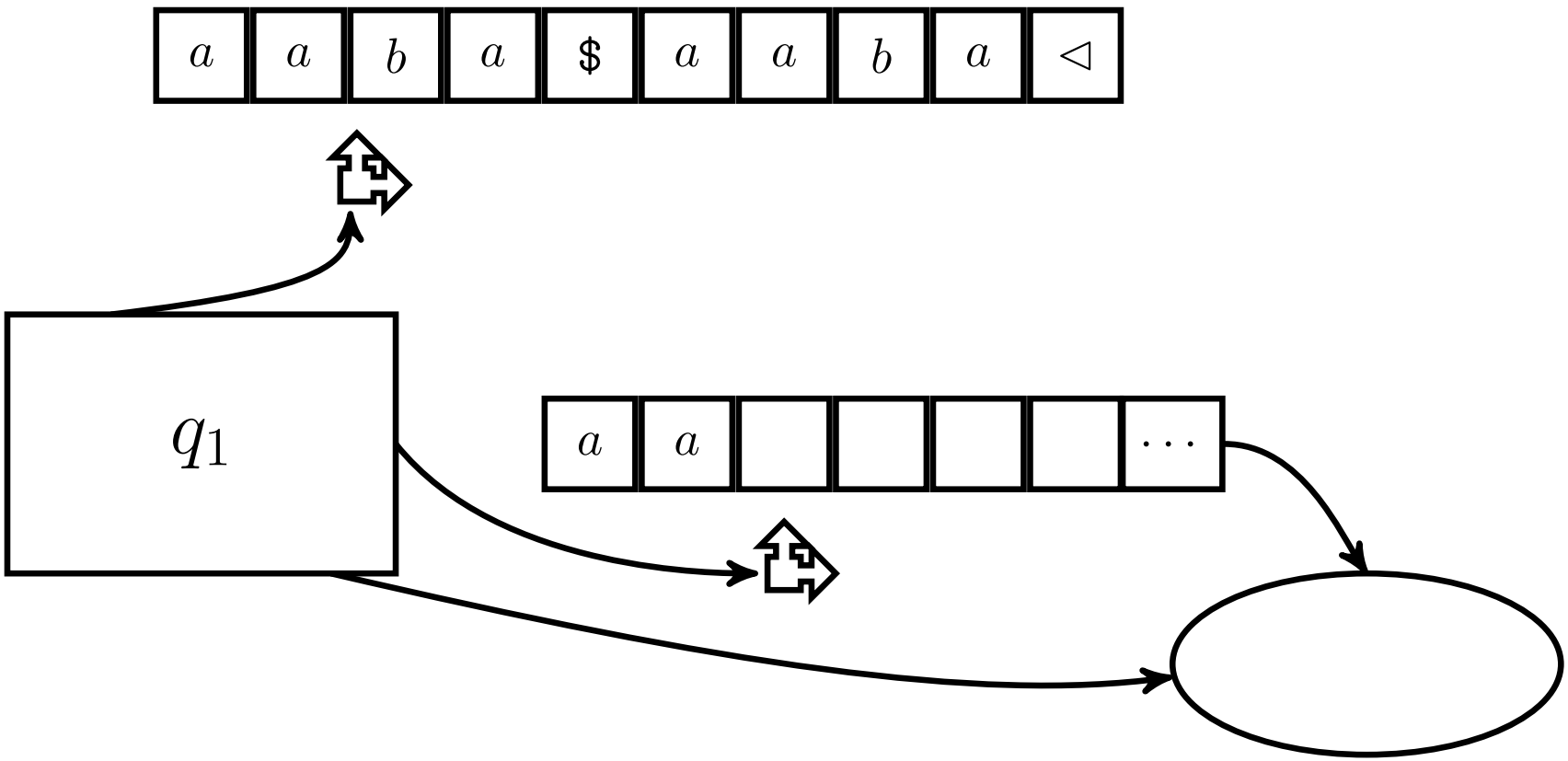
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



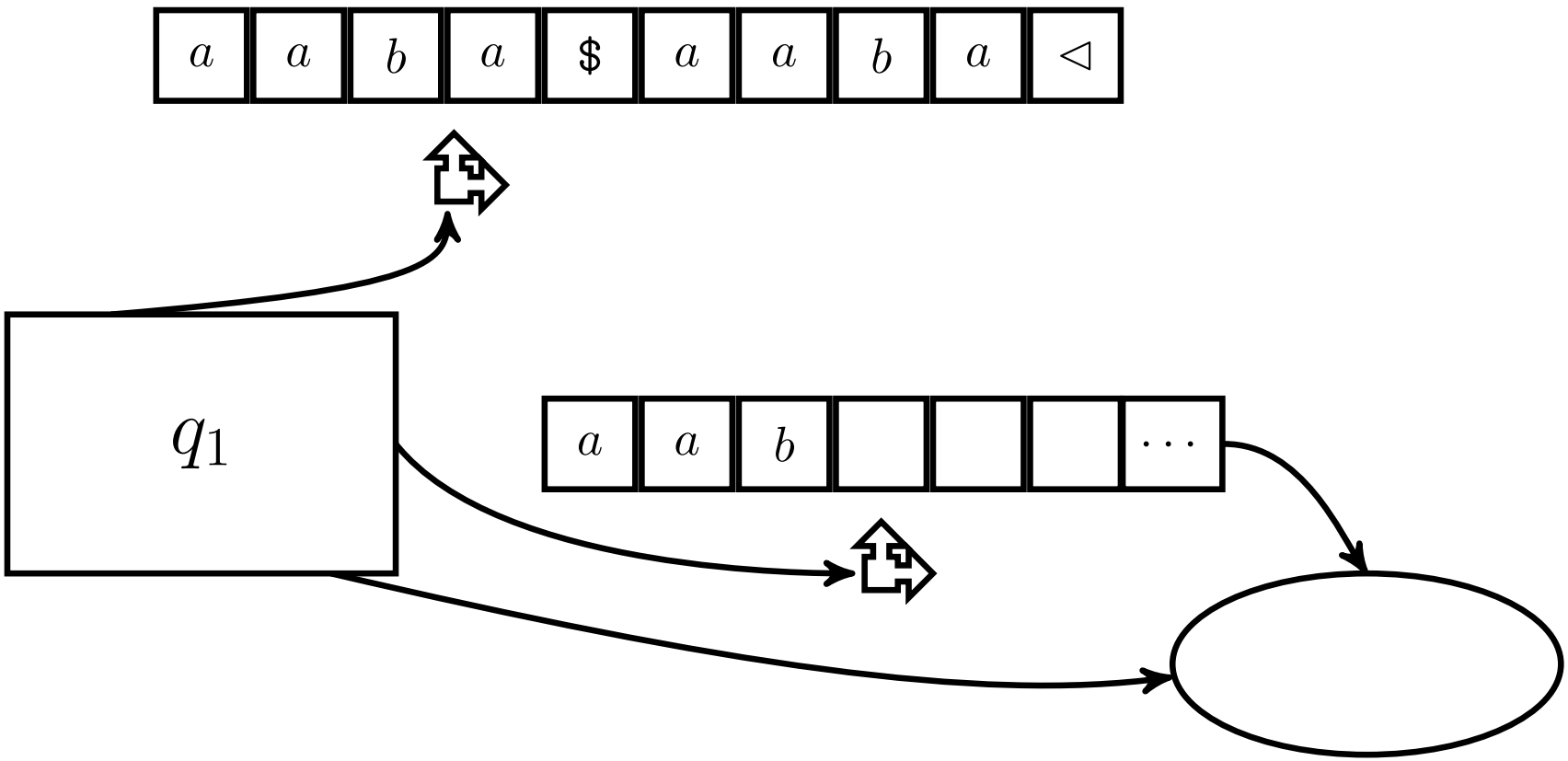
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



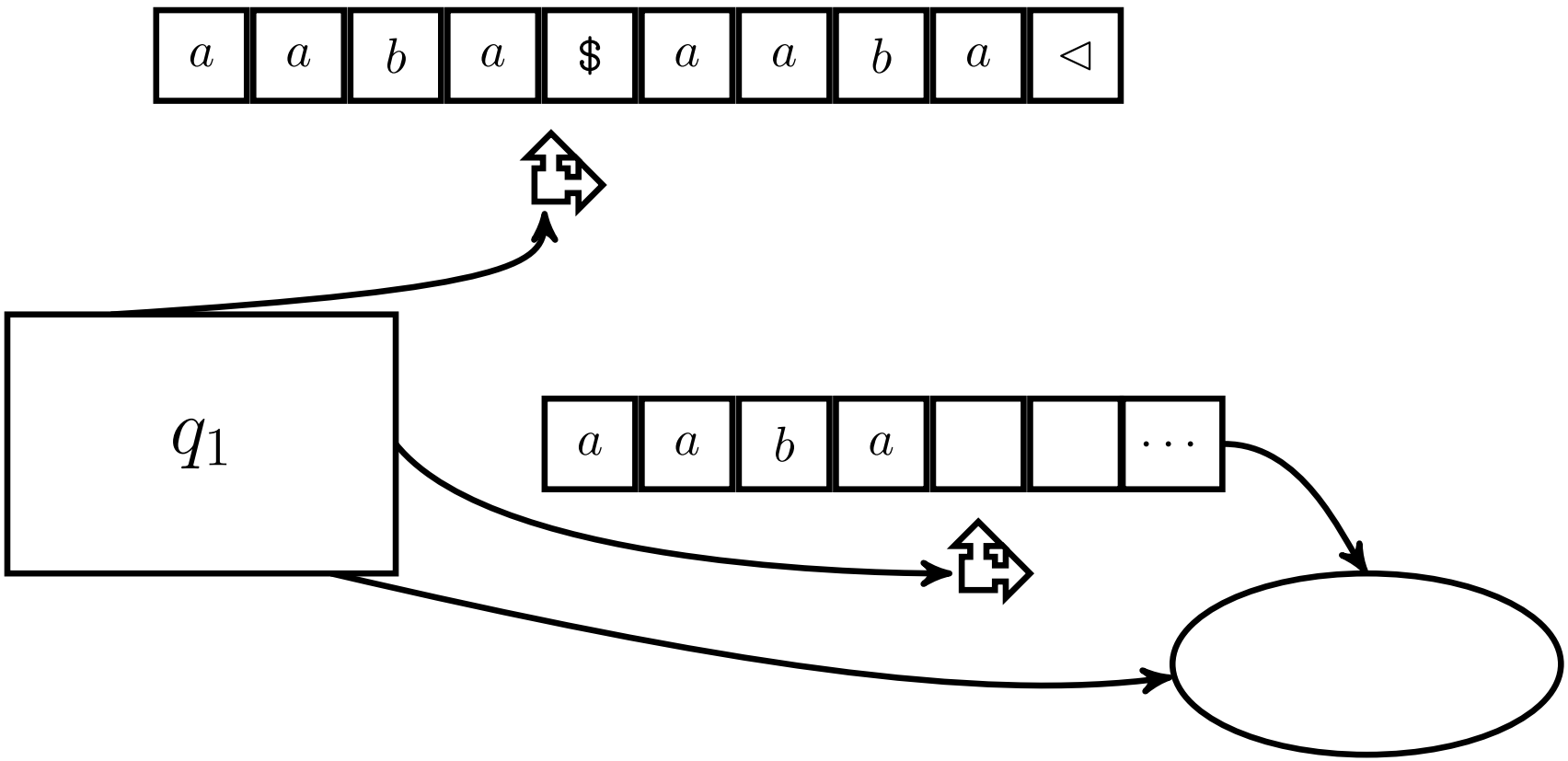
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



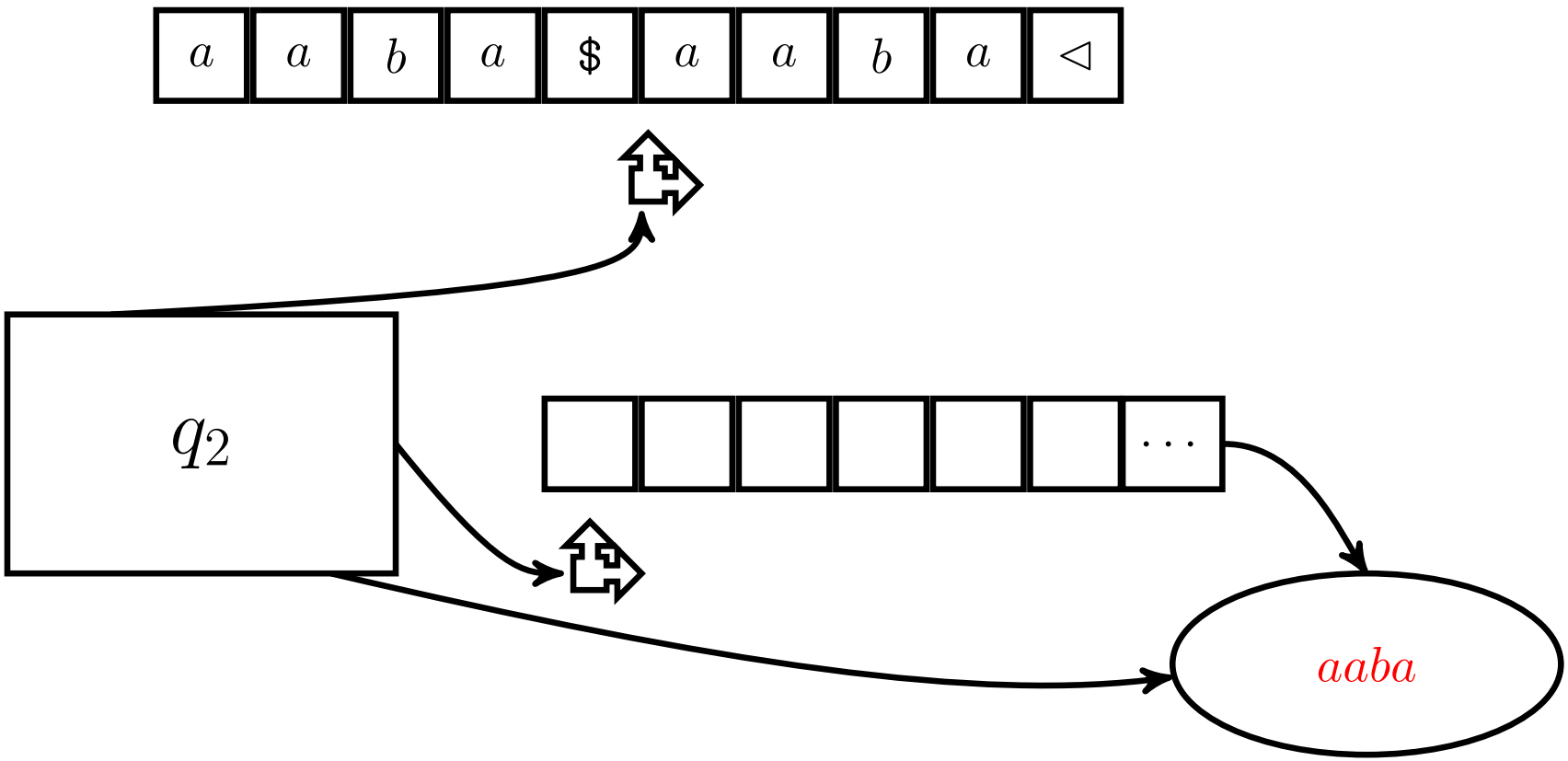
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



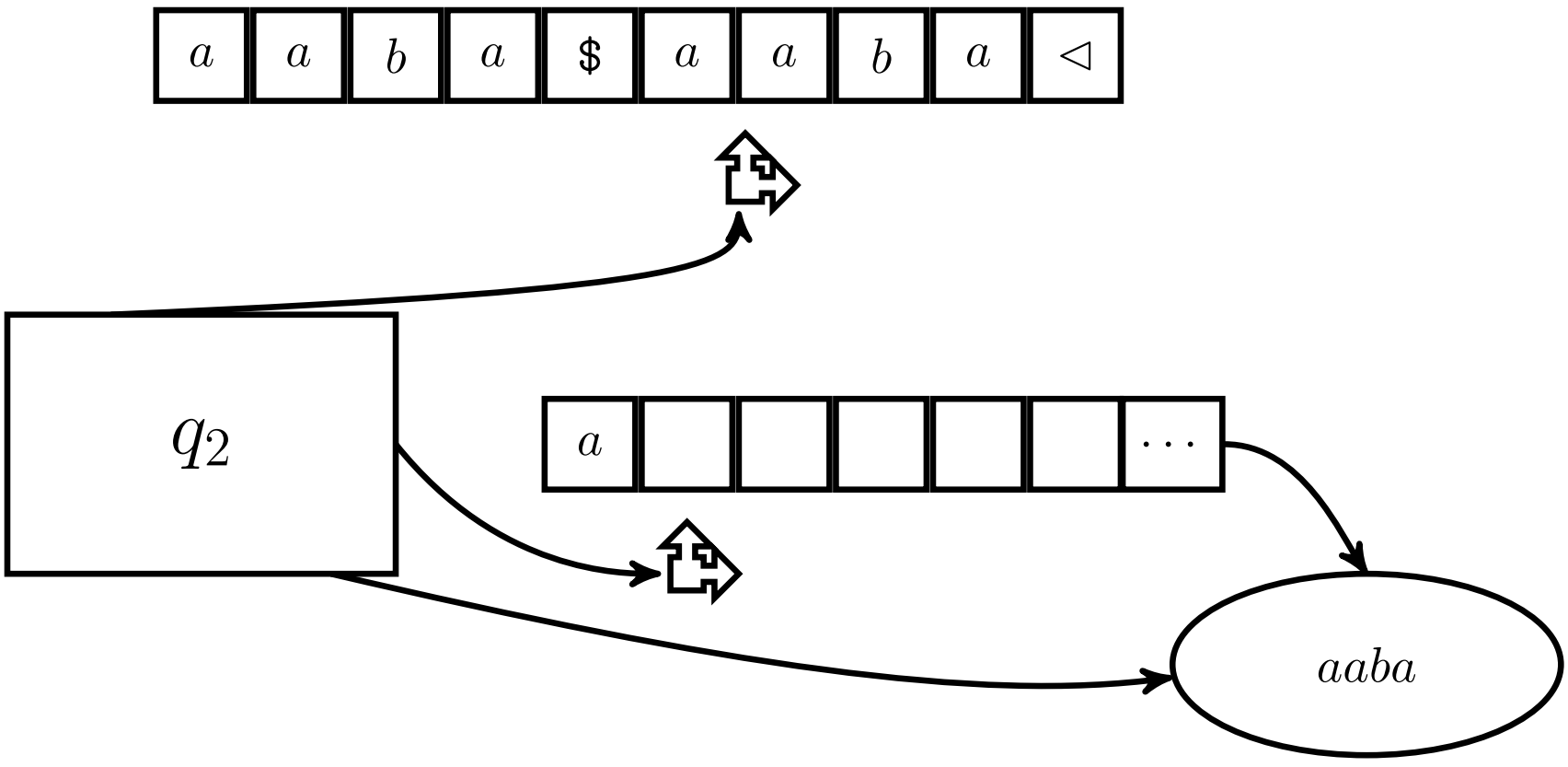
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



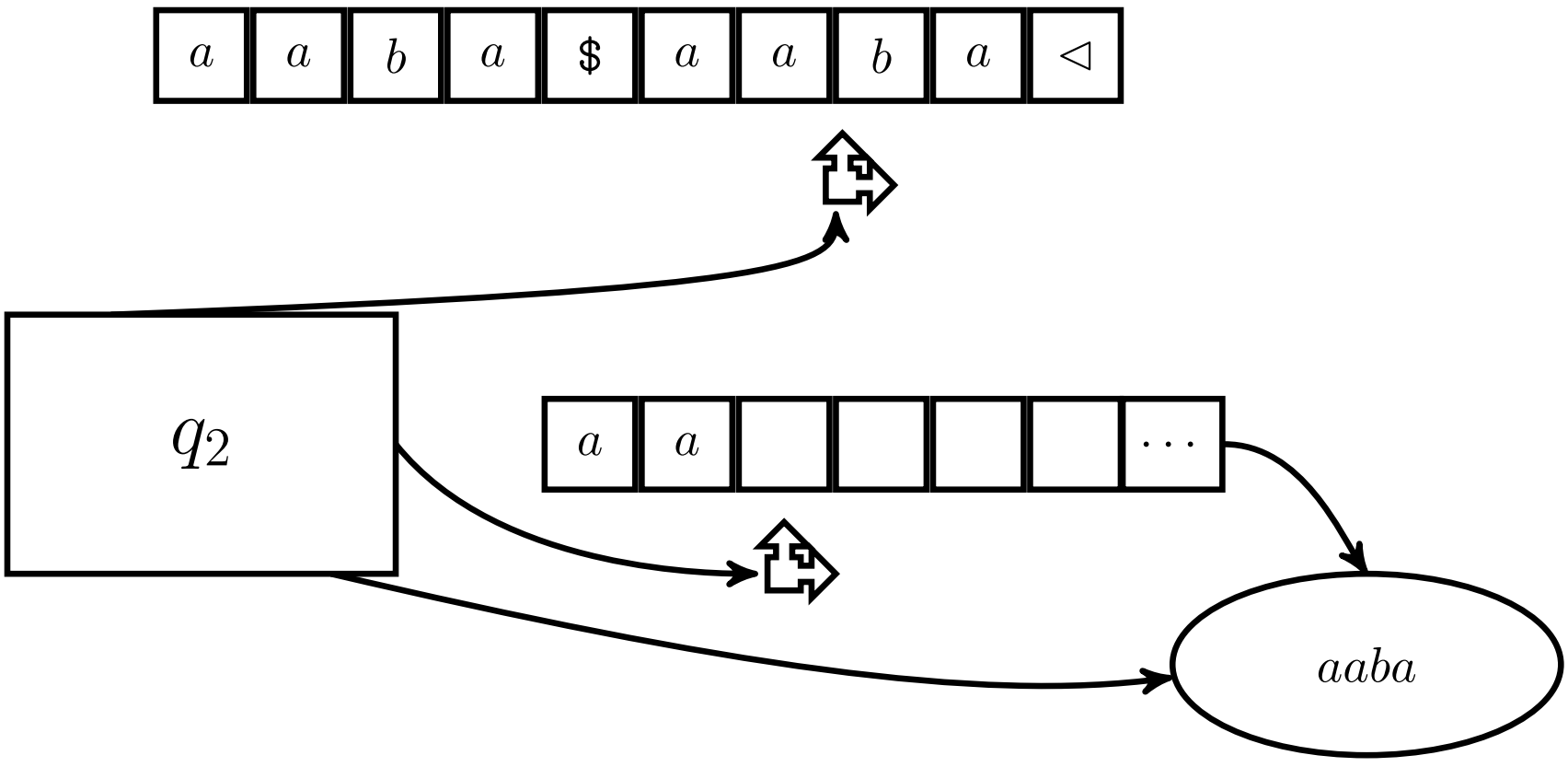
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



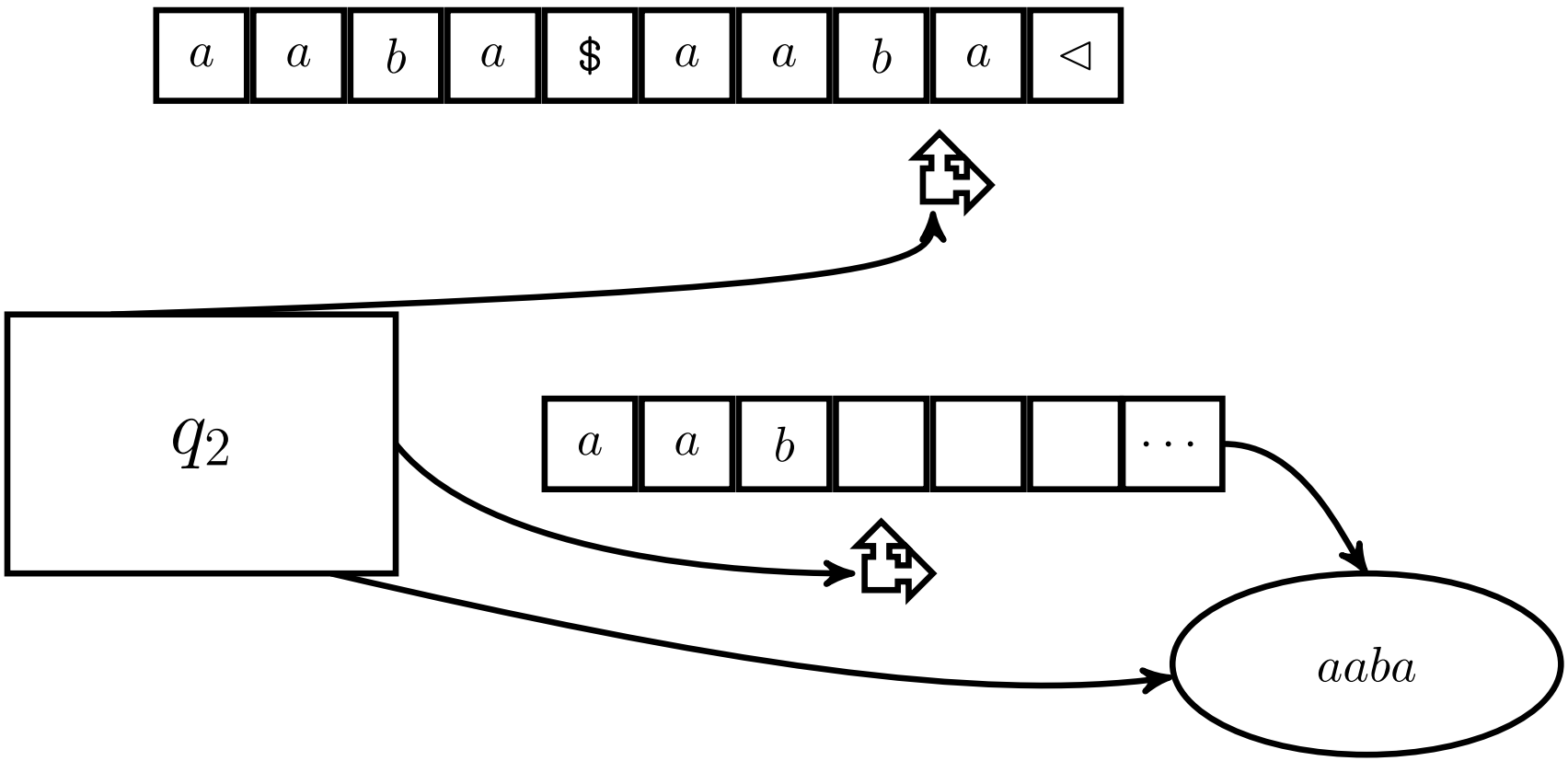
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



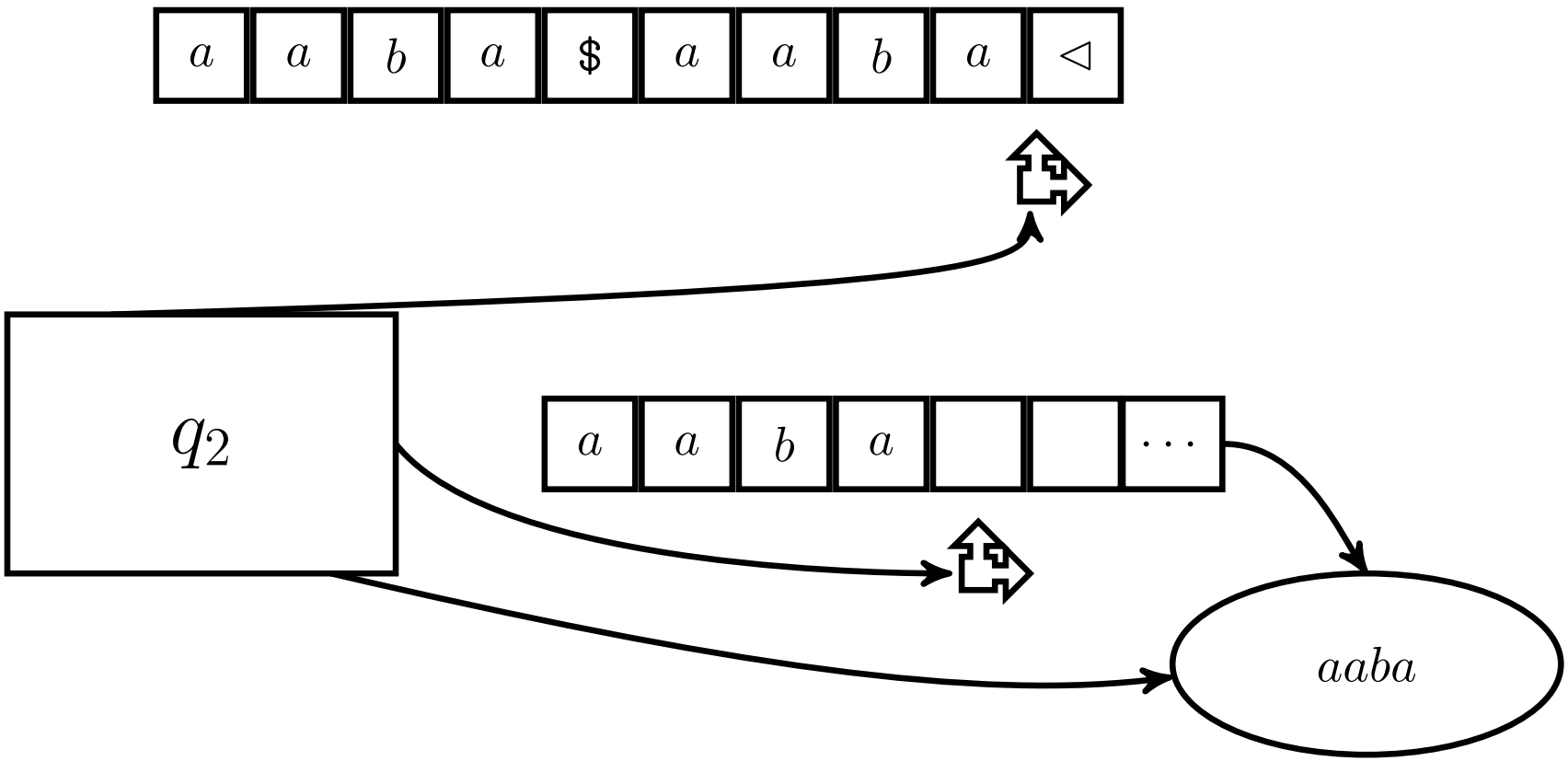
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



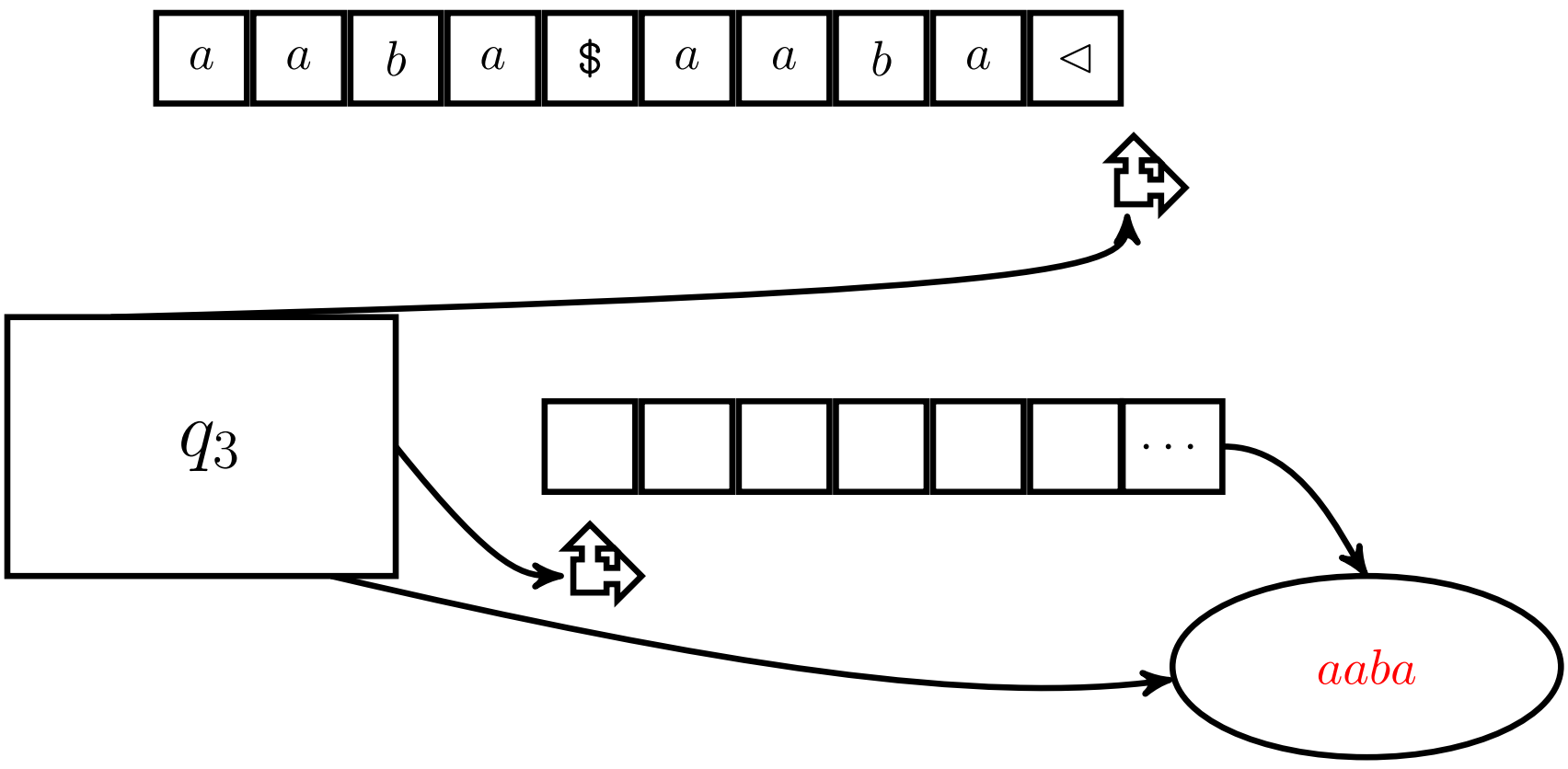
Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



Deterministic Set Automata – Example

$$L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$$



Deterministic Set Automata – Examples

The following languages are accepted by DSA:

- $L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$

Deterministic Set Automata – Examples

The following languages are accepted by DSA:

- $L_1 = \{\, w\$w \mid w \in \{a,b\}^+ \,\}$
- $L_2 = \{\, a^n b^m \$_0 c^n \mid m,n \geq 1 \,\} \cup \{\, a^n b^m \$_1 c^m \mid m,n \geq 1 \,\}$

Deterministic Set Automata – Examples

The following languages are accepted by DSA:

- $L_1 = \{ w\$w \mid w \in \{a, b\}^+ \}$
- $L_2 = \{ a^n b^m \$_0 c^n \mid m, n \geq 1 \} \cup \{ a^n b^m \$_1 c^m \mid m, n \geq 1 \}$
- $L_3 = \{ a^n b^n c^n \mid n \geq 1 \}$

Computational Power

Theorem

Every unary language accepted by a DSA is regular.

Computational Power

Theorem

Every unary language accepted by a DSA is regular.

Theorem

The family of languages accepted by DSA is incomparable with the family of languages accepted by quasi-real-time queue automata.

Computational Power

Theorem

Every unary language accepted by a DSA is regular.

Theorem

The family of languages accepted by DSA is incomparable with the family of languages accepted by quasi-real-time queue automata.

- The non-regular language $\{ a^n \mid n \text{ is a Fibonacci number} \}$ is accepted by some quasi-real-time queue automaton.
- $L_2 = \{ a^n b^m \$_0 c^n \mid m, n \geq 1 \} \cup \{ a^n b^m \$_1 c^m \mid m, n \geq 1 \}$ is not accepted by any quasi-real-time queue automaton.

DSA – **Infinite Action Normal Form**

- The initial state is only visited once at the beginning of the computation.

DSA – **Infinite Action Normal Form**

- The initial state is only visited once at the beginning of the computation.
- Each other state indicates uniquely which action the automaton did in the last computation step, so they are marked by `t+`, `t-`, `i`, or `o`.

DSA – Infinite Action Normal Form

- The initial state is only visited once at the beginning of the computation.
- Each other state indicates uniquely which action the automaton did in the last computation step, so they are marked by `t+`, `t-`, `i`, or `o`.
- Non-marked states are interpreted as states where the last action was a write operation on the tape.

DSA – Infinite Action Normal Form

- The initial state is only visited once at the beginning of the computation.
- Each other state indicates uniquely which action the automaton did in the last computation step, so they are marked by `t+`, `t-`, `i`, or `o`.
- Non-marked states are interpreted as states where the last action was a write operation on the tape.
- All sets L_{s_i,s_j} with s_i and s_j non-writing states that describe all words that can be written on the tape when the computation starts in state s_i with empty tape and ends in state s_j, and in between no other state performing an operation on the set is entered are infinite.

Computational Power (2)

Theorem

The family of languages accepted by DSA is incomparable with the deterministic context-free languages.

Computational Power (2)

Theorem

The family of languages accepted by DSA is incomparable with the deterministic context-free languages.

- $L_1 = \{\, w\$w \mid w \in \{a, b\}^+ \,\}$ is not context free.

Computational Power (2)

Theorem

The family of languages accepted by DSA is incomparable with the deterministic context-free languages.

- $L_1 = \{\, w\$w \mid w \in \{a,b\}^+ \,\}$ is not context free.
- $L = \{\, w\$w^R \mid w \in \{a,b\}^+ \,\}$ is deterministic context free, but is not accepted by any DSA.

Computational Power (2)

Theorem

The family of languages accepted by DSA is incomparable with the deterministic context-free languages.

- $L_1 = \{\, w\$w \mid w \in \{a,b\}^+ \,\}$ is not context free.
- $L = \{\, w\$w^R \mid w \in \{a,b\}^+ \,\}$ is deterministic context free, but is not accepted by any DSA.
- First step: If L is accepted by some DSA, then all possible set operations performed on the first part of the input are a finite number of `in`-operations, and on the second part are a finite number of `test`-operations.

Computational Power (2)

Theorem

The family of languages accepted by DSA is incomparable with the deterministic context-free languages.

- $L_1 = \{ w\$w \mid w \in \{a, b\}^+ \}$ is not context free.
- $L = \{ w\$w^R \mid w \in \{a, b\}^+ \}$ is deterministic context free, but is not accepted by any DSA.
- First step: If L is accepted by some DSA, then all possible set operations performed on the first part of the input are a finite number of `in`-operations, and on the second part are a finite number of `test`-operations.
- Second step: Construct an equivalent one-way multi-head finite automaton accepting L.

Computational Power (3)

Theorem

The family of languages accepted by DSA is incomparable with the family of languages accepted by queue automata with finite turns.

Computational Power (3)

Theorem

The family of languages accepted by DSA is incomparable with the family of languages accepted by queue automata with finite turns.

- $L_3 = \{ a^n b^n c^n \mid n \geq 1 \}$ is not accepted by any finite-turn queue automaton.

Computational Power (3)

Theorem

The family of languages accepted by DSA is incomparable with the family of languages accepted by queue automata with finite turns.

- $L_3 = \{ a^n b^n c^n \mid n \geq 1 \}$ is not accepted by any finite-turn queue automaton.
- Consider the union $L = L' \cup L''$ with
  $L' = \{ a^n b^m c^n \mid m, n \geq 1 \}$ and
  $L'' = \{ a^n b^m c^{n+m} \mid m, n \geq 1 \}$.

Computational Power (3)

Theorem

The family of languages accepted by DSA is incomparable with the family of languages accepted by queue automata with finite turns.

- $L_3 = \{ a^n b^n c^n \mid n \geq 1 \}$ is not accepted by any finite-turn queue automaton.
- Consider the union $L = L' \cup L''$ with
  $L' = \{ a^n b^m c^n \mid m, n \geq 1 \}$ and
  $L'' = \{ a^n b^m c^{n+m} \mid m, n \geq 1 \}$.
- L is accepted by a queue automaton with one turn, but L is not accepted by any DSA.

Closure Properties

Lemma

The family of languages accepted by DSA is closed under complementation.

Closure Properties

Lemma

The family of languages accepted by DSA is closed under complementation.

➜ Idea: Interchange accepting and non-accepting states.

Closure Properties

Lemma

The family of languages accepted by DSA is closed under complementation.

- ➜ Idea: Interchange accepting and non-accepting states.
- ➜ Problems: (1) no next move is defined, (2) an infinite λ-loop is entered, (3) λ-steps leading from an accepting state to a rejecting state and back.

Closure Properties

Lemma

The family of languages accepted by DSA is closed under complementation.

- Idea: Interchange accepting and non-accepting states.
- Problems: (1) no next move is defined, (2) an infinite λ-loop is entered, (3) λ-steps leading from an accepting state to a rejecting state and back.
- Use additional states and infinite action normal form.

Closure Properties

Lemma

The family of languages accepted by DSA is closed under complementation.

- ➜ Idea: Interchange accepting and non-accepting states.
- ➜ Problems: (1) no next move is defined, (2) an infinite λ-loop is entered, (3) λ-steps leading from an accepting state to a rejecting state and back.
- ➜ Use additional states and infinite action normal form.

Lemma

The family of languages accepted by DSA is not closed under union and intersection, but is closed under intersection with regular languages and under union with regular languages.

Decidability Questions

Theorem

It is decidable whether a given deterministic set automaton accepts the empty language.

Decidability Questions

Theorem

It is decidable whether a given deterministic set automaton accepts the empty language.

Theorem (DCFS 2014)

It is decidable whether the language accepted by a given deterministic set automaton is regular.

Decidability Questions

Theorem

It is decidable whether a given deterministic set automaton accepts the empty language.

Theorem (DCFS 2014)

It is decidable whether the language accepted by a given deterministic set automaton is regular.

Corollary

It is decidable whether the language accepted by a given deterministic set automaton is empty, finite, infinite, or equal to a given regular set.