



INF 332: LANGUAGES & AUTOMATA

Chapter 8: Non-deterministic Finite-state Automata with ϵ -transitions

Yliès Falcone

yliès.falcone@univ-grenoble-alpes.fr — www.ylies.fr

Univ. Grenoble-Alpes
Laboratoire d'Informatique de Grenoble - www.liglab.fr

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

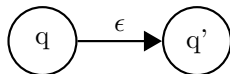
- 1 Motivations
 - Using ϵ -transitions in Practice with Automata
 - Kleene Closure of a Language
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
 - Using ϵ -transitions in Practice with Automata
 - Kleene Closure of a Language
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Use of ϵ -transitions

We allow ϵ as a transition label; such transitions are called ϵ -transitions.



A word is accepted if the input can be read until a final state is reached:
 $\hookrightarrow \epsilon$ -transitions do not “consume” any input symbols.

Underlying property:

$$\forall w \in \Sigma^* : w \cdot \epsilon = \epsilon \cdot w = w.$$

(ϵ is the neutral element of word concatenation.)

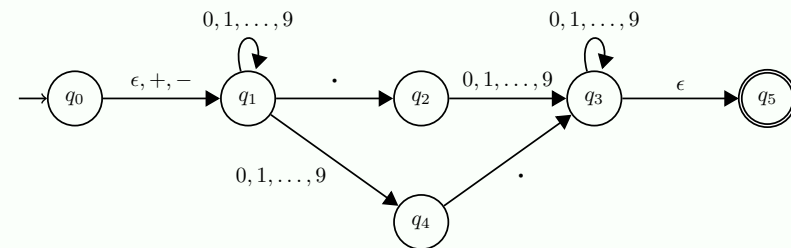
Use of ϵ -transitions

Decimal numbers

Example (Decimal numbers)

A number written in decimal notation consists of:

- an optional sign + or −,
 - a sequence of digits 0, 1, 2, ..., 9,
 - a decimal point,
 - a sequence of digits 0, 1, 2, ..., 9.
- One of the two digit sequences may be empty, but not both.

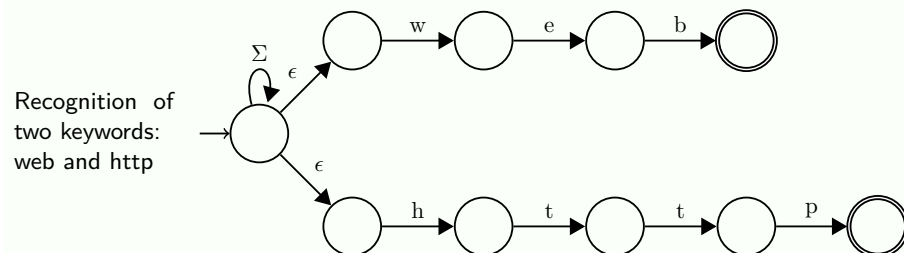


Using ϵ -transitions simplifies the construction of such an automaton: they let us “branch” between optional components (like the sign or the choice of where the digits appear) without consuming input symbols. Without ϵ -transitions, we would need to explicitly duplicate subautomata for each case, which quickly becomes cumbersome.

Use of ϵ -transitions

Keyword recognition and automata transformation

Example (Keyword recognition – compositionality)



Kleene closure of a language

Definition – language view

Let L be a language (not necessarily finite-state) over Σ .

Definition (Kleene closure)

The **Kleene closure** of L , denoted L^* , is the smallest set defined inductively by:

- $\epsilon \in L^*$, and
- if $u \in L$ and $v \in L^*$, then $u \cdot v \in L^*$.
- (equivalently: if $u \in L^*$ and $v \in L$, then $u \cdot v \in L^*$.)

Remark In other words, the Kleene closure of L is the set of all words obtained by a finite concatenation of words from L :

$$L^* = \{\epsilon\} \cup \{a_0 \cdots a_n \mid n \in \mathbb{N}, \forall i \leq n : a_i \in L\}.$$

□

Example (Kleene closure)

- $L_1 = \{ba, cd\}$
- $L_1^* = \{\epsilon, ba, cd, baba, bacd, cdcd, cdba, \dots\}$
- $L_2 = \{aa, b\}$
- L_2^* is the set of all words with an even number of a 's and an unconstrained number of b 's.

For a language consisting of words of length 1 (i.e., a subset of the alphabet), its Kleene closure is the full set of words over that alphabet.

Kleene closure of a language

Automaton view

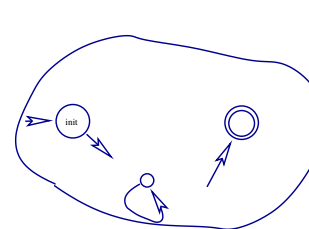
We want to show that if L is a finite-state language, then so is L^* :

$$\forall L \subseteq \Sigma^* : L \in EF \implies L^* \in EF.$$

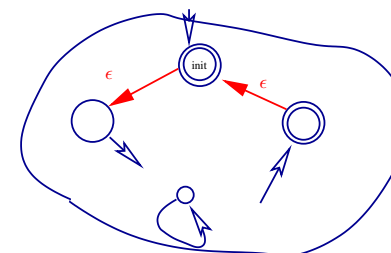
Given an automaton that recognizes L .

Can we construct an automaton that recognizes L^* ?

Yes: starting from any automaton for L , we can easily obtain one for L^* using ϵ -transitions:



Automaton for L



Automaton for L^*

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

1 Motivations

2 Non-deterministic Finite-state Automata with ϵ -transitions

- Definition
- Recognized Language

3 Eliminating ϵ -transitions

4 (Back to the) Closure Properties of Finite-State Languages

5 Summary

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

1 Motivations

2 Non-deterministic Finite-state Automata with ϵ -transitions

- Definition
- Recognized Language

3 Eliminating ϵ -transitions

4 (Back to the) Closure Properties of Finite-State Languages

5 Summary

NFA with ϵ -transitions

Let Σ be an alphabet where the symbol $\epsilon \notin \Sigma$.

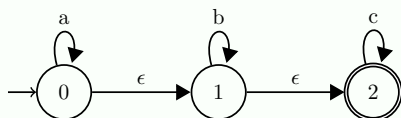
Definition (NFA with ϵ -transitions)

An *automate non-déterministe avec ϵ -transitions* (ϵ -NFA) is given by a quintuple $(Q, \Sigma, q_0, \Delta, F)$ where:

- Q is a finite set of *states*,
- Σ is the alphabet of the automaton,
- $q_0 \in Q$ is the *initial state*,
- $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the *transition relation*,
- $F \subseteq Q$ is the set of *accepting states*.

Example (NFA with ϵ -transitions)

Let $\Sigma = \{a, b, c\}$.



Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
 - Definition
 - Recognized Language
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Configuration

Let $A = (Q, \Sigma, q_0, \Delta, F)$ be an ϵ -NFA.

Definition (Configuration)

A *configuration* of the automaton A is a pair (q, u) where $q \in Q$ and $u \in \Sigma^*$.

Definition (Derivation relation (between configurations))

We define the *derivation relation* $\rightarrow_\Delta$ between configurations:

$$(q, a \cdot u) \rightarrow_\Delta (q', u') \text{ iff } ((q, a, q') \in \Delta \wedge u' = u) \text{ or } (a \cdot u = u' \wedge (q, \epsilon, q') \in \Delta)$$

Notation

- We write $q \xrightarrow{a_1 \dots a_n^*}_\Delta q'$ if there exist $q_1, \dots, q_{n-1}$ such that:

$$(q, a_1, q_1) \in \Delta, (q_1, a_2, q_2) \in \Delta, \dots, (q_{n-1}, a_n, q') \in \Delta.$$

- We write $q \xrightarrow{*}_\Delta q'$ if there exist $a_1, \dots, a_n$ such that $q \xrightarrow{a_1 \dots a_n^*}_\Delta q'$.

Execution

Definition (Execution)

An *execution of the automaton A* is a sequence of configurations $(q_0, u_0) \dots (q_n, u_n)$ such that

$$(q_i, u_i) \rightarrow_\Delta (q_{i+1}, u_{i+1}), \text{ for } i = 0, \dots, n-1.$$

The notions of

- acceptance of a word, and
- recognized language

are defined as in the case of NFAs, *mutatis mutandis*.

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
 - Translation to NFA
 - Direct Translation to DFA
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

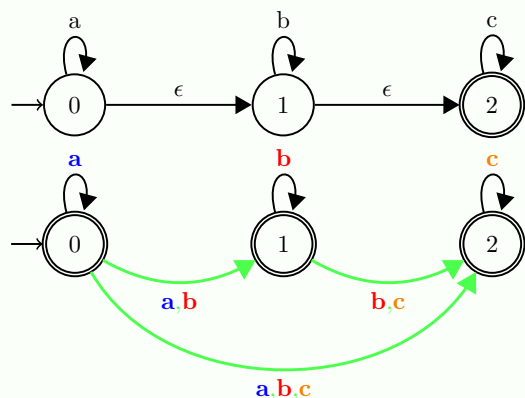
- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
 - Translation to NFA
 - Direct Translation to DFA
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Elimination of ϵ -transitions

The idea on an example

Example (NFA with ϵ -transitions)

Let $\Sigma = \{a, b, c\}$.



Step by step, ϵ -transitions are replaced until we obtain an equivalent NFA without them.

Elimination of ϵ -transitions: translation into NFA

Definition

Let $A = (Q, \Sigma, q_0, \Delta, F)$ be an ϵ -NFA.

Definition (Elimination of ϵ -transitions)

We construct an NFA

$$\epsilon\ell(A) = (Q, \Sigma, q_0, \epsilon\ell(\Delta), \epsilon\ell(F))$$

that recognizes $L(A)$, where:

- The transition relation $\epsilon\ell(\Delta)$ is defined as follows: $(q, a, q') \in \epsilon\ell(\Delta)$ iff there exist $q_1, q_2 \in Q$ such that:
 - 1 $q \xrightarrow{\epsilon^*} q_1$
 - 2 $(q_1, a, q_2) \in \Delta$
 - 3 $q_2 \xrightarrow{\epsilon^*} q'$
- The set of accepting states $\epsilon\ell(F)$ is defined by:

$$\epsilon\ell(F) = \{q \in Q \mid \exists q' \in F : q \xrightarrow{\epsilon^*} q'\}$$

Intuitively: we propagate transitions through ϵ -paths and mark states as accepting if they can reach an accepting state via ϵ -transitions

Correctness of the ϵ -elimination procedure

Let $A = (Q, \Sigma, q_0, \Delta, F)$ be an ϵ -NFA.

Theorem: Correctness of ϵ -elimination

$$L(A) = L(\epsilon\ell(A)).$$

Proof (by induction)

For all $u, u' \in \Sigma^*$ (and all $q, q' \in Q$),

$$(q, u) \xrightarrow{*}_{\epsilon\ell(\Delta)} (q', u') \text{ if and only if } (q, u) \xrightarrow{*}_{\Delta} (q', u').$$

- $\epsilon \in L(A)$ if and only if $\epsilon \in L(\epsilon\ell(A))$.
- Let $u \in \Sigma^*$, and assume that for all $u' \in \Sigma^*$:
 $(q, u) \xrightarrow{*}_{\Delta} (q', u') \text{ iff } (q, u) \xrightarrow{*}_{\epsilon\ell(\Delta)} (q', u')$.

Let $a \in \Sigma$. We must show that for all $u' \in \Sigma^*$:

$$(q, u \cdot a) \xrightarrow{*}_{\Delta} (q', u') \text{ iff } (q, u \cdot a) \xrightarrow{*}_{\epsilon\ell(\Delta)} (q', u').$$

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
 - Translation to NFA
 - Direct Translation to DFA
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

ϵ -closure

Definition

The ϵ -closure of a state q is the set of all states reachable from q by following transitions labeled with ϵ .

Definition (ϵ -closure of a state)

Let $q \in Q$ be a state. We define $ECLOSE(q)$ recursively:

- **Base case:** $q \in ECLOSE(q)$
- **Induction:** If $p \in ECLOSE(q)$ and there exists a transition from p to $r \in Q$ labeled with ϵ , then $r \in ECLOSE(q)$

Equivalently: $ECLOSE(q) = \delta^*(q, \epsilon)$

Definition (ϵ -closure of a set of states)

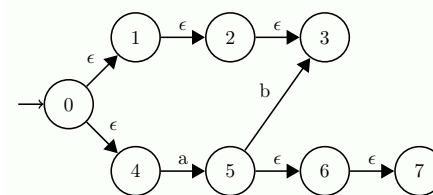
For $S \subseteq Q$:

$$ECLOSE(S) = \bigcup_{q \in S} ECLOSE(q)$$

ϵ -closure

Examples

Example (ϵ -closure)



ϵ -closure of states:

- $ECLOSE(0) = \{0, 1, 2, 3, 4\}$
- $ECLOSE(3) = \{3\}$
- $ECLOSE(5) = \{5, 6, 7\}$

ϵ -closure of sets of states:

- $ECLOSE(\{0, 3\}) = \{0, 1, 2, 3, 4\}$
- $ECLOSE(\{3, 5\}) = \{3, 5, 6, 7\}$

The ϵ -closure groups together all states reachable via ϵ -transitions.

Extended Transition Relation

Definition

We define the extended transition relation $\hat{\delta}$, which allows reading both alphabet symbols and ϵ ; ϵ is treated as a symbol that does not consume input.

Intuitively, $\hat{\delta}(q, w)$ is the set of states reachable by following a path whose edge labels concatenate to w (where ϵ contributes nothing to w).

Definition (Extended transition relation)

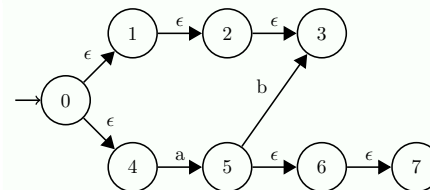
Given $q \in Q$ and $w \in (\Sigma \cup \{\epsilon\})^*$:

- **Base case:** $\hat{\delta}(q, \epsilon) = ECLOSE(q)$
- **Induction:** For $w = x \cdot a$ with $a \in \Sigma$, $\hat{\delta}(q, x \cdot a)$ is defined as follows:
 - let $\{p_1, p_2, \dots, p_k\} = \hat{\delta}(q, x)$,
 - let $\{r_1, r_2, \dots, r_m\} = \bigcup_{i=1}^k \delta(p_i, a)$,
 - then $\hat{\delta}(q, w) = ECLOSE(\{r_1, r_2, \dots, r_m\})$.

Extended Transition Relation

Example

Example (Extended transition relation)



- $\hat{\delta}(0, \epsilon) = ECLOSE(0) = \{0, 1, 2, 3, 4\}$
- $\hat{\delta}(0, a) = \{5, 6, 7\}$
- $\hat{\delta}(0, b) = \emptyset$
- $\hat{\delta}(0, ab) = \{3\}$

Eliminating ϵ -Transitions and On-the-Fly Determinization

Definition of the Determinized Automaton

Let $A = (Q, \Sigma, q_0, \Delta, F)$ be an ϵ -NFA.

Definition (On-the-fly determinization and ϵ -elimination)

The *determinized* automaton of A is the DFA

$$(Q_D, \Sigma, q_D, \delta, F_D)$$

defined as follows:

- $Q_D = \mathcal{P}(Q)$
- $q_D = ECLOSE(q_0)$
- **Transition function:** for any $S \in Q_D$, $a \in \Sigma$:
 - let $\{p_1, p_2, \dots, p_k\} = S$,
 - let $\{r_1, r_2, \dots, r_m\} = \bigcup_{i=1}^k \Delta(p_i, a)$,
 - then $\delta(S, a) = ECLOSE(\{r_1, r_2, \dots, r_m\})$,
- $F_D = \{S \in \mathcal{P}(Q) \mid S \cap F \neq \emptyset\}$.

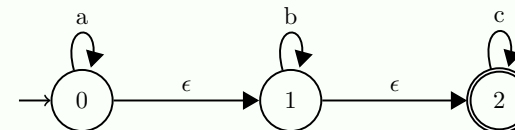
Remark Every state of the determinized automaton (reachable via δ) corresponds to an ϵ -closed set of states in the original ϵ -NFA. ☐

Remark In practice, we construct only the reachable ϵ -closed subsets instead of the entire powerset $\mathcal{P}(Q)$. ☐

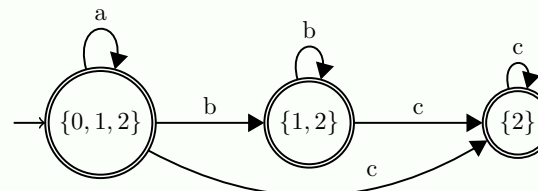
Eliminating ϵ -Transitions and On-the-Fly Determinization

Translation to DFA: Example

Example (Eliminating ϵ -transitions – Translation to DFA)



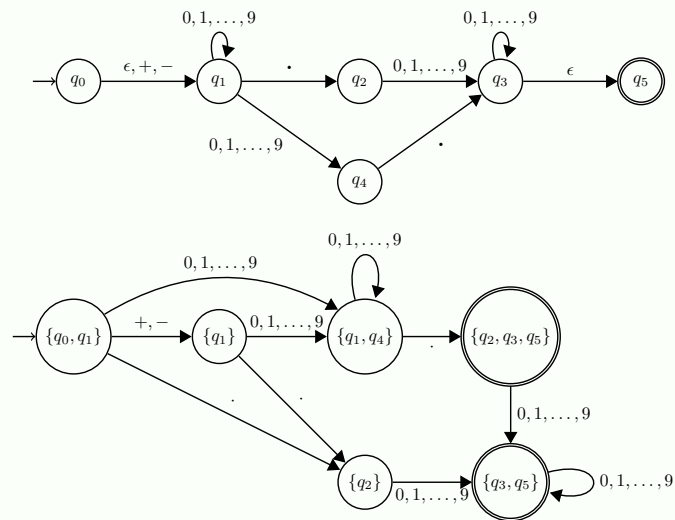
Here, elimination of ϵ -transitions is combined with determinization. Both operations are performed *on the fly*.



Eliminating ϵ -Transitions and On-the-Fly Determinization

Translation to DFA: Another Example

Example (Eliminating ϵ -transitions – Translation to DFA)



Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

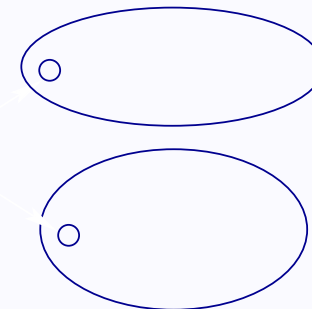
- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
 - Closure under union and concatenation
 - Closure under Reversal
 - Closure under morphism
- 5 Summary

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

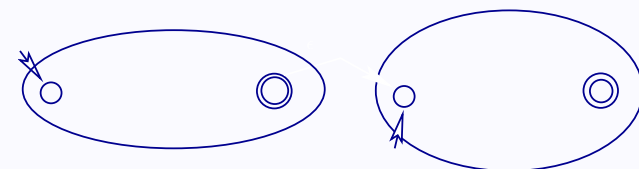
- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
 - Closure under union and concatenation
 - Closure under Reversal
 - Closure under morphism
- 5 Summary

Closure of Finite-State Languages under Union and Concatenation

Union

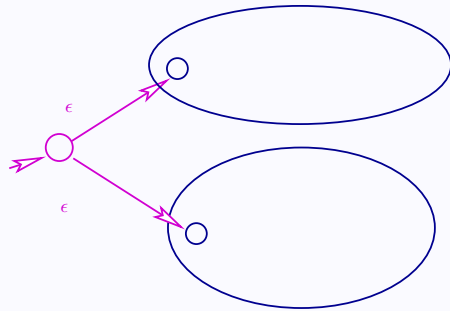


Concatenation

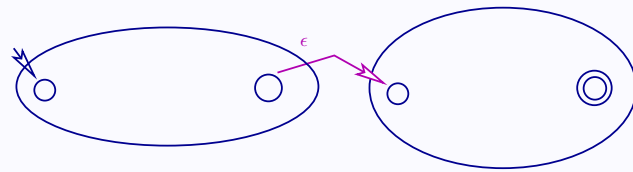


Closure of Finite-State Languages under Union and Concatenation

Union



Concatenation



Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
 - Closure under union and concatenation
 - Closure under Reversal
 - Closure under morphism
- 5 Summary

Reversal Operation

The reversal of a word is the word written *from right to left*.

Definition (Reversal operation – word and language)

- For $w = a_1 \cdot a_2 \cdots a_n$, the *reversal* of w is the word denoted w^R and defined by:

$$w^R = a_n \cdot a_{n-1} \cdots a_1$$

- For $L \subseteq \Sigma^*$, the *reversal* of L is the language, denoted L^R , consisting of the reversals of words in L :

$$L^R = \{w^R \mid w \in L\}$$

Example (Reversal)

For $L = \{001, 10, 111\}$, we have $L^R = \{100, 01, 111\}$.

Closure of Finite-State Languages under Reversal

Closure of EF under the reversal operation

- If $L \subseteq \Sigma^*$ is a finite-state language, then so is L^R .
- Hence EF is closed under the reversal operation.

Informal proof based on automata

Given a finite-state language L and its recognizing automaton A :

- 1 Reverse all the transitions of A .
- 2 Make the initial state of A the unique accepting state.
- 3 Create a new initial state q_0 (if the original initial state was accepting, make q_0 accepting as well).
- 4 Add an ϵ -transition from q_0 to each accepting state of the original automaton.

(The full proof is left as an exercise in tutorials.)

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

1 Motivations

2 Non-deterministic Finite-state Automata with ϵ -transitions

3 Eliminating ϵ -transitions

4 (Back to the) Closure Properties of Finite-State Languages

- Closure under union and concatenation
- Closure under Reversal
- Closure under morphism

5 Summary

Reminder: Group Morphisms

Definition (Group)

A group is a pair $(G, *)$ where G is a set and $*$ is a binary operation on G , such that for all $g_1, g_2, g_3 \in G$:

- $g_1 * g_2 \in G$,
- $g_1 * (g_2 * g_3) = (g_1 * g_2) * g_3$
- there exists a neutral element e_G such that $g_1 * e_G = e_G * g_1 = g_1$
- every element has an inverse

Let $(G, \bullet)$ and $(G', *)$ be two groups whose neutral elements are e_G and $e_{G'}$, respectively.

Definition (Morphism)

A function $f : G \rightarrow G'$ is a group morphism if

$$\forall x, y \in G : f(x \bullet y) = f(x) * f(y)$$

Example (Morphism)

The function $f : (\mathbb{Z}, +) \rightarrow (\mathbb{R}^+, \times)$ defined by $f(n) = 2^n$ is a group morphism.

In what follows, for each alphabet Σ , we consider the group $(\Sigma^*, \cdot)$ where $\cdot$ is the concatenation of words over Σ^* , and morphisms are used to translate words over one alphabet into words over another.

Morphisms on Words

Let Σ and Σ' be two alphabets.

Definition (Word Morphism)

A function $h : \Sigma \rightarrow \Sigma'^*$ induces a morphism $\hat{h} : \Sigma^* \rightarrow \Sigma'^*$ defined by:

- $\hat{h}(\epsilon) = \epsilon$, and
- $\hat{h}(u \cdot a) = \hat{h}(u) \cdot h(a)$.

Remark We can prove that $\hat{h}$ is a morphism by showing that

$$\forall x, y \in \Sigma^* : \hat{h}(x \cdot y) = \hat{h}(x) \cdot \hat{h}(y),$$

using induction on y (or equivalently on $|y|$). $\square$

Example (Word Morphism)

Consider $\Sigma = \{a, b\}$, $\Sigma' = \{0, 1\}$ and the function $h : \Sigma \rightarrow \Sigma'^*$ defined by $h(a) = 0$ and $h(b) = 1 \cdot 1$.

This function h induces a morphism $\hat{h} : \Sigma^* \rightarrow \Sigma'^*$.

For instance,

$$\hat{h}(b \cdot a \cdot a) = 1 \cdot 1 \cdot 0 \cdot 0 = \hat{h}(b) \cdot \hat{h}(a \cdot a).$$

From now on, we write h instead of $\hat{h}$.

Closure of Finite-State Languages under Morphisms

Theorem: Closure of EF under Morphisms

If $L \subseteq \Sigma^*$ is a finite-state language, then so is its image under a morphism h , denoted $h(L)$ and defined by

$$h(L) = \{h(u) \mid u \in L\}.$$

Thus EF is closed under morphisms.

Proof

Based on automata. Left as an exercise.

Example (Closure of EF under Morphisms)

Consider $\Sigma = \{a, b\}$, $\Sigma' = \{0, 1\}$ and the morphism $\hat{h}$ (denoted simply h below) induced by the function $h : \Sigma \rightarrow \Sigma'^*$ defined by $h(a) = 0$ and $h(b) = 1 \cdot 1$.

- The language $L_1 \subseteq \Sigma^*$ consisting of all words with an odd number of a 's is a finite-state language.
- The language $h(L_1) \subseteq \Sigma'^*$ consisting of all words with an odd number of 0's and an even number of 1's is also a finite-state language.

Intuition

Applying a morphism corresponds to *relabeling or expanding transitions* in the automaton of L . Each letter $a \in \Sigma$ is replaced by the automaton for $h(a)$, so the structure remains finite-state. This guarantees that $h(L)$ is recognized by a finite automaton.

Closure of EF under Inverse Morphism

Let h be a morphism and h^{-1} its inverse mapping (preimage under h).

Theorem: Closure of EF under Inverse Morphism

If $L \subseteq \Sigma'^*$ is a finite-state language, then so is its preimage under h , defined by

$$h^{-1}(L) = \{u \in \Sigma^* \mid h(u) \in L\}.$$

Thus EF is closed under inverse morphisms.

Proof

Based on automata. Left as an exercise.

Example (Closure of EF under Inverse Morphism)

Let $\Sigma = \{a, b, c, d\}$, $\Sigma' = \{0, 1, 2\}$, and consider the morphism $h : \Sigma \rightarrow \Sigma'^*$ defined by

$$h(a) = 0, \quad h(b) = 1, \quad h(c) = \epsilon, \quad h(d) = 2.$$

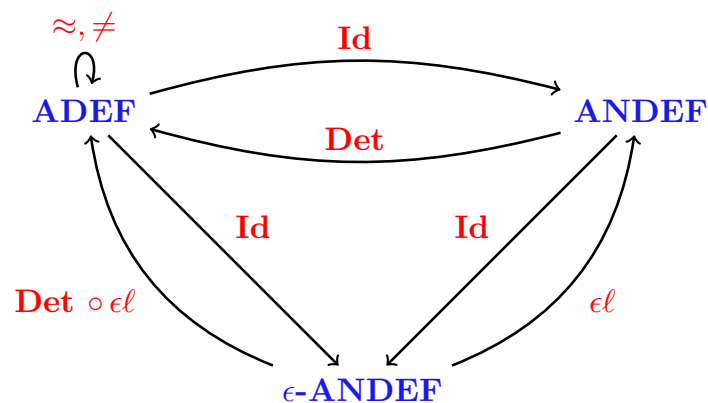
- The language $L_1 \subseteq \Sigma'^*$ consisting of all words with an even number of 0's and no 2's is finite-state.
- Its preimage $h^{-1}(L_1) \subseteq \Sigma^*$ consists of all words with:
 - an even number of a 's,
 - no d 's,
 - arbitrary occurrences of b and c .

This is also finite-state.

Outline Chap. 8 - Non-deterministic Finite-state Automata w/ ϵ -transitions

- 1 Motivations
- 2 Non-deterministic Finite-state Automata with ϵ -transitions
- 3 Eliminating ϵ -transitions
- 4 (Back to the) Closure Properties of Finite-State Languages
- 5 Summary

Summary 1: Transformations Between Automata



Summary 2: Closure of the Class of Finite-State Languages, Decision Problems and Procedures

Closure Properties

Finite-state languages are closed under the following operations:

- | | |
|------------------------|-------------------------------|
| 1 union, intersection, | 4 Kleene star (closure), |
| 2 complement, | 5 reversal, |
| 3 concatenation, | 6 morphism, inverse morphism. |

Each closure property was shown via a corresponding automaton construction.

Decision Problems and Procedures

The following decision problems are decidable:

- | | | |
|---|-----------------|-------------------------|
| 1 state reachability, | 3 emptiness, | 5 language inclusion, |
| 2 co-reachability ($\exists$ a path to a final state), | 4 infiniteness, | 6 language equivalence. |

For each problem, we provided a concrete decision procedure.