Efficient representation of finite sets

Ivo Rapant

January 27, 2018

Roadmap

- Motivation
- Basics
- New Data-Structure
- Operation on fixed-length languages
- Decision Diagrams
- Conclusion

Motivation

Motivation

Constraints:

- finite sets
- fixed-length languages
  - words of the same length

Motivation - Example

$\Sigma = \{a, b\}, n = 5$

Motivation - Example

$\Sigma = \{a, b\}, n = 5$

Language

$\rightarrow L = \{aaaaa, aaaab, aaaba, aaabb, aabaa, aabab, aabba, aabbb,$
$abaaa, abaab, ababa, ababb, abbaa, abbab, abbba, abbbb,$
$baaaa, baaab, baaba, baabb, babaa, babab, babba, babbb,$
$bbaaa, bbaab, bbaba, bbabb, bbbaa, bbbab, bbbba, bbbbb\}$

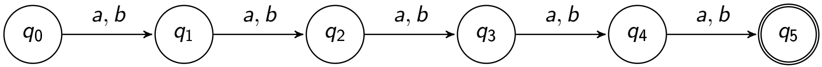
Motivation - Example

$\Sigma = \{a, b\}, n = 5$

Language

$\rightarrow L = \{aaaaa, aaaab, aaaba, aaabb, aabaa, aabab, aabba, aabbb,$
$abaaa, abaab, ababa, ababb, abbaa, abbab, abbba, abbbb,$
$baaaa, baaab, baaba, baabb, babaa, babab, babba, babbb,$
$bbaaa, bbaab, bbaba, bbabb, bbbaa, bbbab, bbbba, bbbbb\}$

DFA

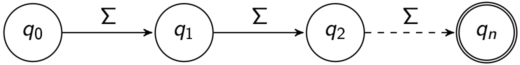


Motivation - General

Observation:

For a Language $L = \Sigma^n$ for a given $n \in \mathbb{N}$
the size of $|L| = |\Sigma|^n$.

When we define the size of automaton as $|A| = |\delta|$, the size of the transition relation δ, and say A_L is the automaton for L, we see that $|A_L| = n$.



Basics

Language L

A language $L \subseteq \Sigma^*$ has length $n \geq 0$ if every word of L has length n. If L has length n for some $n \geq 0$ then we say that L is a fixed-length language, or that it has fixed-length.

Language L

A language $L \subseteq \Sigma^*$ has length $n \geq 0$ if every word of L has length n. If L has length n for some $n \geq 0$ then we say that L is a fixed-length language, or that it has fixed-length.

Given a language $L \subseteq \Sigma^*$ and $a \in \Sigma$, the language L^a is defined by $L^a = \{w \in \Sigma^* \mid aw \in L\}$. So if $|L| = n$, then $|L^a| = n - 1$.

Master Automaton

The master automaton over the alphabet Σ is the tuple $M = (Q_M, \Sigma, \delta_M, F_M)$, where

- Q_M is the set of all fixed-length languages over Σ;
- $\delta : Q_M \times \Sigma \to Q_M$ is given by $\delta(L, a) = L^a$ for every $q \in Q_M$ and $a \in \Sigma$;
- F_M is the singleton set containing the language $\{\epsilon\}$ as only element.

Master Automaton

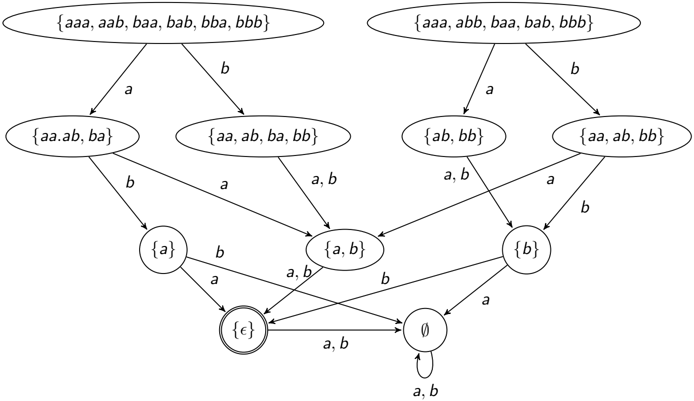


Figure: A fragment of the master automaton for the alphabet $\{a, b\}$

Master Automaton

Proposition 1
Let L be a fixed-length language. The language recognized from the state L of the master automaton is L.

Proposition 2
For every fixed-length language L, the automaton A_L is the minimal DFA recognizing L.

Data Structure for Fixed-length Languages

Data Structure for Fixed-length Languages

Definition

Let $\mathcal{L} = \{L_1, \ldots, L_N\}$ *be a set of languages of the same length over the same alphabet* Σ*. The multi-DFA* $A_\mathcal{L}$ *is the tuple* $A_\mathcal{L} = (Q_\mathcal{L}, \Sigma_\mathcal{L}, \delta_\mathcal{L}, Q_{0\mathcal{L}}, F_\mathcal{L})$*, where* $Q_\mathcal{L}$ *is the set of states of the master automaton reachable from at least one of the states* $L_1, \ldots, L_n$; $Q_{0\mathcal{L}} = \{L_1, \ldots, L_n\}$; $\delta_\mathcal{L}$ *is the projection of* δ_M *onto* Q_L*; and* $F_\mathcal{L} = F_M$.

Example multi-DFA

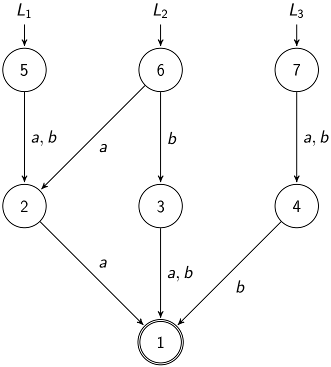


Figure: The multi-DFA for $L_1 = \{aa, ba\}, L_2 = \{aa, ba, bb\}$, and $L_3 = \{ab, bb\}$.

Example multi-DFA

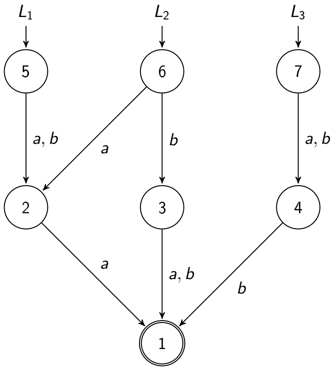


Figure: The multi-DFA for $L_1 = \{aa, ba\}$, $L_2 = \{aa, ba, bb\}$, and $L_3 = \{ab, bb\}$.

Representation of multi-DFAs as

- table of *nodes*
- node: *pair* $\langle q, s \rangle$
  q: *state identifier*
  s: *successor tuple* of the node
- *special case*
  $q = 0$: state accepting empty language

Example multi-DFA

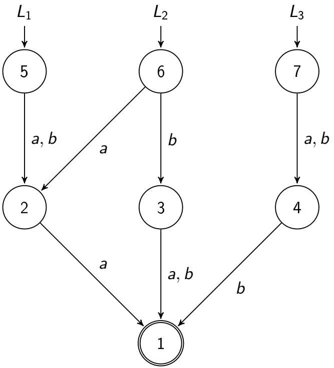


Figure: The multi-DFA for $L_1 = \{aa, ba\}$, $L_2 = \{aa, ba, bb\}$, and $L_3 = \{ab, bb\}$.

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4

Table: The table for the multi-DFA

Procedure *make*

make(s)

- returns the state of table T having s as successor tuple
- if such state doesn't exist, it adds a new node $\langle q, s \rangle$ to T, with a fresh identifier q

multi-DFA

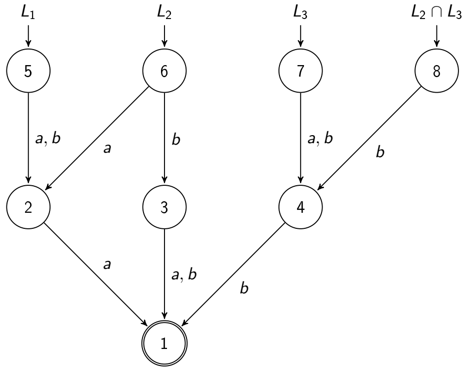


Figure: The multi-DFA for with $L_1 = \{aa, ba\}$, $L_2 = \{aa, ba, bb\}$, and $L_3 = \{ab, bb\}$.

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4

Table: The table for the multi-DFA

Operations on fixed-length languages

Operations of fixed-length languages

Input: multi-DFAs represented as a table of nodes

Operations:

- Intersection
- Union
- Complement
- Emptiness
- Universal
- Inclusion

Operations on fixed-length languages

inter(q_1, q_2)

Input: states q_1, q_2 of the same length

Output: state recognizing $L(q_1) \cap L(q_2)$

if $G(q_1, q_2)$ is not empty **then return** $G(q_1, q_2)$
if $q_1 = q_\emptyset$ **or** $q_2 = q_\emptyset$ **then return** $q_\emptyset$
else if $q_1 = q_\varepsilon$ **and** $q_2 = q_\varepsilon$ **then return** q_ϵ
else $/*\ q_1, q_2 \notin \{q_\emptyset, q_\varepsilon\}\ */$
    for all $i = 1, \ldots, m$ **do** $r_i \leftarrow inter(q_1^{a_i}, q_2^{a_i})$
    $G(q_1, q_2) \leftarrow \texttt{make}(r_1, \ldots, r_m)$
    return $G(q_1, q_2)$

An execution of *inter*

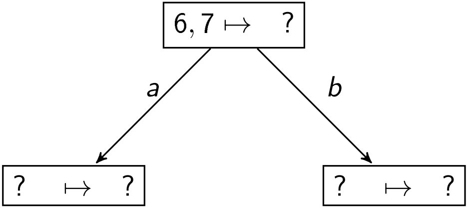
Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4

$inter(6, 7)$

$\boxed{6, 7 \mapsto \quad ?}$

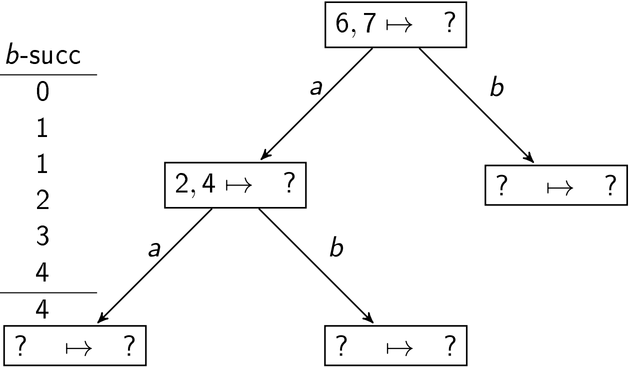
An execution of *inter*

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



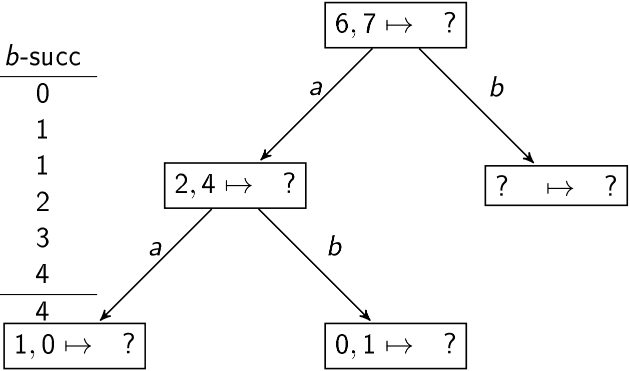
An execution of *inter*

Ident.	*a*-succ	*b*-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



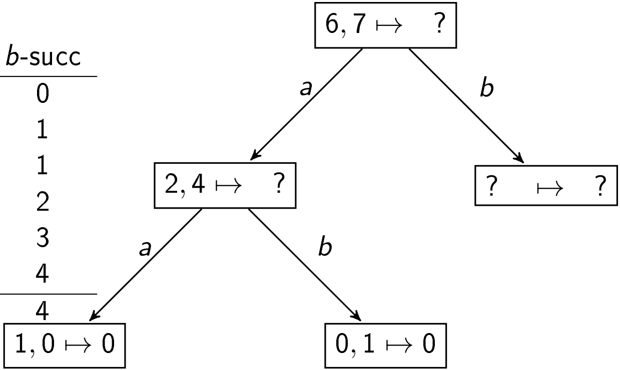
An execution of *inter*

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



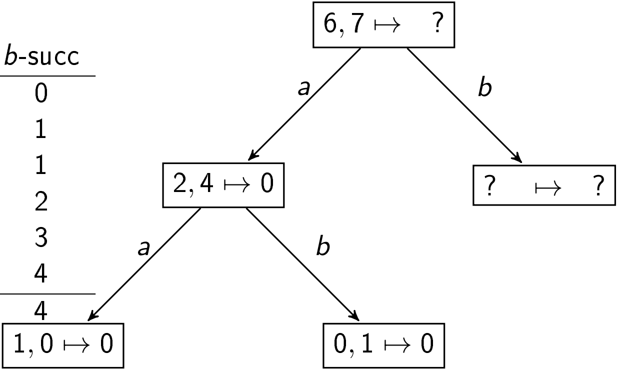
An execution of *inter*

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



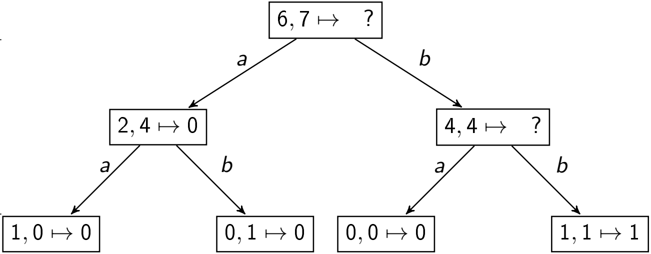
An execution of *inter*

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



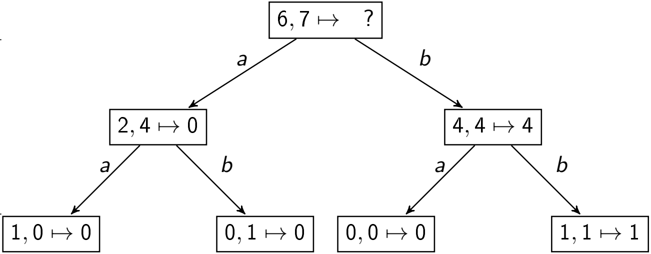
An execution of *inter*

Ident.	*a*-succ	*b*-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



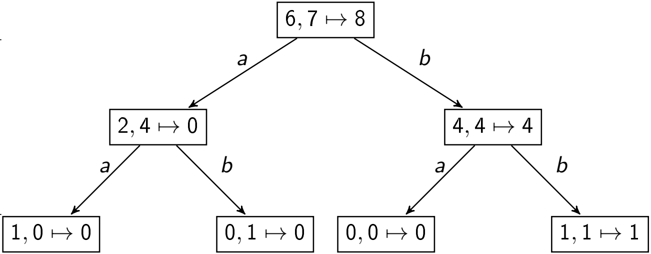
An execution of *inter*

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



An execution of *inter*

Ident.	a-succ	b-succ
2	1	0
3	1	1
4	0	1
5	2	2
6	2	3
7	4	4
8	0	4



Decision Diagrams

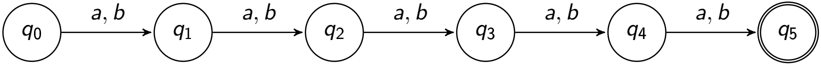
Motivation

$\Sigma = \{a, b\}, n = 5$

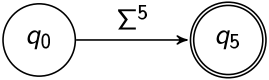
Language

$\rightarrow L = \{aaaaa, aaaab, aaaba, aaabb, aabaa, aabab, aabba, aabbb,$
$abaaa, abaab, ababa, ababb, abbaa, abbab, abbba, abbbb,$
$baaaa, baaab, baaba, baabb, babaa, babab, babba, babbb,$
$bbaaa, bbaab, bbaba, bbabb, bbbaa, bbbab, bbbba, bbbbb\}$

DFA



DD



Decision Diagrams

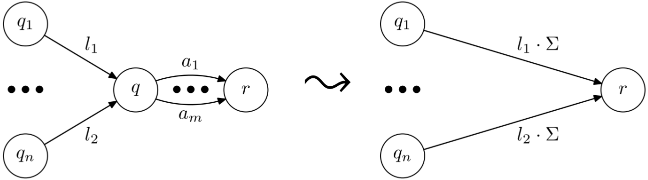
A *decision diagram* (DD) is an automaton $A = (Q, \Sigma, \delta, Q_0, F)$ whose transitions are labelled by regular expressions of the form

$$a\Sigma^n = a\underbrace{\Sigma\Sigma\ldots\Sigma\Sigma}_{n}$$

and satisfies the following determinacy condition:
for every $q \in Q$ and $a \in \Sigma$ there is exactly one $k \in \mathbb{N}$ such that $\delta(q, a\Sigma^k) \neq \emptyset$, and for this k there is a state q' such that $\delta(q, a\Sigma^k) = \{q'\}$.

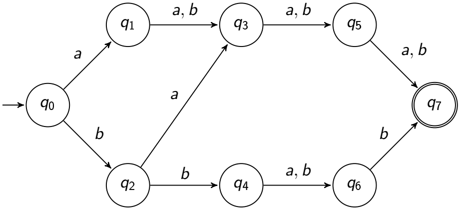
Decision Diagrams

Reduction Rule



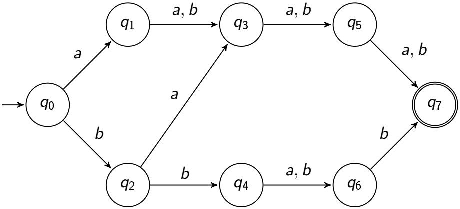
Decision Diagrams

DFA for a language of length four:

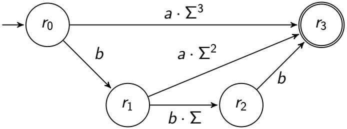


Decision Diagrams

DFA for a language of length four:



DD for the same language:



Decision Diagrams and Kernels

A fixed-length language L over an alphabet Σ is a kernel *if* $L = \emptyset, L = \{\epsilon\}$*, or there are* $a, b \in \Sigma$ *such that* $L^a \neq L^b$*. The* kernel *of a fixed-length language L, denoted by $\langle L \rangle$, is the unique kernel satisfying* $L = \Sigma^k \langle L \rangle$ *for some* $k \geq 0$*.*

Decision Diagrams and Kernels

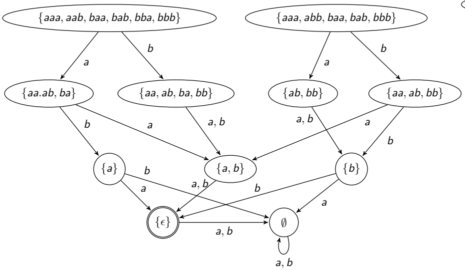


Figure: Fragment of the master automaton

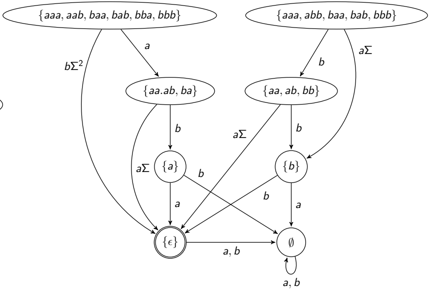


Figure: Fragment of the master decision diagram

DD

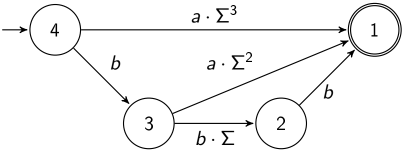


Figure: multi-DD

DD

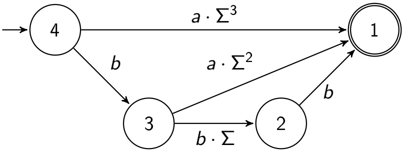


Figure: multi-DD

Representation of multi-DDs as

- table of *kernodes*
- kernode: triple $\langle q, l, s \rangle$

DD

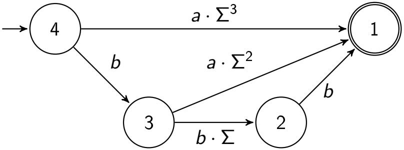


Figure: multi-DD

Ident.	Length	a-succ	b-succ
4	4	1	3
3	3	1	2
2	1	0	1

Table: The table for the multi-DD

Procedure *kmake*

kmake(l,s)

- behaves like *make*
- returns kernode q of length l with s as successor-tuple
- if such state doesn't exist it adds a new kernode $\langle q, l, s \rangle$with a fresh identifier q

Operations on Kernels

kinter(q_1, q_2)

Input: states q_1, q_2 recognizing $\langle L_1 \rangle, \langle L_2 \rangle$

Output: state recognizing $\langle L_1 \cap L_2 \rangle$

if $G(q_1, q_2)$ is not empty **then return** $G(q_1, q_2)$

if $q_1 = q_\emptyset$ **or** $q_2 = q_\emptyset$ **then return** $q_\emptyset$

if $q_1 \neq q_\emptyset$ **and** $q_2 \neq q_\emptyset$ **then**

  if $l_1 < l_2$ /* lengths of the kernodes for q_1, q_2 */ **then**

    for all $i = 1, \ldots, m$ **do** $r_i \leftarrow kinter(q_1, q_2^{a_i})$

    $G(q_1, q_2) \leftarrow \texttt{kmake}(l_2, r_1, \ldots, r_m)$

  else if $l_1 \quad l_2$ **then**

    for all $i = 1, \ldots, m$ **do** $r_i \leftarrow kinter(q_1^{a_i}, q_2)$

    $G(q_1, q_2) \leftarrow \texttt{kmake}(l_1, r_1, \ldots, r_m)$

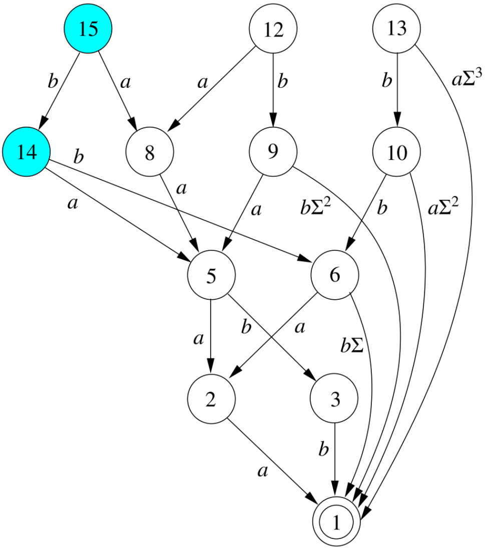
  else /* $l_1 = l_2$ */

    for all $i = 1, \ldots, m$ **do** $r_i \leftarrow kinter(q_1^{a_i}, q_2^{a_i})$

    $G(q_1, q_2) \leftarrow \texttt{kmake}(l_1, r_1, \ldots, r_m)$

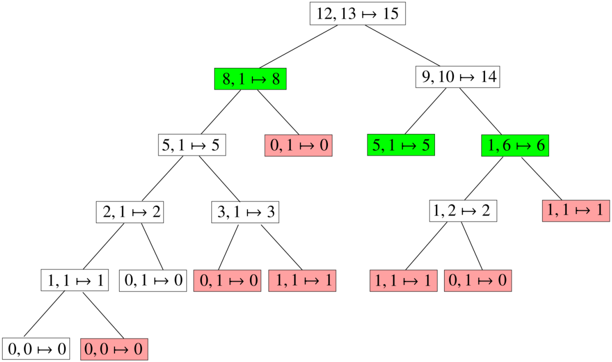
return $G(q_1, q_2)$

An execution of *kinter*



Ident.	Length	*a*-succ	*b*-succ
2	1	1	0
3	1	0	1
5	2	2	3
6	2	2	1
8	3	5	0
9	3	5	1
10	3	1	6
12	4	8	9
13	4	1	10
14	3	5	6
15	4	8	14

An execution of *kinter*



Ident.	Length	*a*-succ	*b*-succ
2	1	1	0
3	1	0	1
5	2	2	3
6	2	2	1
8	3	5	0
9	3	5	1
10	3	1	6
12	4	8	9
13	4	1	10
14	3	5	6
15	4	8	14

Conclusion

References

- Automata Theory - An algorithmic approach: Chapter 7
  https://www7.in.tum.de/~esparza/autoskript.pdf