Finite-State Systems

Gérard Berry

http://www-sop.inria.fr/members/Gerard.Berry/

Collège de France
Informatics and Digital Sciences Chair

INRIA Sophia-Antipolis

Edinburgh Informatics Forum, September 30th, 2010

Finite-State Systems

Gérard Berry

http://www-sop.inria.fr/members/Gerard.Berry/

Collège de France
Informatics and Digital Sciences Chair

INRIA Sophia-Antipolis

Edinburgh Informatics Forum, September 30th, 2010

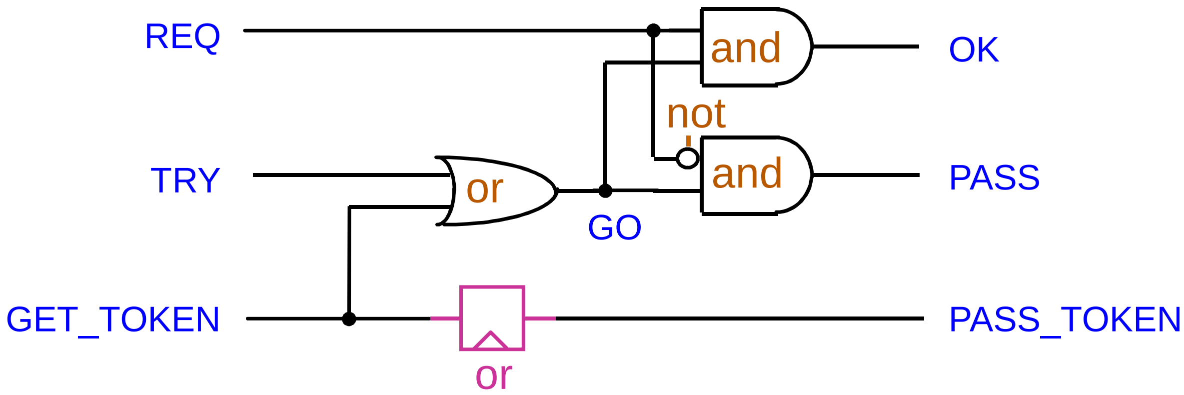
Finite-State Systems

- General principles :
  - the system only uses finite resources
  - it proceeds by a sequence of elementary transitions
- Multiple formalisms
  - machines: finite automata, Boolean circuits
  - regular expressions: shell, lex, perl, etc.
  - hierarchical state machines: Statecharts , SyncCharts
  - synchronous languages : Esterel, Lustre, Signal
  - semantics : regular languages

Application Fields

- Language analysis
  - natural languages
  - programming languages
- Electronic circuits
  - data path / control path
  - memory / cache handling
  - NoCs (Networks on Chips), USB, SATA, etc.
- Communication protocols
  - inititiation and maintenance of communication links
  - error detection and handling, packet retransmission
- Industry
  - real-time control, programmable logic controllers (PLCs)

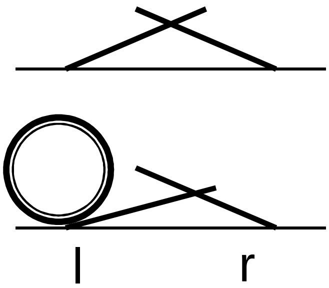
Boolean Circuits



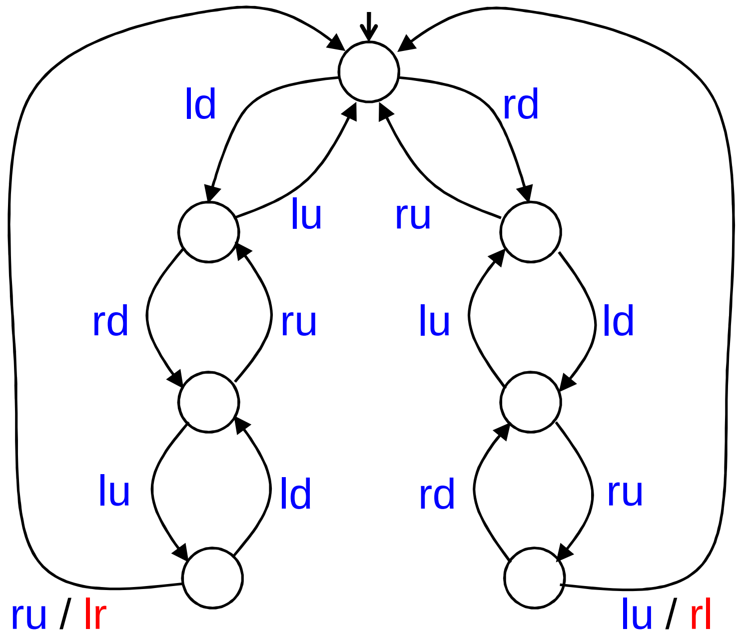
OK = REQ and GO
PASS = not REQ and GO
GO = TRY or GET_TOKEN
PASS_TOKEN = reg(GET_TOKEN)

Automata : Explicit State Machines

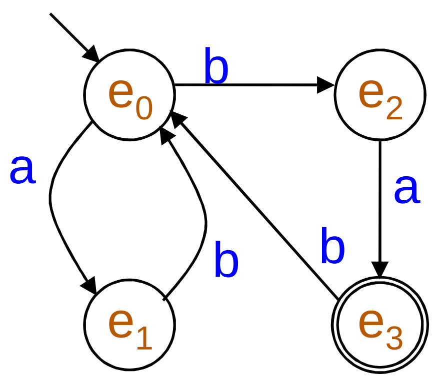
axle counter



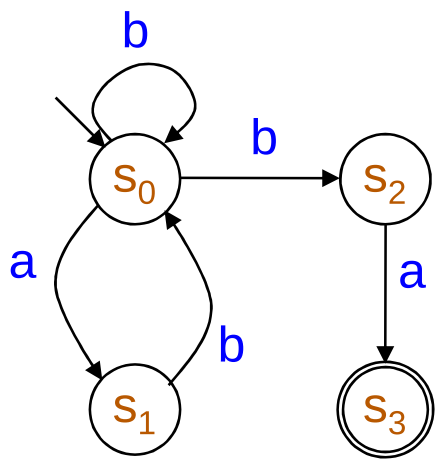
- inputs:
  ld, lu, rd, ru
- outputs:
  → lr, ← rl



Recognizer Automata



Deterministic



Non-deterministic

(w.r.t. s_0 and b)

G. Berry, L3, Edinburgh, 30/09/2010

Agenda

1. Regular Expressions
2. Derivatives and Translation to Automata
3. The Berry-Sethi Algorithm
4. Determinization and minimization
5. From Automata to Boolean Circuits
6. Automatic Sequences and Transcendental Numbers

Agenda

1. Regular Expressions
2. Derivatives and Translation to Automata
3. The Berry-Sethi Algorithm
4. Determinization and minimization
5. From Automata to Boolean Circuits
6. Automatic Sequences and Transcendental Numbers

Regular Languages

- Alphabet : finite set of letters a, b, c, x, y...
- Word: finite sequence u, v, w of letters

  ε = empty word
- Concatenation of words:

  $ab \cdot cd = abcd$
- Language L: set of words
- Regular languages: the smallest family of languages containing the finite languages and closed by concatenation and union

Regular Expressions

- 0 — $L(0) = \emptyset$
- 1 — $L(1) = \{\varepsilon\}$
- a — $L(a) = \{a\}$
- $e+e'$ — $L(e+e') = L(e) \cup L(e')$
- $e \cdot e'$ or $e\,e'$ — $L(e \cdot e') = \{uu' \mid u \in L(e),\ u' \in L(e')\}$
- e^* — $L(e^*) = \{u_0 u_1 \ldots u_n \mid n \geq 0,\ u_i \in L(e)\}$

$$e+e' = e'+e$$
$$e+(e'+e'') = (e+e')+e''$$
$$e \cdot (e'+e'') = e \cdot e' + e \cdot e''$$
$$(e+e') \cdot e'' = e \cdot e'' + e' \cdot e''$$

$$0+e = e+0 = e$$
$$0 \cdot e = e \cdot 0 = 0$$
$$1 \cdot e = e \cdot 1 = 1$$

Test Expression: (ab+b)* ba

ba

abba

bba

abbabba

bbbbababbbbba

...

Test Expression: (ab+b)* ba

ba

abba

bba

abbabba

bbbbababbbbba

...

Test for Empty Word Generation

- $\varepsilon(L) = 1$ if $\varepsilon \in L$
  $= 0$ otherwise

- $\varepsilon(0) = 0$
- $\varepsilon(1) = 1$
- $\varepsilon(a) = 0$
- $\varepsilon(e+e') = \varepsilon(e) + \varepsilon(e')$
- $\varepsilon(e \cdot e') = \varepsilon(e) \cdot \varepsilon(e')$
- $\varepsilon(e^*) = 1$

Agenda

1. Regular Expressions
2. Derivatives and Translation to Automata
3. The Berry-Sethi Algorithm
4. Determinization and minimization
5. From Automata to Boolean Circuits
6. Automatic Sequences and Transcendental Numbers

Derivatives of Regular Languages

$L \rightarrow$ ba, abba, bba, abbabba, bbbbababbbbba,...

$a^{-1}(L) \rightarrow$ ba, abba, bba, abbabba, bbbbababbbbba,...

$b^{-1}(L) \rightarrow$ ba, abba, bba, abbabba, bbbbababbbbba,...

$u^{-1}(L) = \{ v \mid uv \in L \}$ $ua^{-1}(L) = a^{-1}(u^{-1}(L))$

what remains to be written after writing u

Derivatives of Regular Expressions

$a^{-1}(e)$ = regular expression generating $a^{-1}(L(e))$

- $a^{-1}(0) = 0$
- $a^{-1}(1) = 0$
- $a^{-1}(a) = 1$
- $a^{-1}(b) = 0$ si $b \neq a$
- $a^{-1}(e + e') = a^{-1}(e) + a^{-1}(e')$
- $a^{-1}(e \cdot e') = a^{-1}(e) \cdot e' + \varepsilon(e) \cdot a^{-1}(e')$
- $a^{-1}(e^*) = a^{-1}(e) \cdot e^*$

G. Berry, L3, Edinburgh, 30/09/2010 17

Test Expression: $(ab+b)^*\ ba$

- $a^{-1} \rightarrow$ ba, abba, bba, abbabba, bbbbababbbbba,...
- $a^{-1}((ab+b)^* ba) = a^{-1}((ab+b)^*) \cdot ba$
  $+ \varepsilon((ab+b)^*) \cdot a^{-1}(ba)$
  $= a^{-1}(ab+b) \cdot (ab+b)^* \cdot ba + 0$
  $= b \cdot (ab+b)^* \cdot ba$ OK

- $b^{-1} \rightarrow$ ba, abba, bba, abbabba, bbbbababbbbba,...
- $b^{-1}((ab+b)^* ba) = b^{-1}((ab+b)^*) \cdot ba$
  $+ \varepsilon((ab+b)^*) \cdot b^{-1}(ba)$
  $= b^{-1}(ab+b) \cdot (ab+b)^* \cdot ba$
  $+ a$
  $= (ab+b)^* ba + a$ OK

G. Berry, L3, Edinburgh, 30/09/2010 18

From Derivatives to Automata

Theorem (Brzozowski) : a regular expression e has at most a finite number of distinct (simplified) derivatives $u^{-1}(e)$

Corollary : one can construct a deterministic finite automaton recognizing L(e) as follows:

- the states are the distinct derivatives of e
- the initial state is e
- there is a-labeled transition from $u^{-1}(e)$ to $ua^{-1}(e)$ if both derivatives are nonempty
- a state e' is final if ε(e') = 1

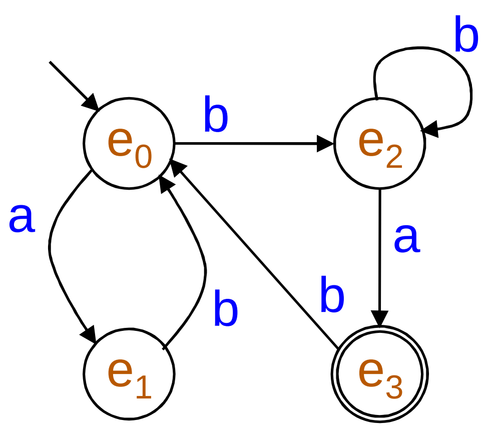
Convergent Iterative Process

- $e = e_0 = (ab+b)^*ba$
- $a^{-1}(e_0) = b(ab+b)^*ba = e_1$
- $b^{-1}(e_0) = (ab+b)^*ba + a = e_2$
- $a^{-1}(e_1) = 0$
- $b^{-1}(e_1) = (ab+b)^*ba = e_0$
- $a^{-1}(e_2) = b(ab+b)^*ba + 1 = e_3$
- $b^{-1}(e_2) = (ab+b)^*ba + a = e_2$
- $a^{-1}(e_3) = 0$
- $b^{-1}(e_3) = (ab+b)^*ba = e_0$

G. Berry, L3, Edinburgh, 30/09/2010 20

Constructing the Deterministic Automaton

- $a^{-1}(e_0) = e_1$
- $b^{-1}(e_0) = e_2$
- $a^{-1}(e_1) = 0$
- $b^{-1}(e_1) = e_0$
- $a^{-1}(e_2) = e_3$
- $b^{-1}(e_2) = e_2$
- $a^{-1}(e_3) = 0$
- $b^{-1}(e_3) = e_0$
- $\varepsilon(e_3) = 1$



deterministic automaton
(Brzozoswki)

Other Results

Corollary : the class of regular languages is closed by complementation and intersection

proof: for negation, revert terminal and non-terminal states
intersection implied by union and negation

Remark: derivation works with negation and intersection operators in expressions (but explosive):

$a^{-1}(\neg e) = \neg a^{-1}(e) \qquad a^{-1}(e \cap e') = a^{-1}(e) \cap a^{-1}(e')$

Reciprocal theorem: every language recognized by an automaton is regular

proof : iteratively construct a regular expression by transitive closure of a state-indexed square matrix of expressions, starting from automata transitions

Exponential Explosion Danger Both Ways !

- The Brzozowski finite automation may be exponentially bigger than e
- For some expressions, this is true for all equivalent deterministic automata
- Conversely, for some automata, any equivalent regular expression may be exponentially bigger

Agenda

1. Regular Expressions
2. Derivatives and Translation to Automata
3. The Berry-Sethi Algorithm
4. Determinization and minimization
5. From Automata to Boolean Circuits
6. Automatic Sequences and Transcendental Numbers

Linear Expression

- an expression is linear if it contains each letter at most once
- linearize expressions by uniquely indexing letters

$s_0 = (a_0 b_1 + b_2)^* b_3 a_4$

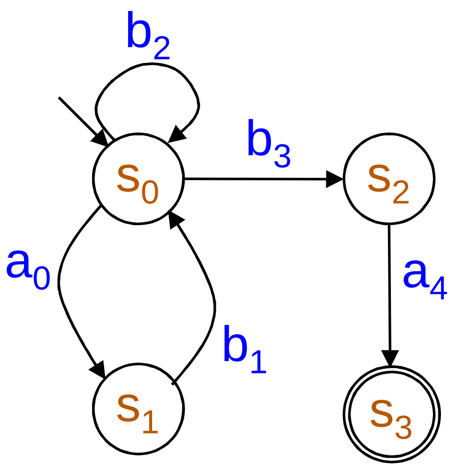
$a_0^{-1}(s_0) = b_1(a_0 b_1 + b_2)^* b_3 a_4 = s_1$

$b_2^{-1}(s_0) = (a_0 b_1 + b_2)^* b_3 a_4 = s_0$

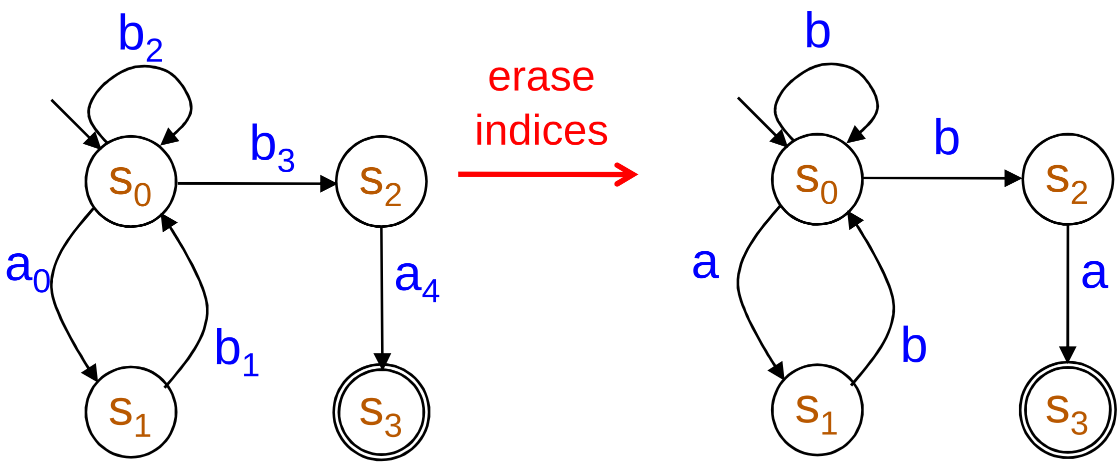
$b_3^{-1}(s_0) = a_4 = s_2$

$b_1^{-1}(s_1) = s_0$

$a_4^{-1}(s_0) = \varepsilon = s_3$



Non-Deterministic Automaton



Non-deterministic automaton recognizing $L(e_0)$

Linear Derivatives Are Simple

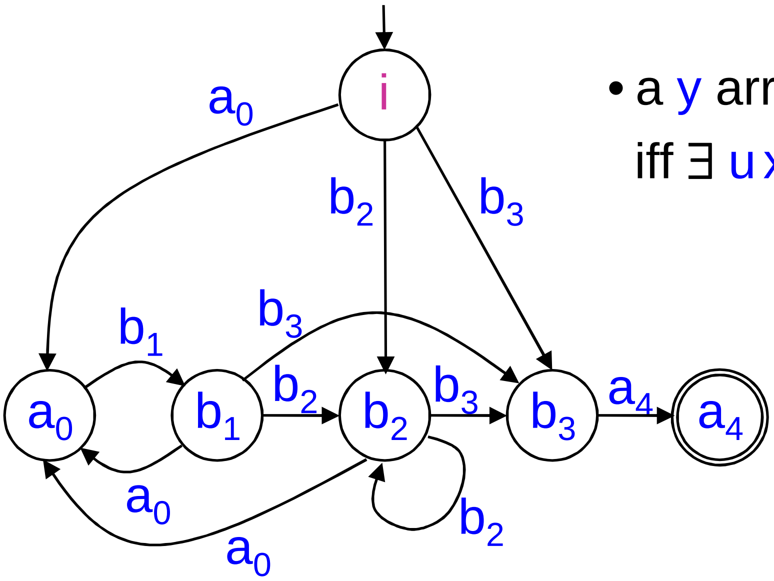
Theorem : if e is linear, then a non-null derivative is fully characterized by its last letter : all derivatives of the form $ux^{-1}(e)$ are null or equal

Proof : by induction on $|e|$ (size of e). Assume $ux^{-1}(e)$ non-null.

1. obvious if $e = x$ (then $|u| = 0$)
2. if $e = e_1 + e_2$, then x appears either in e_1 or in e_2 but not in both
   - if x appears in e_1, then $ux^{-1}(e) = ux^{-1}(e_1)$ with $|e_1| < |e|$
   - if x appears in e_2, then $ux^{-1}(e) = ux^{-1}(e_2)$ with $|e_2| < |e|$
3. if $e = e_1 \cdot e_2$, similar proof by computing $ux^{-1}(e_1 \cdot e_2)$ as a sum of two terms such that only one can contain x
4. if $e = e_1^*$, similar proof to case 3

The Automaton of a Linear Expression

$s_0 = (a_0 b_1 + b_2)^* b_3 a_4$



- an arrow from i to x if $\exists\, u = xu \in L(s_0)$
- a y arrow from x to y iff $\exists\, u\,xy\,v \in L(s_0)$

The Berry-Sethi Algorithm

Building a non-deterministic automaton for e linear :

1. build an initial state plus a state per letter x, representing $ux^{-1}(e)$ for all u
2. add an arrow labeled x from the initial state to any state x such that x can appear as first letter, i.e., $x^{-1}(e) \neq 0$
3. build an arrow labeled y from state x to state y iff $ux^{-1}(e) \neq 0$ and $uxy^{-1}(e) \neq 0$, i.e. there exists $uxyv \in L(e)$, i.e. iff y may follow x in $L(e)$
4. set state x terminal if there exists a word $ux \in L(e)$

Computing first et last

- first (0) = first(1) = ∅
- first (a) = {a}
- first(e+e') = first(e) ∪ first(e')
- first(e·e') = first(e) if ε(e) = 0
  = first(e) ∪ first(e') otherwise
- first(e*) = first(e)

first letters that e can write

- last (0) = last(1) = ∅
- last (a) = {a}
- last(e+e') = last(e) ∪ last(e')
- last(e·e') = last(e') if ε(e')=0
  = last(e) ∪ last(e') otherwise
- last(e*) = last(e)

last letters that e can write

Computing whether y can follow x

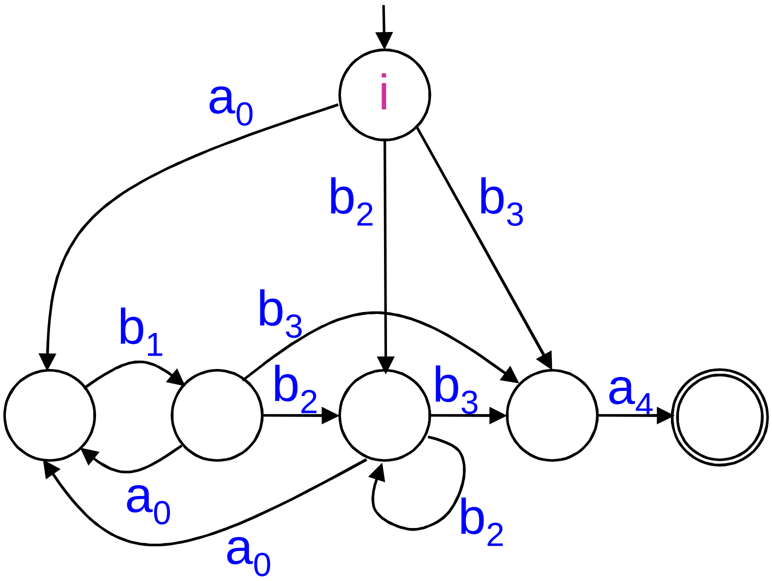
follow(e, x, y) = 1 iff $\exists$ w x y w' $\in$ L(e)

- follow(0, x, y) = follow(1, x, y) = follow(a, x, y) = 0
- follow(e+e', x, y) = follow(e, x, y) $\vee$ follow(e', x, y)
- follow(e · e', x, y) = follow(e, x, y)
  + follow(e', x, y)
  + (x $\in$ last(e) · y $\in$ first(e'))
- follow(e*, x, y) = follow(e, x, y)
  + (x $\in$ last(e) · y $\in$ first(e))

Exercice : find more efficient formulae

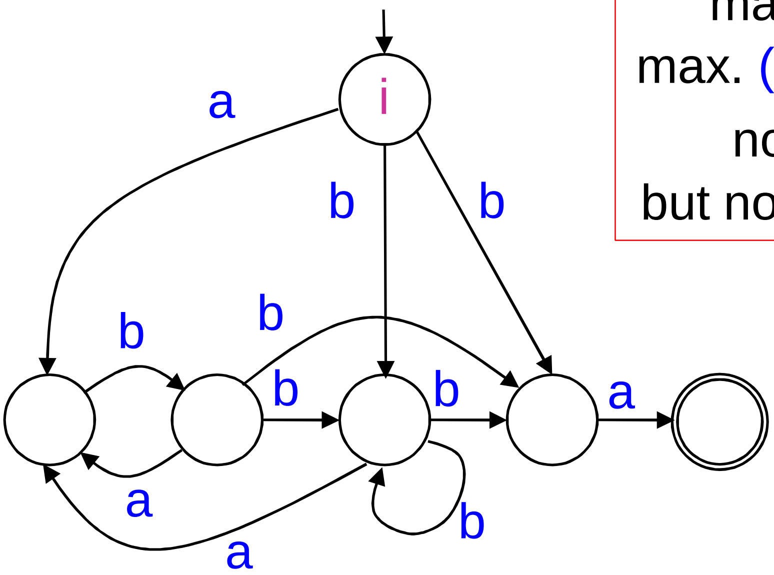
The Berry-Sethi Algorithm

$s_0 = (a_0 b_1 + b_2)^* b_3 a_4$



Erasing Indices

e = (ab+b)*ba



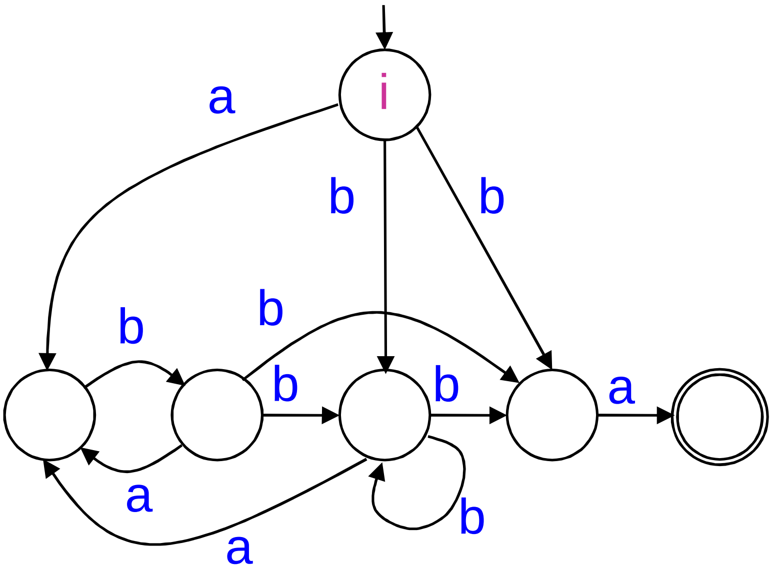
for n letter occurrences
max. n+1 states
max. $(n+1)^2$ transitions

no explosion !
but non-deterministic..

G. Berry, L3, Edinburgh, 30/09/2010 33

Optimizing Away the Initial State

e = (ab+b)*ba



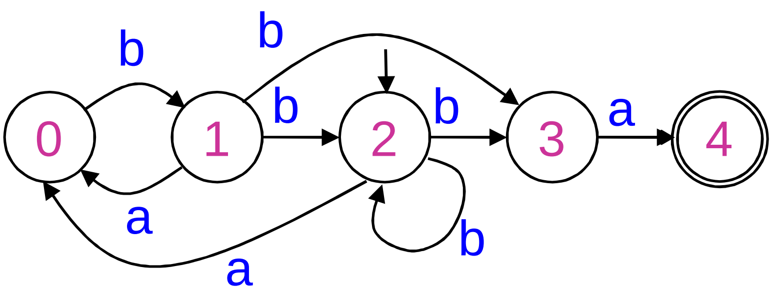
Agenda

1. Regular Expressions
2. Derivatives and Translation to Automata
3. The Berry-Sethi Algorithm
4. Determinization and minimization
5. From Automata to Boolean Circuits
6. Automatic Sequences and Transcendental Numbers

Determinization

e = (ab+b)*ba

00100 : initial

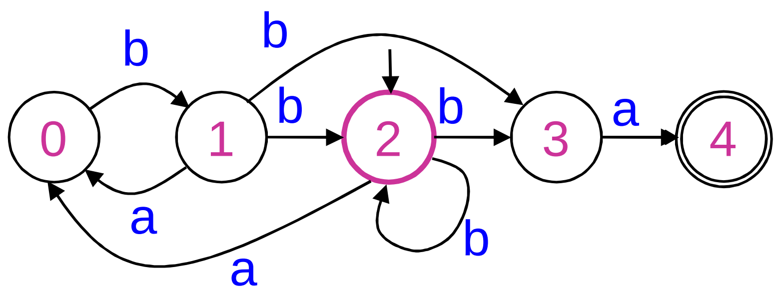


Determinization

$e = (ab+b)^*ba$

00100 : initial

00100 : a →

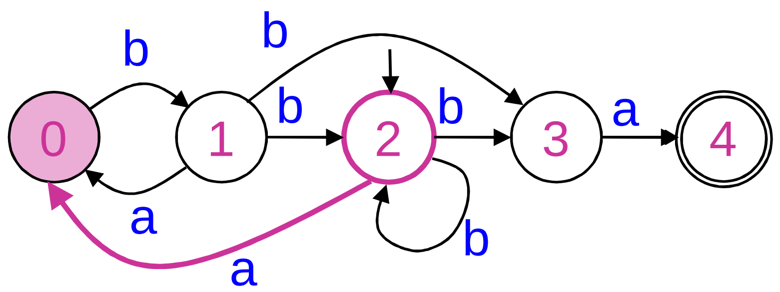


Determinization

00100 : a → 10000

e = (ab+b)*ba

00100 : initial

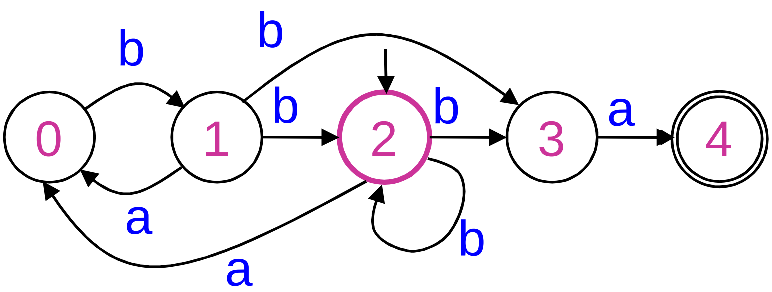


Determinization

$e = (ab+b)^{*}ba$

00100 : initial

00100 : a → 10000
b →

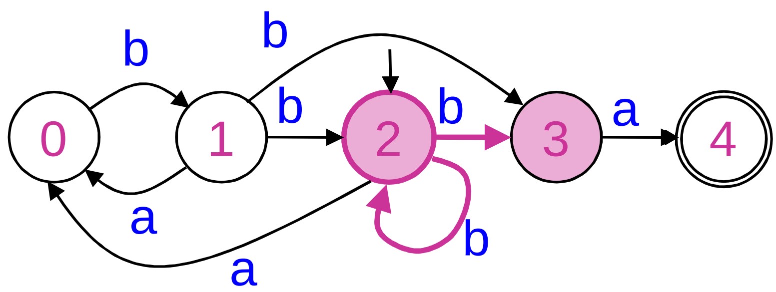


Determinization

$e = (ab+b)^*ba$

00100 : initial

00100 : a → 10000
b → 00110



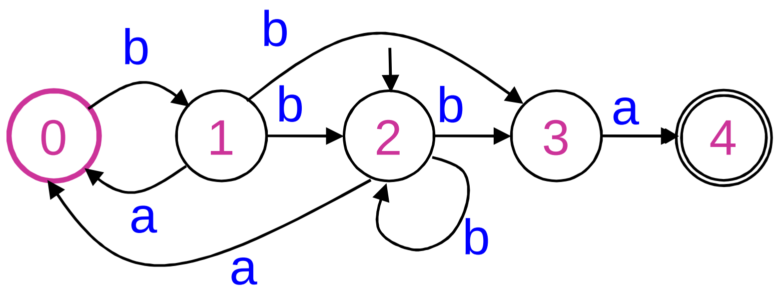
Determinization

e = (ab+b)*ba

00100 : initial

00100 : a → 10000
b → 00110

10000 : b →



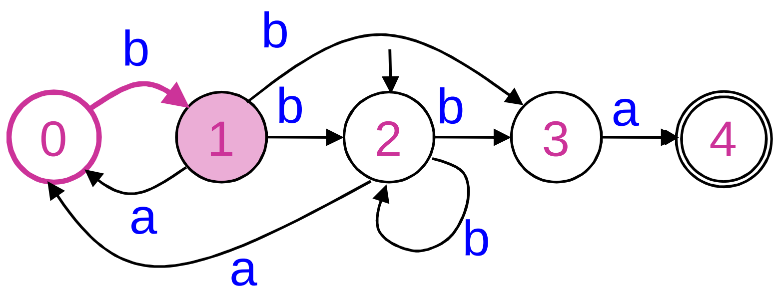
Determinization

e = (ab+b)*ba

00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000



G. Berry, L3, Edinburgh, 30/09/2010 42

Determinization

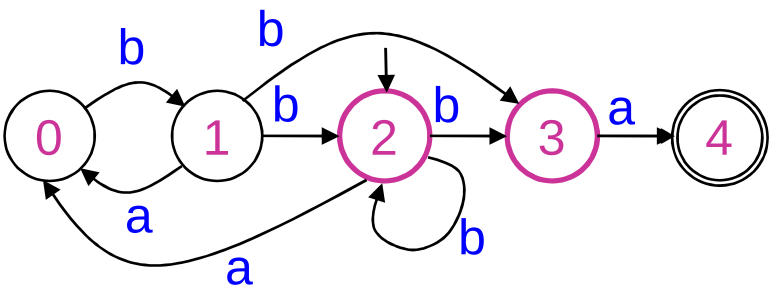
e = (ab+b)*ba

00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a →



Determinization

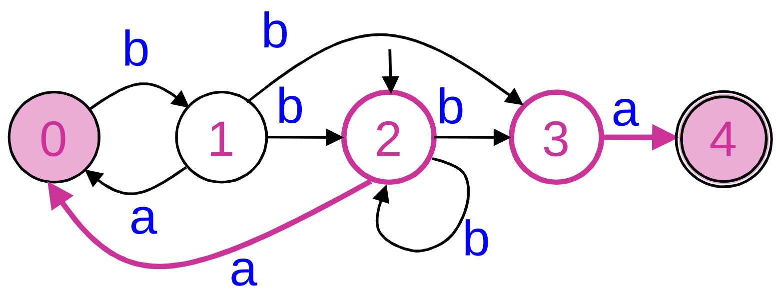
e = (ab+b)*ba

00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → 10001



Determinization

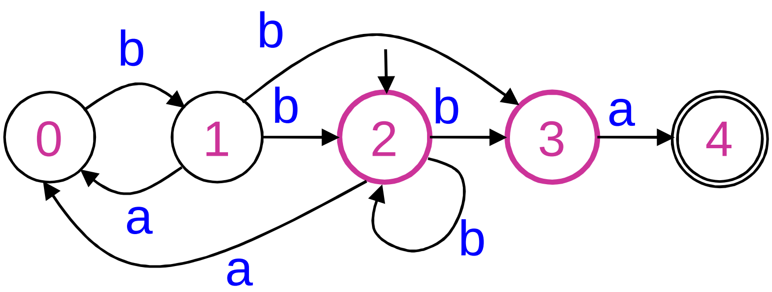
e = (ab+b)*ba

00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → <u>10001</u>
b →



Determinization

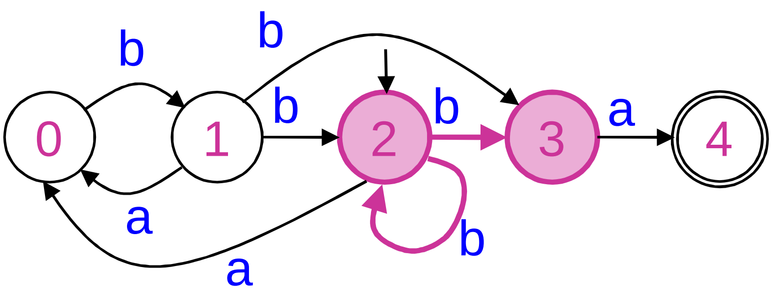
e = (ab+b)*ba

00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → 10001
b → 00110



Determinization

$e = (ab+b)^{*}ba$

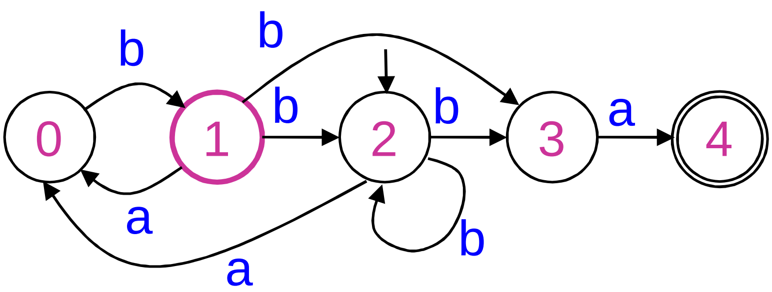
00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → <u>10001</u>
b → 00110

01000 : a



Determinization

e = (ab+b)*ba

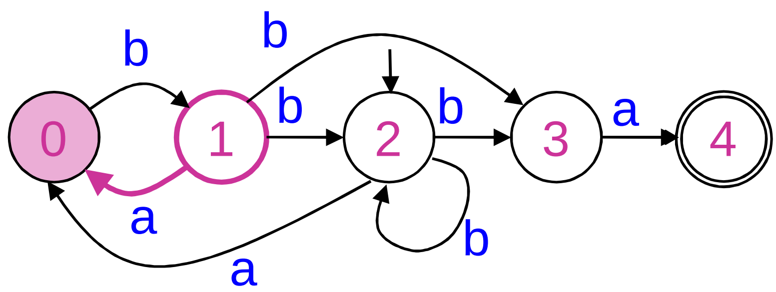
00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → 10001
b → 00110

01000 : a → 10000



Determinization

e = (ab+b)*ba

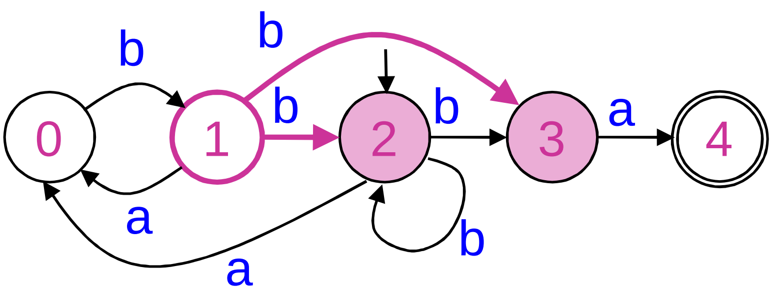
00100 : initial

00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → 10001
b → 00110

01000 : a → 10000
b → 00110



Determinization

$e = (ab+b)^{*}ba$

00100 : initial

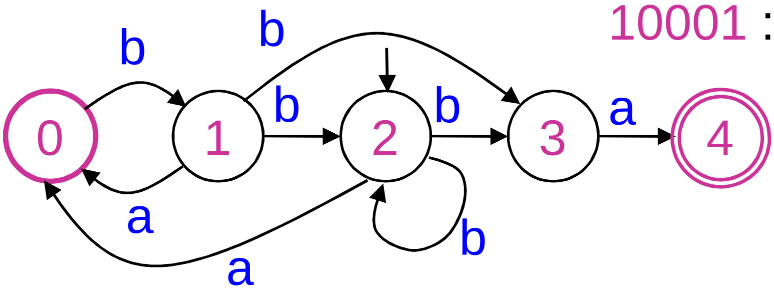
00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → <u>10001</u>
b → 00110

01000 : a → 10000
b → 00110

10001 : b



Determinization

e = (ab+b)*ba

00100 : initial

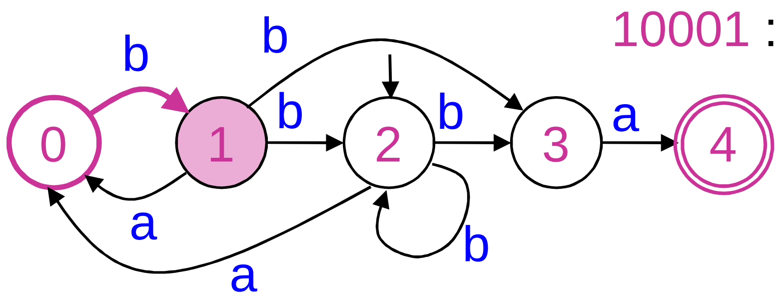
00100 : a → 10000
b → 00110

10000 : b → 01000

00110 : a → <u>10001</u>
b → 00110

01000 : a → 10000
b → 00110

10001 : b → 01000



Determinization

$e = (ab+b)^*ba$

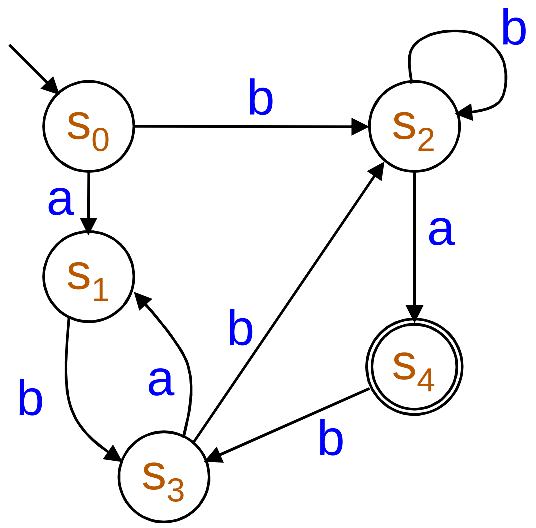
$s_0 = 00100$

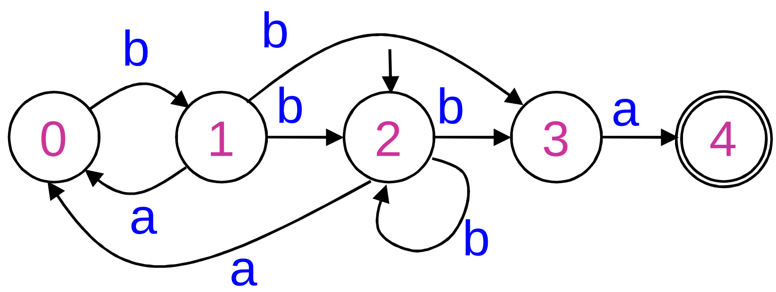
$s_1 = 10000$

$s_2 = 00110$

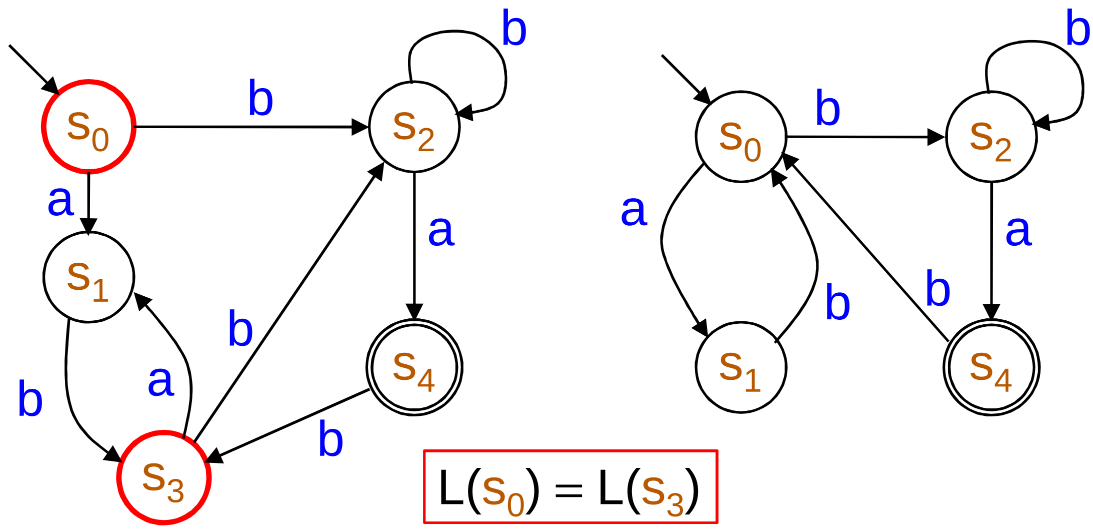
$s_3 = 01000$

$s_4 = 10001$





Minimizing Deterministic Automata



- Theorem : for any regular language L, there exists a minimal deterministic automaton recognizing L.
  Idea : states= nonempty $u^{-1}(L)$
- Bonus : there exists a fast minimization algorithm

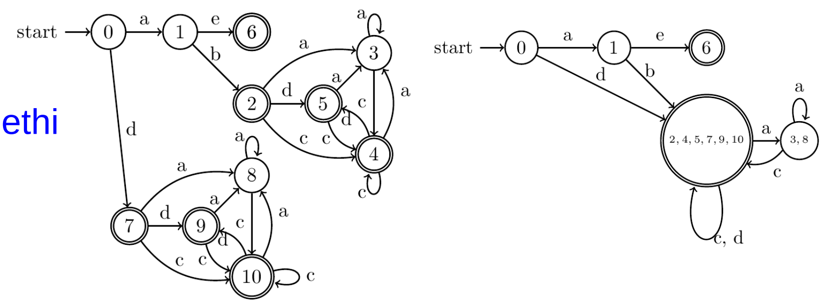
Summary

- A regular language is generated (or recognized) by a regular expression or by a finite automaton
- One can construct a deterministic automaton from a regular explosion, but risking an exponential explosion
- One can efficiently construct a non-deterministic aotomaton of maximal size n^2 for an expression of size n
- On can determinize this automaton, but with potential exponential explosion
- One can efficiently minimize a deterministic automaton (unless one has exploded beforehand)

Other Results

- In general, there is no minimal non-deterministic automaton for a given language
- One can improve upon the Berry-Sethi construction (cf. Antimirov, Champarnaud, Razet)

Berry-Sethi



Antimirov Champarnaud-Ziadi

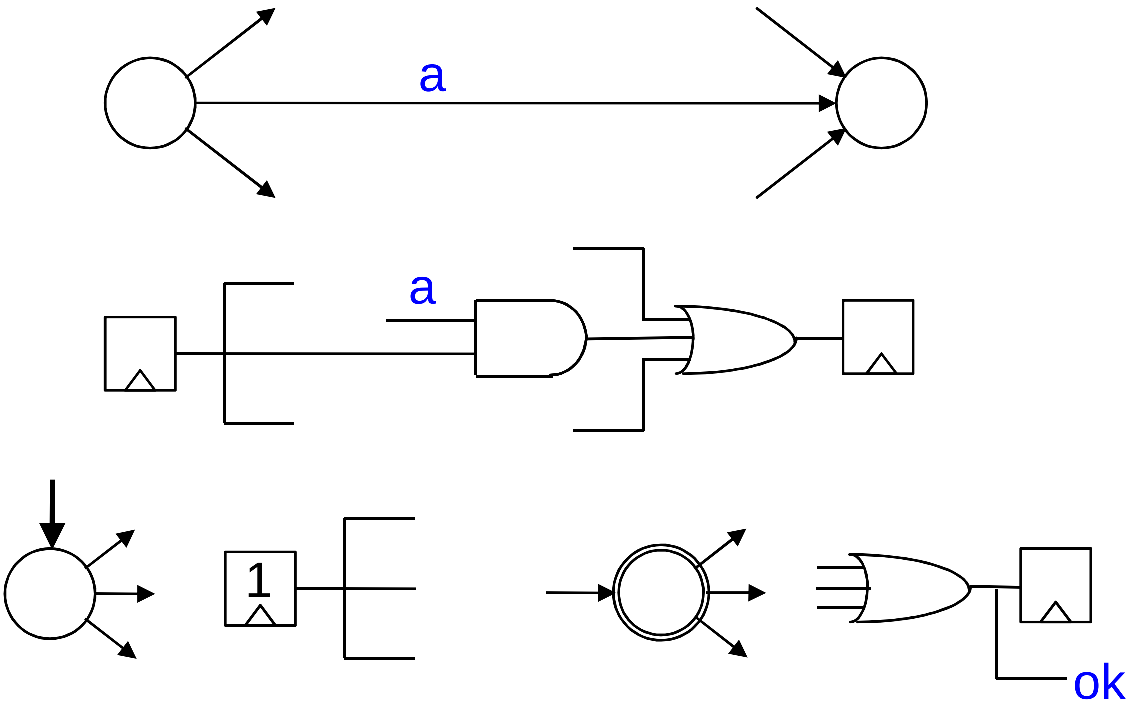
$$E = a(b(a^*c + d)^* + e) + d(a^*c + d)^*$$

G. Berry, L3, Edinburgh, 30/09/2010 55

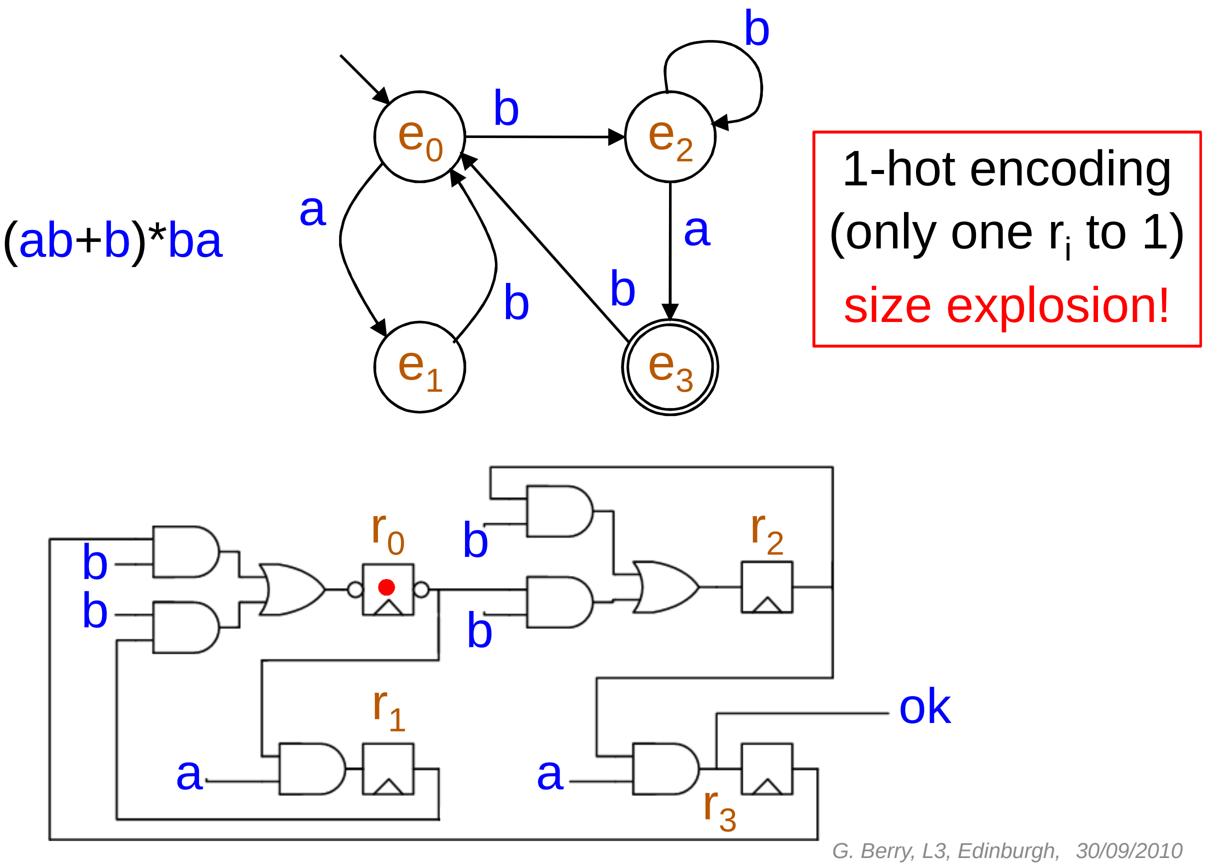
Agenda

1. Derivatives and Translation to Automata
2. The Berry-Sethi Algorithm
3. Determinization and minimization
4. From Automata to Boolean Circuits
5. Automatic Sequences and Transcendental Numbers

Implementation by Boolean Circuits

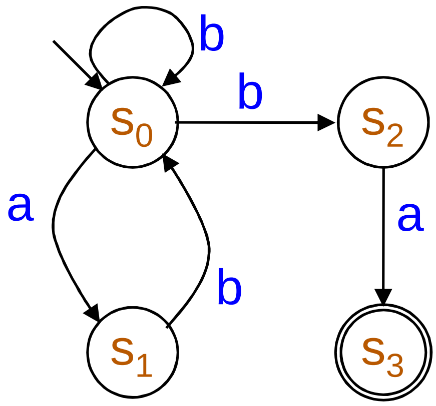


The Deterministic Case

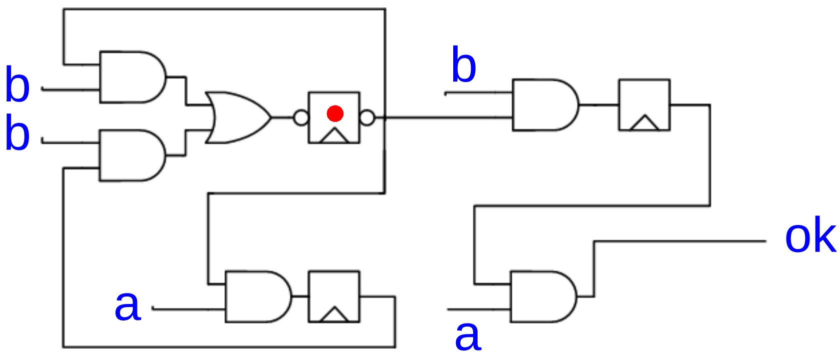


The Non-Deterministic Case

(ab+b)*ba

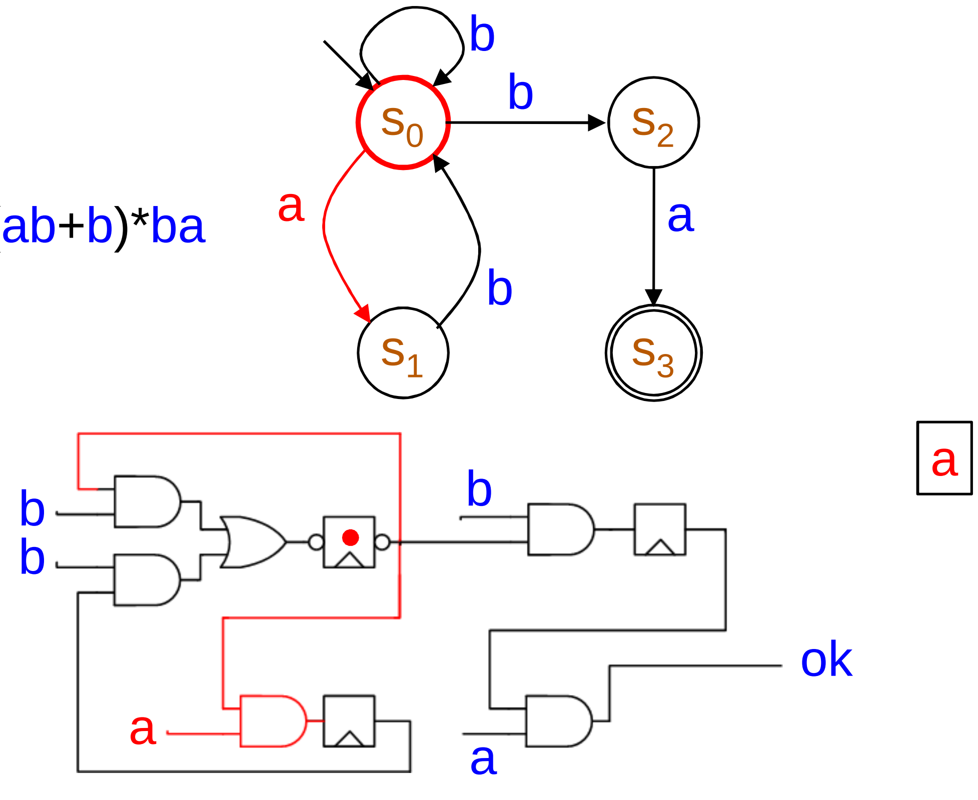


no size explosion
⇒ much better!

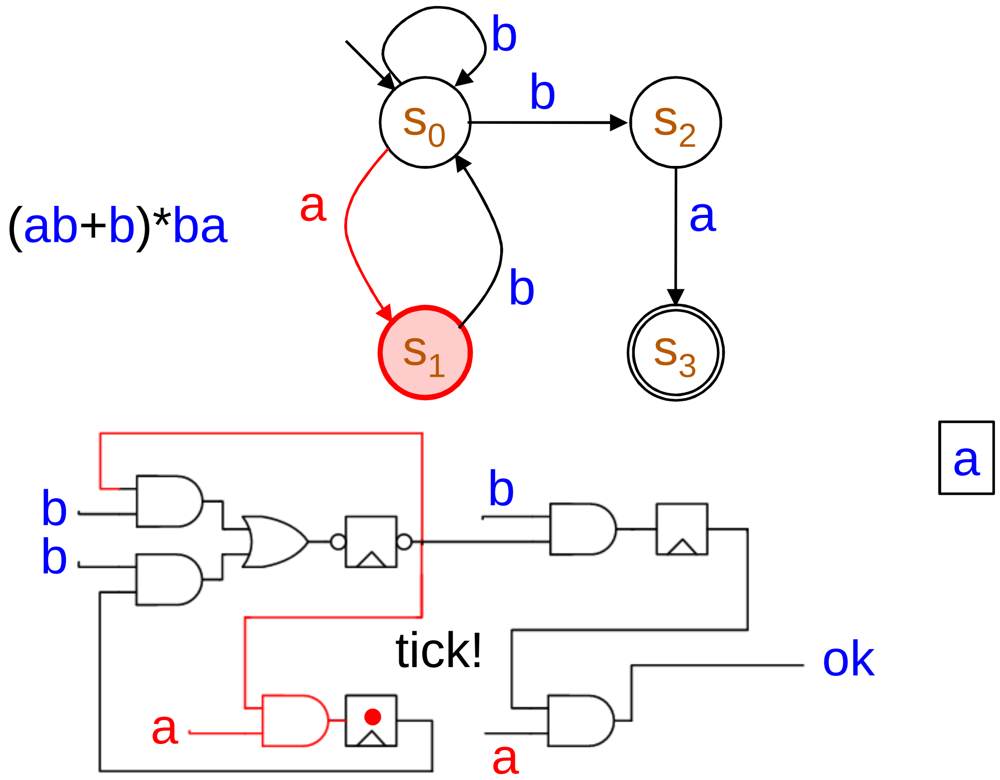


On-The-Fly Subset Construction

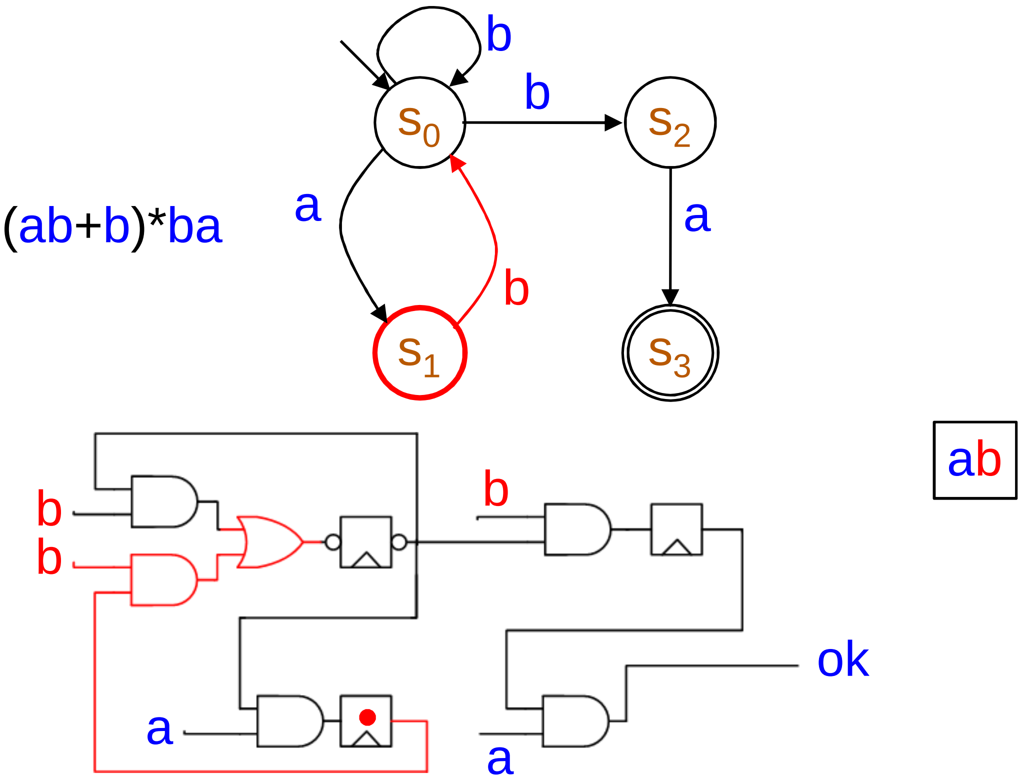
(ab+b)*ba



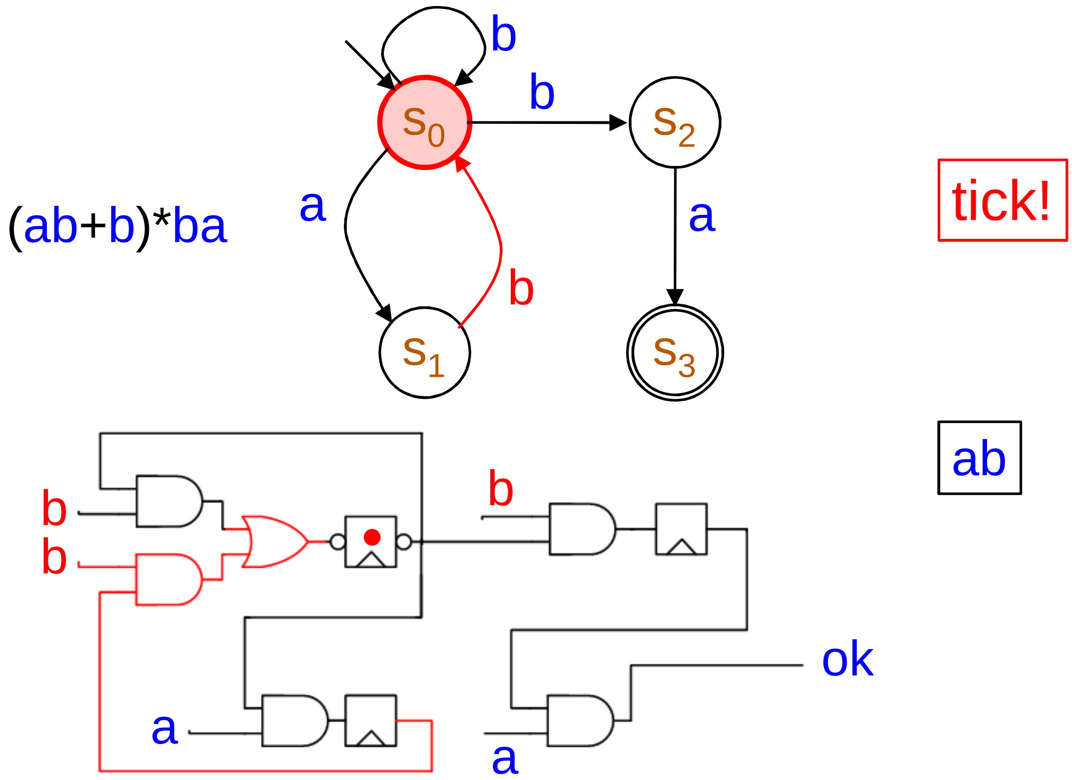
On-The-Fly Subset Construction



On-The-Fly Subset Construction

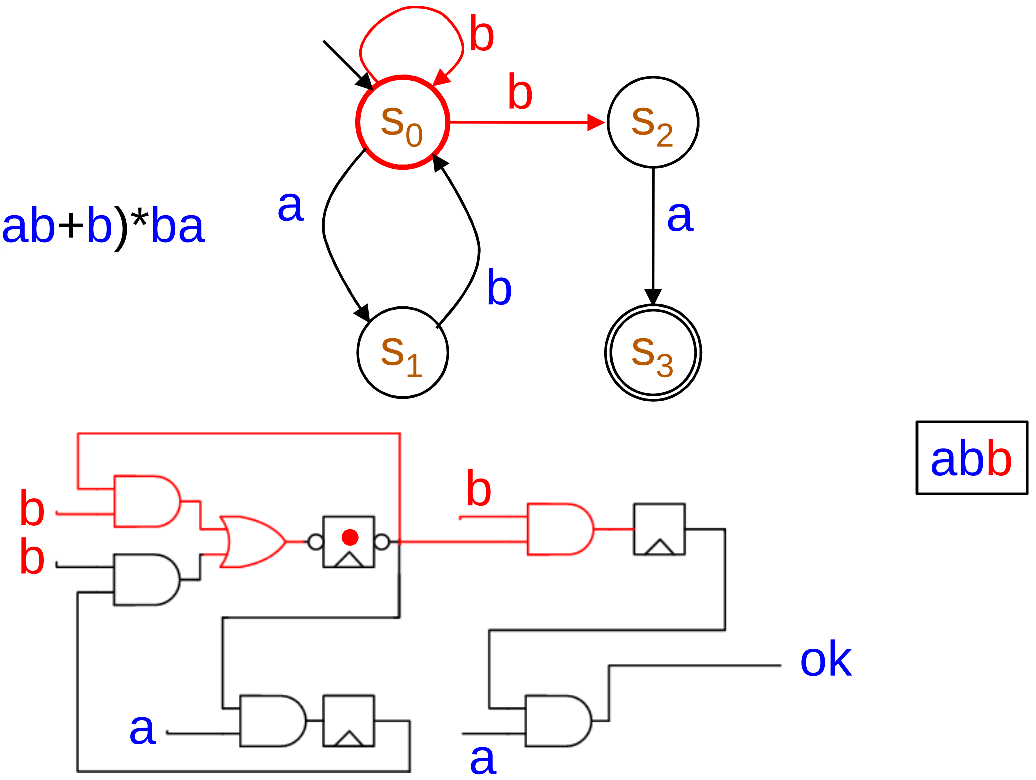


On-The-Fly Subset Construction



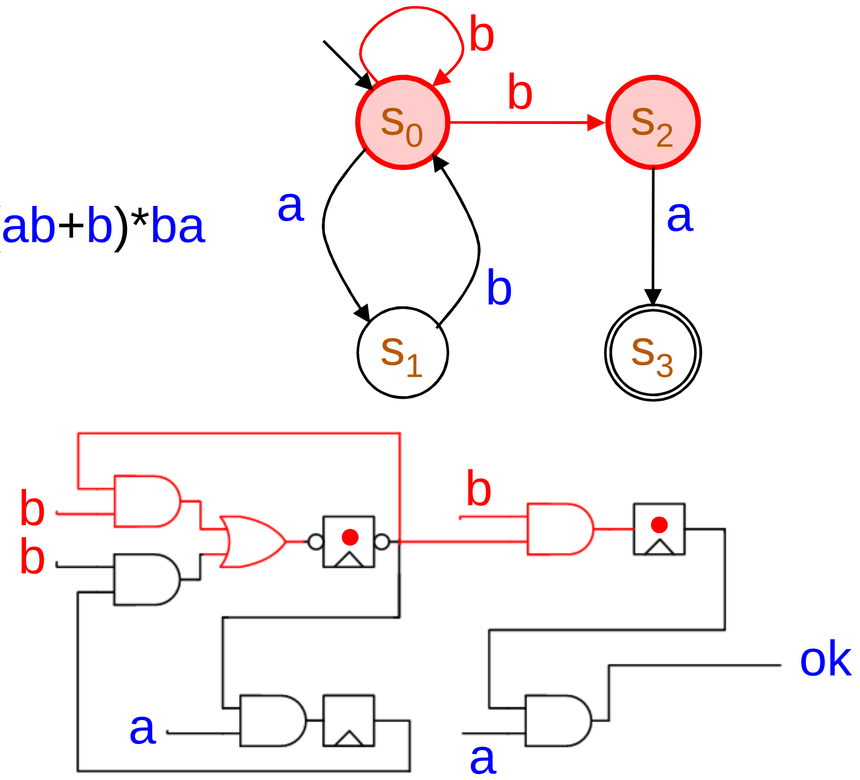
On-The-Fly Subset Construction

(ab+b)*ba



On-The-Fly Subset Construction

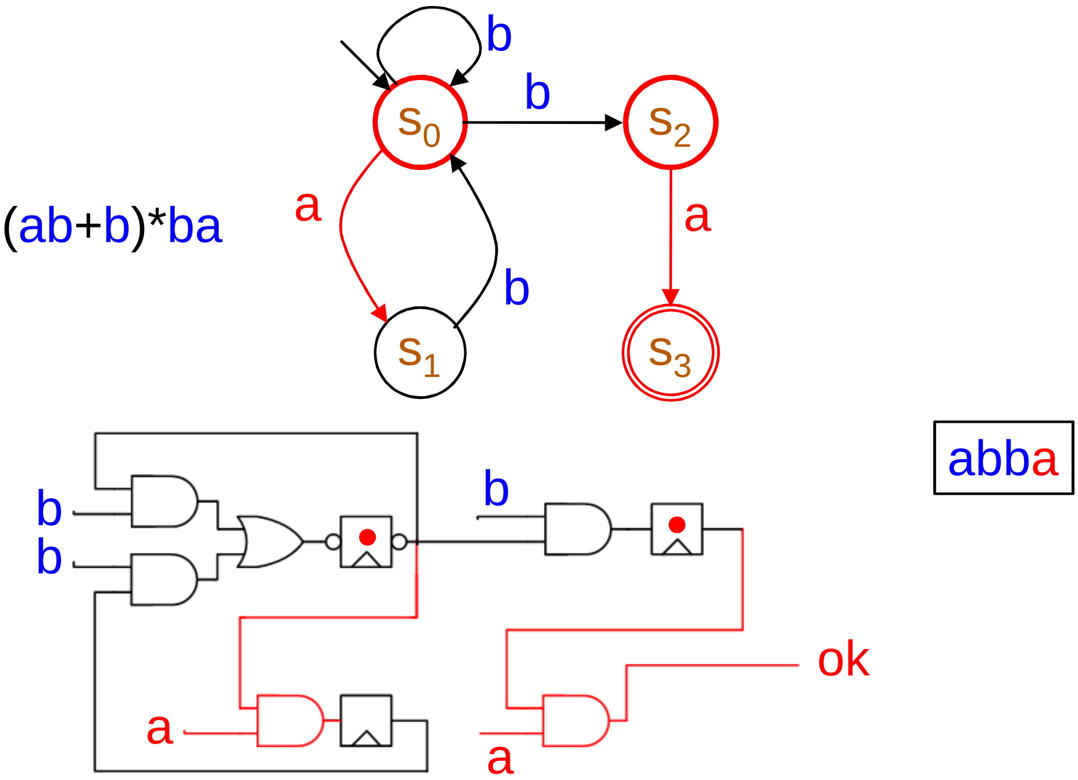
(ab+b)*ba



tick!

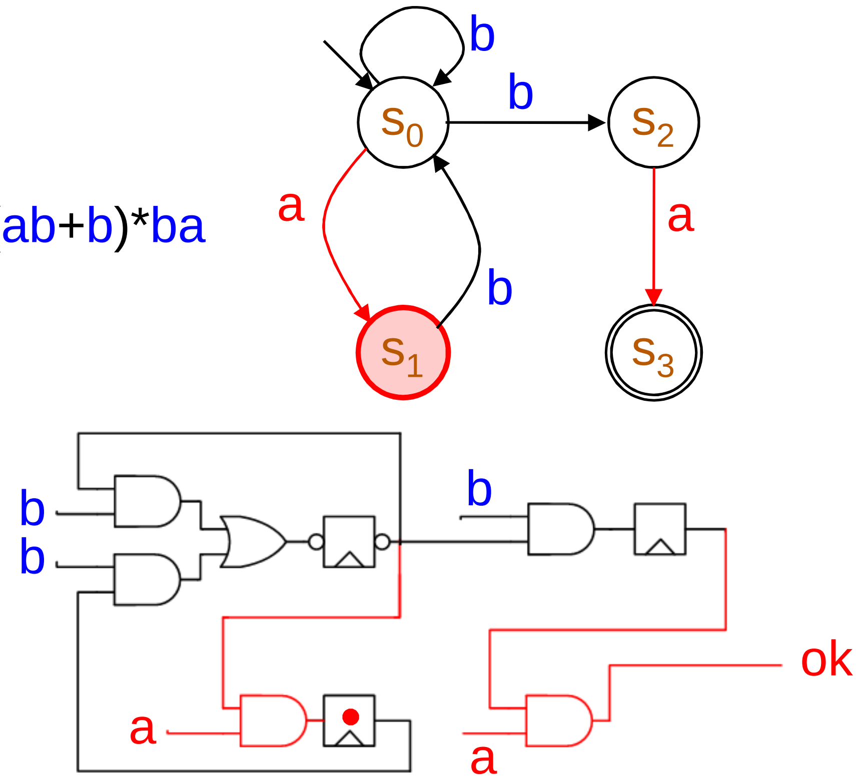
abb

On-The-Fly Subset Construction



On-The-Fly Subset Construction

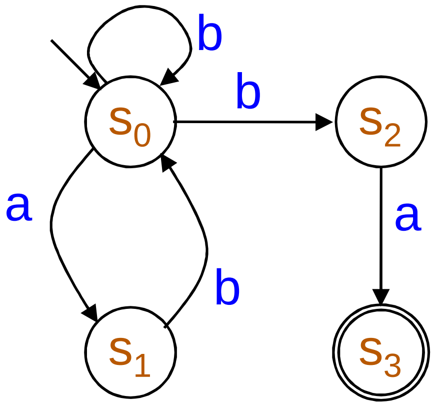
(ab+b)*ba



tick!

abba

Esterel v7 implementation




```
module Autom :
input a, b ;
output ok;
local {r0, r1, r2} : reg;
refine r0 : init true;
sustain {
   next r0 <= (r0 or r1) and b,
   next r1 <= r0 and a,
   next r2 <= r0 and b,
   ok <= r2 and a }
end module
```

Linear compiling into C, C++, VHDL, Verilog, etc.

Fundamental Practical Result

Any regular expression of size n is recognized by a circuit with $n+1$ registers and at most n^2 gates

- Scales up in size, is always superfast
- Almost always better than determinization
- The circuit can be cleverly optimized (see Synchronous Languages appendix)

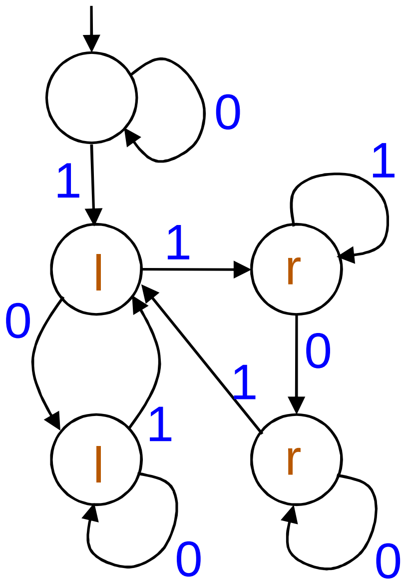
Agenda

1. Regular Expressions
2. Derivatives and Translation to Automata
3. The Berry-Sethi Algorithm
4. Determinization and minimization
5. From Automata to Boolean Circuits
6. Automatic Sequences and Transcendental Numbers

Paper-Folding Automatic Sequence

- Fold a paper, always on the same size
- Unfold it and list 0 / 1 direction changes
- Play the automaton from left to right

	1000 : l
1 : l	1001 : l
10 : l	1010 : l
11 : r	1011 : r
100 : l	1100 : r
101 : l	1101 : l
110 : r	1110 : r
111 : r	1111 : r



automatic sequence

Paper Folding Transcendence

- From an automatic sequence s_i, build the real number
  $0,s_1s_2...s_n...$
- Theorem : this number is either rational or transcendtal

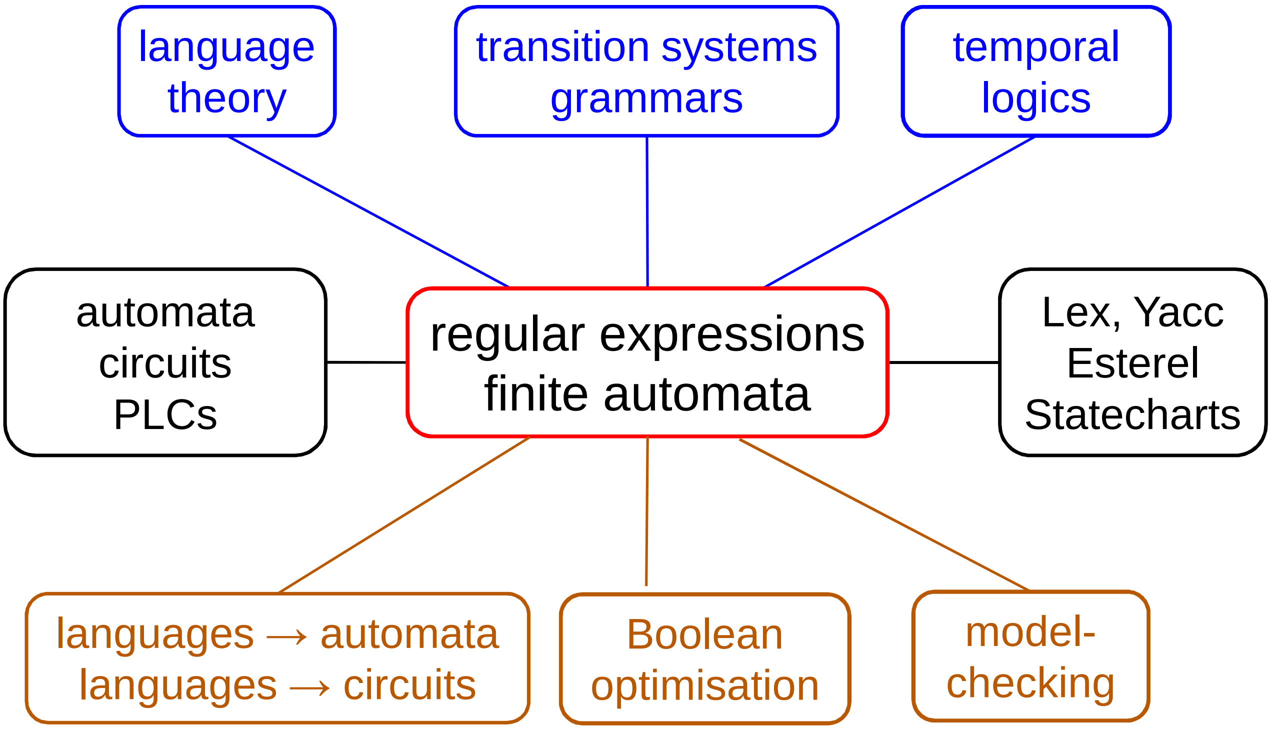
	1000 : l
1 : l	1001 : l
10 : l	1010 : l
11 : r	1011 : r
100 : l	1100 : r
101 : l	1101 : l
110 : r	1110 : r
111 : r	1111 : r

l=1, r=0

PF = 0,1101100111001001...
is transcendental

cf. Allouche, Mendès-France, Rauzy, etc.

The Finite-State Systems Map



References

Elements of Automata Theory
Jacques Sakarovitch
Cambridge University Press, 2009

From Regular Expressions to Deterministic Automata
Gérard Berry and Ravi Sethi
Theoretical Computer Science 48 (1986) 117-126.

Automatic Sequences: Theory, Applications, Generalizations
Jean-Paul Allouche and Jeffrey Shallit
Cambridge University Press (2003)