MODULAR DESCRIPTIONS OF REGULAR FUNCTIONS

PAUL GASTIN
LSV, ENS PARIS-SACLAY

CAI 2019

STRING TO STRING FUNCTIONS

STRING TO STRING FUNCTIONS

- Erase comments from a LaTeX file

  First \emph{sequential} functions % one-way input-deterministic

  We suffer only 2\% of failures.

STRING TO STRING FUNCTIONS

- Erase comments from a LaTeX file

  First \emph{sequential} functions % one-way input-deterministic

  We suffer only 2\% of failures.

- Increment numbers in a file

  Exams will take place in May 2019. Students should …

  Exams will take place in May 2020. Students should …

STRING TO STRING FUNCTIONS

- Erase comments from a LaTeX file

  First \emph{sequential} functions % one-way input-deterministic

  We suffer only 2\% of failures.

- Increment numbers in a file

  Exams will take place in May 2019. Students should …

  Exams will take place in May 2020. Students should …

- Reorder arguments (bibtex -> bbl)

  Author = {Engelfriet, Joost and Hoogeboom, Hendrik Jan},

  Year = {2001},

  [EH01] Joost Engelfriet and Hendrik Jan Hoogeboom.

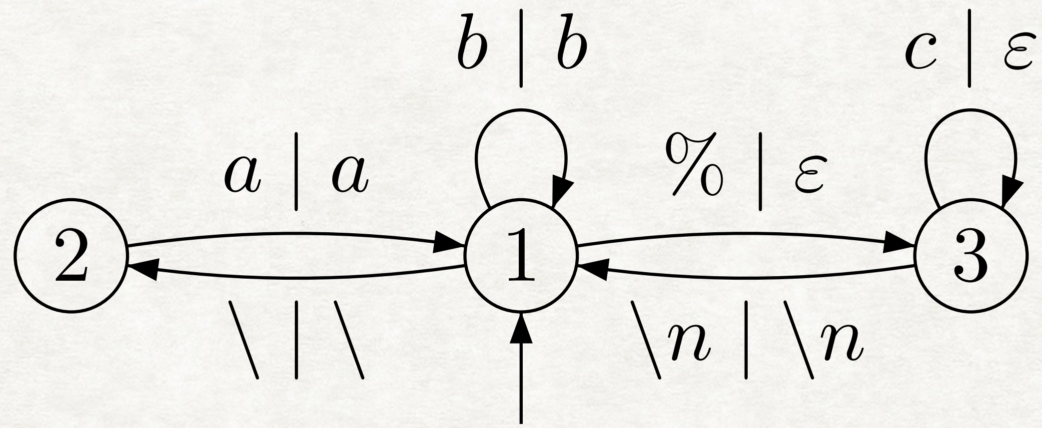
  [EH01] J. Engelfriet and H.J. Hoogeboom.

SUMMARY

- Operational models (transducers)
  - 1DFT = sequential functions
  - f1NFT = rational functions
  - 2DFT = regular functions
  - Transducers with registers
- Modular descriptions
  - Rational expressions
  - Composition

SEQUENTIAL FUNCTIONS - 1DFT

- Deterministic left to right parsing of the input
- Produce output along the way



$a \in \Sigma$

$b \in \Sigma \setminus \{\backslash, \%\}$

$c \in \Sigma \setminus \{\backslash n\}$

First \emph{sequential} functions % one-way input-deterministic

We suffer only 2\% of failures.

SEQUENTIAL FUNCTIONS - 1DFT

Exams will take place in May 2019. Students should …

Exams will take place in May 2020. Students should …

Increment is not sequential if the least significant bit (lsb) is on the right

SEQUENTIAL FUNCTIONS - 1DFT

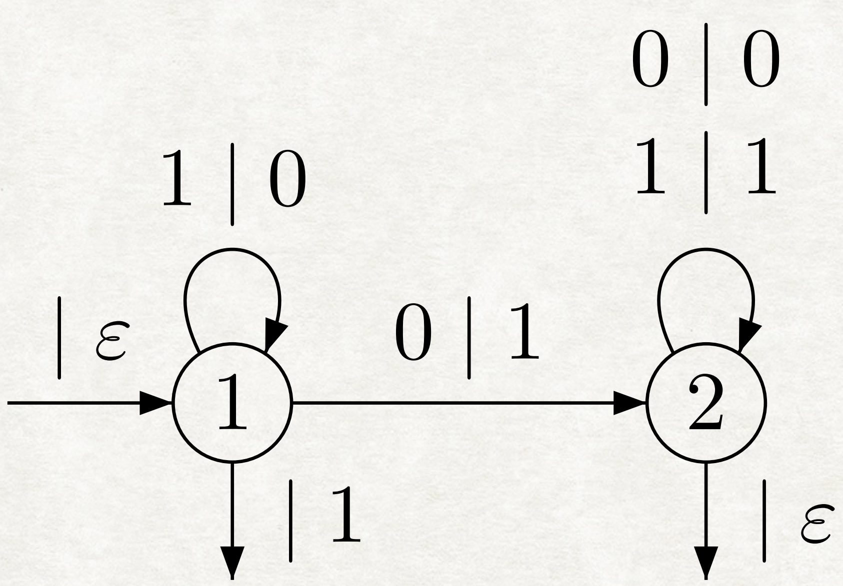
Exams will take place in May 2019. Students should …

Exams will take place in May 2020. Students should …

Increment is not sequential if the least significant bit (lsb) is on the right

---

Increment of a number with lsb on the left is sequential.



1111100101101

0000010101101

11111111

000000001

SEQUENTIAL FUNCTIONS - 1DFT

- Deterministic left to right parsing of the input
- Produce output along the way

SEQUENTIAL FUNCTIONS - 1DFT

- Deterministic left to right parsing of the input
- Produce output along the way

➡ Sequential functions are closed under composition of functions

SEQUENTIAL FUNCTIONS - 1DFT

- Deterministic left to right parsing of the input
- Produce output along the way

➡ Sequential functions are closed under composition of functions

➡ Sequential transducers can be minimized (canonical)

SEQUENTIAL FUNCTIONS - 1DFT

- Deterministic left to right parsing of the input
- Produce output along the way

➡ Sequential functions are closed under composition of functions

➡ Sequential transducers can be minimized (canonical)

➡ Equivalence is decidable for sequential transducers

SUMMARY

- Operational models (transducers)
  - 1DFT = sequential functions
  - f1NFT = rational functions
  - 2DFT = regular functions
  - Transducers with registers
- Modular descriptions
  - Rational expressions
  - Composition

RATIONAL FUNCTIONS - f1NFT

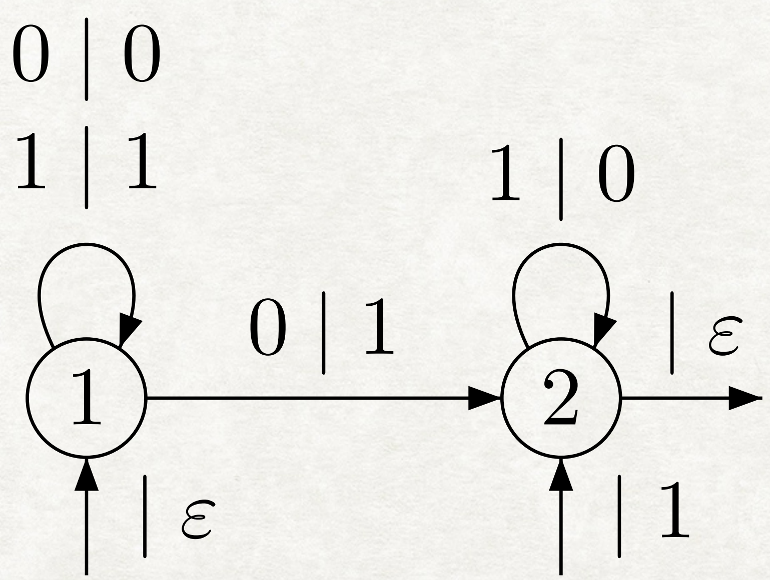
RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way

Increment of a number with lsb on the right is a rational function



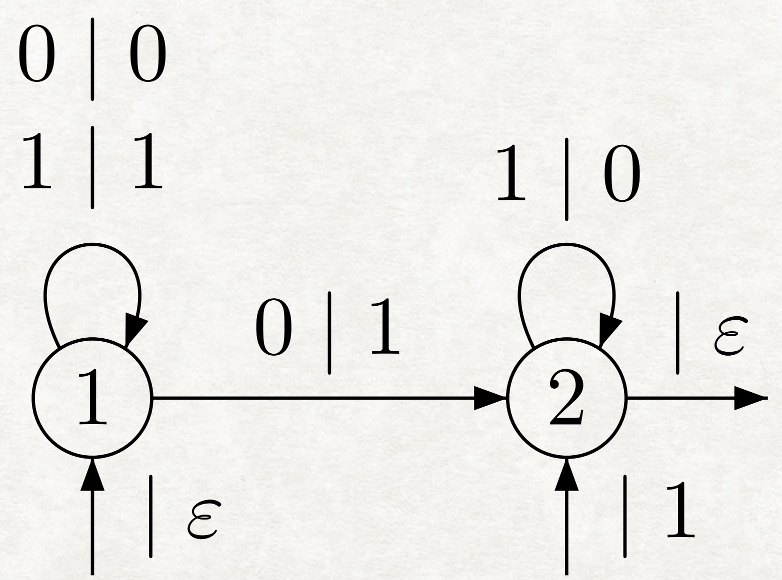
1011010011111
1011010100000

11111111
100000000

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

Increment of a number with lsb on the right is a rational function



1011010011111
1011010100000

11111111
100000000

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

RATIONAL FUNCTIONS - f1NFT

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

Let $w = a_1 a_2 \ldots a_n \in \mathsf{dom}(\mathcal{A})$ with $n > 2m^2$.
Assume $\mathcal{A}$ is functional on all words of length $< n$.

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots p_{n-1} \xrightarrow{a_n} p_n$$
$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n$$

Let $0 \leq i < j < k \leq n$ with $(p_i, q_i) = (p_j, q_j) = (p_k, q_k)$.

$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i \xrightarrow{a_{i+1} \cdots a_j | x_2} p_j \xrightarrow{a_{j+1} \cdots a_k | x_3} p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$
$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i \xrightarrow{a_{i+1} \cdots a_j | y_2} q_j \xrightarrow{a_{j+1} \cdots a_k | y_3} q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

[5] Berstel, Transductions and Context-Free Languages, 1979

RATIONAL FUNCTIONS - f1NFT

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

Let $w = a_1 a_2 \ldots a_n \in \mathsf{dom}(\mathcal{A})$ with $n > 2m^2$.
Assume $\mathcal{A}$ is functional on all words of length $< n$.
Let $0 \leq i < j < k \leq n$ with $(p_i, q_i) = (p_j, q_j) = (p_k, q_k)$.

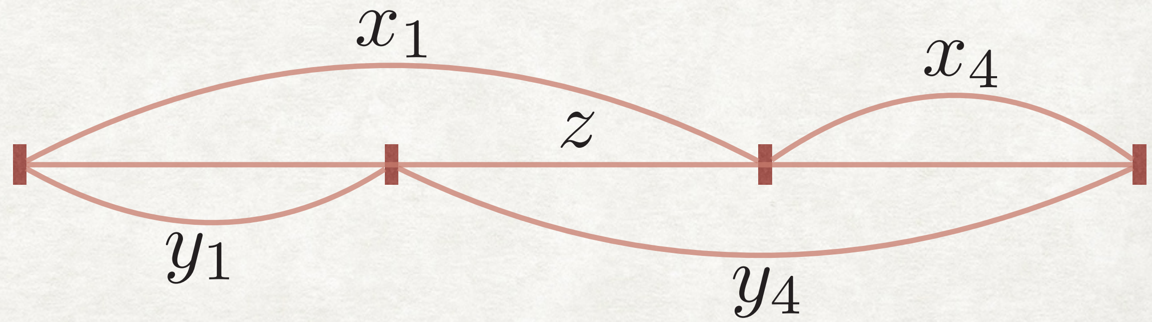
$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i \xrightarrow{a_{i+1} \cdots a_j | x_2} p_j \xrightarrow{a_{j+1} \cdots a_k | x_3} p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$

$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i \xrightarrow{a_{i+1} \cdots a_j | y_2} q_j \xrightarrow{a_{j+1} \cdots a_k | y_3} q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i = p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$

$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i = q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

$$x_1 x_4 = y_1 y_4 \implies \begin{aligned} x_1 &= y_1 z \\ z x_4 &= y_4 \end{aligned}$$



[5] Berstel, Transductions and Context-Free Languages, 1979

RATIONAL FUNCTIONS - f1NFT

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

Let $w = a_1 a_2 \ldots a_n \in \mathsf{dom}(\mathcal{A})$ with $n > 2m^2$.
Assume $\mathcal{A}$ is functional on all words of length $< n$.
Let $0 \leq i < j < k \leq n$ with $(p_i, q_i) = (p_j, q_j) = (p_k, q_k)$.

$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i \xrightarrow{a_{i+1} \cdots a_j | x_2} p_j \xrightarrow{a_{j+1} \cdots a_k | x_3} p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$

$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i \xrightarrow{a_{i+1} \cdots a_j | y_2} q_j \xrightarrow{a_{j+1} \cdots a_k | y_3} q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

$$x_1 x_4 = y_1 y_4 \implies \begin{array}{rcl} x_1 & = & y_1 z \\ z x_4 & = & y_4 \end{array}$$

$$x_1 x_2 x_4 = y_1 y_2 y_4 \implies y_1 z x_2 x_4 = y_1 y_2 z x_4 \implies z x_2 = y_2 z$$

[5] Berstel, Transductions and Context-Free Languages, 1979

RATIONAL FUNCTIONS - f1NFT

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

Let $w = a_1 a_2 \ldots a_n \in \mathsf{dom}(\mathcal{A})$ with $n > 2m^2$.
Assume $\mathcal{A}$ is functional on all words of length $< n$.
Let $0 \leq i < j < k \leq n$ with $(p_i, q_i) = (p_j, q_j) = (p_k, q_k)$.

$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i \xrightarrow{a_{i+1} \cdots a_j | x_2} p_j \xrightarrow{a_{j+1} \cdots a_k | x_3} p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$

$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i \xrightarrow{a_{i+1} \cdots a_j | y_2} q_j \xrightarrow{a_{j+1} \cdots a_k | y_3} q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

$$x_1 x_4 = y_1 y_4 \implies \begin{array}{rcl} x_1 & = & y_1 z \\ z x_4 & = & y_4 \end{array}$$

$$x_1 x_2 x_4 = y_1 y_2 y_4 \implies y_1 z x_2 x_4 = y_1 y_2 z x_4 \implies z x_2 = y_2 z$$

$$x_1 x_3 x_4 = y_1 y_3 y_4 \implies z x_3 = y_3 z$$

[5] Berstel, Transductions and Context-Free Languages, 1979

RATIONAL FUNCTIONS - f1NFT

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

Let $w = a_1 a_2 \ldots a_n \in \mathsf{dom}(\mathcal{A})$ with $n > 2m^2$.
Assume $\mathcal{A}$ is functional on all words of length $< n$.
Let $0 \leq i < j < k \leq n$ with $(p_i, q_i) = (p_j, q_j) = (p_k, q_k)$.

$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i \xrightarrow{a_{i+1} \cdots a_j | x_2} p_j \xrightarrow{a_{j+1} \cdots a_k | x_3} p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$

$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i \xrightarrow{a_{i+1} \cdots a_j | y_2} q_j \xrightarrow{a_{j+1} \cdots a_k | y_3} q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

$$x_1 x_4 = y_1 y_4 \implies \begin{array}{rcl} x_1 & = & y_1 z \\ z x_4 & = & y_4 \end{array}$$

$$x_1 x_2 x_4 = y_1 y_2 y_4 \implies y_1 z x_2 x_4 = y_1 y_2 z x_4 \implies z x_2 = y_2 z$$

$$x_1 x_3 x_4 = y_1 y_3 y_4 \implies z x_3 = y_3 z$$

$$x_1 x_2 x_3 x_4 = y_1 z x_2 x_3 x_4 = y_1 y_2 z x_3 x_4 = y_1 y_2 y_3 z x_4 = y_1 y_2 y_3 y_4$$

[5] Berstel, Transductions and Context-Free Languages, 1979

RATIONAL FUNCTIONS - f1NFT

Lemma Let $\mathcal{A}$ be a 1NFT with m states.
If $\mathcal{A}$ is functional on all words of length $\leq 2m^2$
Then $\mathcal{A}$ is functional.

Let $w = a_1 a_2 \dots a_n \in \mathsf{dom}(\mathcal{A})$ with $n > 2m^2$.
Assume $\mathcal{A}$ is functional on all words of length $< n$.
Let $0 \leq i < j < k \leq n$ with $(p_i, q_i) = (p_j, q_j) = (p_k, q_k)$.

$$p_0 \xrightarrow{a_1 \cdots a_i | x_1} p_i \xrightarrow{a_{i+1} \cdots a_j | x_2} p_j \xrightarrow{a_{j+1} \cdots a_k | x_3} p_k \xrightarrow{a_{k+1} \cdots a_n | x_4} p_n$$

$$q_0 \xrightarrow{a_1 \cdots a_i | y_1} q_i \xrightarrow{a_{i+1} \cdots a_j | y_2} q_j \xrightarrow{a_{j+1} \cdots a_k | y_3} q_k \xrightarrow{a_{k+1} \cdots a_n | y_4} q_n$$

$$x_1 x_4 = y_1 y_4 \implies \begin{array}{rcl} x_1 & = & y_1 z \\ z x_4 & = & y_4 \end{array}$$

$$x_1 x_2 x_4 = y_1 y_2 y_4 \implies y_1 z x_2 x_4 = y_1 y_2 z x_4 \implies z x_2 = y_2 z$$

$$x_1 x_3 x_4 = y_1 y_3 y_4 \implies z x_3 = y_3 z$$

$$x_1 x_2 x_3 x_4 = y_1 z x_2 x_3 x_4 = y_1 y_2 z x_3 x_4 = y_1 y_2 y_3 z x_4 = y_1 y_2 y_3 y_4$$

$\mathcal{A}$ is functional on w.

[5] Berstel, Transductions and Context-Free Languages, 1979

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

➡ Equivalence is decidable [1] in PTIME [2] for f1NFT

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

➡ Equivalence is decidable [1] in PTIME [2] for f1NFT

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two f1NFT.

Check that $\mathsf{dom}(\mathcal{A}_1) = \mathsf{dom}(\mathcal{A}_2)$.

Check that $\mathcal{A}_1 \uplus \mathcal{A}_2$ is functional.

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

➡ Equivalence is decidable [1] in PTIME [2] for f1NFT

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

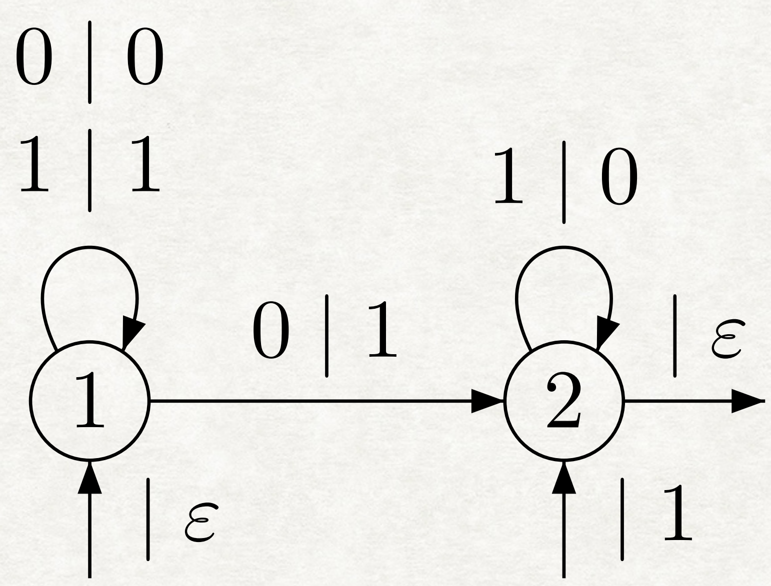
➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

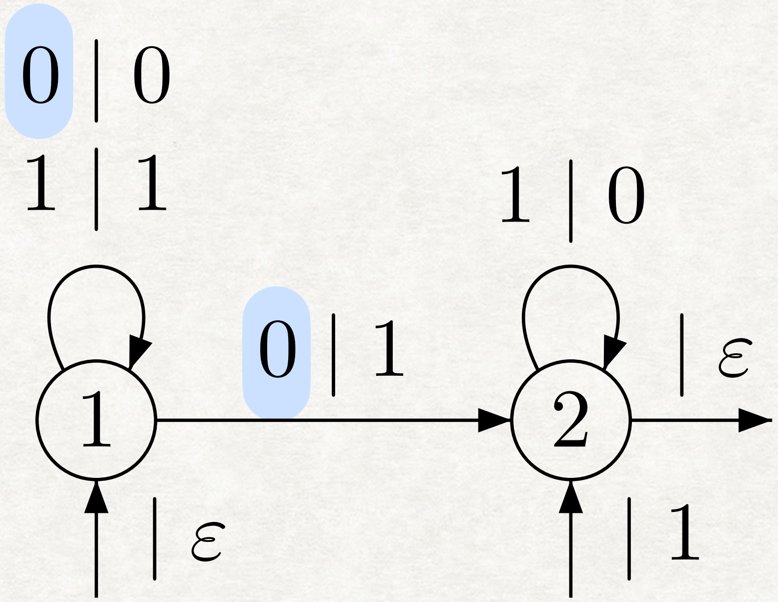


[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

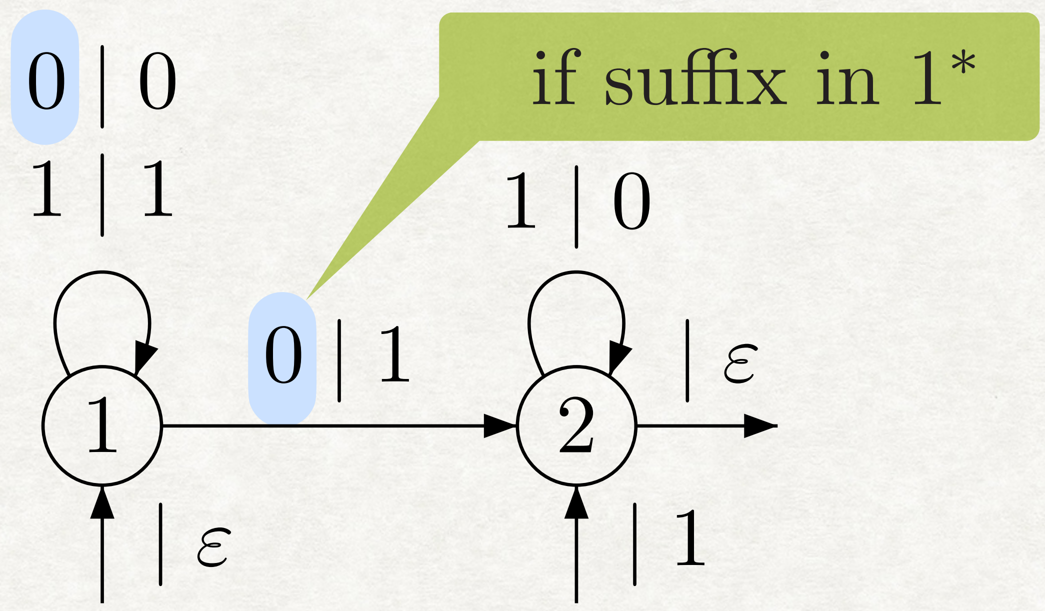


[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

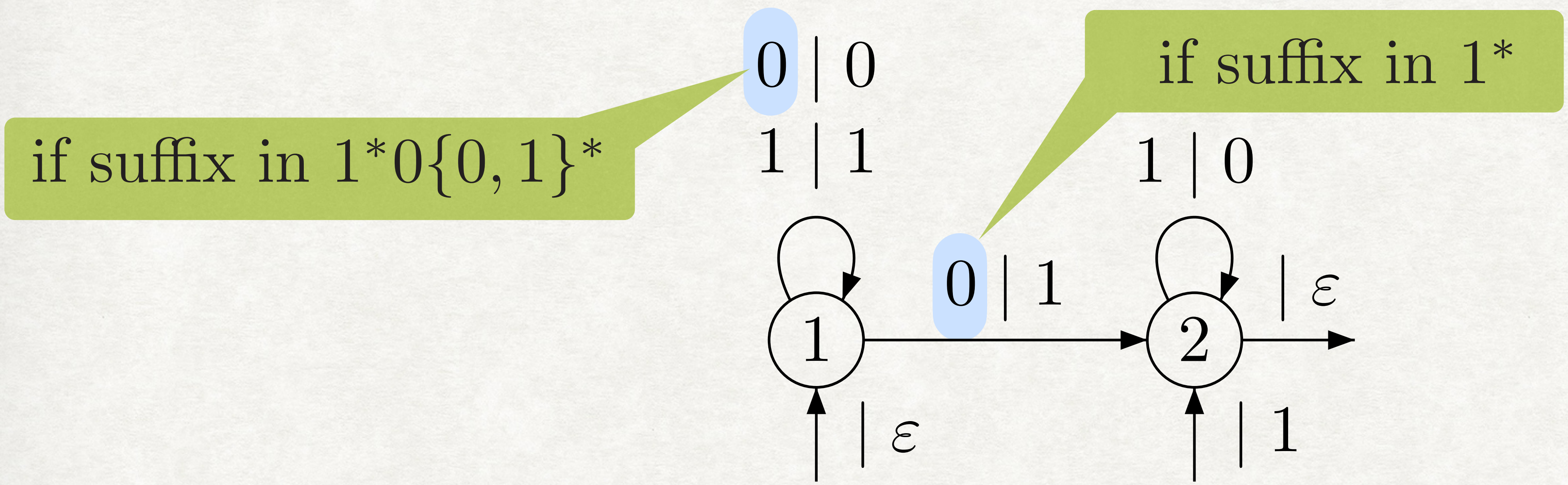


[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]



1011010011111

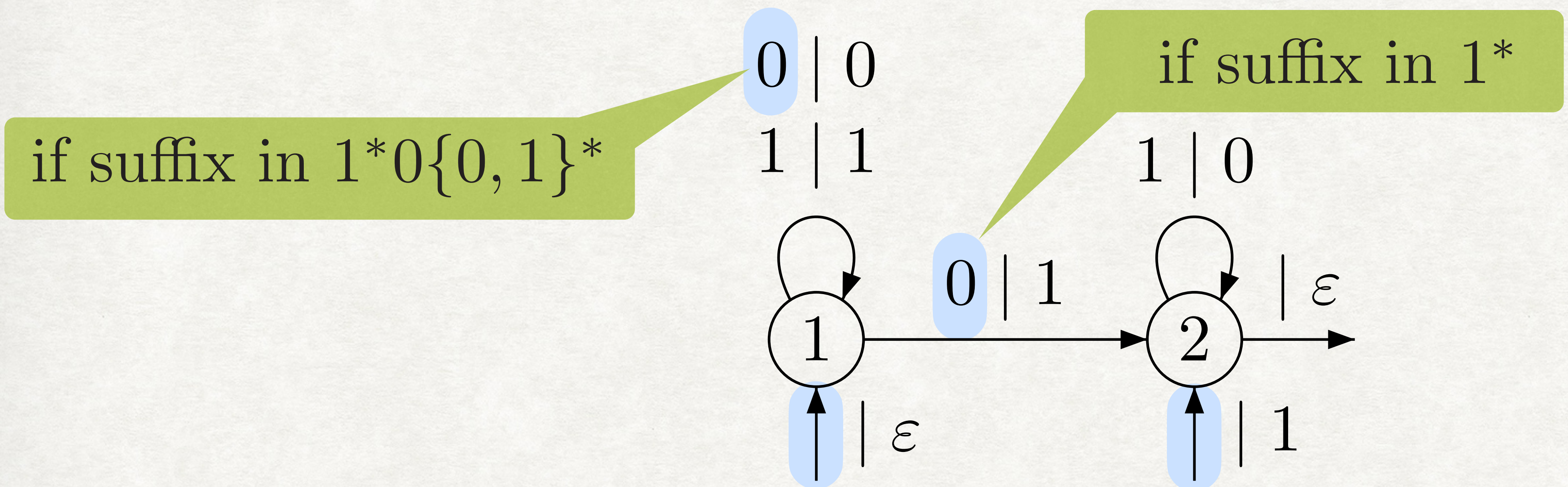
1011010100000

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]



1011010011111

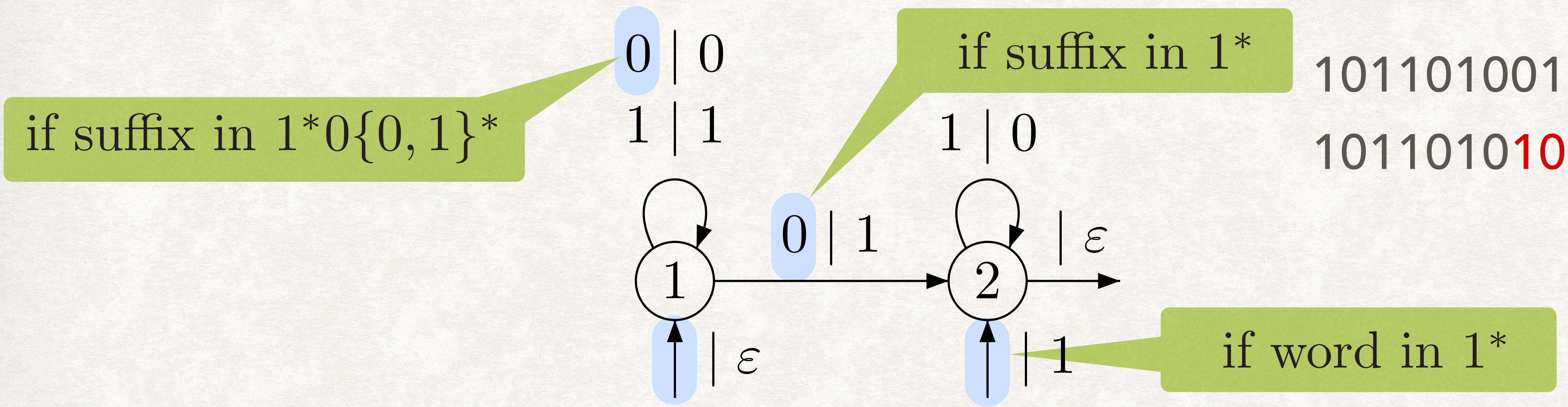
1011010100000

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]



1011010011111

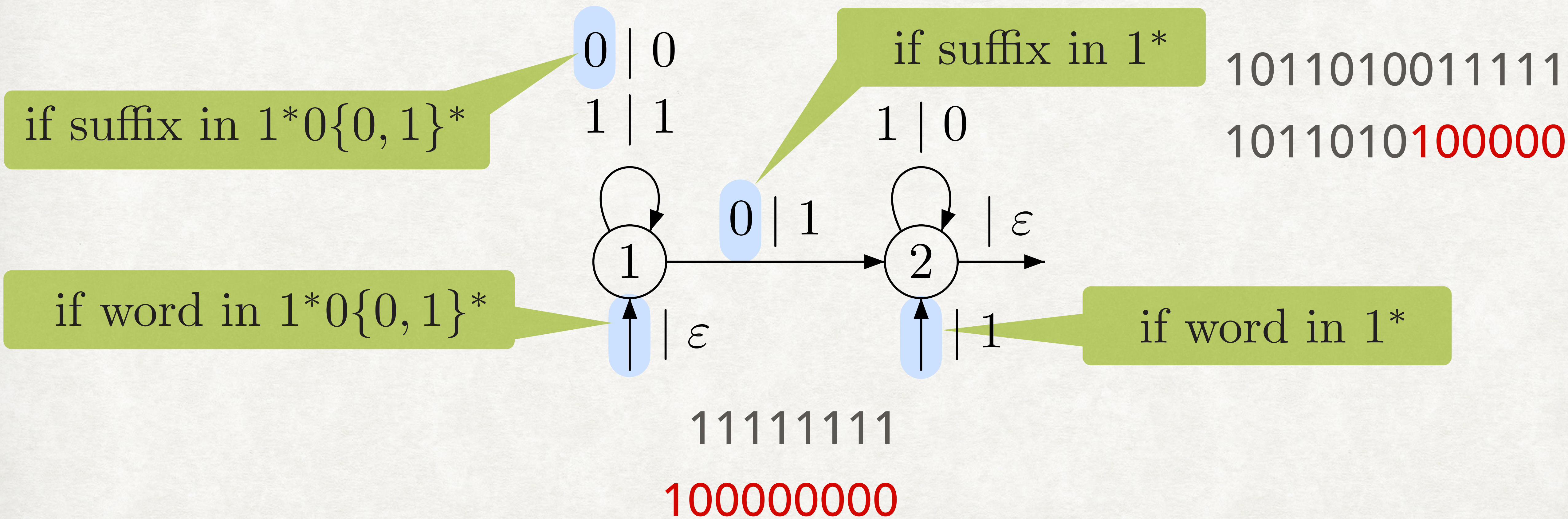
1011010100000

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]



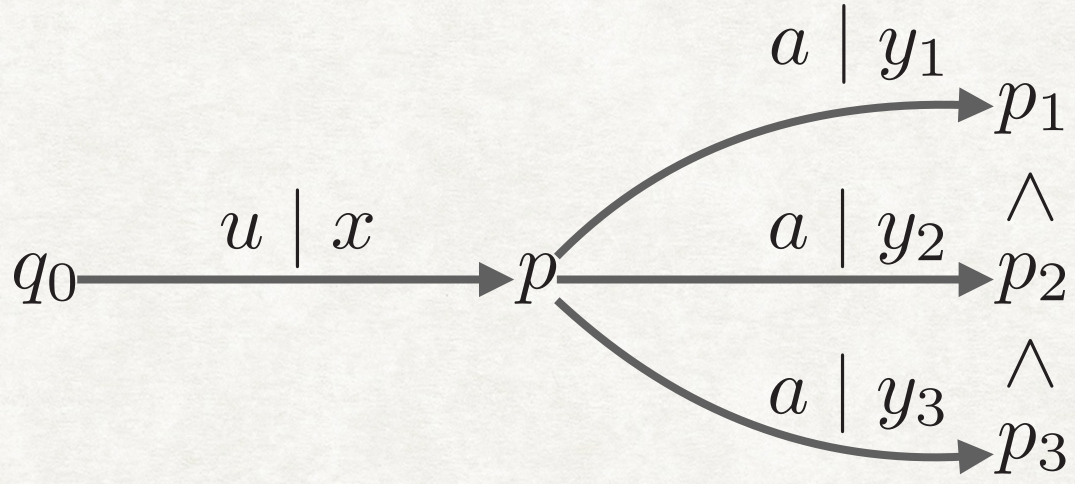
[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

Let $\mathcal{A}$ be a f1NFT with states in Q. Consider a total order $(Q, <)$.
Given $w = uav$, select the least accepting path wrt. lexicographic order.



$v \notin \mathsf{dom}(\mathcal{A}, p_1)$

$v \in \mathsf{dom}(\mathcal{A}, p_2)$

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

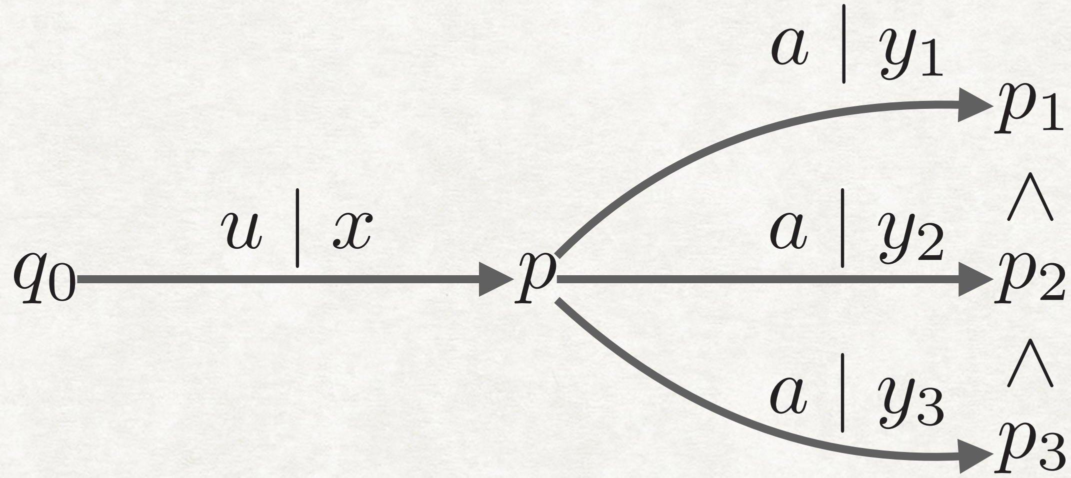
RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

Let $\mathcal{A}$ be a f1NFT with states in Q. Consider a total order $(Q, <)$.

Given $w = uav$, select the least accepting path wrt. lexicographic order.



$v \notin \mathsf{dom}(\mathcal{A}, p_1)$

$v \in \mathsf{dom}(\mathcal{A}, p_2)$

First, deterministic with look-ahead.

Then unambiguous without look-ahead.

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

➡ Equivalence is decidable [1] in PTIME [2] for f1NFT

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

➡ Equivalence is decidable [1] in PTIME [2] for f1NFT

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

➡ Decidable in PTIME if a rational function is sequential [3,4]

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

[4] Choffrut, Une caractérisation des fonctions séquentielles … en tant que relations rationnelles, 1977

RATIONAL FUNCTIONS - f1NFT

- Non-deterministic left to right parsing of the input
- Produce output along the way
- 1UFT: Unambiguous left to right parsing of the input

➡ Rational transducers (1NFT) need not be functional

➡ Functionality is decidable [1] in PTIME [2] for rational transducers

➡ Equivalence is decidable [1] in PTIME [2] for f1NFT

➡ f1NFT = 1DFT with look-ahead = 1UFT [3]

➡ Decidable in PTIME if a rational function is sequential [3,4]

➡ Rational functions are closed under composition

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

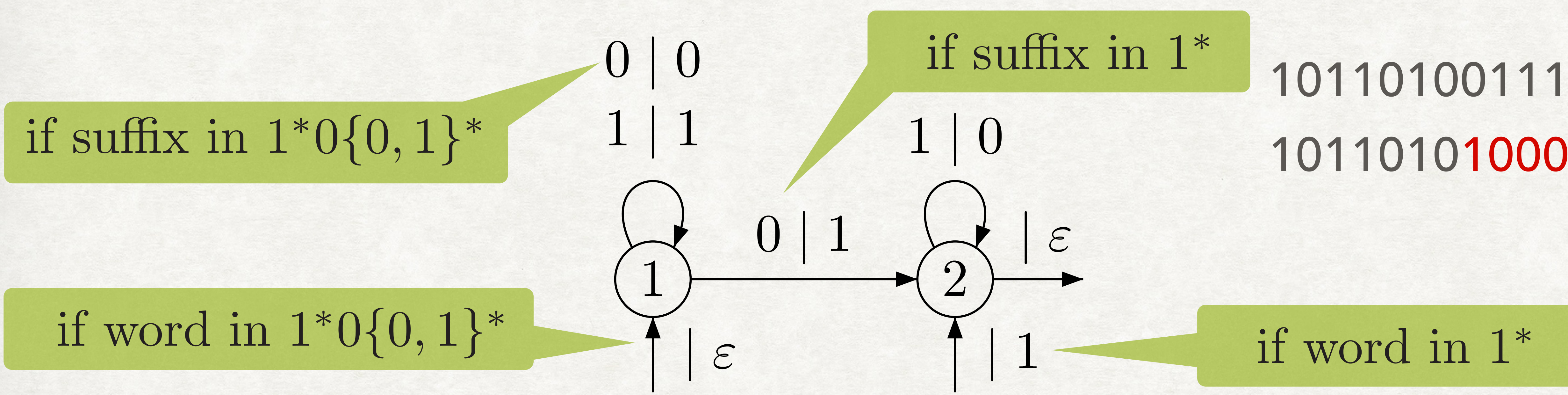
[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

[4] Choffrut, Une caractérisation des fonctions séquentielles … en tant que relations rationnelles, 1977

SUMMARY

- Operational models (transducers)
  - 1DFT = sequential functions
  - f1NFT = rational functions
  - 2DFT = regular functions
  - Transducers with registers
- Modular descriptions
  - Rational expressions
  - Composition

REGULAR FUNCTIONS - 2DFT



10110100011111

10110100100000

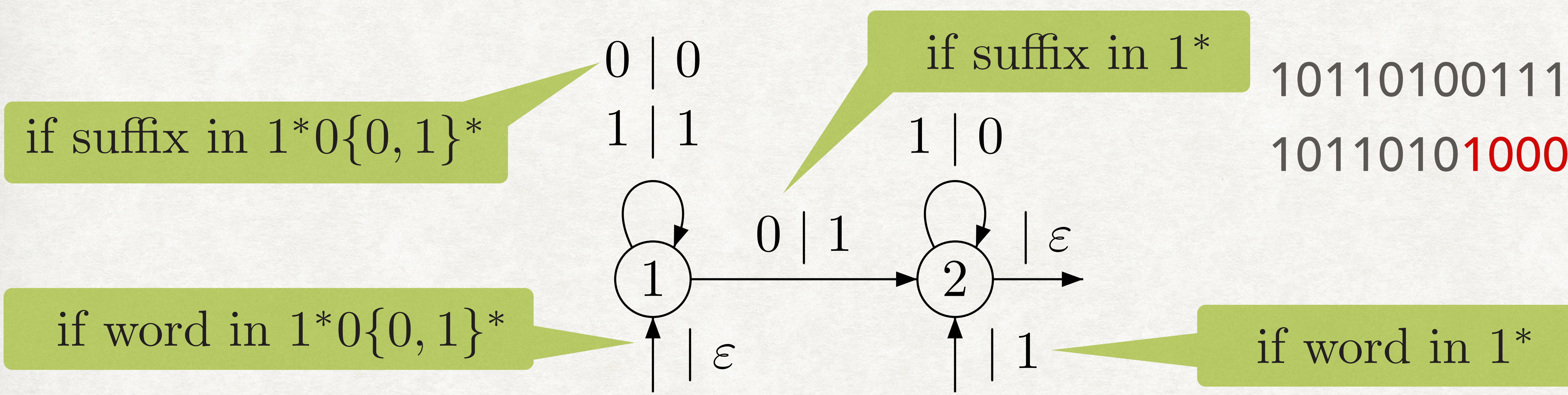
11111111

100000000

REGULAR FUNCTIONS - 2DFT

Increment of a number with lsb on the right

We have a 1DFT with look-ahead: locate the last 0



1011010011111

1011010100000

11111111

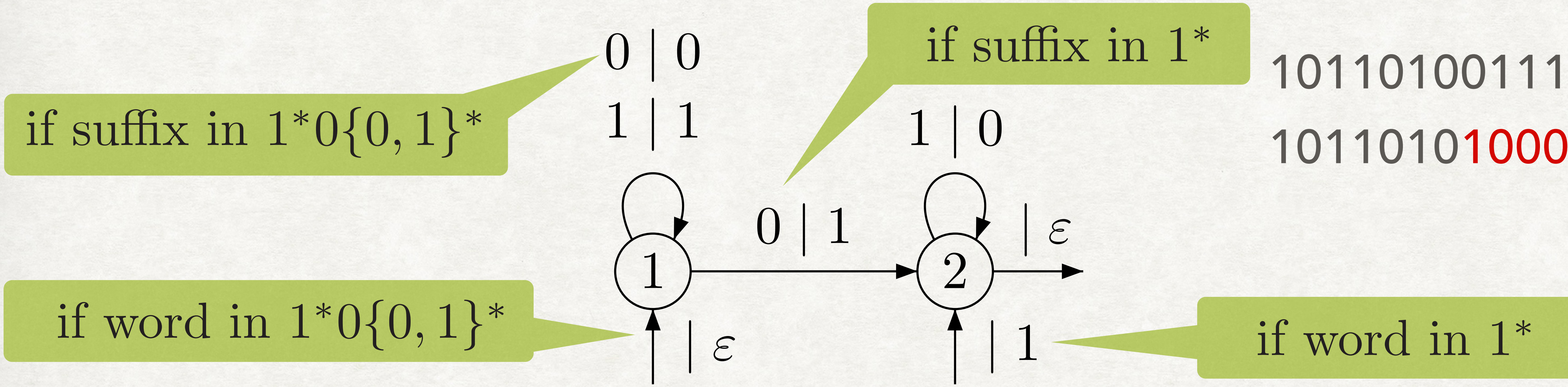
100000000

REGULAR FUNCTIONS - 2DFT

Increment of a number with lsb on the right

We have a 1DFT with look-ahead: locate the last 0

Implement the look-ahead with deterministic 2-way parsing



1011010011111

1011010100000

11111111

100000000

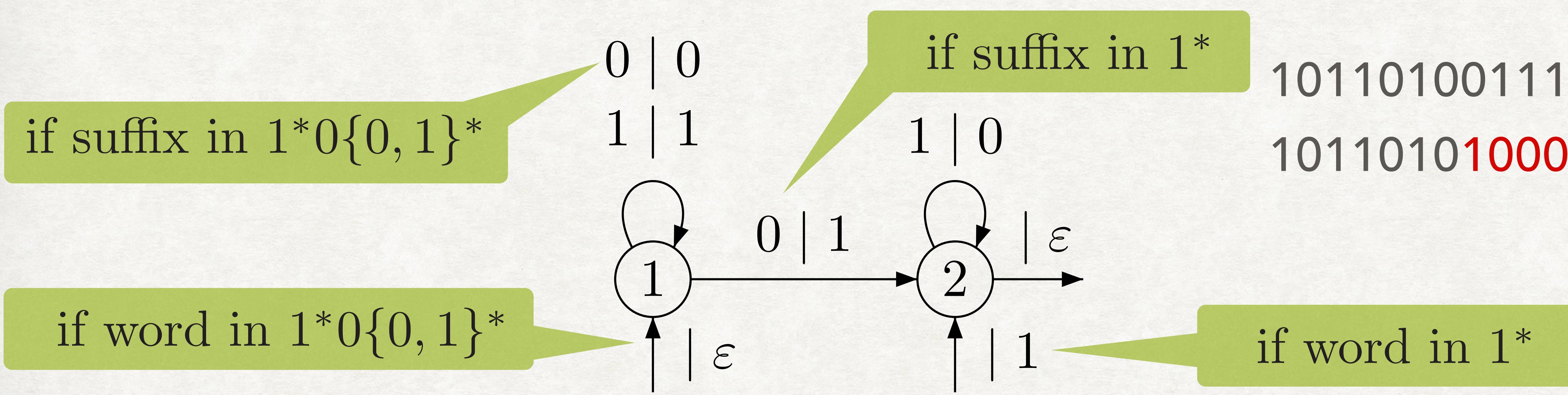
REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

Increment of a number with lsb on the right

We have a 1DFT with look-ahead: locate the last 0

Implement the look-ahead with deterministic 2-way parsing



1011010011111

1011010100000

11111111

100000000

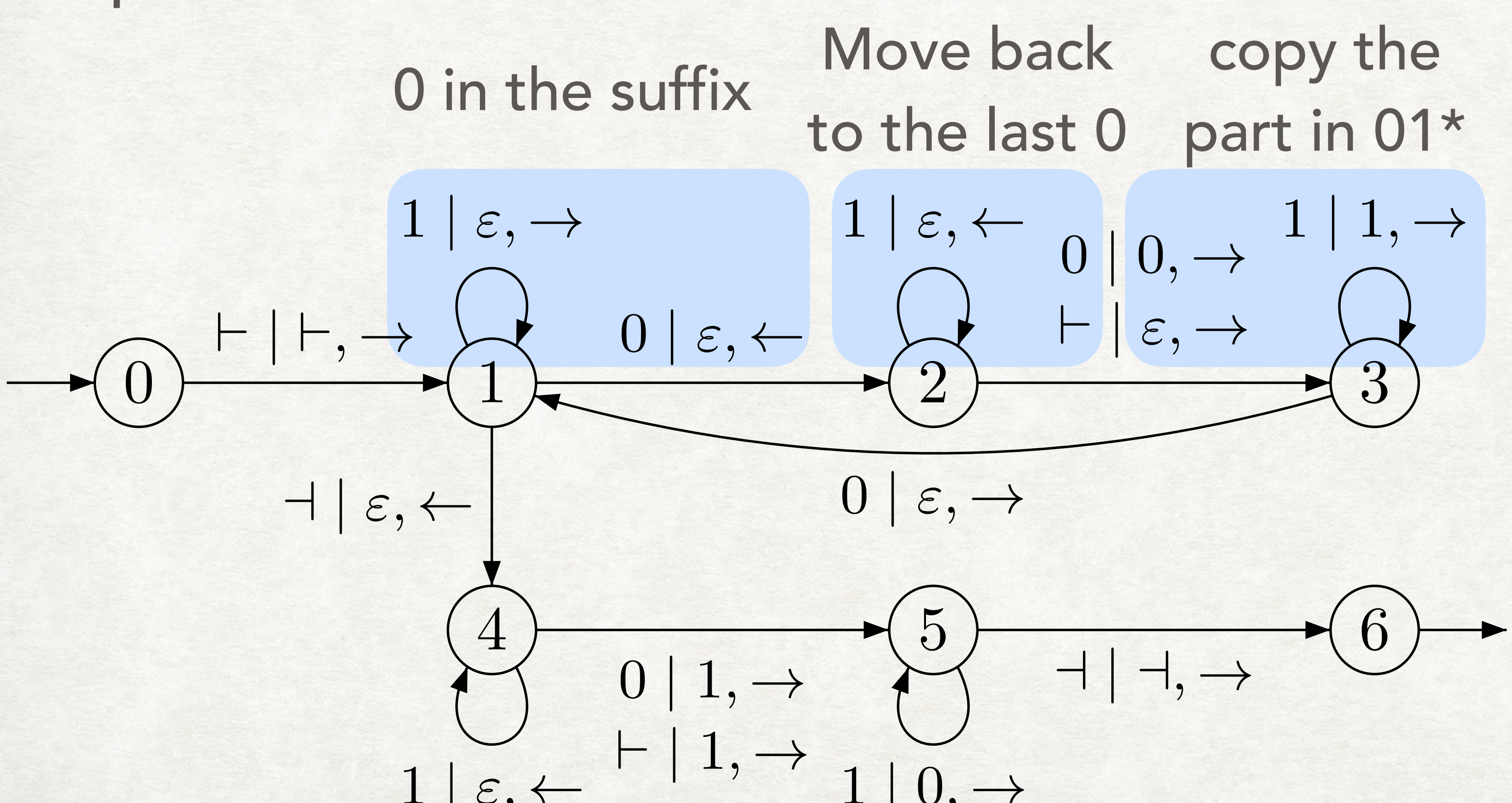
REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

Increment of a number with lsb on the right

We have a 1DFT with look-ahead: locate the last 0

Implement the look-ahead with deterministic 2-way parsing



1011010011111
1011010100000

11111111
100000000

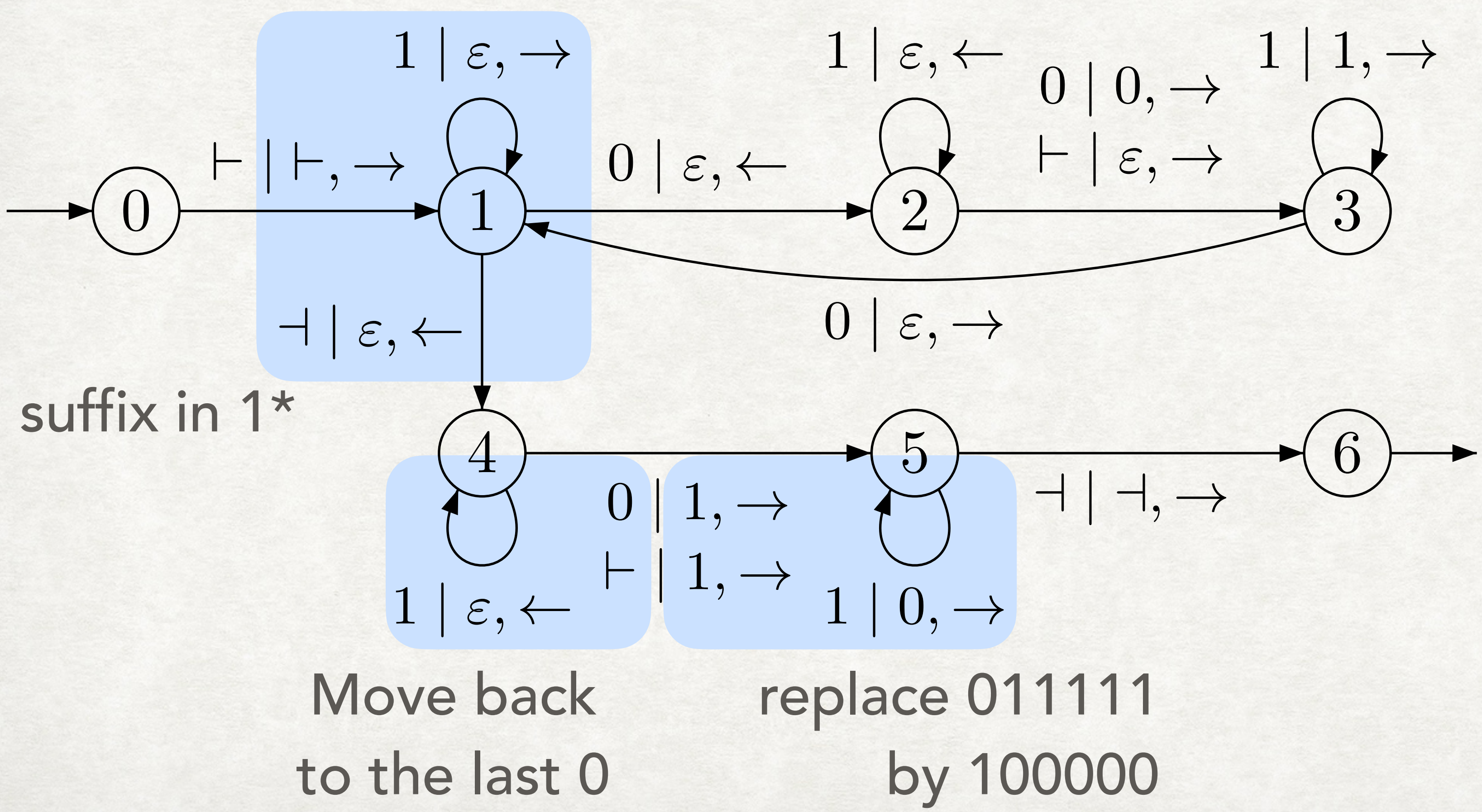
REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

Increment of a number with lsb on the right with a 2DFT

We have a 1DFT with look-ahead: locate the last 0

Implement the look-ahead with 2-way parsing



1011010011111

1011010100000

11111111

100000000

REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

VERY ROBUST CLASS OF TRANSFORMATIONS

REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

VERY ROBUST CLASS OF TRANSFORMATIONS

➡ 2DFT = reversible 2DFT [7]

[7] Dartois, Fournier, Jecker, Lhote, On reversible transducers, 2017

REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

VERY ROBUST CLASS OF TRANSFORMATIONS

➡ 2DFT = reversible 2DFT [7]

➡ Regular functions are closed under composition [6]

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

[7] Dartois, Fournier, Jecker, Lhote, On reversible transducers, 2017

REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

VERY ROBUST CLASS OF TRANSFORMATIONS

➡ 2DFT = reversible 2DFT [7]

➡ Regular functions are closed under composition [6]

➡ f2NFT = 2DFT with look-ahead & look-behind = 2UFT = 2DFT [8]

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

[7] Dartois, Fournier, Jecker, Lhote, On reversible transducers, 2017

[8] Engelfriet & Hoogeboom, MSO definable string transductions and two-way finite-state transducers, 2001

REGULAR FUNCTIONS - 2DFT

- Deterministic 2-way parsing of the input
- Produce output along the way

VERY ROBUST CLASS OF TRANSFORMATIONS

➡ 2DFT = reversible 2DFT [7]

➡ Regular functions are closed under composition [6]

➡ f2NFT = 2DFT with look-ahead & look-behind = 2UFT = 2DFT [8]

➡ Equivalence is decidable for 2DFT [9]

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

[7] Dartois, Fournier, Jecker, Lhote, On reversible transducers, 2017

[8] Engelfriet & Hoogeboom, MSO definable string transductions and two-way finite-state transducers, 2001

[9] Culik & Karhumäki, The equivalence of finite valued transducers (on HDT0L languages) is decidable, 1986

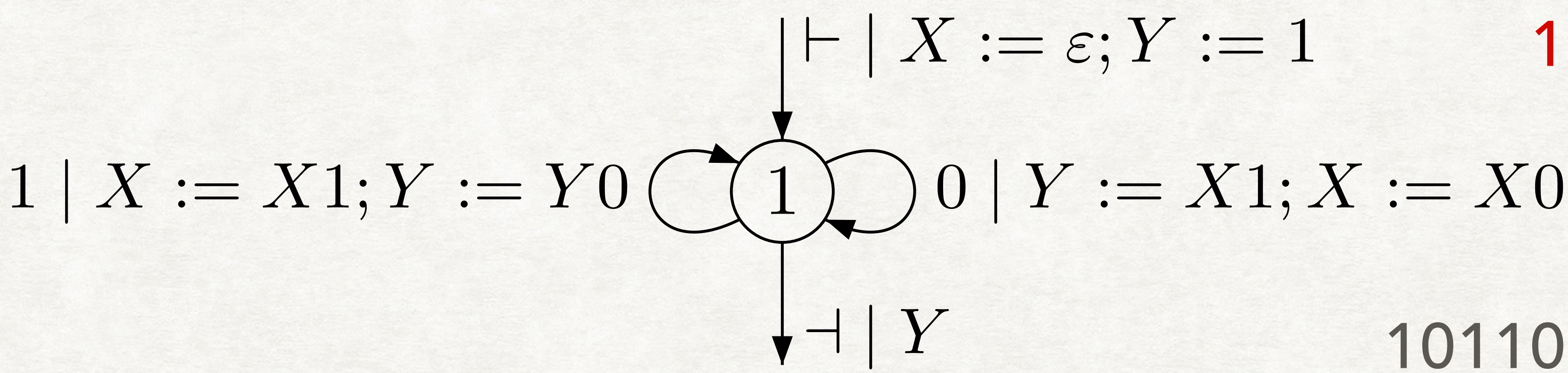
SUMMARY

- Operational models (transducers)
  - 1DFT = sequential functions
  - f1NFT = rational functions
  - 2DFT = regular functions
  - Transducers with registers
- Modular descriptions
  - Rational expressions
  - Composition

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers

11111111
100000000

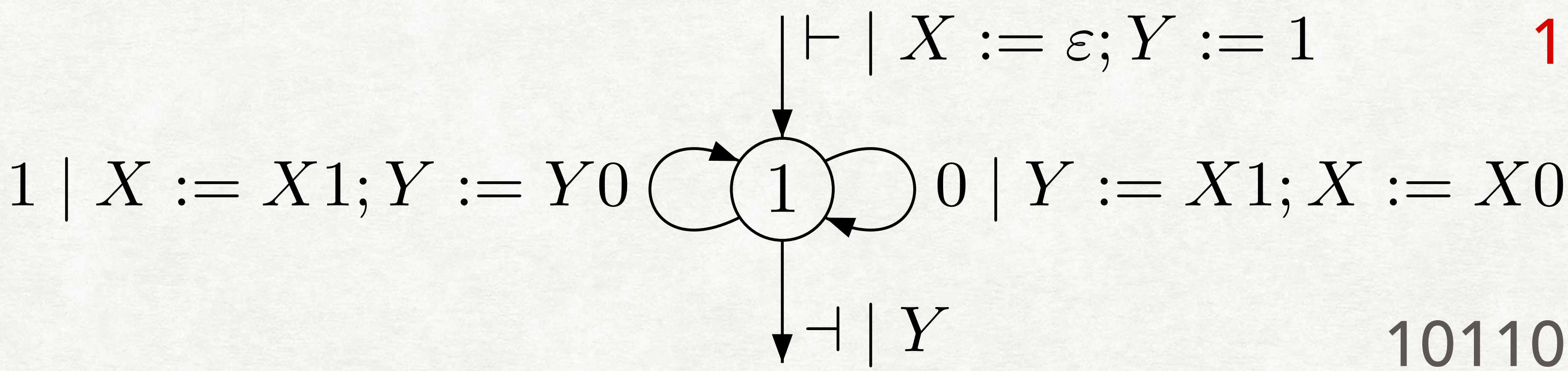


1011010011111
1011010100000

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers



11111111
100000000

1011010011111
1011010100000

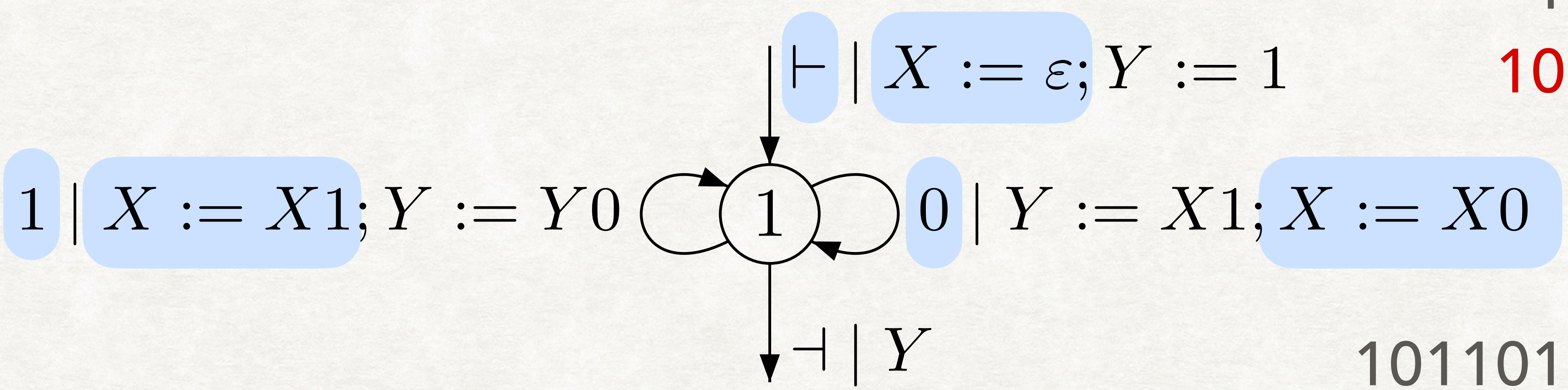
- X keeps a copy of the input binary number

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers

11111111
100000000



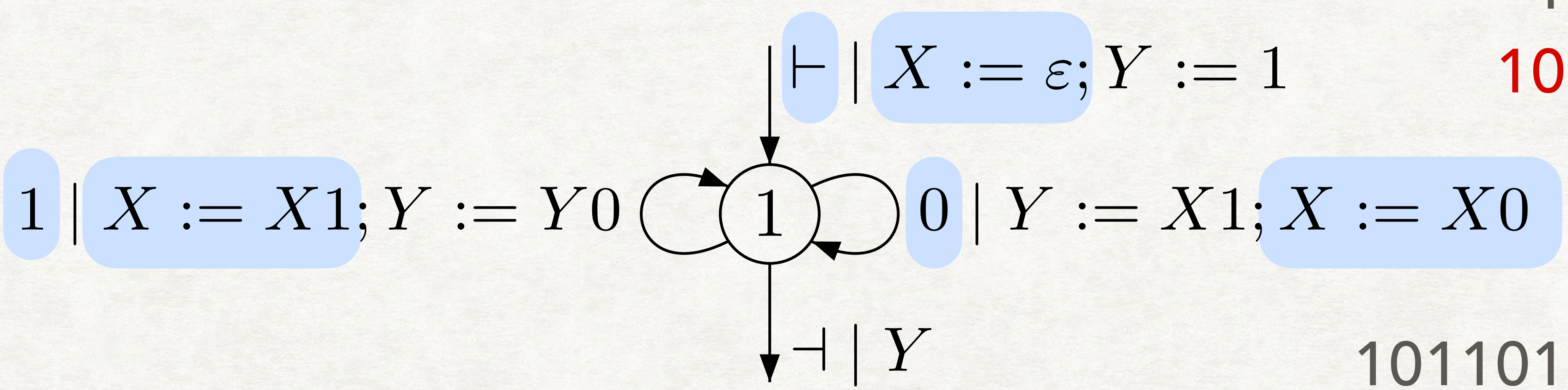
1011010011111
1011010100000

- X keeps a copy of the input binary number

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers



11111111
100000000

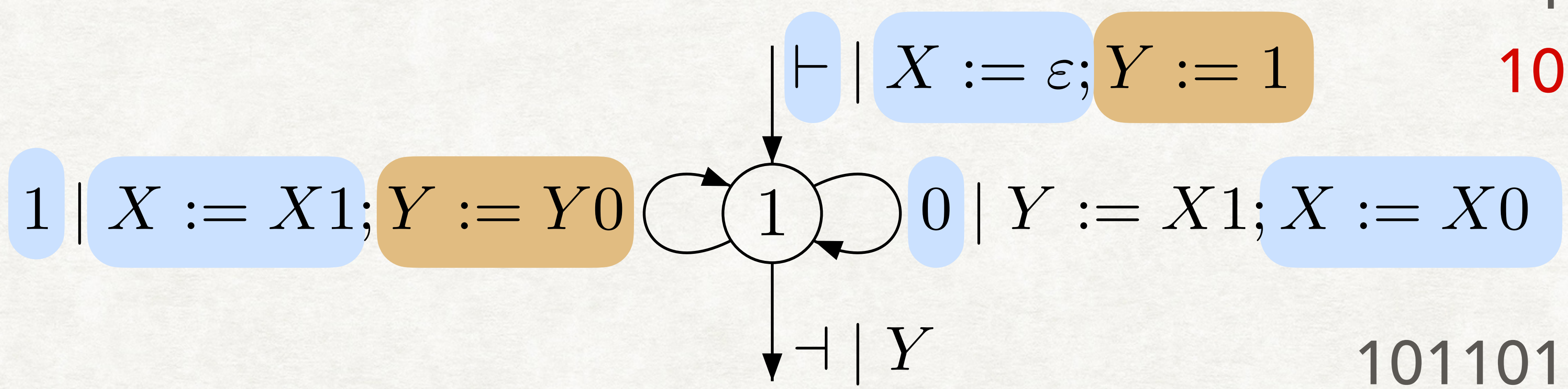
1011010011111
1011010100000

- X keeps a copy of the input binary number
- Y contains its increment

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers



11111111
100000000

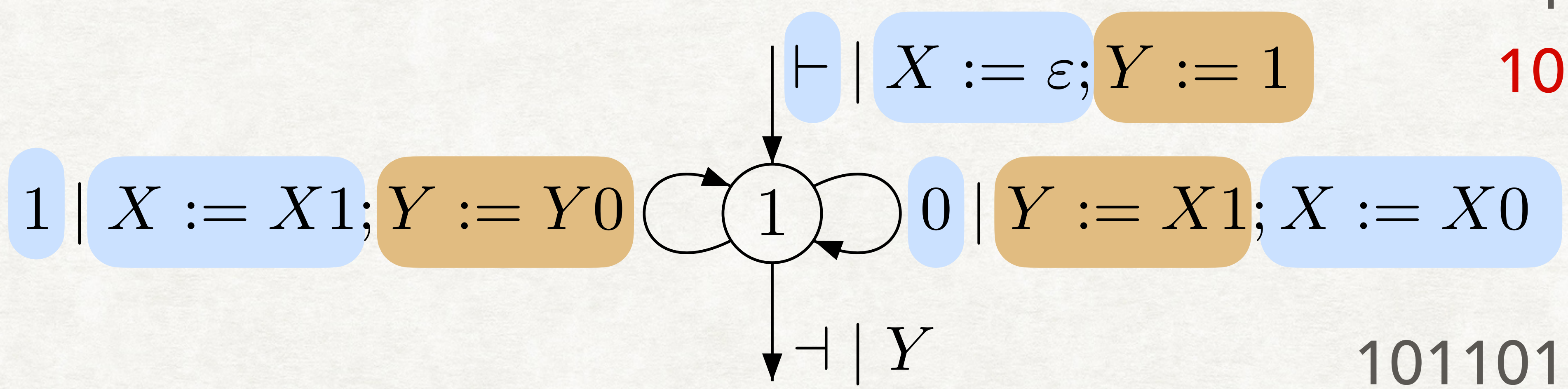
1011010011111
1011010100000

- X keeps a copy of the input binary number
- Y contains its increment

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers



11111111
100000000

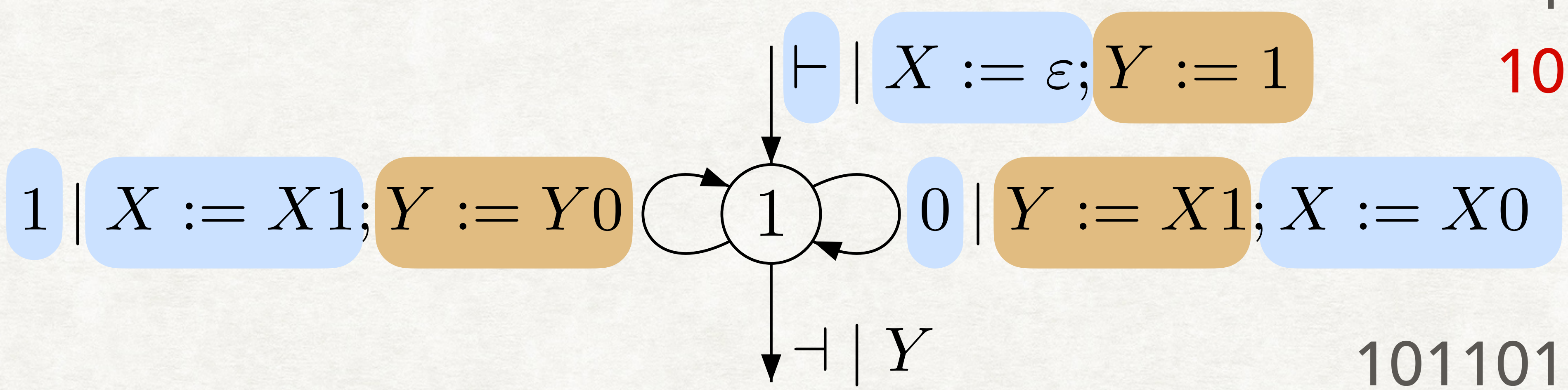
1011010011111
1011010100000

- X keeps a copy of the input binary number
- Y contains its increment

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

SIMPLE PROGRAMS: REGISTERS

- Deterministic parsing of the input
- Produce output in registers



11111111
100000000

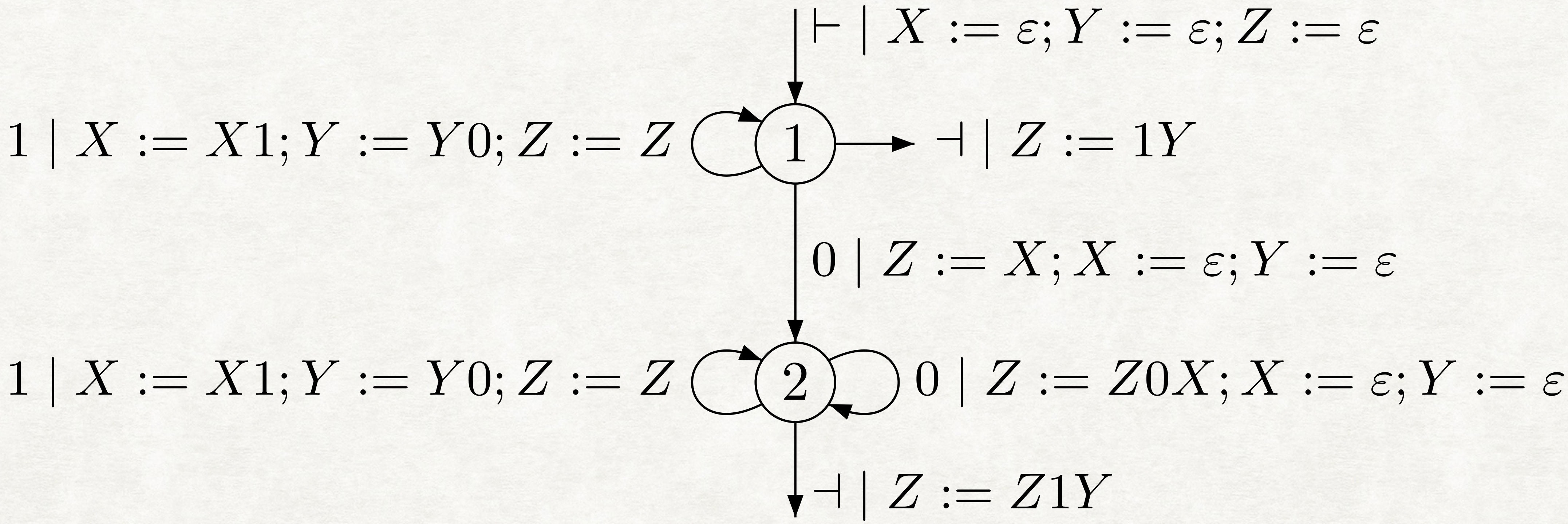
1011010011111
1011010100000

- X keeps a copy of the input binary number
- Y contains its increment

- Register updates: X := u | X := Xu | X := Yu (with u finite string)
- 1-way or 2-way
- Simple programs may be composed
- Simple programs = 2DFT

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

STREAMING STRING TRANSDUCERS



11111111

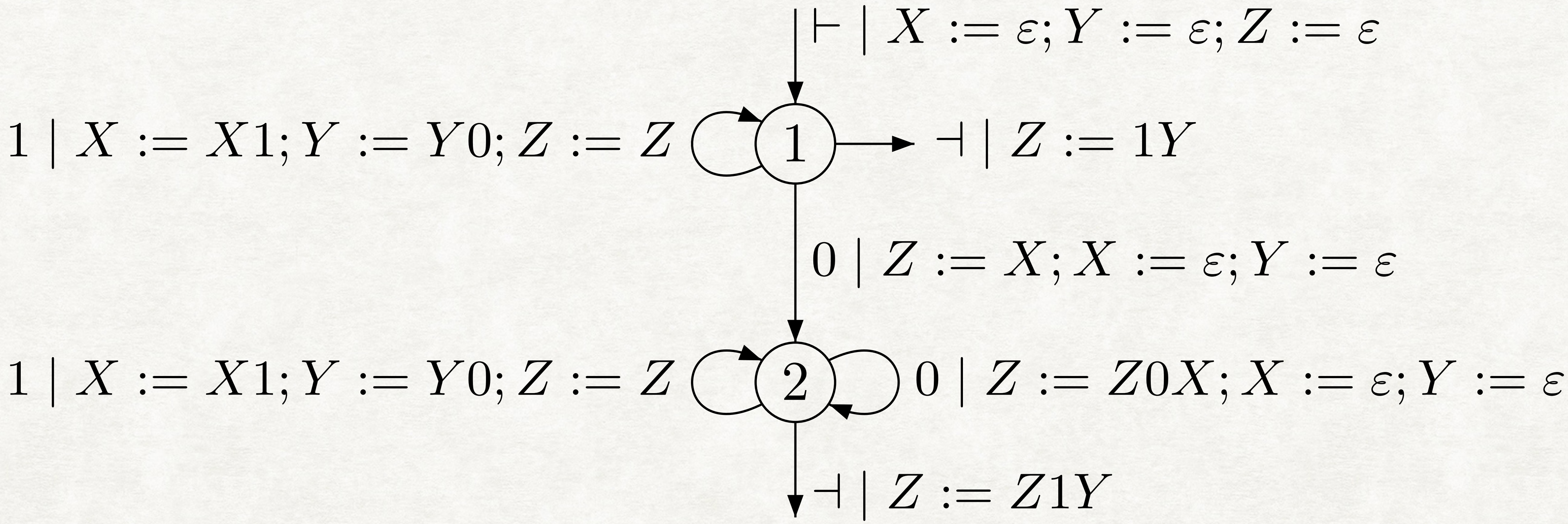
100000000

1011010011111

1011010100000

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



11111111

100000000

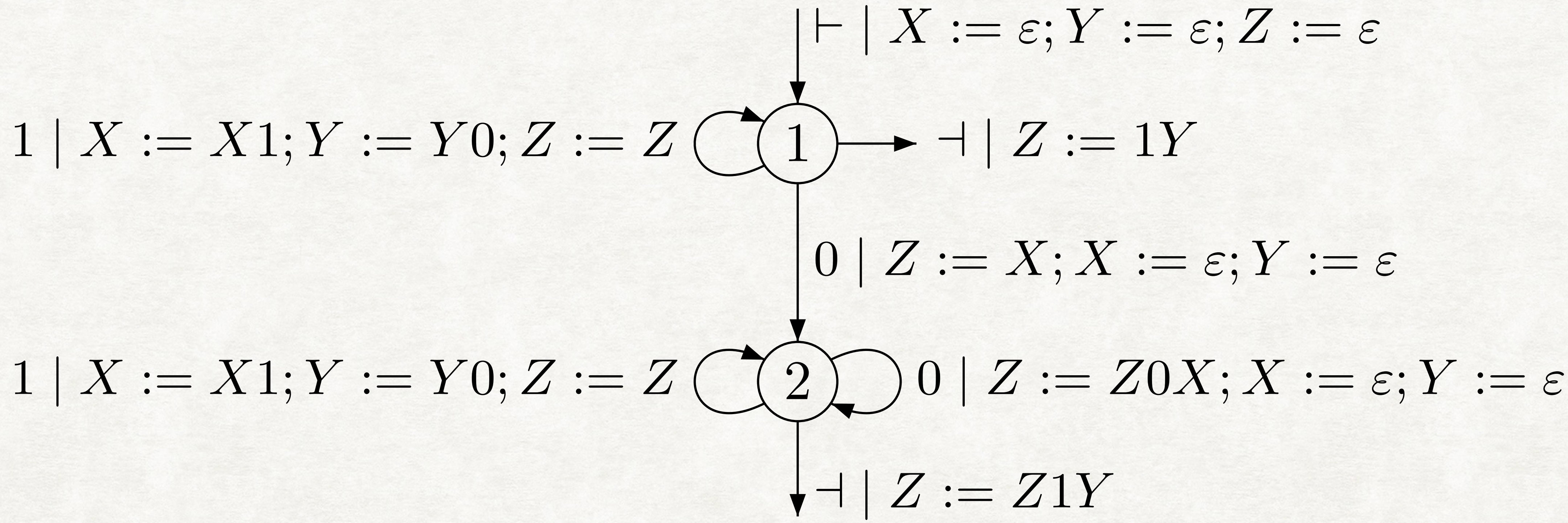
1011010011111

1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



- X keeps a copy of the last sequence of 1's

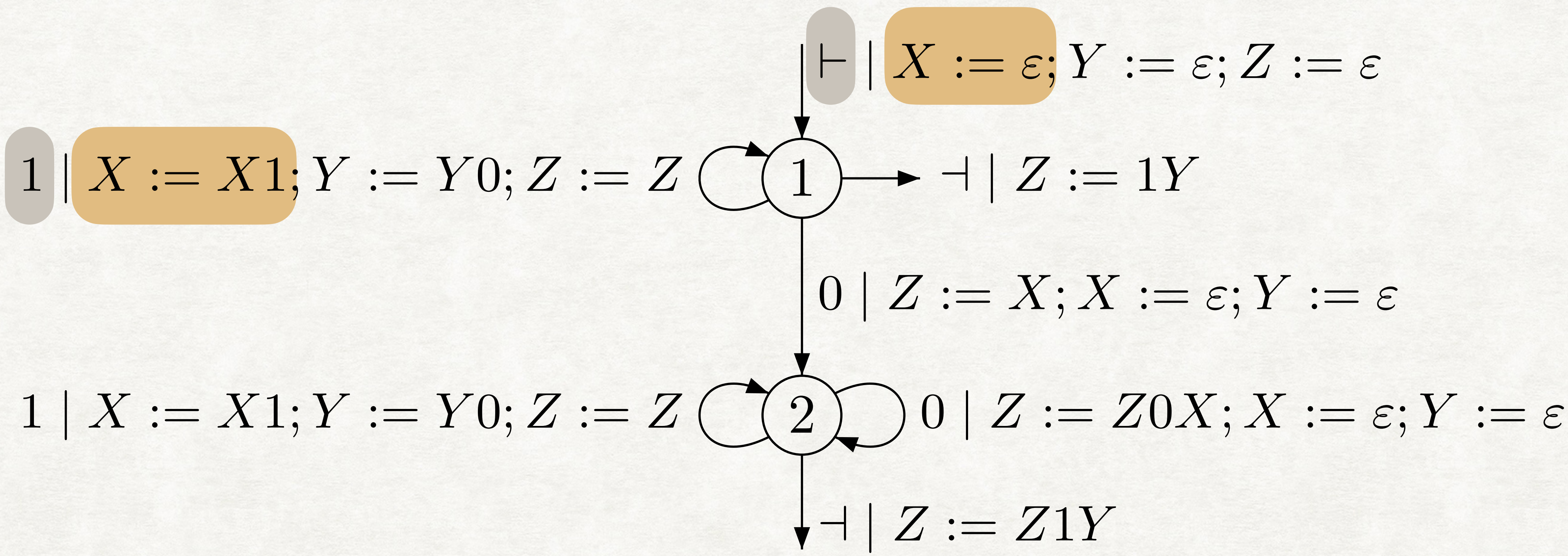
11111111
100000000

1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



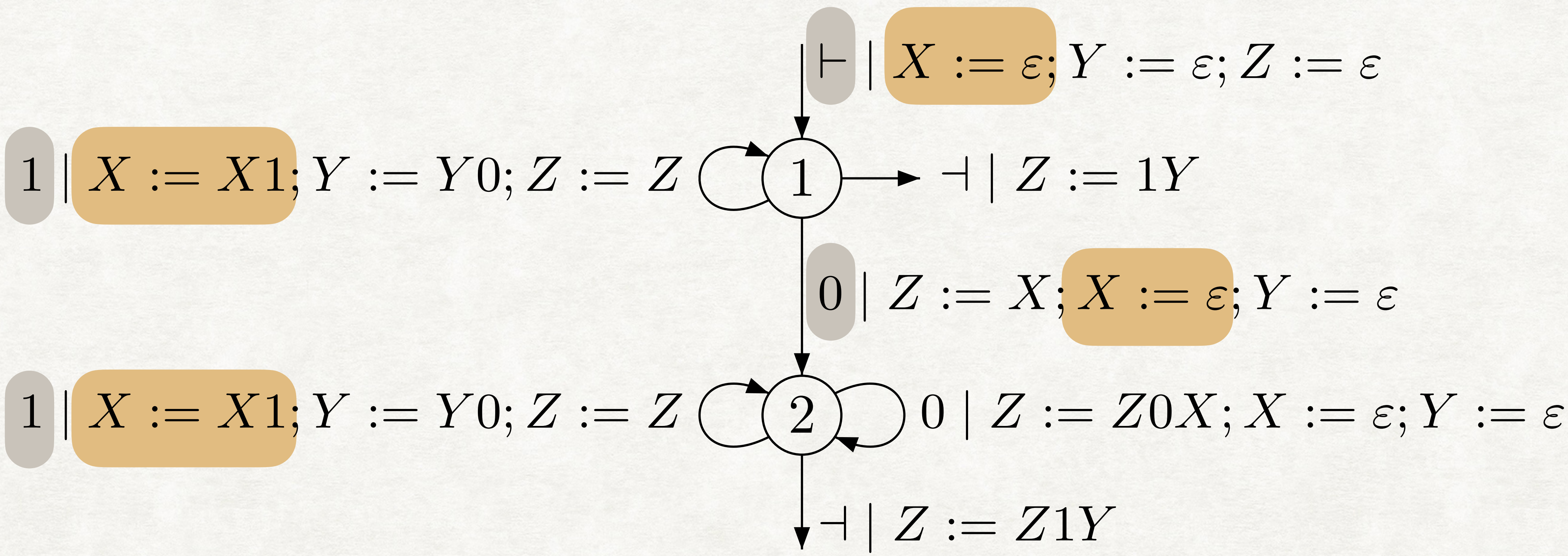
- X keeps a copy of the last sequence of 1's

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



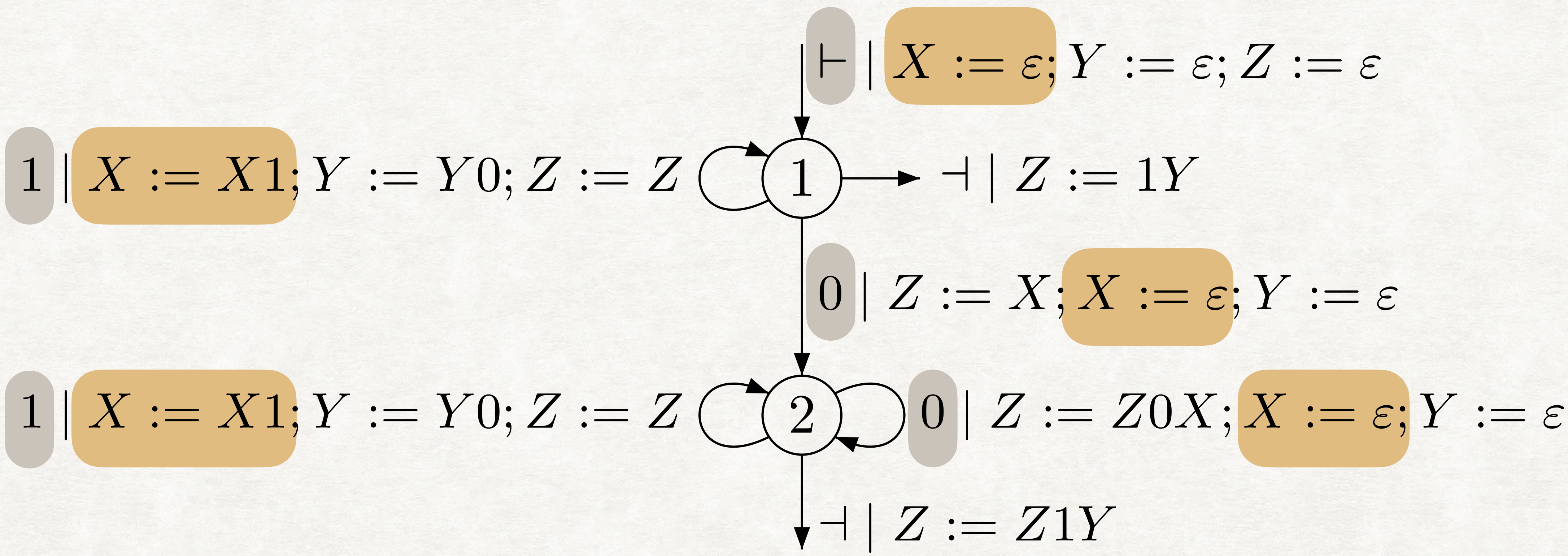
- X keeps a copy of the last sequence of 1's

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



- X keeps a copy of the last sequence of 1's

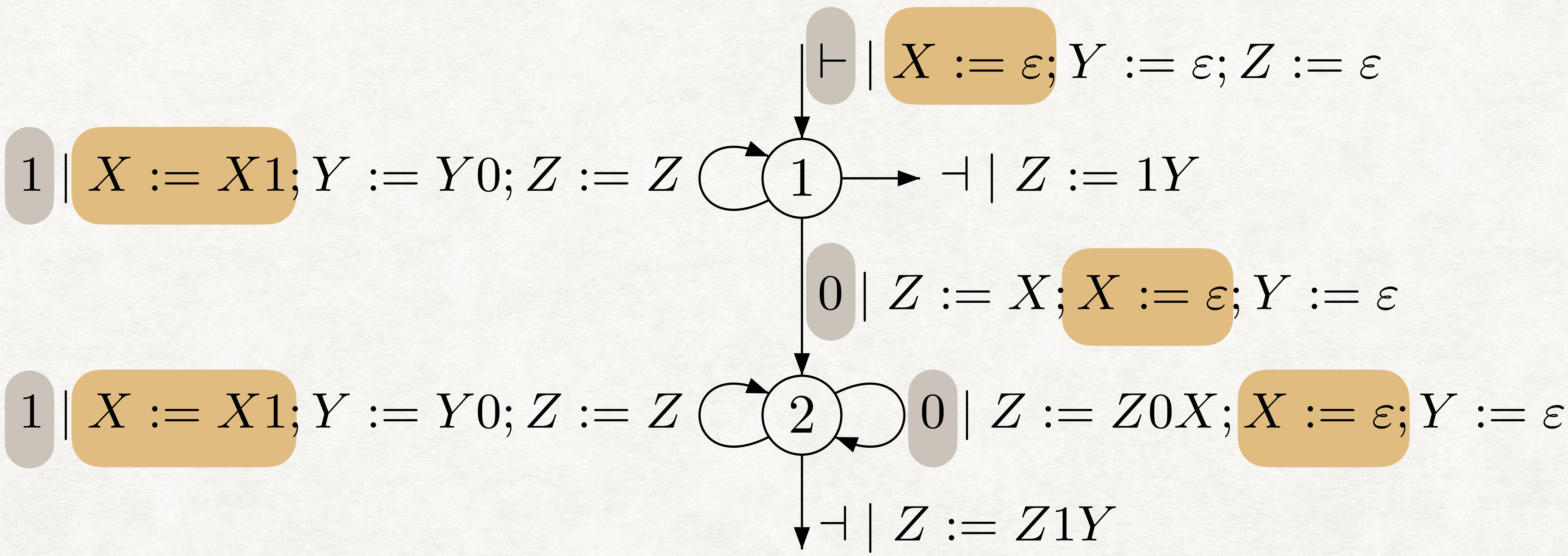
11111111
100000000

1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



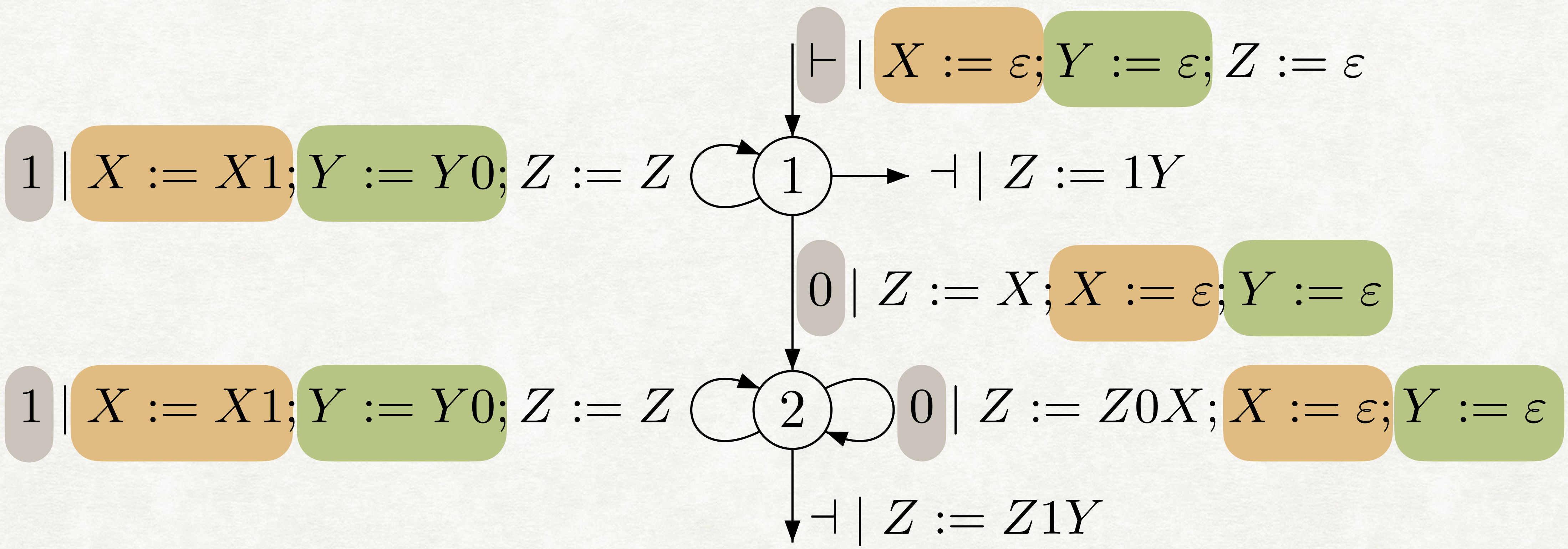
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



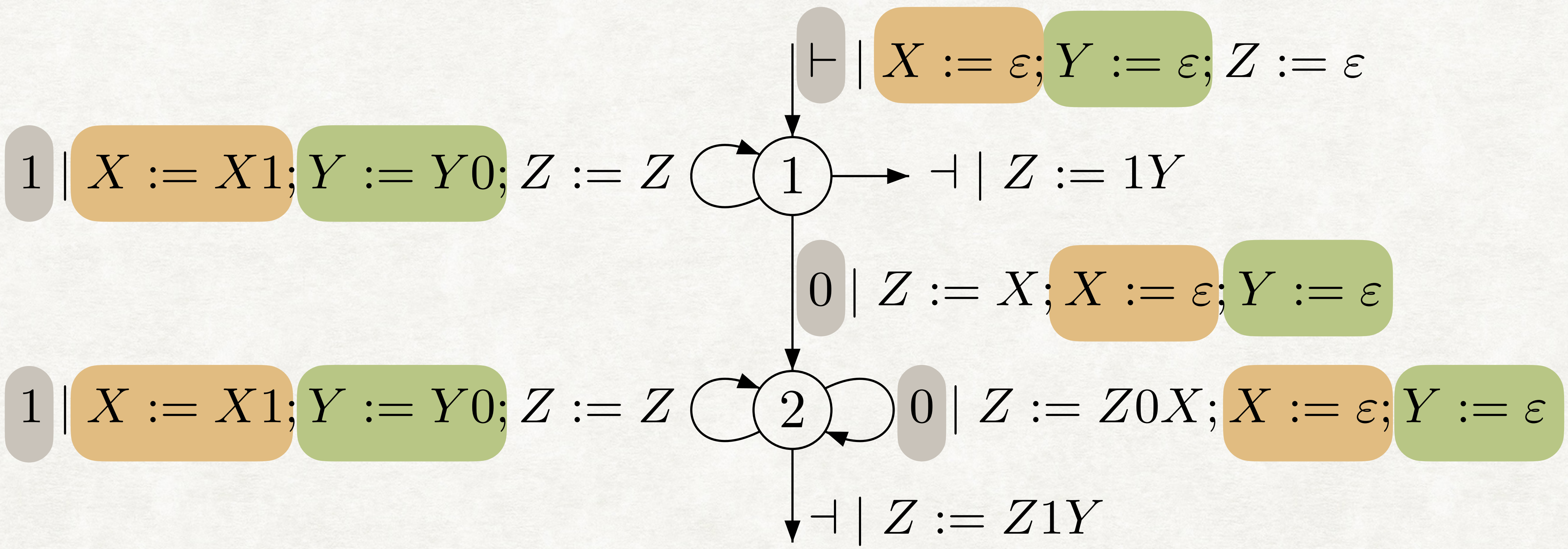
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



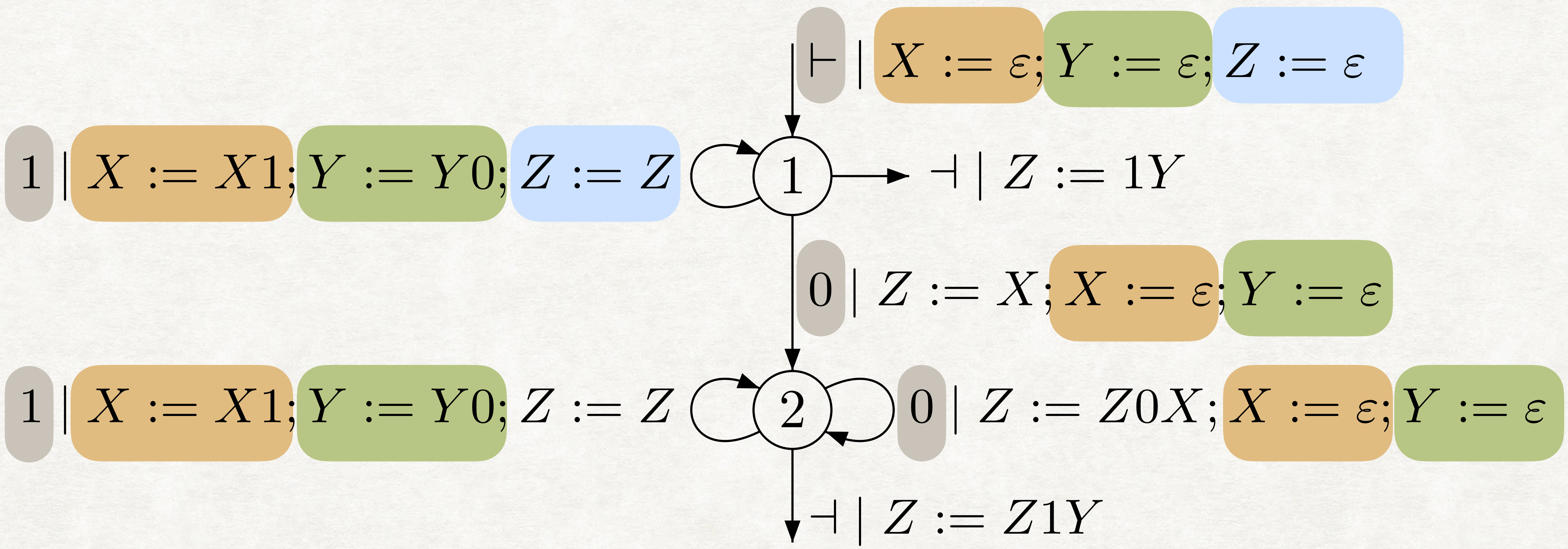
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



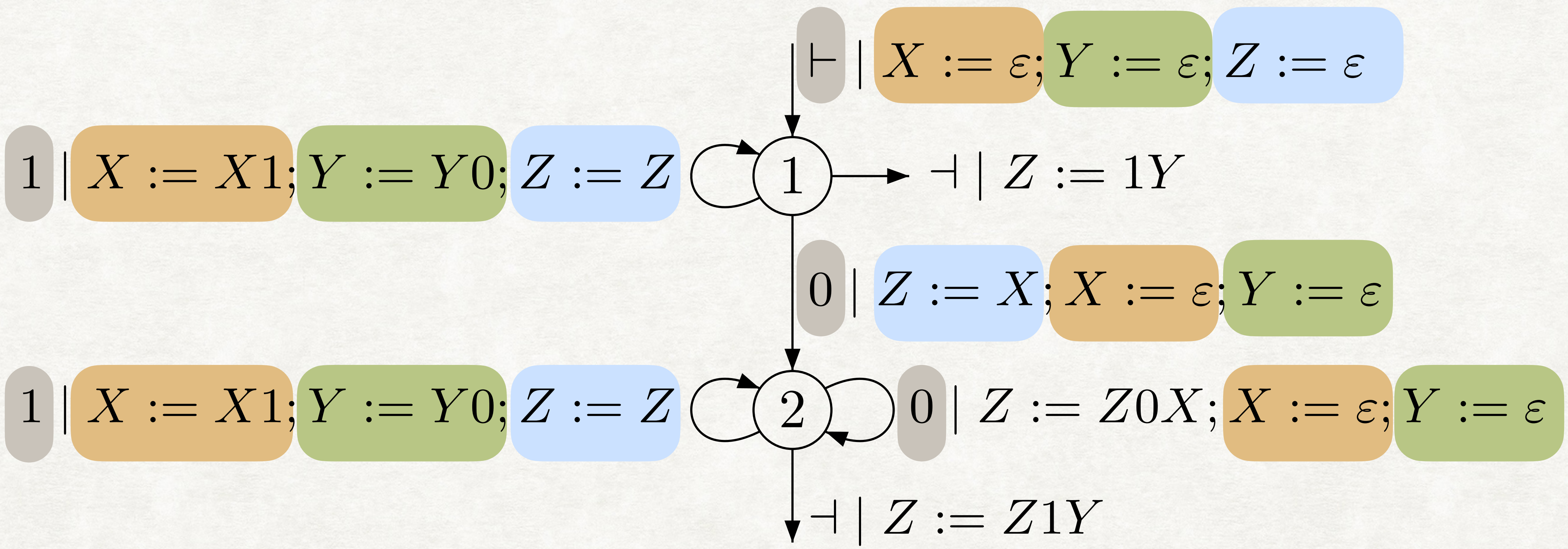
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



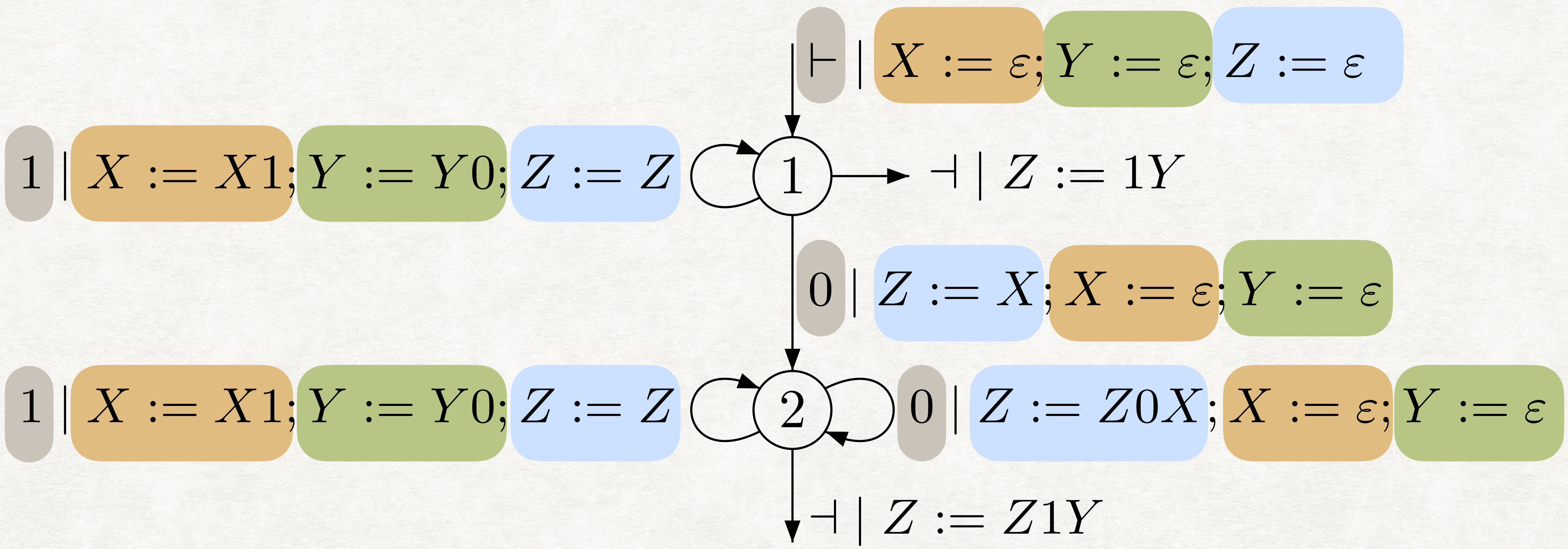
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



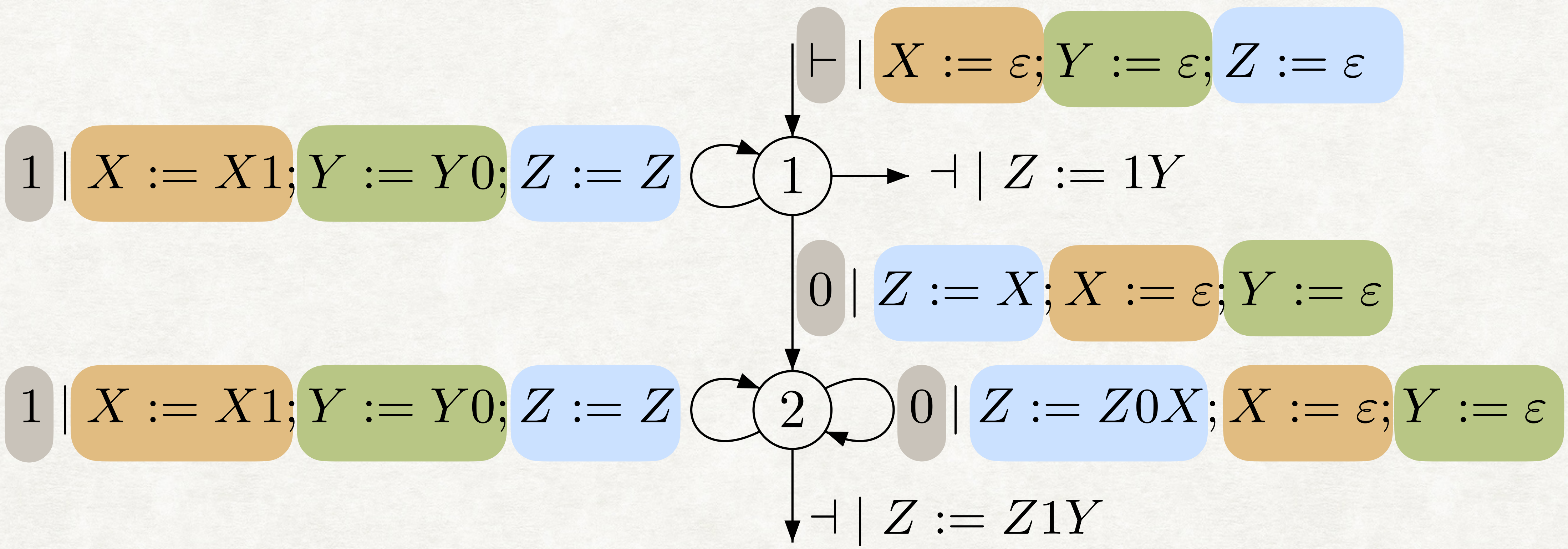
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



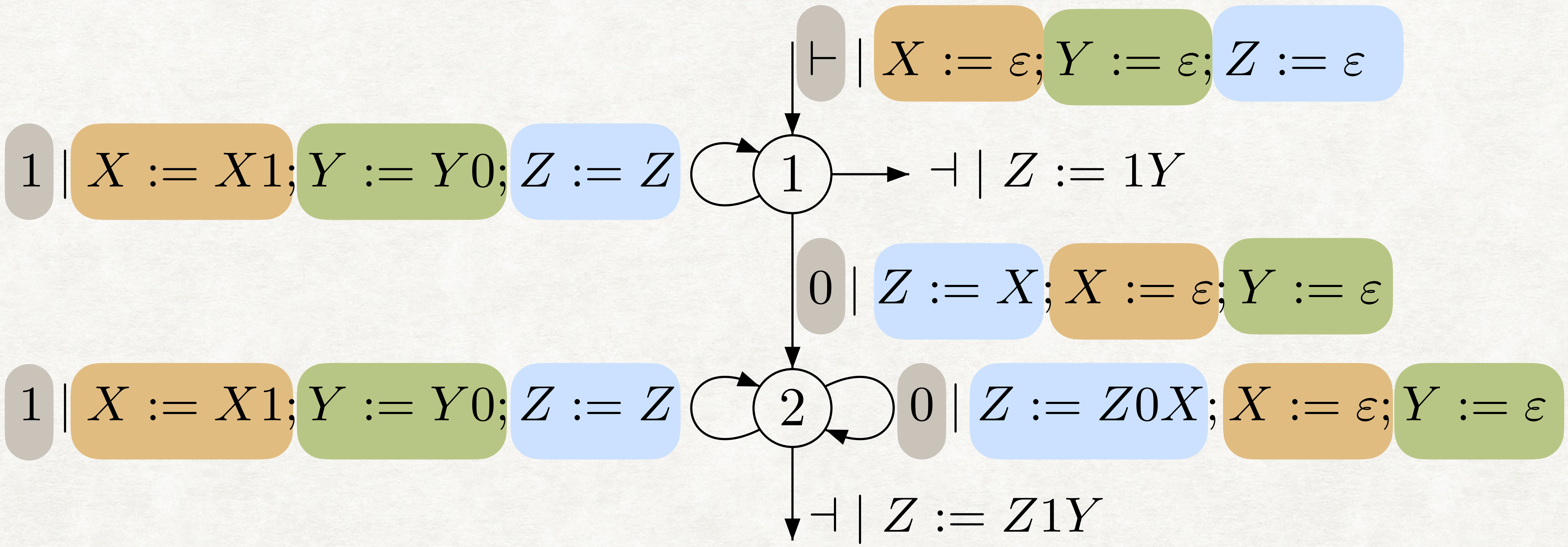
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition
- Register updates: X := u or X := u Y v or X := u Y v X w etc

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



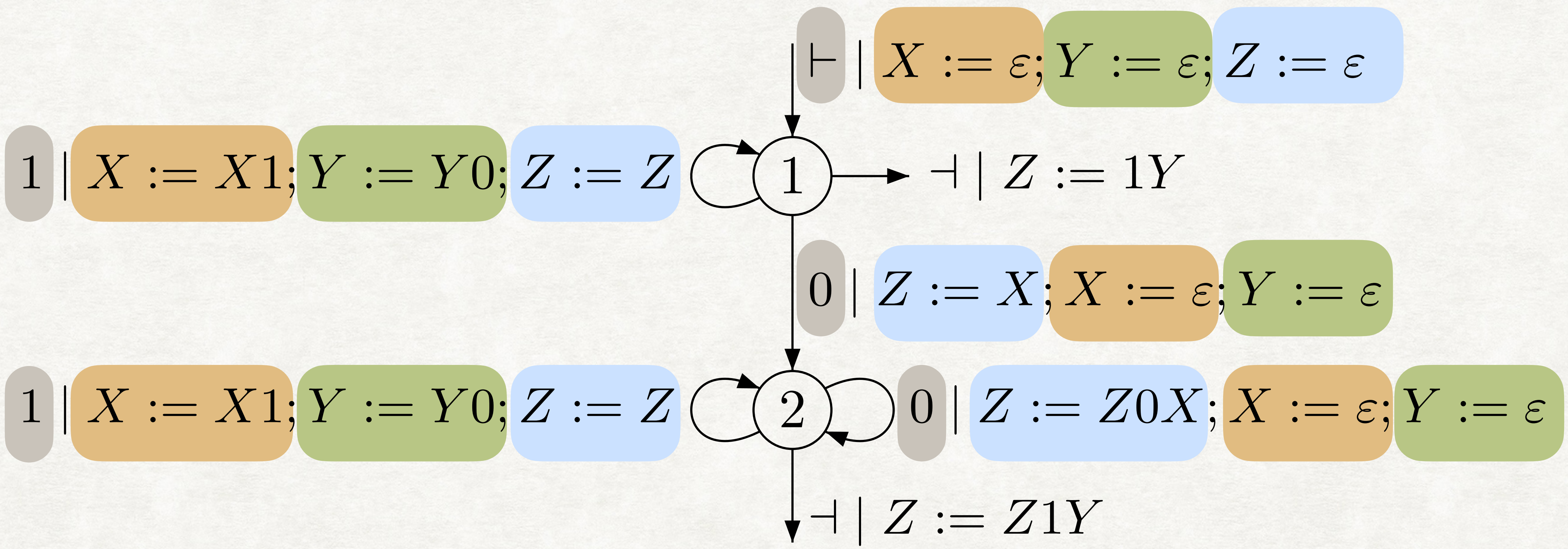
- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition
- Register updates: X := u or X := u Y v or X := u Y v X w etc
- copyless updates: Y := X1; X := X0 disallowed (X is duplicated)

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

STREAMING STRING TRANSDUCERS



- X keeps a copy of the last sequence of 1's
- Y is a sequence of 0's of same length
- Z keeps a copy of the input up to the last 0

11111111
100000000
1011010011111
1011010100000

- Deterministic 1-way parsing of the input and no composition
- Register updates: X := u or X := u Y v or X := u Y v X w etc
- copyless updates: Y := X1; X := X0 disallowed (X is duplicated)
- SST = 2DFT

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

SUMMARY

- Operational models (transducers)
  - 1DFT = sequential functions
  - f1NFT = rational functions
  - 2DFT = regular functions
  - Transducers with registers
- Modular descriptions
  - Rational expressions
  - Composition

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means "read u and output v"

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means “read u and output v”

$(1 \mid 0)^*$ replaces 1^n by 0^n

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means "read u and output v"

$(1 \mid 0)^*$ replaces 1^n by 0^n

copy $:= ((0 \mid 0) + (1 \mid 1))^*$

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means "read u and output v"

$(1 \mid 0)^*$ replaces 1^n by 0^n

$\mathsf{copy} := ((0 \mid 0) + (1 \mid 1))^*$

$\mathsf{inc0} := \mathsf{copy} \cdot (0 \mid 1) \cdot (1 \mid 0)^*$ $\qquad$ $\mathsf{dom}(\mathsf{inc0}) = (0+1)^* 0 1^*$

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means "read u and output v"

$(1 \mid 0)^*$ replaces 1^n by 0^n

$\mathsf{copy} := ((0 \mid 0) + (1 \mid 1))^*$

$\mathsf{inc0} := \mathsf{copy} \cdot (0 \mid 1) \cdot (1 \mid 0)^*$ $\qquad \mathsf{dom}(\mathsf{inc0}) = (0 + 1)^* 01^*$

$\mathsf{inc1} := (\varepsilon \mid 1) \cdot (1 \mid 0)^*$ $\qquad \mathsf{dom}(\mathsf{inc1}) = 1^*$

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means "read u and output v"

$(1 \mid 0)^*$ replaces 1^n by 0^n

$\mathsf{copy} := ((0 \mid 0) + (1 \mid 1))^*$

$\mathsf{inc0} := \mathsf{copy} \cdot (0 \mid 1) \cdot (1 \mid 0)^*$ $\qquad \mathsf{dom}(\mathsf{inc0}) = (0 + 1)^* 01^*$

$\mathsf{inc1} := (\varepsilon \mid 1) \cdot (1 \mid 0)^*$ $\qquad \mathsf{dom}(\mathsf{inc1}) = 1^*$

$\mathsf{inc} := \mathsf{inc0} + \mathsf{inc1}$ $\qquad \mathsf{dom}(\mathsf{inc}) = (0 + 1)^*$

REGULAR TRANSDUCER EXPRESSIONS

- Compositional and modular
- Unambiguous input parsing with regular expressions

$(u \mid v)$ means "read u and output v"

$(1 \mid 0)^*$ replaces 1^n by 0^n

$\mathsf{copy} := ((0 \mid 0) + (1 \mid 1))^*$

$\mathsf{inc0} := \mathsf{copy} \cdot (0 \mid 1) \cdot (1 \mid 0)^*$ $\qquad \mathsf{dom}(\mathsf{inc0}) = (0+1)^*01^*$

$\mathsf{inc1} := (\varepsilon \mid 1) \cdot (1 \mid 0)^*$ $\qquad \mathsf{dom}(\mathsf{inc1}) = 1^*$

$\mathsf{inc} := \mathsf{inc0} + \mathsf{inc1}$ $\qquad \mathsf{dom}(\mathsf{inc}) = (0+1)^*$

$\mathsf{copy} \cdot (1 \mid 0)^*$ is ambiguous

$1011 = 10 \cdot 11 = 101 \cdot 1 = 1011 \cdot \varepsilon$

SIMPLE RTE

SIMPLE RTE

- Compositional and modular
- Unambiguous input parsing

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^*$$

SIMPLE RTE

- Compositional and modular
- Unambiguous input parsing

UNAMBIGUOUS INPUT PARSING

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^*$$

SIMPLE RTE

- Compositional and modular
- Unambiguous input parsing
- 1-way parsing (no reverse)
- No duplication, no composition

UNAMBIGUOUS INPUT PARSING

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^*$$

SIMPLE RTE

- Compositional and modular
- Unambiguous input parsing
- 1-way parsing (no reverse)
- No duplication, no composition

UNAMBIGUOUS INPUT PARSING

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^*$$

➡ SRTE = Rational functions (1UFT = 1DFT with look-ahead = f1NFT)

SIMPLE RTE

- Compositional and modular
- Unambiguous input parsing
- 1-way parsing (no reverse)
- No duplication, no composition

UNAMBIGUOUS INPUT PARSING

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^*$$

➡ SRTE = Rational functions (1UFT = 1DFT with look-ahead = f1NFT)

➡ Special case of weighted automata (unambiguous)

[11] Schützenberger, On the definition of a family of automata, 1961

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times

➡ Hadamard product $$(f \odot g)(w) = f(w) \cdot g(w)$$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times

➡ Hadamard product $$(f \odot g)(w) = f(w) \cdot g(w)$$

duplicate: $w \mapsto w\$w$

$$(\mathsf{copy} \cdot (\varepsilon \mid \$)) \odot \mathsf{copy}$$

copy $:= ((0 \mid 0) + (1 \mid 1))^*$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times

➡ Hadamard product $$(f \odot g)(w) = f(w) \cdot g(w)$$

duplicate: $w \mapsto w\$w$

$$(\mathsf{copy} \cdot (\varepsilon \mid \$)) \odot \mathsf{copy}$$

exchange: $u\#v \mapsto vu$

$$\Big(\mathsf{erase} \cdot (\# \mid \varepsilon) \cdot \mathsf{copy}\Big) \odot \Big(\mathsf{copy} \cdot (\# \mid \varepsilon) \cdot \mathsf{erase}\Big)$$

$\mathsf{copy} := ((0 \mid 0) + (1 \mid 1))^*$ $\mathsf{erase} := ((0 \mid \varepsilon) + (1 \mid \varepsilon))^*$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ Composition $\qquad (f\circ g)(w) = f(g(w))$

$$f,g ::= (u,v) \mid f+g \mid f\cdot g \mid f^* \mid f\odot g \mid f\circ g$$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ Composition $$(f\circ g)(w) = f(g(w))$$

$$f, g ::= (u, v) \mid f+g \mid f\cdot g \mid f^* \mid f\odot g \mid f\circ g$$

UNAMBIGUOUS INPUT PARSING

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

➡ Composition $f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

➡ Composition $f, g ::= (u,v) \mid f+g \mid f\cdot g \mid f^* \mid f\odot g \mid f\circ g$

$$f := (\text{duplicate}\cdot(\#\mid\#))^*$$

$$f\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_1\$u_1\#u_2\$u_2\#u_3\$u_3\cdots u_n\$u_n\#$$

duplicate: $w \mapsto w\$w$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

➡ Composition $f,g ::= (u,v) \mid f+g \mid f\cdot g \mid f^* \mid f\odot g \mid f\circ g$

$$f := (\text{duplicate}\cdot(\#\mid\#))^*$$

$$f\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_1\$u_1\#u_2\$u_2\#u_3\$u_3\cdots u_n\$u_n\#$$

$$g := \text{erase}\cdot(\$\mid\varepsilon)\cdot(\text{exchange}\cdot(\$\mid\#))^*\cdot\text{erase}\cdot(\#\mid\varepsilon)$$

duplicate: $w\mapsto w\$w$ erase: $w\mapsto\varepsilon$ exchange: $u\#v\mapsto vu$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

➡ Composition $f, g ::= (u,v) \mid f+g \mid f\cdot g \mid f^* \mid f \odot g \mid f \circ g$

$$f := (\text{duplicate}\cdot(\# \mid \#))^*$$

$$f\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_1\$u_1\#u_2\$u_2\#u_3\$u_3\cdots u_n\$u_n\#$$

$$g := \text{erase}\cdot(\$ \mid \varepsilon)\cdot(\text{exchange}\cdot(\$ \mid \#))^*\cdot\text{erase}\cdot(\# \mid \varepsilon)$$

$$g\circ f\colon u_1\#u_2\#\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

$$h = g \circ f$$

duplicate: $w \mapsto w\$w$ erase: $w \mapsto \varepsilon$ exchange: $u\#v \mapsto vu$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- A 2DFT may produce output while reading its input backwards

$$\text{reverse}: a_1 a_2 \cdots a_n \mapsto a_n \cdots a_2 a_1$$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- A 2DFT may produce output while reading its input backwards

$$\text{reverse}: a_1 a_2 \cdots a_n \mapsto a_n \cdots a_2 a_1$$

➡ Not possible with

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- A 2DFT may produce output while reading its input backwards

$$\text{reverse}\colon a_1 a_2 \cdots a_n \mapsto a_n \cdots a_2 a_1$$

➡ Not possible with

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

➡ Add reverse as basic function

$$f, g ::= \text{reverse} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- A 2DFT may produce output while reading its input backwards

$$\text{reverse}: a_1a_2\cdots a_n \mapsto a_n\cdots a_2a_1$$

➡ Not possible with

$$f,g ::= (u,v) \mid f+g \mid f\cdot g \mid f^* \mid f\odot g \mid f\circ g$$

➡ Add reverse as basic function

$$f,g ::= \text{reverse} \mid (u,v) \mid f+g \mid f\cdot g \mid f^* \mid f\odot g \mid f\circ g$$

➡ RTE-chr = 2DFT = Regular functions

WITH COMPOSITION, HADAMARD PRODUCT AND REVERSE

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- Add reverse as basic function

$$f, g ::= \text{reverse} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- Add reverse as basic function

$$f, g ::= \textsf{reverse} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

- Replace Hadamard product with $\textsf{duplicate}\colon w \mapsto w\$w$

$$f \odot g = (f \cdot (\$ \mid \varepsilon) \cdot g) \circ \textsf{duplicate}$$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- Add reverse as basic function

$$f, g ::= \text{reverse} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

- Replace Hadamard product with $\text{duplicate}\colon w \mapsto w\$w$

$$f \odot g = (f \cdot (\$ \mid \varepsilon) \cdot g) \circ \text{duplicate}$$

$$f, g ::= \text{reverse} \mid \text{duplicate} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \circ g$$

RTE WITH COMPOSITION AND REGULAR FUNCTIONS

- Add reverse as basic function

$$f, g ::= \textsf{reverse} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid f \circ g$$

- Replace Hadamard product with duplicate: $w \mapsto w\$w$

$$f \odot g = (f \cdot (\$ \mid \varepsilon) \cdot g) \circ \textsf{duplicate}$$

$$f, g ::= \textsf{reverse} \mid \textsf{duplicate} \mid (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \circ g$$

- RTE-cdr = 2DFT = Regular functions

WITH COMPOSITION, DUPLICATE AND REVERSE

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ 2-chained Kleene iteration [12] $[K,h]^{2+}$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ 2-chained Kleene iteration [12] $[K,h]^{2+}$

parse an input word as $w = u_1u_2\cdots u_n$ with $u_1,\ldots,u_n \in K$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ 2-chained Kleene iteration [12] $[K,h]^{2+}$

parse an input word as $w = u_1u_2\cdots u_n$ with $u_1,\ldots,u_n \in K$

$$[K,h]^{2+}\colon w \mapsto h(u_1u_2)h(u_2u_3)\cdots h(u_{n-1}u_n)$$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ 2-chained Kleene iteration [12] $[K,h]^{2+}$

parse an input word as $w = u_1u_2\cdots u_n$ with $u_1,\ldots,u_n \in K$

$$[K,h]^{2+}\colon w \mapsto h(u_1u_2)h(u_2u_3)\cdots h(u_{n-1}u_n)$$

UNAMBIGUOUS INPUT PARSING

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

$$h\colon u_1\#u_2\#u_3\cdots u_n\# \mapsto u_2u_1\#u_3u_2\#\cdots u_nu_{n-1}\#$$

h cannot be described using $+$, $\cdot$, $*$, $\odot$

➡ 2-chained Kleene iteration [12] $[K,h]^{2+}$

parse an input word as $w = u_1u_2\cdots u_n$ with $u_1,\ldots,u_n \in K$

$$[K,h]^{2+}\colon w \mapsto h(u_1u_2)h(u_2u_3)\cdots h(u_{n-1}u_n)$$

$$h = [\{0,1\}^*\#, \textsf{exchange}\cdot(\# \mid \#)]^{2+}$$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

➡ 2-chained Kleene iteration [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, f]^{2+}$$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

➡ 2-chained Kleene iteration [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, f]^{2+}$$

- A 2DFT may produce output while reading its input backwards

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

➡ 2-chained Kleene iteration [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, f]^{2+}$$

- A 2DFT may produce output while reading its input backwards

reverse cannot be described using $+$, $\cdot$, $*$, $\odot$, $2+$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

➡ 2-chained Kleene iteration [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, f]^{2+}$$

- A 2DFT may produce output while reading its input backwards

reverse cannot be described using $+$, $\cdot$, $*$, $\odot$, $2+$

➡ Reversed 2-chained Kleene iteration [12] $[K, h]^{r\text{-}2+}$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

➡ 2-chained Kleene iteration [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, f]^{2+}$$

- A 2DFT may produce output while reading its input backwards

reverse cannot be described using $+$, $\cdot$, $*$, $\odot$, $2+$

➡ Reversed 2-chained Kleene iteration [12] $[K, h]^{r\text{-}2+}$

parse an input word as $w = u_1 u_2 \cdots u_n$ with $u_1, \ldots, u_n \in K$

UNAMBIGUOUS INPUT PARSING

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces

➡ 2-chained Kleene iteration [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, f]^{2+}$$

- A 2DFT may produce output while reading its input backwards

reverse cannot be described using $+$, $\cdot$, $*$, $\odot$, $2+$

➡ Reversed 2-chained Kleene iteration [12] $[K, h]^{r\text{-}2+}$

parse an input word as $w = u_1 u_2 \cdots u_n$ with $u_1, \ldots, u_n \in K$

$$[K, h]^{r\text{-}2+} : w \mapsto h(u_{n-1}u_n) \cdots h(u_2u_3)h(u_1u_2)$$

UNAMBIGUOUS INPUT PARSING

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces
- A 2DFT may produce output while reading its input backwards

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces
- A 2DFT may produce output while reading its input backwards

➡ Full RTE [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, h]^{2+} \mid [K, h]^{r\text{-}2+}$$

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

FULL RTE FOR REGULAR FUNCTIONS

- A 2DFT (or 2UFT) may read its input several times in pieces
- A 2DFT may produce output while reading its input backwards

➡ Full RTE [12]

$$f, g ::= (u, v) \mid f + g \mid f \cdot g \mid f^* \mid f \odot g \mid [K, h]^{2+} \mid [K, h]^{r\text{-}2+}$$

➡ Full RTE = 2DFT = Regular functions

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

CONCLUDING REMARKS

CONCLUDING REMARKS

- Sequential functions: 1DFT
- Rational functions: f1NFT = 1UFT = 1DFT with l-a = simple RTE
- Regular functions RF: 2DFT = 2DFT with l-a & l-b = 2UFT = f2NFT

CONCLUDING REMARKS

- Sequential functions: 1DFT
- Rational functions: f1NFT = 1UFT = 1DFT with l-a = simple RTE
- Regular functions RF: 2DFT = 2DFT with l-a & l-b = 2UFT = f2NFT
- Equivalence is decidable
- Closed under composition [6]

CONCLUDING REMARKS

- Sequential functions: 1DFT
- Rational functions: f1NFT = 1UFT = 1DFT with l-a = simple RTE
- Regular functions RF: 2DFT = 2DFT with l-a & l-b = 2UFT = f2NFT
- Equivalence is decidable
- Closed under composition [6]
- RF = simple programs with registers [6]
- RF = streaming string transducers [10]

CONCLUDING REMARKS

- Sequential functions: 1DFT
- Rational functions: f1NFT = 1UFT = 1DFT with l-a = simple RTE
- Regular functions RF: 2DFT = 2DFT with l-a & l-b = 2UFT = f2NFT
- Equivalence is decidable
- Closed under composition [6]
- RF = simple programs with registers [6]
- RF = streaming string transducers [10]
- RF = regular expressions with composition, reverse, duplicate
- RF = regular expressions with (reversed) 2-chained Kleene-+ [12]

CONCLUDING REMARKS

- Sequential functions: 1DFT
- Rational functions: f1NFT = 1UFT = 1DFT with l-a = simple RTE
- Regular functions RF: 2DFT = 2DFT with l-a & l-b = 2UFT = f2NFT
- Equivalence is decidable
- Closed under composition [6]
- RF = simple programs with registers [6]
- RF = streaming string transducers [10]
- RF = regular expressions with composition, reverse, duplicate
- RF = regular expressions with (reversed) 2-chained Kleene-+ [12]
- RF = MSO transductions [8]

CONCLUDING REMARKS

- Sequential functions: 1DFT
- Rational functions: f1NFT = 1UFT = 1DFT with l-a = simple RTE
- Regular functions RF: 2DFT = 2DFT with l-a & l-b = 2UFT = f2NFT
- Equivalence is decidable
- Closed under composition [6]
- RF = simple programs with registers [6]
- RF = streaming string transducers [10]
- RF = regular expressions with composition, reverse, duplicate
- RF = regular expressions with (reversed) 2-chained Kleene-+ [12]
- RF = MSO transductions [8]
- RF = Regular list functions [16]

REFERENCES

[1] Schützenberger, Sur les relations rationnelles, 1975

[2] Gurari & Ibarra, A note on finite-valued and finitely ambiguous transducers, 1983

[3] Weber & Klemm, Economy of description for single-valued transducers, 1995

[4] Choffrut, Une caractérisation des fonctions séquentielles … en tant que relations rationnelles, 1977

[5] Berstel, Transductions and Context-Free Languages, 1979

[6] Chytil & Jákl, Serial composition of 2-way finite-state transducers and simple programs on strings, 1977

[7] Dartois, Fournier, Jecker, Lhote, On reversible transducers, 2017

[8] Engelfriet & Hoogeboom, MSO definable string transductions and two-way finite-state transducers, 2001

[9] Culik & Karhumäki, The equivalence of finite valued transducers (on HDT0L languages) is decidable, 1986

[10] Alur & Deshmukh, Nondeterministic Streaming String Transducers, 2011

[11] Schützenberger, On the definition of a family of automata, 1961

[12] Alur & Freilich & Raghothaman, Regular combinators for string transformations, 2014

[13] Alur & D'Antoni & Raghothaman, DReX: Declarative Language for … Regular String Transformations, 2015

[14] Dave & Gastin & Krishna, Regular Transducer Expressions for Regular Transformations, 2018

[15] Baudru & Reynier, From Two-Way Transducers to Regular Function Expressions, 2018

[16] M. Bojanczyk & L. David & S. Krishna, Regular and First-Order List Functions, 2018