
チューリングマシンを学ぼう！
2021/01/29 マルレク基礎

はじめに

「マルレク基礎」は、コンピュータ・サイエンスの基礎を改めて学ぼうという方を対象としたオンライン・セミナーである。

今回は、コンピュータ・サイエンスの基礎として「チューリングマシン」を取りあげる。

初めてチューリングマシンを学ぶ人を念頭において、チューリングマシンの基礎をわかりやすく解説したいと思う。

同時に、過去にチューリングマシンを学んだことがある人にも、知識の再整理に役に立つと考えている。

はじめに

第一部の「チューリングマシンの基本」では、まず、チューリングマシンの基本的な部品である「テープ」と「ヘッド」について、それがどんな機能を持っているかを話す。

ついで、チューリングマシンで一番重要な概念である「状態」と「状態の遷移」の話をする。

ヘッドが文字が書かれたテープの上を、文字を読み書きしながら移動するにつれて、マシンの状態は変わっていく。単純だが、このチューリングマシンの動作のイメージを持つことは、とても大事である。

はじめに

第二部の「チューリングマシンをプログラムする」では、チューリングマシンで簡単なプログラミングを始める。
ただ、チューリングマシンの「命令」をどのように表現するかについては、統一的な方法が決まっているわけではない。ここでは、「命令」の表記を一つ導入して、その表記のもとでいくつかの簡単なプログラムを書いてみる。

プログラミングの経験のある人は、実際に、自分でサンプルのチューリングマシンのプログラムを書いてみることをお勧めする。変数も配列も関数の呼び出しもない、プリミティブなものだが、それが、普通の日常のプログラミングと本質的には同じものであることが、きっとわかると思う。

はじめに

第三部の「複数のテープと複数のマシン」では、複数のテープを持つようにチューリングマシンの拡張し、また、複数のチューリングマシンを考える。

ただ、このセミナーの基本的な獲得目標は、複数のテープを持つ一つのチューリングマシンの振る舞いを理解することにおこうと思っている。複数のチューリングマシンを扱った、最後の二つの節はオプションである。

今回のセミナーでは、触れることができなかったのだが、チューリングマシンは、「計算可能性」「計算複雑性」の理論に深い繋がりがある。それについては、後日、「マルゼミ」でとりあげようと思っている。

図は、Interactive Proofを初めて導入した Goldwasserらの論文に出てきたもので、Interactive Proofが二台のチューリングマシンA, Bと、7本のテープで表現されている。

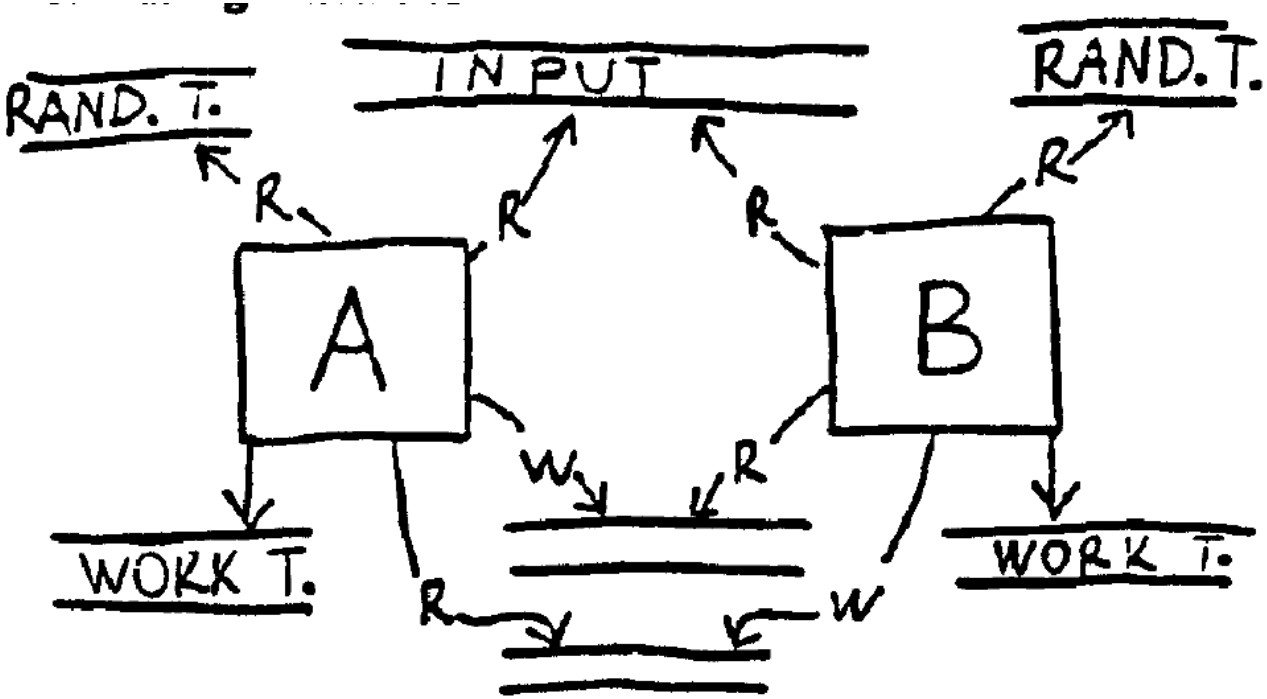


Fig. 2: an interactive pair of Turing machines

https://dl.acm.org/doi/pdf/10.1145/22145.22178

Agenda

チューリングマシンを学ぼう！

Part I

チューリングマシンの基本

Part II

チューリングマシンをプログラムする

Part III

複数のテープと複数のマシン

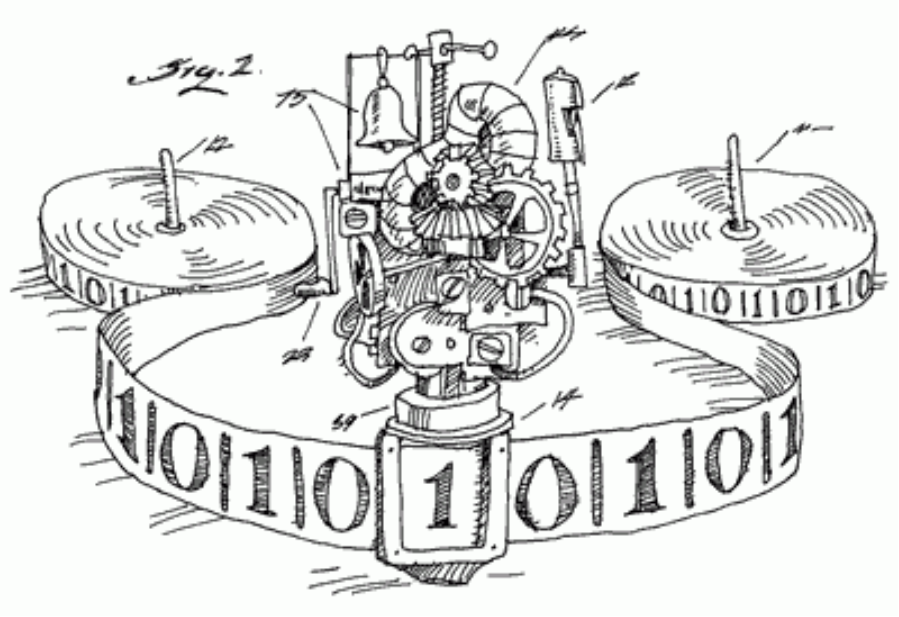
Part I

チューリングマシンの基礎

Agenda Part I
チューリングマシンの基本

- テープとヘッド
  - テープ
  - ヘッドの動き
  - ヘッドの役割（読み取り）
  - ヘッドの役割（書き込み）
- 状態と状態の遷移
  - テープの状態とヘッドの位置
  - チューリングマシンの状態
  - ヘッドの移動とチューリングマシンの状態の遷移
  - チューリングマシンの状態の遷移のサンプル

テープとヘッド



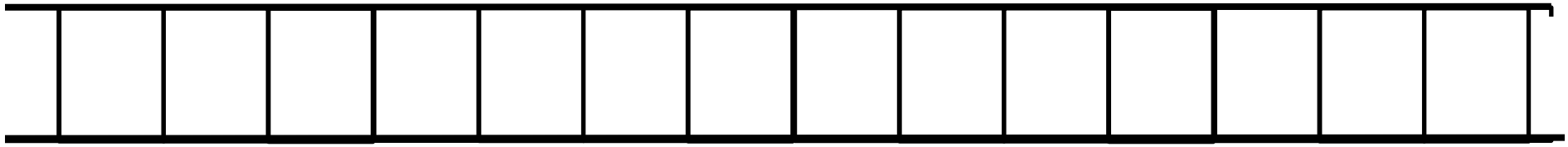
「チューリングマシンを学ぼう！」

Part Ⅰ　チューリングマシンの基本

テープ

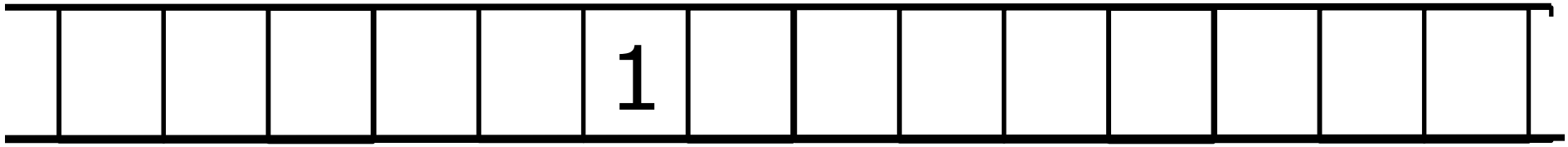
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。



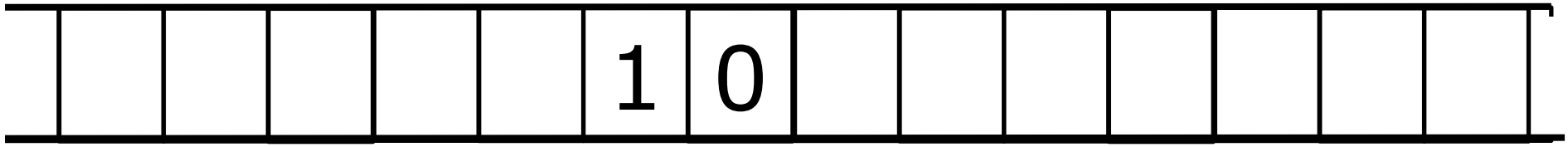
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。



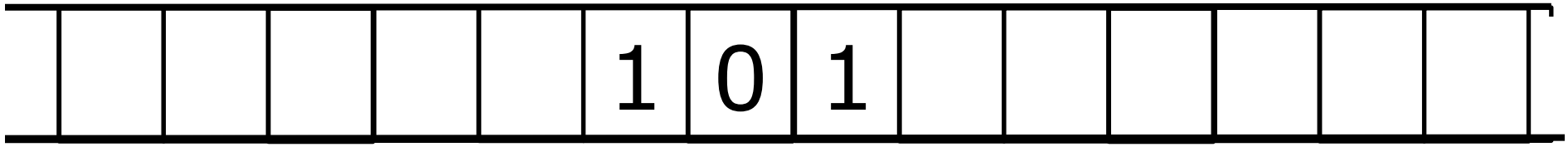
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。



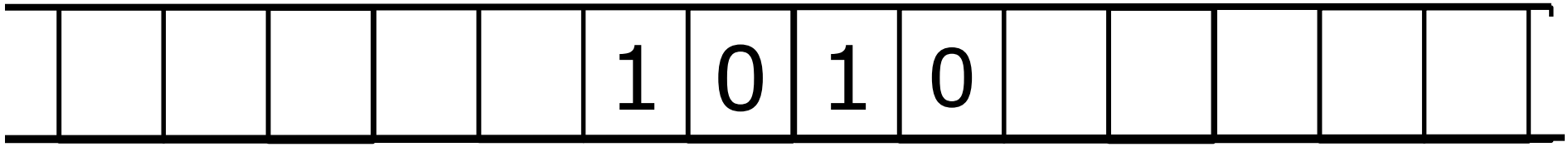
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。



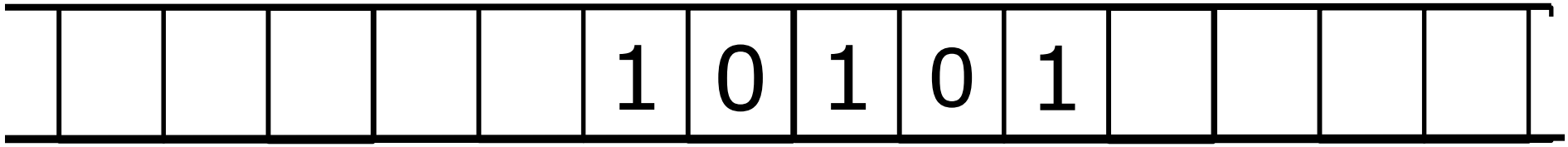
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。



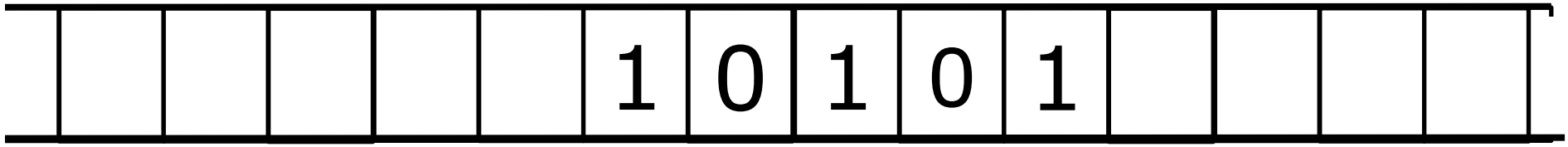
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。



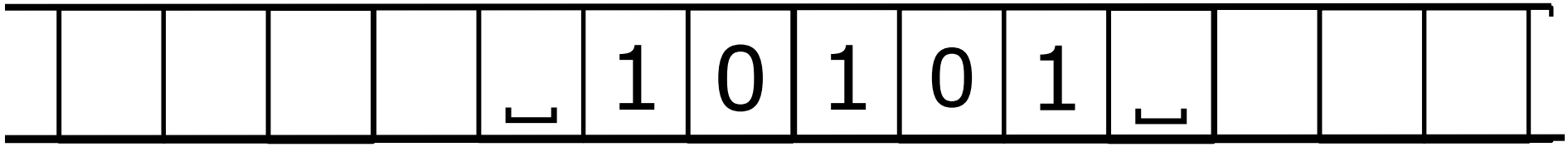
テープ

- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。
  この例の場合、“10101”の文字列がテープに記されている。



テープ

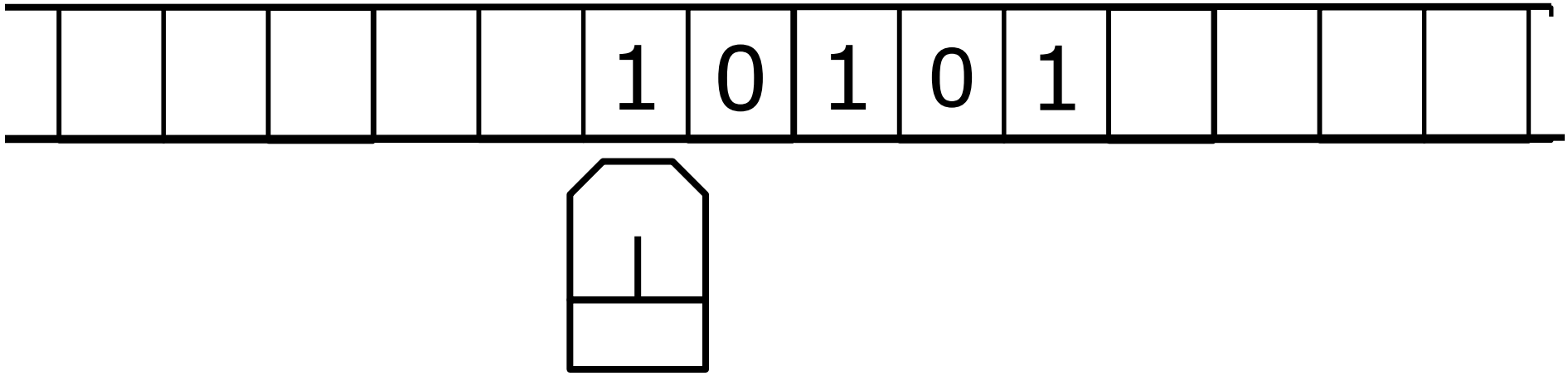
- テープは、左右に伸びる長いもので、マス目で区切られている。
- テープの上には、一マスに一つの文字が記されている。
  この例の場合、"10101"の文字列がテープに記されている。
- 文字が記されていないマス目は、「空白」（記号'␣' あるいは 'B'）が記されているという。（省略してもいい）



ヘッドの動き

ヘッドの動き

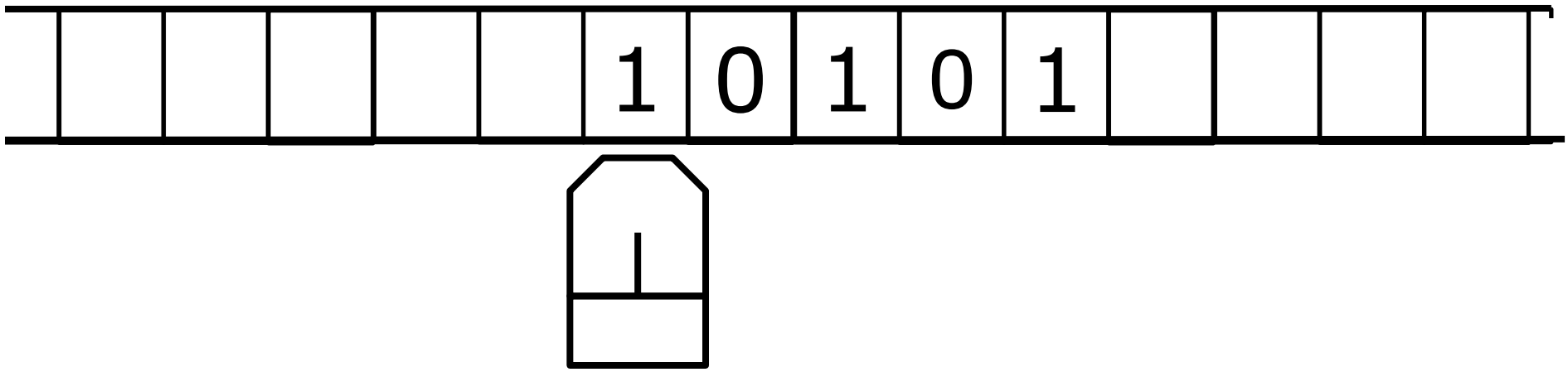
- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

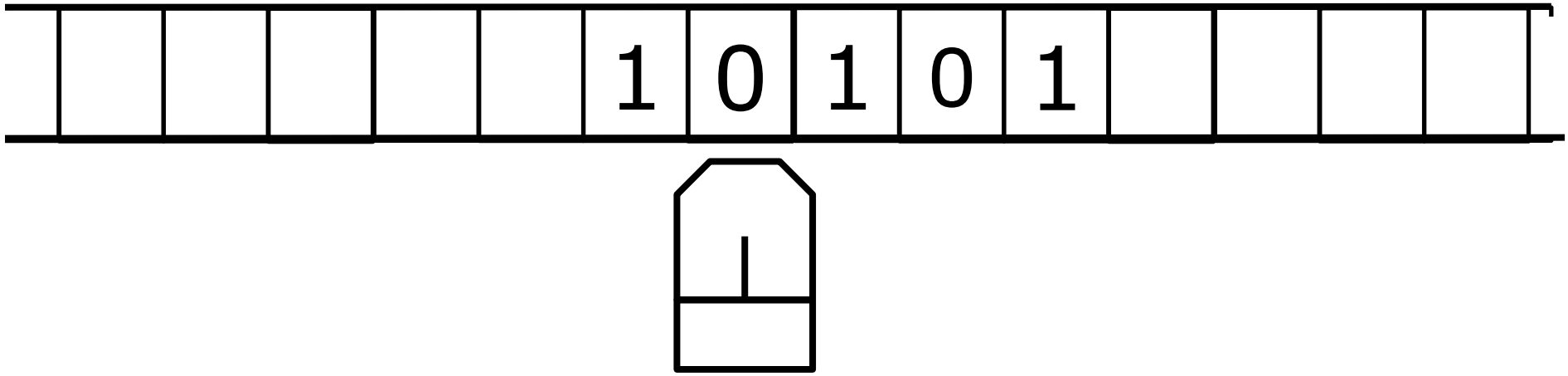
R⇒



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

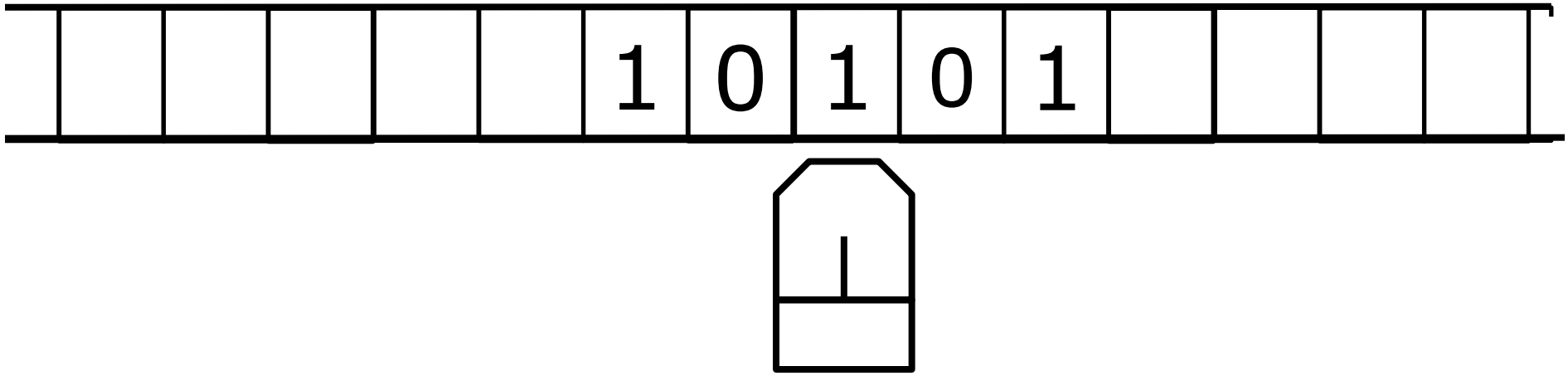
R⇒



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

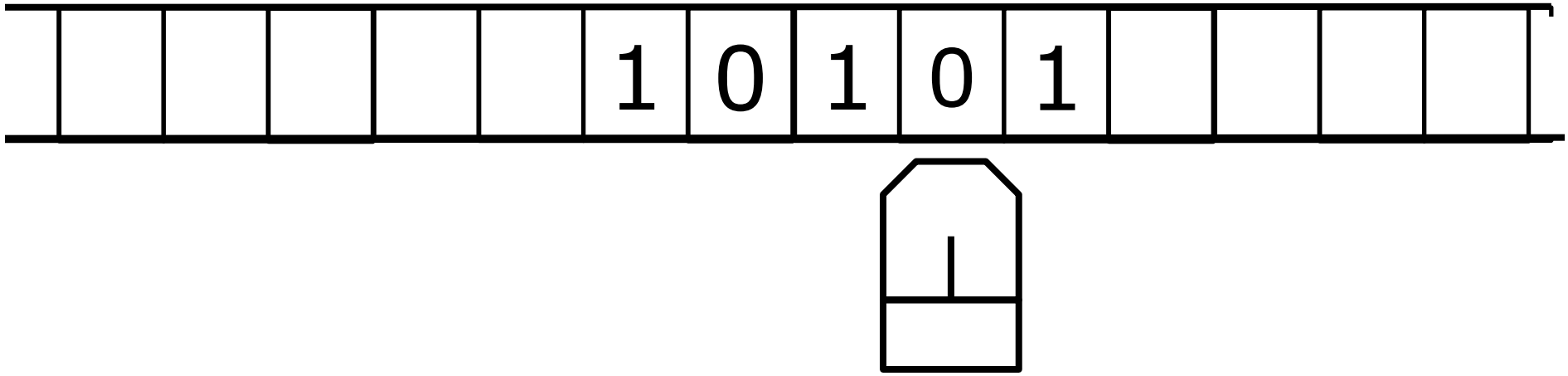
R⇒



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

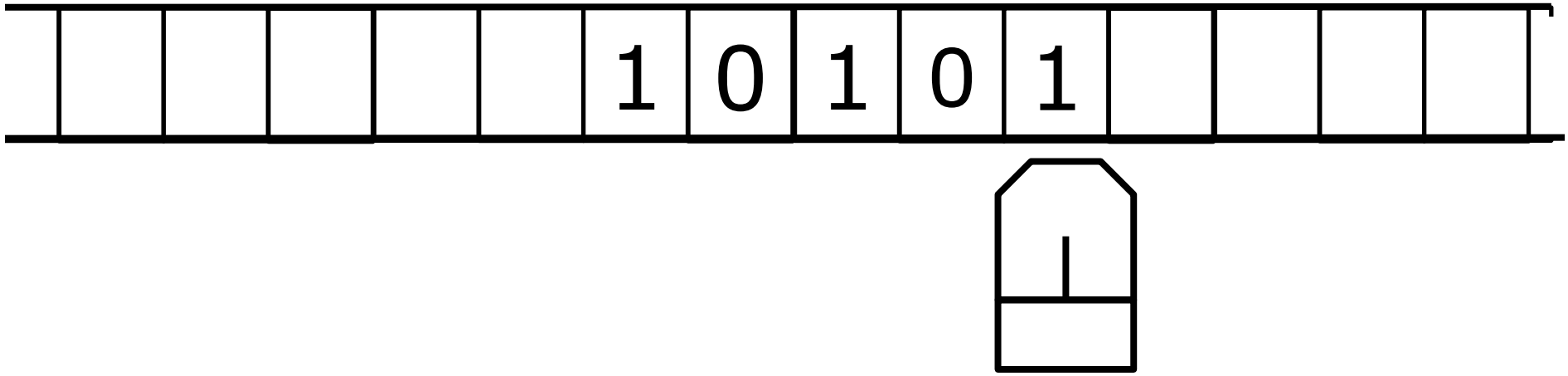
R⇒



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

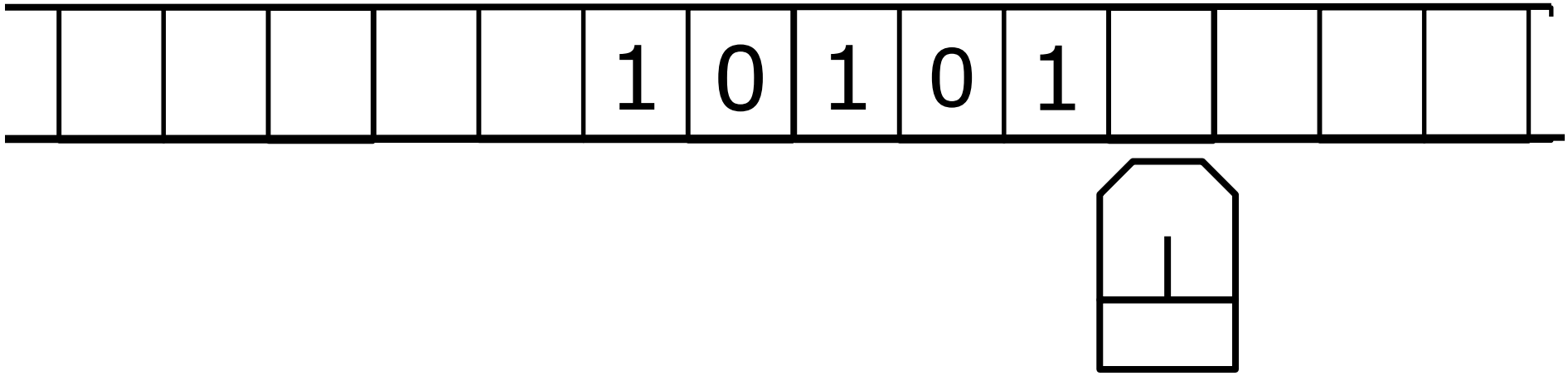
R⇒



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

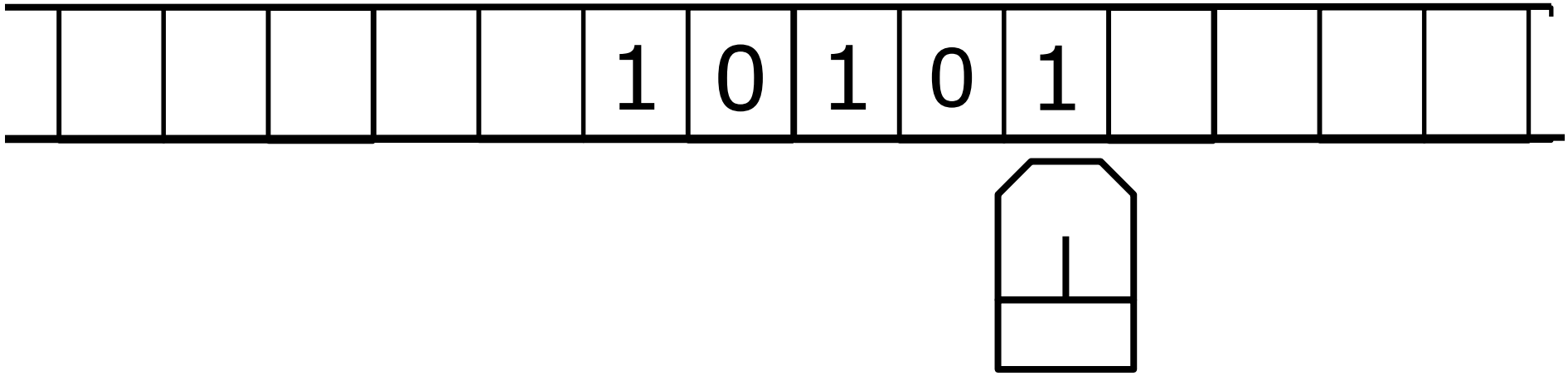
⇦L



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

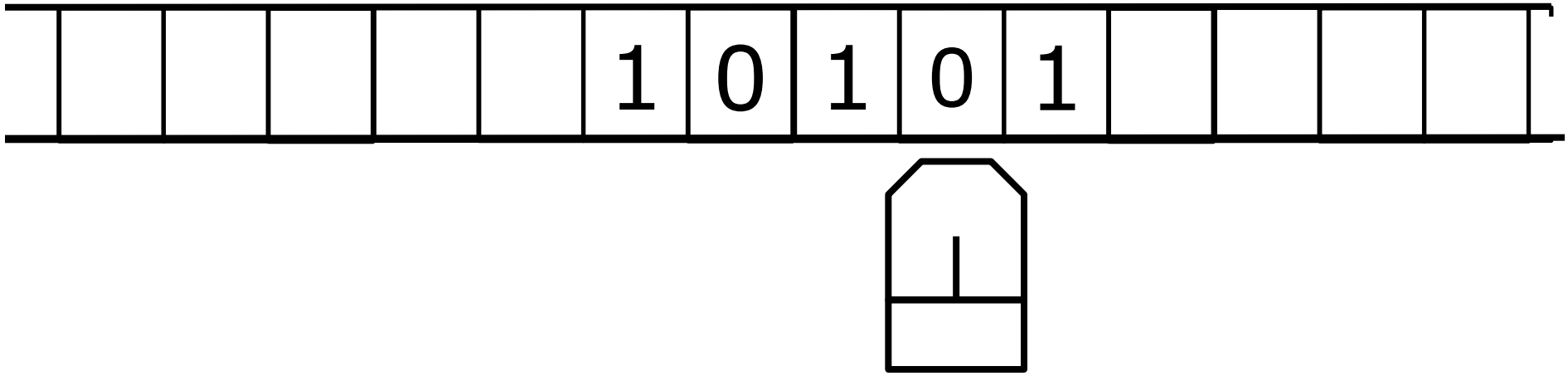
⇦L



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

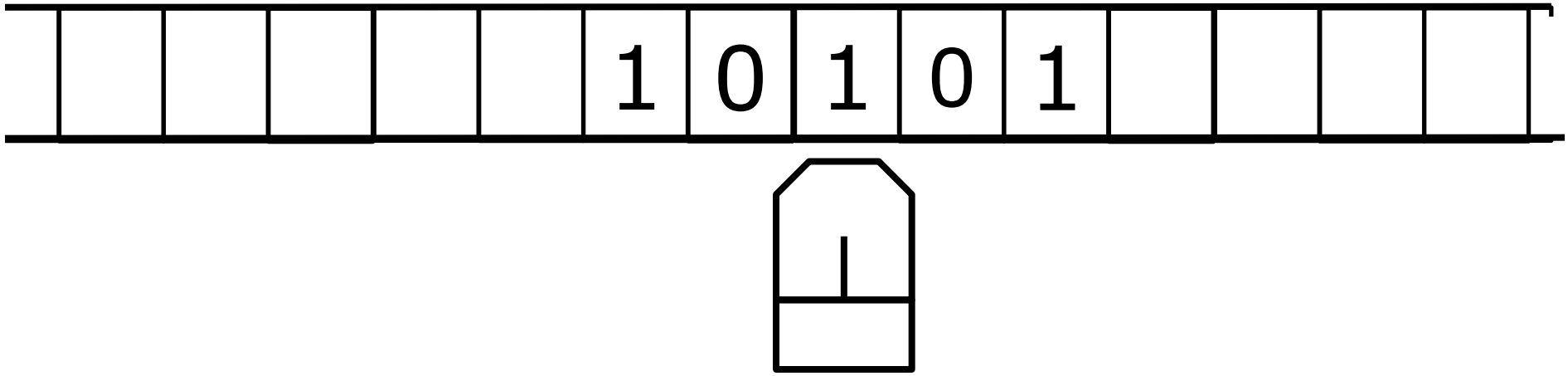
⇦L



ヘッドの動き

- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

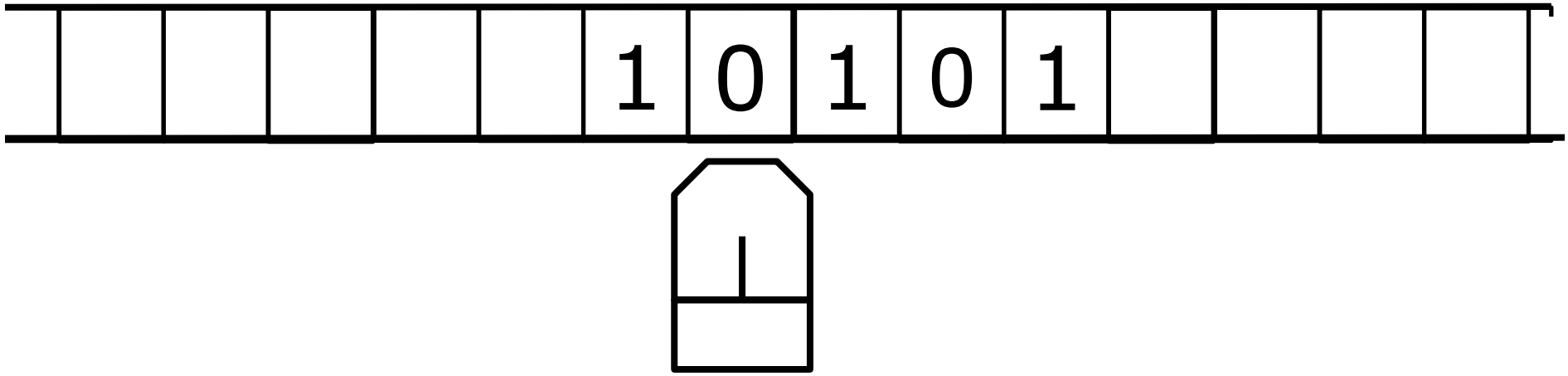
⇦L



ヘッドの動き

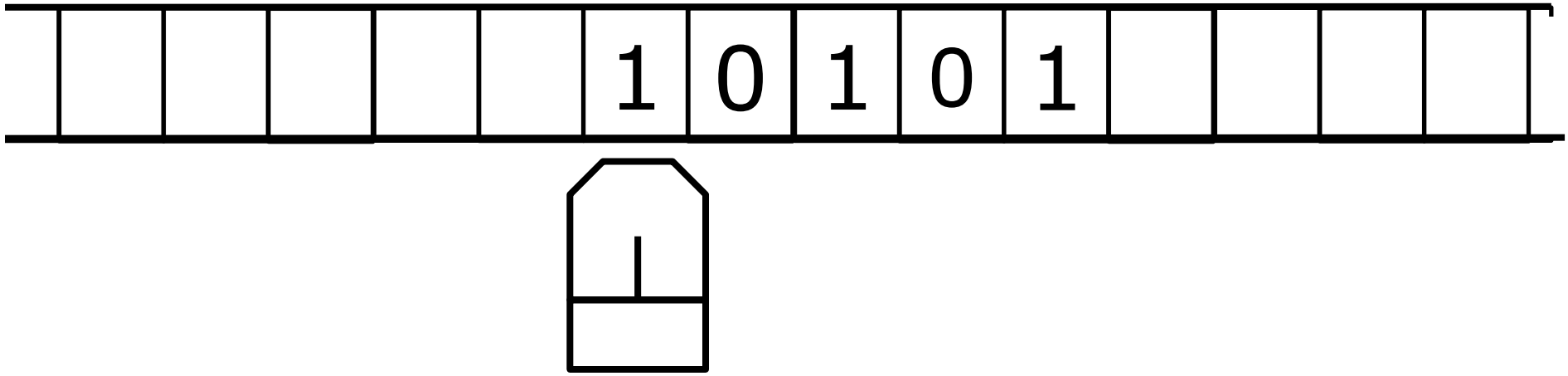
- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。

⇦L



ヘッドの動き

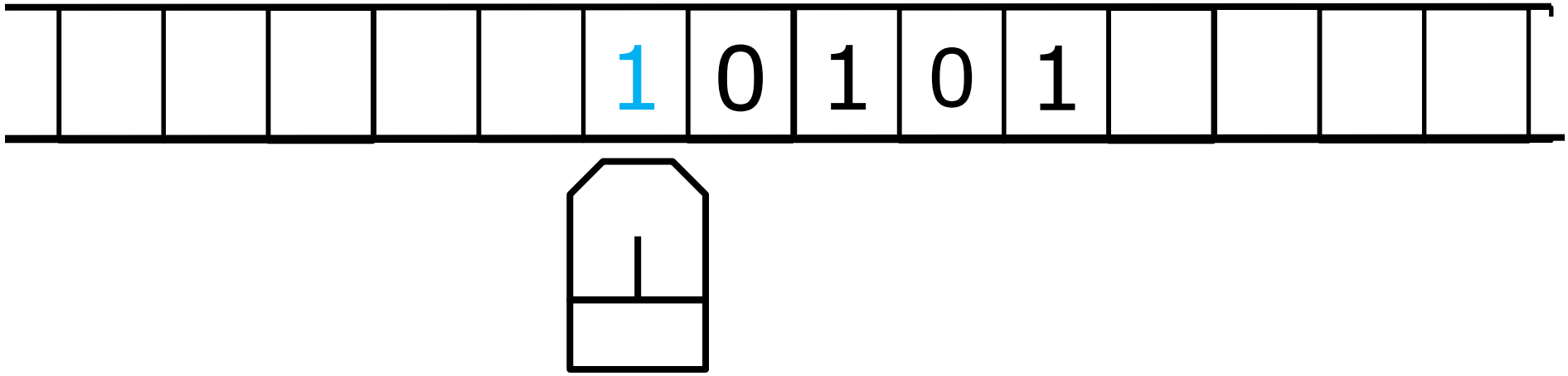
- ヘッドは、テープの一つのマス目の上を、左右に一マスずつ移動できる。右への移動をR、左への移動をLで表す。



ヘッドの役割（読み取り）

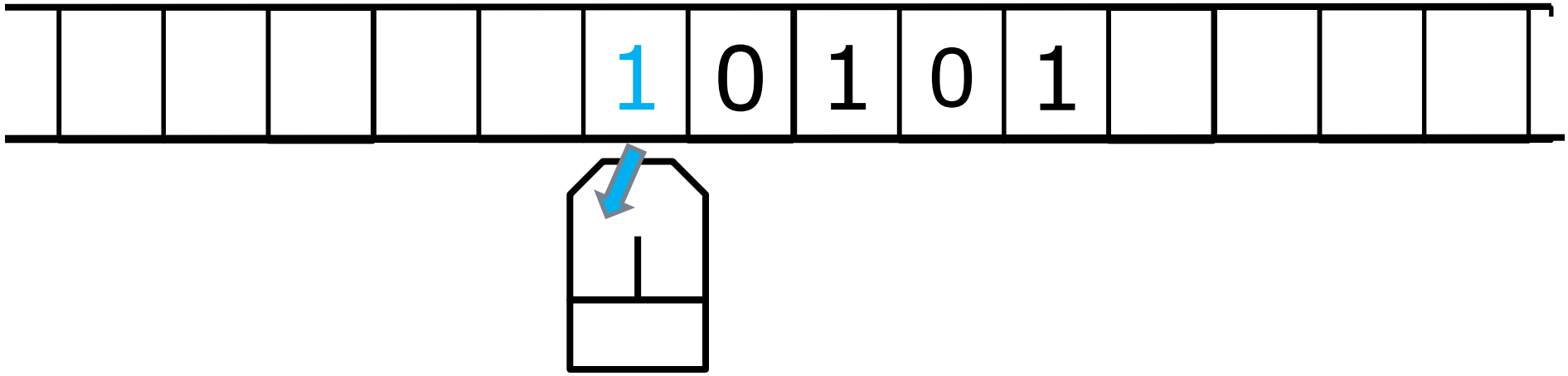
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



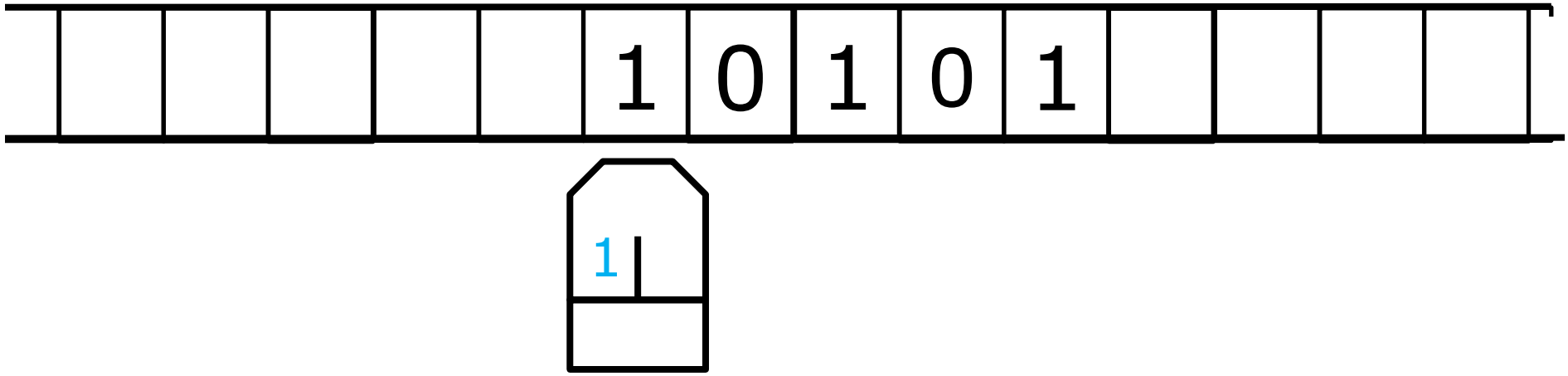
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



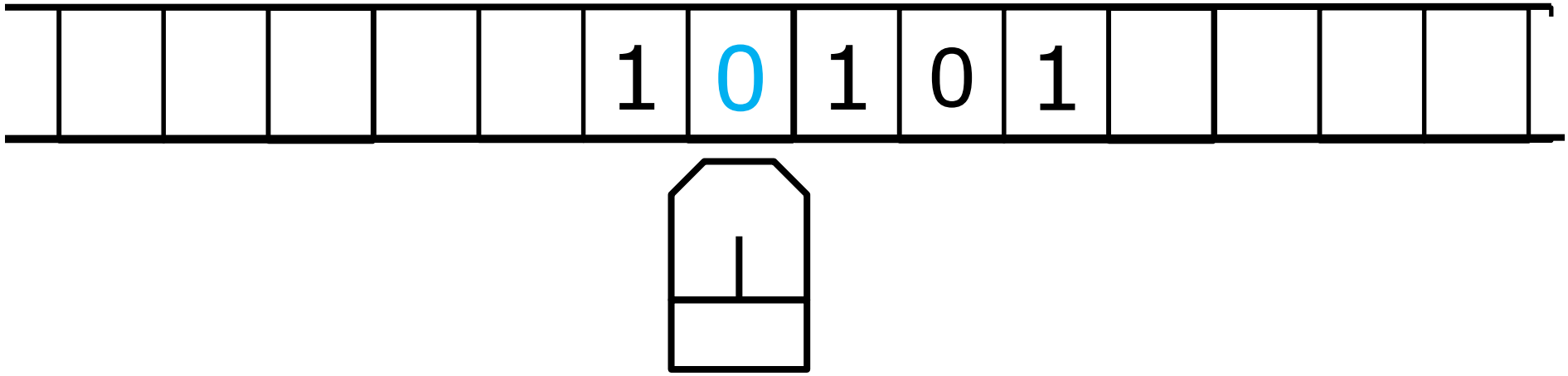
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



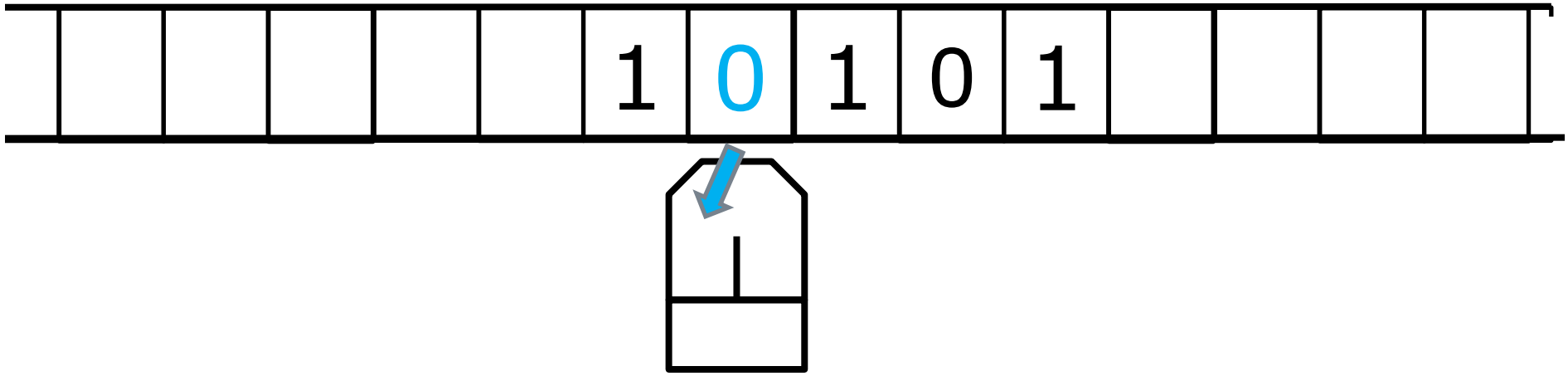
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



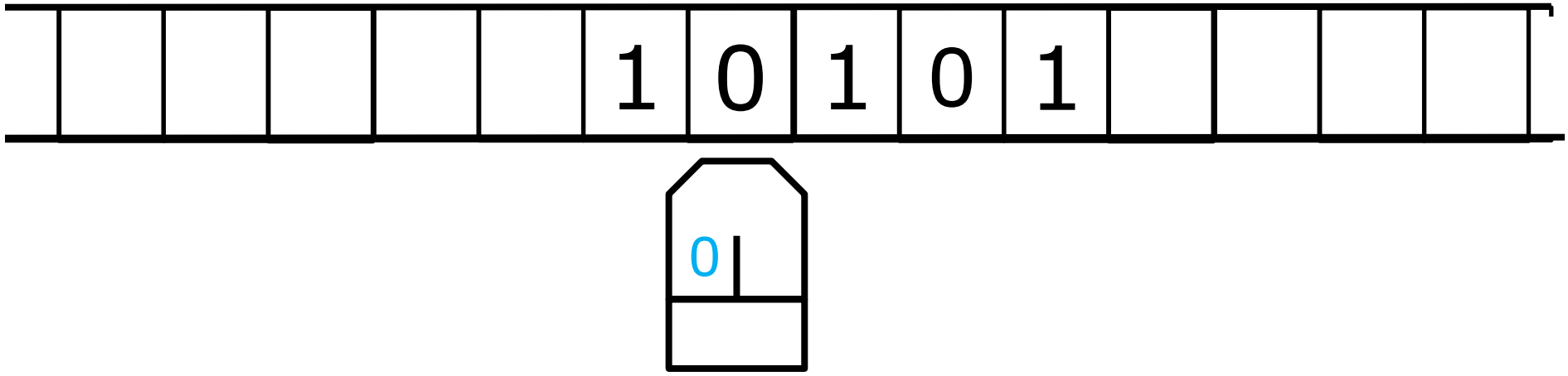
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



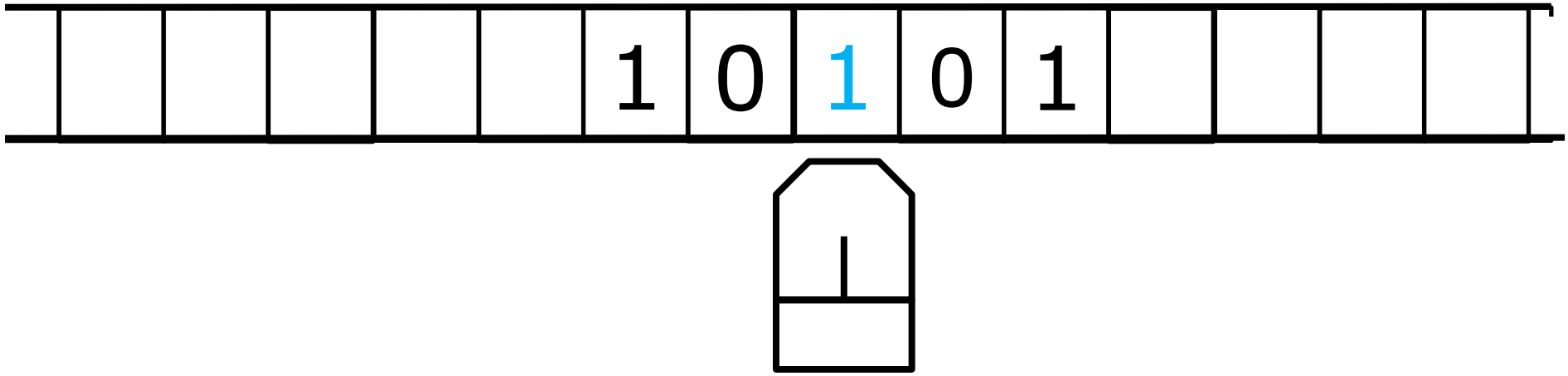
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



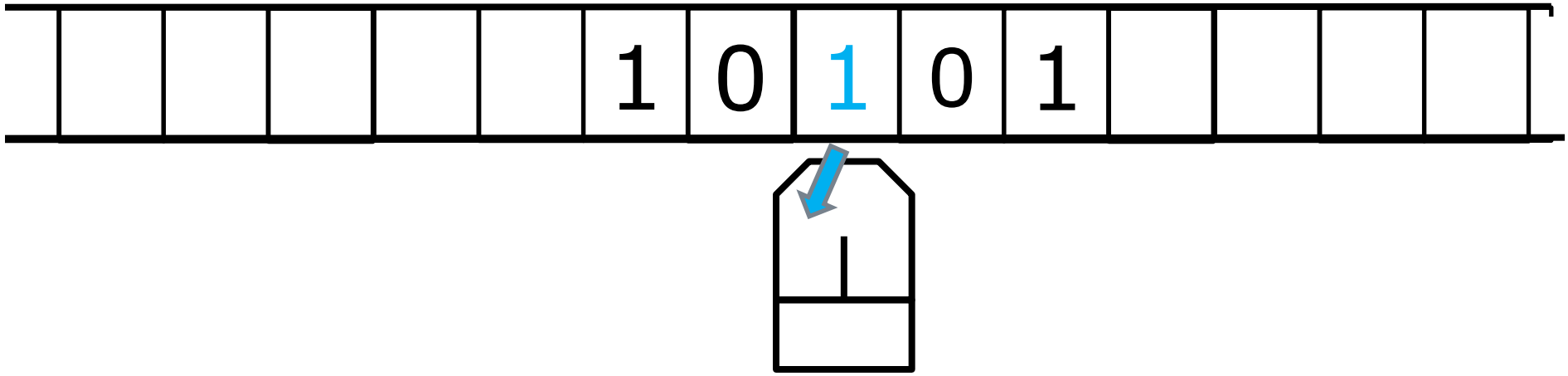
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



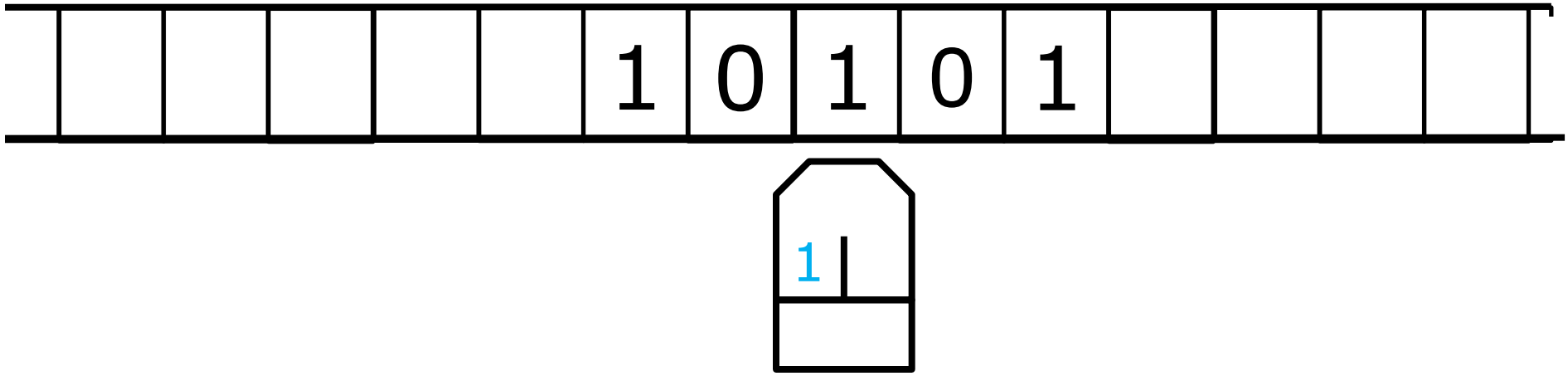
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



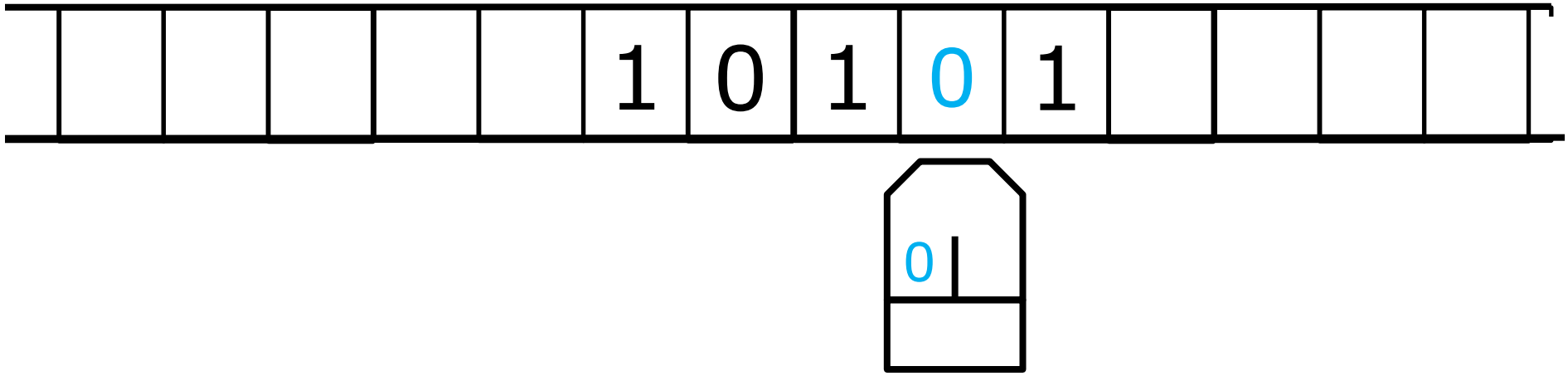
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



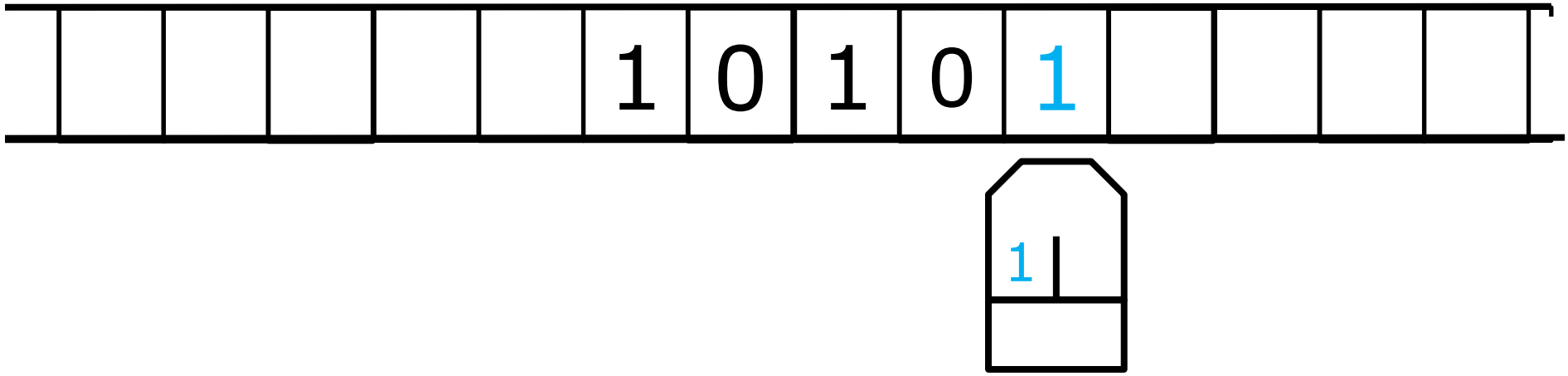
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



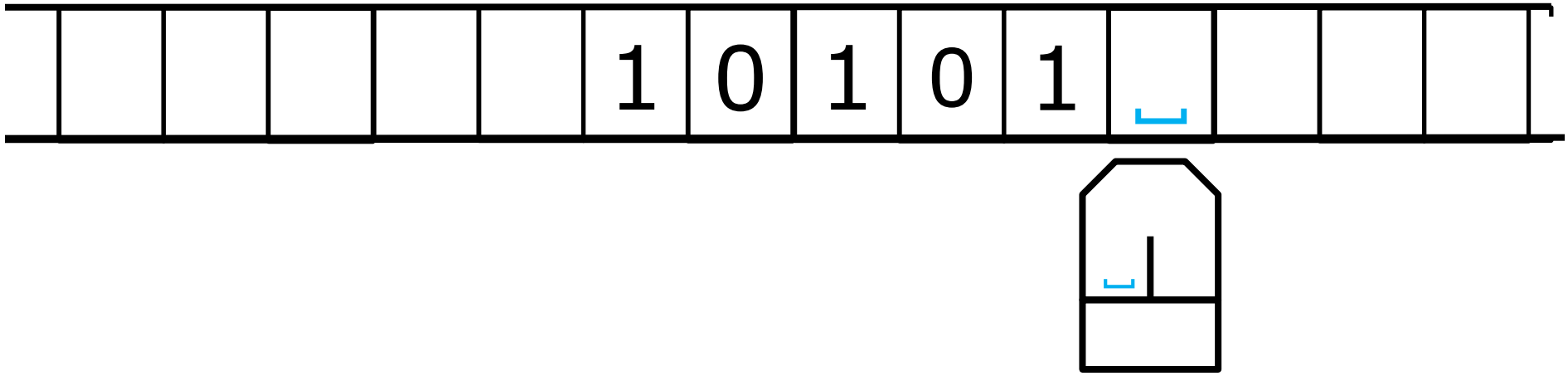
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



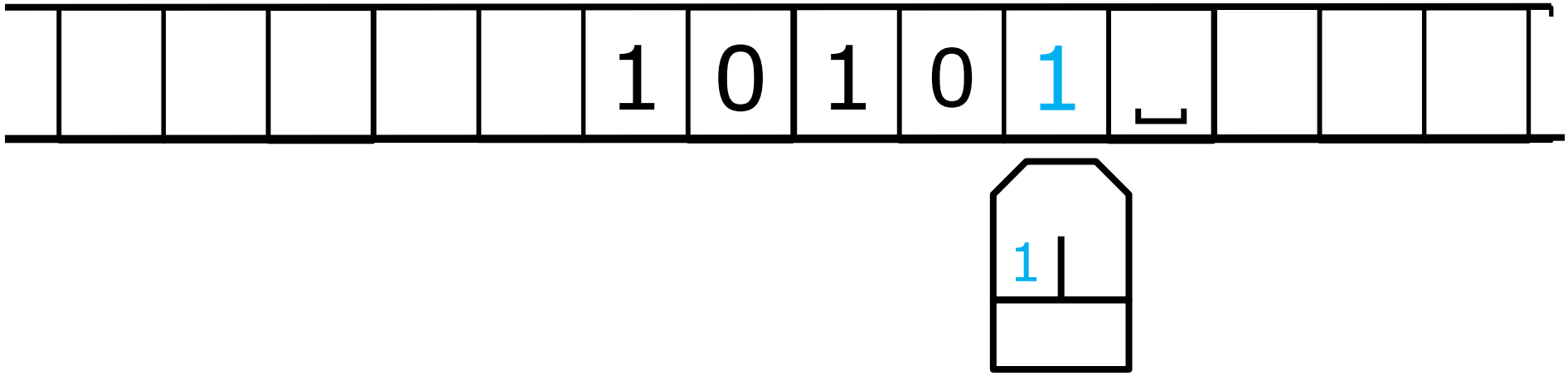
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



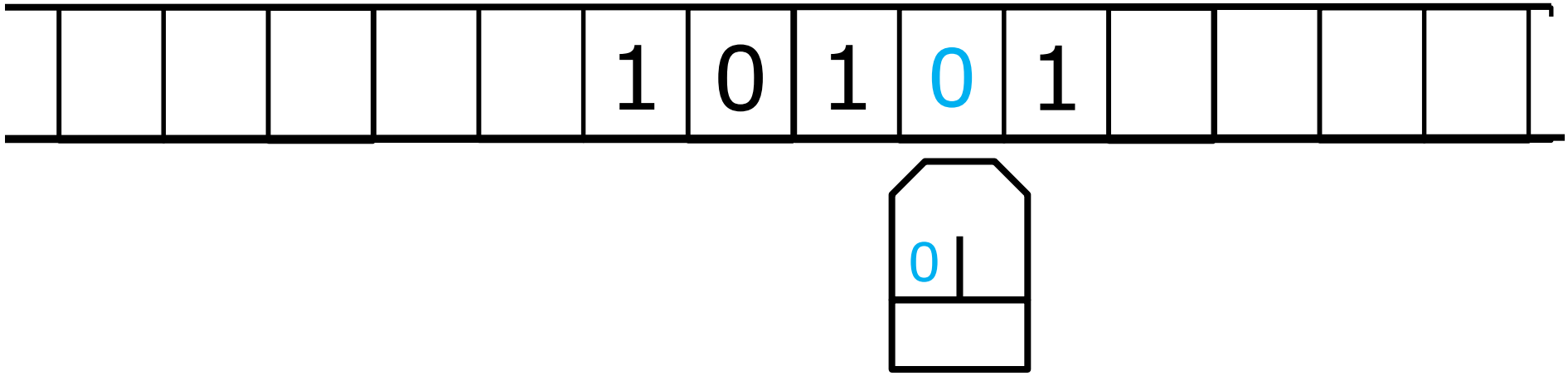
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



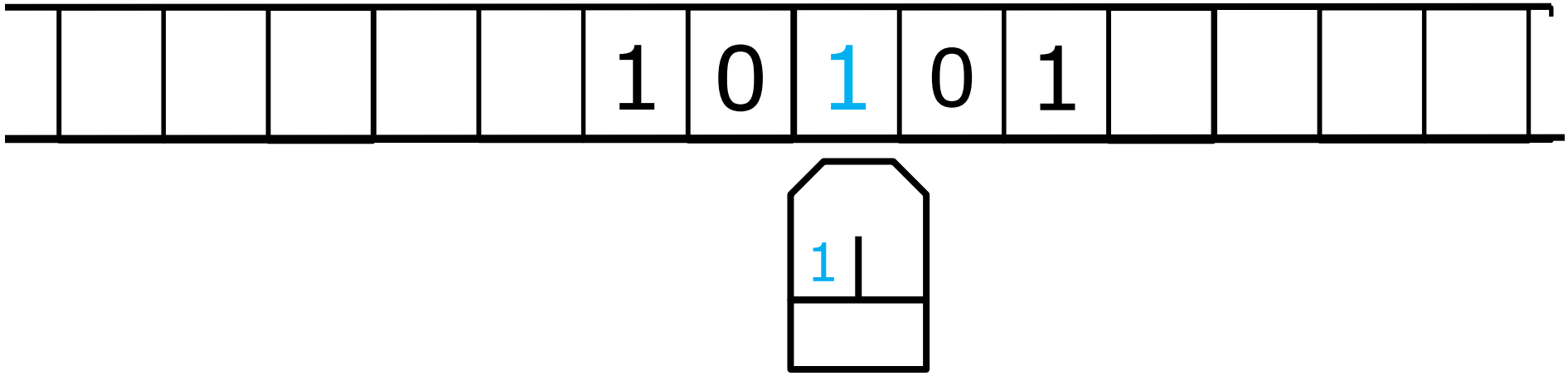
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



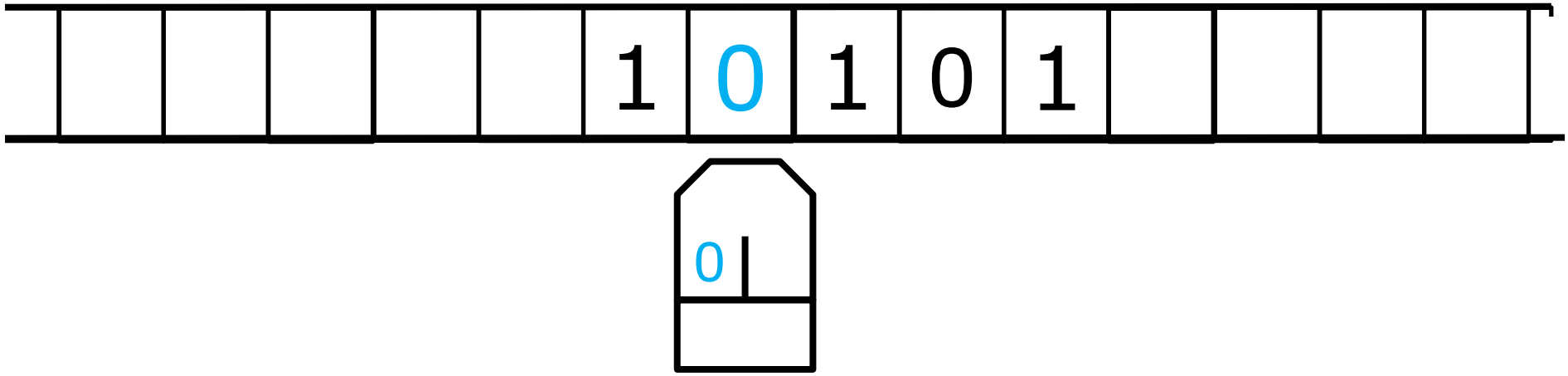
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



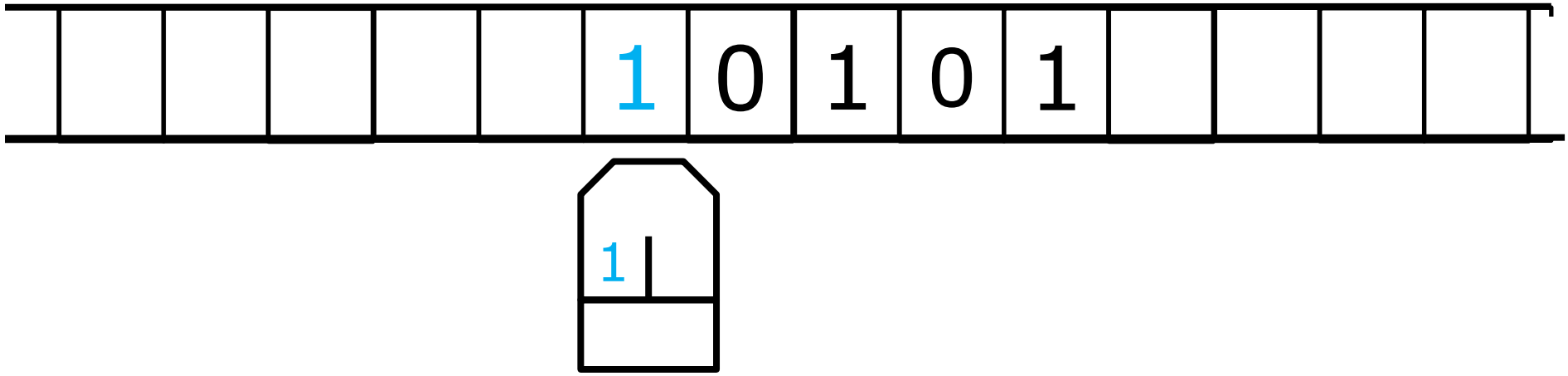
ヘッドの役割（読み取り）

- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



ヘッドの役割（読み取り）

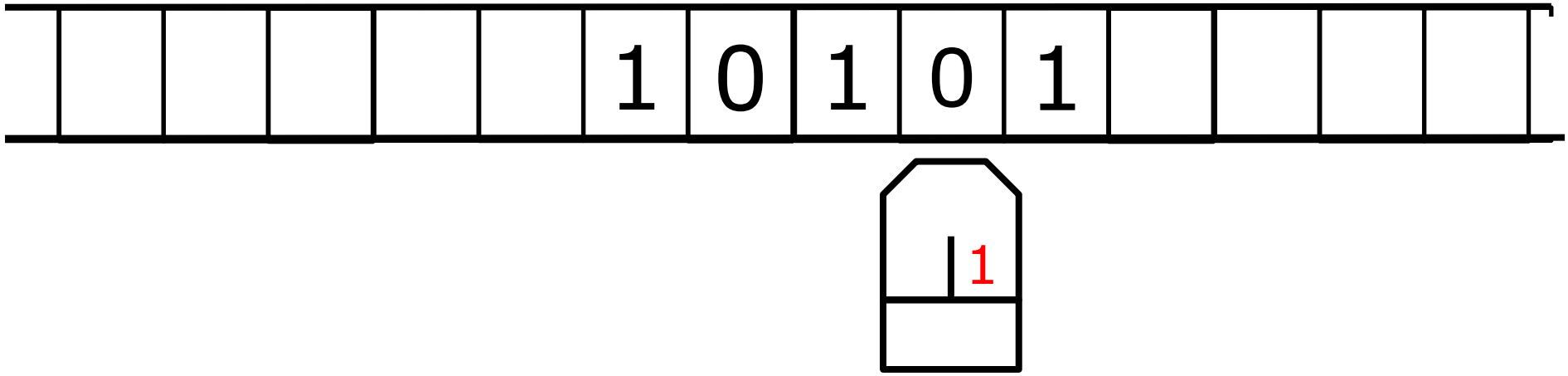
- ヘッドは、ヘッドが置かれたテープのマス目の文字を、一文字読み取る。



ヘッドの役割（書き込み）

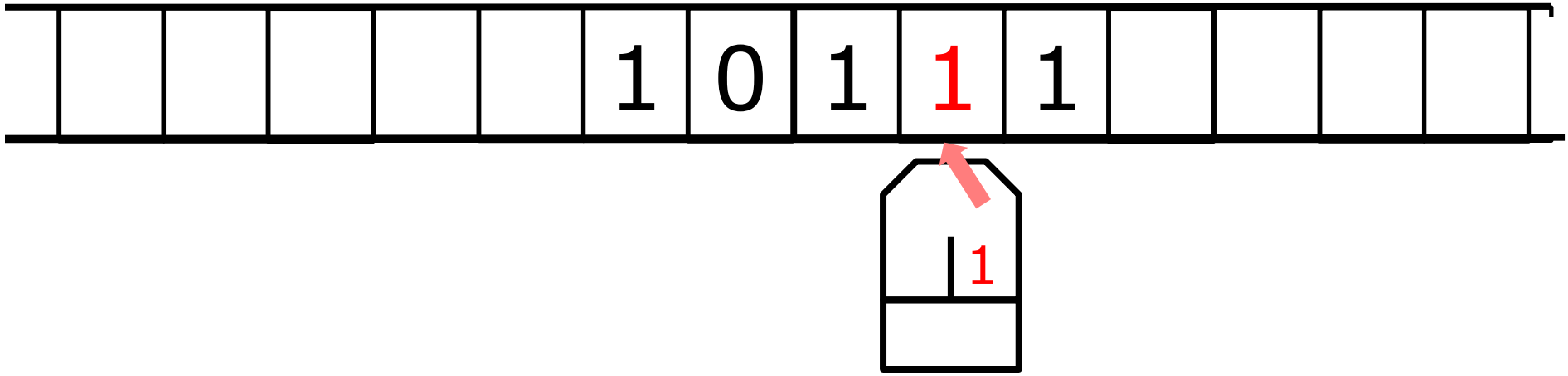
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



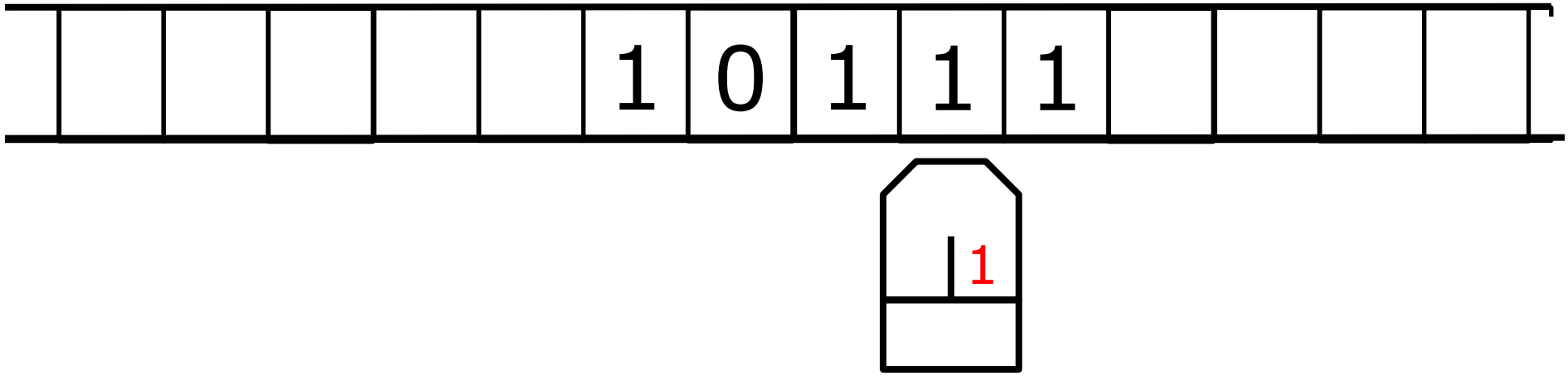
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



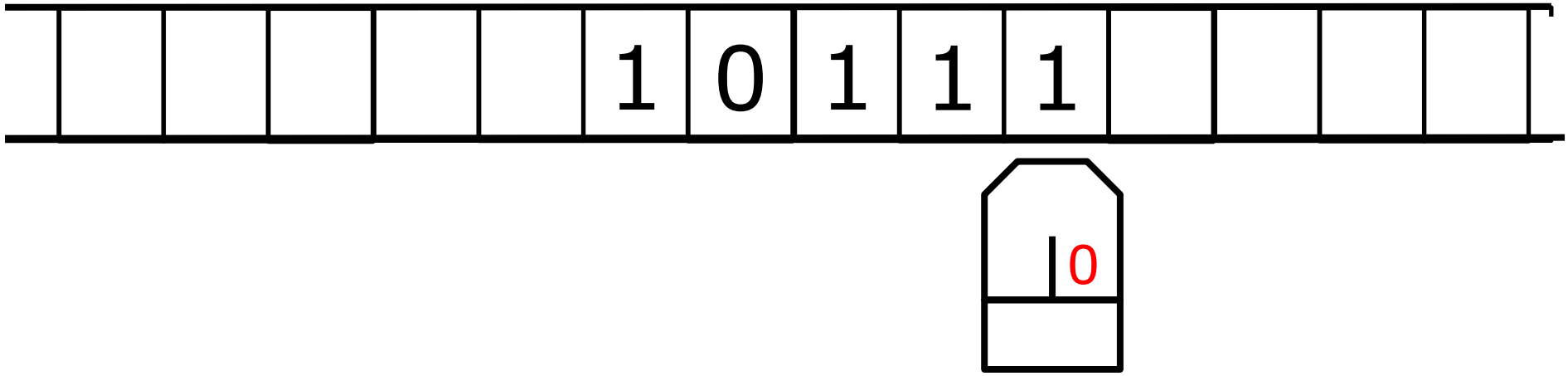
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



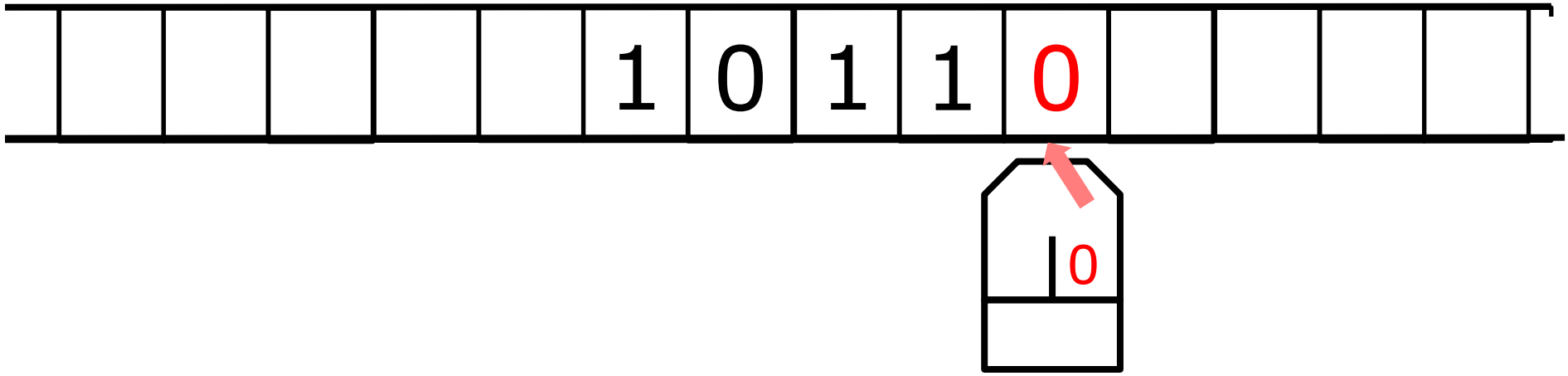
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



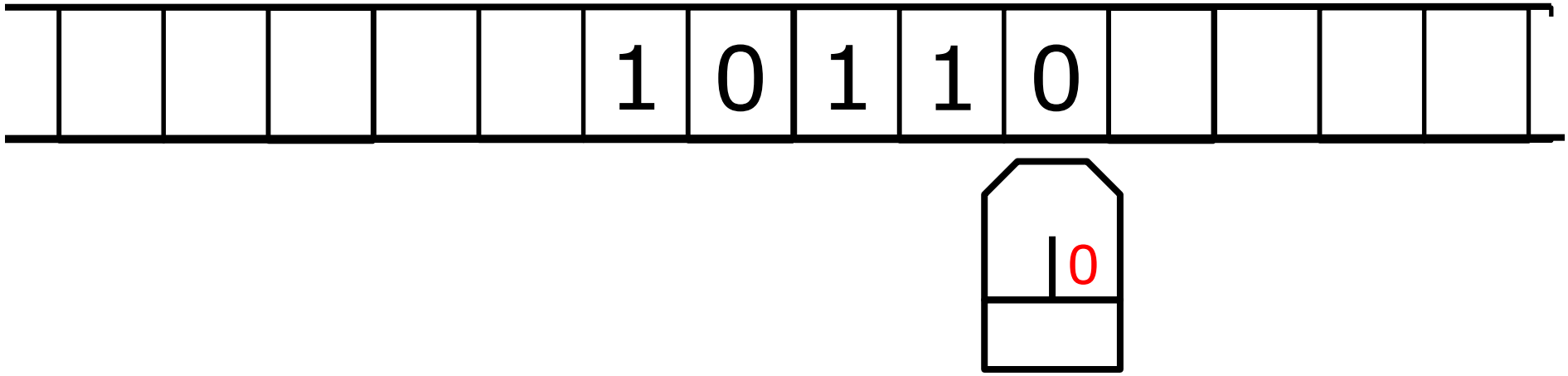
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



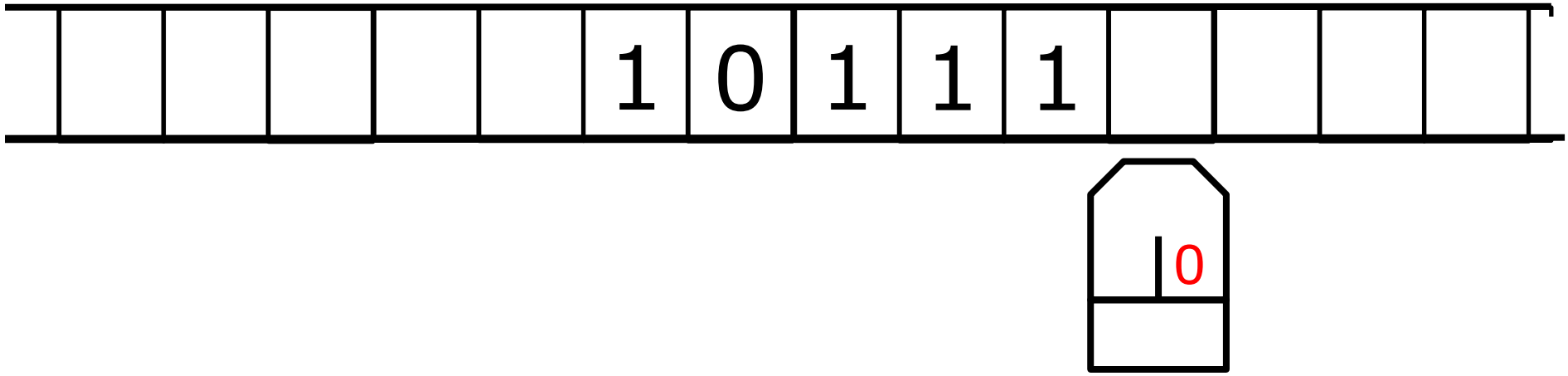
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



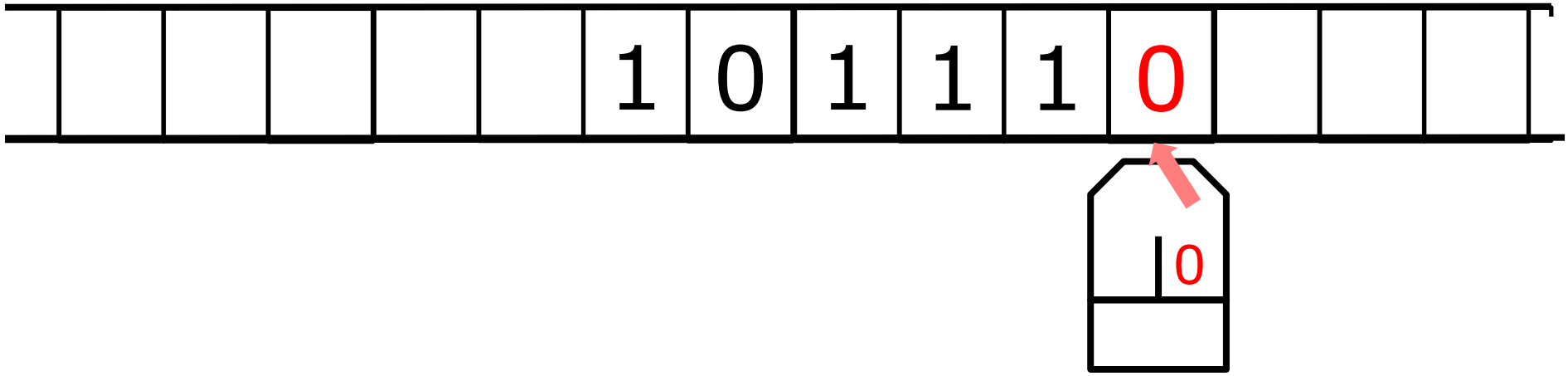
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



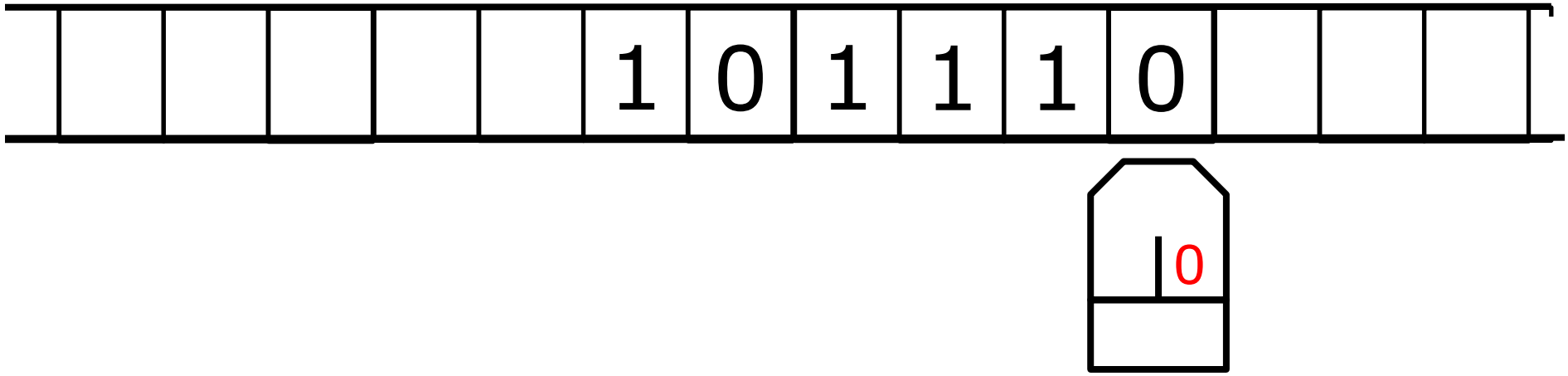
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



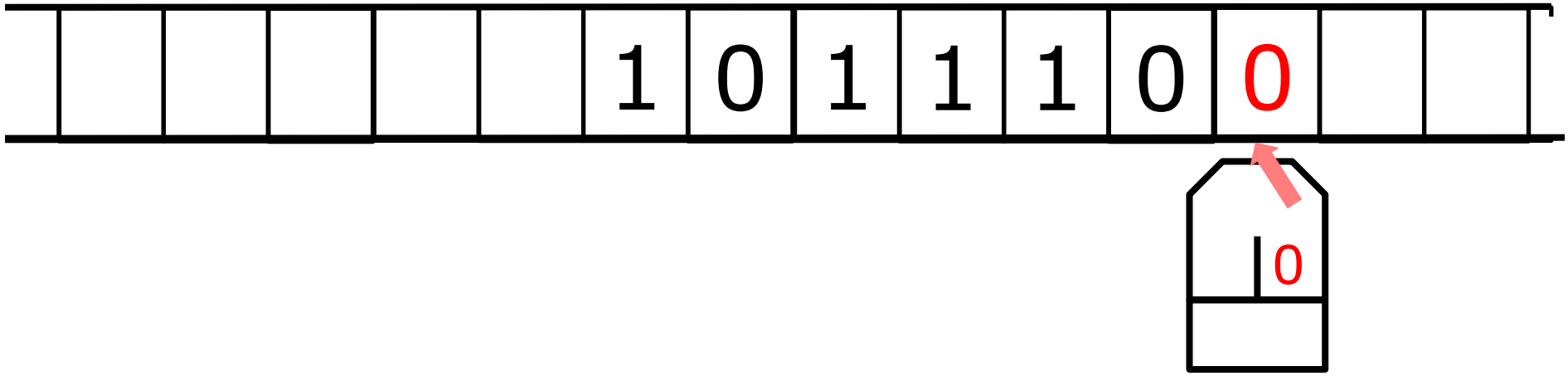
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



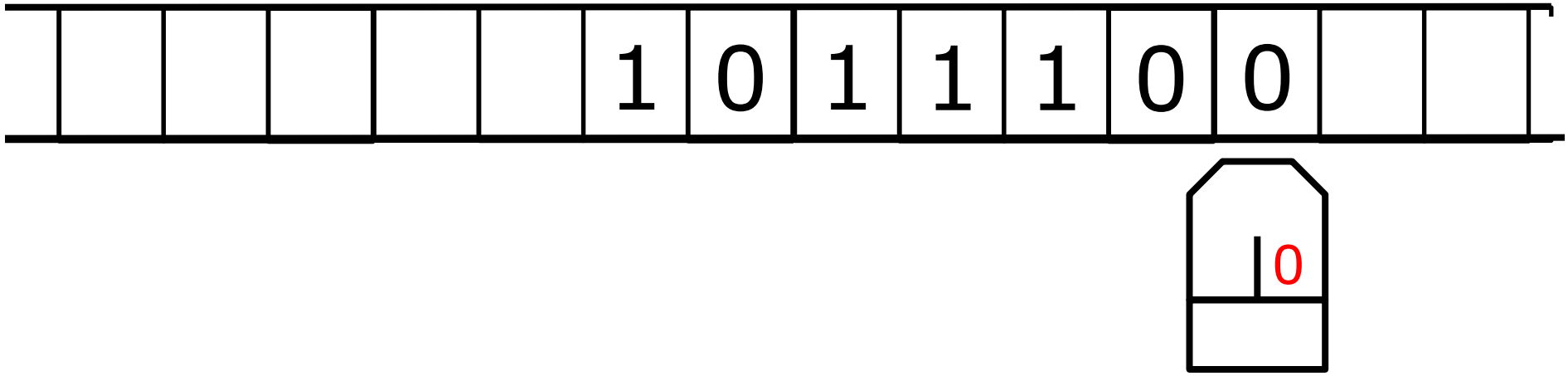
ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。

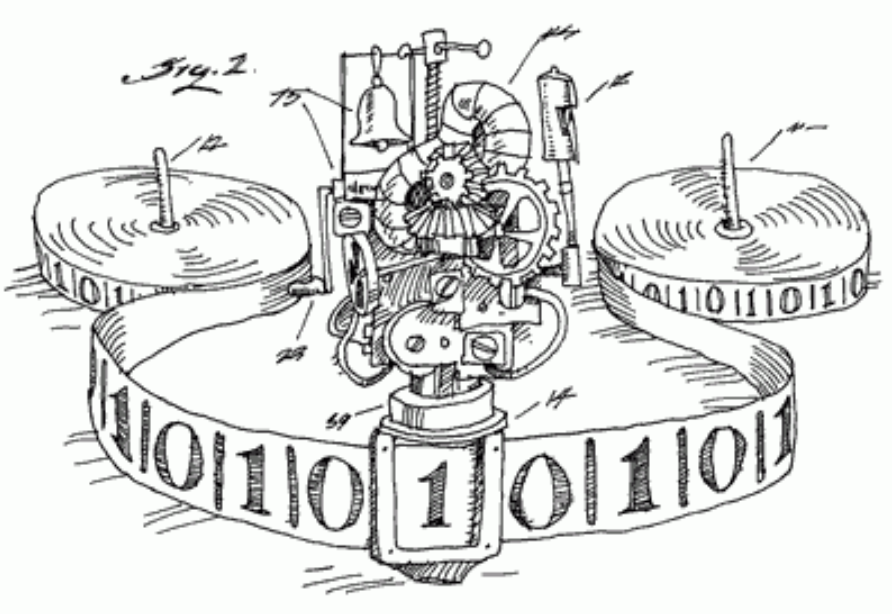


ヘッドの役割（書き込み）

- ヘッドは、ヘッドが置かれたテープのマス目に、文字を一文字書き込む。



状態と状態の遷移



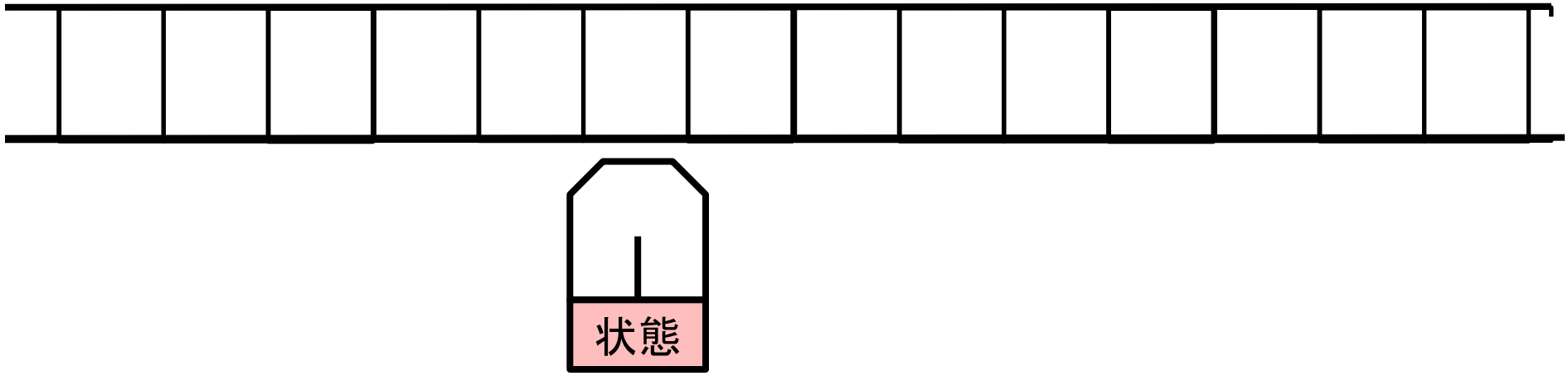
「チューリングマシンを学ぼう！」

Part Ⅰ　チューリングマシンの基本

チューリングマシンの状態

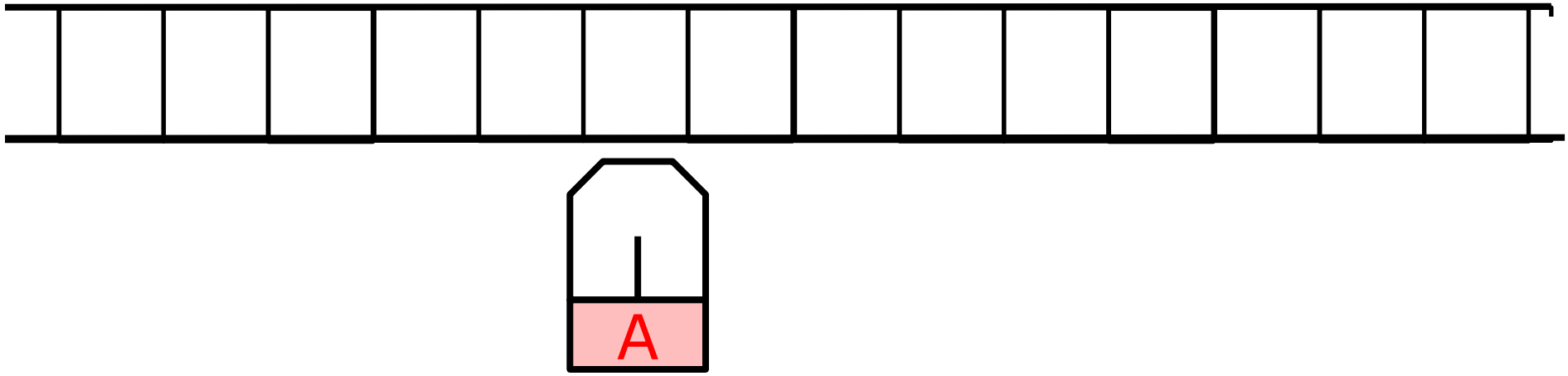
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。



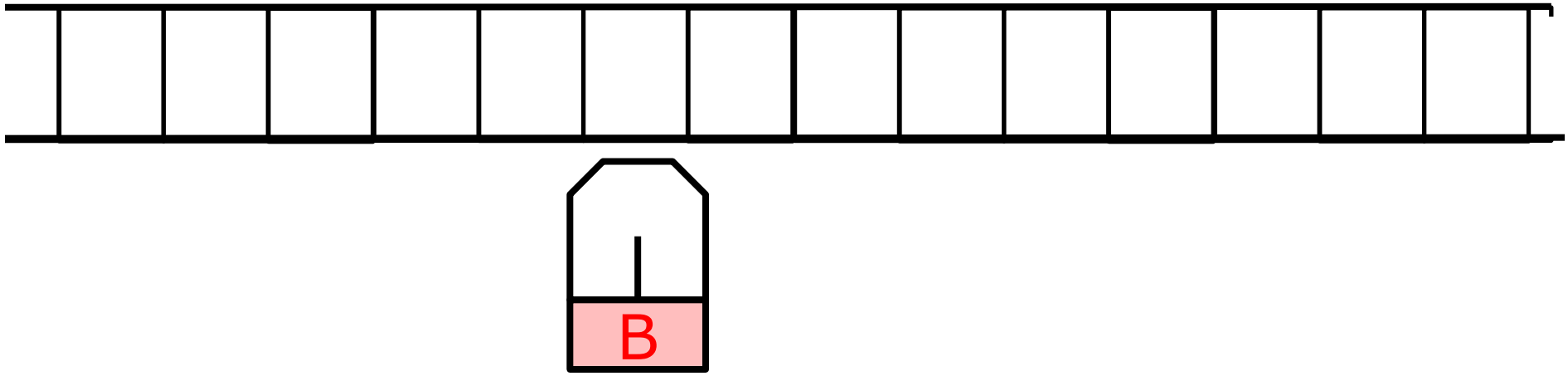
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'A'にあることを表している。



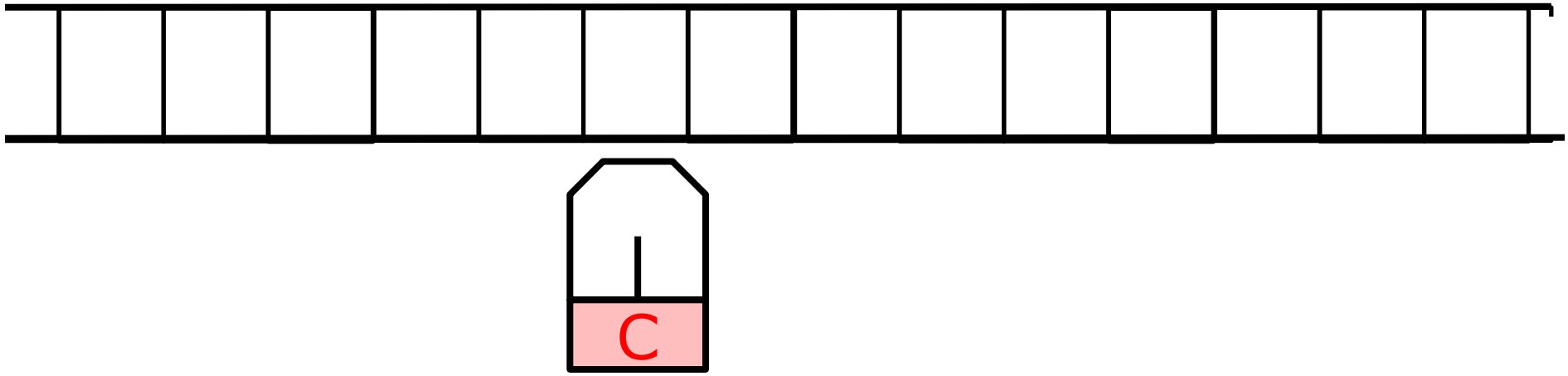
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'B'にあることを表している。



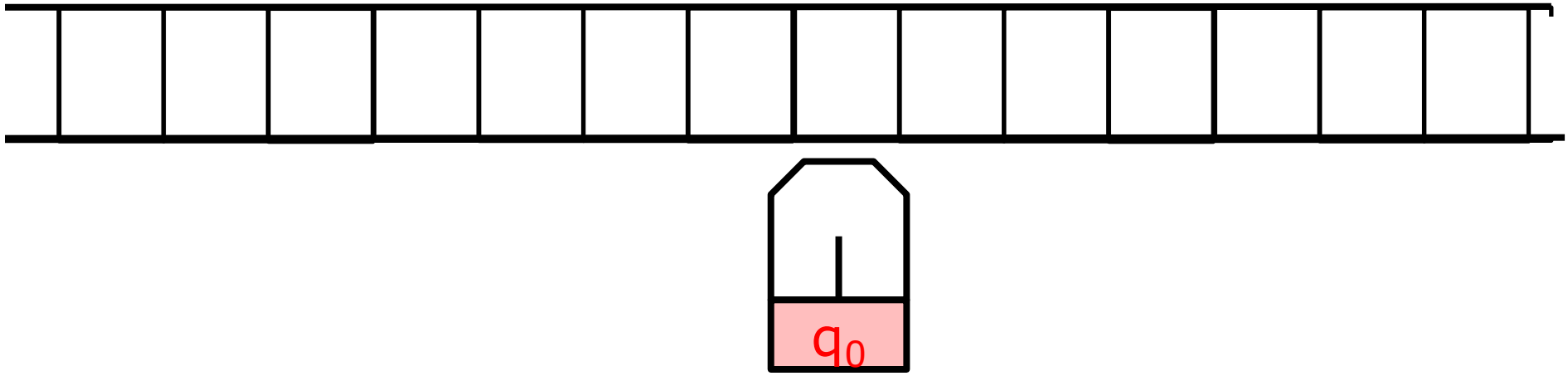
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'C'にあることを表している。



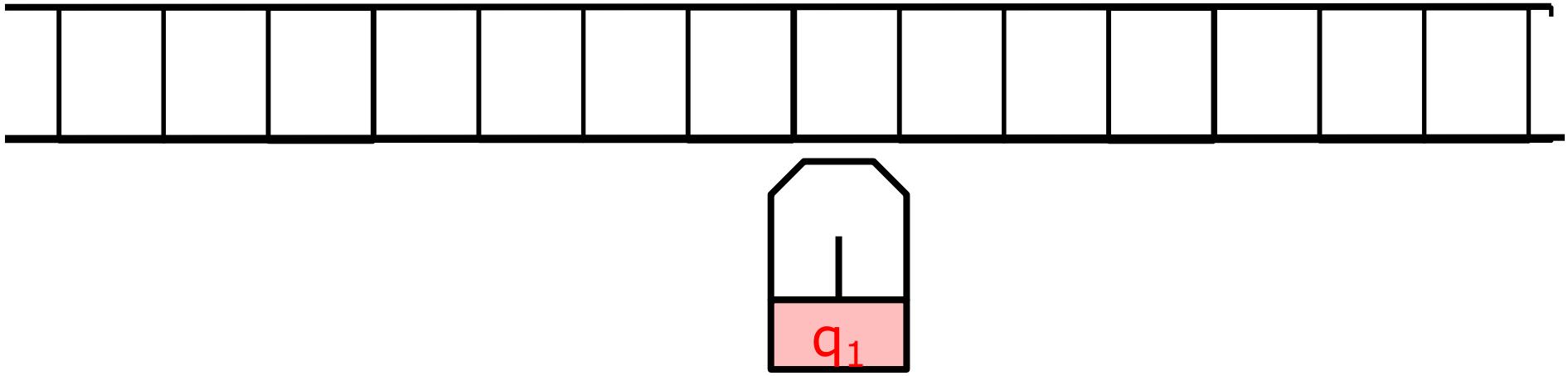
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'q_0'にあることを表している。



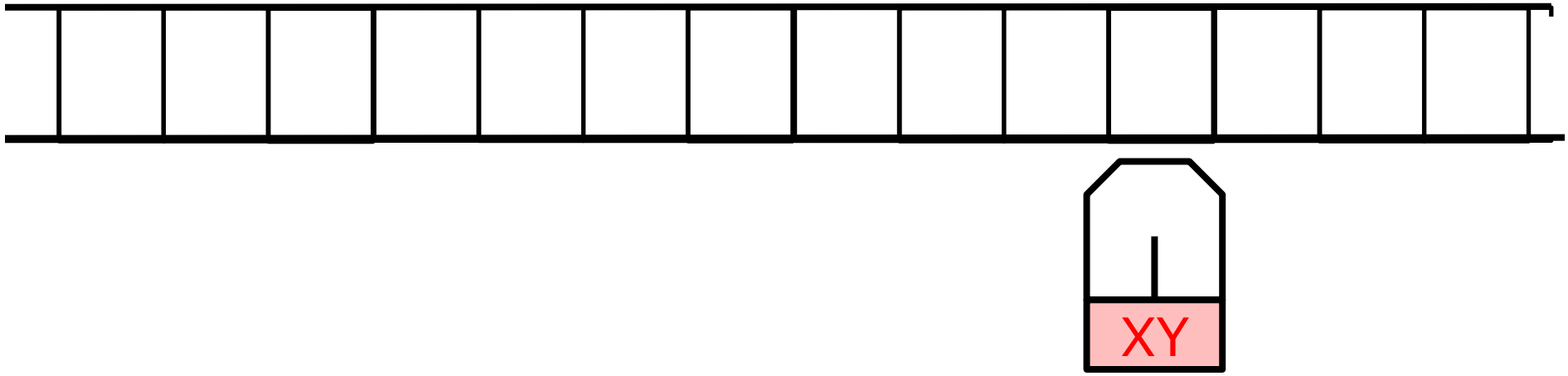
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'q_1'にあることを表している。



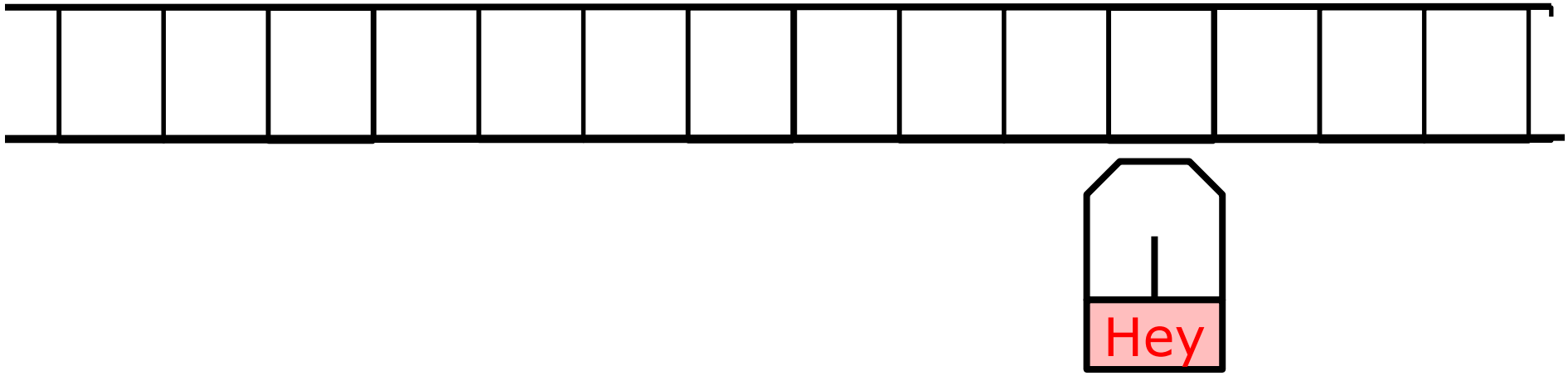
チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'XY'にあることを表している。



チューリングマシンの状態

- チューリングマシンは、「状態」を持つ。
- 図は、チューリングマシンが、状態'Hey'にあることを表している。



テープの状態とヘッドの位置

チューリングマシンの状態

-- テープの状態とヘッドの位置 --

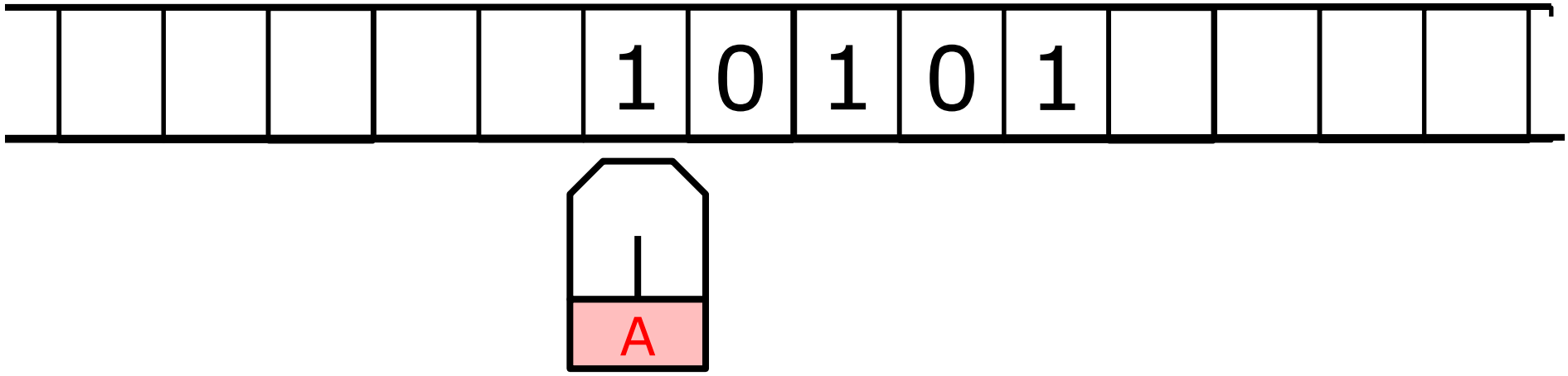
- チューリングマシンの状態は、大雑把にいうと、テープの状態とヘッドの位置で決まる。

マシンとテープとヘッドの初期状態

- チューリングマシンの状態は、大雑把にいうと、テープの状態とヘッドの位置で決まる。
- まず最初に、チューリングマシンの初期状態と、テープの初期状態（テープ上に一マスに一文字の文字が、複数のマス目に文字列として置かれている）と、ヘッドの最初の位置が与えられる。

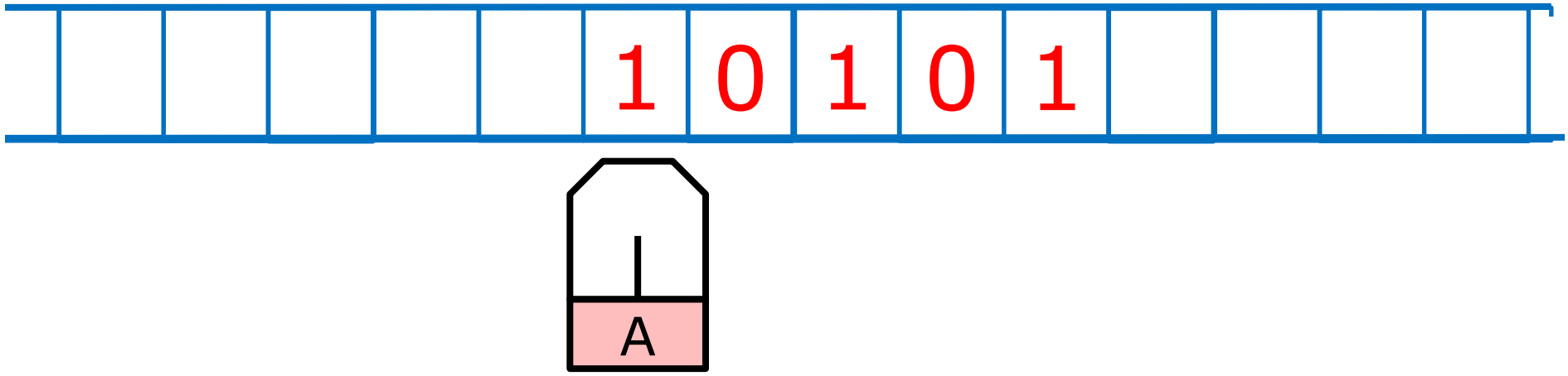
マシンとテープとヘッドの初期状態

- この例では、
  - マシンの初期状態は Aで、



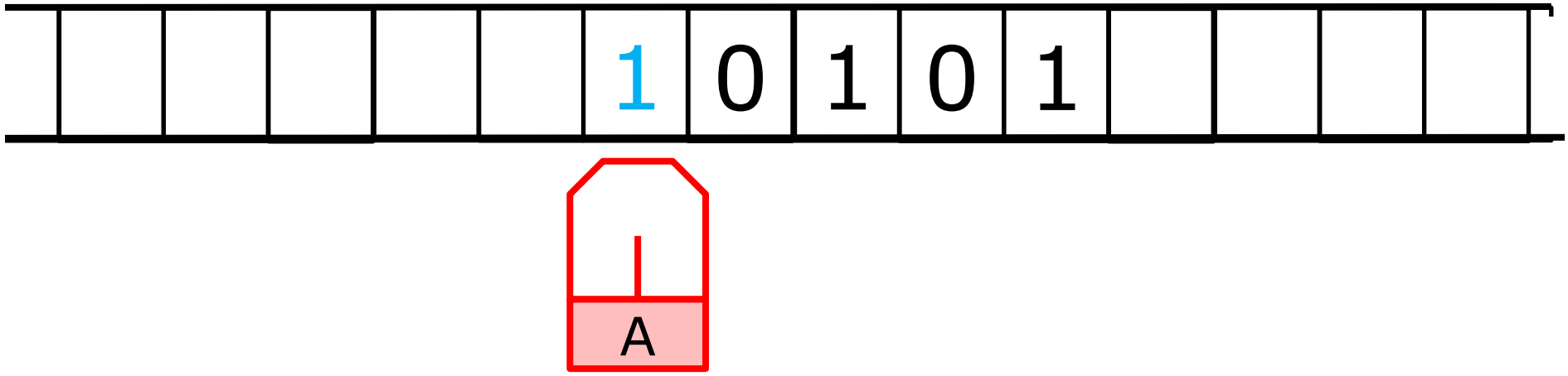
マシンとテープとヘッドの初期状態

- この例では、
  - マシンの初期状態は Aで、
  - テープの初期状態は ”10101”で、



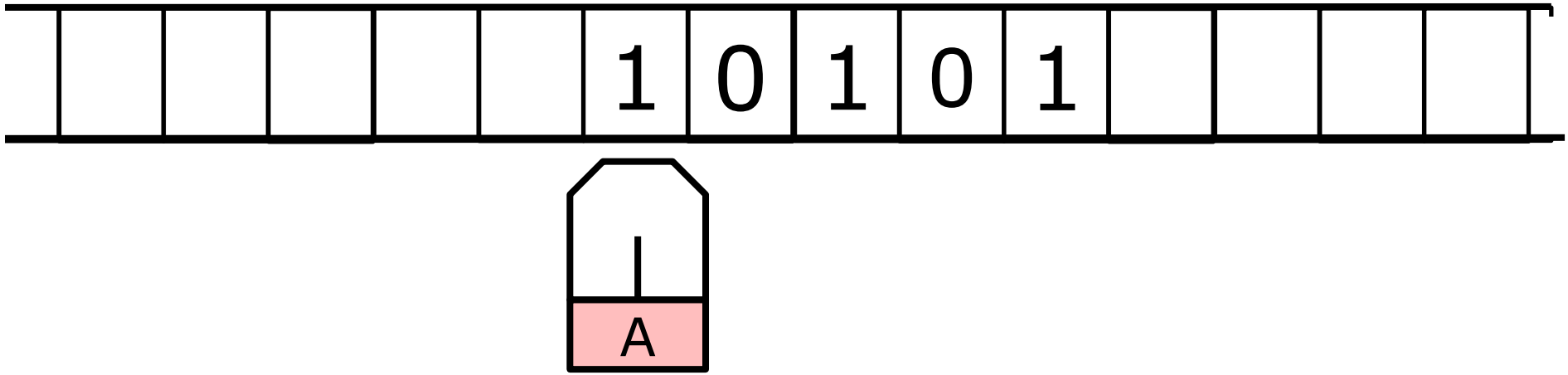
マシンとテープとヘッドの初期状態

- この例では、
  - マシンの初期状態は Aで、
  - テープの初期状態は ”10101”で、
  - ヘッドの初期状態は、”10101”の最初の”1”の位置にある。



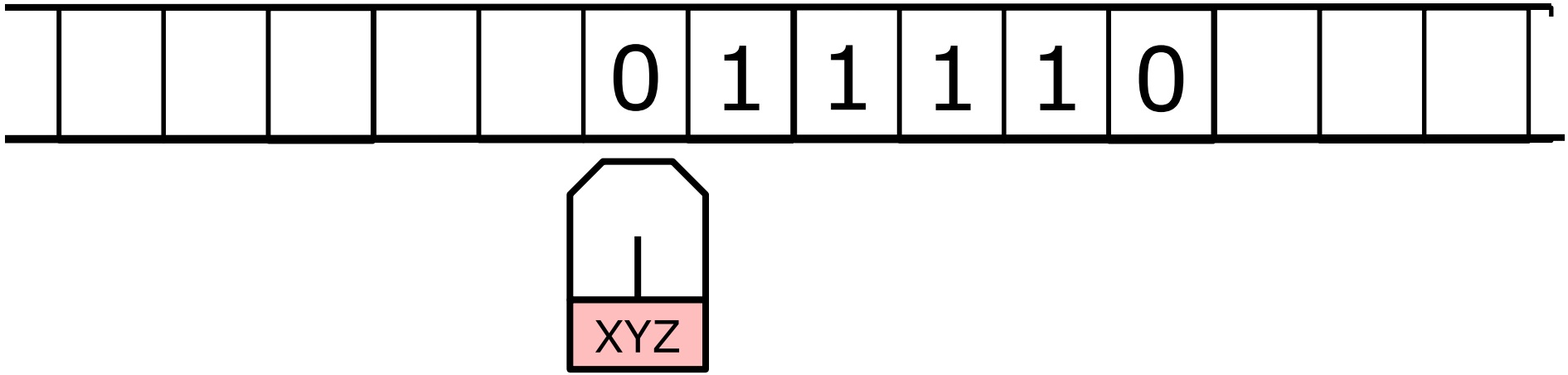
マシンとテープとヘッドの初期状態のサンプル

- この例では、
  - マシンの初期状態は Aで、
  - テープの初期状態は ”10101”で、
  - ヘッドの初期状態は、”10101”の最初の”1”の位置にある。



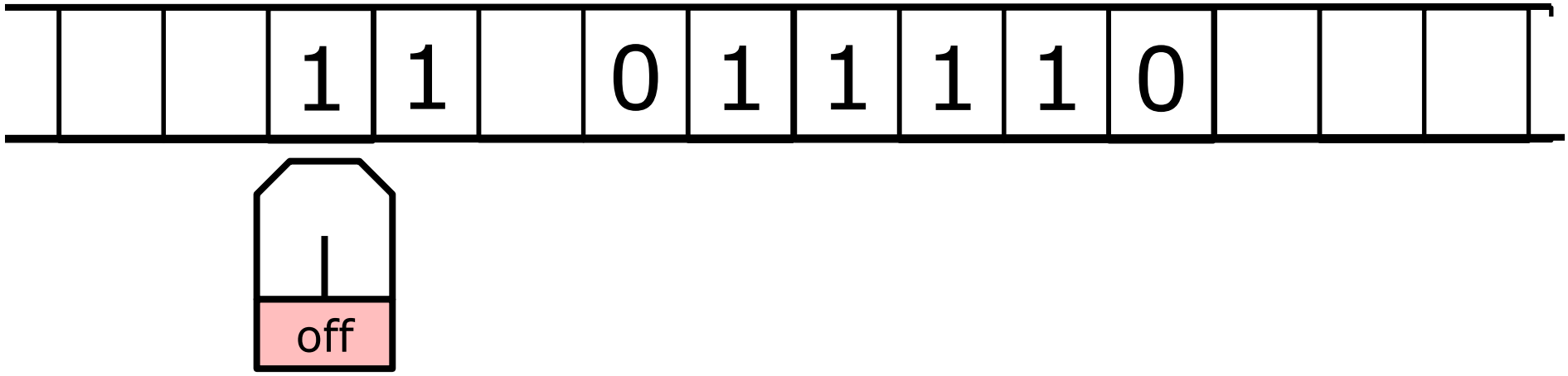
マシンとテープとヘッドの初期状態のサンプル

- この例では、
  - マシンの初期状態は XYZで、
  - テープの初期状態は ”011110”で、
  - ヘッドの初期状態は、”011110”の最初の”0”の位置にある。



マシンとテープとヘッドの初期状態のサンプル

- この例では、
  - マシンの初期状態は offで、
  - テープの初期状態は ″11␣011110″で、
  - ヘッドの初期状態は、″ 11␣″の最初の″1″の位置にある。



ヘッドの移動と
チューリングマシンの状態の遷移

チューリングマシンの状態の遷移

- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。

チューリングマシンの状態の遷移

- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。
  - 「基本的には」というのは、ヘッドが移動してもマシンの状態が変わらないこともあるからである。

チューリングマシンの状態の遷移

- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。
  - 「基本的には」というのは、ヘッドが移動してもマシンの状態が変わらないこともあるからである。
- ヘッドは、一つのステップで、右あるいは左に一マスだけ移動する。その時、基本的にはその時に限って状態の変化が起きる。

チューリングマシンの状態の遷移

- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。
  - 「基本的には」というのは、ヘッドが移動してもマシンの状態が変わらないこともあるからである。
- ヘッドは、一つのステップで、右あるいは左に一マスだけ移動する。その時、基本的にはその時に限って状態の変化が起きる。
- チューリングマシンの状態の遷移を定義するには、ヘッドの右あるいは左への移動が起きるときに、チューリングマシンの状態の変化を定義すればいい。

チューリングマシンの状態の遷移

- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。
  - 「基本的には」というのは、ヘッドが移動してもマシンの状態が変わらないこともあるからである。
- ヘッドは、一つのステップで、右あるいは左に一マスだけ移動する。その時、基本的にはその時に限って状態の変化が起きる。
- チューリングマシンの状態の遷移を定義するには、ヘッドの右あるいは左への移動が起きるときに、チューリングマシンの状態の変化を定義すればいい。
- こうして定義された状態は、次のステップ、すなわち次の位置にヘッドが移動した時のチューリングマシンの状態である。

チューリングマシンの状態の遷移

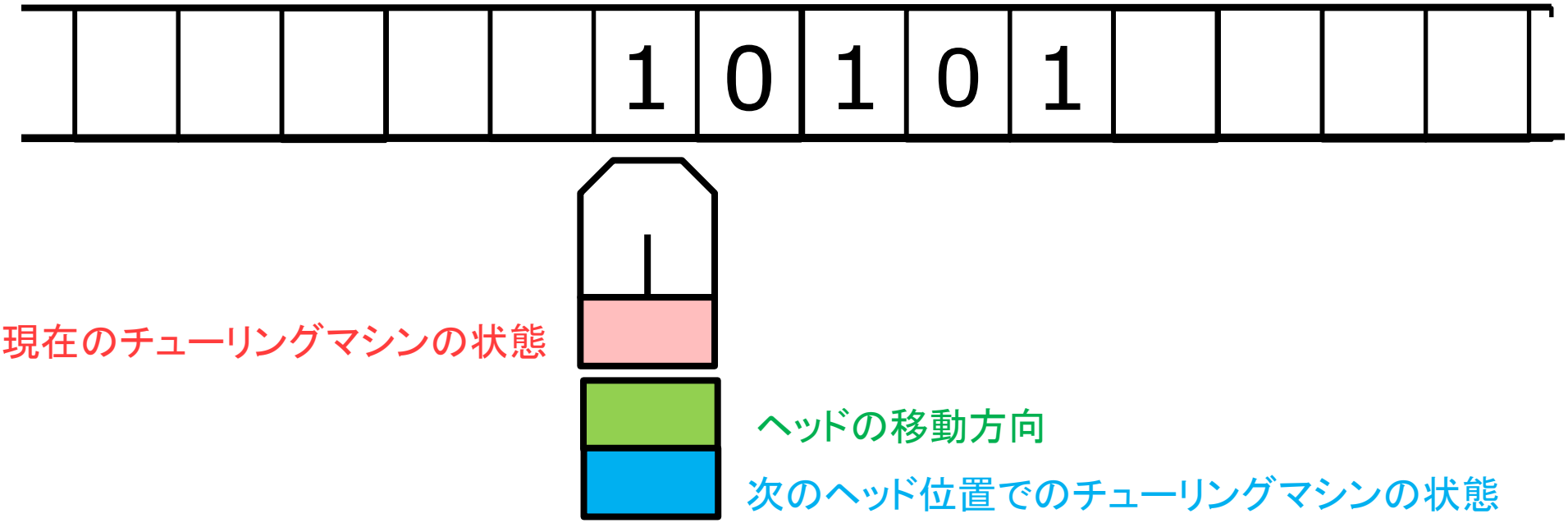
- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。
  - 「基本的には」というのは、ヘッドが移動してもマシンの状態が変わらないこともあるからである。
- ヘッドは、一つのステップで、右あるいは左に一マスだけ移動する。その時、基本的にはその時に限って状態の変化が起きる。
- チューリングマシンの状態の遷移を定義するには、ヘッドの右あるいは左への移動が起きるときに、チューリングマシンの状態の変化を定義すればいい。
- こうして定義された状態は、次のステップ、すなわち次の位置にヘッドが移動した時のチューリングマシンの状態である。
  - どのような条件の時にヘッドの移動が起きるかは、後で述べる。

チューリングマシンの状態の遷移

- チューリングマシンの状態の変化は、基本的には、ヘッドの移動で起こる。
  - 「基本的には」というのは、ヘッドが移動してもマシンの状態が変わらないこともあるからである。
- ヘッドは、一つのステップで、右あるいは左に一マスだけ移動する。その時、基本的にはその時に限って状態の変化が起きる。
- チューリングマシンの状態の遷移を定義するには、ヘッドの右あるいは左への移動が起きるときに、チューリングマシンの状態の変化を定義すればいい。
- こうして定義された状態は、次のステップ、すなわち次の位置にヘッドが移動した時のチューリングマシンの状態である。
  - どのような条件の時にヘッドの移動が起きるかは、後で述べる。

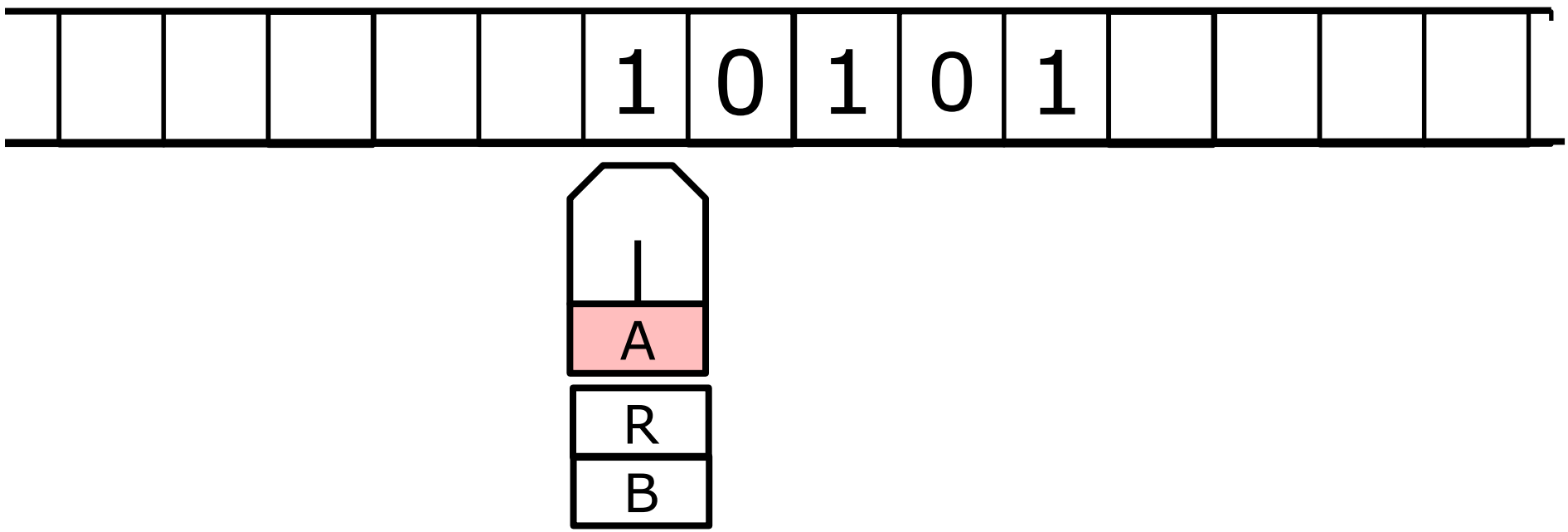
ヘッドの移動と
チューリングマシンの状態の遷移

- チューリングマシンの状態の遷移を見るために、これまでのヘッドの図の下に、ヘッドの移動方向と、次のヘッド位置でのチューリングマシンの状態を書き込んでみよう。



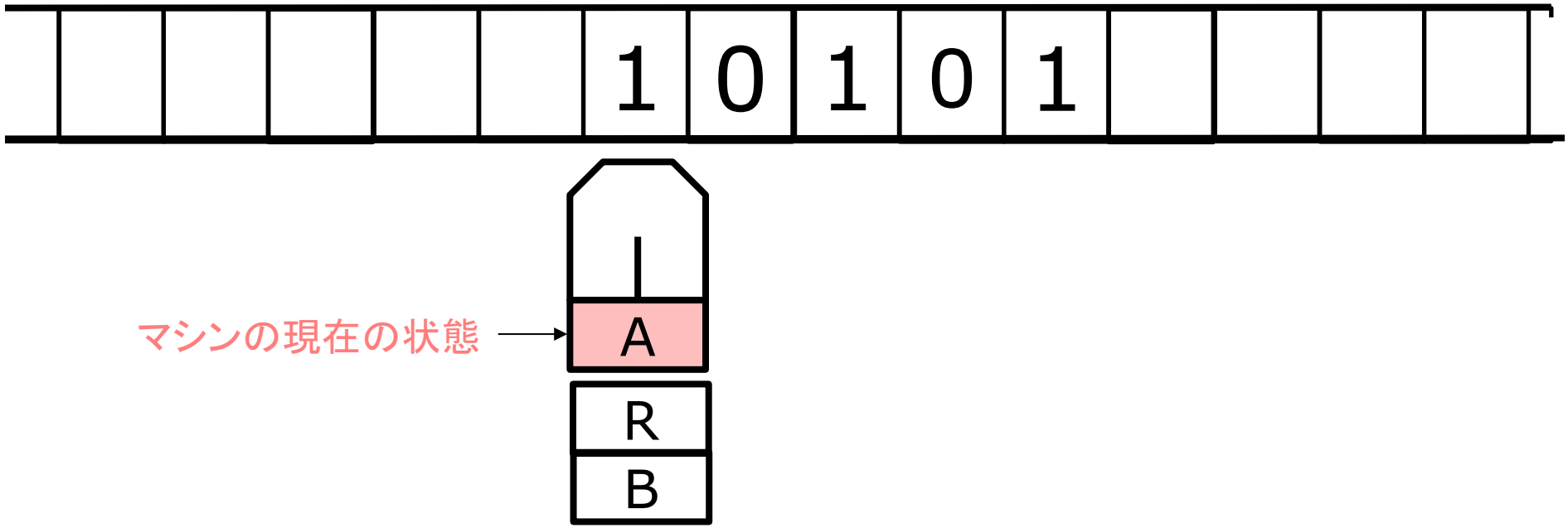
ヘッドの移動と
チューリングマシンの状態の遷移

- この例は、



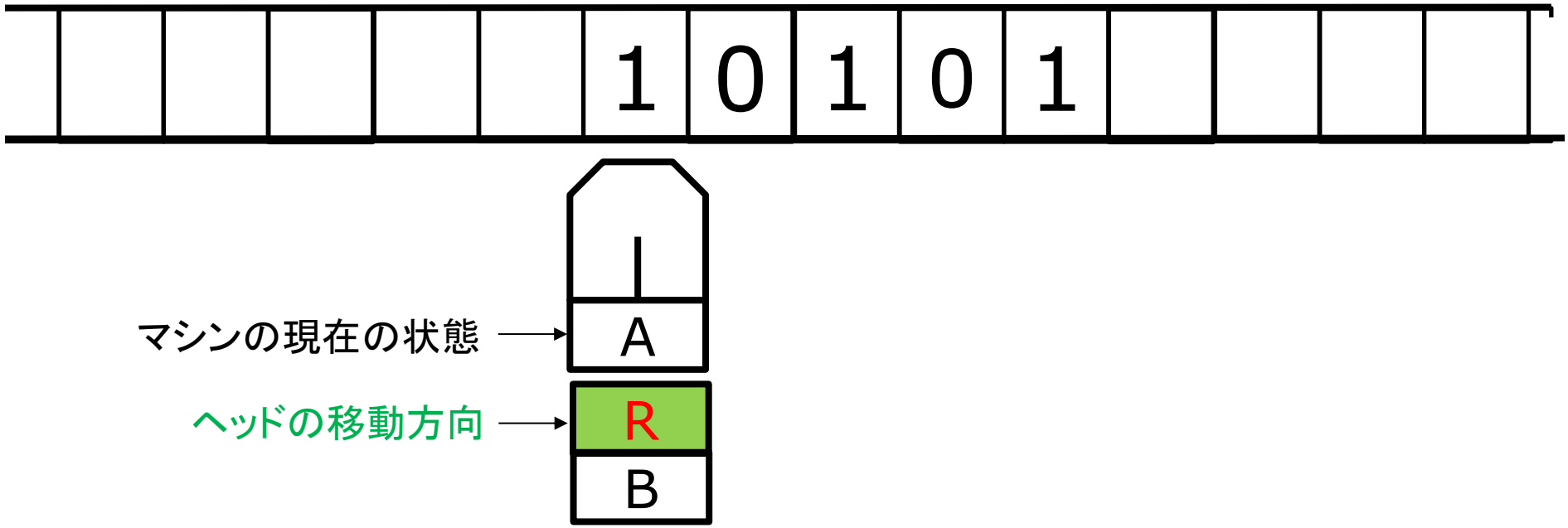
ヘッドの移動と
チューリングマシンの状態の遷移

- この例は、
  - マシンの現在の状態は Aで、



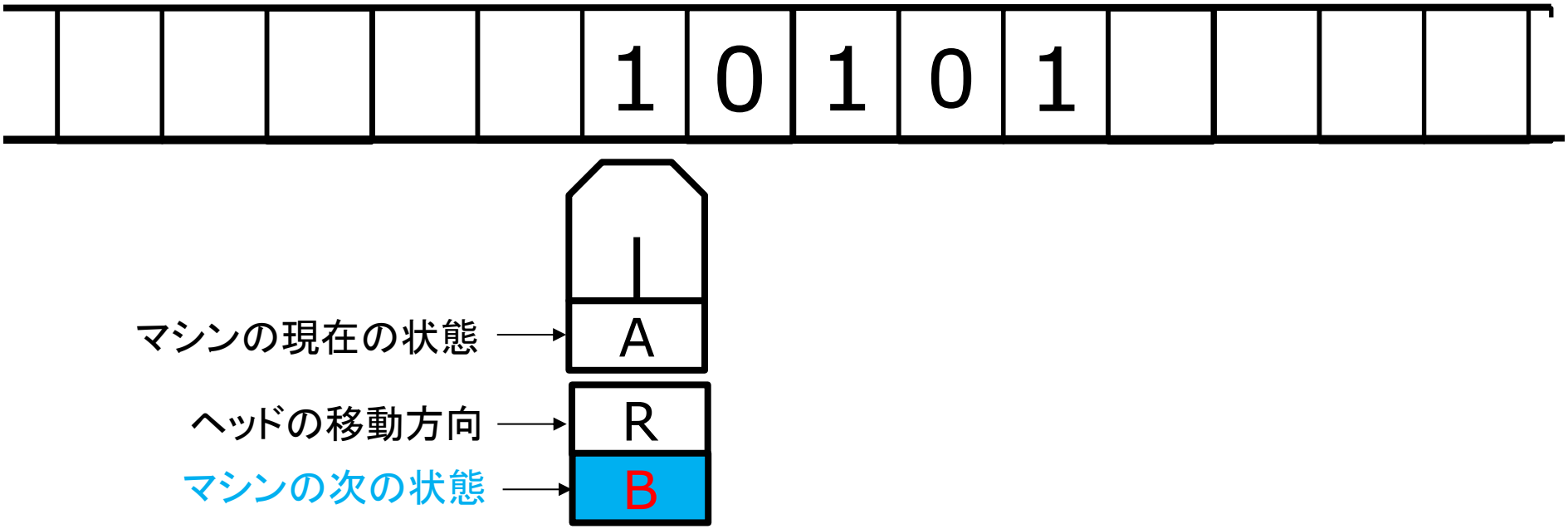
ヘッドの移動と
チューリングマシンの状態の遷移

- この例は、
  - マシンの現在の状態は Aで、
  - 次のステップでのヘッドの移動方向はR(右に一マス)” で、



ヘッドの移動と
チューリングマシンの状態の遷移

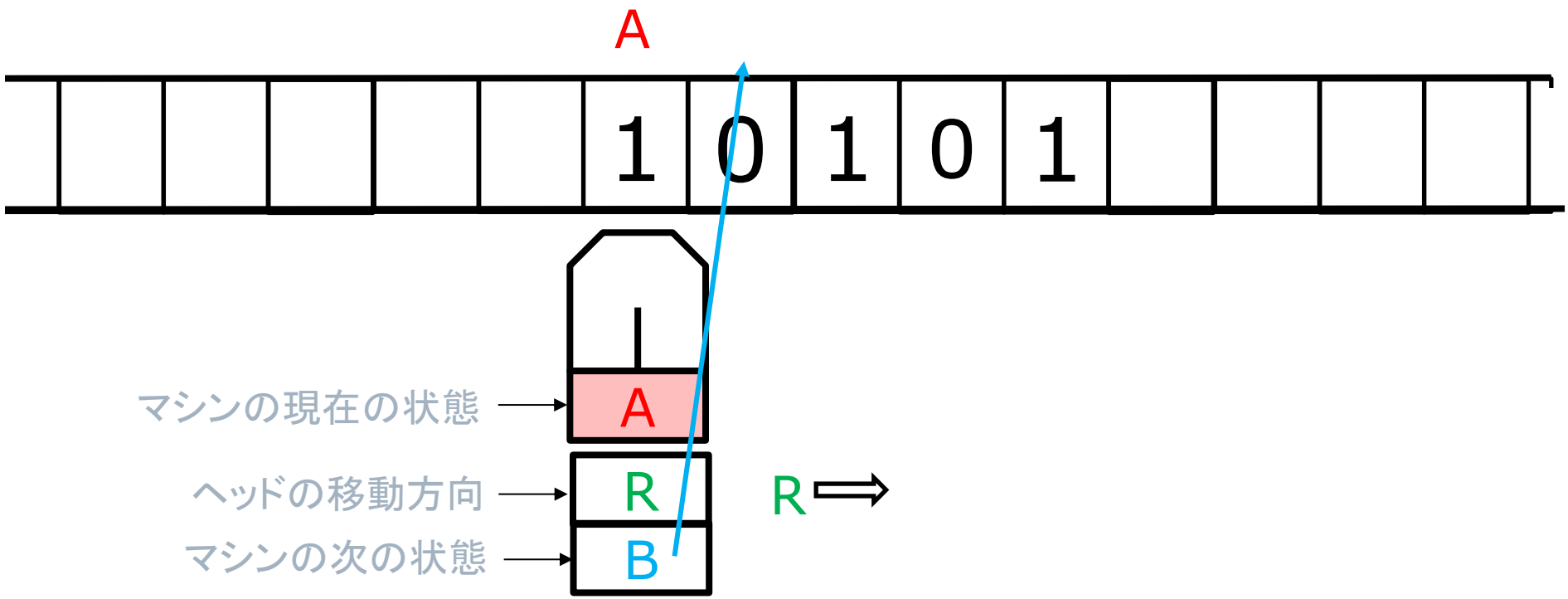
- この例は、
  - マシンの現在の状態は Aで、
  - 次のステップでのヘッドの移動方向はR(右に一マス)″ で、
  - マシンの次の状態は Bに変わること を表している。



チューリングマシンの状態の遷移のサンプル

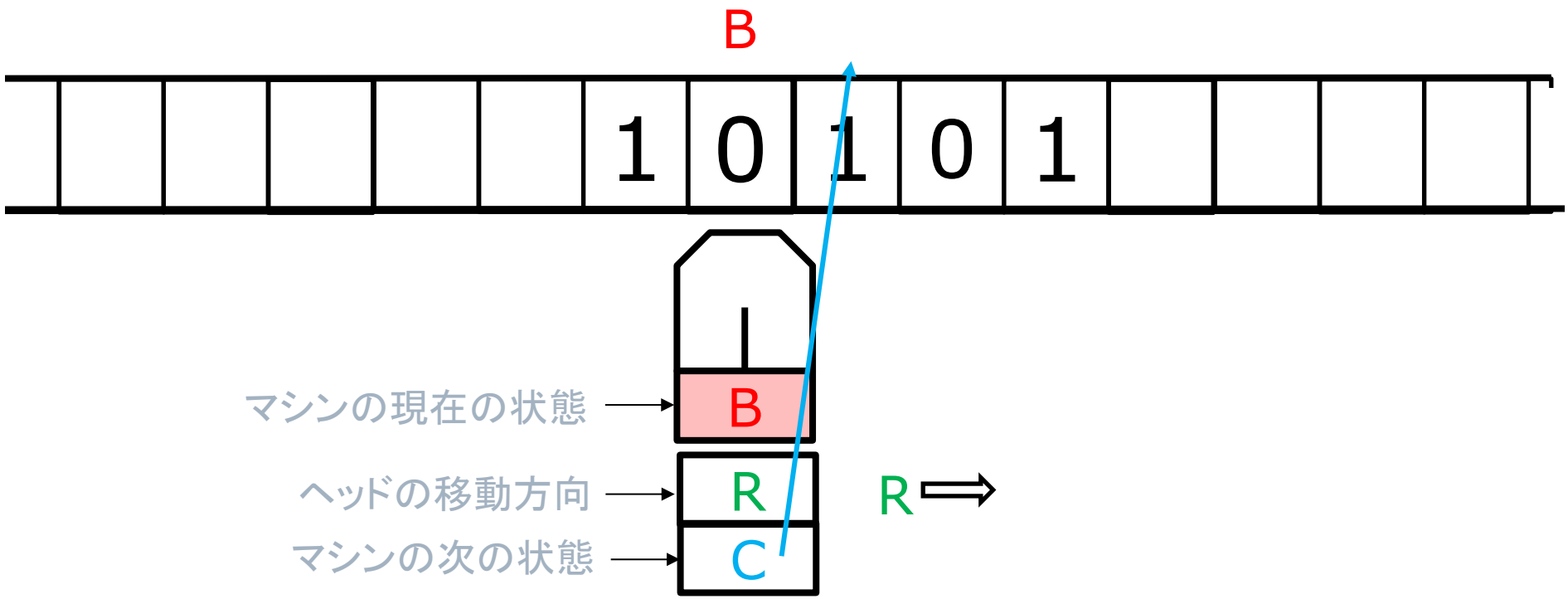
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



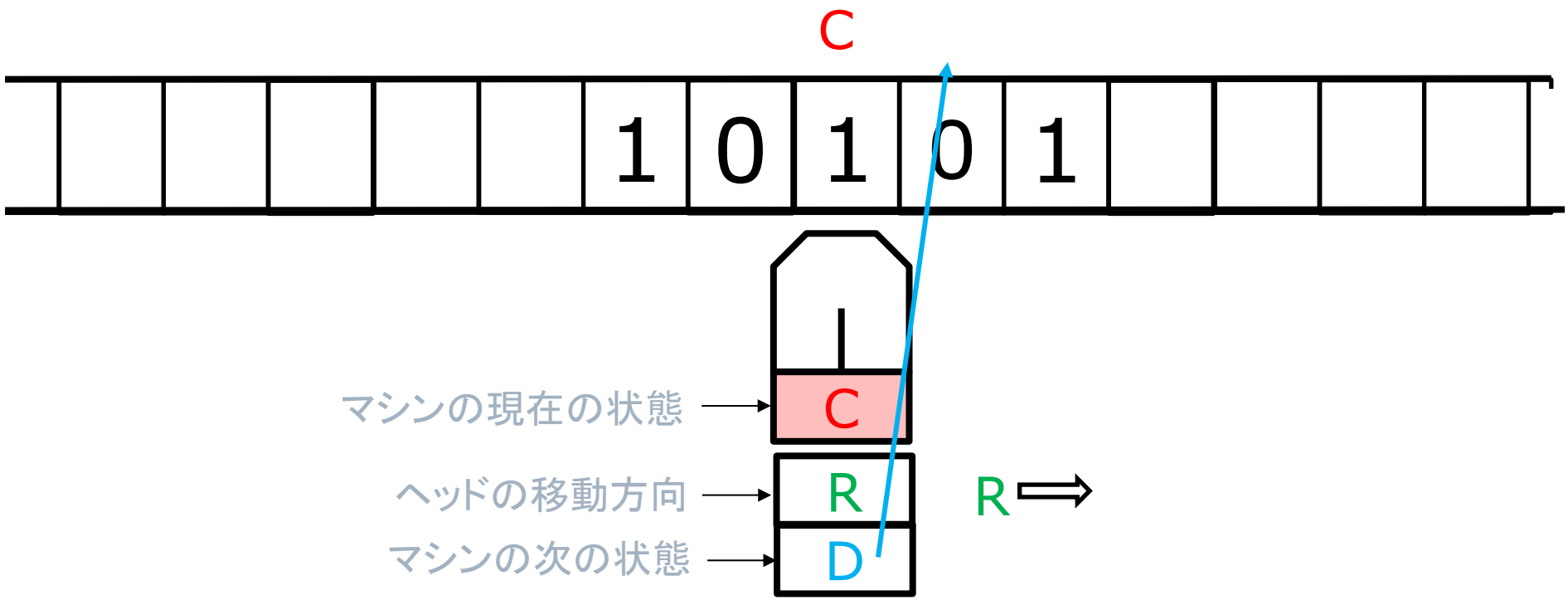
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



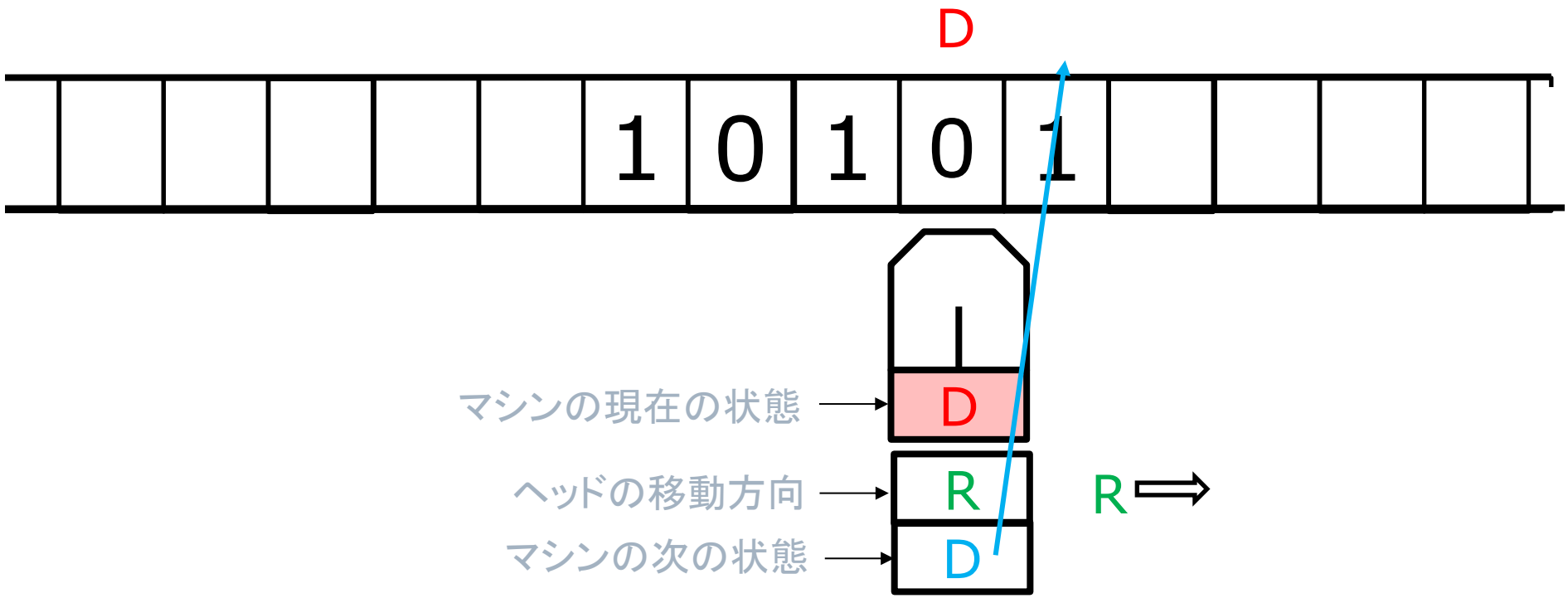
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



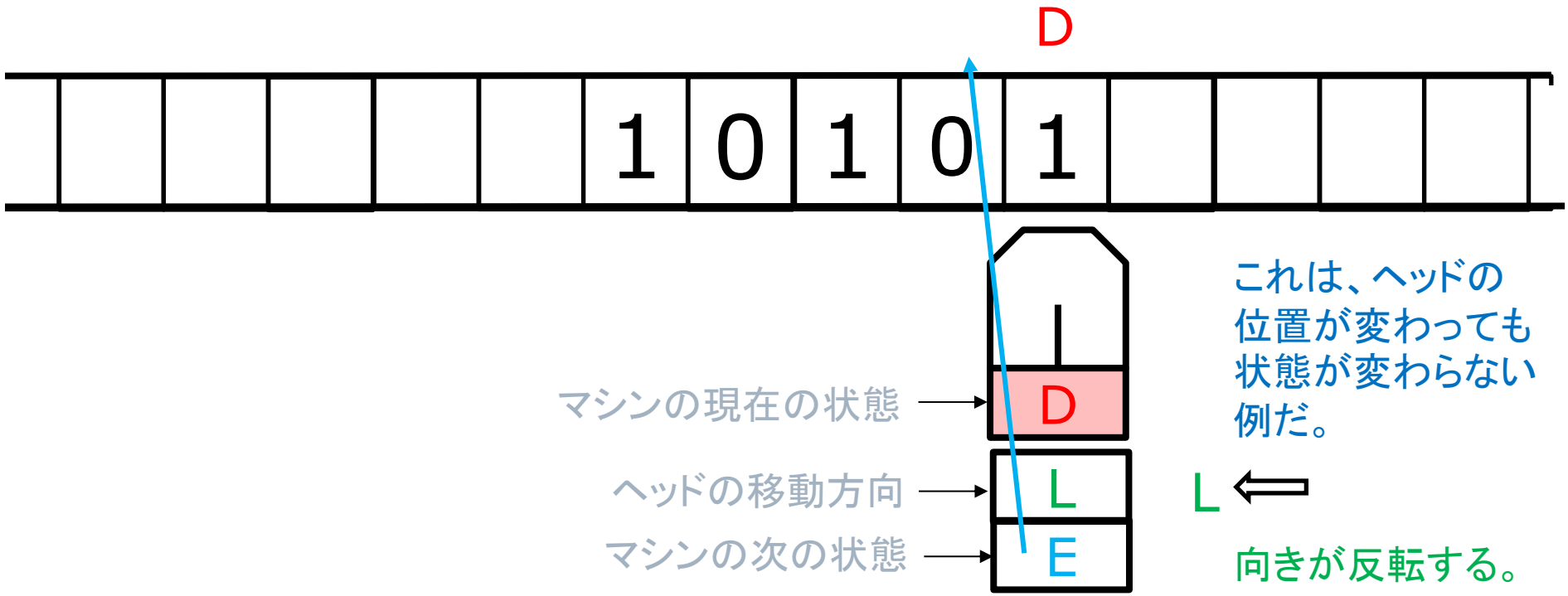
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



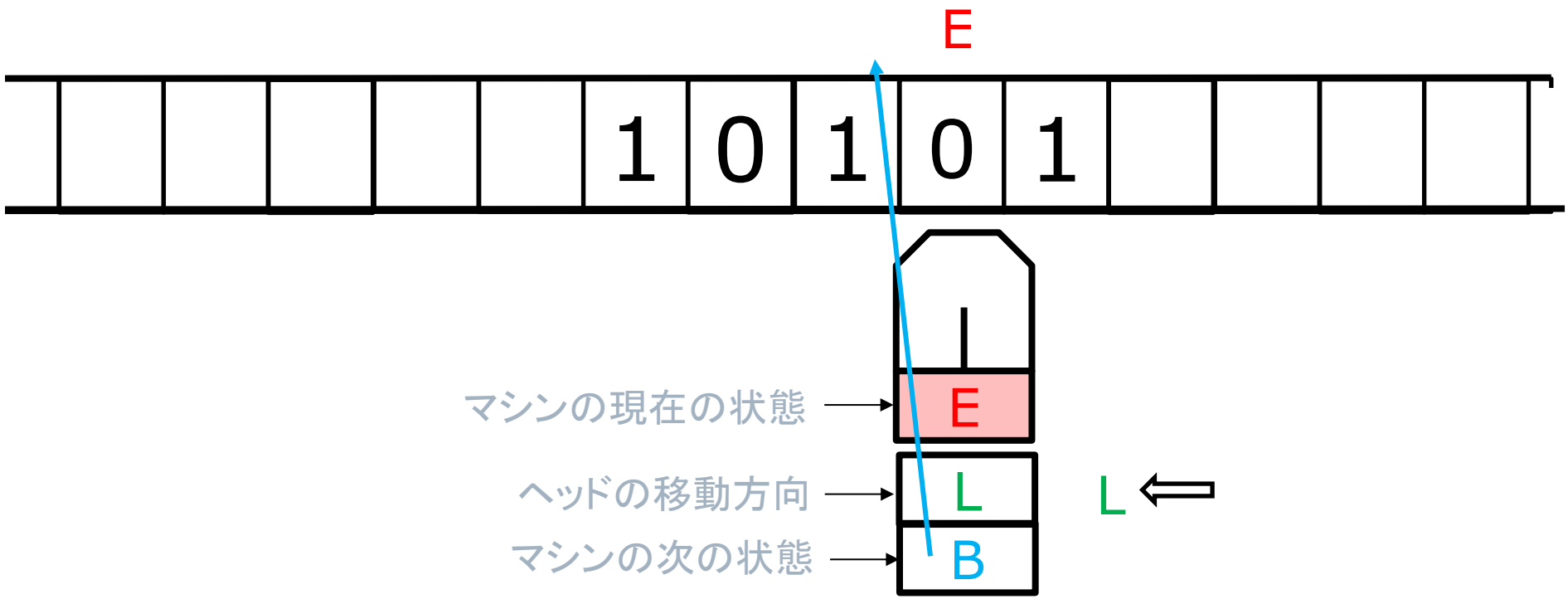
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



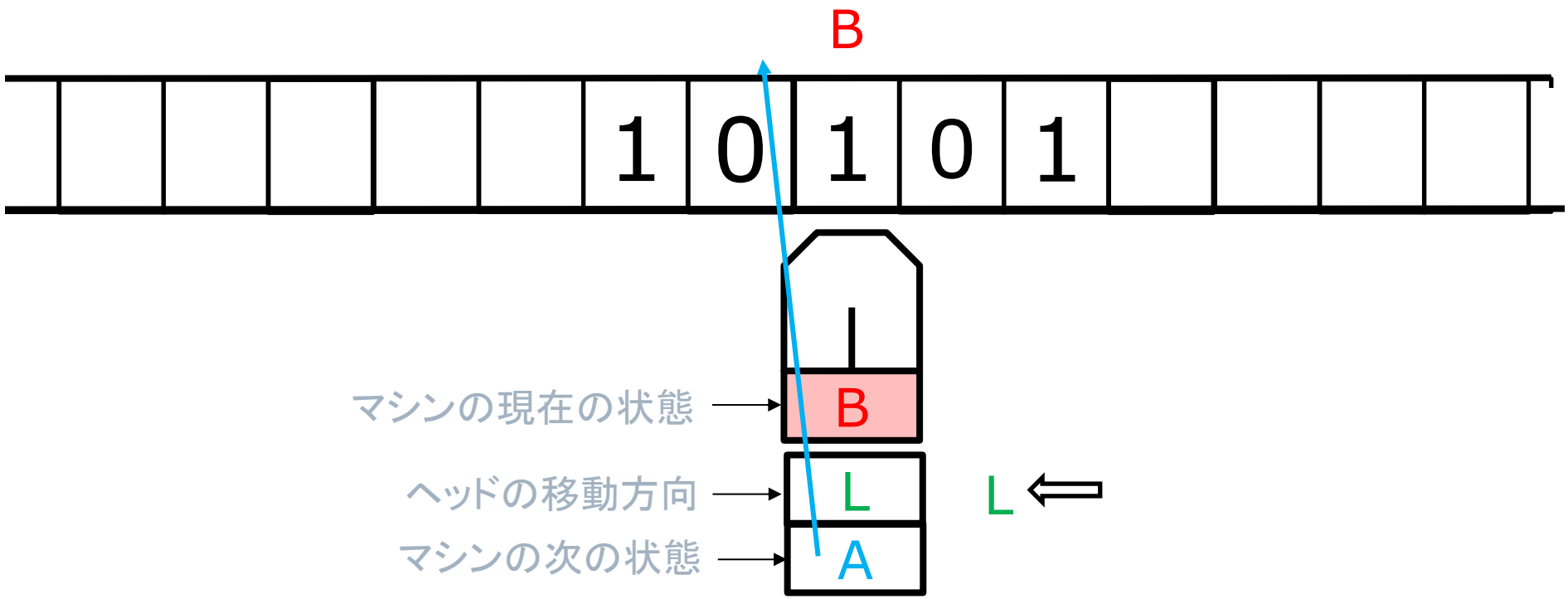
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



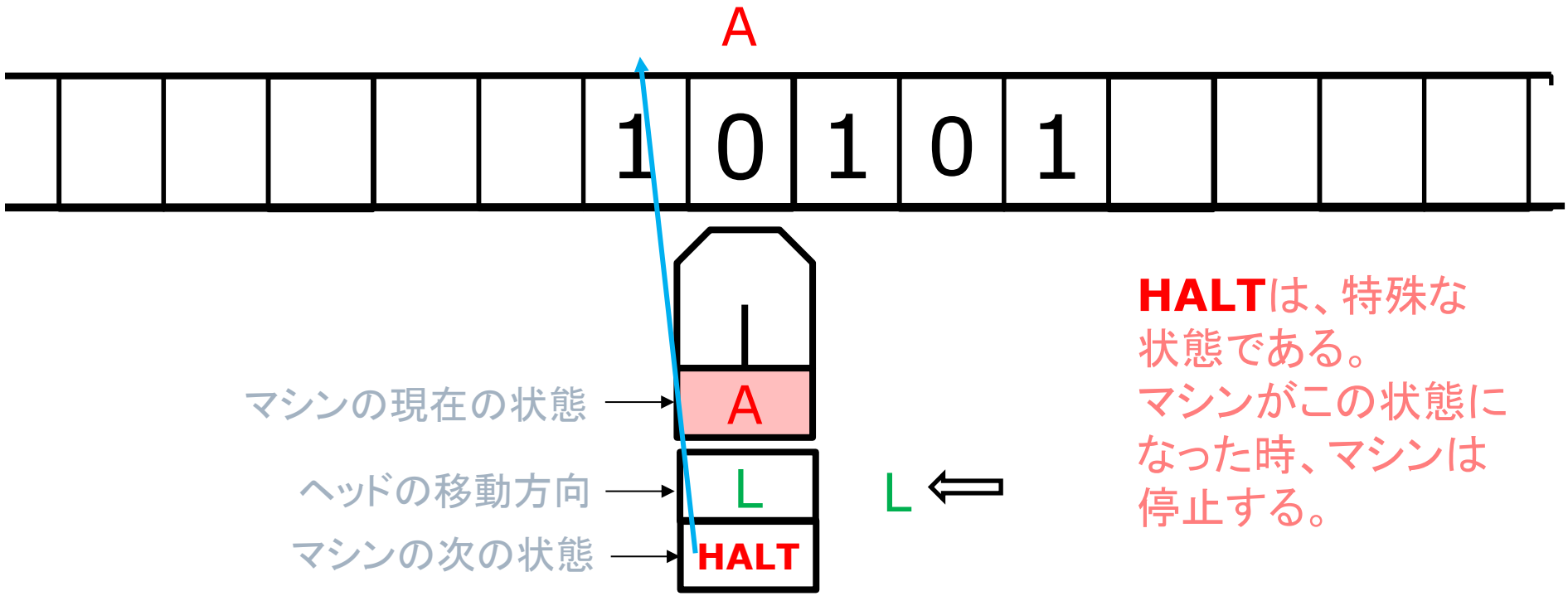
チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



チューリングマシンの状態の遷移のサンプル

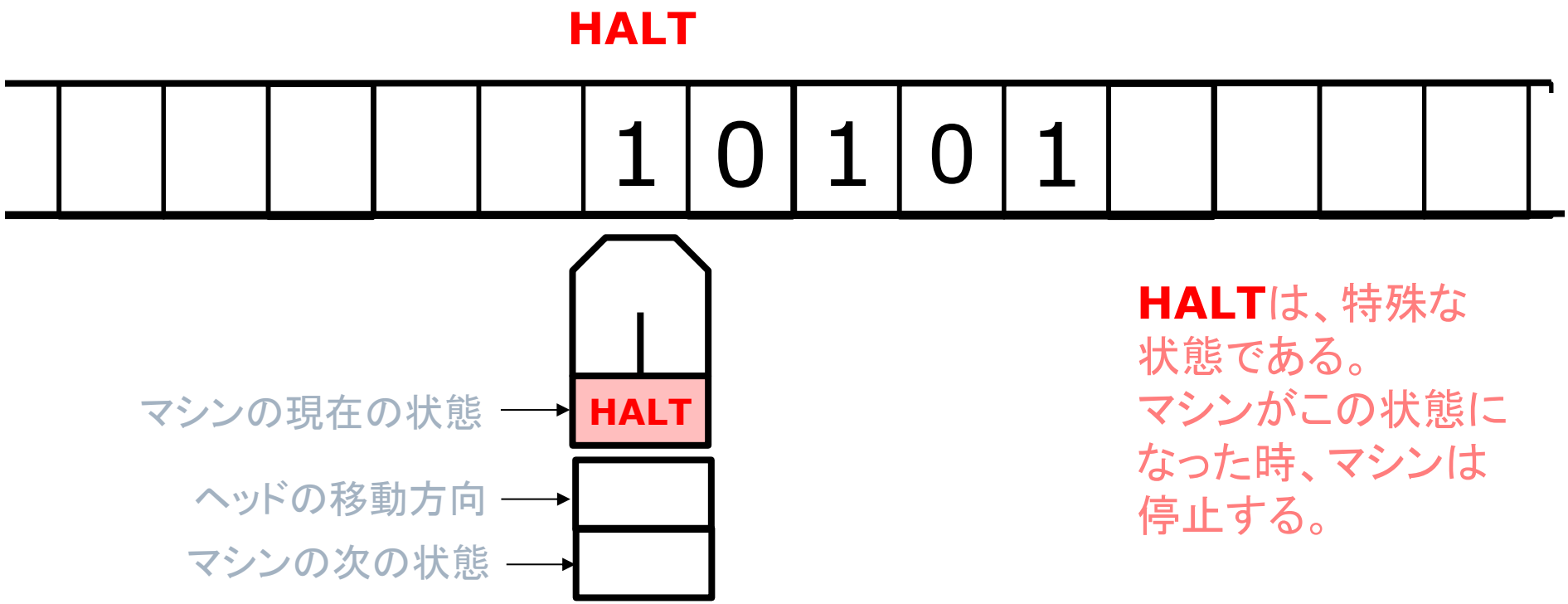
- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。



HALTは、特殊な状態である。
マシンがこの状態になった時、マシンは停止する。

チューリングマシンの状態の遷移のサンプル

- 以下のサンプルでは、ヘッドが位置した時点でのチューリングマシンの状態を、テープの上部にも書き込んでいる。

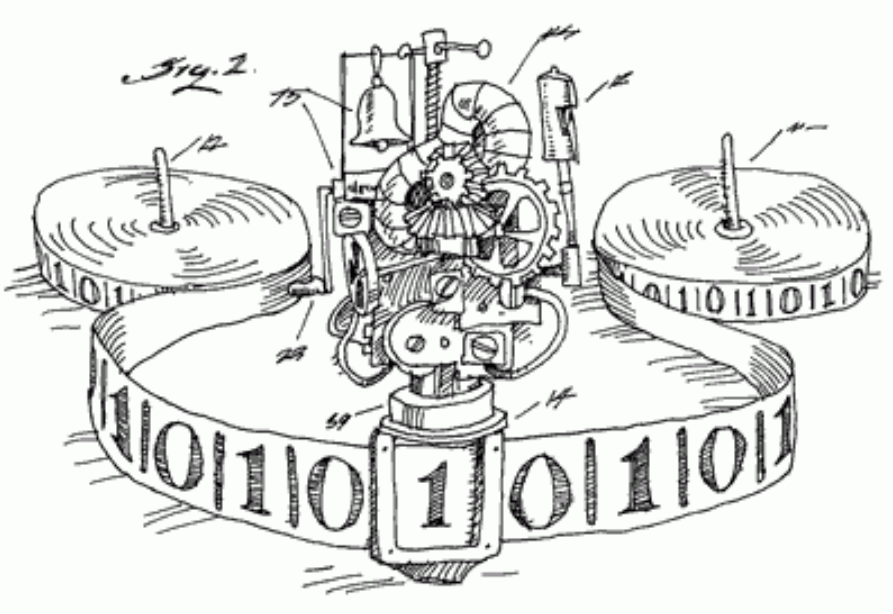



Part II
チューリングマシンをプログラムする

Agenda Part II
チューリングマシンをプログラムする

- チューリングマシンの命令とその実行
  - チューリングマシンの「命令」
  - チューリングマシンの「命令」を図で表す
  - チューリングマシンの「命令」を文字列で表す
  - ある命令セットが与えられた時の
    チューリングマシンの動作を見る
  - 命令のノテーションについて
- 簡単なチューリングマシンのプログラム・サンプル
  - 文字列の長さを求める
  - 文字列中の'1'の数の偶奇を求める
  - 文字列のコピー

チューリングマシンの命令とその実行



「チューリングマシンを学ぼう！」

Part II チューリングマシンをプログラムする

チューリングマシンの「命令」

チューリングマシンの「命令」

- 先に見た例では、ヘッドは移動し、チューリングマシンの状態も変化したが、どのような条件で、ヘッドの移動が起きるかをみてこなかった。
- また、実際には、ヘッドの書き出しによって、テープの状態も変化するのだが、先の例では、テープの状態の変化をみてこなかった。

チューリングマシンの「命令」

- 先に見た例では、ヘッドは移動し、チューリングマシンの状態も変化したが、どのような条件で、ヘッドの移動が起きるかをみてこなかった。
- また、実際には、ヘッドの書き出しによって、テープの状態も変化するのだが、先の例では、テープの状態の変化をみてこなかった。
- チューリングマシンのこうした振る舞いは、チューリングマシンの「命令」によって定義されている。
- ここでは、チューリングマシンの命令について説明する。

チューリングマシンの「命令」
命令の構成要素

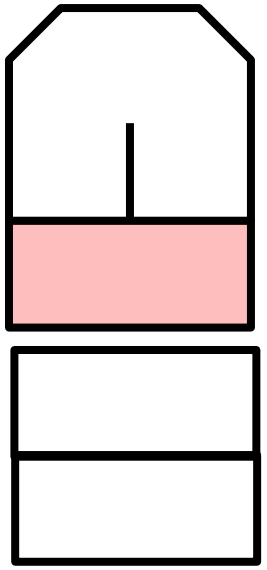
- チューリングマシンの命令は、次の要素から構成されている。

1. 現在のチューリングマシンの状態　(State-Now)
2. ヘッド位置のテープ上の一文字　　(Char-Read)
3. ヘッド位置に書き出される一文字　(Char-Write)
4. ヘッドが次に進む方向 (R, L)　　(Move-Direction)
5. 次のステップでのチューリングマシンの状態　(Next-State)

チューリングマシンの「命令」を図で表す

チューリングマシンの「命令」を図で表す

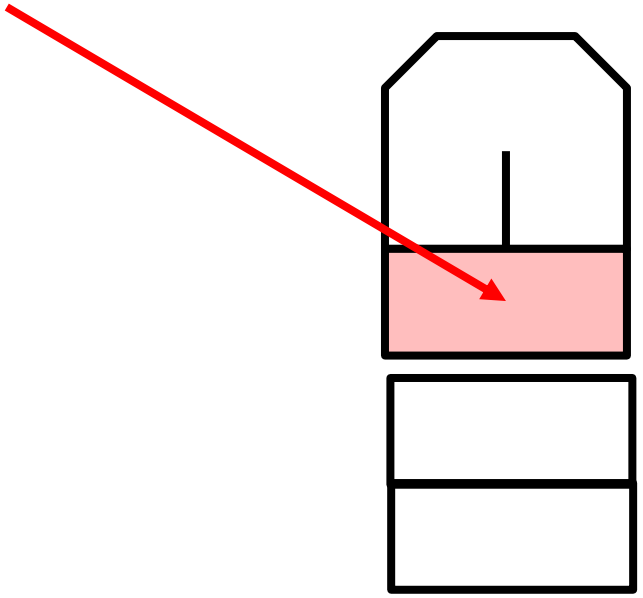
- チューリングマシンの命令を、次のような図で表すことができる。



チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

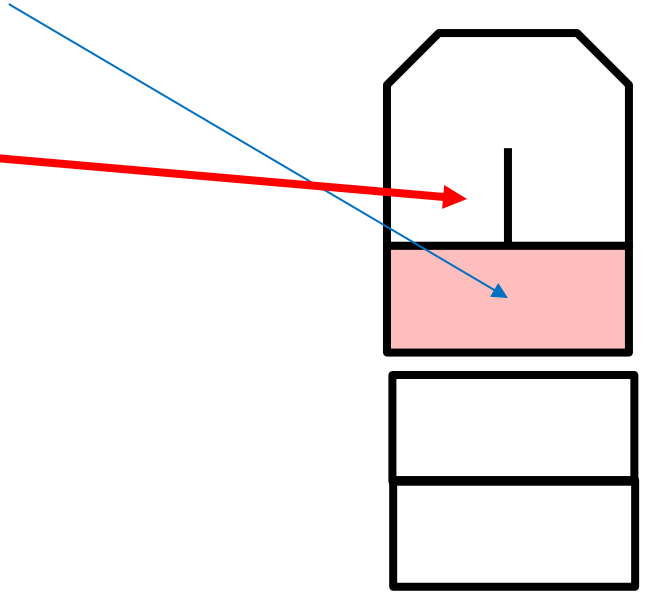
1. 現在のチューリングマシンの状態 (State-Now)



チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

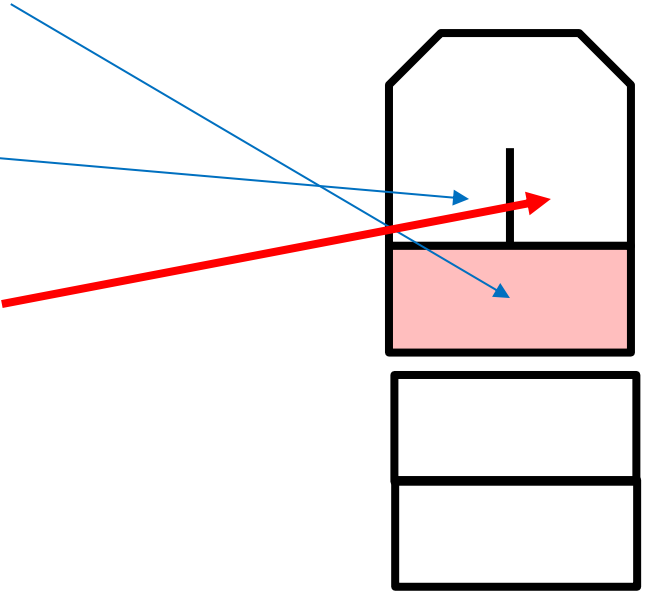
1. 現在のチューリングマシンの状態 (State-Now)
2. ヘッド位置のテープ上の一文字 (Char-Read)



チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

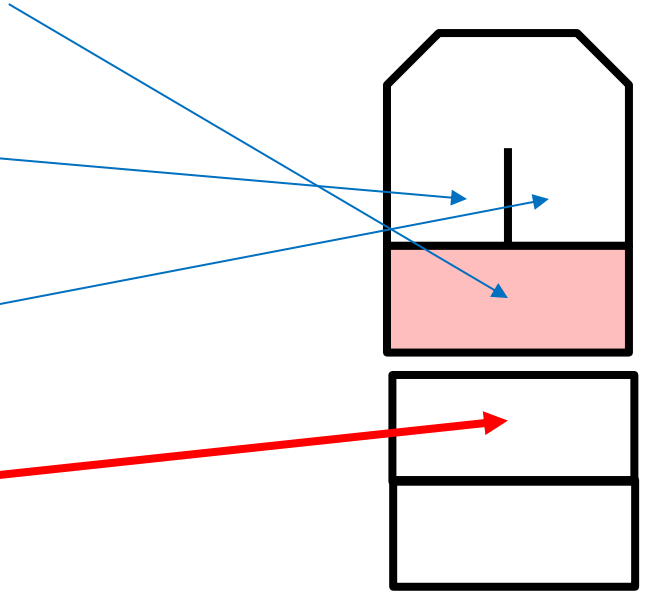
1. 現在のチューリングマシンの状態 (State-Now)
2. ヘッド位置のテープ上の一文字 (Char-Read)
3. ヘッド位置に書き出される一文字 (Char-Write)



チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

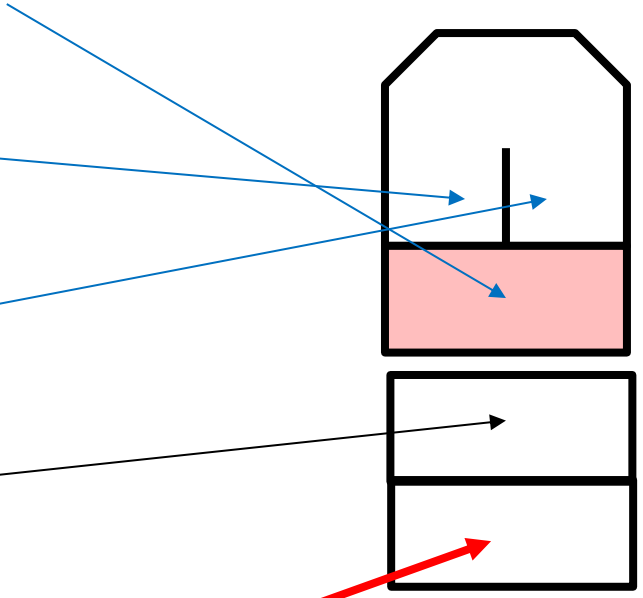
1. 現在のチューリングマシンの状態 (State-Now)
2. ヘッド位置のテープ上の一文字 (Char-Read)
3. ヘッド位置に書き出される一文字 (Char-Write)
4. ヘッドが次に進む方向 (R, L) (Move-Direction)



チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

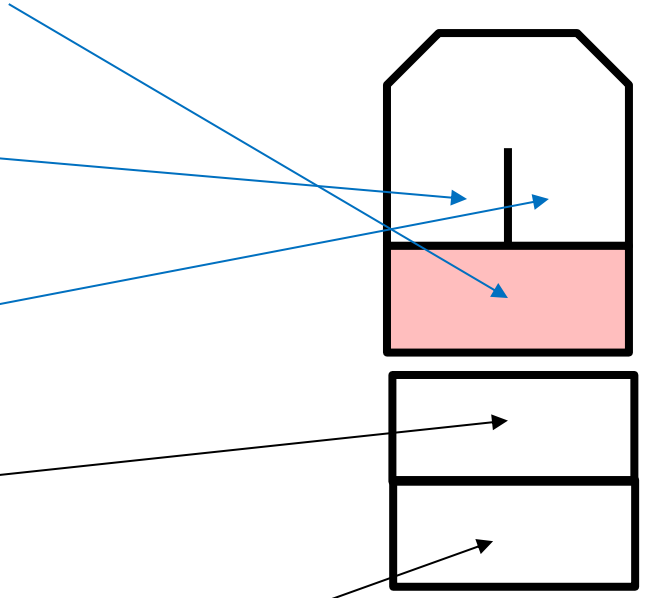
1. 現在のチューリングマシンの状態 (State-Now)
2. ヘッド位置のテープ上の一文字 (Char-Read)
3. ヘッド位置に書き出される一文字 (Char-Write)
4. ヘッドが次に進む方向 (R, L) (Move-Direction)
5. 次のステップでのチューリングマシンの状態 (Next-State)



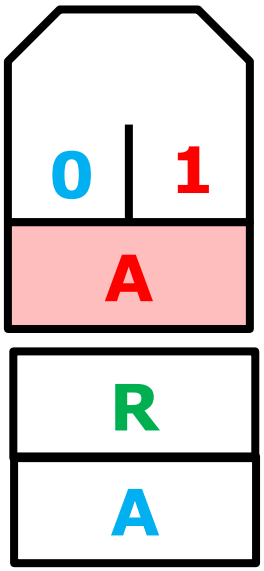
チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

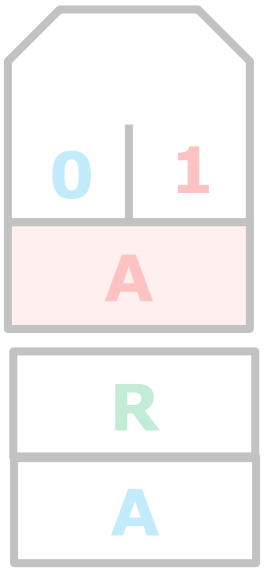
1. 現在のチューリングマシンの状態 (State-Now)
2. ヘッド位置のテープ上の一文字 (Char-Read)
3. ヘッド位置に書き出される一文字 (Char-Write)
4. ヘッドが次に進む方向 (R, L) (Move-Direction)
5. 次のステップでのチューリングマシンの状態 (Next-State)

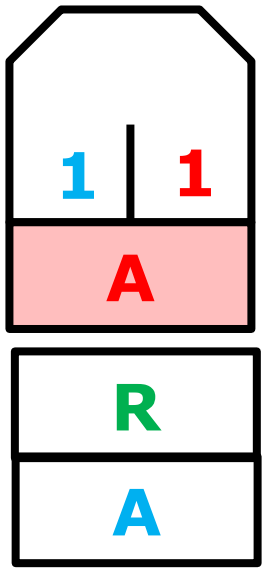


次のものは、図で表された
チューリングマシンの命令である



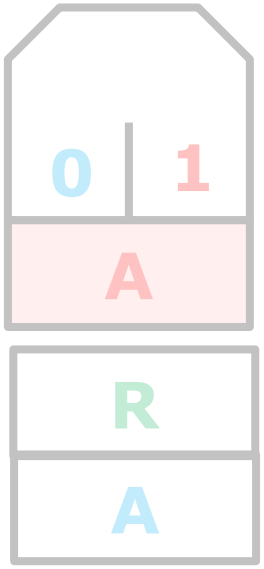
次のものは、図で表された
チューリングマシンの命令である

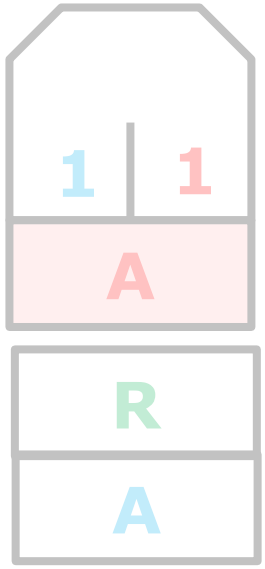


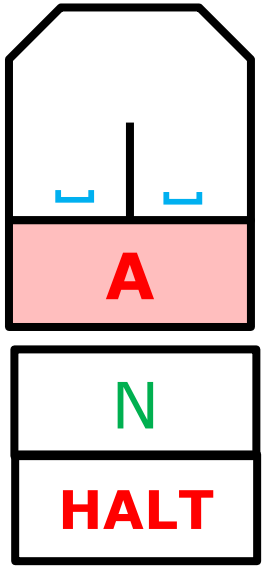


␣

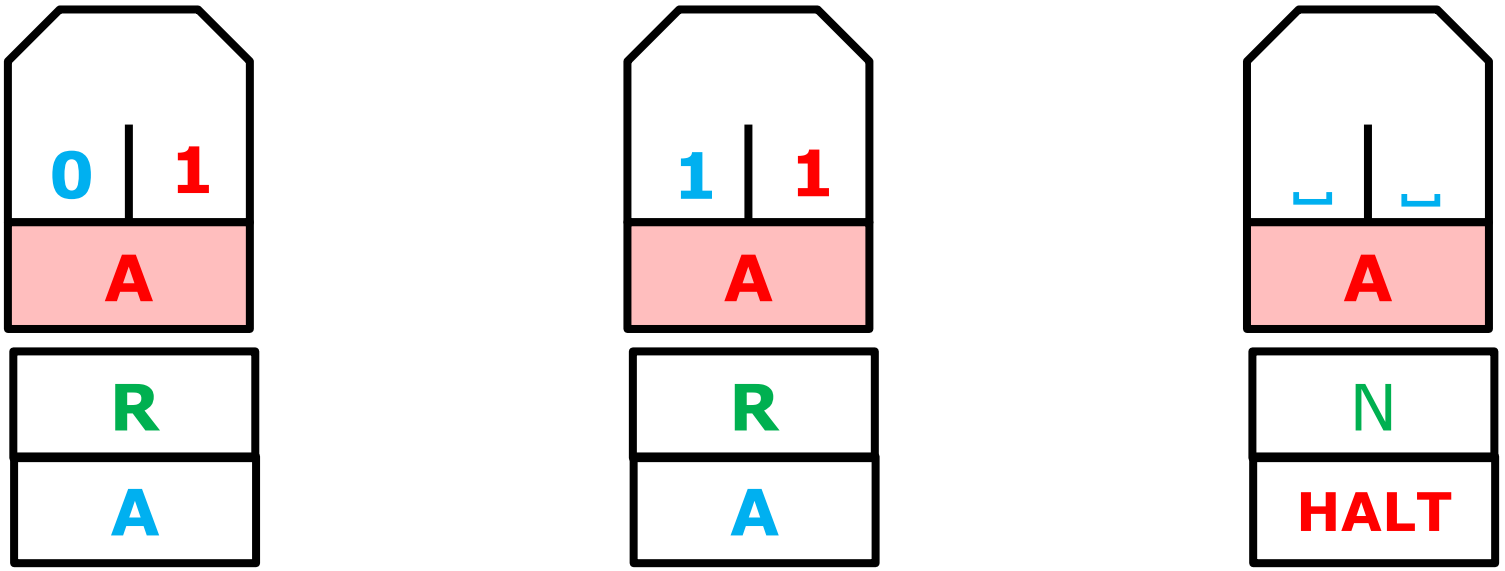
次のものは、図で表された
チューリングマシンの命令である





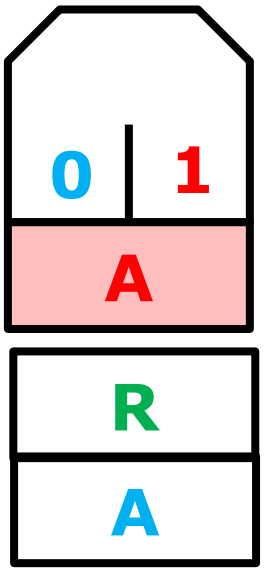


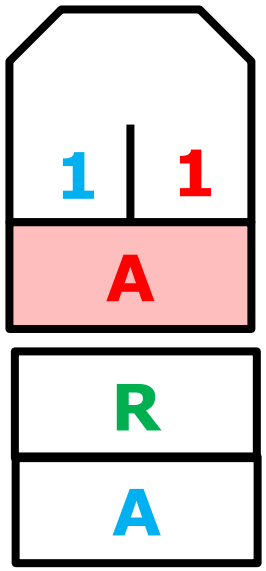
次のものは、図で表された
チューリングマシンの命令である

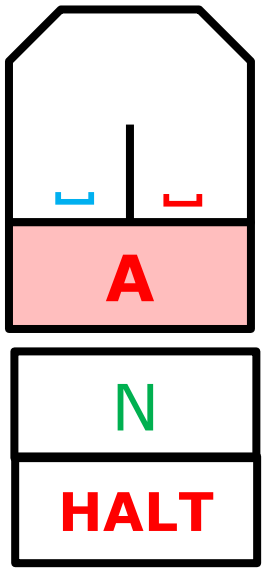


チューリングマシンの「命令」を文字列で表す

図で表された命令を
文字列で表す







A0→1:RA　　A1→1:RA　　A␣→␣:NHALT

文字列で表された命令の意味

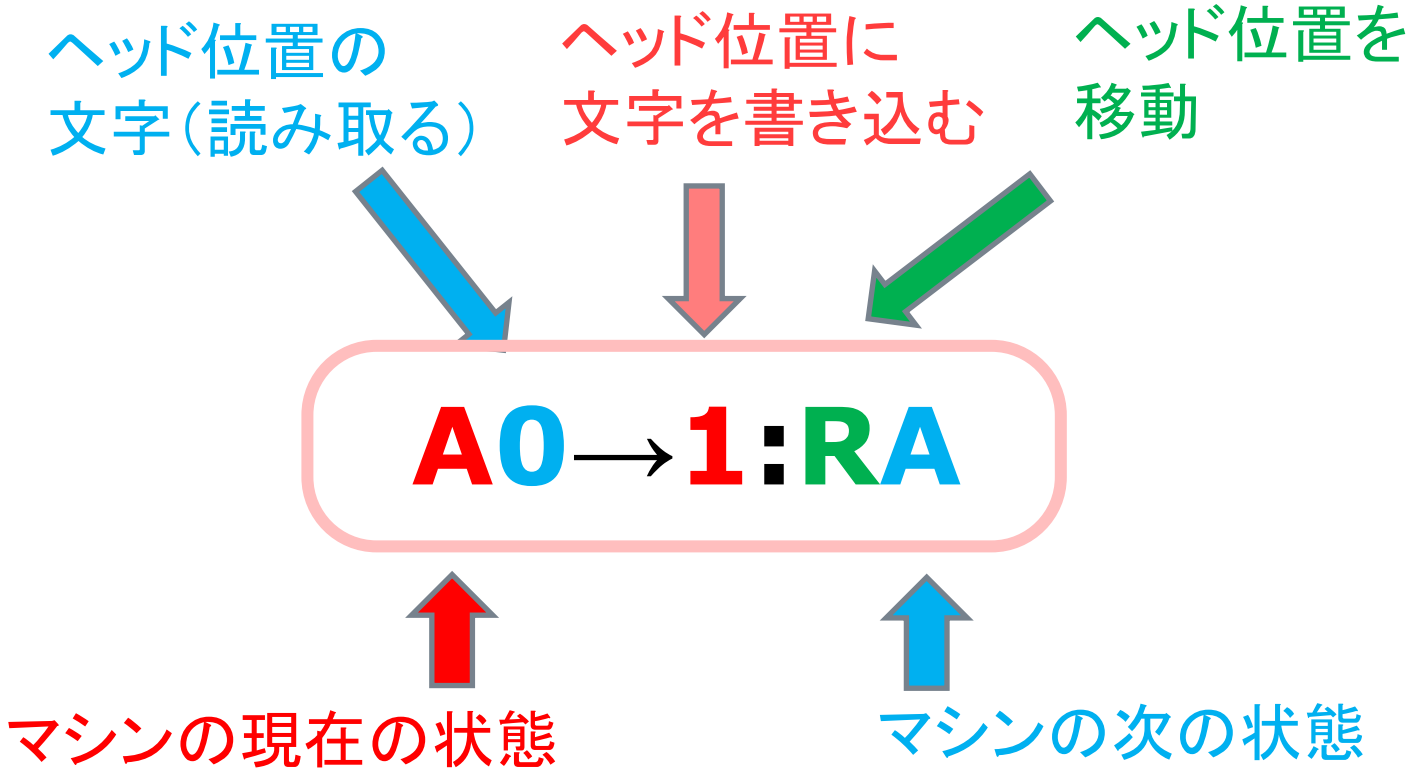
文字列で表された命令は、次のような意味を持つ

- **A0→1:RA**
  - 現在の状態が**A**でヘッドが**0**の上にあるなら
    ヘッド位置に**1**を書き込み、ヘッドを右に進め、状態を**A**にする。

- **A1→1:RA**
  - 現在の状態が**A**でヘッドが**1**の上にあるなら
    ヘッド位置に**1**を書き込み、ヘッドを右に進め、状態を**A**にする。

- **A␣→␣:NHALT**
  - 現在の状態が**A**でヘッドが␣（空白）の上にあるなら
    状態を**HALT**にして、停止する。

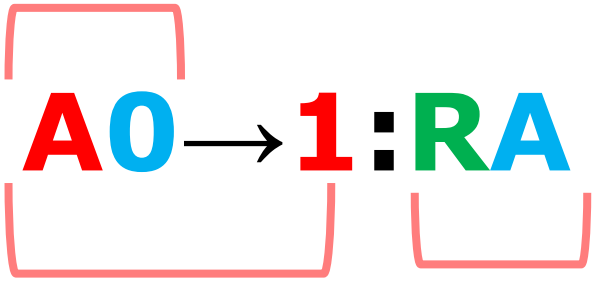
文字列で表された命令の意味



現在の状態がAでヘッドが0の上にあるなら、ヘッド位置に1を書き込み、ヘッドを右に進め、状態をAにする。

文字列で表された命令の意味

命令を特徴付ける
基本的な情報
状態＋ヘッド上の文字



現時点のヘッド位置
での状態とヘッド上の
文字の読み・書き

ヘッドの移動と
移動後の新しい
状態

現在の状態がAでヘッドが0の上にあるなら、ヘッド位置に1を書き込み、ヘッドを右に進め、状態をAにする。

複数の命令の集まりが、チューリングマシンの動作を決定する

例えば、ここでみた次の三つの命令の集まりが、このチューリングマシンの動作を決定する。

A0→1:RA
A1→1:RA
A␣→␣:NHALT

チューリングマシンの振る舞いは、命令セットによって特徴付けられる。

こうした命令セットは、コンピュータのプログラムにあたるものである。

チューリングマシンの命令セットの様々な表現

Turingマシンの命令セット（プログラム）の例を示す。この表の、例えば、最初の State Aの行で、On '0' の欄の 'B1R' は、「状態Aで、ヘッドが'0'の上にあるなら、状態をBに変えて、1を書き込んで、ヘッドを右（R）に移動する」という命令を表す。

State	on 0	on 1
A	B1R	D0L
B	C1R	F0R
C	C1L	A1L
D	E0L	H1L
E	A1L	B0R

命令セット本体

	on 0			on 1		
	Print	Move	Goto	Print	Move	Goto
A	1	right	B	0	left	D
B	1	right	C	0	right	F
C	1	left	C	1	left	A
D	0	left	E	1	left	H
E	1	left	A	0	right	B

命令セットの説明

Hは特別な「状態」で、この状態の時、マシンは「停止」する

チューリングマシンの命令の様々な表現

- 先のテーブルでの命令セットの表現は、次の表での命令セットの表現と同じものである。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC

- ここでは、A,B,C,D,E,Fの6種類の状態を持つチューリングマシンの命令セット（プログラム）が定義されている。

- いずれの場合でも、命令を特徴付ける基本的な情報は、状態＋ヘッド上の文字である。

ある命令セットが与えられた時のチューリングマシンの動作を見る

初期状態

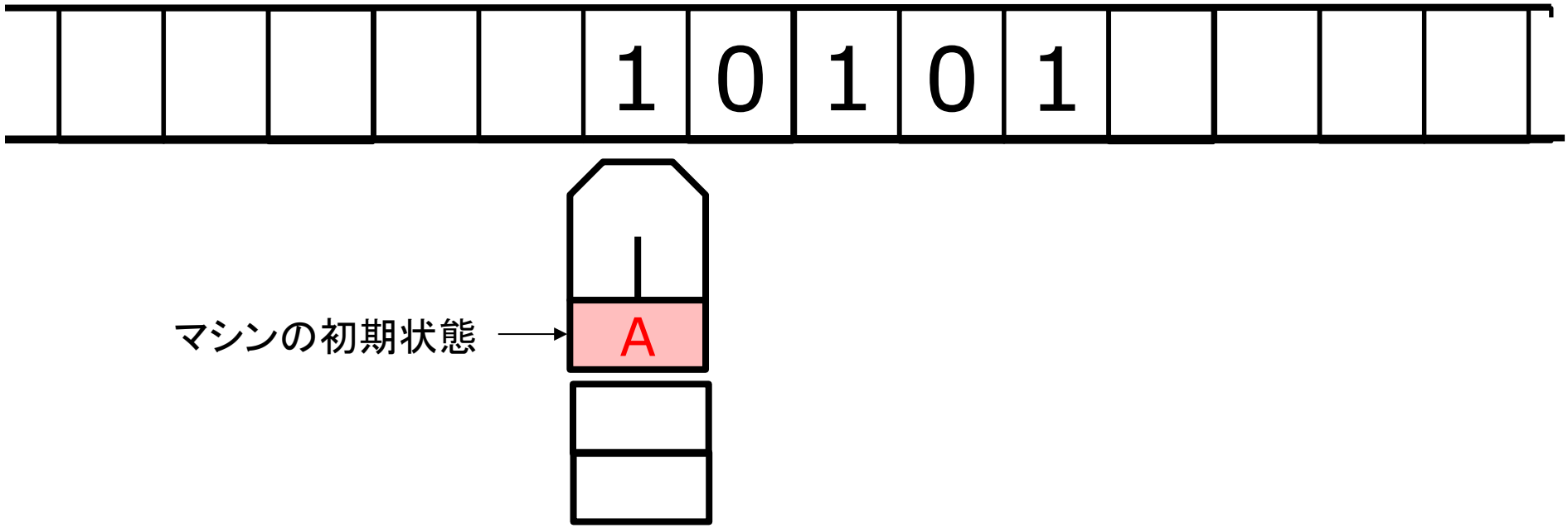
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

マシンの初期状態はAで、テープの状態は次のようになっている



ヘッド位置の文字を読み取る

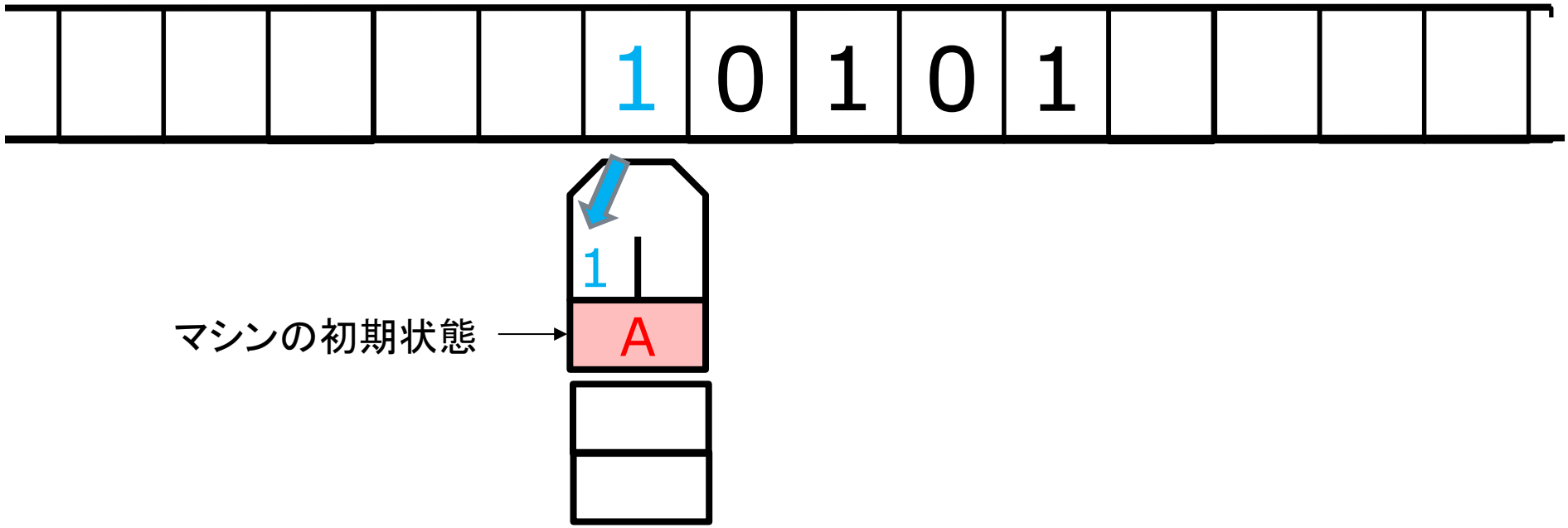
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

ヘッド位置の文字を読み取る



「状態＋ヘッド位置の文字」に合致する命令を探す

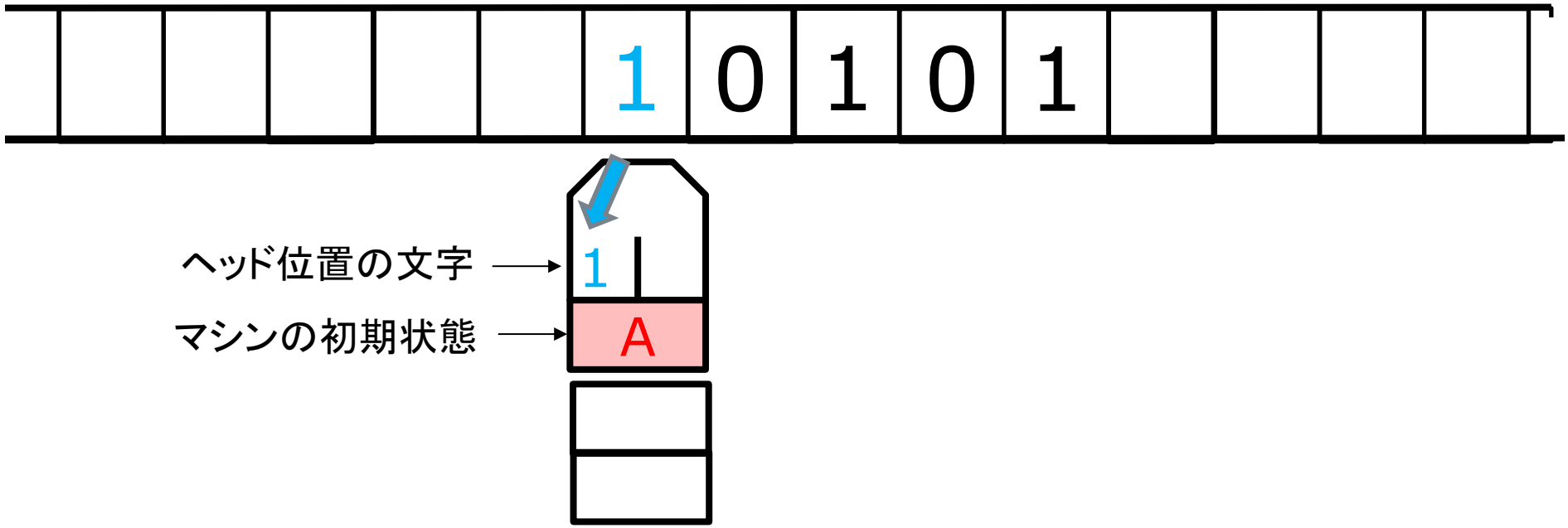
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

「状態＋ヘッド位置の文字」 A1 に合致する命令を探す



合致する命令をロードする

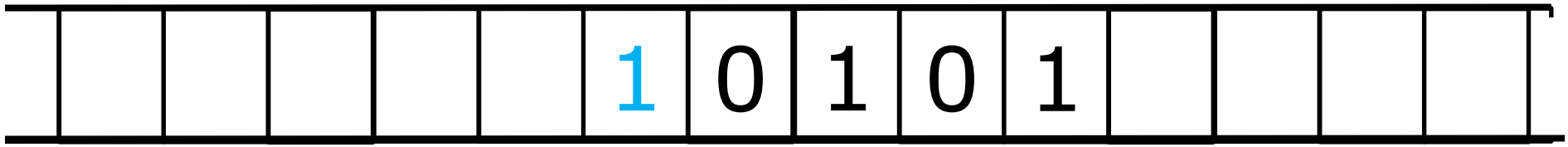
次の命令セットが与えられたとする

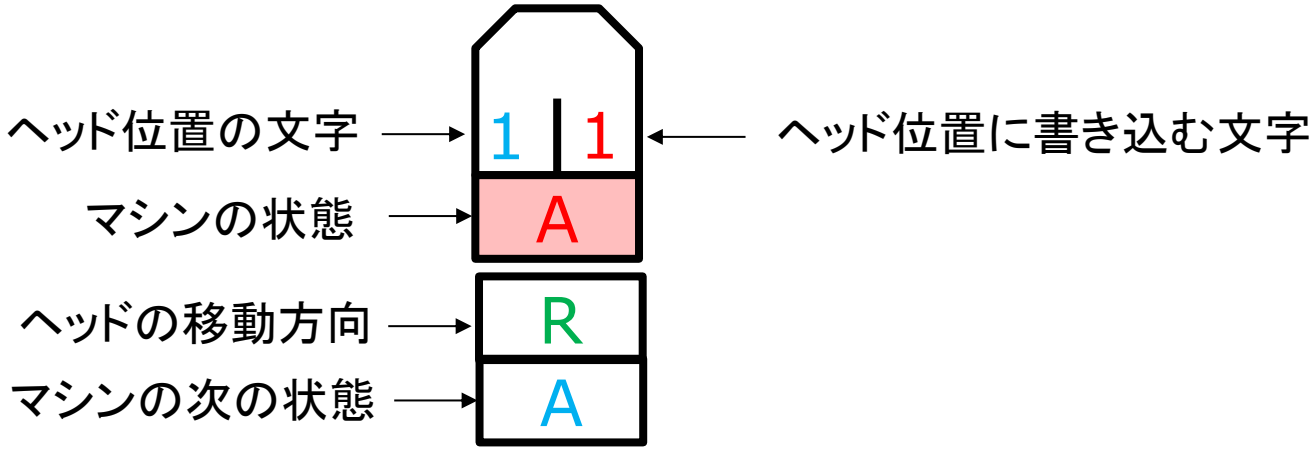
A0→1:RA

A1→1:RA

A␣→␣:NHALT

合致する命令をロードする





命令実行:テープに書き込む

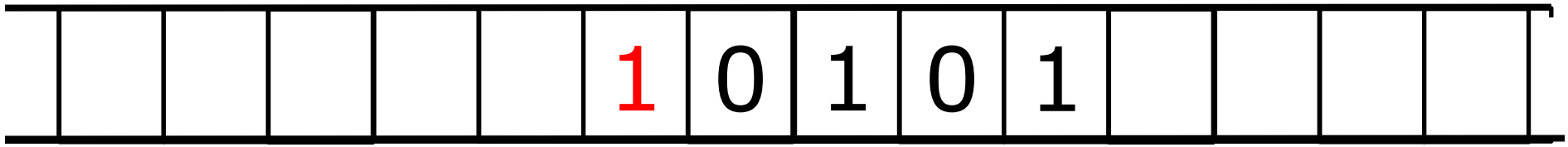
次の命令セットが与えられたとする

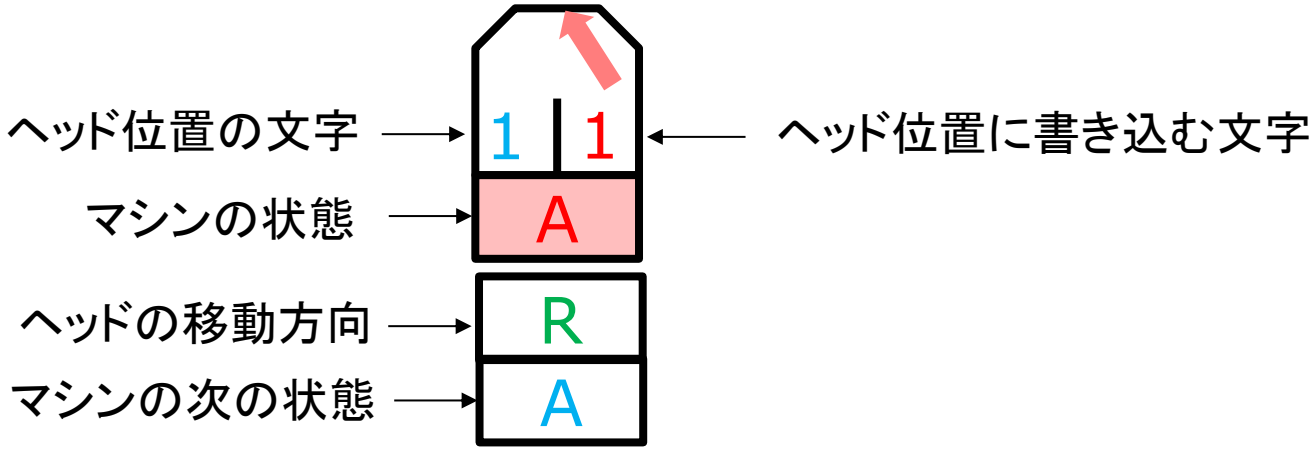
A0→1:RA

A1→1 RA

A␣→:HALT

命令実行:テープに書き込む





命令実行：ヘッド移動・状態更新

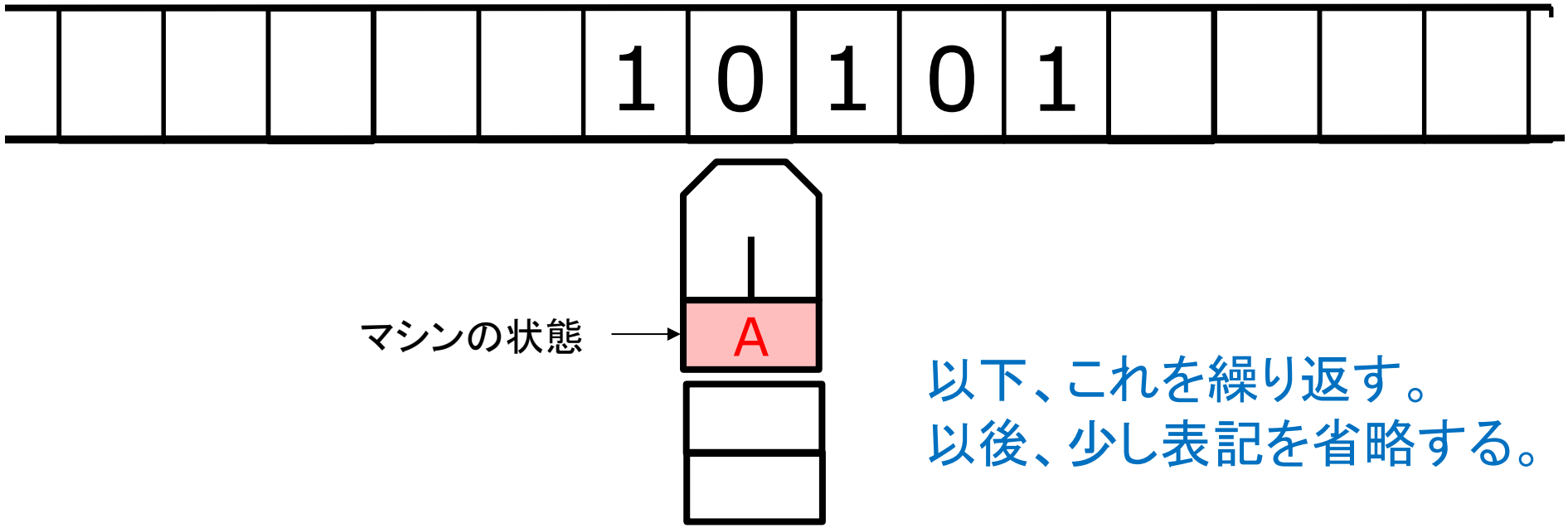
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行：ヘッド移動・状態更新



以下、これを繰り返す。
以後、少し表記を省略する。

命令をロードする

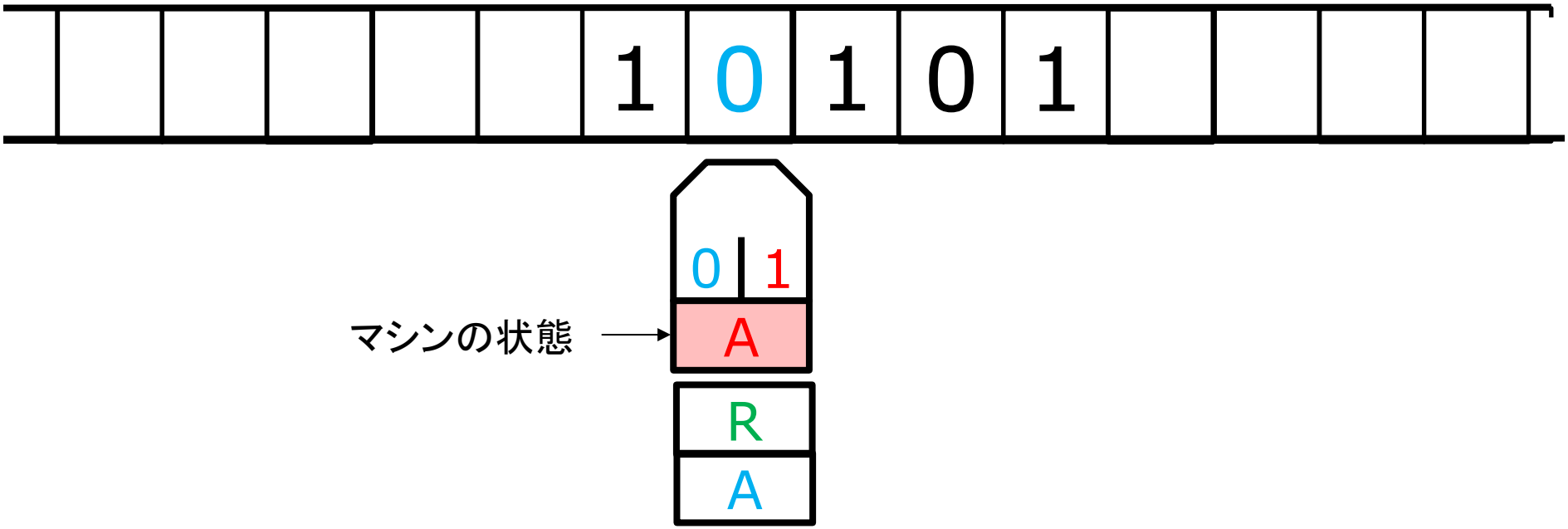
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令をロードする



命令実行：テープに書き込む

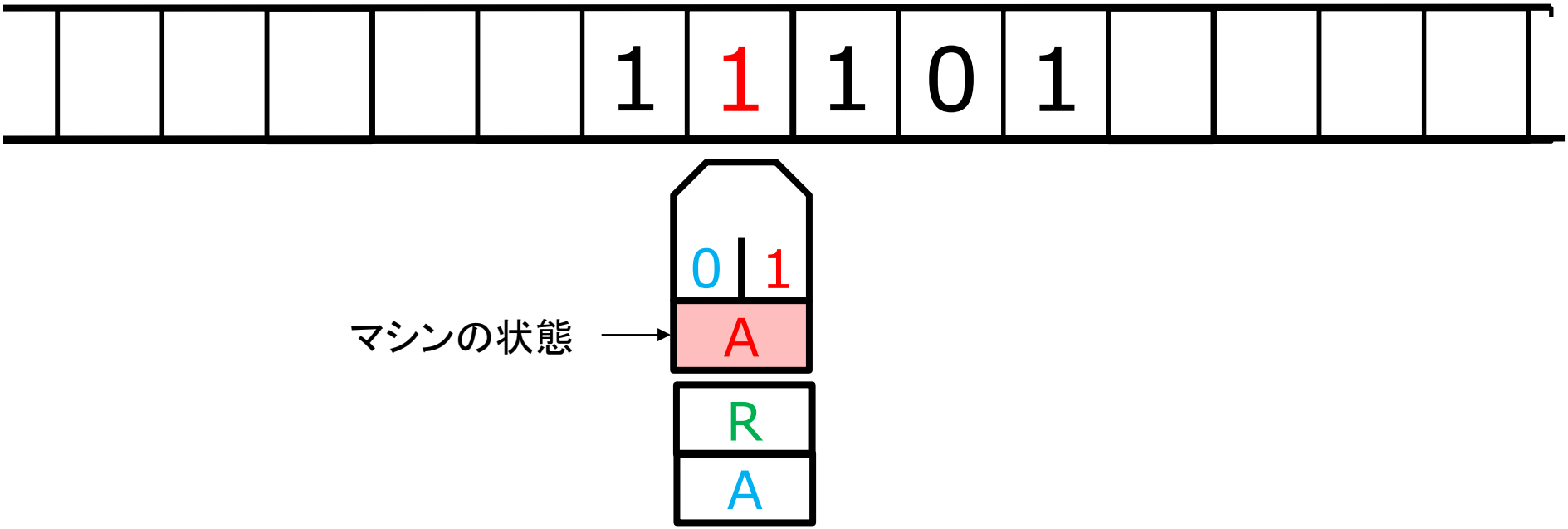
次の命令セットが与えられたとする

A0→1：RA

A1→1：RA

A␣→␣：NHALT

命令実行：テープに書き込む



命令実行：ヘッド移動・状態更新・命令ロード

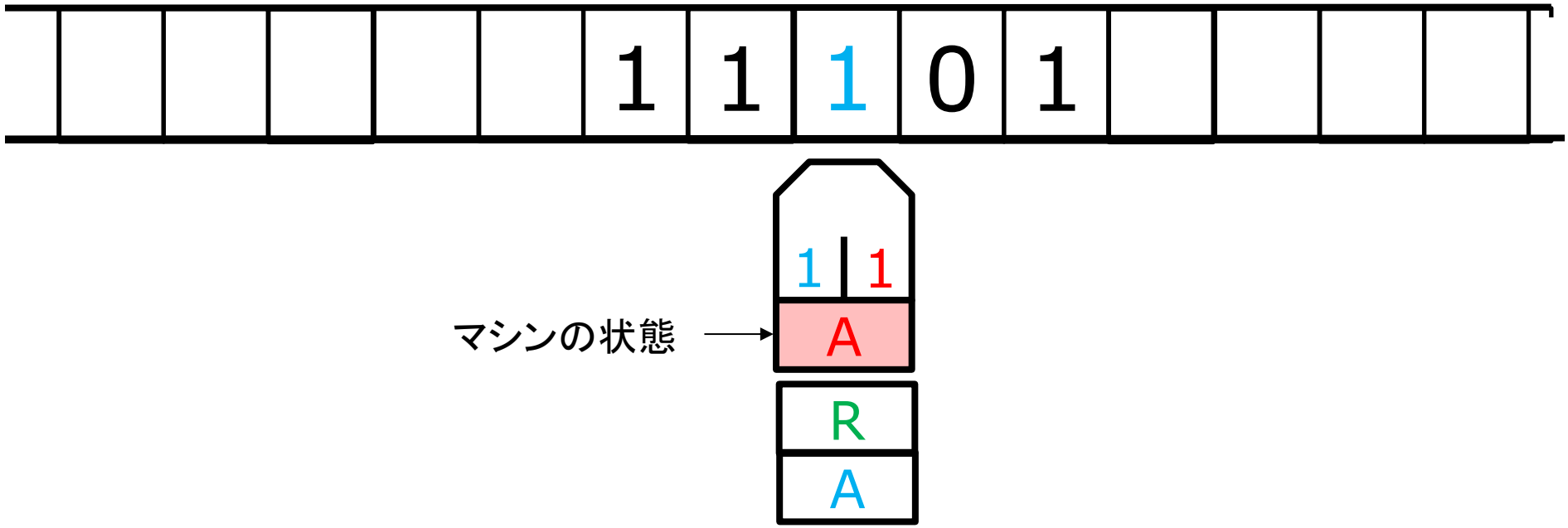
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行：ヘッド移動・状態更新・命令ロード



命令実行:テープに書き込む

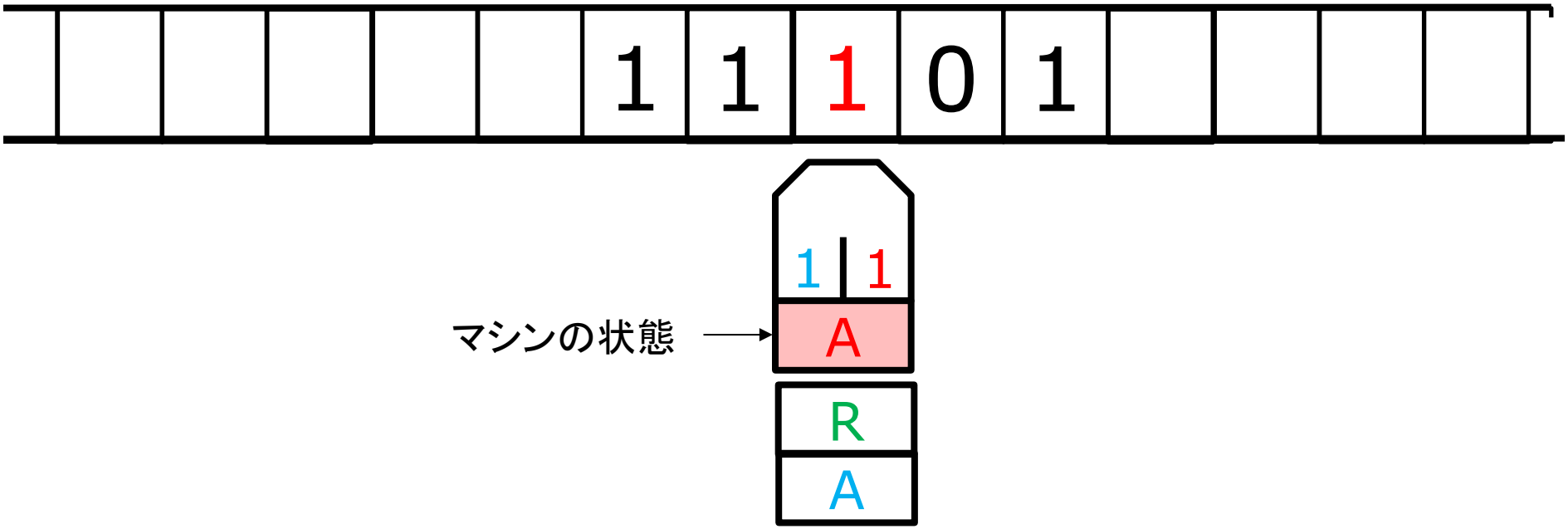
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→:HALT

命令実行:テープに書き込む



命令実行：ヘッド移動・状態更新・命令ロード

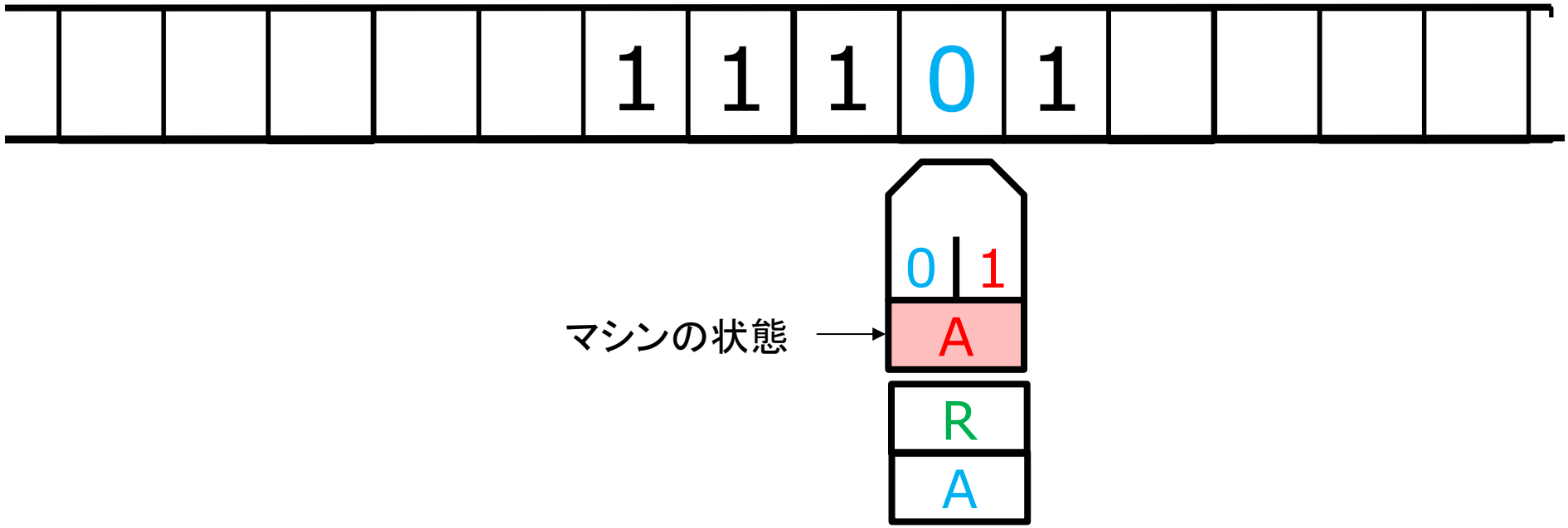
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行：ヘッド移動・状態更新・命令ロード



命令実行:テープに書き込む

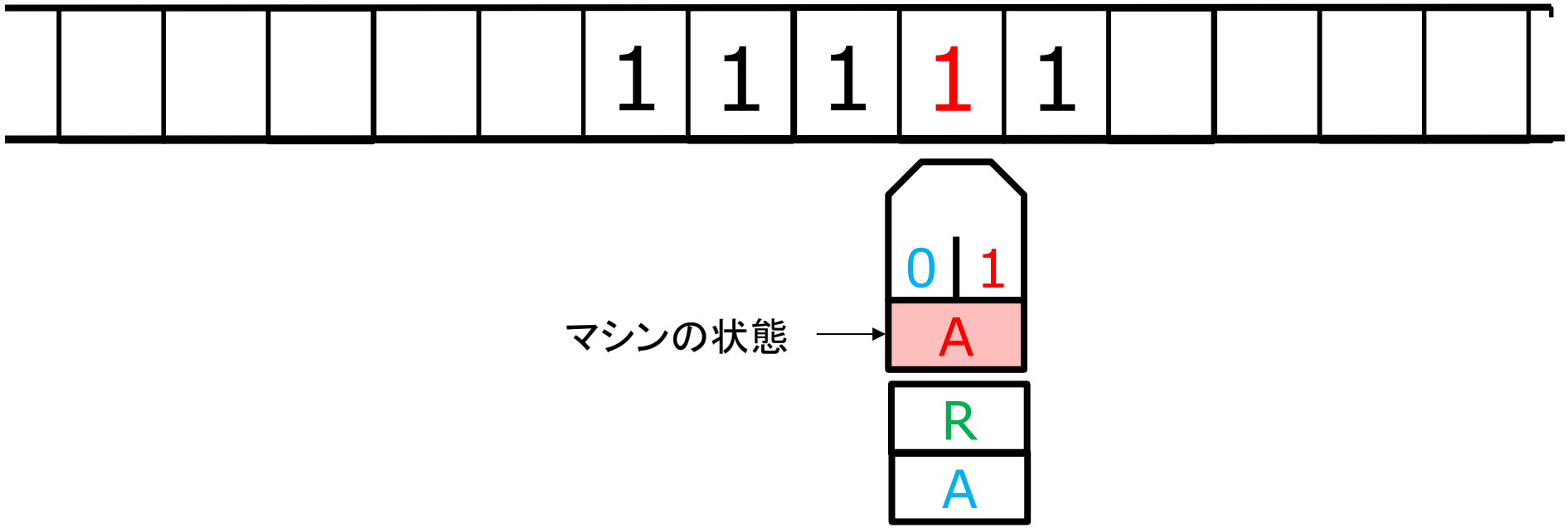
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行:テープに書き込む



命令実行：ヘッド移動・状態更新・命令ロード

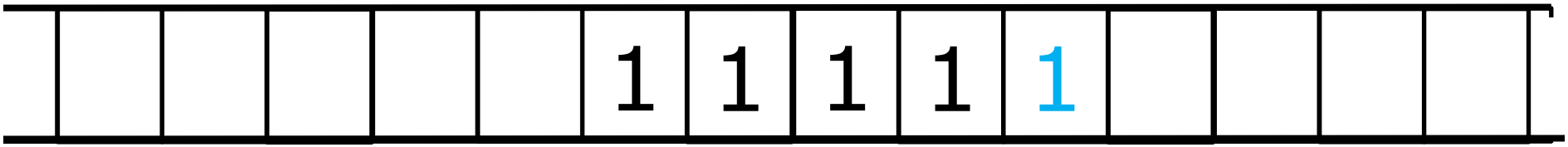
次の命令セットが与えられたとする

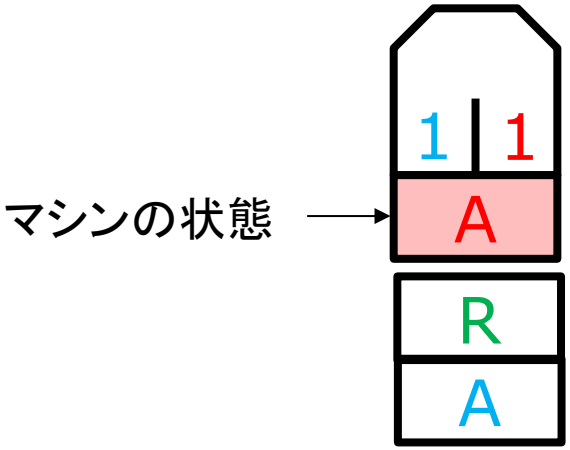
A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行：ヘッド移動・状態更新・命令ロード





命令実行:テープに書き込む

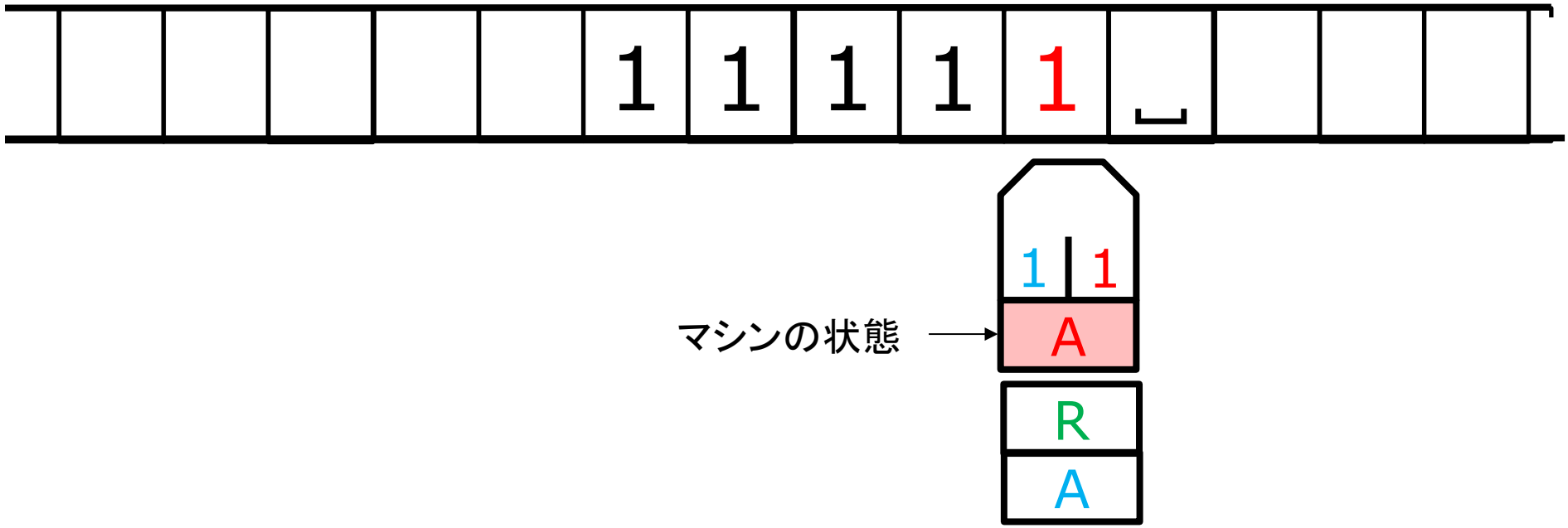
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行:テープに書き込む



命令実行：ヘッド移動・状態更新・命令ロード

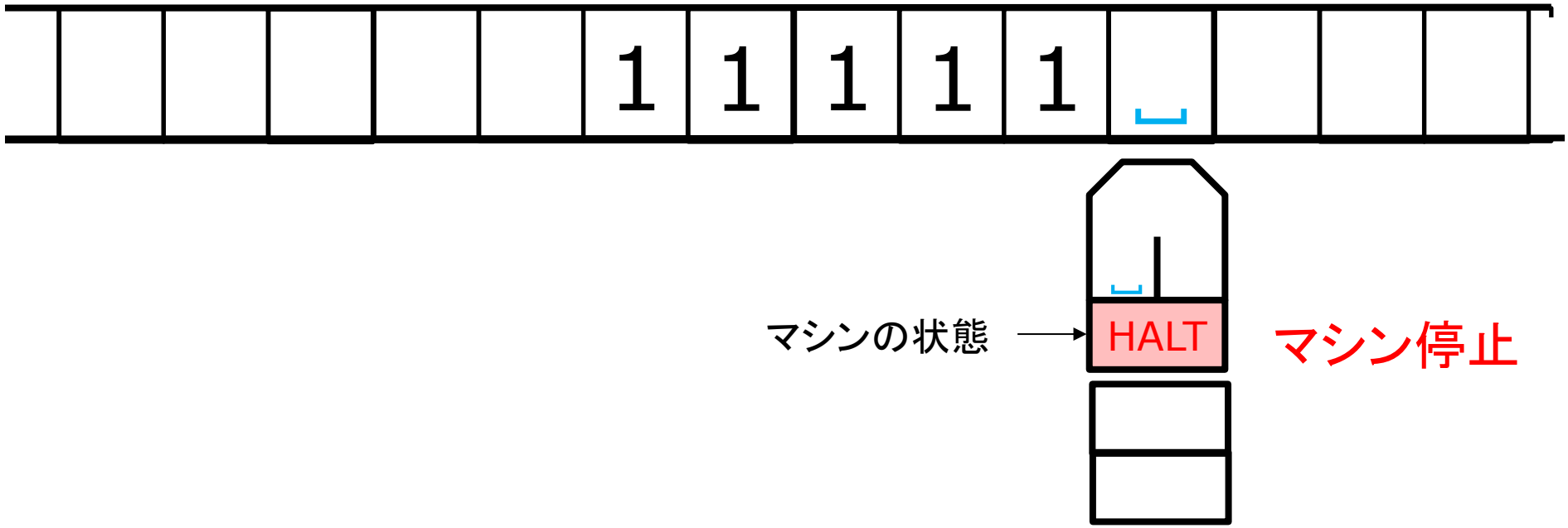
次の命令セットが与えられたとする

A0→1:RA

A1→1:RA

A␣→␣:NHALT

命令実行：ヘッド移動・状態更新・命令ロード

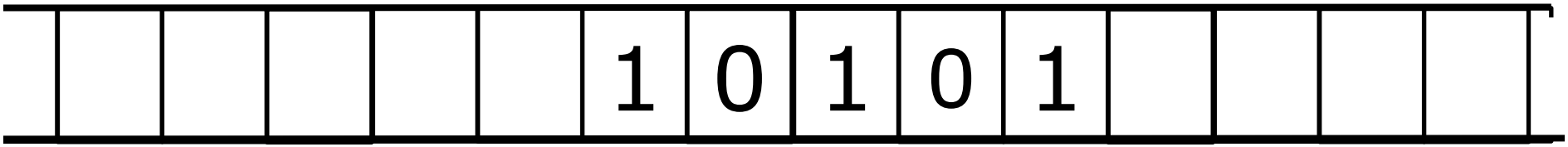


命令セット(プログラム)の実行結果

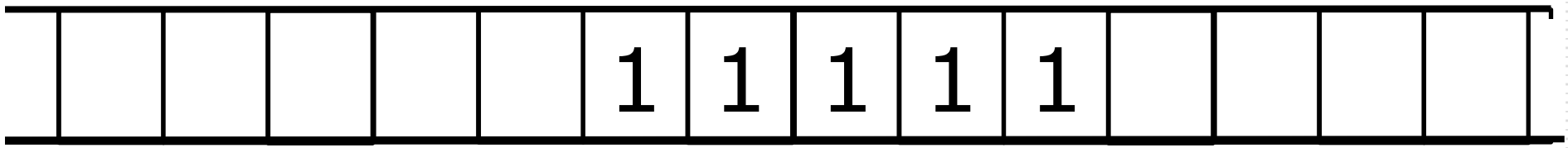
次の命令セット(プログラム)が与えられていた

A0→1:RA
A1→1:RA
A␣→␣:NHALT

マシンの初期状態はAで、テープの状態は次のようになっていた



命令セット(プログラム)の実行結果



文字列中の'0'が'1'に置き換わっている。

命令のノテーションについて

チューリングマシンの「命令」
命令の構成要素

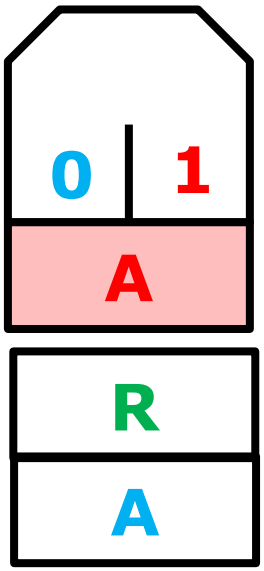
□ チューリングマシンの命令は、次の要素から構成されている。

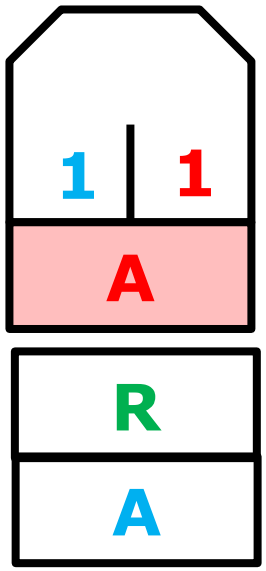
1. 現在のチューリングマシンの状態　(State-Now)
2. ヘッド位置のテープ上の一文字　(Char-Read)
3. ヘッド位置に書き出される一文字　(Char-Write)
4. ヘッドが次に進む方向 (R, L)　(Move-Direction)
5. 次のステップでのチューリングマシンの状態　(Next-State)

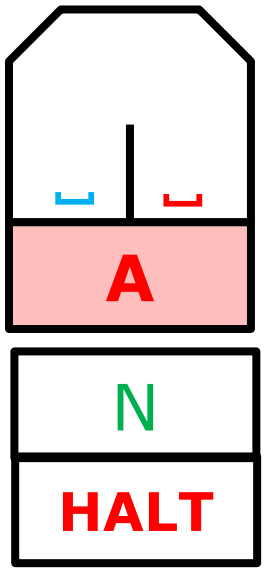
チューリングマシンの命令は、基本的には、この5つの要素の組み(c1,c2,c3,c4,c5)である。

ただ、それでは見にくいので、前回はそれを図形で表したり、文字列で表したりした。

図で表された命令と
文字列で表された命令

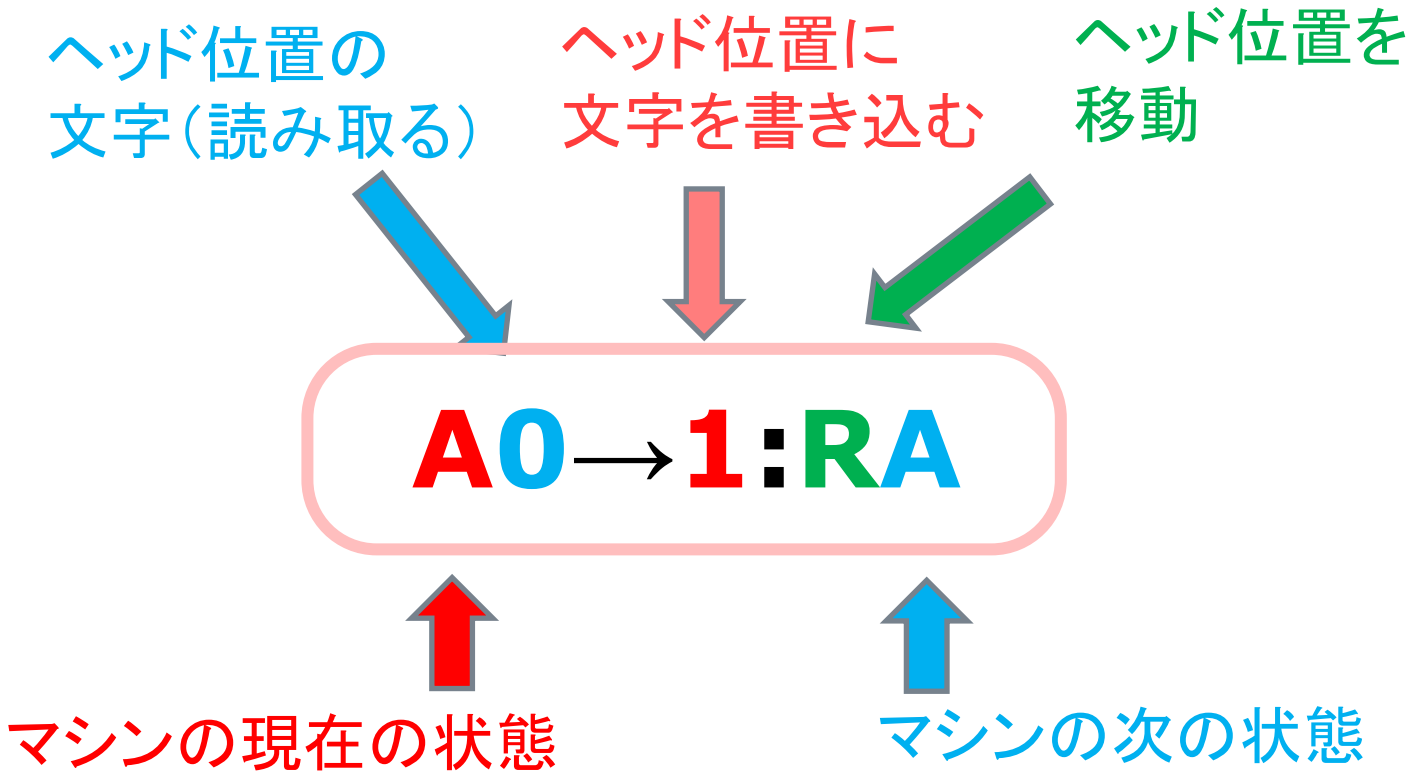






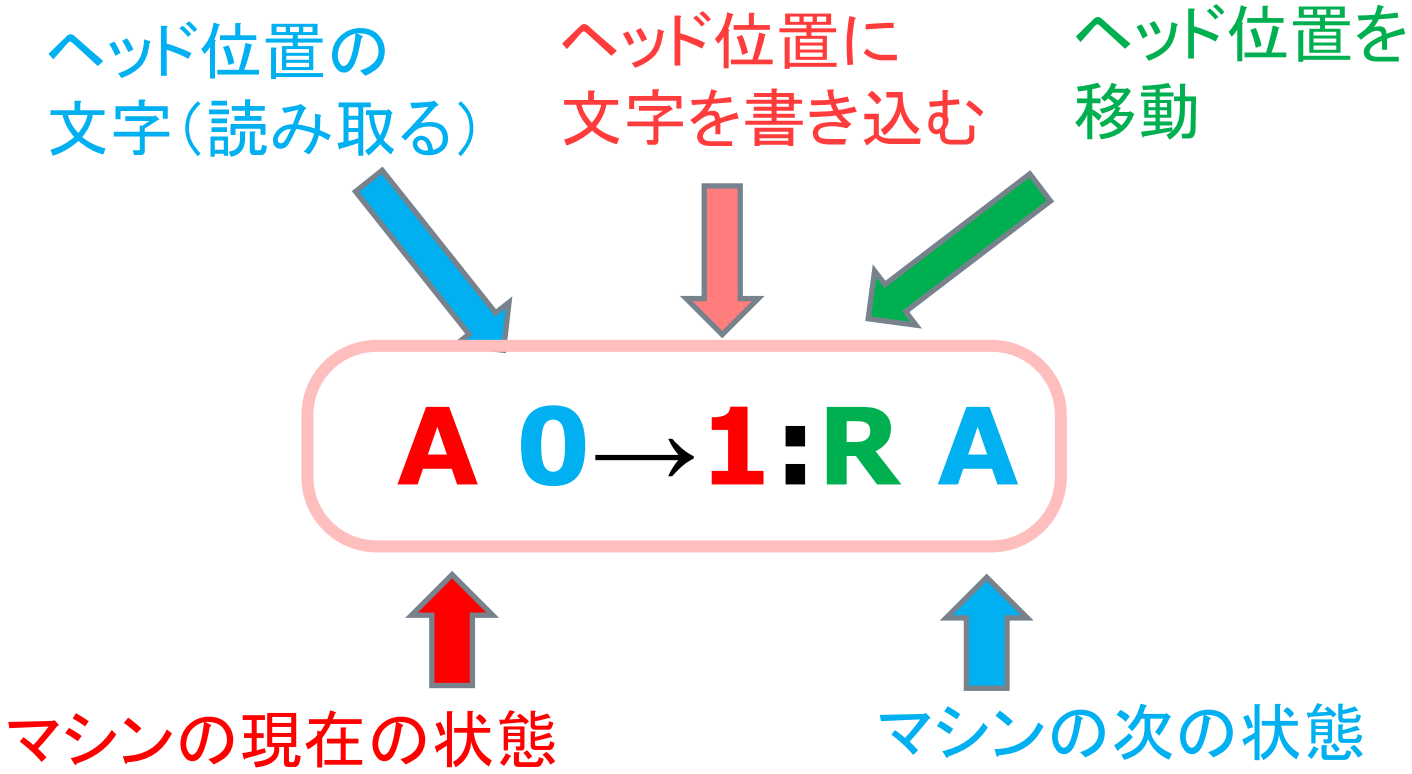
A0→1:RA　　A1→1:RA　　A␣→␣:NHALT

前回見た、文字列で表された命令



現在の状態がAでヘッドが0の上にあるなら、ヘッド位置に1を書き込み、ヘッドを右に進め、状態をAにする。

文字列で表された命令（修正版）



現在の状態がAでヘッドが0の上にあるなら、ヘッド位置に1を書き込み、ヘッドを右に進め、状態をAにする。

修正版のノテーションについて

- 状態を表すリテラルが一文字であるとは限らないので、修正版のノテーションでは、状態を表すリテラルと文字との間にスペースを入れた。
- 状態を表すリテラルは、その状態の意味を表した文字列を使った方が、プログラムがわかりやすくなるからである。
- また、ヘッドの移動を表す L,Rと状態を表すリテラルとの間にもスペースを入れた。(これは必須ではないのだが)
- 少し見やすくなったが、これにはトレードオフがある。
  スペース(ブランク)をテープ上の文字として読み取る場合の表記が曖昧になる。その時は、シングルクォートでくくった ' ' や ' ␣ ' や ' 」 やBを使うことにする。

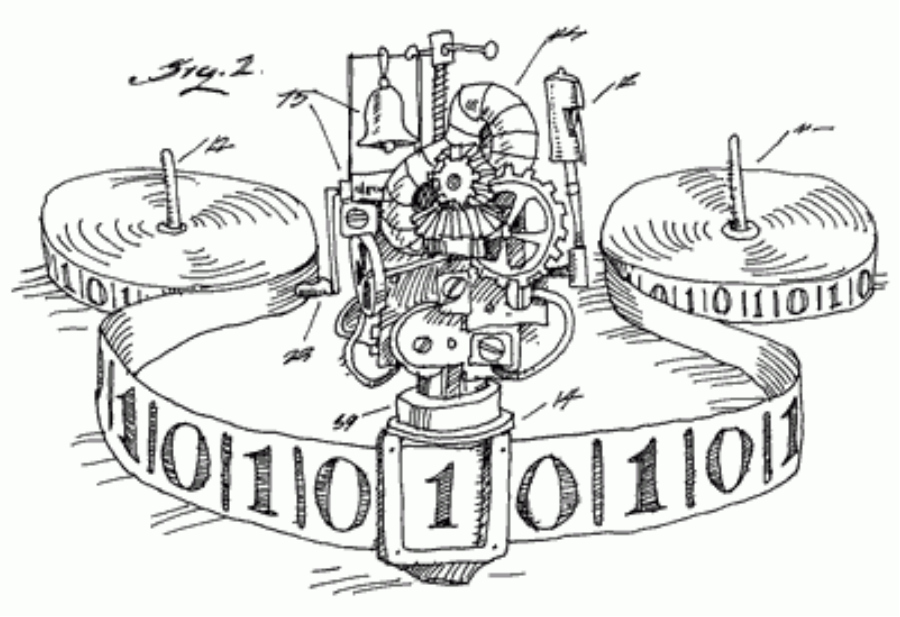
命令のノテーションについて いくつかのアイデア

1. A␣1RA　　状態が一文字なら一番簡潔
2. A␣→1:RA　　前回の表記
3. And ␣→1:R Or　　今回の修正版
4. And ` ′→`1′:R Or　　文字であることを強調する
5. And,` ′,1,R,Or
6. (And,␣,1,R,Or)　　これが元々の5つ組である

その他

- ヘッド移動のL, Rにヘッドを移動しないNを加える
- テープに書き込みをしないを、”A 0→ϕ:R A”で表す
- Back0 [X]?->1:L,R Fetch　これはJumpだ。 等々

簡単なチューリングマシンのプログラム

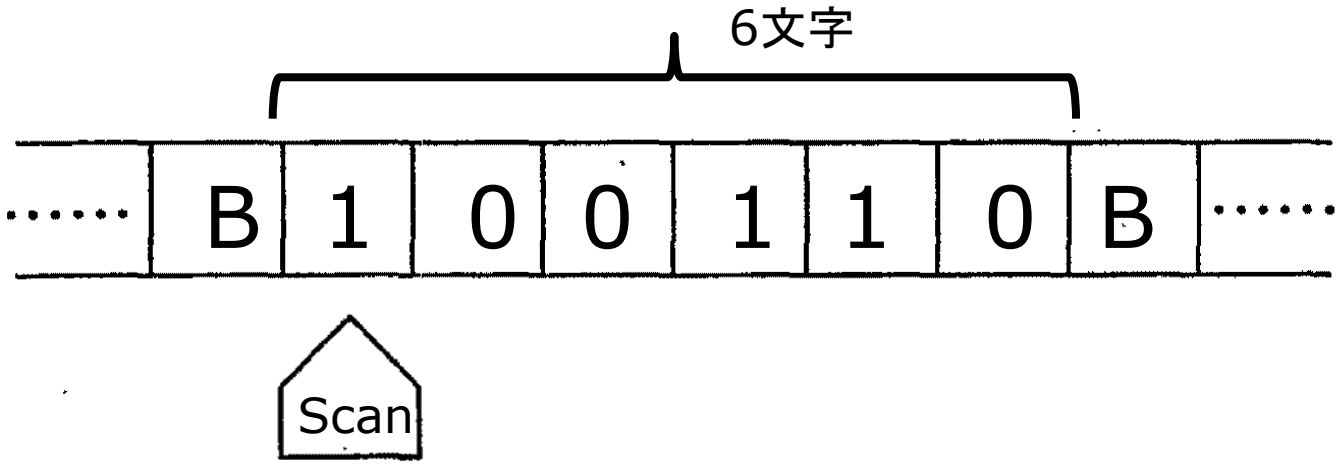


「チューリングマシンを学ぼう！」
Part II チューリングマシンをプログラムする

文字列の長さを求める

例1： 文字列の長さを求める

問題1：B(Blank:何も書かれていない空白のマス目）とBのあいだにある、0と1の記号の列（文字列）の長さを求めよ。



この例なら、6を返せばいい。

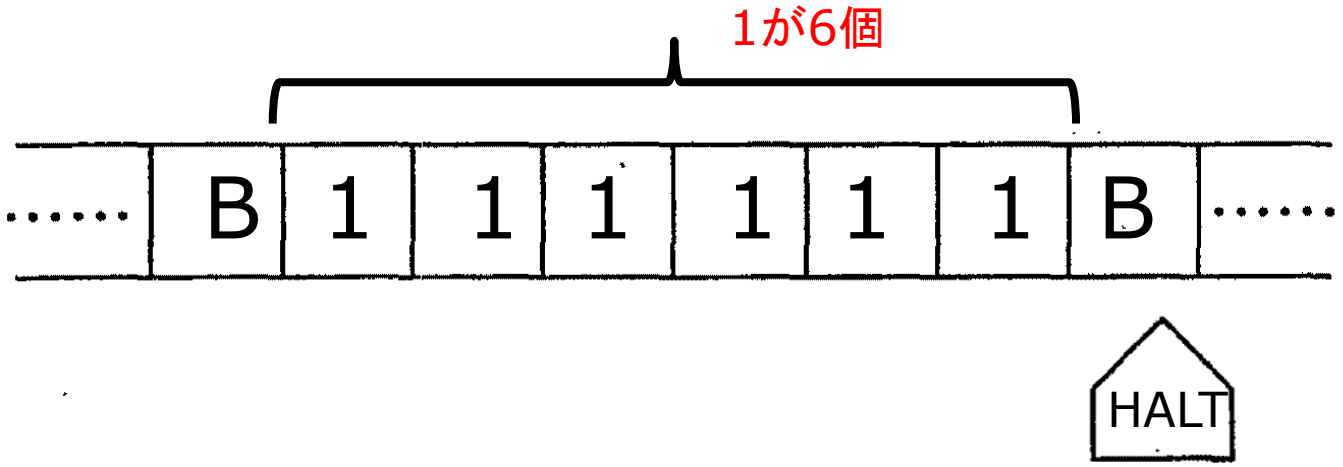
例1： 文字列の長さを求める

これは、状態一つでも解ける。この状態をScanとしよう。
次の命令セットを考える。

1. Scan 1→ 1:R Scan
   /* ‘1’の上だと何もしないで、ヘッドを右に進める。状態はそのまま。*/
2. Scan 0→ 1:R Scan
   /* ‘0’の上だ と‘1’に変えて、ヘッドを右に進める。状態はそのまま。*/
3. Scan ␣→ ␣:N Halt
   /* 空白に当たったら、ヘッド動かさず（N）停止する。 */

例1： 文字列の長さを求める

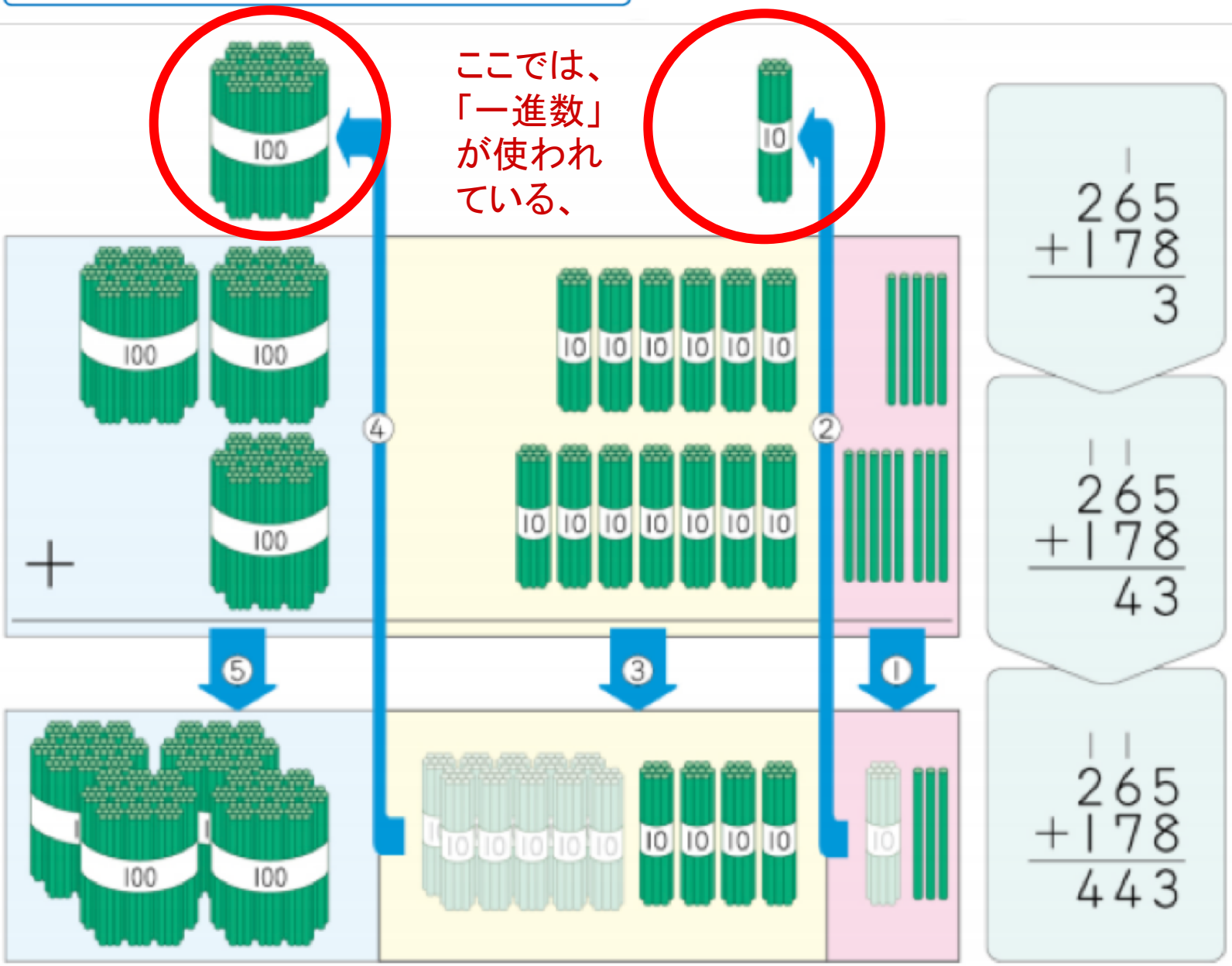
テープの状態は次のようになる。



こんな状態で停止する。実は、これでいいのだ。
これは、答え'6'の「一進数」表示だ！

3 265＋178 の筆算のしかたを考えましょう。

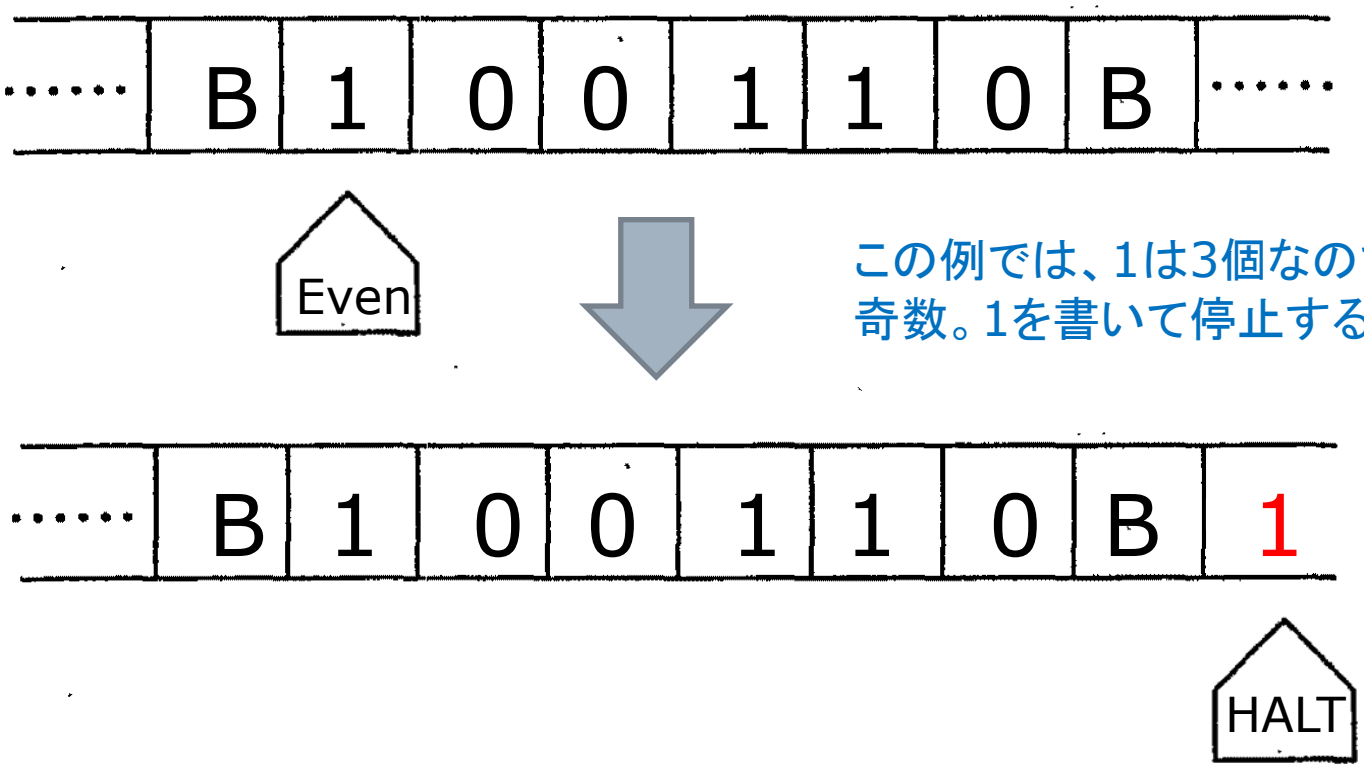
啓林館 3年上 p.57



文字列中の'1'の数の偶奇を求める

例2： 文字列中の'1'の数の偶奇を求める

問題2：BとBのあいだにある、0と1の文字列中の1の数の偶奇を求めよ。偶数なら0を奇数なら1を返せ。



例2： 文字列中の'1'の数の偶奇を求める

二つの状態 Even と Odd を使う。初期状態をEvenとする。

状態Evenの時、

1. '0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. '1'の上だと何も印字しないで、ヘッドを右に進める。状態はOddに。

状態Oddの時、

1. '0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. '1'の上だと何も印字しないで、ヘッドを右に進める。状態はEvenに。

ただ、これだと文字列末の'B'にヘッドは来た時の動作が定義されていないし、結果の印字もできていない。

新たに、二つの状態、Print_EvenとPrint_Oddを追加する。

例2： 文字列中の'1'の数の偶奇を求める

状態Evenの時、

1. '0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. '1'の上だと何も印字しないで、ヘッドを右に進める。状態はOddに。
3. 'B'の上だと何も印字しないで、ヘッドを右に進める。状態はPrintEvenに。

状態Oddの時、

1. '0'の上だと何も印字しないで、ヘッドを右に進める。状態はそのまま。
2. '1'の上だと何も印字しないで、ヘッドを右に進める。状態はEvenに。
3. 'B'の上だと何も印字しないで、ヘッドを右に進める。状態はPrintOddに。

状態Print_Evenの時、

1. 0を印字して停止する。

状態Print_Oddの時、

1. 1を印字して停止する。

例2： 文字列中の'1'の数の偶奇を求める

/* 状態 Evenの時; 初期状態は Evenとする */

1. Even 0→0 :R Even　　/* '0'の時パリティは変わらない */
2. Even 1→1 :R Odd　　/* '1'の時パリティは反転する */
3. Even ␣→␣ :R Print_Even　/* 文字列の最後 */

/*状態 Oddの時 */

4. Odd 0→0 :R Odd　　/* '0'の時パリティは変わらない */
5. Odd 1→1 :R Even　　/* '1'の時パリティは反転する */
6. Odd ␣→␣ :R Print_Odd　　/* 文字列の最後 */

/* 状態 Print_Evenの時 */

7. Print_Even ␣→0:R HALT　/* '0'をプリントして停止 */

/* 状態 Print_Oddの時 */

8. Print_Odd ␣→1:R HALT　　/* '1'をプリントして停止 */

文字列のコピー

例3: 文字列のコピー

問題3:AとBとの間にある、0と1からなる文字列を、BCの間にコピーする。

A011011BC --> A011011B011011C

A101010BC --> A101010B101010C

A1011BC --> A1011B1011C

A101BC --> A101B101C

基本的な考え方

次の例で考えてみよう。これは一文字だけのコピーである。

A1BC --> A1B1C

まずコピーされる文字'1'を読み取らねばならない。

A1BC

コピーされる文字を一文字読む状態をFetchと呼ぶことにしよう。

読んだ文字の情報をBとCの間まで運んでいけばいいのだが、
チューリングマシンには、そうした情報を記憶する機能はないし、
ヘッドの移動のたびに状態は変わりうる。

一つ方法がある。「文字'1'をコピー中」という状態Copy1を考えて
その状態を保持したままヘッドをCの位置まで移動し、そこに'1'を
書き込めばいい。

一文字をコピーする命令セット

Cが消えてしまったので、状態をMark1に変えて、最後尾に ‘C’を改めて書き込んで終了である。

こうした動作を実行する命令セットは次のようになる。

```
Fetch 1->1:R Copy1      /* ‘1’を読んだら状態をCopy1に */
Copy1 B->B:R Copy1      /* ‘B’を読み飛ばす */
Copy1 C->1:R Mark1      /* ‘C’を ‘1’に変える */
Mark1 ␣->C:N Halt       /* 最後尾に ‘C’を書き込んで停止 */
```

次の命令セットを追加すれば、‘0’一文字のコピーもできる

```
Fetch 0->0:R Copy0      /* ‘0’を読んだら状態をCopy0に */
Copy0 B->B:R Copy0      /* ‘B’を読み飛ばす */
Copy0 C->1:R Mark0      /* ‘C’を ‘1’に変える */
Mark0 ␣->C:N Halt       /* 最後尾に ‘C’を書き込んで停止 */
```

複数の文字をコピーするには

先の命令セットでは、一文字のコピーしかできない。

複数の文字をコピーするには、最後尾の ‘C’ を書き込んだ後、再び A,Bの間にヘッドを戻して、次にコピーすべき一文字のFetchをする必要がある。

どこまでヘッドを戻すかだが、Fetchした時に、Fetch済みの目印の文字（例えば、‘X’とか ‘Y’ ）を書き込んでおくといい。最後尾のヘッドは、今度は、左に向かって、この目印まで戻る。そして、その次の文字をFetchする。

このヘッドが、最後尾からFetch済みの目印まで戻る状態を、Back0, Back1としよう。

A101BC --> A101B101C の実行と状態の変化の概略

```
Fetch:    A101BC -> AX01BC
Copy1:    AX01BC -> AX01B1
Mark1:    AX01B1 -> AX01B1C
Back1:    AX01B1C -> A101B1C
```

1文字目のコピー

A101BC --> A101B101C
の実行と状態の変化の概略

```
Fetch:    A101BC -> AX01BC
Copy1:    AX01BC -> AX01B1
Mark1:    AX01B1 -> AX01B1C
Back1:    AX01B1C -> A101B1C
Fetch:    A101B1C -> A1Y1B1C
Copy0:    A1Y1B1C -> A1Y1B10
Mark0:    A1Y1B10 -> A1Y1B10C
Back0:    A1Y1B10C -> A101B10C
```

2文字目のコピー

A101BC --> A101B101C
の実行と状態の変化の概略

```
Fetch:    A101BC -> AX01BC
Copy1:    AX01BC -> AX01B1
Mark1:    AX01B1 -> AX01B1C
Back1:    AX01B1C -> A101B1C
Fetch:    A101B1C -> A1Y1B1C
Copy0:    A1Y1B1C -> A1Y1B10
Mark0:    A1Y1B10 -> A1Y1B10C
Back0:    A1Y1B10C -> A101B10C
Fetch:    A101B10C -> A10XB10C
Copy1:    A10XB10C -> A10XB101
Mark1:    A10XB101 -> A10XB101C
Back1:    A10XB101C -> A101B101C
Fetch:    A101B101C
Halt
```

3文字目のコピー

複数の文字をコピーする命令セット

先のCopy0,Copy1にしろ、このBack0,Back1にしろ、特定の文字（前者は ‘C’、後者は ‘X’あるいは ‘Y’）までヘッドを移動する命令があると便利である。（ここでは、それを導入してみたが、その意味については、各人で考えてみよ。）

```
Fetch 0->X:R Copy0
Fetch 1->Y:R Copy1
Fetch B-B :N HALT
Copy0 [C]?->0:R,R Mark0
Copy1 [C]?->1:R,R Mark1
Mark0 ␣->C:N Back0
Mark1 ␣->C:N Back0
Back0 [X]?->1:L,R Fetch
Back1 [Y]?->0:L,R Fetch
```

複数の文字をコピーする命令セットへのコメント

- 状態：Fetch /*文字列から一文字読み取る。*/
  - ‘0’であれば、それを’X’に書き換え、状態をCopy0に変え、右に進む
  - ‘1’であれば、それを’Y’に書き換え、状態をCopy1に変え、右に進む
  - ‘B’であれば、コピー終了　/ それ以外なら、状態を変えず、右に進む
- 状態： Copy* /* ヘッドを’C’が見つかるまで右移動する。*/
  - ‘C’以外なら、状態を変えず、右に進む
  - ‘C’で、状態がCopy0であれば、その位置に’0’を書き込み、状態をMark0に変え、右に進む
  - ‘C’で、状態がCopy1であれば、その位置に’1’を書き込み、状態をMark1に変え、右に進む
- 状態：Mark* /* ’C’を書き込み、状態をBack0 / Back1 に変える。 */（略）
- 状態：Back* /* ヘッドを、’X’あるいは’Y’が見つかるまで戻す。*/
  - ‘X’, ‘Y’ 以外なら、左に一つ進む
  - ‘X’で、状態がBack0であれば、’X’の位置に’0’を書き込み、状態をFetchに変えて、右に一つ進む
  - ‘Y’で、状態がBack1であれば、’Y’の位置に’1’を書き込み、状態をFetchに変えて、右に一つ進む

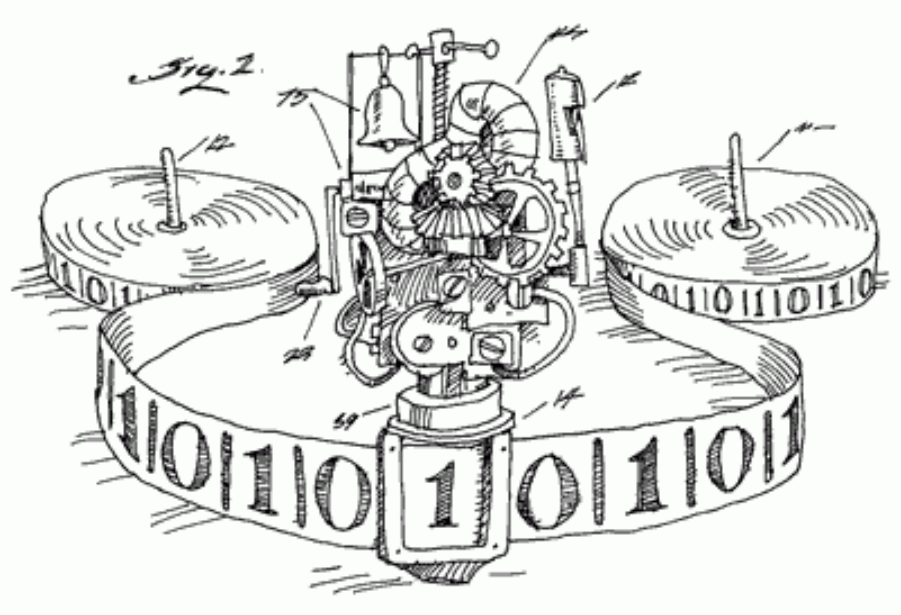
Part III

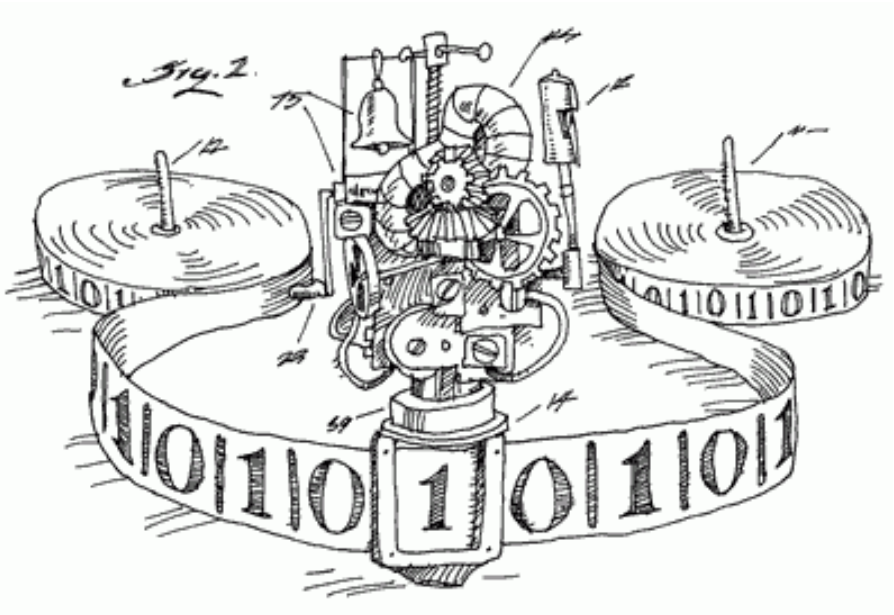
複数のテープと複数のマシン

Agenda Part III
複数のテープと複数のマシン

- 複数のテープを持つチューリングマシン
  - 複数のテープに対応して命令を拡張する
  - テープ1のビット列をテープ2にコピーする
  - テープ1のビット列がテープ2のビット列に等しいかをチェックする
- 2つのチューリングマシンを考える
- ControllerとExecuter -- 1ステップ実行 –
  - 一台のコントローラと一台の実行マシン
  - ビット列コピーの例で、Controller, Executerの命令セットを考える
  - 命令の実行を一時停止する

複数のテープを持つチューリングマシン





「チューリングマシンを学ぼう！」

Part III　複数のテープと複数のマシン

複数のテープに対応して
命令を拡張する

チューリングマシンの「命令」を構成する5つの要素

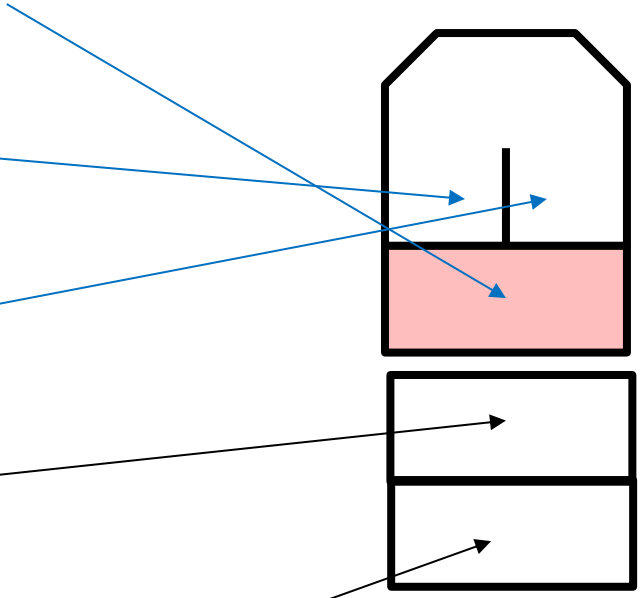
- チューリングマシンの命令を、次の5つの要素から構成される。

1. 現在のチューリングマシンの状態
2. ヘッド位置のテープ上の一文字
3. ヘッド位置に書き出される一文字
4. ヘッドが次に進む方向 (R, L, N)
5. 次のステップでのチューリングマシンの状態

チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

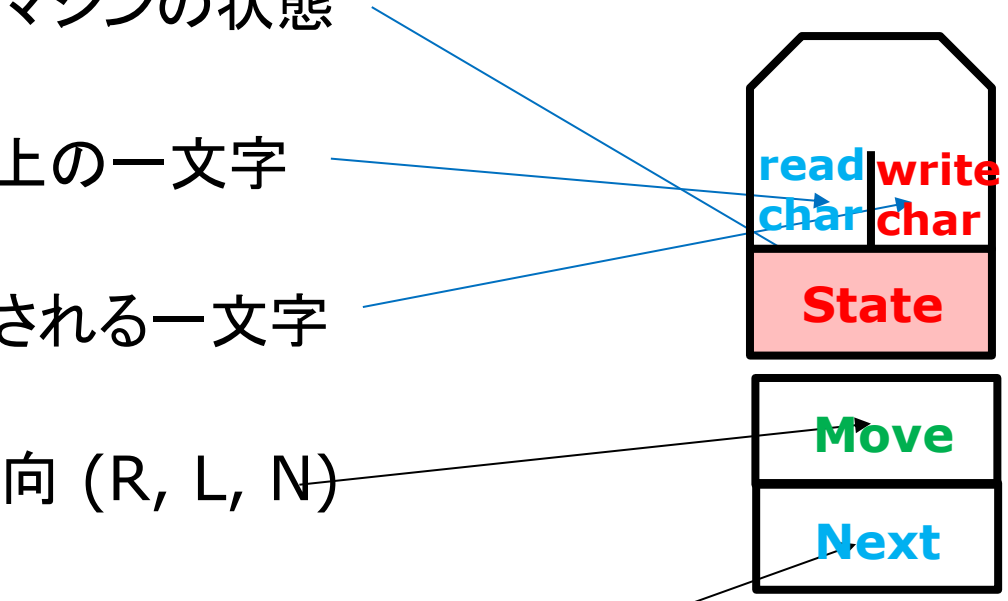
1. 現在のチューリングマシンの状態
2. ヘッド位置のテープ上の一文字
3. ヘッド位置に書き出される一文字
4. ヘッドが次に進む方向 (R, L, N)
5. 次のステップでのチューリングマシンの状態



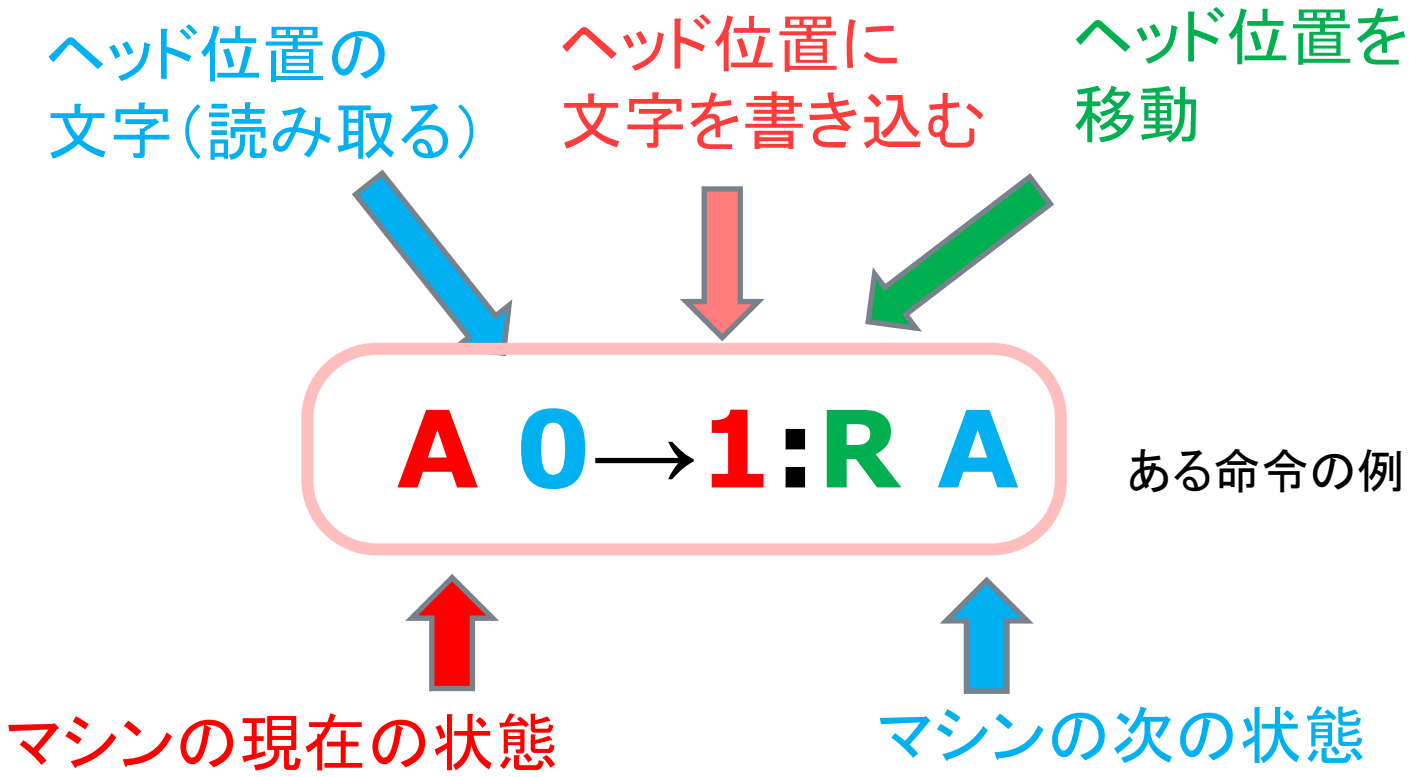
チューリングマシンの「命令」を図で表す

- チューリングマシンの命令を、次のように図で表すことができる。

1. 現在のチューリングマシンの状態
2. ヘッド位置のテープ上の一文字
3. ヘッド位置に書き出される一文字
4. ヘッドが次に進む方向 (R, L, N)
5. 次のステップでのチューリングマシンの状態

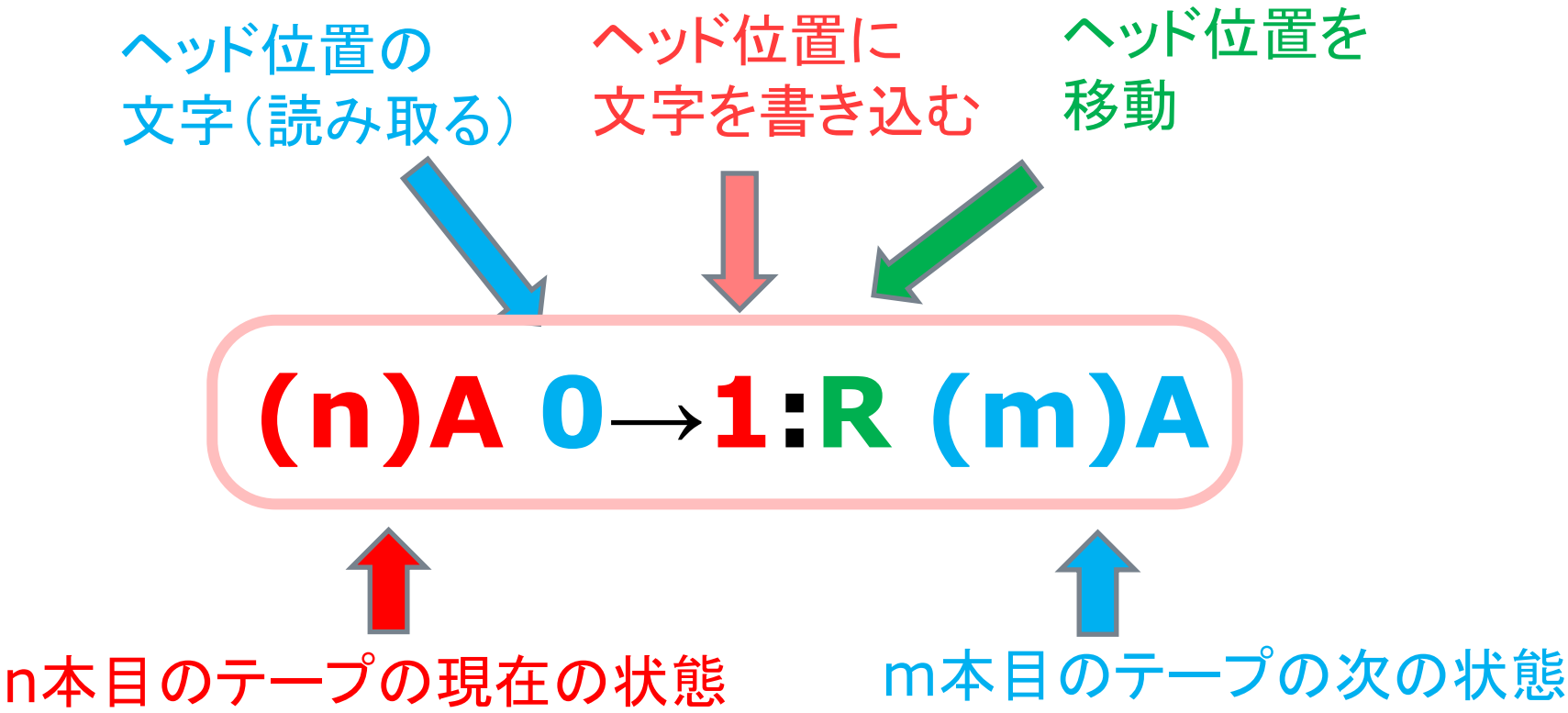


文字列で命令を表す



現在の状態が**A**でヘッドが**0**の上にあるなら、ヘッド位置に**1**を書き込み、ヘッドを右に進め、状態を**A**にする。

複数のテープに対応して拡張された命令



n本目のテープの現在の状態が**A**でヘッドが**0**の上にあるなら、ヘッド位置に**1**を書き込み、ヘッドを右に進め、m本目のテープの状態を**A**にする。

テープ1のビット列をテープ2にコピーする命令セット（プログラム）

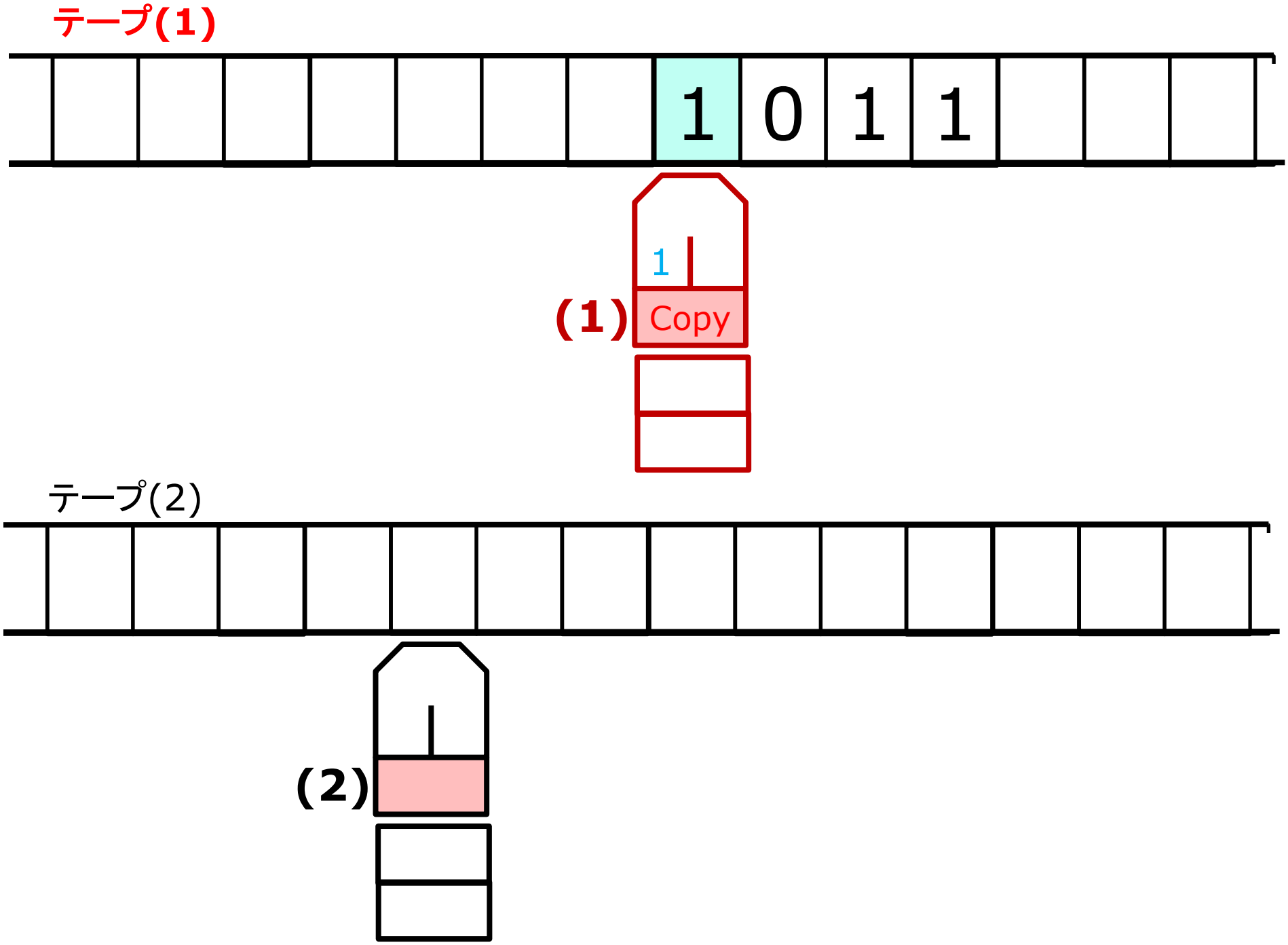
/* 一本目のテープの文字を読み取り */
/* 二本目のテープに制御を移す　　*/

1. (1)Copy 0→0:R (2)Paste0
2. (1)Copy 1→1:R (2)Paste1
3. (1)Copy ␣→␣:N (1)Copy-End

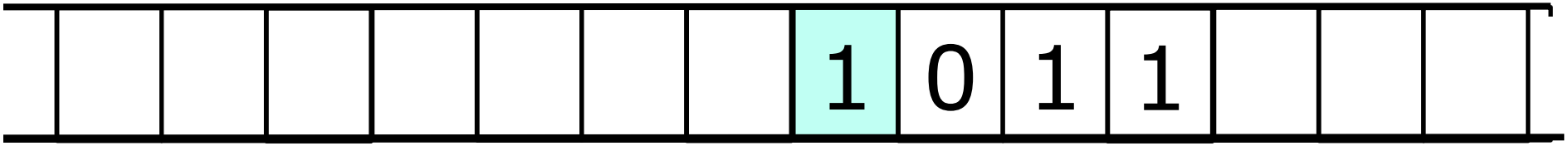
/* 二本目のテープに一本目のテープの　*/
/* 文字を書き込み、一本目に制御を返す */

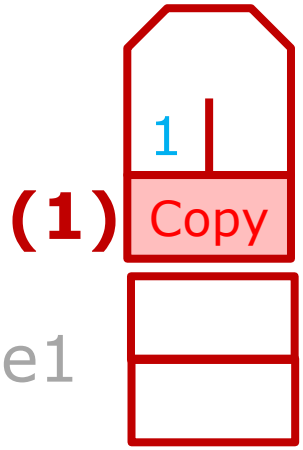
4. (2)Paste0 ␣→0:R (1)Copy
5. (2)Paste1 ␣→1:R (1)Copy

テープ1のビット列を
テープ2にコピーする


テープ(1)
1
0
1
1
1
(1)
Copy
テープ(2)
(2)

テープ(1)

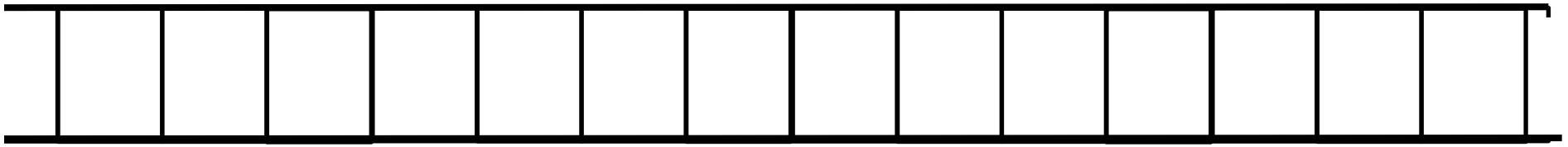


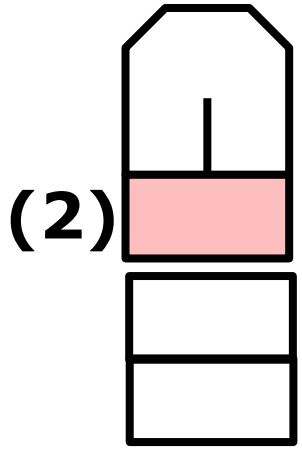


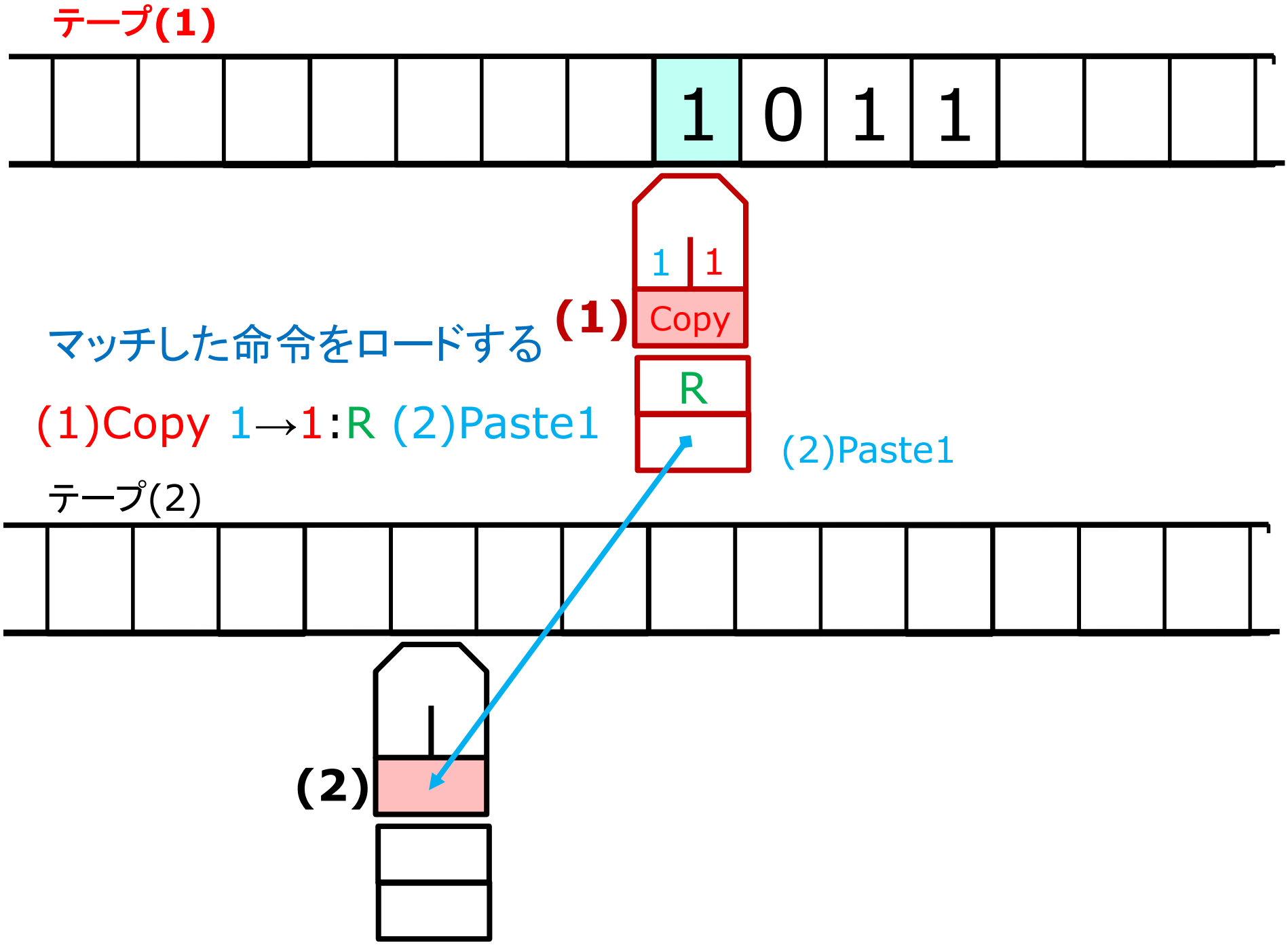
マッチする命令を探す

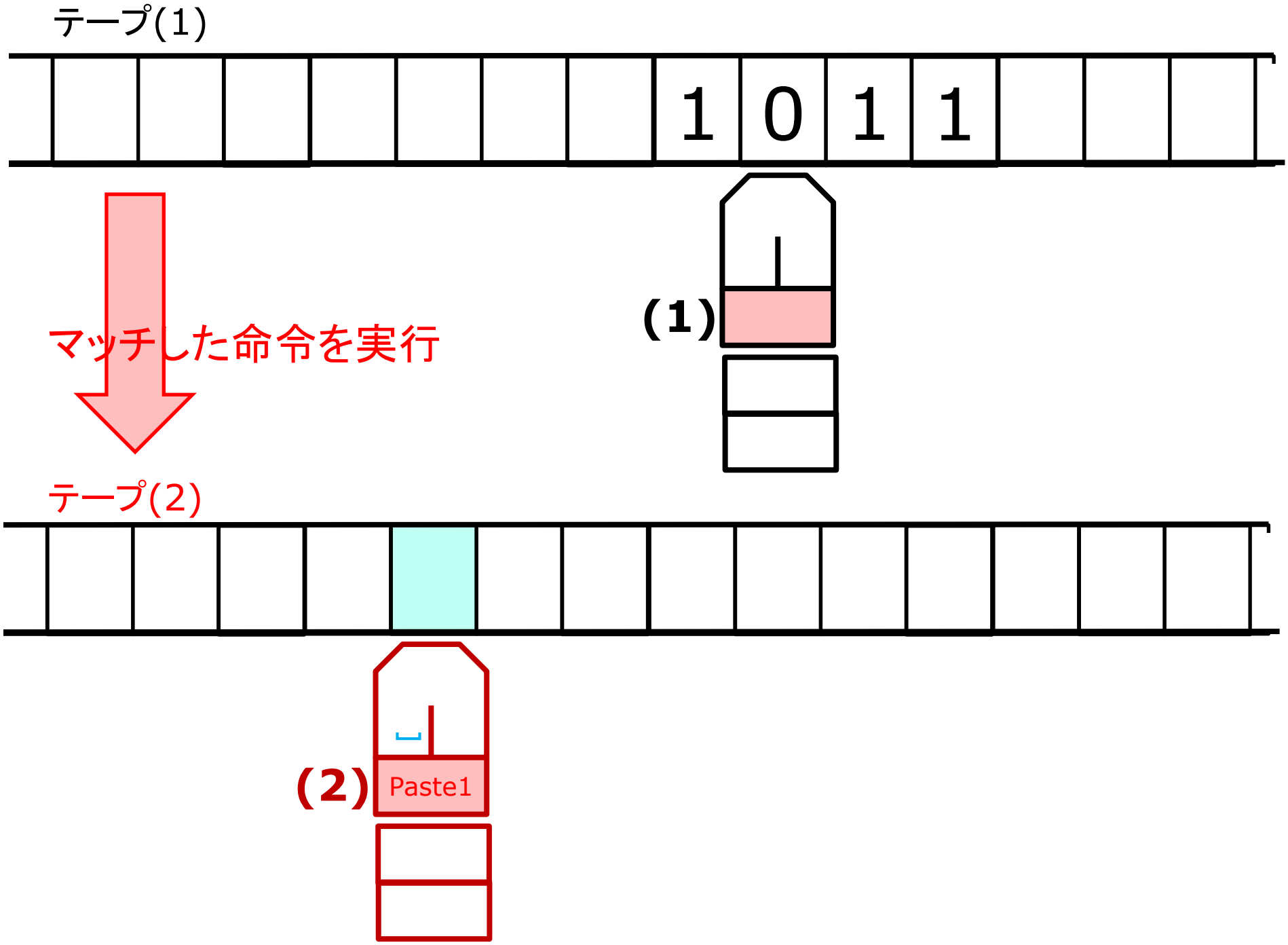
(1)Copy 1→1:R (2)Paste1

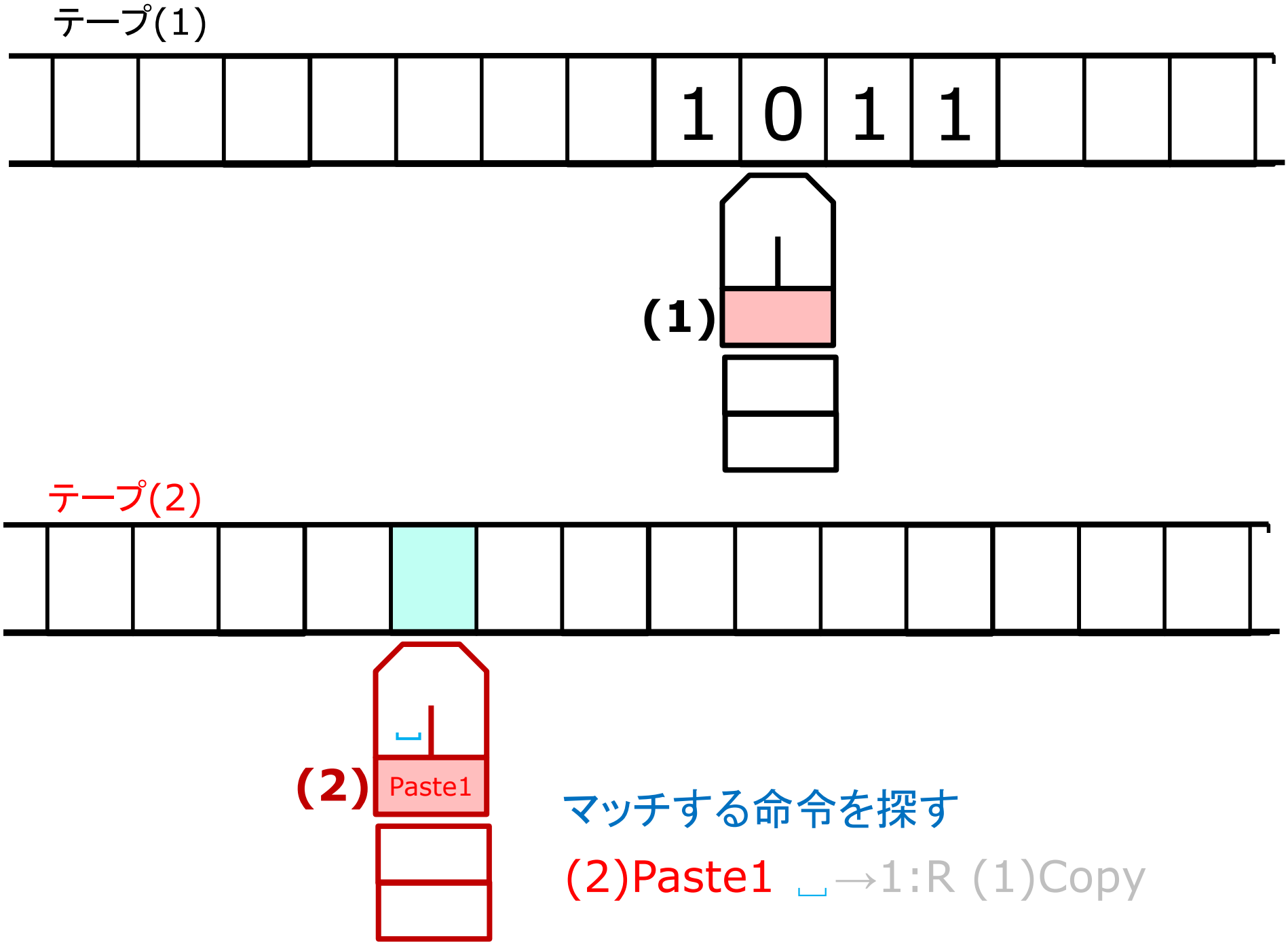
テープ(2)

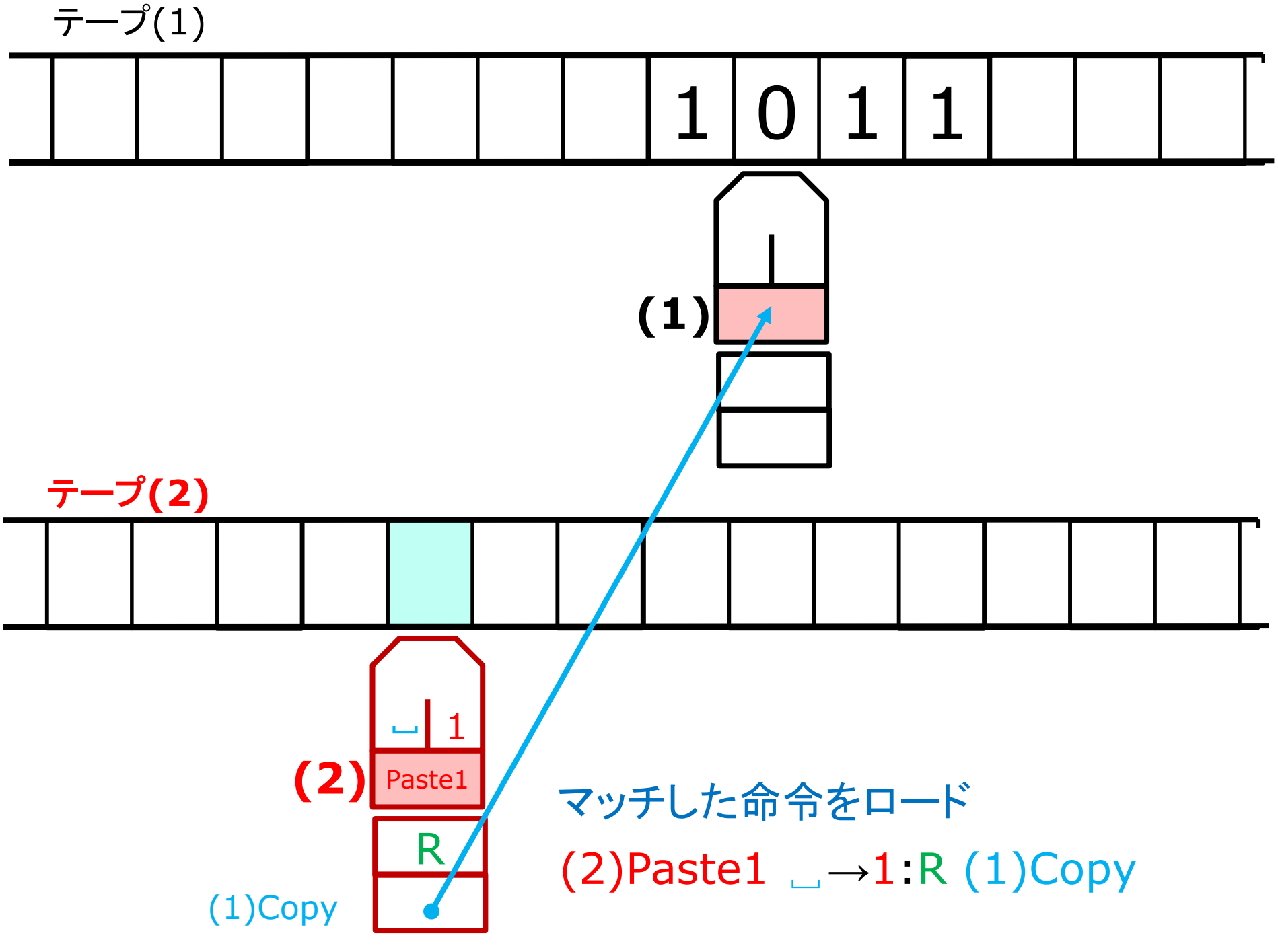


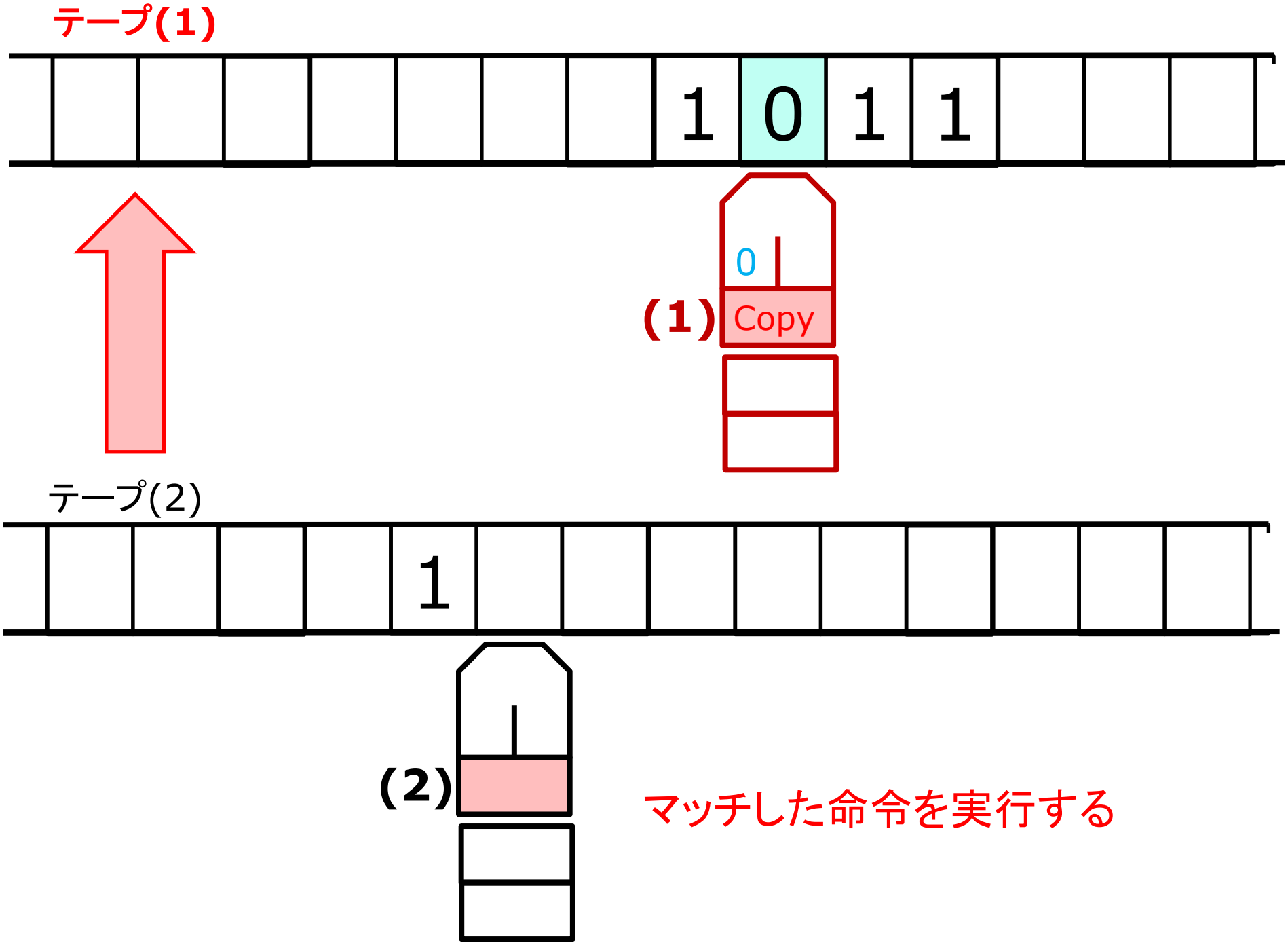



テープ(1)
1 0 1 1
1 1
(1)
Copy
R
マッチした命令をロードする
(1)Copy 1→1:R (2)Paste1
(2)Paste1
テープ(2)
(2)

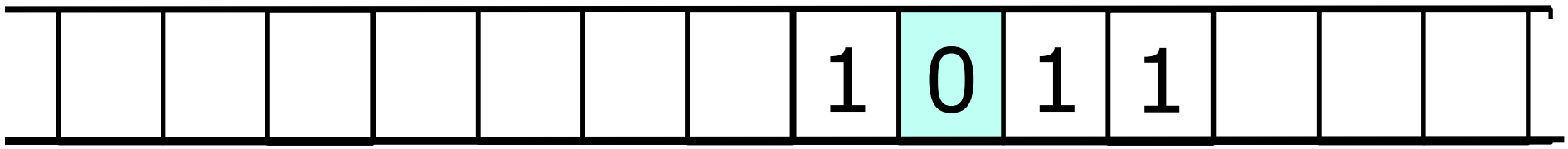

テープ(1)
1 0 1 1
(1)
マッチした命令を実行
テープ(2)
␣
(2)
Paste1

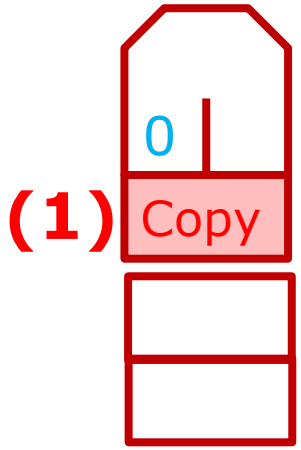

テープ(1)
1
0
1
1
(1)
テープ(2)
(2)
Paste1
マッチする命令を探す
(2)Paste1 ␣→1:R (1)Copy


テープ(1)
1 0 1 1
(1)
テープ(2)
␣ 1
(2)
Paste1
R
(1)Copy
マッチした命令をロード
(2)Paste1 ␣→1:R (1)Copy


テープ(1)
1
0
1
1
0
(1)
Copy
テープ(2)
1
(2)
マッチした命令を実行する

テープ(1)

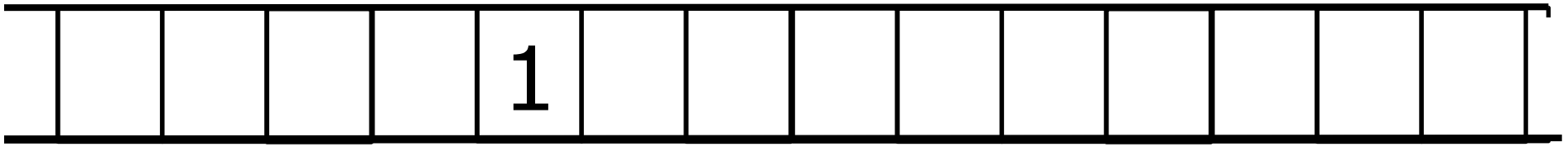


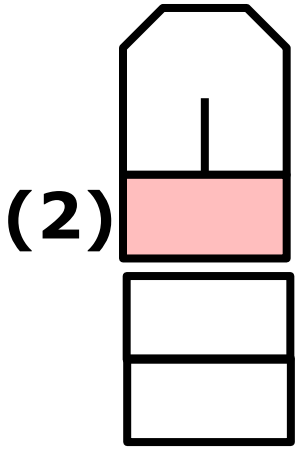


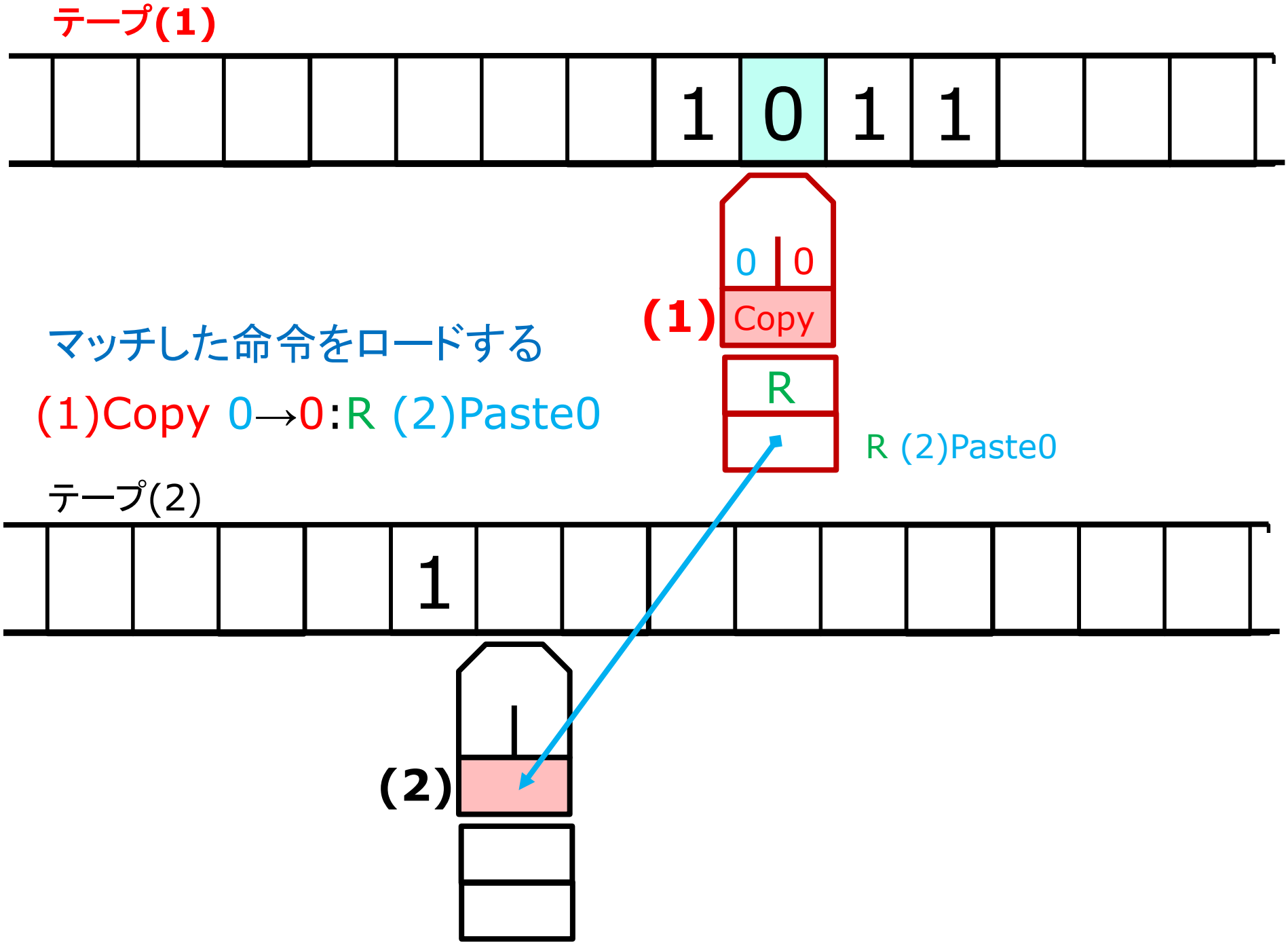
マッチする命令を探す

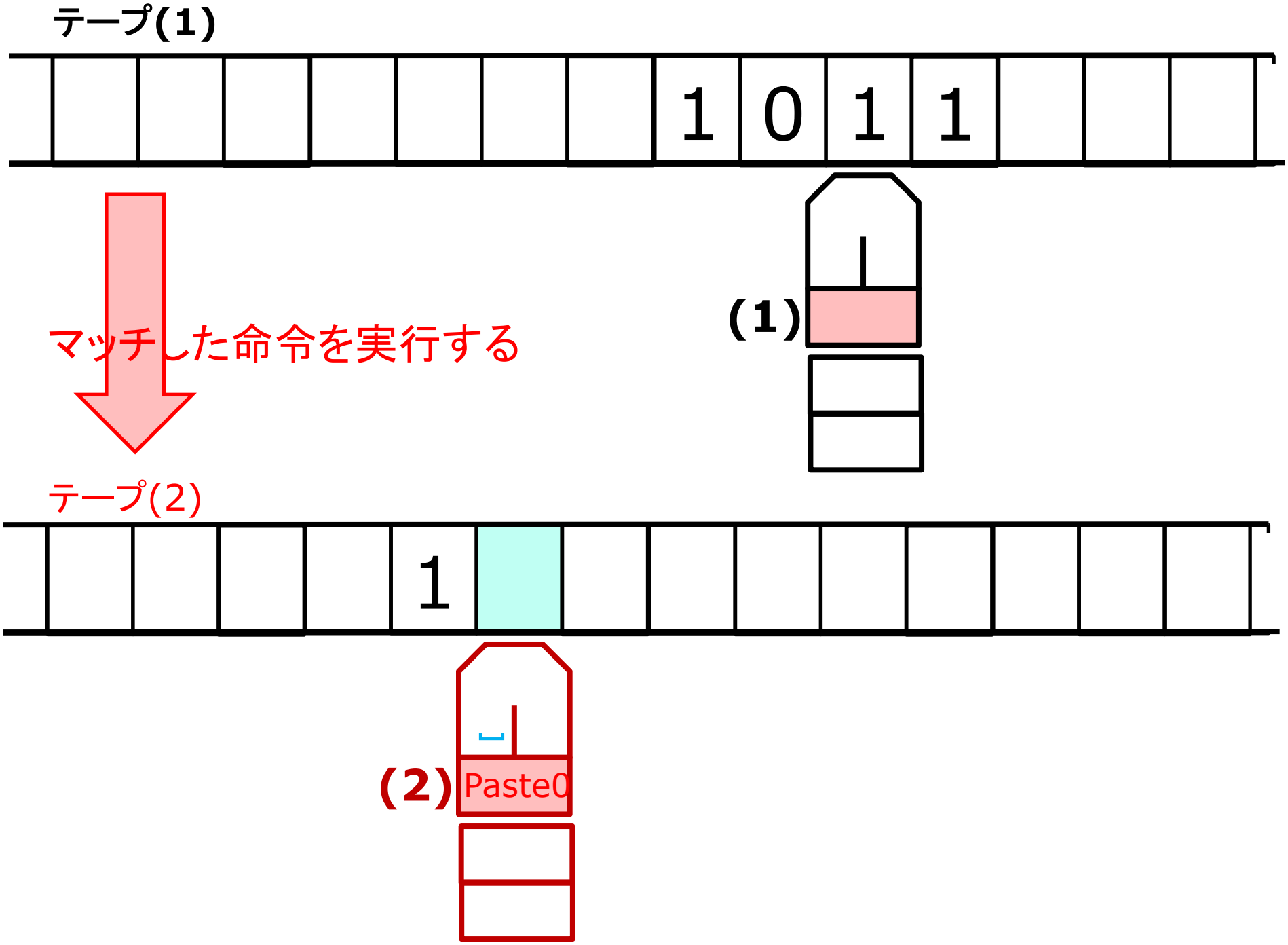
(1)Copy 0→0:R (2)Paste0

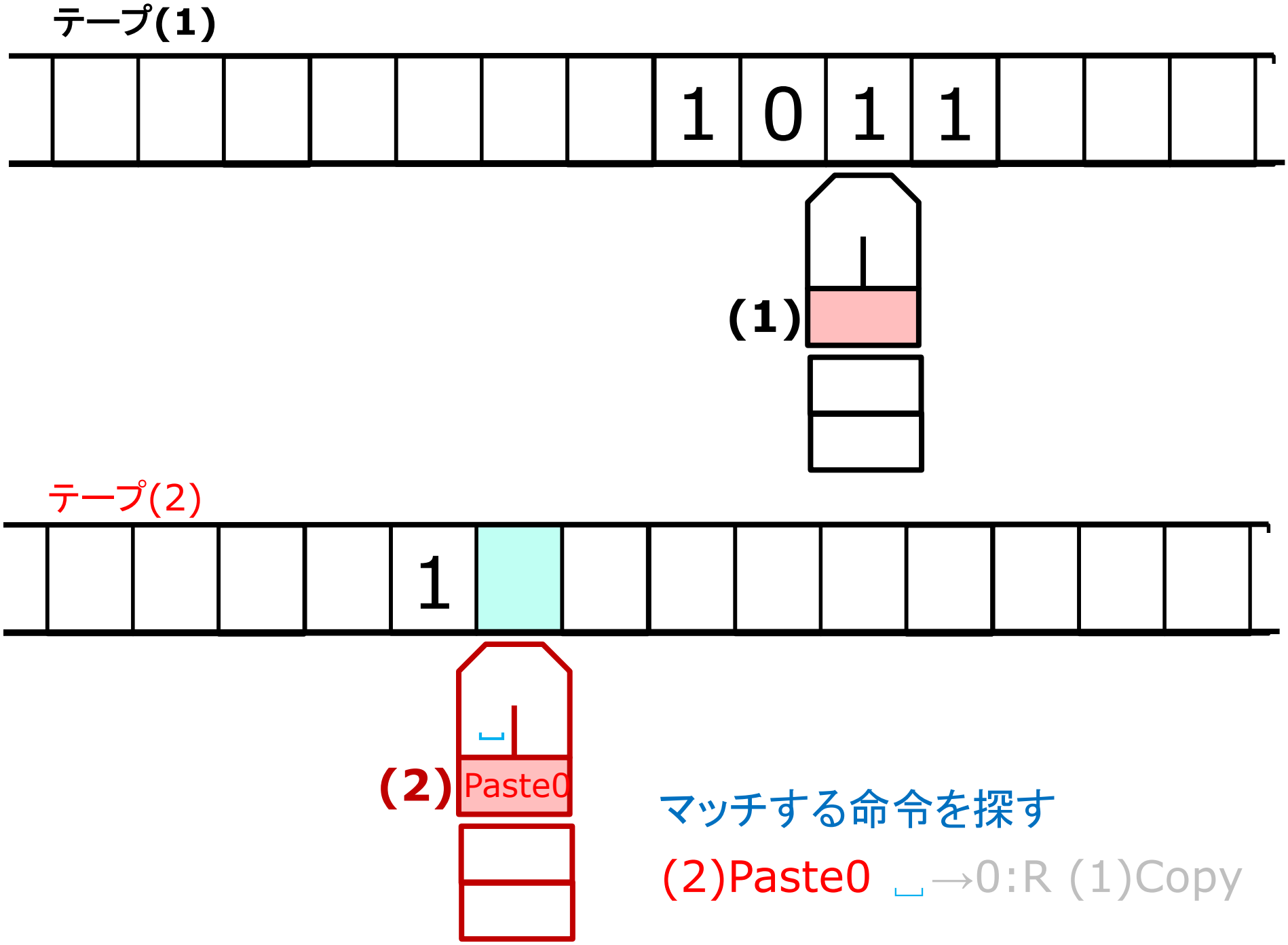
テープ(2)

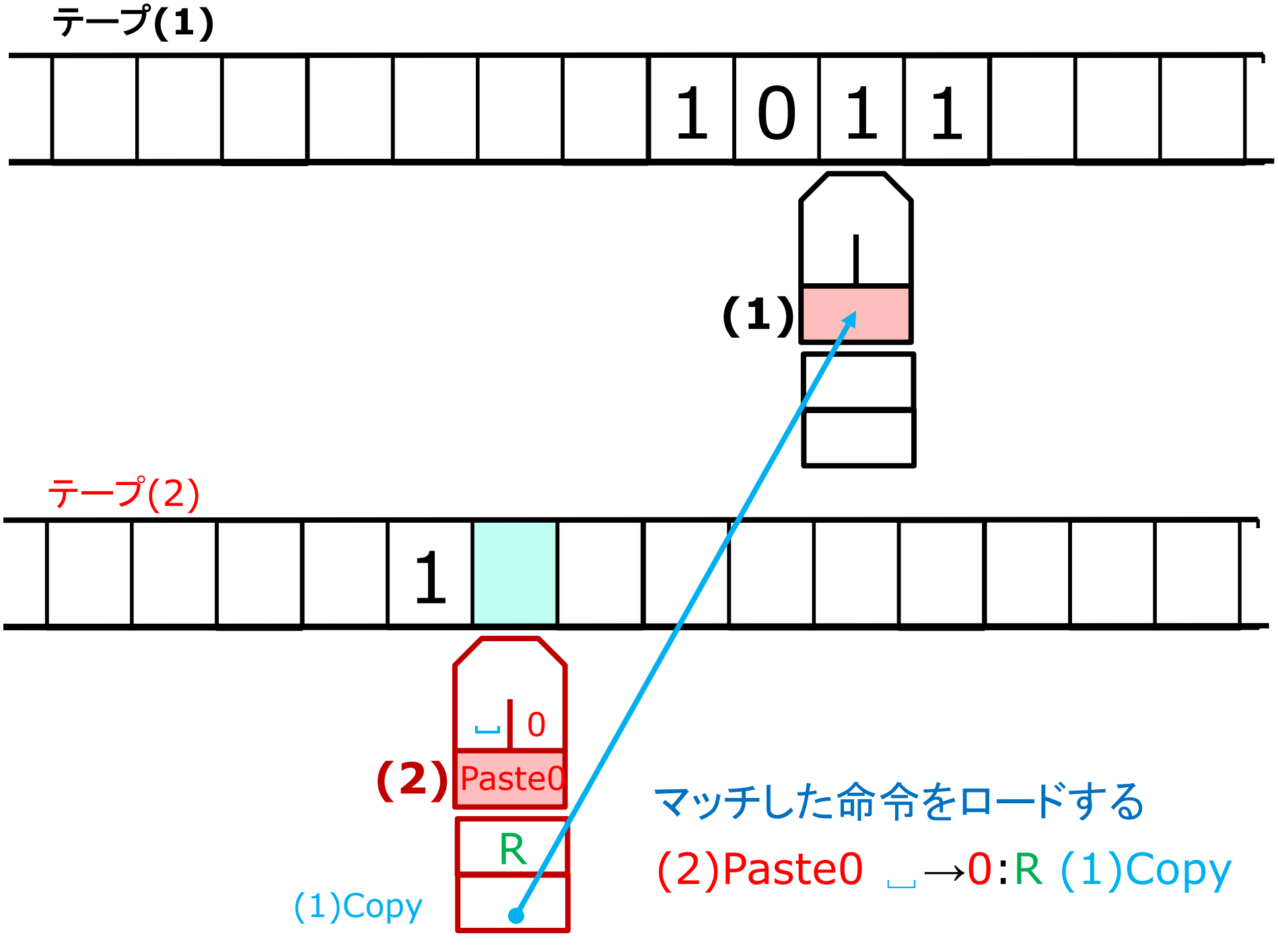


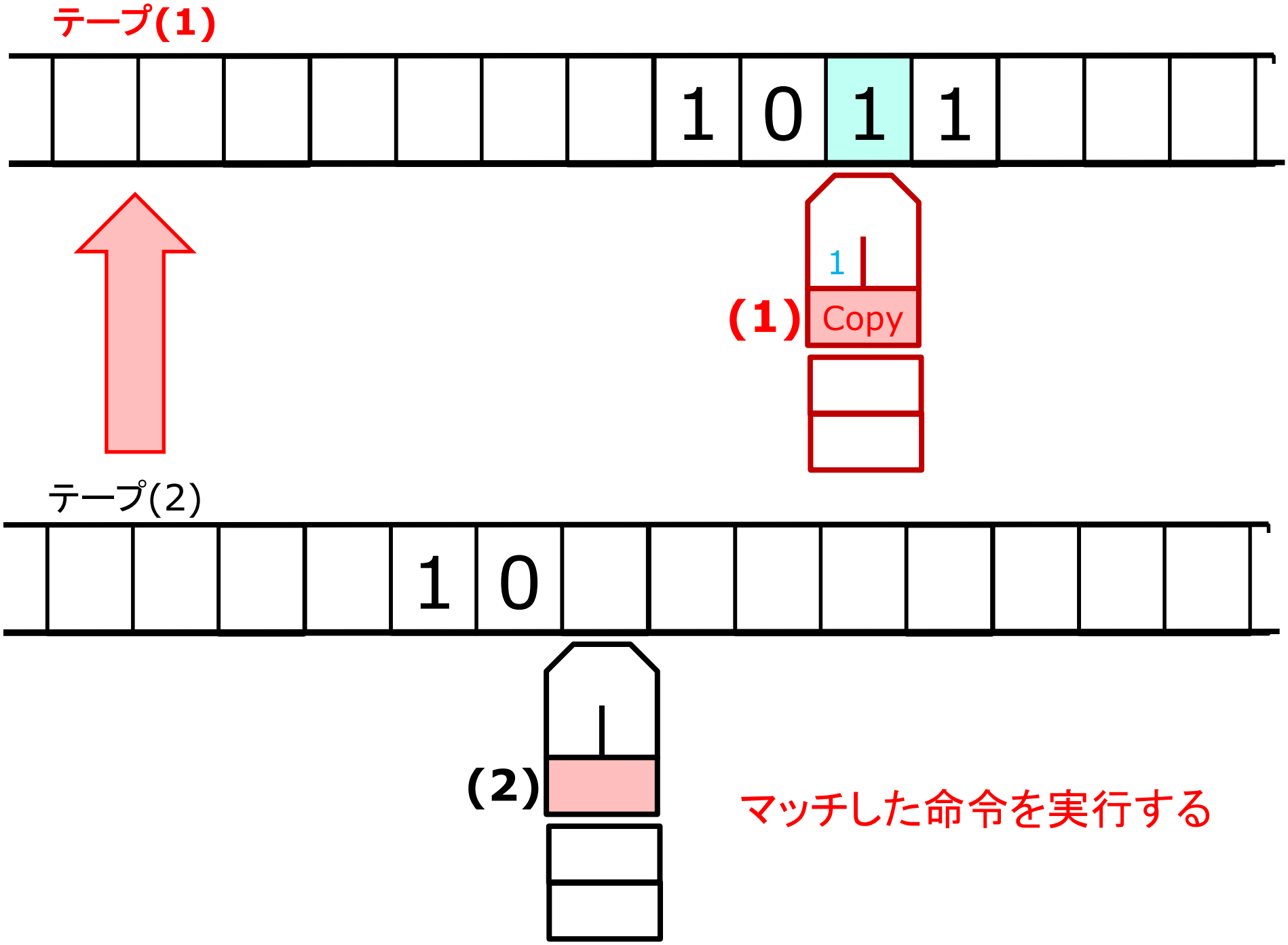



テープ(1)
1
0
1
1
0
0
(1)
Copy
R
R (2)Paste0
マッチした命令をロードする
(1)Copy 0→0:R (2)Paste0
テープ(2)
1
(2)

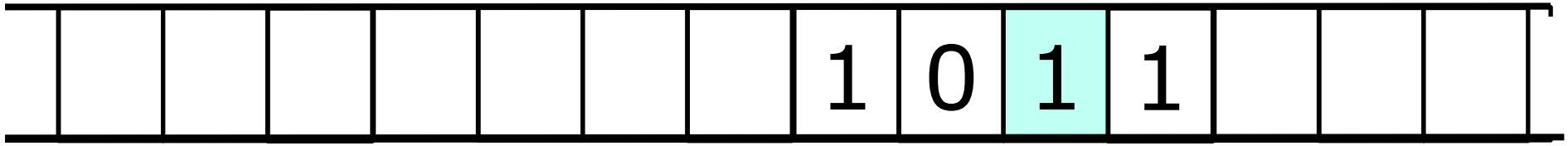

テープ(1)
1 0 1 1
(1)
マッチした命令を実行する
テープ(2)
1
(2)
Paste0

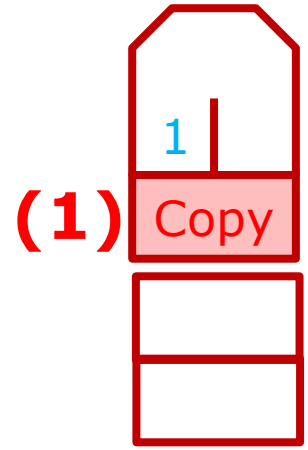

テープ(1)
1 0 1 1
(1)
テープ(2)
1
(2) Paste0
␣
マッチする命令を探す
(2)Paste0 ␣→0:R (1)Copy


テープ(1)
1 0 1 1
(1)
テープ(2)
1
␣ 0
(2) Paste0
R
(1)Copy
マッチした命令をロードする
(2)Paste0 ␣→0:R (1)Copy


テープ(1)
1 0 1 1
1
(1)
Copy
テープ(2)
1 0
(2)
マッチした命令を実行する

テープ(1)

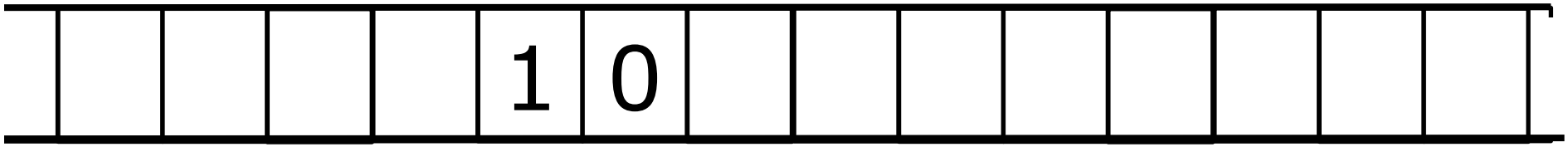


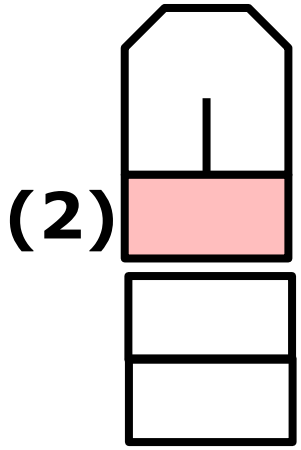


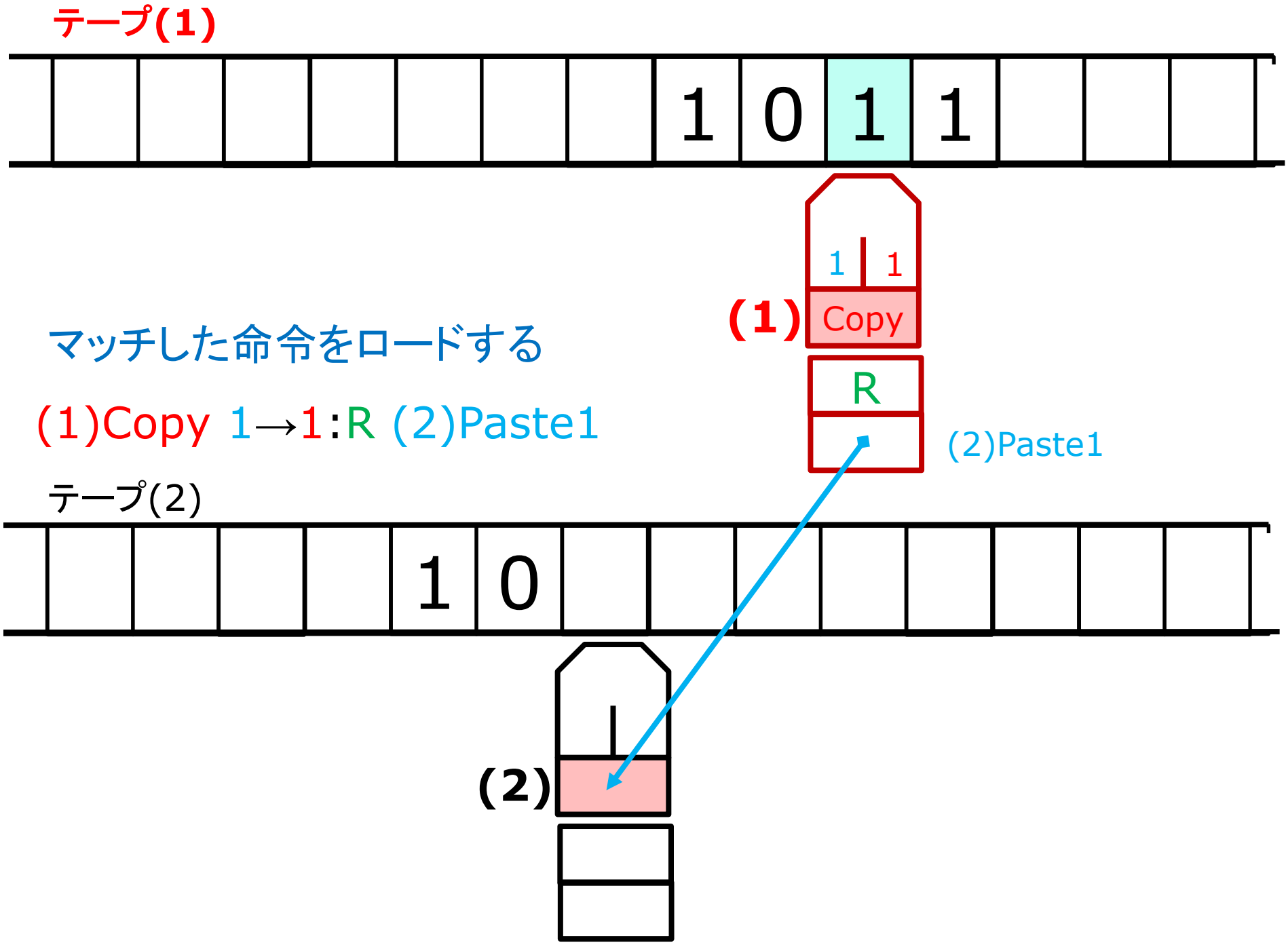
マッチする命令を探す

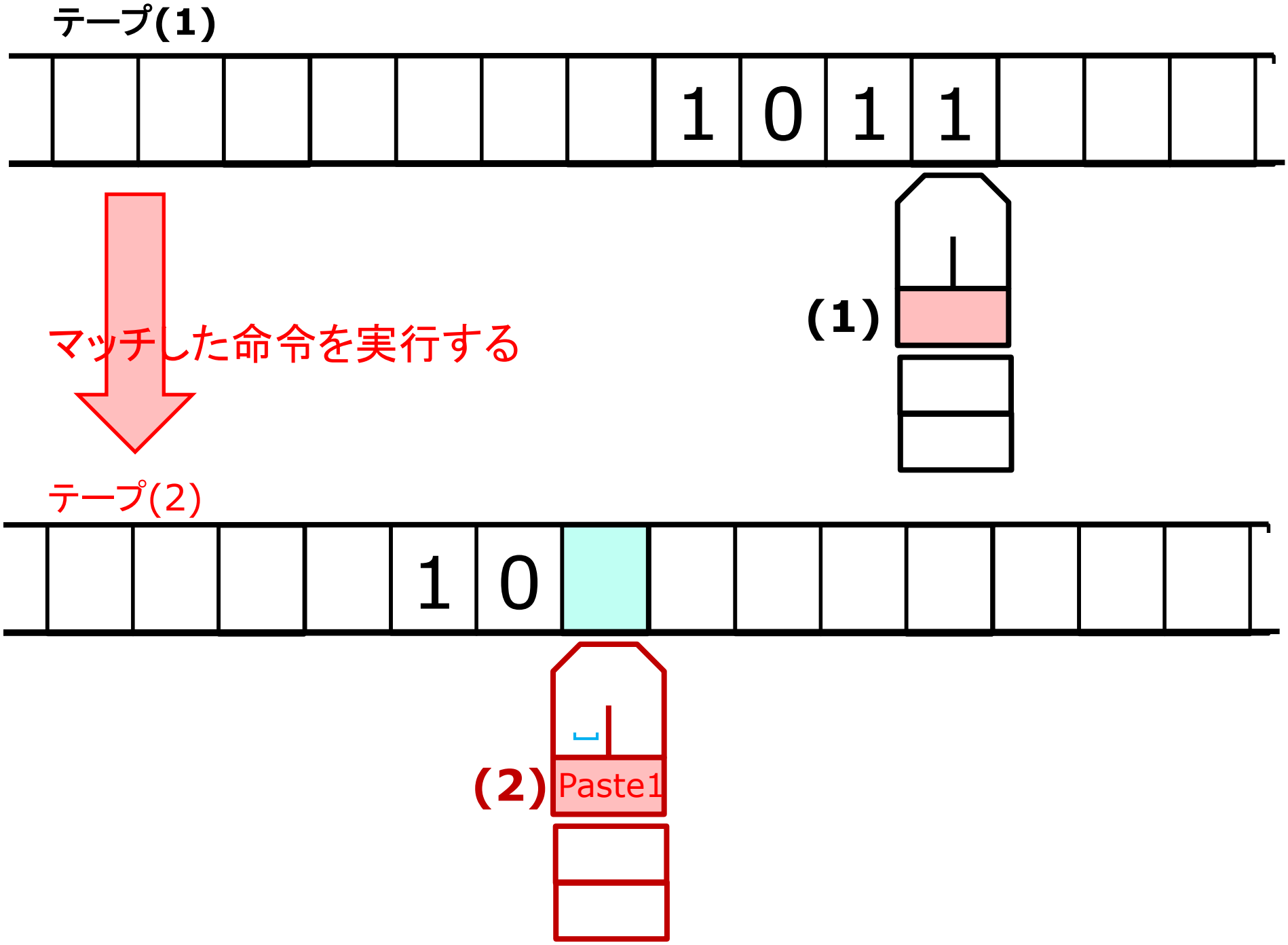
(1)Copy 1→1:R (2)Paste1

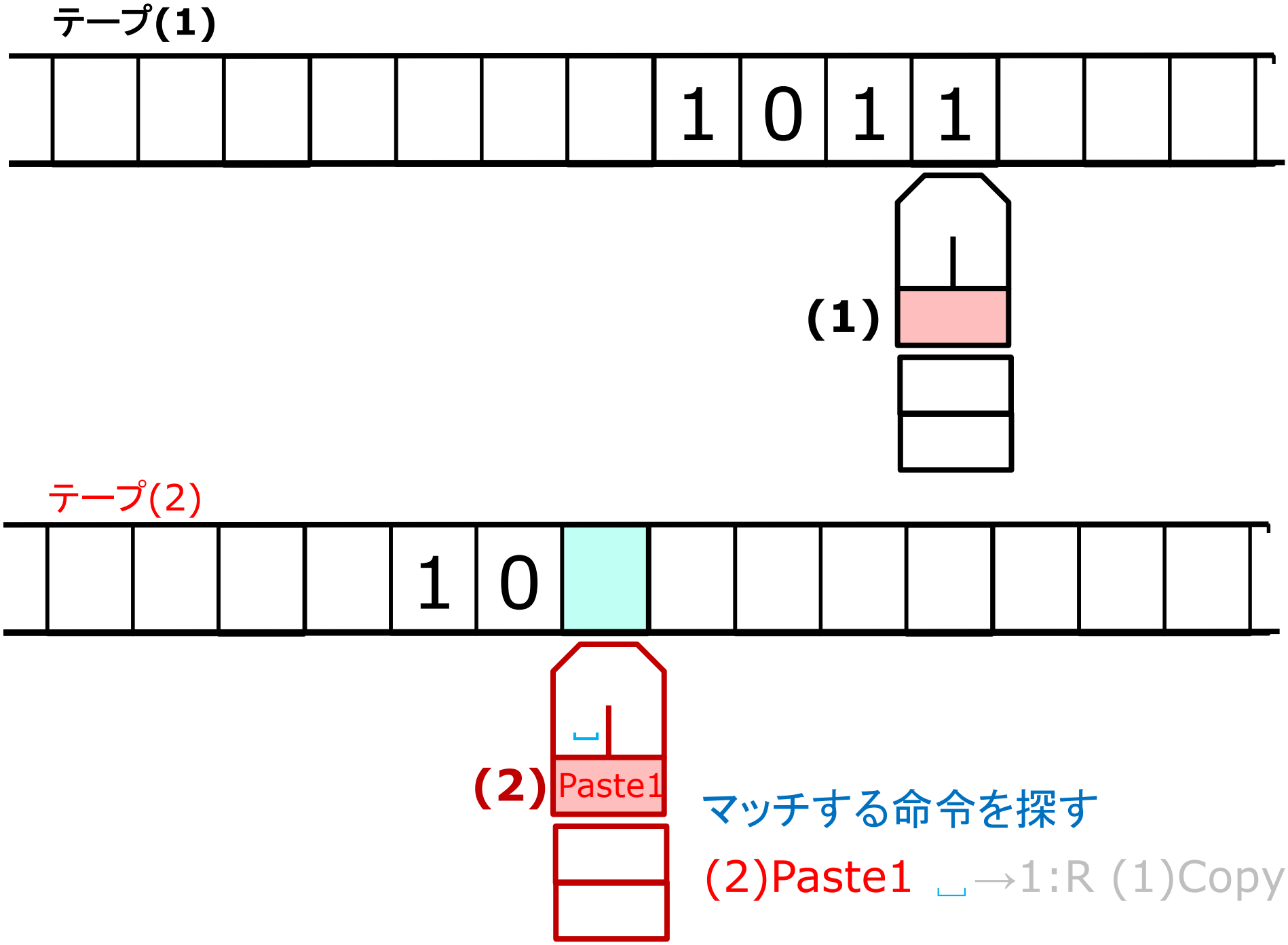
テープ(2)

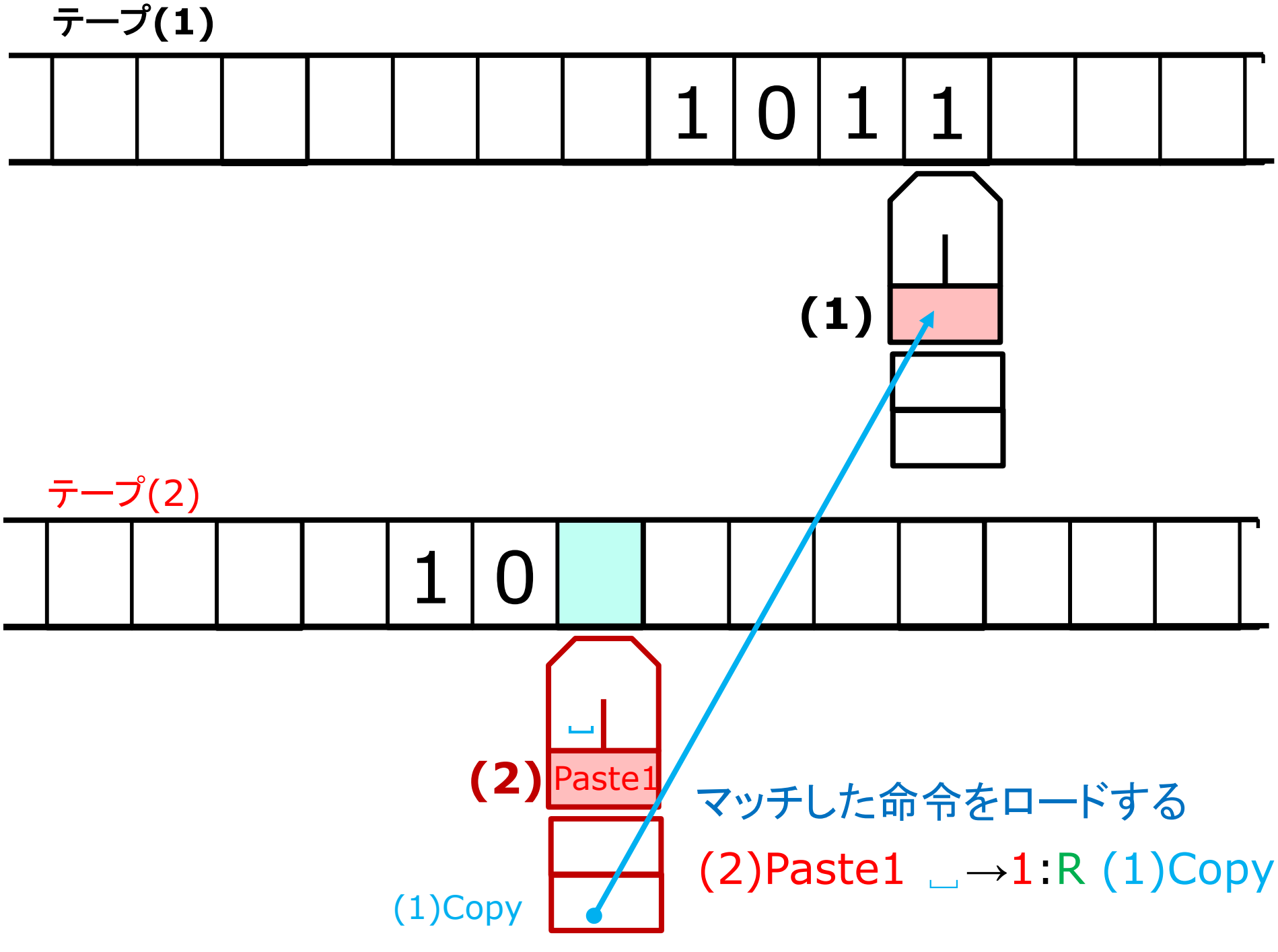


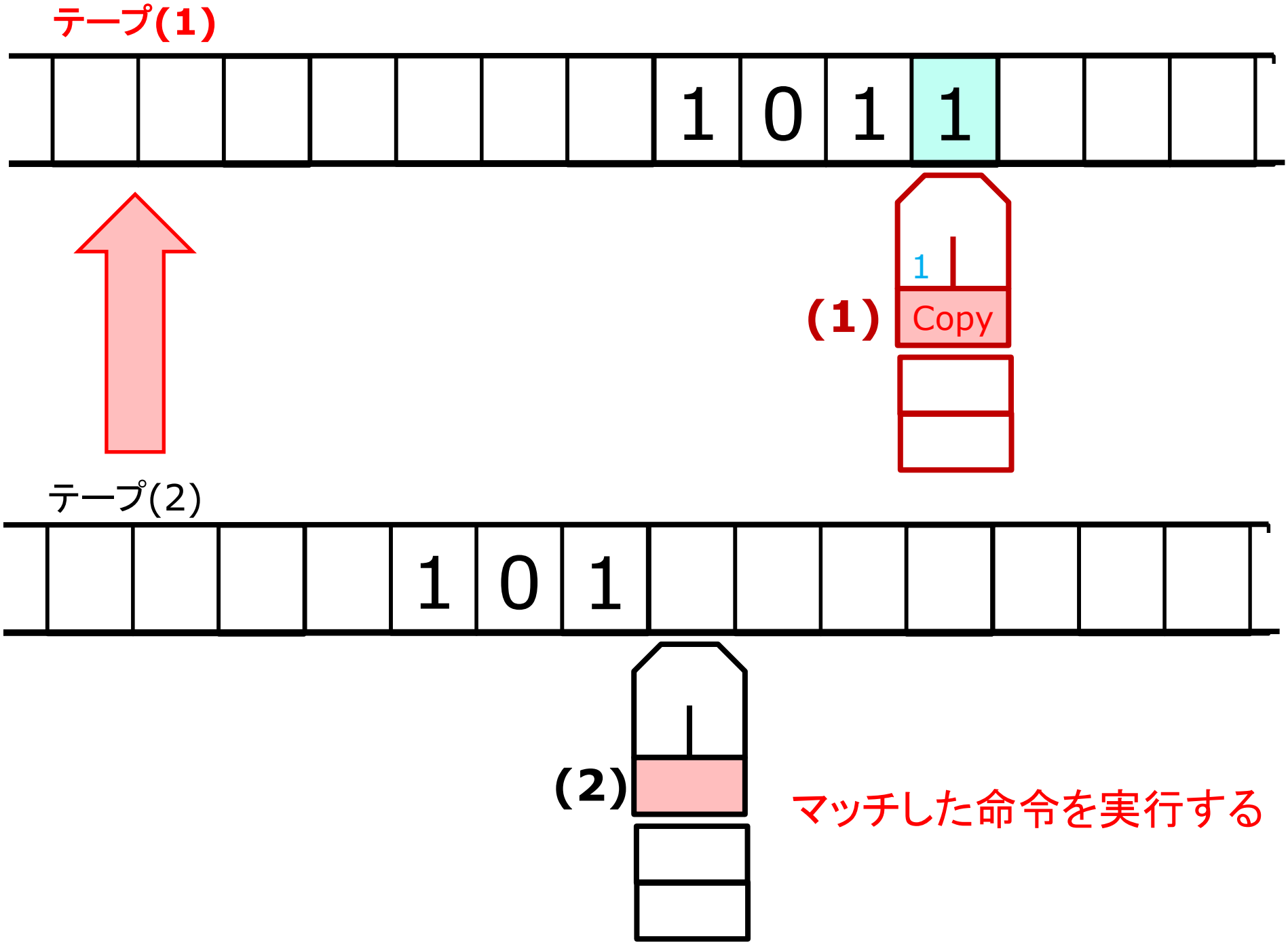



テープ(1)
1 0 1 1
1 1
(1) Copy
R
(2)Paste1
マッチした命令をロードする
(1)Copy 1→1:R (2)Paste1
テープ(2)
1 0
(2)

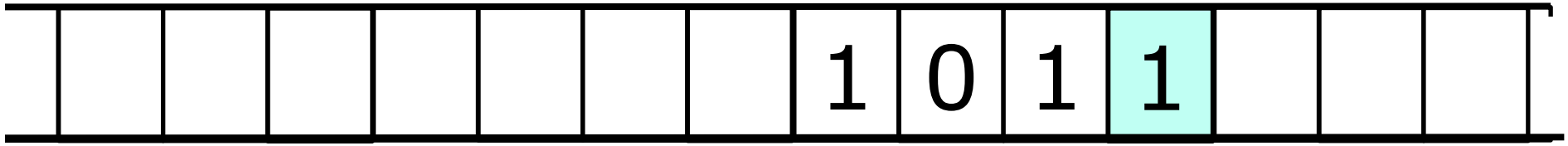

テープ(1)
1 0 1 1
(1)
マッチした命令を実行する
テープ(2)
1 0
(2)
Paste1

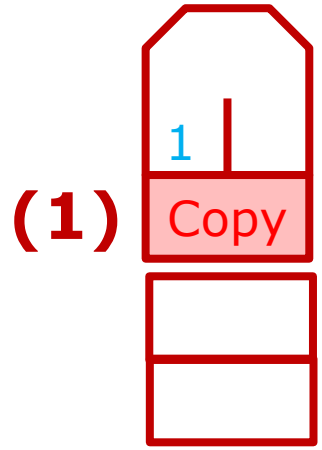

テープ(1)
1 0 1 1
(1)
テープ(2)
1 0
(2) Paste1
マッチする命令を探す
(2)Paste1 ␣ →1:R (1)Copy


テープ(1)
1 0 1 1
(1)
テープ(2)
1 0
(2) Paste1
(1)Copy
マッチした命令をロードする
(2)Paste1 ␣ →1:R (1)Copy


テープ(1)
1 0 1 1
1
(1)
Copy
テープ(2)
1 0 1
(2)
マッチした命令を実行する

テープ(1)

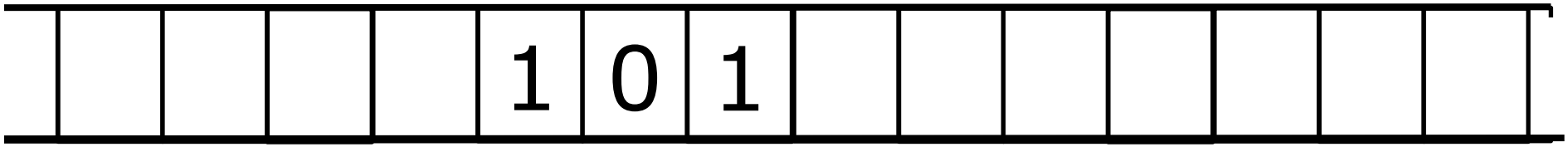


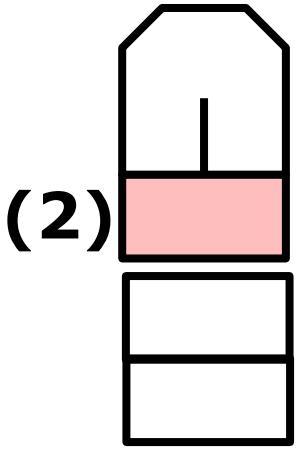


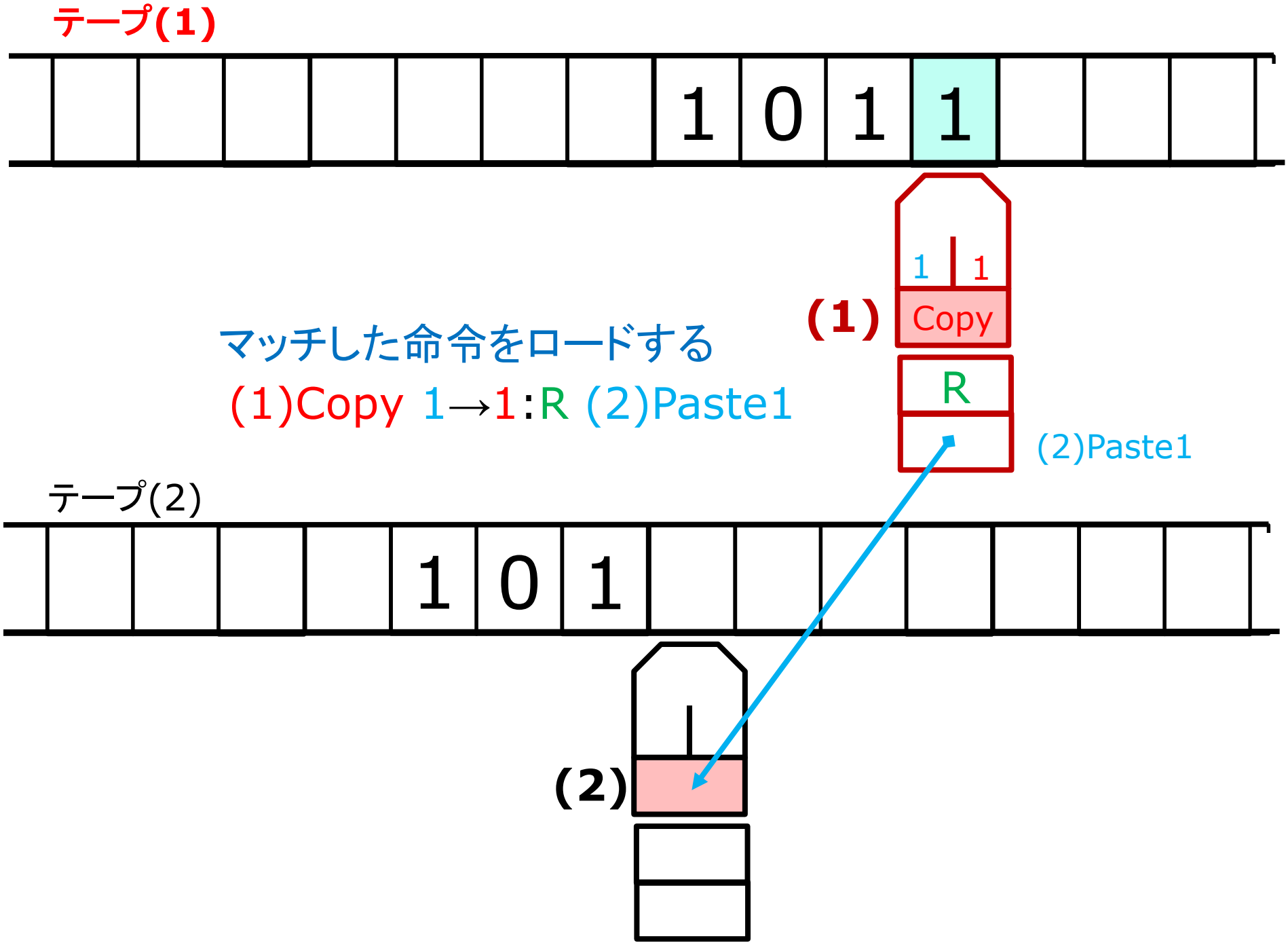
マッチする命令を探す

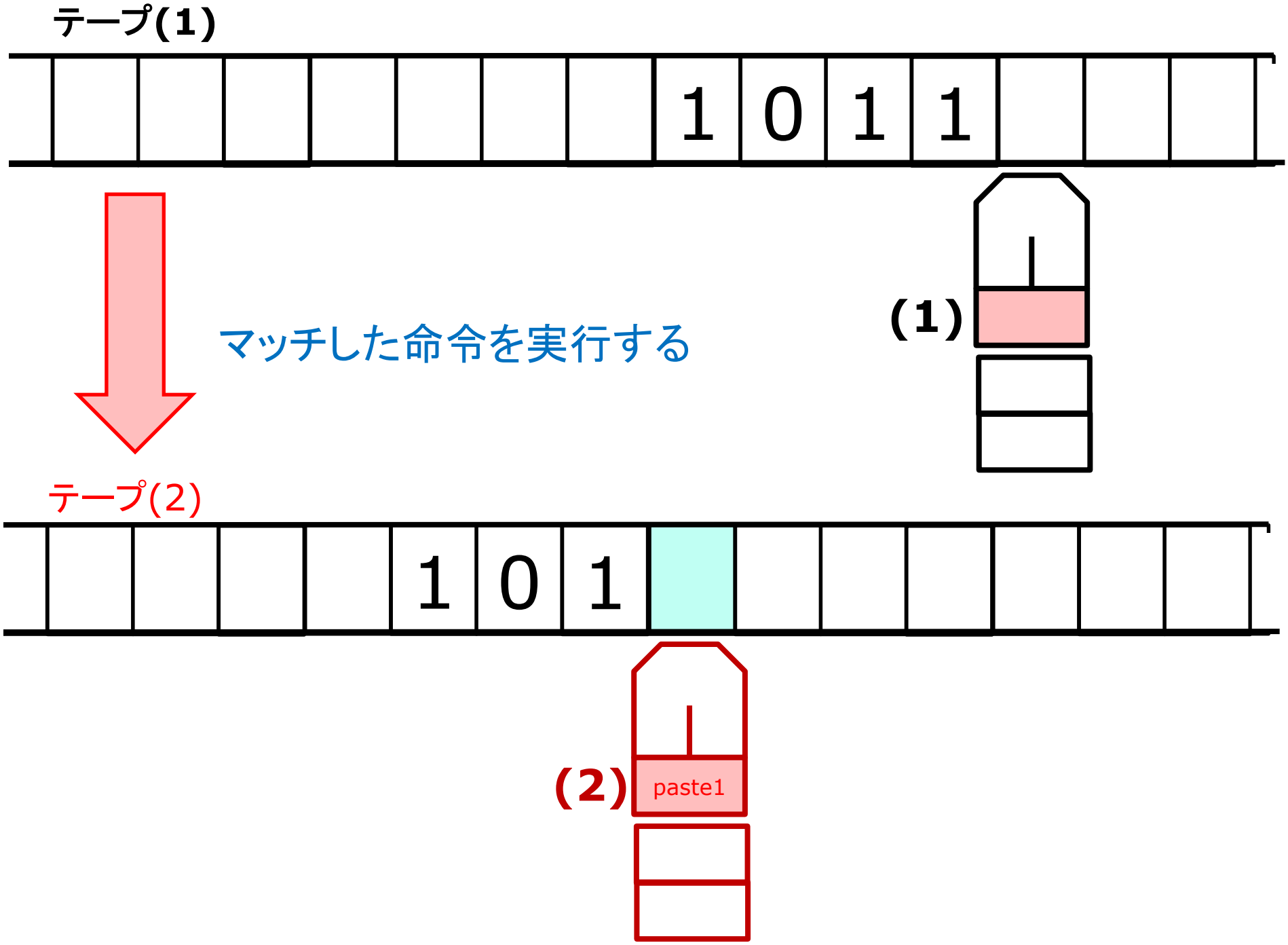
(1)Copy 1→1:R (2)Paste1

テープ(2)

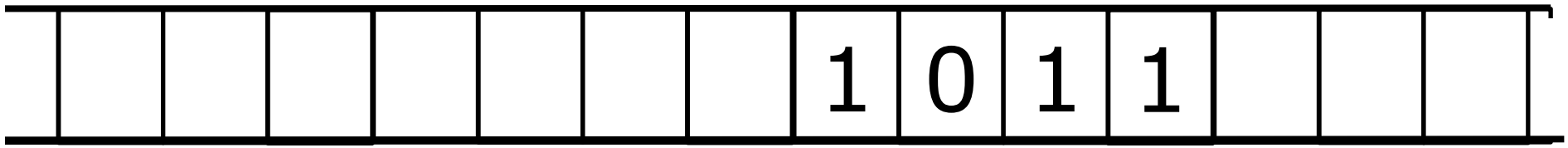


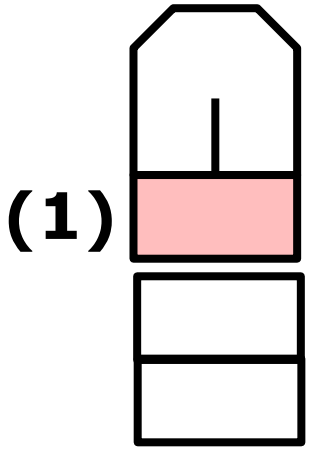



テープ(1)
1 0 1 1
1 1
(1)
Copy
R
(2)Paste1
マッチした命令をロードする
(1)Copy 1→1:R (2)Paste1
テープ(2)
1 0 1
(2)

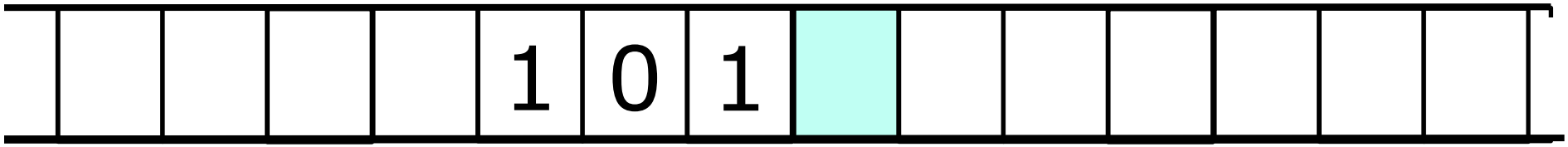

テープ(1)
1 0 1 1
(1)
マッチした命令を実行する
テープ(2)
1 0 1
(2)
paste1

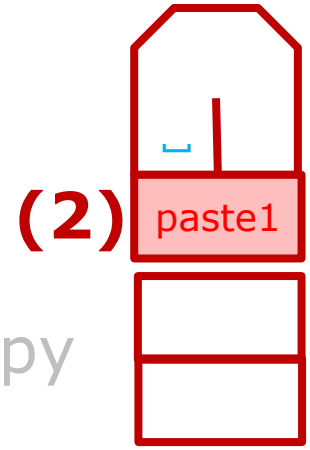
テープ(1)





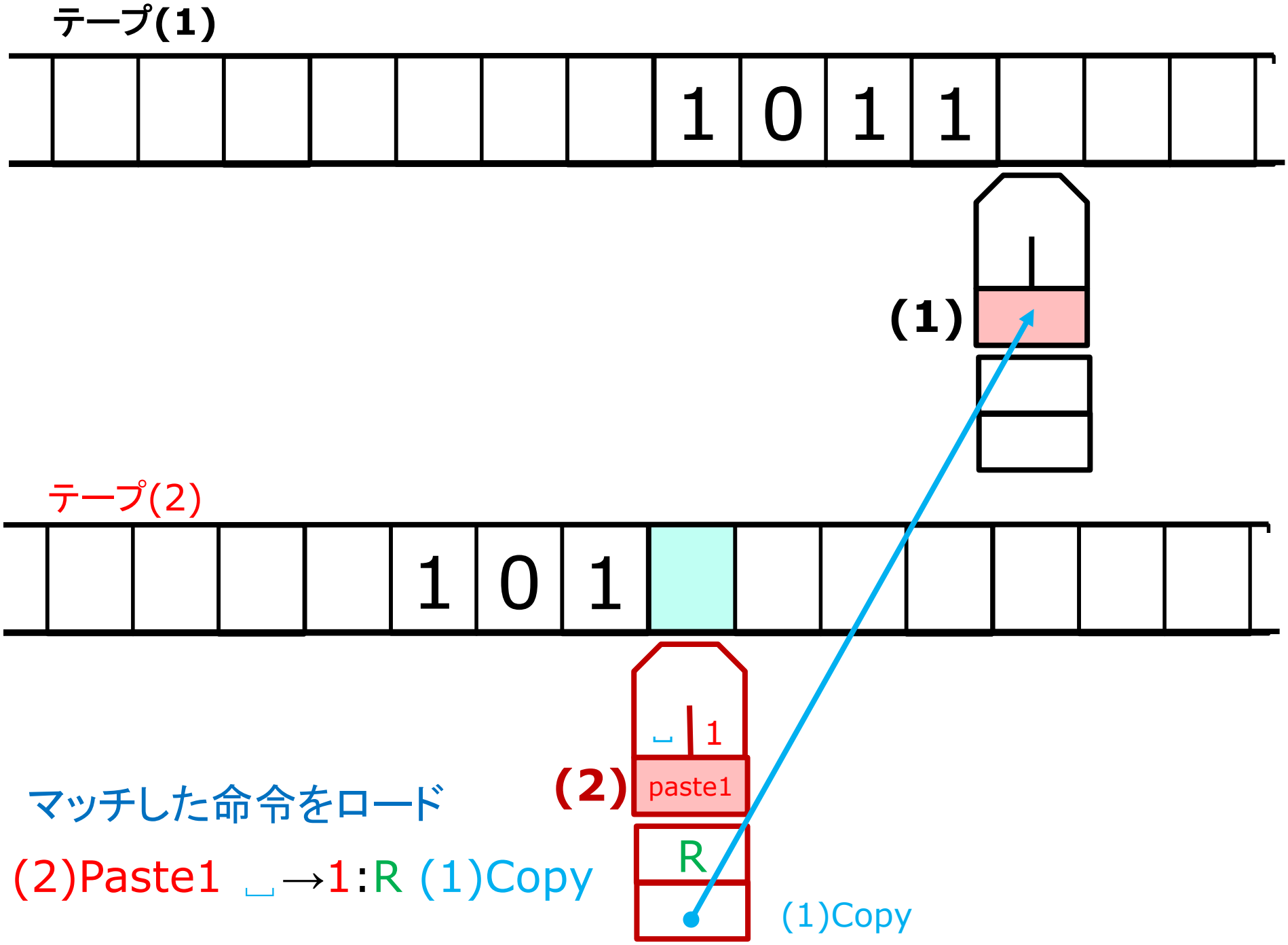
テープ(2)

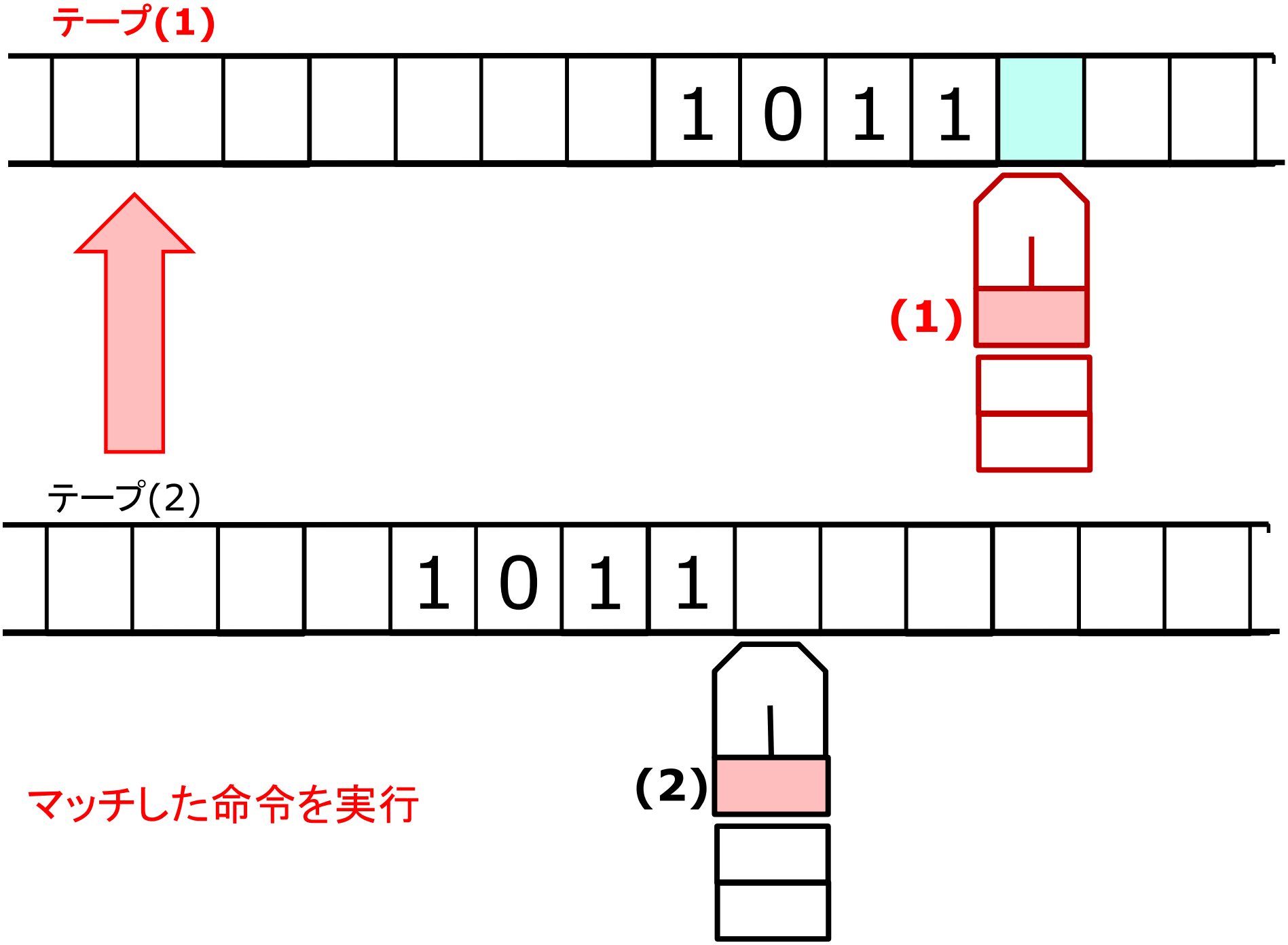




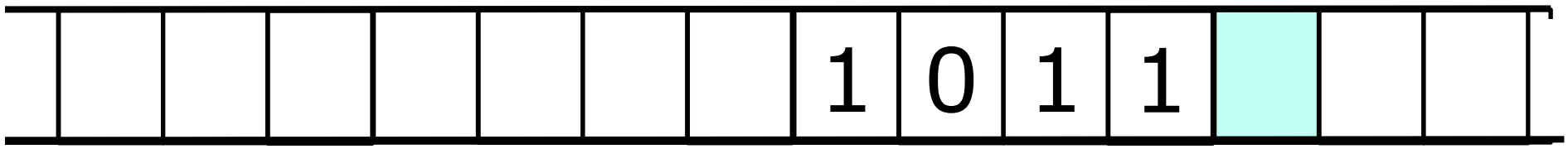
マッチする命令を探す

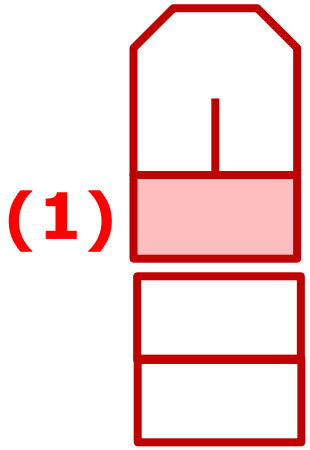
(2)Paste1 ␣→1:R (1)Copy


テープ(1)
1 0 1 1
(1)
テープ(2)
1 0 1
␣ 1
(2)
paste1
R
(1)Copy
マッチした命令をロード
(2)Paste1 ␣→1:R (1)Copy


テープ(1)
1 0 1 1
(1)
テープ(2)
1 0 1 1
(2)
マッチした命令を実行

テープ(1)

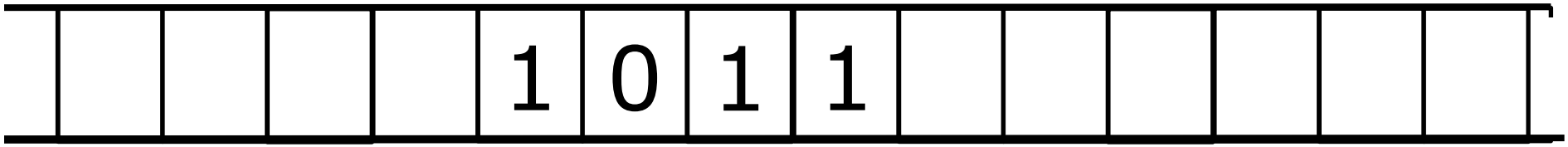


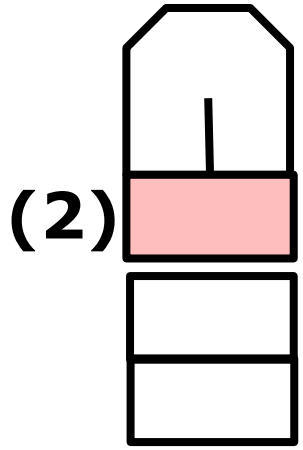


マッチする命令を探して実行

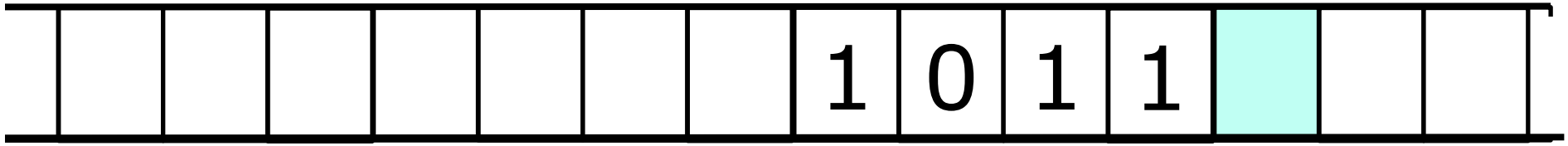
(1)Copy ␣→␣:N (1)Copy-End

テープ(2)





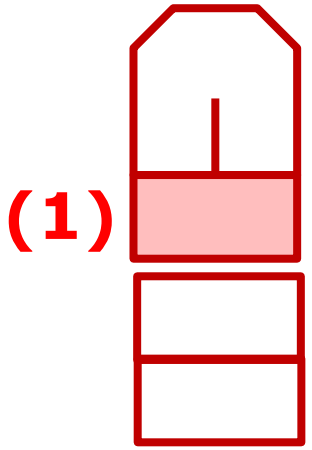
テープ(1)



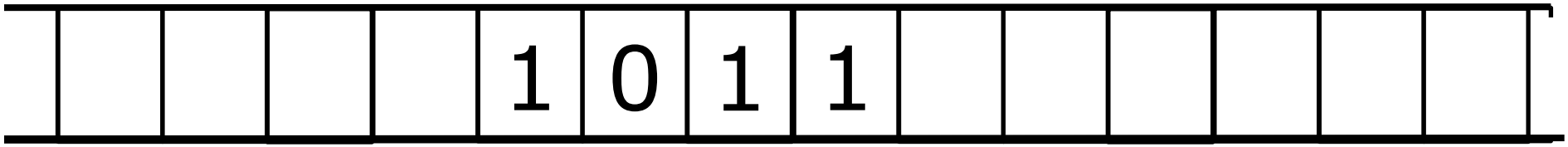
マッチする命令を探して実行

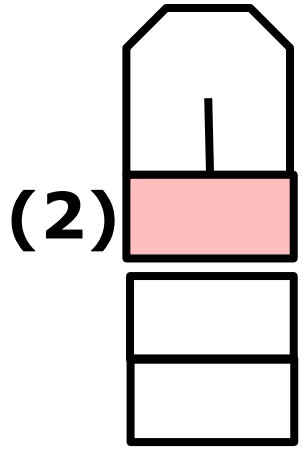
(1)Copy ␣→␣:N (1)Copy-End

コピー終了



テープ(2)





テープ1のビット列が
テープ2のビット列に
等しいかをチェックする

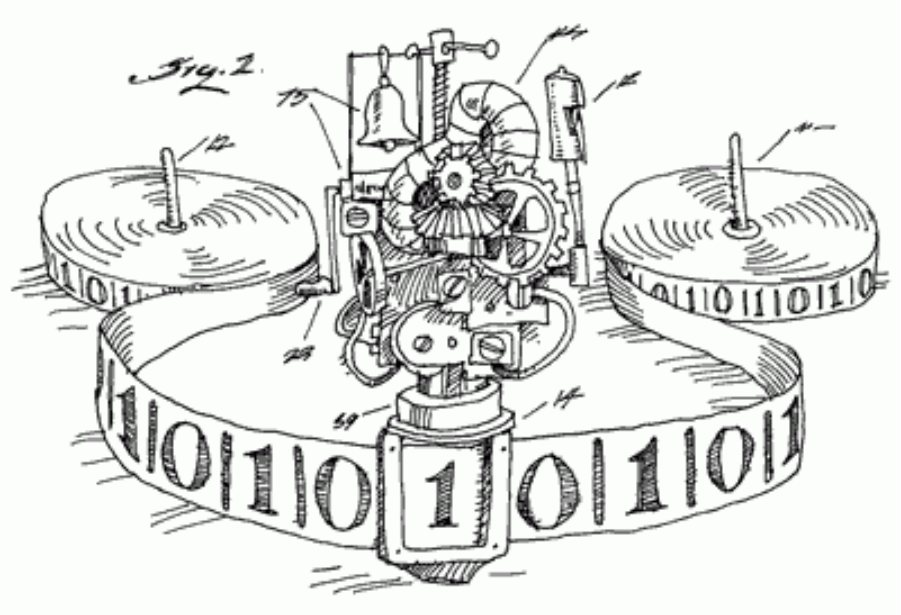
テープ1のビット列がテープ2のビット列に等しいかをチェックするプログラム

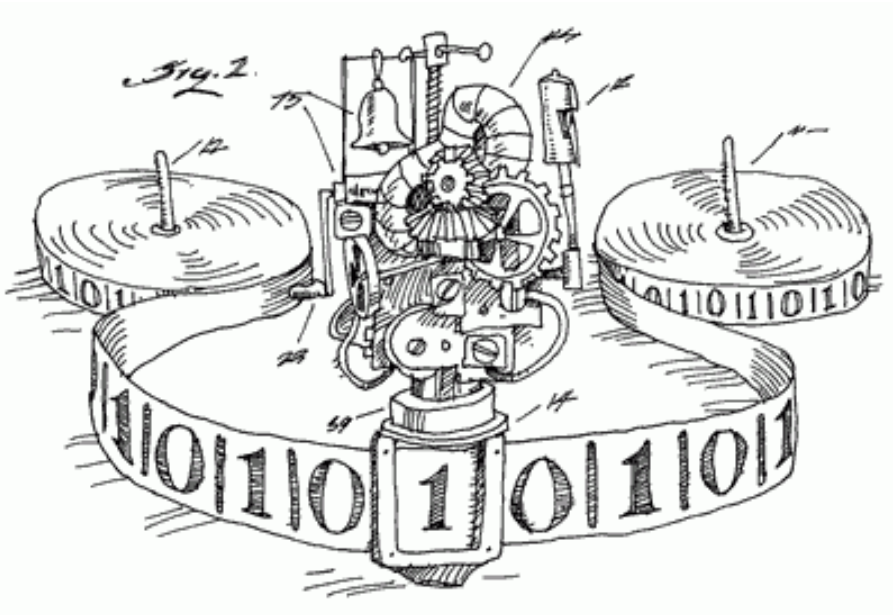
1. (1)Equal? 0→0:R (2)Compare0
2. (1)Equal? 1→1:R (2)Compare1
3. (1)Equal? ␣→␣:R (1)Equal?-End-True

4. (2)Compare0 0→0:R (1)Equal?
5. (2)Compare0 1→1:R (1)Equal?-End-False
6. (2)Compare0 ␣→␣:R (1)Equal?-End-False

7. (2)Compare1 0→0:R (1)Equal?-End-False
8. (2)Compare1 1→1:R (1)Equal?
9. (2)Compare1 ␣→␣:R (1)Equal?-End-False

2つのチューリングマシンを考える





「チューリングマシンを学ぼう！」

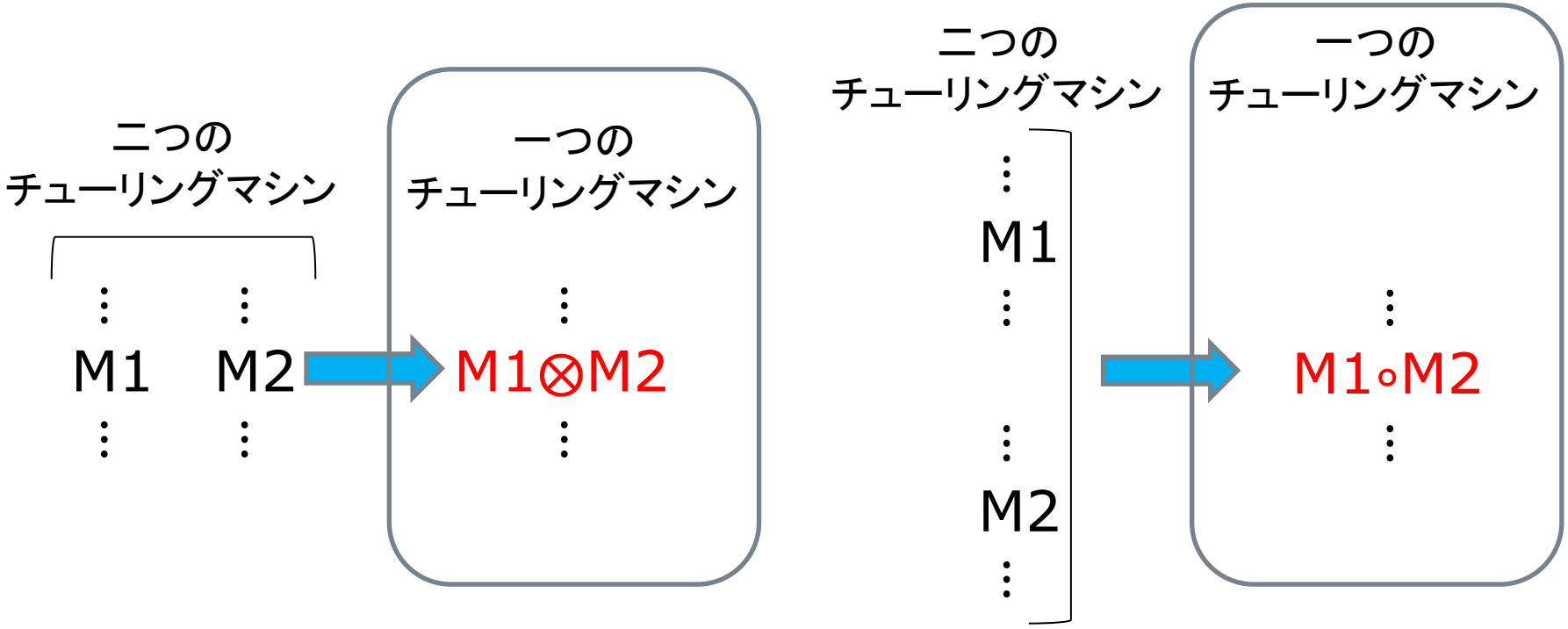
Part III　複数のテープと複数のマシン

二つのチューリングマシンの働きを一つのチューリングマシンにまとめる

- ここでは、複数のチューリングマシンの働きを、一つのチューリングマシンにまとめることを考えてみよう。

- 基本的には、二つのチューリングマシンM1, M2が、独立にパラレルに走る場合 と、連結してシリアルに走る場合を考えればいい。これらを次のように M1⊗M2, M1◦M2と表そう。

二つのチューリングマシンの働きを一つのチューリングマシンにまとめる



M1, M2が、独立に
パラレルに走る場合

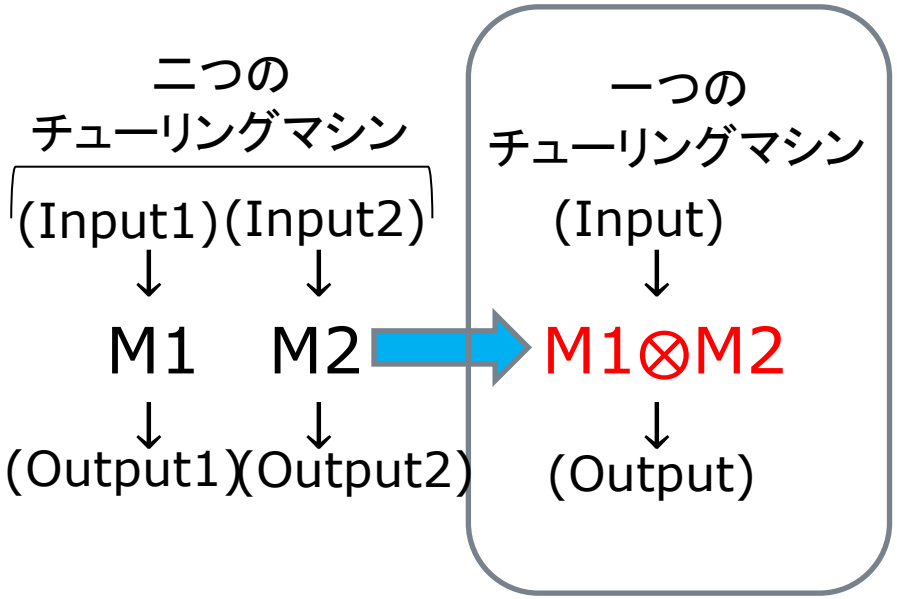
M1, M2が、連結して
シリアルに走る場合

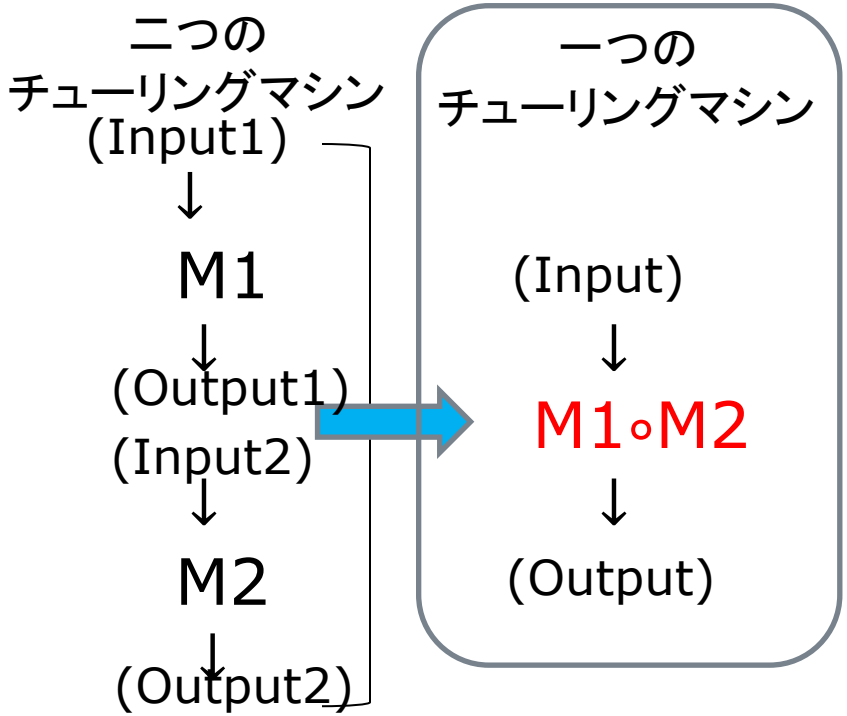
マシンへの入力と出力と
マシンの命令セットのノテーション

- チューリングマシンMは、入力用のテープ (Input) と出力用のテープ (Output) の二つのテープを持つことにしよう。
- マシンの初期状態とテープの初期状態(Input)が与えられて、マシンが計算をはじめて、停止した時のテープの状態を(Output)とする。

- また、チューリングマシンMの命令セットを [M]で表すことにしよう。
- 入力(Input)に対する、チューリングマシンMの出力が(Output)であることを、[M](Input)=(Output)と表す。

二つのチューリングマシンの働きを
一つのチューリングマシンにまとめる

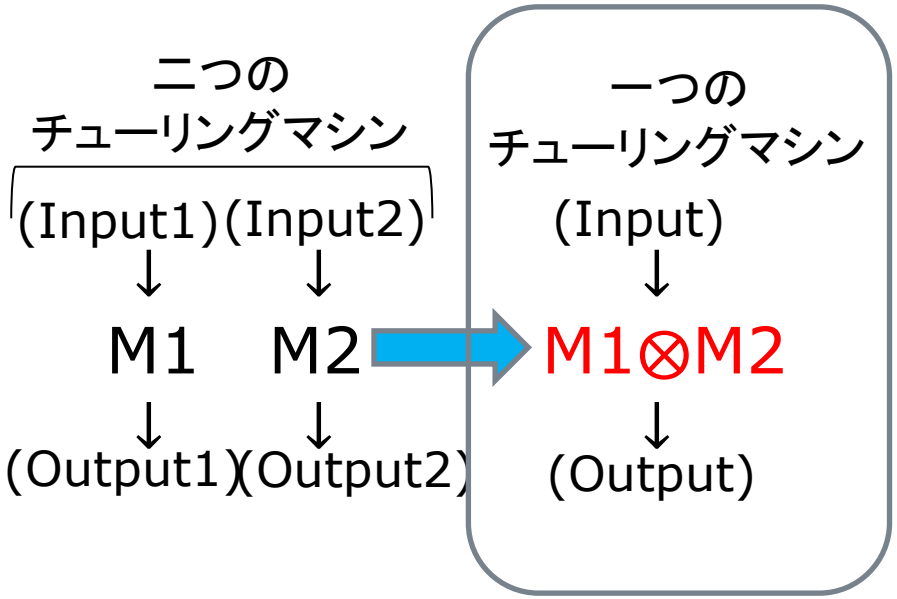




[M1](Input1)=(Output1),
[M2](Input2)=(Output2)
である。

この時、[M1⊗M2]、[M1◦M2]は、
どのように表されるか？

M1⊗M2の場合

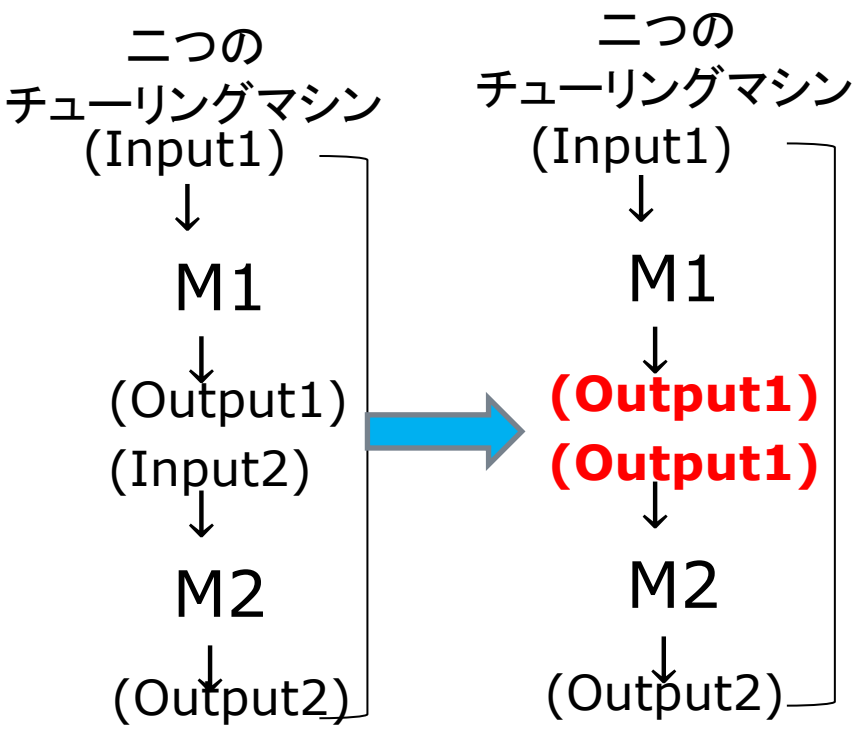


(Input)=(Input1)⊗(Input2)
(Output)=
(Output1)⊗(Output2)
なのだが(分離可能)、
[M1⊗M2]を[M1],[M2]で表すのは、あまり簡単ではない。
それは、M1, M2の
並列プログラミングを記述することだから。後回しにしよう。

[M1](Input1)=(Output1),
[M2](Input2)=(Output2)

むしろ、入力もプロセスも出力も分離可能なら、M1⊗M2は、二つの独立したプロセスと考える方が見通しはいい。⊗の無視。

M1◦M2の場合



[M1](Input1)=(Output1),
[M2](Output1)=(Output2)

M1の出力(Output1)を
そのままM2の入力に与える
ことで、M1とM2を連結する。

ただし、このままだと、二つの
処理を、一つのチューリング
マシンで処理したことにはなら
ない。

一つのチューリングマシン
M1◦M2の処理として記述する
には、どうすればいいか？

一つのマシンM1◦M2の処理としてまとめる

- マシンM1◦M2は、入力用のテープ(Input)、出力用のテープ(Output)の他に、三番目の中間作業用のテープ(Work)を持つ。

- マシンM1の出力を(Work)で受け取り、この(Work)をマシンM2の入力として渡す。
  - (Work)=[M1](Input)
  - (Output)=[M2](Work)

- マシンM1の終了後、マシンM2を起動する。

命令セットの書き換え
$[M1]^{*}$と$_{*}[M2]$

命令セットの書き換え$[M1]^{*}$と$_{*}[M2]$を次のように定義する。
[M1],[M2]は、あらかじめ状態の名前の衝突が起きないように書き換えておく。例えば、M1,M2の状態には、//M1/, //M2/といった接頭辞をつけておけばいい。

$[M1]^{*}$

- [M1]中の(Output)を全て(Work)に書き換える。
- [M1]中の終了状態(Halt)を全てM2の初期状態(例えば//M2/(Work)S_0)に書き換える。

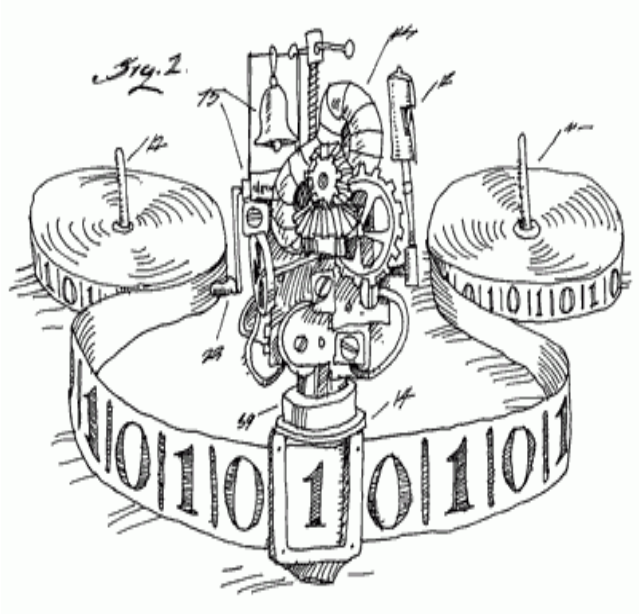
$_{*}[M2]$

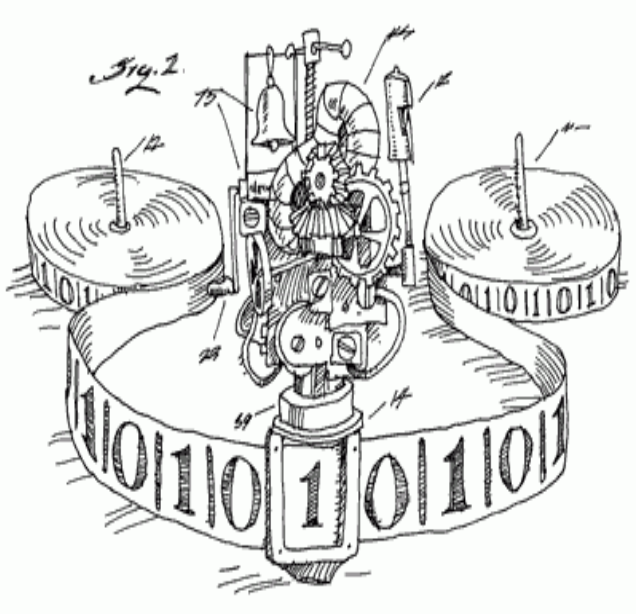
- [M2]中の(Input)を全て(Work)に書き換える。

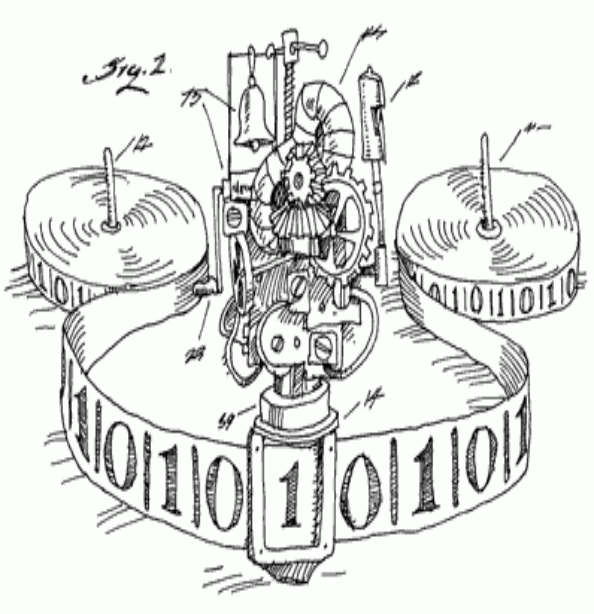
この時、$[M1 \circ M2] = [M1]^{*} \cup_{*} [M2]$

ControllerとExecuter

-- 1ステップ実行 --







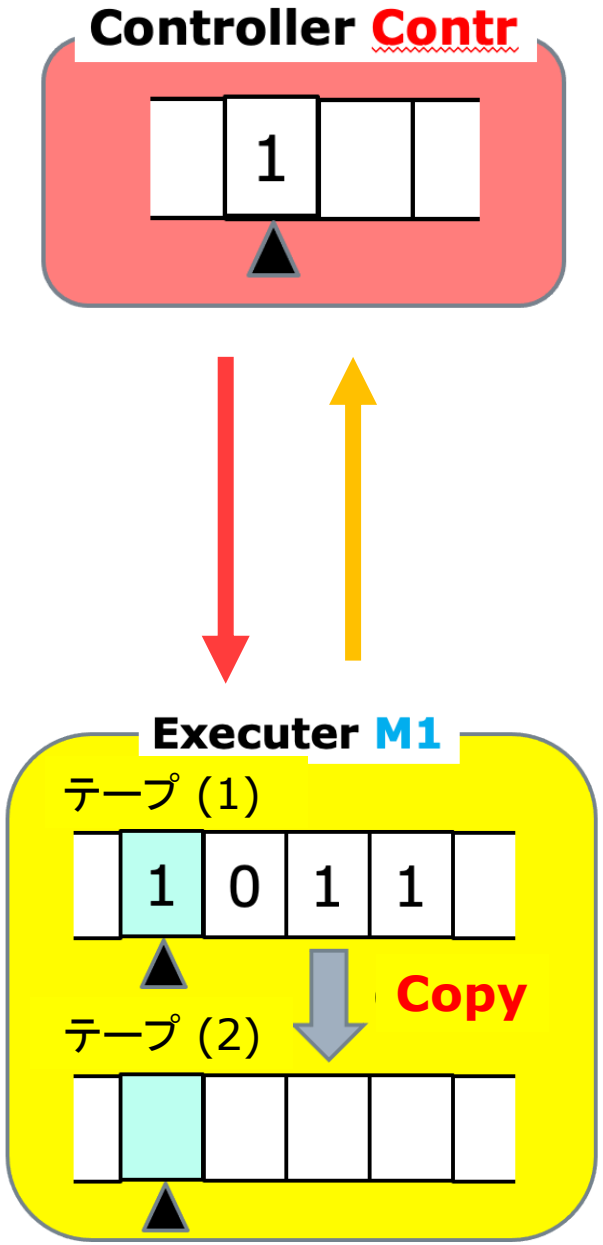
「チューリングマシンを学ぼう！」

Part III　複数のテープと複数のマシン

ControllerとExecuter
一台のコントローラと一台の実行マシン

ControllerとExecuter

実行マシンM1のCopy命令を
1ステップずつ実行させる
コントローラーContr を考えて
みよう。



実行マシンMの命令の形

- Executer マシンをMとする。
- Mの命令セットは、次のような形の命令からできている。

元のMの命令

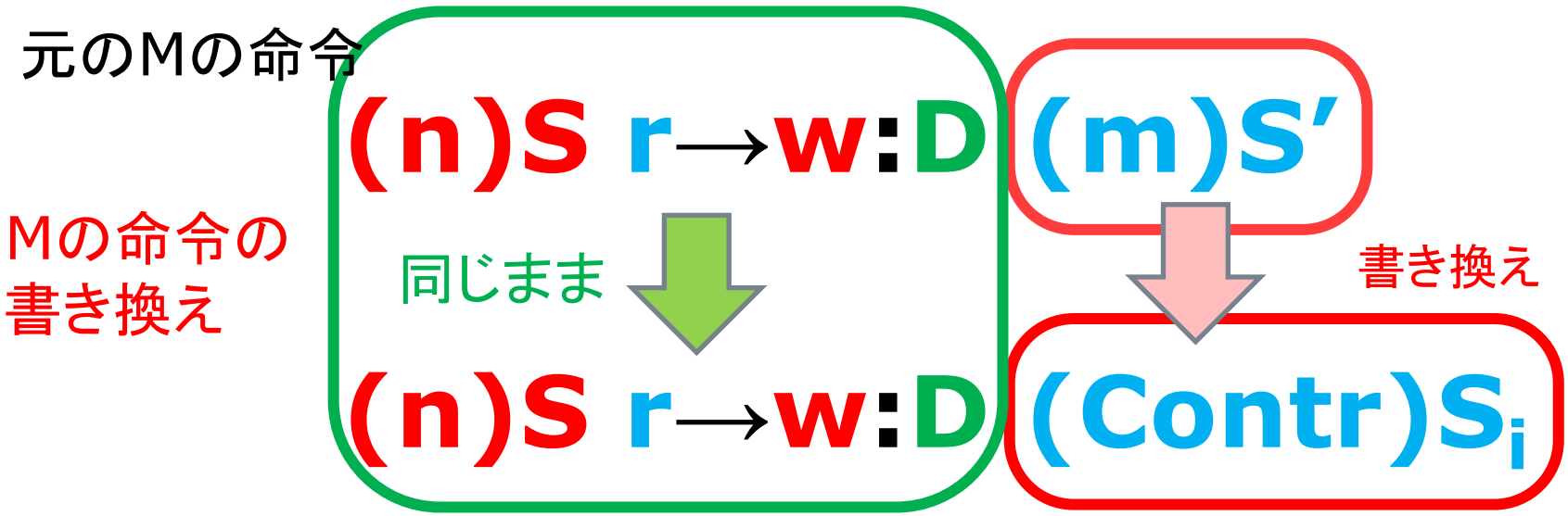
(n)S r→w:D (m)S′

n本目のテープの現在の状態

m本目のテープの次の状態

実行マシンMの命令の書き換え

- Executer マシンをMとする。
- Mの命令セットは、次のような形の命令からできている。
- この命令の最後の部分の「次の状態」を、Controller Contr 上の新しい状態S_i で全て置き換える。



書き換えられた実行マシンMの命令

- Executer マシンをMとする。
- Mの命令セットは、次のような形の命令からできている。
- この命令の最後の部分の「次の状態」を、Controller Contr 上の新しい状態S_i で全て置き換える。
- 命令実行後、制御は、コントローラーに移る。

元のMの命令

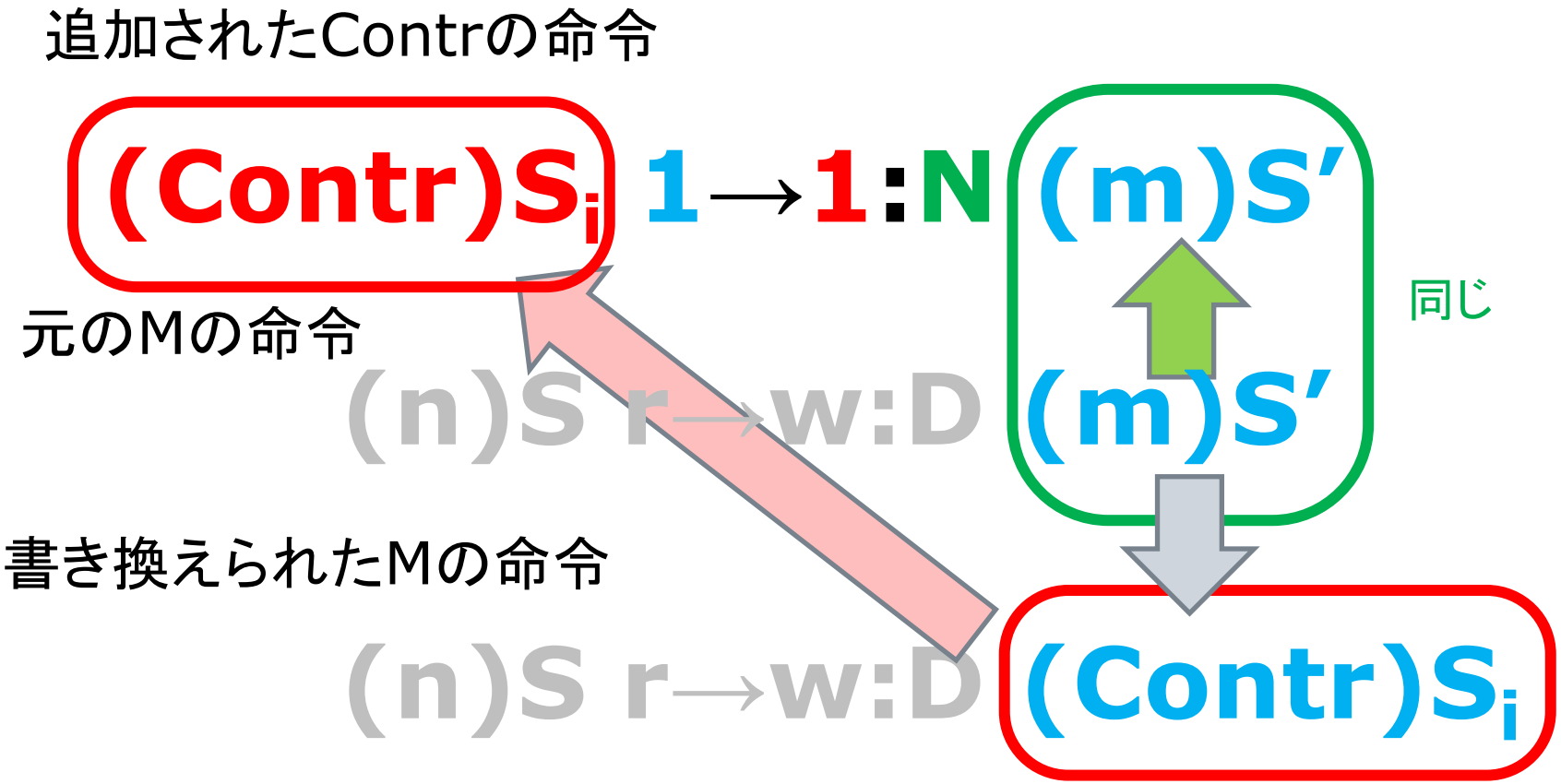
(n)S r→w:D (m)S'

書き換えられたMの命令

(n)S r→w:D **(Contr)S_i**

コントローラーに命令を追加する

- Controller Contrに次の新しい命令を追加する。



追加されたコントローラの命令

- Controller Contrに次の新しい命令を追加する。
- 命令を実行すると、制御は実行マシンに移る。

追加されたContrの命令

この意味は、後で説明する

(Contr)S_i 1→1:N (m)S′

元のMの命令

(n)S r→w:D (m)S′

書き換えられたMの命令

(n)S r→w:D (Contr)S_i

ControllerとExecuterの命令

- Controller ContrとExecuter Mは、次の命令を持つ。

追加されたContrの命令

(Contr)S_i 1→1:N (m)S′

(n)S r→w:D (m)S′

書き換えられたMの命令

(n)S r→w:D (Contr)S_i

ビット列コピーの例で、
Controller，Executerの命令セットを考える

ビット列コピーの例で、Controller，Executerの命令セットを考える

次のビット列コピーの例で、Controller，Executerの命令セットを考えてみよう。

/* 一本目のテープの文字を読み取り */
/* 二本目のテープに制御を移す　　*/
1. (1)Copy 0→0:R (2)Paste0
2. (1)Copy 1→1:R (2)Paste1
3. (1)Copy ␣→␣:N (1)Copy-End

/* 二本目のテープに一本目のテープの　*/
/* 文字を書き込み、一本目に制御を返す */
4. (2)Paste0 ␣→0:R (1)Copy
5. (2)Paste1 ␣→1:R (1)Copy

ビット列コピーの例での、命令セットの書き換え

1. (1)Copy 0→0:R (2)Paste0
   - (1)Copy 0→0:R (Contr)S1
   - (Contr)S1 1→1:N (2)Paste0
2. (1)Copy 1→1:R (2)Paste1
   - (1)Copy 1→1:R (Contr)S2
   - (Contr)S2 1→1:N (2)Paste1
3. (1)Copy ␣→␣:N (1)Copy-End
   - (1)Copy ␣→␣:N (Contr)S3
   - (Contr)S3 1→1:N (Contr)Halt
4. (2)Paste0 ␣→0:R (1)Copy
   - (2)Paste0 ␣→0:R (Contr)S0
   - (Contr)S0 1→1:N (1)Copy
5. (2)Paste1 ␣→1:R (1)Copy
   - (2)Paste1 ␣→0:R (Contr)S0
   - (Contr)S0 1→1:N (1)Copy

Controller ContrとExecuter Mの命令セット

コントローラ Contrの命令セット

- (Contr)S0 1→1:N (1)Copy
- (Contr)S1 1→1:N (2)Paste0
- (Contr)S2 1→1:N (2)Paste1
- (Contr)S3 1→1:N (Contr)Halt

実行マシンMの命令セット

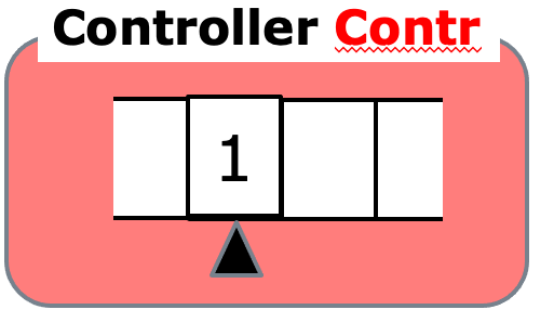
- (1)Copy 0→0:R (Contr)S1
- (1)Copy 1→1:R (Contr)S2
- (1)Copy ␣→␣:N (Contr)S3
- (2)Paste0 ␣→0:R (Contr)S0
- (2)Paste1 ␣→1:R (Contr)S0

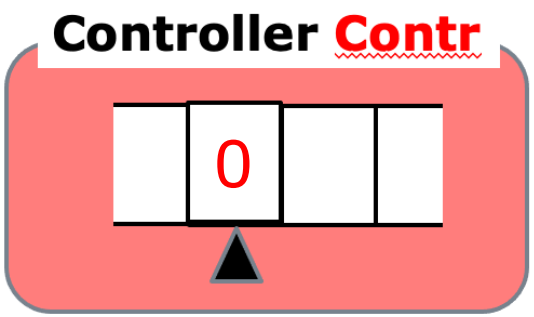
命令の実行を一時停止する

命令の実行を一時停止する状態

- 先の例では、制御はコントローラーと実行マシンの間を往復するが、1 ステップ実行と言っても、命令がストップすることはない。
- 命令の実行を一時停止するには、命令の実行を一時停止する状態を付け加えなければならない。状態Sが、一時停止状態にあることを、^Sで表そう。
- 実行マシンは、停止状態を返すことにする。
- この時、例えば、次のような命令を、コントローラーに付け加える。すなわち、ヘッドが1の上にある時には、停止状態を解いて命令を実行し、ヘッドが0の時には停止状態を続ける。

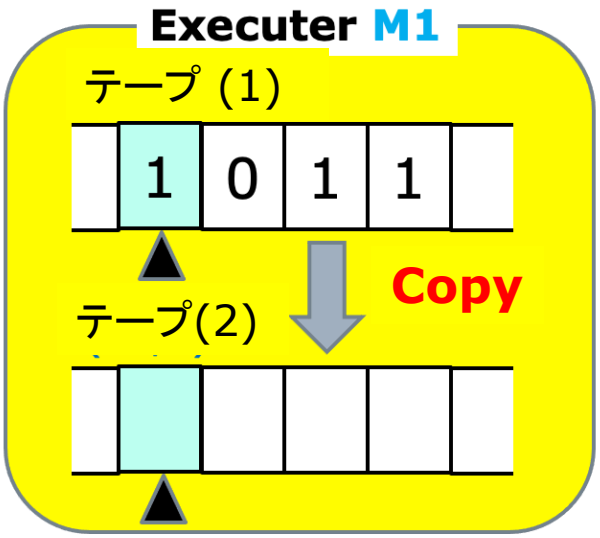
```
^S 1→1:N S     /* 停止解除命令 */
^S 0→0:N ^S    /* 停止継続命令*/
```

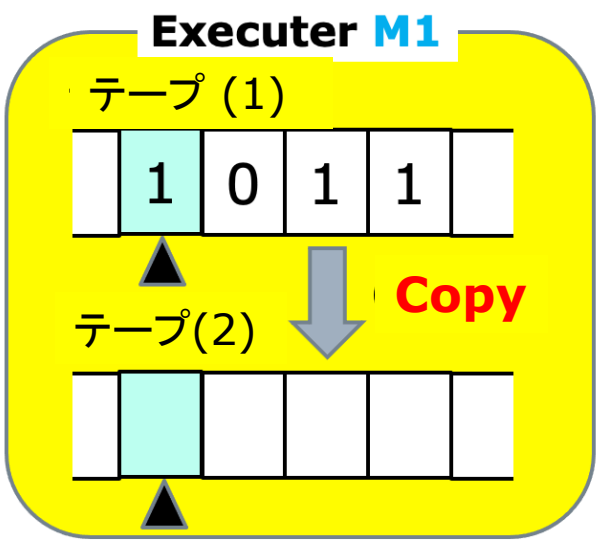




ヘッドが 1の上にある時
チューリングマシンM1を
1 ステップずつ実行する

ヘッドが 0の上にある時
チューリングマシンM1の
ステップ実行を停止する





実行マシンMの命令セットの変更

元の実行マシンMの命令セット

- (1)Copy 0→0:R (Contr)S1
- (1)Copy 1→1:R (Contr)S2
- (1)Copy ␣→␣:N (Contr)S3
- (2)Paste0 ␣→0:R (Contr)S0
- (2)Paste1 ␣→1:R (Contr)S0

変更された実行マシンMの命令セット

- (1)Copy 0→0:R ^(Contr)S1
- (1)Copy 1→1:R ^(Contr)S2
- (1)Copy ␣→␣:N ^(Contr)S3
- (2)Paste0 ␣→0:R ^(Contr)S0
- (2)Paste1 ␣→1:R ^(Contr)S0

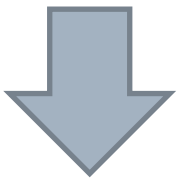
実行マシンは、停止状態をコントローラに返す

Controllerの命令セットの変更と追加

元のコントローラ Contrの命令セット

- (Contr)S0 1→1:N (1)Copy
- (Contr)S1 1→1:N (2)Paste0
- (Contr)S2 1→1:N (2)Paste1
- (Contr)S3 1→1:N (Contr)Halt

ヘッドが 1の上にある時
チューリングマシンM1を
1 ステップずつ実行する



実行マシンは、
コントローラに
停止状態を返す

ヘッドが 0の上にある時
チューリングマシンM1の
ステップ実行を停止する

Contr命令セットの変更

- ^(Contr)S0 1→1:N (1)Copy
- ^(Contr)S1 1→1:N (2)Paste0
- ^(Contr)S2 1→1:N (2)Paste1
- ^(Contr)S3 1→1:N (Contr)Halt

Contrの命令セットの追加

- ^(Contr)S0 0→0:N ^(Contr)S0
- ^(Contr)S1 0→0:N ^(Contr)S1
- ^(Contr)S2 0→0:N ^(Contr)S2
- ^(Contr)S3 0→0:N ^(Contr)S3