more models of computation

concerning computability
and P(whatever that is)

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

1 Time and Space bounded Turing machines

Let $t,s:\mathbb{N}_0 \rightarrow \mathbb{N}_0$ be functions.

def: t-time or space bounded Turing machins

- a TM M is $t(n)$-time bounded if for all n and every input w of length n machine M started with w makes at most $t(n)$ steps.
- a TM M is $s(n)$-space bounded if for all n and every input w of length n machine M started with w the number of tape cells visited or occupied by the input (they may not all be visited) is at most $s(n)$.

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

1 Time and Space bounded Turing machines

Let $t, s : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ be functions.

def: t-time or space bounded Turing machins

- a TM M is $t(n)$-time bounded if for all n and every input w of length n machine M started with w makes at most $t(n)$ steps.
- a TM M is $s(n)$-space bounded if for all n and every input w of length n machine M started with w the number of tape cells visited or occupied by the input (they may not all be visited) is at most $s(n)$.

tape bounds < n: make input tape read only

not studied here

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

1 Time and Space bounded Turing machines

Let $t, s : \mathbb{N}_0 \to \mathbb{N}_0$ be functions.

def: t-time or space bounded Turing machins

- a TM M is $t(n)$-time bounded if for all n and every input w of length n machine M started with w makes at most $t(n)$ steps.
- a TM M is $s(n)$-space bounded if for all n and every input w of length n machine M started with w the number of tape cells visited or occupied by the input (they may not all be visited) is at most $s(n)$.

tape bounds < n: make input tape read only

not studied here

2 Simulating k-tape Turing machines

The mother of all simulation theorems:

Lemma 1.
- *Every function computable by a k-tape TM M is computable by a 1-tape TM M'*
- *if M is $t(n)$-time bounded, then M' is $O(t^2(n) + n \cdot t(n))$-time bounded*
- *if M is $s(n)$-space bounded, then M' is $(s(n)+2)$-space bounded.*

time bound: if M does not read the whole input we might have $t(n) < n$.

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

1 Time and Space bounded Turing machines

Let $t, s : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ be functions.

def: t-time or space bounded Turing machins

- a TM M is $t(n)$-time bounded if for all n and every input w of length n machine M started with w makes at most $t(n)$ steps.
- a TM M is $s(n)$-space bounded if for all n and every input w of length n machine M started with w the number of tape cells visited or occupied by the input (they may not all be visited) is at most $s(n)$.

tape bounds < n: make input tape read only

not studied here

2 Simulating k-tape Turing machines

The mother of all simulation theorems: of CS

Lemma 1.
- *Every function computable by a k-tape TM M is computable by a 1-tape TM M'*
- *if M is $t(n)$-time bounded, then M' is $O(t^2(n) + n \cdot t(n))$-time bounded*
- *if M is $s(n)$-space bounded, then M' is $(s(n)+2)$-space bounded.*

time bound: if M does not read the whole input we might have $t(n) < n$.

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

1 Time and Space bounded Turing machines

Let $t,s : \mathbb{N}_0 \to \mathbb{N}_0$ be functions.

def: t-time or space bounded Turing machins

- a TM M is $t(n)$-time bounded if for all n and every input w of length n machine M started with w makes at most $t(n)$ steps.
- a TM M is $s(n)$-space bounded if for all n and every input w of length n machine M started with w the number of tape cells visited or occupied by the input (they may not all be visited) is at most $s(n)$.

tape bounds < n: make input tape read only

not studied here

2 Simulating k-tape Turing machines

The mother of all simulation theorems: of CS

Lemma 1.
- *Every function computable by a k-tape TM M is computable by a 1-tape TM M'*
- *if M is $t(n)$-time bounded, then M' is $O(t^2(n) + n \cdot t(n))$-time bounded*
- *if M is $s(n)$-space bounded, then M' is $(s(n)+2)$-space bounded.*

time bound: if M does not read the whole input we might have $t(n) < n$.

what if we would simulate:

- k many 1-tape machines
- by one 1-tape machine

Simulations by 1-tape Turing machines

question: how powerful are 1 tape Turing machines? Answer: as powerful as anything you have seen.

1 Time and Space bounded Turing machines

Let $t, s : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ be functions.

def: t-time or space bounded Turing machins

- a TM M is $t(n)$-time bounded if for all n and every input w of length n machine M started with w makes at most $t(n)$ steps.
- a TM M is $s(n)$-space bounded if for all n and every input w of length n machine M started with w the number of tape cells visited or occupied by the input (they may not all be visited) is at most $s(n)$.

tape bounds < n: make input tape read only

not studied here

2 Simulating k-tape Turing machines

The mother of all simulation theorems: of CS

Lemma 1. • *Every function computable by a k-tape TM M is computable by a 1-tape TM M'*

- *if M is $t(n)$-time bounded, then M' is $O(t^2(n) + n \cdot t(n))$-time bounded*
- *if M is $s(n)$-space bounded, then M' is $(s(n)+2)$-space bounded.*

time bound: if M does not read the whole input we might have $t(n) < n$.

what if we would simulate:

- k many 1-tape machines
- by one 1-tape machine

then:

- same proof would work (as you will see)
- M' would be an OS kernel for 1 tape machines
- that's what we did with CVM!

2 Simulating k-tape Turing machines

The mother of all simulation theorems:

Lemma 1.
- *Every function computable by a k-tape TM M is computable by a* 1*-tape TM* M'
- *if* M *is* $t(n)$*-time bounded, then* M' *is* $O(t^2(n) + n \cdot t(n))$*-time bounded*
- *if* M *is* $s(n)$*-space bounded, then* M' *is* $(s(n)+2)$*-space bounded.*

time bound: if M *does not read the whole input we might have* $t(n) < n$*.*

We will simulate a given k-tape TM

$$M = (Z, A, \delta, z_0, E)$$

by a 1 tape TM

$$M' = (Z', A', \delta', z'_0, E')$$

We will mostly consider A' and δ'.

2.1 the crucial trick: dividing a tape into tracks

- We divide a finite portion of the tape of M' into $k+1$ tracks $0,\dots,k$ (everything else is blank).
- on track 0 we store the state of M (surrounded by blanks).
- on track $i > 0$ we store the non blank portion of tape i of M

$$A = \{0,1,B,\#\} \cup A'' \qquad A' = A \cup (Z \cup \{B\}) \times A^k$$

- For $a \in A'$ component a_i belongs to track i.

So we Code configuration of figure 1 by configuration of M' of figure 2

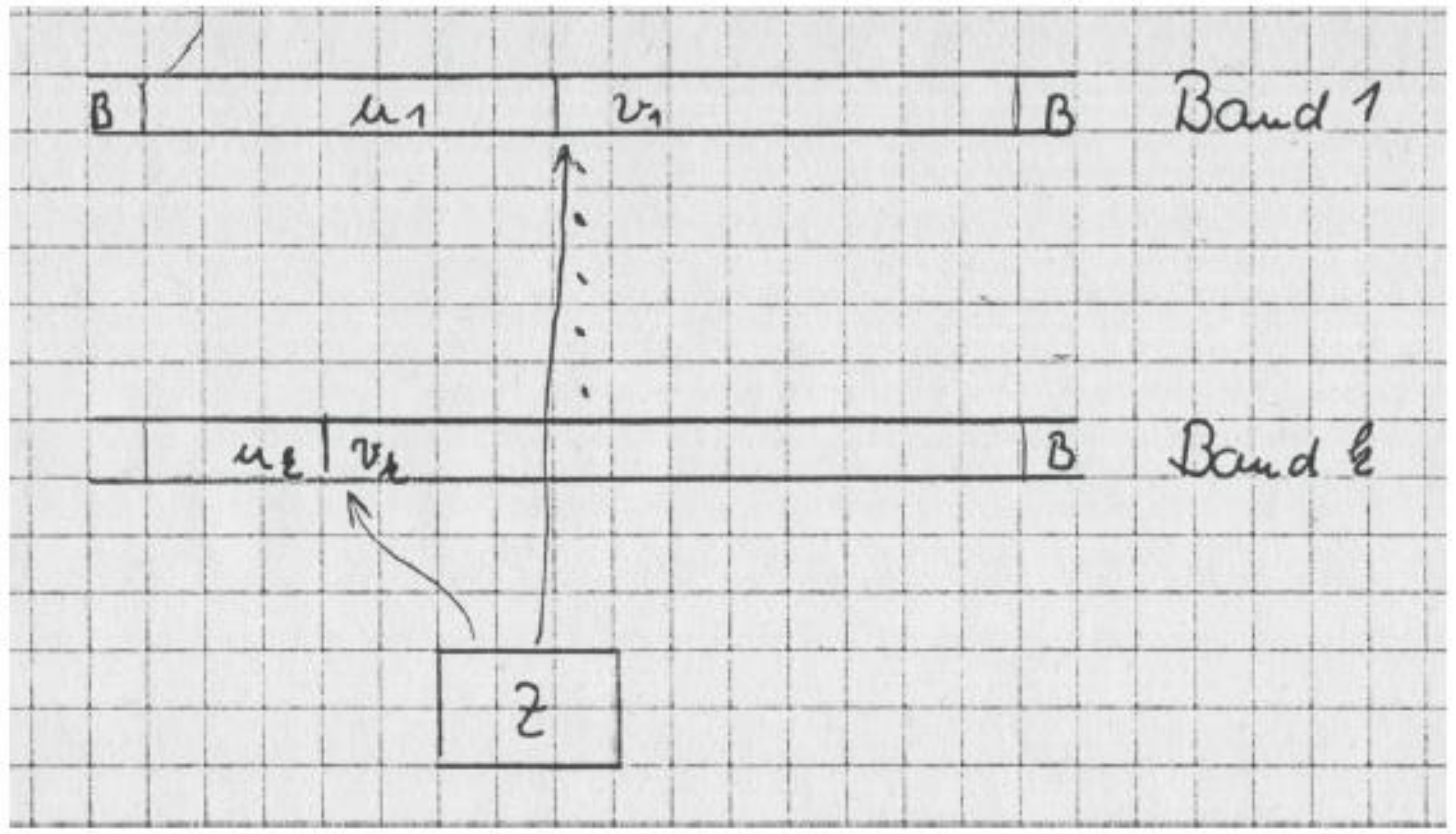

Figure 1: configuration of k-tape TM M. ('Band' = tape).

2.1 the crucial trick: dividing a tape into tracks

- We divide a finite portion of the tape of M' into $k+1$ tracks $0,\dots,k$ (everything else is blank).
- on track 0 we store the state of M (surrounded by blanks).
- on track $i>0$ we store the non blank portion of tape i of M

$$A = \{0,1,B,\#\} \cup A'' \qquad A' = A \cup (Z \cup \{B\}) \times A^k$$

- For $a \in A'$ component a_i belongs to track i.

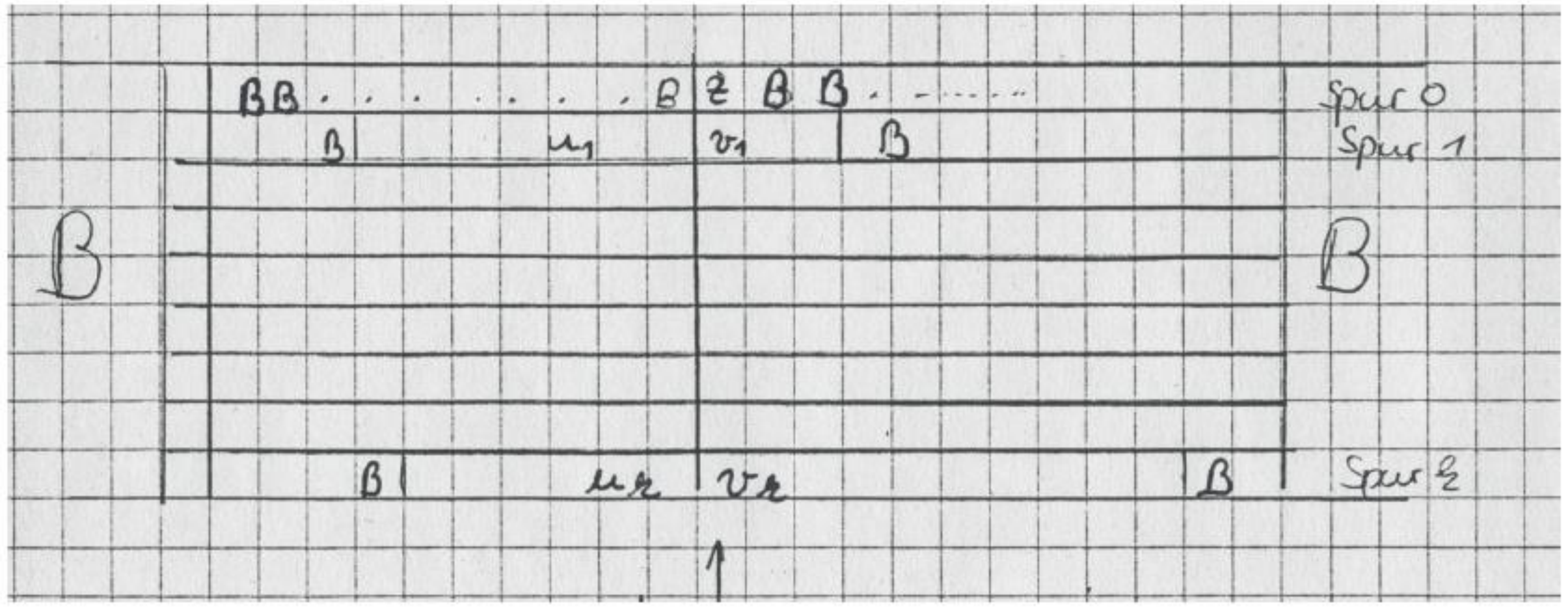

Figure 2: Configuration of 1-tape TM M' coding the configuration of figure 1 ('Spur'= track). Vertical slices are elements of A'.

So we Code configuration of figure 1 by configuration of M' of figure 2

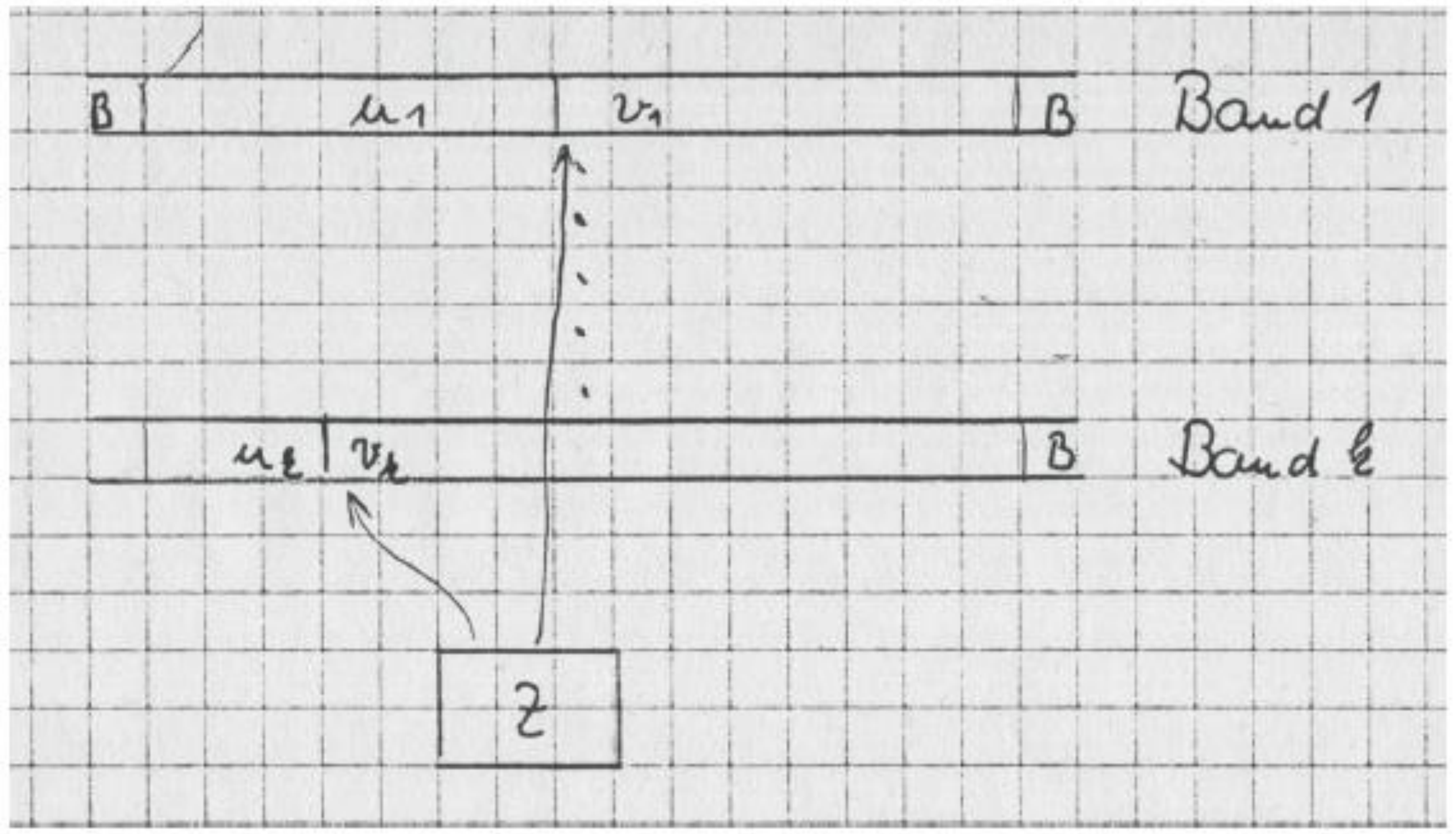

Figure 1: configuration of k-tape TM M. ('Band' = tape).

So we Code configuration of figure 1 by configuration of M' of figure 2

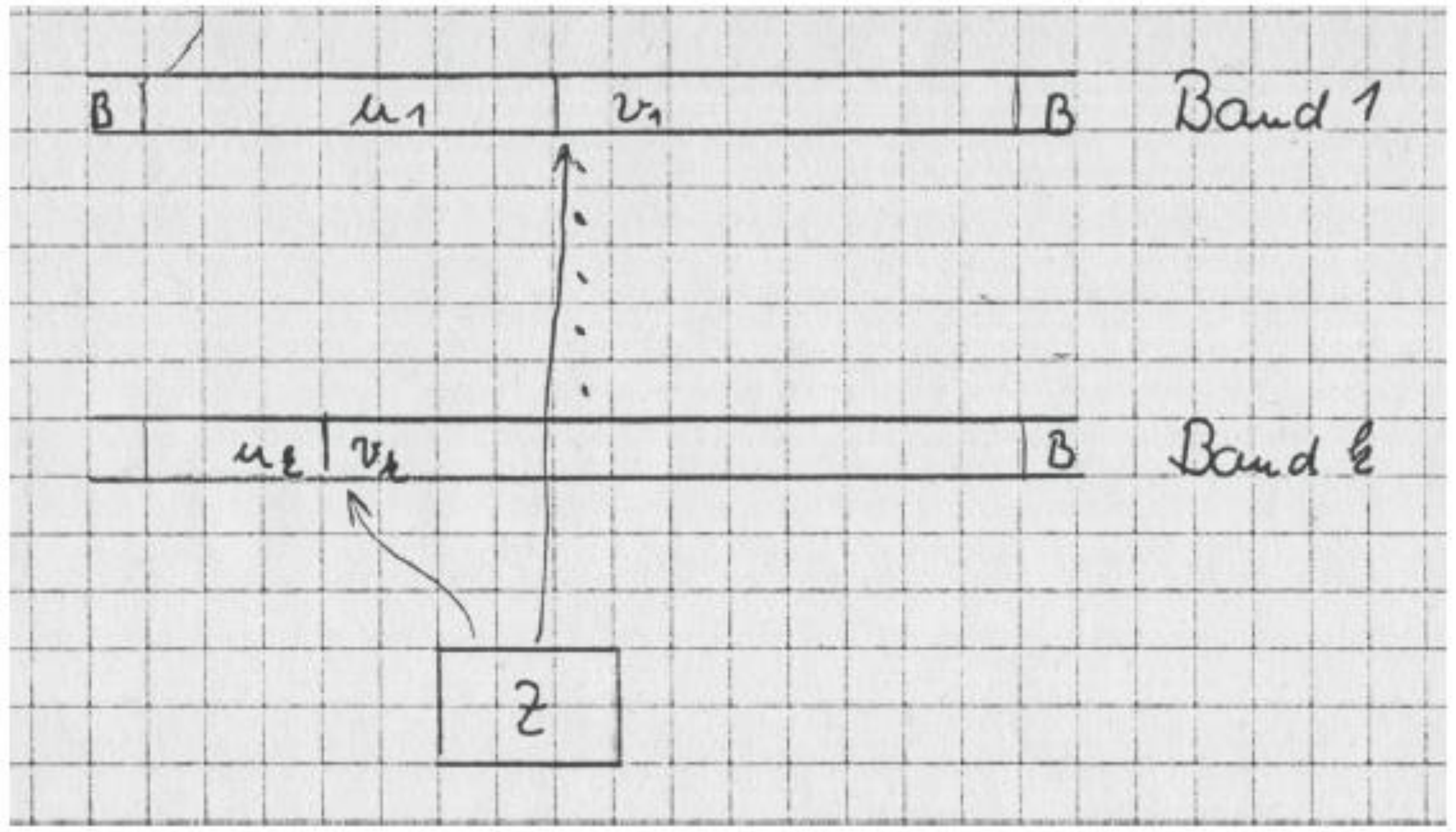

Figure 1: configuration of k-tape TM M. ('Band' = tape).

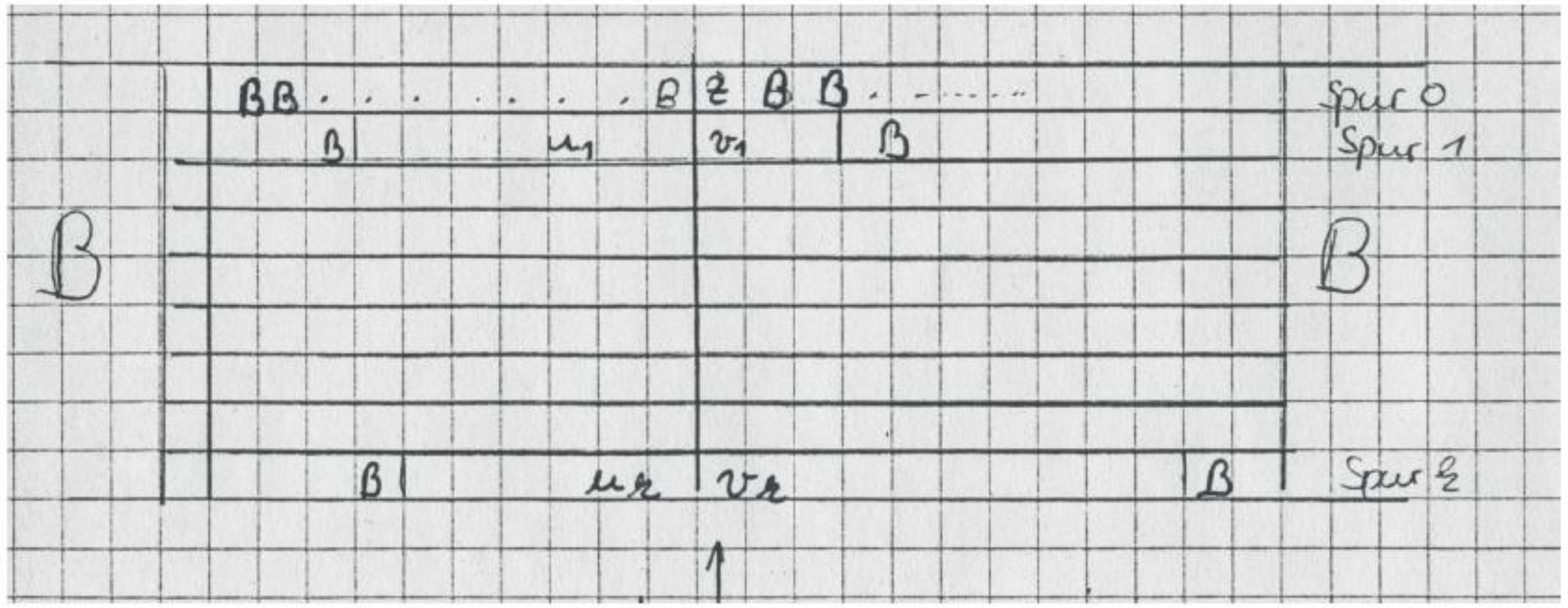

Figure 2: Configuration of 1-tape TM M' coding the configuration of figure 1 ('Spur'= track). Vertical slices are elements of A'.

2.2 Preprocessor

Preprocessing by M' has to establish the situation of figure 3 for the start configuartion. Thus the configuration of M shown in figure 3 has to be transformed into the configuration shown in figure 4. If $|w| = n$ this takes time $2n+4$ and space $n+2$

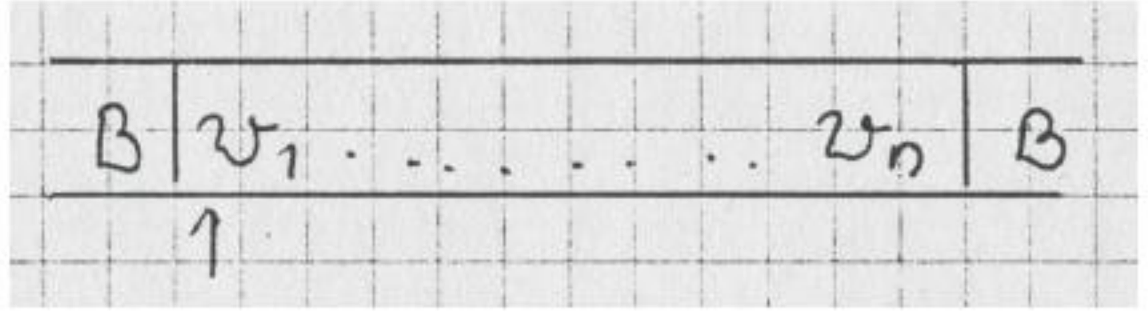

Figure 3: start configuration of k-tape TM M. ('Band' = tape).

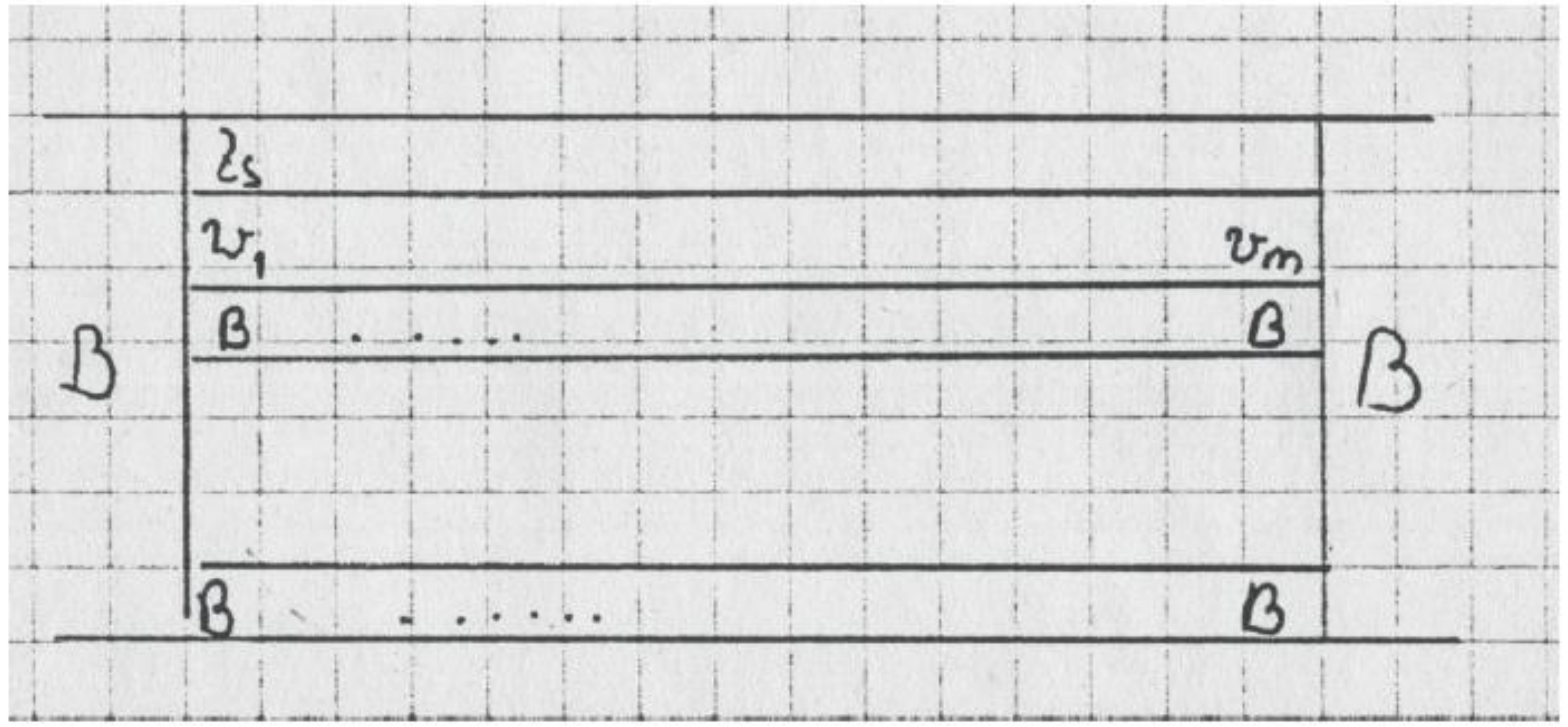

Figure 4: After the preprocessing the oringinal input occupies track 1. The start state is on track 0 at the start of the insciption

So we Code configuration of figure 1 by configuration of M' of figure 2

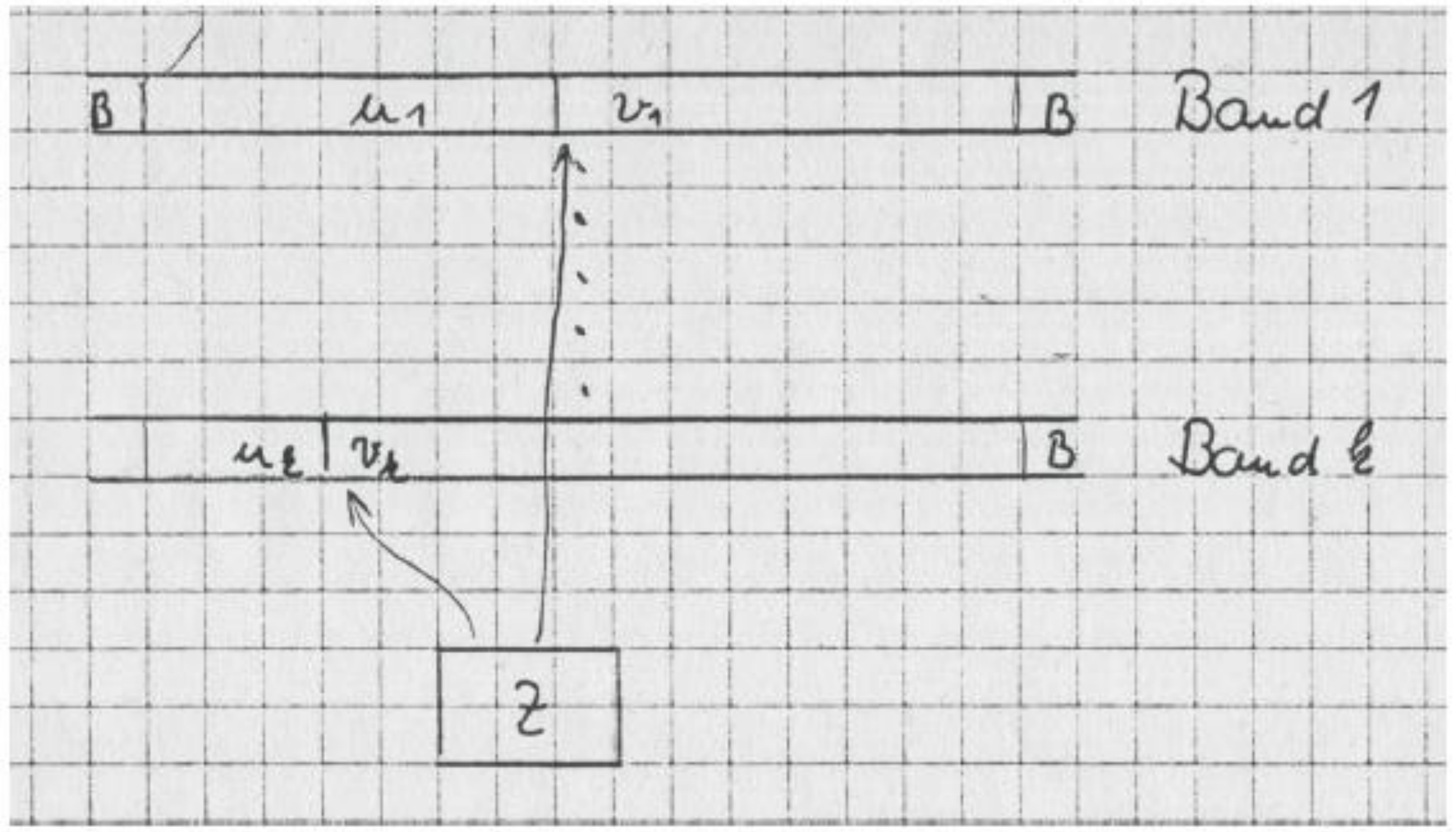

Figure 1: configuration of k-tape TM M. ('Band' = tape).

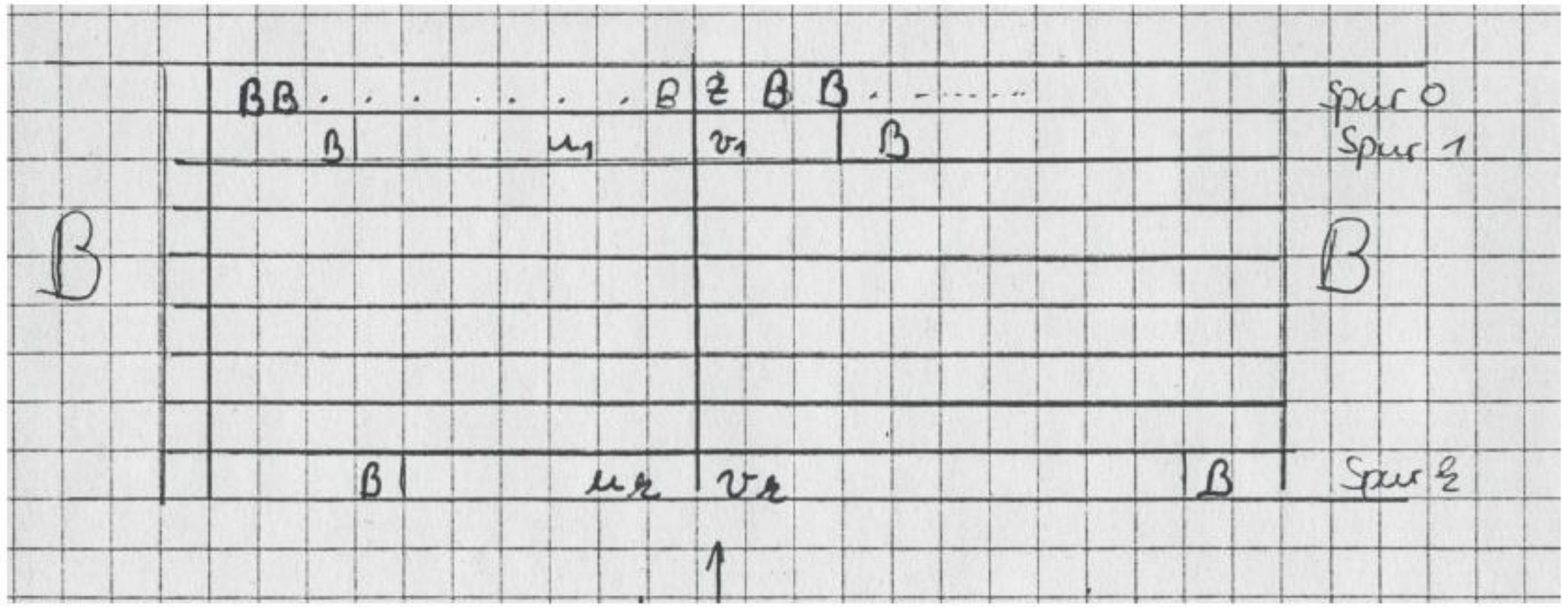

Figure 2: Configuration of 1-tape TM M' coding the configuration of figure 1 ('Spur'= track). Vertical slices are elements of A'.

2.2 Preprocessor

Preprocessing by M' has to establish the situation of figure 3 for the start configuartion. Thus the configuration of M shown in figure 3 has to be transformed into the configuration shown in figure 4. If $|w| = n$ this takes time $2n+4$ and space $n+2$

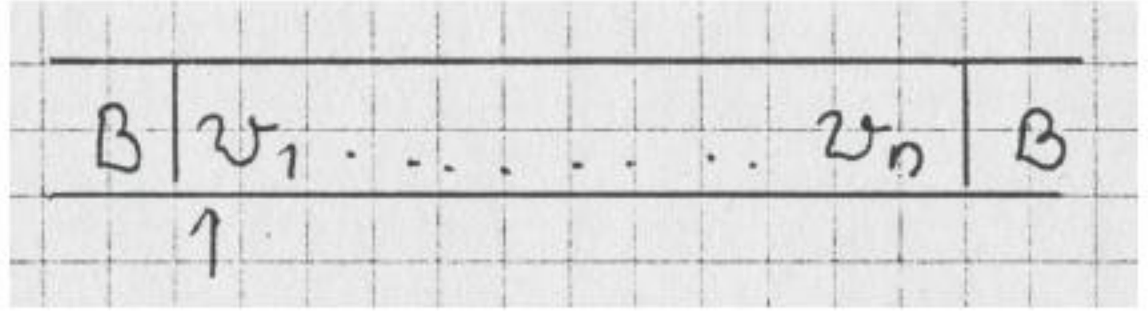

Figure 3: start configuration of k-tape TM M. ('Band' = tape).

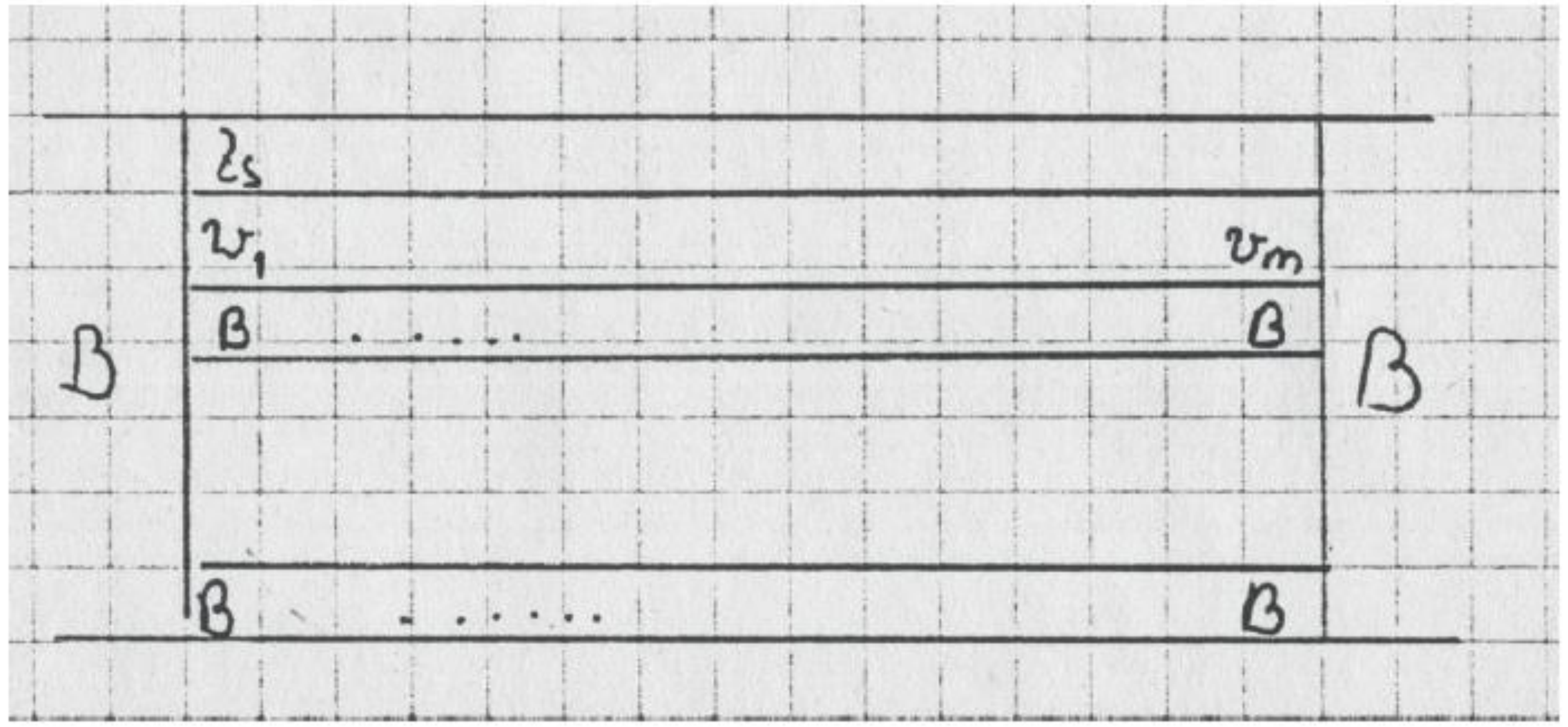

Figure 4: After the preprocessing the oringinal input occupies track 1. The start state is on track 0 at the start of the insciption

So we Code configuration of figure 1 by configuration of M' of figure 2

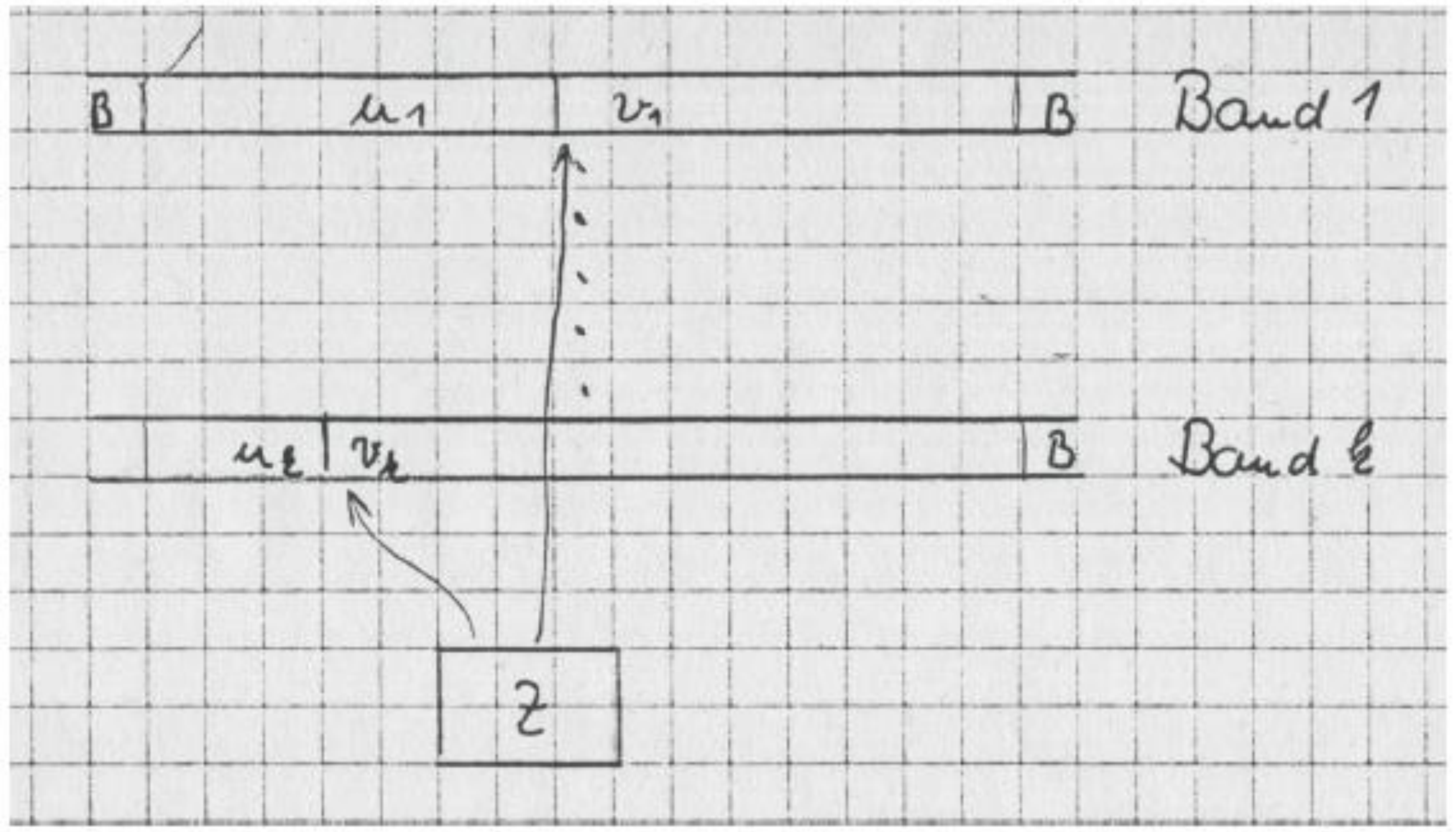

Figure 1: configuration of k-tape TM M. ('Band' = tape).

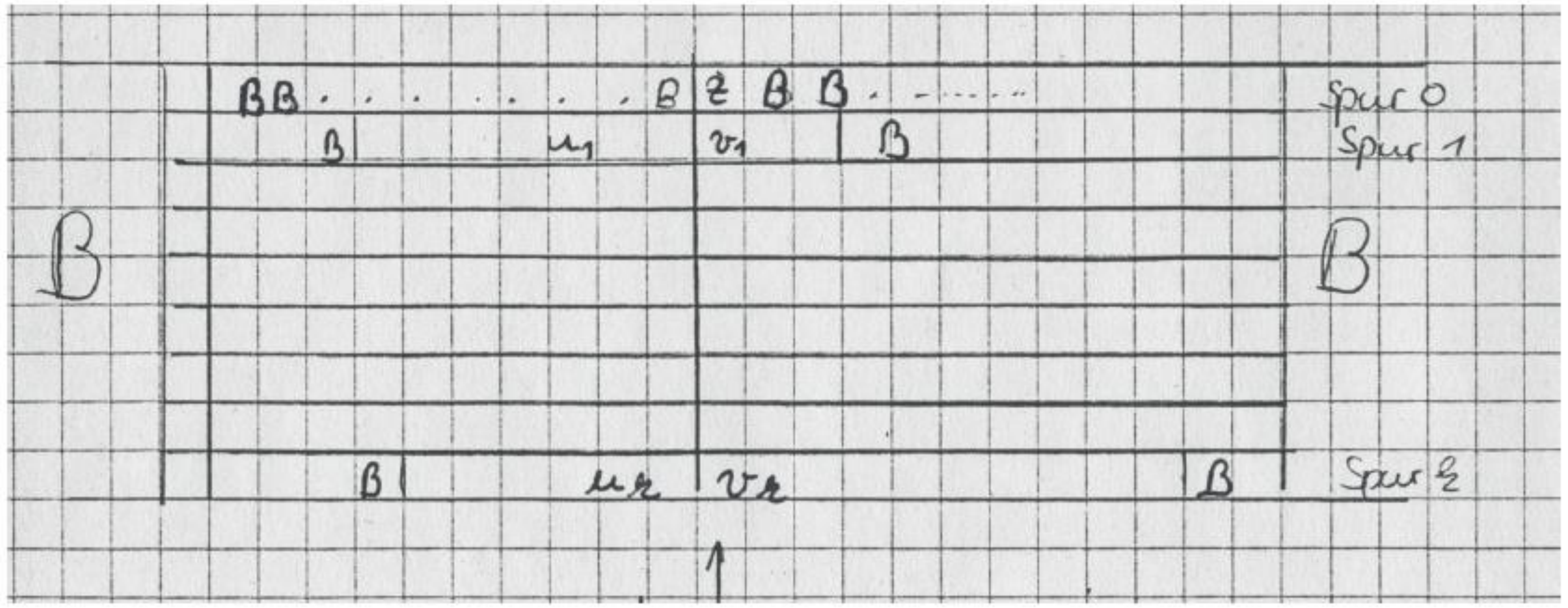

Figure 2: Configuration of 1-tape TM M' coding the configuration of figure 1 ('Spur'= track). Vertical slices are elements of A'.

2.3 simulating a step of M.

- M' reads under its head the state $z \notin E$ and all symbols a_i under the k heads of M. Let

$$\delta(z, a_1, \ldots, a_k) = (z', c_1, \ldots, c_k, s_1, \ldots, s_k)$$

then M'

- replaces on track 0 z by z'

- prints c_i on track i for all $i > 0$.
- if $s_i = L$, i.e. M moves head i to the right, machine M shifts track i to the left using a variant of machine $shiftr\ tape$ 1. This might require to divide a symbol $B \in A$ into tracks (a symbol in A')
- if $s_i = R$, i.e. M moves head i to the right, machine M shifts track i to the left using a variant of machine $shiftl\ tape$ 1.This might require to divide a symbol $B \in A$ into tracks (a symbol in A')
- after the last shift of an inscription the head returns to the cell, which has on track 0 a non blank symbol.

If the non blank portion of the tape of M' has length s this takes time and space $O(s)$.

So we Code configuration of figure 1 by configuration of M' of figure 2

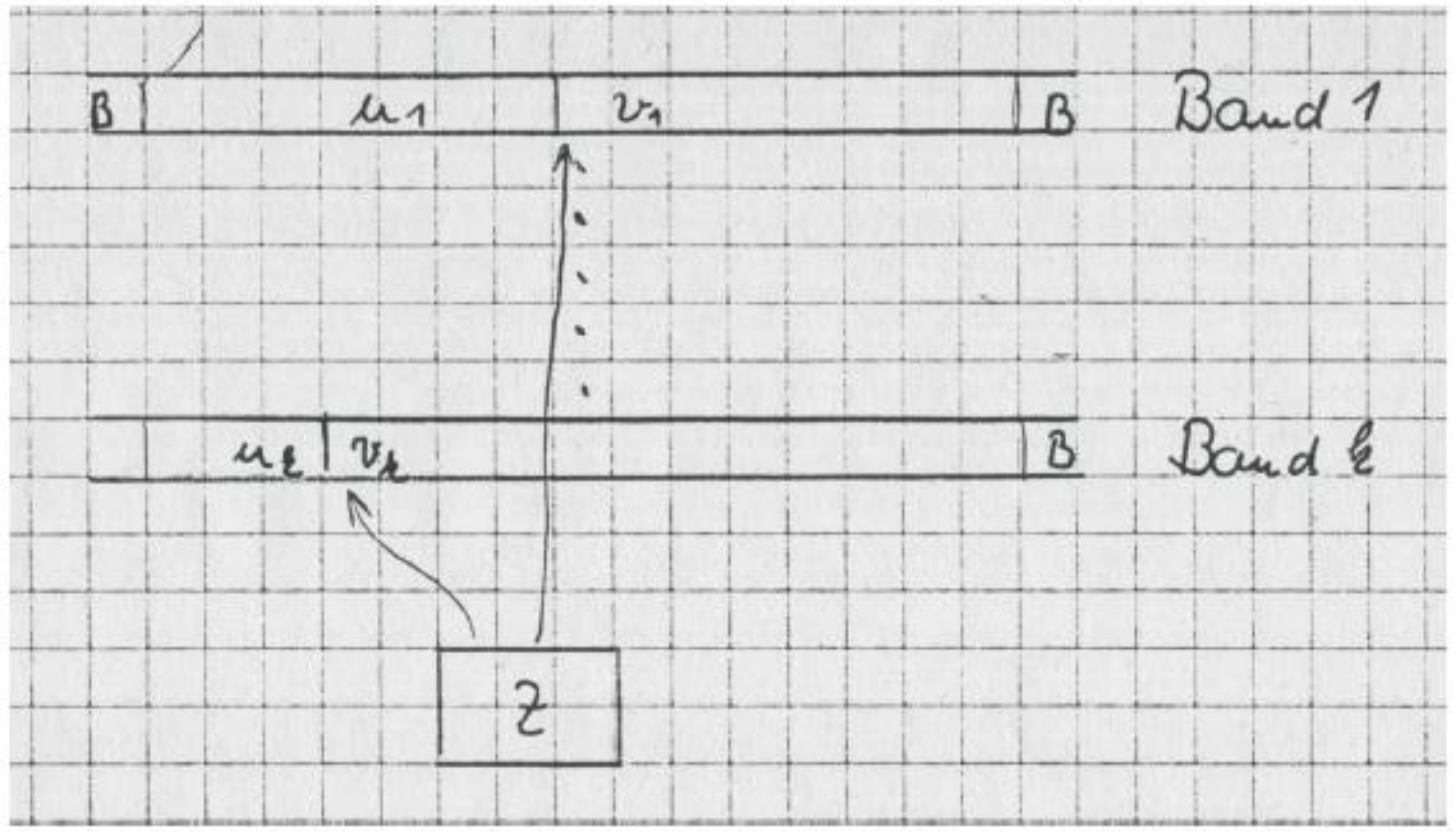

Figure 1: configuration of k-tape TM M. ('Band' = tape).

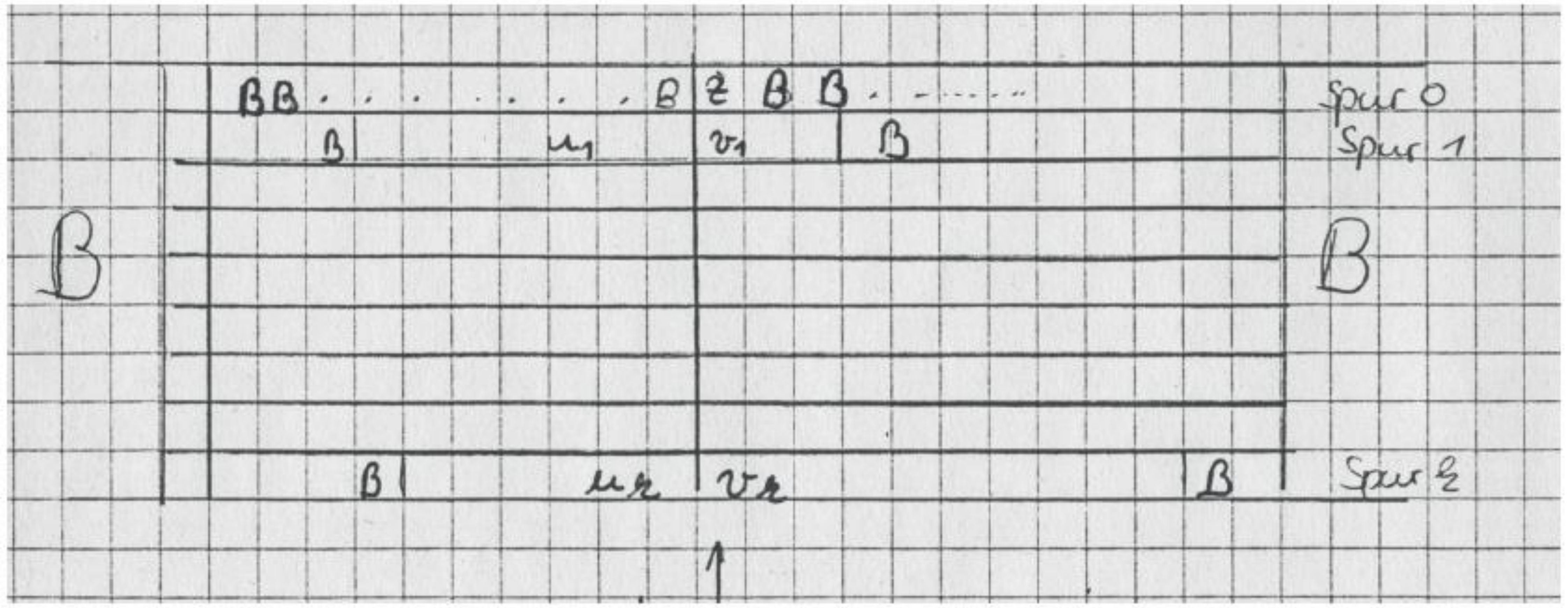

Figure 2: Configuration of 1-tape TM M' coding the configuration of figure 1 ('Spur'= track). Vertical slices are elements of A''.

2.4 post processor

If M' reads under its head a state $z_e \in E$. The configuration of M' is as in figure 5. Machine M has to undo the partition of the tape into tracks and has present the inscription of track 1 on the tape.

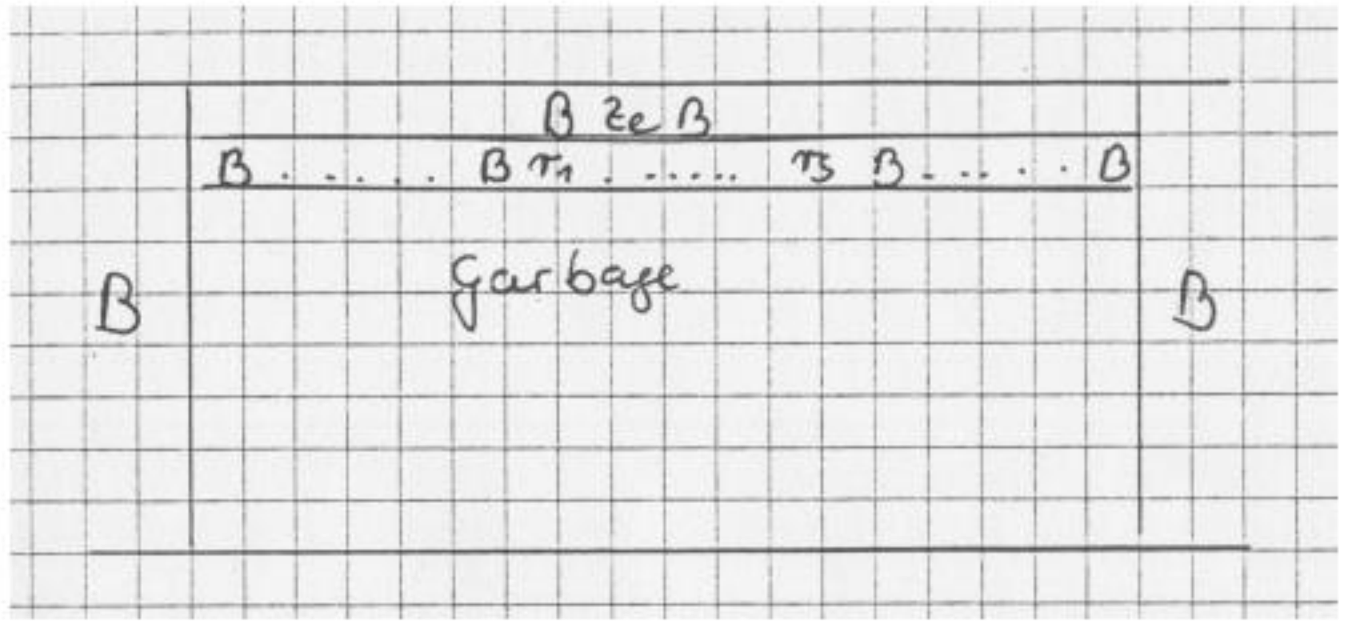

Figure 5: configuration of M' coding an end configuration of M.

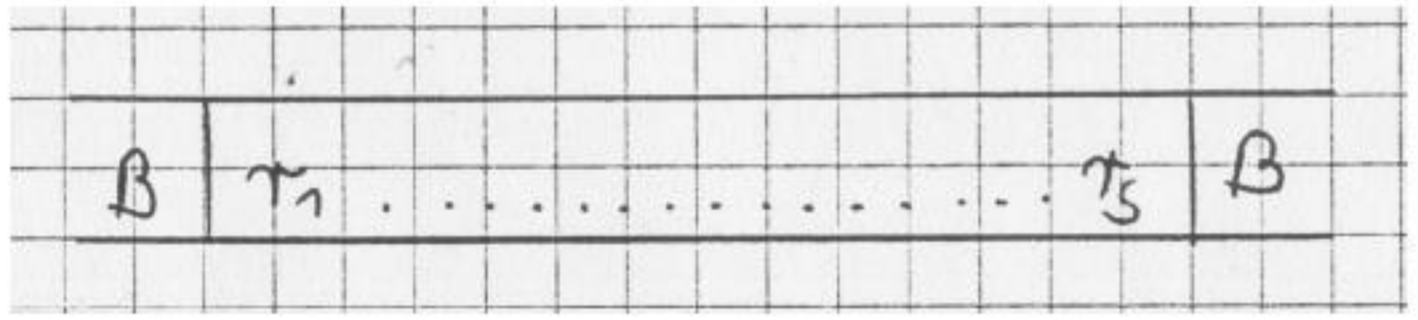

Figure 6: End configuration of M' corresponding the the configuartion of M in figure 5

If the non blank portion of the tape of M' has length s this takes time $O(s)$ and space at most s.

So we Code configuration of figure 1 by configuration of M' of figure 2

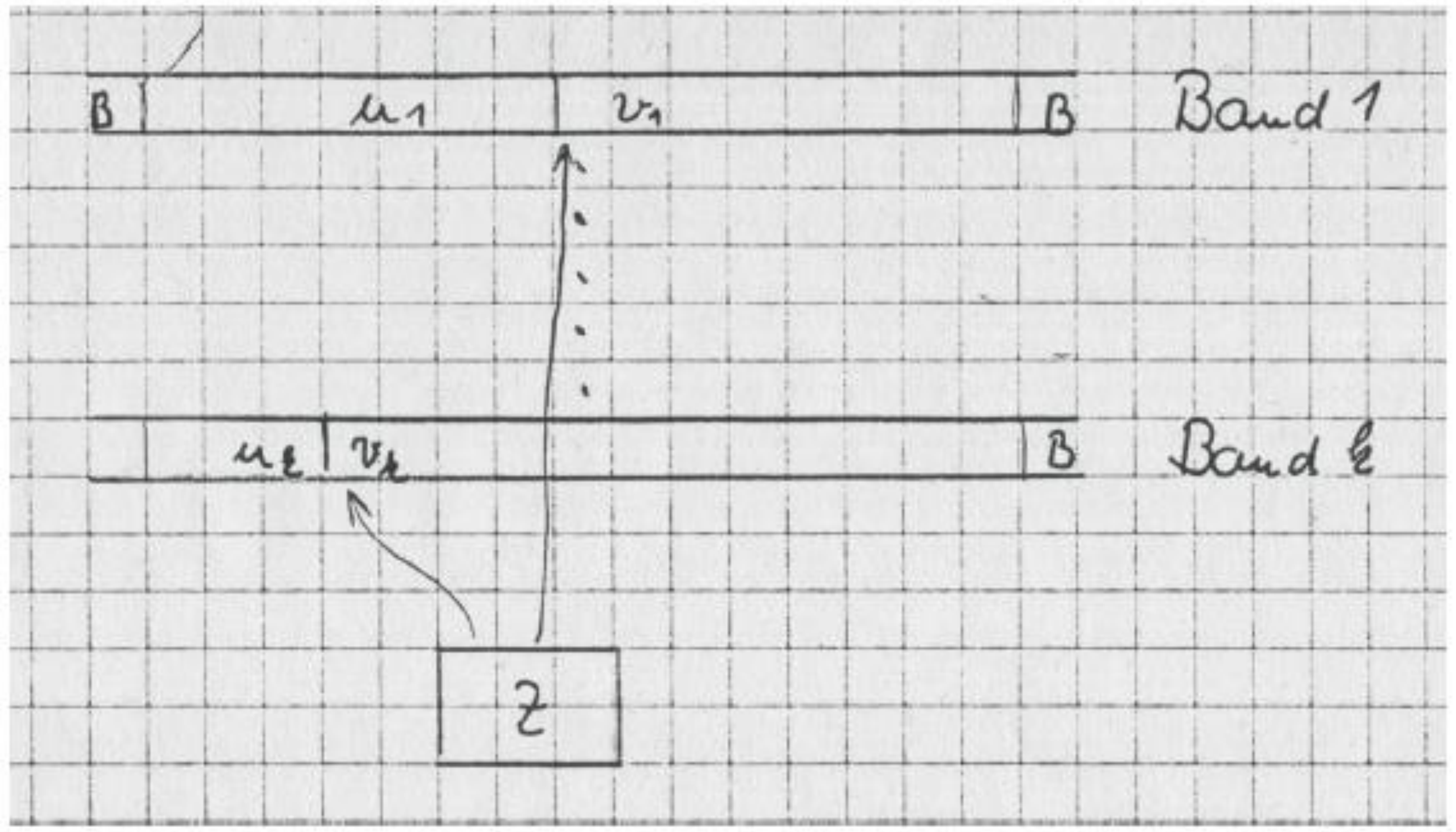

Figure 1: configuration of k-tape TM M. ('Band' = tape).

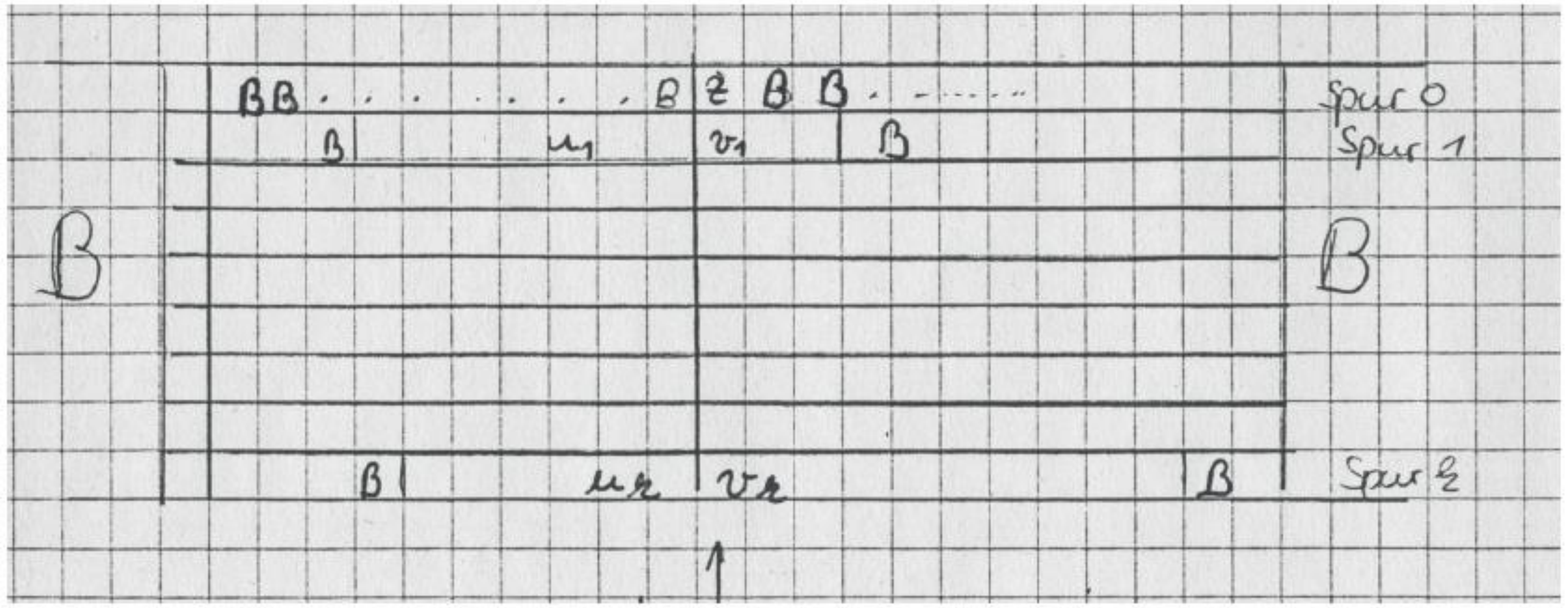

Figure 2: Configuration of 1-tape TM M' coding the configuration of figure 1 ('Spur'= track). Vertical slices are elements of A''.

2.5 complexity bounds

.

- length s of non blank portion of tape:

$$s \leq s(n) \leq n + O(t(n))$$

- simulation time $t'(n)$ bounded by

$$\boxed{t'(n) \leq O(s(n) \cdot t(n)) = O(t^2(n) + n \cdot t(n))}$$

3 Random Access Machines

3.1 Random Access Machines (RAM's): an infinite variant of MIPS

in 32 bit MIPS we have

- addresses $a \in \mathbb{B}^{32}$
- $c.pc \in \mathbb{B}^{32}$
- register contents $c.gpr(r) \in \mathbb{B}^{32}$
- memory contents $c.m(a) \in \mathbb{B}^{32}$

3.1 Random Access Machines (RAM's): an infinite variant of MIPS

in 32 bit MIPS we have

- addresses $a \in \mathbb{B}^{32}$
- $c.pc \in \mathbb{B}^{32}$
- register contents $c.gpr(r) \in \mathbb{B}^{32}$
- memory contents $c.m(a) \in \mathbb{B}^{32}$

We replace $\mathbb{B}^{32}$ by

$$V = \bigcup_{i \geq 32} \mathbb{B}^i$$

- addresses $a \in V$
- $c.pc \in V$
- register contents $c.gpr(r) \in V$
- memory contents $c.m(a) \in V$
- program occupies finite portion of initial memory $c_0.m$
- all other memory locations initially 0^{32}
- no multiplicaion instruction

3.1 Random Access Machines (RAM's): an infinite variant of MIPS

in 32 bit MIPS we have

- addresses $a \in \mathbb{B}^{32}$
- $c.pc \in \mathbb{B}^{32}$
- register contents $c.gpr(r) \in \mathbb{B}^{32}$
- memory contents $c.m(a) \in \mathbb{B}^{32}$

We replace $\mathbb{B}^{32}$ by

$$V = \bigcup_{i \geq 32} \mathbb{B}^i$$

- addresses $a \in V$
- $c.pc \in V$
- register contents $c.gpr(r) \in V$
- memory contents $c.m(a) \in V$
- program occupies finite portion of initial memory $c_0.m$
- all other memory locations initially 0^{32}
- no multiplicaion instruction

invariant :

$$c.m(a) \neq 0^{32} \quad \text{for only finitely many addreses } a$$

input tape : connect machine with a set of I/O ports to a tape devices capable of reading 32 bits at a time from the input tape.

output tape : connect machine with a set of I/O ports to a tape devices capable of writing 32 bits at a time to the output tape.

3.1 Random Access Machines (RAM's): an infinite variant of MIPS

in 32 bit MIPS we have

- addresses $a \in \mathbb{B}^{32}$
- $c.pc \in \mathbb{B}^{32}$
- register contents $c.gpr(r) \in \mathbb{B}^{32}$
- memory contents $c.m(a) \in \mathbb{B}^{32}$

We replace $\mathbb{B}^{32}$ by

$$V = \bigcup_{i \geq 32} \mathbb{B}^i$$

- addresses $a \in V$
- $c.pc \in V$
- register contents $c.gpr(r) \in V$
- memory contents $c.m(a) \in V$
- program occupies finite portion of initial memory $c_0.m$
- all other memory locations initially 0^{32}
- no multiplicaion instruction

invariant :

$$c.m(a) \neq 0^{32} \quad \text{for only finitely many addreses } a$$

input tape : connect machine with a set of I/O ports to a tape devices capable of reading 32 bits at a time from the input tape.

output tape : connect machine with a set of I/O ports to a tape devices capable of writing 32 bits at a time to the output tape.

measuring complexity

- input length: number of bits read by tape device.
- time: number of MIPS instructions executed
- used space: number of bits in PC, CPU registers and memory locations with $c.m(a) \neq 0^{32}$. Attention: this 'jumps over' non written memory cells without counting them.

3.2 space bound for time bounded RAM's

Lemma 2. *Let M be a RAM and let L be the length of the initial program. Then after t steps:*

- *no content of the pc, a register or a memory location is longer than $32+t$*
- *at most $L+t$ memory locations have a value $c.m(a) \neq 0^{32}$*

3.2 space bound for time bounded RAM's

Lemma 2. *Let M be a RAM and let L be the length of the initial program. Then after t steps:*

- *no content of the pc, a register or a memory location is longer than $32+t$*
- *at most $L+t$ memory locations have a value $c.m(a) \neq 0^{32}$*

- Let $\lambda(t)$ be the length of the largest value after t steps. We have

$$\lambda(0) = 32 \quad \text{and} \quad \lambda(t+1) \leq \lambda(t)+1$$

- Let $\sigma(t)$ be the number of memory locations a with $c^t.m(a) \neq 0^{32}$. Then

$$\sigma(0) = L \quad \text{and} \quad \sigma(t+1) \leq \sigma(t)+1$$

3.2 space bound for time bounded RAM's

Lemma 2. *Let M be a RAM and let L be the length of the initial program. Then after t steps:*

- *no content of the pc, a register or a memory location is longer than $32+t$*
- *at most $L+t$ memory locations have a value $c.m(a) \neq 0^{32}$*

4 simulation of RAMs by k-tape TM's

Lemma 3.

- *every RAM M can be simulated ba a multi tape TM M'*
- *if M is $t(n)$-time, bounded, then M' is $O(t^3)$-time bounded*

3.2 space bound for time bounded RAM's

Lemma 2. *Let M be a RAM and let L be the length of the initial program. Then after t steps:*

- *no content of the pc, a register or a memory location is longer than $32+t$*
- *at most $L+t$ memory locations have a value $c.m(a) \neq 0^{32}$*

4 simulation of RAMs by k-tape TM's

Lemma 3.
- *every RAM M can be simulated ba a multi tape TM M'*
- *if M is $t(n)$-time, bounded, then M' is $O(t^3)$-time bounded*

note

- the first part is an important part of showing that 'computability' is the same for all sufficiently powerful models, that we know.
- the second is important part of showing, that 'computability in polynomial time', i.e. computable by an $O(n^k)$)-time bounded machine for some k, is the same for *reasonable* machines.
- non reasonable power is gained, if we allow the RAM to multiply with 1 step, because lemma 2 falls apart.
- If we charge time $O(\ell)$ for a multiplication of ℓ bit numbers, the RAM model stay reasonable; it is called log-RAM.

4 simulation of RAMs by k-tape TM's

Lemma 3. • *every RAM M can be simulated ba a multi tape TM M'*

- *if M is $t(n)$-time, bounded, then M' is $O(t^3)$-time bounded*

4.1 coding RAM configurations on tapes

- read input from tape 1 (you have no choice)
- $c.pc$ on tape 2
- $c.gpr(r)$ on tape $r+3$ for $r = 0, \ldots, 31$
- $c.m$ on tape 35. Store there *in any order* the sequence of

$$\#a\#c.m(a)\# \quad \text{for } c.m(a) \neq 0$$

- the current instruction on tape 36
- the effective address on tape 37
- produce output on tape 38
- keep a few tapes for auxiliary computations (like multiplications)

4 simulation of RAMs by k-tape TM's

Lemma 3.
- *every RAM M can be simulated ba a multi tape TM M'*
- *if M is $t(n)$-time, bounded, then M' is $O(t^3)$-time bounded*

4.1 coding RAM configurations on tapes

- read input from tape 1 (you have no choice)
- $c.pc$ on tape 2
- $c.gpr(r)$ on tape $r+3$ for $r = 0, \ldots, 31$
- $c.m$ on tape 35. Store there *in any order* the sequence of

$$\#a\#c.m(a)\# \quad \text{for } c.m(a) \neq 0$$

- the current instruction on tape 36
- the effective address on tape 37
- produce output on tape 38
- keep a few tapes for auxiliary computations (like multiplications)

4.2 simulating step t

- Let PC be the current content of tape 2. Search for a string $\#PC\#I\#$ on tape 35.
- copy I to tape 36
- do an operation among registers or
- compute an effectice address EA on tape 37.
- with EA (if used) search $\#EA\#D\#$ on tape 35.
- If a load is performed, copy D to the appropriate register.
- If a store is performed copy the appropriate register content R into the place of D. If $|R| > |D|$ this may involve shifting the portion of tape 35 to the right of D by $|R| - |D|$positions to the right. If no record with EA was found, create $\#EA\#R\#$ at the end of the tape.
- record values sent to the output tape of M on tape 38

4 simulation of RAMs by k-tape TM's

Lemma 3.
- *every RAM M can be simulated ba a multi tape TM M'*
- *if M is $t(n)$-time, bounded, then M' is $O(t^3)$-time bounded*

4.1 coding RAM configurations on tapes

- read input from tape 1 (you have no choice)
- $c.pc$ on tape 2
- $c.gpr(r)$ on tape $r+3$ for $r = 0,\ldots,31$
- $c.m$ on tape 35. Store there *in any order* the sequence of

$$\#a\#c.m(a)\# \quad \text{for } c.m(a) \neq 0$$

- the current instruction on tape 36
- the effective address on tape 37
- produce output on tape 38
- keep a few tapes for auxiliary computations (like multiplications)

4.3 counting TM steps

Consider RAM computation of length $\leq t$ with input w of length n. By lemma 2:

- length of registers and memory contents

$$|PC|,|EA|,|R|,|I|,|D| \leq \lambda(t(n)) = 32 + t(n) = O(t(n))$$

- finding location of PC or EA on tape 35: from left to right $\sigma(t)$ comparisons, using 2 tapes each taking time$O(\lambda(t))$

$$O(\sigma(t)\cdot\lambda(t)) = O(t^2)$$

- length of tape 35 which codes the memeory

$$LM(t) = O(\sigma(t)\cdot\lambda(t)) = O(t^2)$$

- simulating CPU internal computations:

$$O(\lambda(t))$$

- moving a portion of tape 35 to the right (by at most $\lambda(t)$ positions) using 2 tapes

$$O(LM(t)) = O(t^2(n))$$

- thus simulating t steps:

$$O(t^3(n))$$

4 simulation of RAMs by k-tape TM's

Lemma 3.
- *every RAM M can be simulated ba a multi tape TM M'*
- *if M is $t(n)$-time, bounded, then M' is $O(t^3)$-time bounded*

4.1 coding RAM configurations on tapes

- read input from tape 1 (you have no choice)
- $c.pc$ on tape 2
- $c.gpr(r)$ on tape $r+3$ for $r = 0, \ldots, 31$
- $c.m$ on tape 35. Store there *in any order* the sequence of

$$\#a\#c.m(a)\# \quad \text{for } c.m(a) \neq 0$$

- the current instruction on tape 36
- the effective address on tape 37
- produce output on tape 38
- keep a few tapes for auxiliary computations (like multiplications)

4.4 cleaning up in the end

- $$\textit{erase tape } 1, \ \textit{tape } 1 = \textit{tape } 38$$
- as M can only output $32 = O(1)$ bits per step, this takes time $O(t(n))$

5 Some more simulations

5.1 two pushdown machines

Lemma 4. *k-tape Turing machines M can be simulated by 2-pushdown machines M'*

5 Some more simulations

5.1 two pushdown machines

Lemma 4. *k-tape Turing machines M can be simulated by 2-pushdown machines M'*

- Simulate M' by a 1 tape TM M''
- simulate M'' by keeping in one pd-tape the portion of the TM tape to the left of the head and in the second pd-tape. A head move of M'' is the simulated by popping a symbol from one pd-tape, storing it in the finite control and then pushing it on the other pd tape.

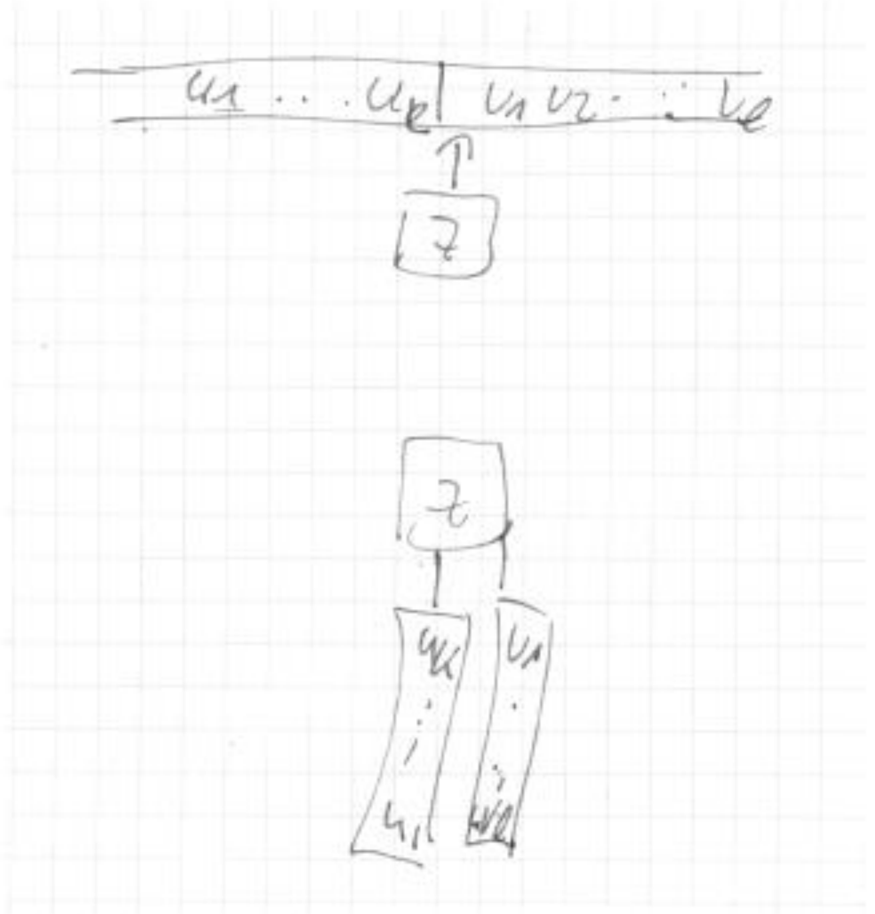

Figure 7: simulating a 1-tape TM by a 2-pd-machine

5 Some more simulations

5.1 two pushdown machines

Lemma 4. *k-tape Turing machines M can be simulated by 2-pushdown machines M'*

- Simulate M' by a 1 tape TM M''
- simulate M'' by keeping in one pd-tape the portion of the TM tape to the left of the head and in the second pd-tape. A head move of M'' is the simulated by popping a symbol from one pd-tape, storing it in the finite control and then pushing it on the other pd tape.

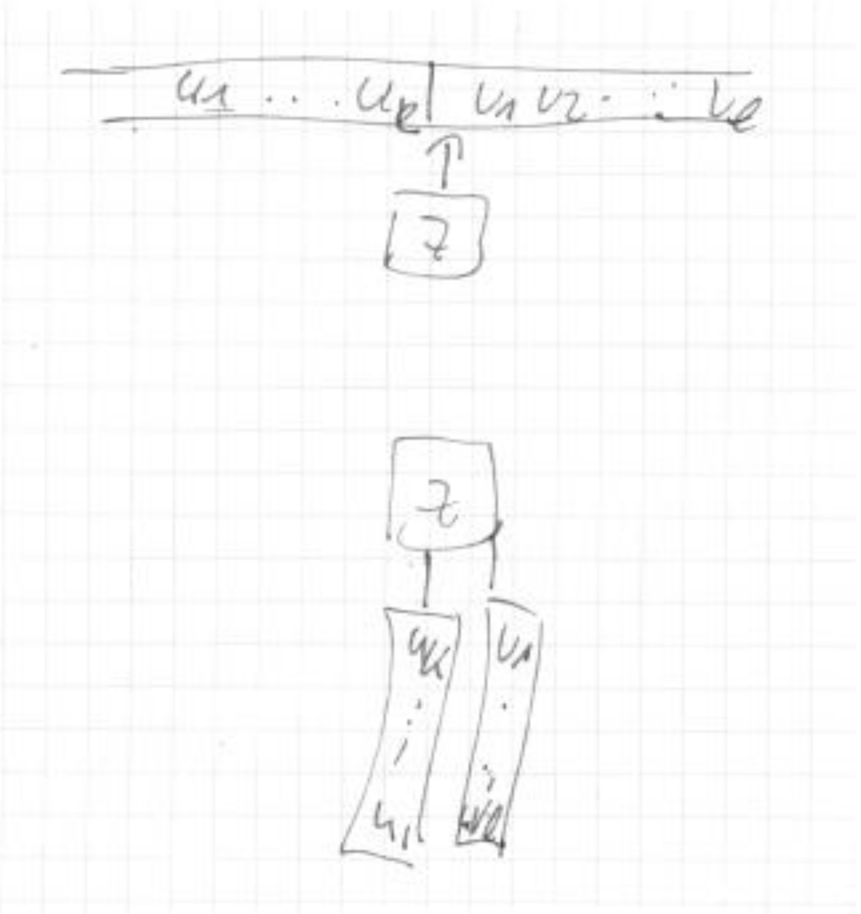

Figure 7: simulating a 1-tape TM by a 2-pd-machine

5.2 TM's can be simulated by RAMs.

Lemma 5. *k-tape Turing machines M can be simulated by RAMs M'*

- w.l.o.g $k = 1$.
- simulate it by a 2 pd-machine M''
- from some RAM location A on store one pd-tape on the even addresses $a > A$ and the other pd-tape by the odd addresses $a > A$.

5.3 abstract C-machines

unbounded abstract C machine

- you can allow values of integer variables in $\mathbb{Z}$
- but you don't have to: in the abstract C0-machine heap and stack are already unbounded.
- you have to provide instructions which read a word from an input tape and write a word to an output tape

5.3 abstract C-machines

unbounded abstract C machine

- you can allow values of integer variables in $\mathbb{Z}$
- but you don't have to: in the abstract C0-machine heap and stack are already unbounded.
- you have to provide instructions which read a word from an input tape and write a word to an output tape

Lemma 6. *Abstract C machines can be simulated by RAM's*

Proof. Use a compiler $\square$

Lemma 7. *2-pushdown machines can be simulated by abstract C-macines*

Proof. simulate each pd-tape by by a doubly lionked list on the heap. $\square$

5.3 abstract C-machines

unbounded abstract C machine

- you can allow values of integer variables in $\mathbb{Z}$
- but you don't have to: in the abstract C0-machine heap and stack are already unbounded.
- you have to provide instructions which read a word from an input tape and write a word to an output tape

Lemma 6. *Abstract C machines can be simulated by RAM's*

Proof. Use a compiler $\square$

Lemma 7. *2-pushdown machines can be simulated by abstract C-macines*

Proof. simulate each pd-tape by by a doubly lionked list on the heap. $\square$

5.4 the missing jewel (proven by Church):

Lemma 8. *Let M be a 1-tape TM and $f_M : \mathbb{N}_0^k \to \mathbb{N}_0$ be the function computed by M. Then M is μ-recursive.*

5.3 abstract C-machines

unbounded abstract C machine

- you can allow values of integer variables in $\mathbb{Z}$
- but you don't have to: in the abstract C0-machine heap and stack are already unbounded.
- you have to provide instructions which read a word from an input tape and write a word to an output tape

Lemma 6. *Abstract C machines can be simulated by RAM's*

Proof. Use a compiler $\square$

Lemma 7. *2-pushdown machines can be simulated by abstract C-macines*

Proof. simulate each pd-tape by by a doubly lionked list on the heap. $\square$

5.4 the missing jewel (proven by Church):

Lemma 8. *Let M be a 1-tape TM and $f_M : \mathbb{N}_0^k \to \mathbb{N}_0$ be the function computed by M. Then M is μ-recursive.*

- after this was known Church stated: we have it
- the community/man kind accepted it
- we show the proof next