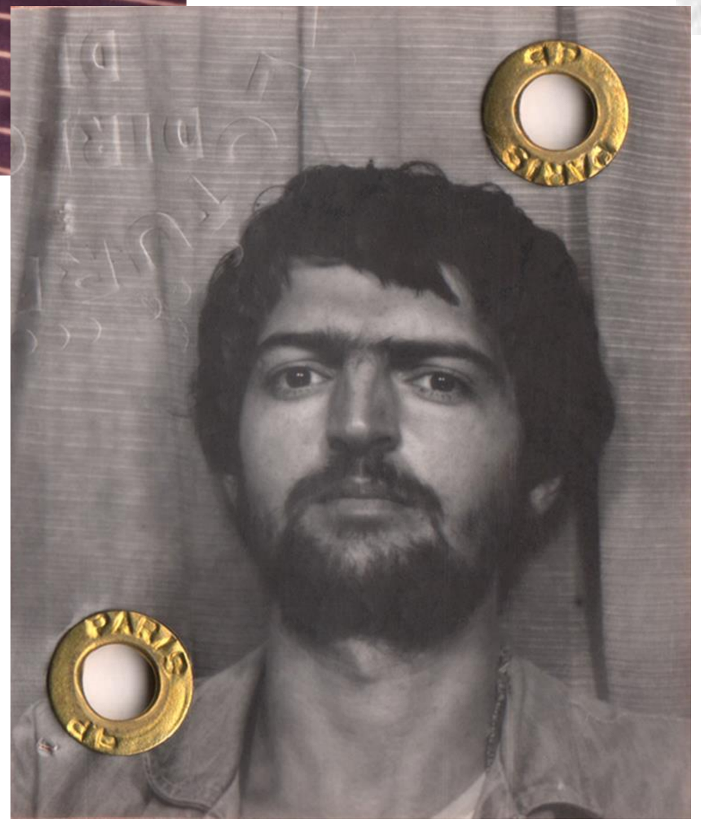
Determinism versus Nondeterminism

- $TIME(t(n)) \subseteq NTIME(t(n))$
- is the inclusion proper?
- intuitively yes for time constructible t(n)
- proof of $\boxed{TIME(n) \subsetneq NTIME(n):}$ Paul, Pippenger, Szemeredi and Trotter 1983
- based on: Reischuk and Paul 1978.

- $TIME(t(n)) \subseteq NTIME(t(n))$
- is the inclusion proper?
- intuitively yes   for time constructible t(n)
- proof of $\boxed{TIME(n) \subsetneq NTIME(n):}$ Paul, Pippenger, Szemeredi and Trotter 1983
- based on: Reischuk and Paul 1978.



1 Uniformly time bounded nondeterministic machines

reminder: nondeterministic Turing machines

- $$M = (Z, \Sigma, \delta, z_0, Z_A)$$
- set valued transition function
  $$\delta : Z \times \Sigma \to 2^{Z \times \Sigma \times \{L,N,R\}}$$
- transition *relation* $k \vdash k'$
- computations: *sequences*
  $$(k_0, k_1, \ldots, k_t) \quad \text{with} \quad k_i \vdash k_{i+1} \quad \text{for all } i$$
  computation is *accepting* if
  $$z_t \in Z_A$$
- M *accepts* w if *there exists* an accepting computation of M started with w.
- accepted language
  $$L_M = \{w \ : \ M \text{ accepts}\}$$

1 Uniformly time bounded nondeterministic machines

reminder: nondeterministic Turing machines

- $$M = (Z, \Sigma, \delta, z_0, Z_A)$$
- set valued transition function
$$\delta : Z \times \Sigma \to 2^{Z \times \Sigma \times \{L,N,R\}}$$
- transition *relation* $k \vdash k'$
- computations: *sequences*
$$(k_0, k_1, \dots, k_t) \quad \text{with} \quad k_i \vdash k_{i+1} \quad \text{for all } i$$
computation is *accepting* if
$$z_t \in Z_A$$
- M *accepts* w if *there exists* an accepting computation of M started with w.
- accepted language
$$L_M = \{w \ : \ M \text{ accepts}\}$$

def: computation tree: $tree(M,w)$ of M started with w

- root: start configuration $z_0 w$
- let k be a node in the tree, then each successor configuration k' of k, i.e. satisfying $k \vdash k'$ is a son of k
- this are all nodes

1 Uniformly time bounded nondeterministic machines

reminder: nondeterministic Turing machines

- $$M = (Z, \Sigma, \delta, z_0, Z_A)$$
- set valued transition function
$$\delta : Z \times \Sigma \to 2^{Z \times \Sigma \times \{L,N,R\}}$$
- transition *relation* $k \vdash k'$
- computations: *sequences*
$$(k_0, k_1, \ldots, k_t) \quad \text{with} \quad k_i \vdash k_{i+1} \quad \text{for all } i$$
computation is *accepting* if
$$z_t \in Z_A$$
- *M accepts w* if *there exists* an accepting computation of M started with w.
- accepted language
$$L_M = \{w \ : \ M \text{ accepts}\}$$

def: computation tree: $tree(M,w)$ of M started with w

- root: start configuration $z_0 w$
- let k be a node in the tree, then each successor configuration k' of k, i.e. satisfying $k \vdash k'$ is a son of k
- this are all nodes

acceptig nodes in the tree:

- a leaf is accepting if it's state is accepting
- an inner node k is accepting if *there exists* a son k' of k which is accepting.

Lemma 1. *M accepts w iff the root of $tree(M,w)$ is accepting*

1 Uniformly time bounded nondeterministic machines

reminder: nondeterministic Turing machines

- $$M = (Z, \Sigma, \delta, z_0, Z_A)$$
- set valued transition function
$$\delta : Z \times \Sigma \to 2^{Z \times \Sigma \times \{L,N,R\}}$$
- transition *relation* $k \vdash k'$
- computations: *sequences*
$$(k_0, k_1, \ldots, k_t) \quad \text{with} \quad k_i \vdash k_{i+1} \quad \text{for all } i$$
computation is *accepting* if
$$z_t \in Z_A$$
- *M accepts w* if *there exists* an accepting computation of M started with w.
- accepted language
$$L_M = \{w \ : \ M \text{ accepts}\}$$

def: computation tree: $tree(M, w)$ of M started with w

- root: start configuration $z_0 w$
- let k be a node in the tree, then each successor configuration k' of k, i.e. satisfying $k \vdash k'$ is a son of k
- this are all nodes

acceptig nodes in the tree:

- a leaf is accepting if it's state is accepting
- an inner node k is accepting if *there exists* a son k' of k which is accepting.

Lemma 1. *M accepts w iff the root of $tree(M, w)$ is accepting*

def: uniformly time bonded machines A nondeterministic Turing machine is *uniformly* $t(n)$ time bounded, if *every* computation of the machine with an input of length n makes at most $t(n)$ steps.

If M is uniformly $t(n)$ time bounded if for every input w with length n its computation tree $tree(M, w)$ has depth at most $t(n)$.

1 Uniformly time bounded nondeterministic machines

reminder: nondeterministic Turing machines

- $$M = (Z, \Sigma, \delta, z_0, Z_A)$$
- set valued transition function
  $$\delta : Z \times \Sigma \to 2^{Z \times \Sigma \times \{L,N,R\}}$$
- transition *relation* $k \vdash k'$
- computations: *sequences*
  $$(k_0, k_1, \ldots, k_t) \quad \text{with} \quad k_i \vdash k_{i+1} \quad \text{for all } i$$
  computation is *accepting* if
  $$z_t \in Z_A$$
- *M accepts w* if *there exists* an accepting computation of M started with w.
- accepted language
  $$L_M = \{w \,:\, M \text{ accepts}\}$$

def: computation tree: $tree(M,w)$ of M started with w

- root: start configuration $z_0 w$
- let k be a node in the tree, then each successor configuration k' of k, i.e. satisfying $k \vdash k'$ is a son of k
- this are all nodes

acceptig nodes in the tree:

- a leaf is accepting if it's state is accepting
- an inner node k is accepting if *there exists* a son k' of k which is accepting.

Lemma 1. *M accepts w iff the root of $tree(M,w)$ is accepting*

def: uniformly time bonded machines A nondeterministic Turing machine is *uniformly $t(n)$* time bounded, if *every* computation of the machine with an input of length n makes at most $t(n)$ steps.

If M is uniformly $t(n)$ time bounded if for every input w with length n its computation tree $tree(M,w)$ has depth at most $t(n)$.

Lemma 2. *For time constructible $t(n)$ every $t(n)$ time bounded nondeterministic Turing machine can be simulated by a uniformly $t(n)$ time bounded Turing machine.*

- with inputs of length n first compute $t(n)$. Then count steps on an extra tape.
- if the time bound is exceeded, stop and reject.

1 Uniformly time bounded nondeterministic machines

reminder: nondeterministic Turing machines

- $$M = (Z, \Sigma, \delta, z_0, Z_A)$$
- set valued transition function
$$\delta : Z \times \Sigma \to 2^{Z \times \Sigma \times \{L,N,R\}}$$
- transition *relation* $k \vdash k'$
- computations: *sequences*
$$(k_0, k_1, \ldots, k_t) \quad \text{with} \quad k_i \vdash k_{i+1} \quad \text{for all } i$$
computation is *accepting* if
$$z_t \in Z_A$$
- M *accepts* w if *there exists* an accepting computation of M started with w.
- accepted language
$$L_M = \{w \ : \ M \text{ accepts}\}$$

def: computation tree: $tree(M,w)$ of M started with w

- root: start configuration $z_0 w$
- let k be a node in the tree, then each successor configuration k' of k, i.e. satisfying $k \vdash k'$ is a son of k
- this are all nodes

acceptig nodes in the tree:

- a leaf is accepting if it's state is accepting
- an inner node k is accepting if *there exists* a son k' of k which is accepting.

Lemma 1. *M accepts w iff the root of $tree(M,w)$ is accepting*

def: uniformly time bonded machines A nondeterministic Turing machine is *uniformly* $t(n)$ time bounded, if *every* computation of the machine with an input of length n makes at most $t(n)$ steps.

If M is uniformly $t(n)$ time bounded if for every input w with length n its computation tree $tree(M,w)$ has depth at most $t(n)$.

Lemma 2. *For time constructible $t(n)$ every $t(n)$ time bounded nondeterministic Turing machine can be simulated by a uniformly $t(n)$ time bounded Turing machine.*

- with inputs of length n first compute $t(n)$. Then count steps on an extra tape.
- if the time bound is exceeded, stop and reject.

Lemma 3. *For time constructible functions $t(n)$ the class $NTIME(t(n))$ is the set of languages accepted by uniformly $O(t(n))$-timebounded Turing machines.*

2 Alternating Turing machines

def: alternating Turing machines

$$M = (E, U, \Sigma, \delta, z_0, Z_A)$$

- E finite set of *existential* states (correspondig exactly to states of nondeterministic machines)
- U finite set of *universal states* with $U \cap E = \emptyset$. The set of states is

$$Z = E \cup U$$

- Σ, δ, z_0, Z_A as for nondeterministic machines

2 Alternating Turing machines

def: alternating Turing machines

$$M = (E, U, \Sigma, \delta, z_0, Z_A)$$

- E finite set of *existential* states (correspondig exactly to states of nondeterministic machines)
- U finite set of *universal states* with $U \cap E = \emptyset$. The set of states is

$$Z = E \cup U$$

- Σ, δ, z_0, Z_A as for nondeterministic machines

def: semantics of alternating machines

- computation trees defined as for nondeterministic machines
- a leaf is accepting, when its state is accepting and rejecting otherwise.
- an inner node k with existential state is accepting if *there exists* a son k' of k which is accepting. Node k is rejecting otherwise
- an inner node k with universal state is accepting if *all* sons k' of k are accepting. Node k is rejecting otherwise

2 Alternating Turing machines

def: alternating Turing machines

$$M = (E, U, \Sigma, \delta, z_0, Z_A)$$

- E finite set of *existential* states (correspondig exactly to states of nondeterministic machines)
- U finite set of *universal states* with $U \cap E = \emptyset$. The set of states is

$$Z = E \cup U$$

- Σ, δ, z_0, Z_A as for nondeterministic machines

def: semantics of alternating machines

- computation trees defined as for nondeterministic machines
- a leaf is accepting, when its state is accepting and rejecting otherwise.
- an inner node k with existential state is accepting if *there exists* a son k' of k which is accepting. Node k is rejecting otherwise
- an inner node k with universal state is accepting if *all* sons k' of k are accepting. Node k is rejecting otherwise

def: time bounded alternating machines: this is defined as for uniformly time bounded nondeterministic machines. M is $t(n)$ time bounded, if for all inputs w of length n the depth of the computation tree $tree(M,w)$ is bounded by $t(n)$.

2 Alternating Turing machines

def: alternating Turing machines

$$M = (E, U, \Sigma, \delta, z_0, Z_A)$$

- E finite set of *existential* states (correspondig exactly to states of nondeterministic machines)
- U finite set of *universal states* with $U \cap E = \emptyset$. The set of states is

$$Z = E \cup U$$

- Σ, δ, z_0, Z_A as for nondeterministic machines

def: semantics of alternating machines

- computation trees defined as for nondeterministic machines
- a leaf is accepting, when its state is accepting and rejecting otherwise.
- an inner node k with existential state is accepting if *there exists* a son k' of k which is accepting. Node k is rejecting otherwise
- an inner node k with universal state is accepting if *all* sons k' of k are accepting. Node k is rejecting otherwise

def: time bounded alternating machines: this is defined as for uniformly time bounded nondeterministic machines. M is $t(n)$ time bounded, if for all inputs w of length n the depth of the computation tree $tree(M, w)$ is bounded by $t(n)$.

alternating complexity classes:

$$ATIME(t(n)) = \{L \ : \ L \text{ accepted by an } O(t(n)) - \text{time bounded alternating TM}\}$$

Alternating polynomial time is defined as

$$AP = \bigcup_k ATIME(n^k)$$

2 Alternating Turing machines

def: alternating Turing machines

$$M = (E, U, \Sigma, \delta, z_0, Z_A)$$

- E finite set of *existential* states (correspondig exactly to states of nondeterministic machines)
- U finite set of *universal states* with $U \cap E = \emptyset$. The set of states is

$$Z = E \cup U$$

- Σ, δ, z_0, Z_A as for nondeterministic machines

def: semantics of alternating machines

- computation trees defined as for nondeterministic machines
- a leaf is accepting, when its state is accepting and rejecting otherwise.
- an inner node k with existential state is accepting if *there exists* a son k' of k which is accepting. Node k is rejecting otherwise
- an inner node k with universal state is accepting if *all* sons k' of k are accepting. Node k is rejecting otherwise

def: time bounded alternating machines: this is defined as for uniformly time bounded nondeterministic machines. M is $t(n)$ time bounded, if for all inputs w of length n the depth of the computation tree $tree(M, w)$ is bounded by $t(n)$.

alternating complexity classes:

$$ATIME(t(n)) = \{L \ : \ L \text{ accepted by an } O(t(n)) - \text{time bounded alternating TM}\}$$

Alternating polynomial time is defined as

$$AP = \bigcup_k ATIME(n^k)$$

example 1 (Sipser): A *tautology* is a Boolean expression A such that $\varphi(A) = 1$ for all valuations φ.

Lemma 4.

$$\{A \ : \ A \text{ is a tautology}\} \in AP$$

- with input A universally (i.e. in universal states) guess assignment φ
- for the guessed assignment evaluate $\varphi(A)$
- accept if $\varphi(A) = 1$, otherwise reject.

2 Alternating Turing machines

def: alternating Turing machines

$$M = (E, U, \Sigma, \delta, z_0, Z_A)$$

- E finite set of *existential* states (correspondig exactly to states of nondeterministic machines)
- U finite set of *universal states* with $U \cap E = \emptyset$. The set of states is

$$Z = E \cup U$$

- Σ, δ, z_0, Z_A as for nondeterministic machines

def: semantics of alternating machines

- computation trees defined as for nondeterministic machines
- a leaf is accepting, when its state is accepting and rejecting otherwise.
- an inner node k with existential state is accepting if *there exists* a son k' of k which is accepting. Node k is rejecting otherwise
- an inner node k with universal state is accepting if *all* sons k' of k are accepting. Node k is rejecting otherwise

def: time bounded alternating machines: this is defined as for uniformly time bounded nondeterministic machines. M is $t(n)$ time bounded, if for all inputs w of length n the depth of the computation tree $tree(M, w)$ is bounded by $t(n)$.

alternating complexity classes:

$ATIME(t(n)) = \{L : L$ accepted by an $O(t(n))$ – time bounded alternating TM$\}$

Alternating polynomial time is defined as

$$AP = \bigcup_k ATIME(n^k)$$

example 1 (Sipser): A *tautology* is a Boolean expression A such that $\varphi(A) = 1$ for all valuations φ.

Lemma 4.

$$\{A : A \text{ is a tautology}\} \in AP$$

- with input A universally (i.e. in universal states) guess assignment φ
- for the guessed assignment evaluate $\varphi(A)$
- accept if $\varphi(A) = 1$, otherwise reject.

example 2 (Sipser): Each Boolean Expression with n variables computes a Boolean function $f_A : \mathbb{B}^n \to \mathbb{B}$. A Boolean expression A is *minimal* if it is the shortest expression computing f_A

Lemma 5.

$$\{A : A \text{ is minimal}\} \in AP$$

- with input A universally guess a shorter expression B
- then existentially guess a valuation φ for A and B
- evaluate both $\varphi(A)$ and $\varphi(B)$
- accept if $\varphi(a) \neq \varphi(B)$, otherwise reject.

3 Alternation bounded Turing Machines

def: alternation bounded Turing machines An alternating Turing machine M is x-alternation bounded if for for all inputs w holds: on any path in the computation tree of M started with w existential and universal states change/alternate at mist x times.

examples

- the machine accepting tautologies was 0-alternation bounded
- the machine accepting minimal expressions was 1-alternation bounded.

3 Alternation bounded Turing Machines

def: alternation bounded Turing machines An alternating Turing machine M is x-alternation bounded if for for all inputs w holds: on any path in the computation tree of M started with w existential and universal states change/alternate at mist x times.

examples

- the machine accepting tautologies was 0-alternation bounded
- the machine accepting minimal expressions was 1-alternation bounded.

complexity classes

$$
\begin{aligned}
ATIME_k^x(t(n)) &= \{L : L \text{ accepted by an } i-\text{alternation bounded } k-\text{tape TM}\} \\
ATIME^x(t(n)) &= \bigcup_k ATIME_k^x(t(n)) \\
ATIME^{fin}(t(n)) &= \bigcup ATIME^x(t(n))
\end{aligned}
$$

The latter class contains the languages which can be accepted with finitely many alternations

tape reduction

Lemma 6. *For time constructible $t(n)$*

$$\boxed{ATIME^x((t(n)) \subseteq ATIME_2^{2x+1}(t(n))}$$

3 Alternation bounded Turing Machines

def: alternation bounded Turing machines An alternating Turing machine M is x-alternation bounded if for for all inputs w holds: on any path in the computation tree of M started with w existential and universal states change/alternate at mist x times.

examples

- the machine accepting tautologies was 0-alternation bounded
- the machine accepting minimal expressions was 1-alternation bounded.

complexity classes

$$\begin{aligned} ATIME^x_k(t(n)) &= \{L : L \text{ accepted by an } i-\text{alternation bounded } k-\text{tape TM}\} \\ ATIME^x(t(n)) &= \bigcup_k ATIME^x_k(t(n)) \\ ATIME^{fin}(t(n)) &= \bigcup ATIME^x(t(n)) \end{aligned}$$

The latter class contains the languages which can be accepted with finitely many alternations

tape reduction

Lemma 6. *For time constructible $t(n)$*

$$\boxed{ATIME^x((t(n)) \subseteq ATIME^{2x+1}_2(t(n))}$$

def: step record of machine M : for a k-tape machine M this is a pair

$$((z,a),(z',b,m)) \in (Z \times \Sigma^k) \times (Z \times \Sigma^k \times \{L,N,R\}^k)$$

satisfying

$$(z',b,m) \in \delta(z,a)$$

and records the step, when machine M reads in state z symbols a on the tapes and then i) goes to state z', prints symbols b and makes head movements prescribed by m.

value of a stepping function from OS-Support in hardware lab

I took the concept from from complexity theory

3 Alternation bounded Turing Machines

complexity classes

$$\begin{aligned} ATIME_k^x(t(n)) &= \{L : L \text{ accepted by an } i-\text{alternation bounded } k-\text{tape TM}\} \\ ATIME^x(t(n)) &= \bigcup_k ATIME_k^x(t(n)) \\ ATIME^{fin}(t(n)) &= \bigcup ATIME^x(t(n)) \end{aligned}$$

The latter class contains the languages which can be accepted with finitely many alternations

tape reduction

Lemma 6. *For time constructible $t(n)$*

$$\boxed{ATIME^x((t(n)) \subseteq ATIME_2^{2x+1}(t(n))}$$

def: step record of machine M : for a k-tape machine M this is a pair

$$((z,a),(z',b,m)) \in (Z \times \Sigma^k) \times (Z \times \Sigma^k \times \{L,N,R\}^k)$$

satisfying

$$(z',b,m) \in \delta(z,a)$$

and records the step, when machine M reads in state z symbols a on the tapes and then i) goes to state z', prints symbols b and makes head movements prescribed by m.

simulating k-tape machines M with input w . Let $|w| = n$

- for $i = 0$ to $c \cdot t(n)$ guess each entry $s(i)$ of step sequence $s[0 : c \cdot t(n)]$ in two steps:

  1. existentially guess for components $s(i).a$ of the step sequence, i.e the symbols read in step i. If M starts in a universal state this adds an extra level of alternation. Also initialize

     $$s(0).z = z_0$$

     to the start state of M

  2. set

     $$s(i).z = s(i-1).z'$$

     and guess

     $$(s(i).z', s(i).b, s(i).m) \in \delta(s(i).z, s(i).a)$$

     This guess is done existentially if $s(i).z$ is existential and universally if $s(i).z$ is universal. The number of alternations is at most doubled.

3 Alternation bounded Turing Machines

complexity classes

$$
\begin{aligned}
ATIME_k^x(t(n)) &= \{L : L \text{ accepted by an } i-\text{alternation bounded } k-\text{tape TM}\} \\
ATIME^x(t(n)) &= \bigcup_k ATIME_k^x(t(n)) \\
ATIME^{fin}(t(n)) &= \bigcup ATIME^x(t(n))
\end{aligned}
$$

The latter class contains the languages which can be accepted with finitely many alternations

tape reduction

Lemma 6. *For time constructible $t(n)$*

$$\boxed{ATIME^x((t(n)) \subseteq ATIME_2^{2x+1}(t(n))}$$

def: step record of machine M : for a k-tape machine M this is a pair

$$((z,a),(z',b,m)) \in (Z \times \Sigma^k) \times (Z \times \Sigma^k \times \{L,N,R\}^k)$$

satisfying

$$(z',b,m) \in \delta(z,a)$$

and records the step, when machine M reads in state z symbols a on the tapes and then i) goes to state z', prints symbols b and makes head movements prescribed by m.

simulating k-tape machines M with input w . Let $|w| = n$

- for $i = 0$ to $c \cdot t(n)$ guess each entry $s(i)$ of step sequence $s[0 : c \cdot t(n)]$ in two steps:
  1. existentially guess for components $s(i).a$ of the step sequence, i.e the symbols read in step i. If M starts in a universal state this adds an extra level of alternation. Also initialize
  $$s(0).z = z_0$$
  to the start state of M
  2. set
  $$s(i).z = s(i-1).z'$$
  and guess
  $$(s(i).z', s(i).b, s(i).m) \in \delta(s(i).z, s(i).a)$$
  This guess is done existentially if $s(i).z$ is existential and universally if $s(i).z$ is universal. The number of alternations is at most doubled.
- if the last guessed state is rejecting reject.
- if the last guessed state is accepting check the consistency of the symbols b written and the guess of symbols a for the k tapes of M sequentially (using 2 tapes for each check). To check tape j
  1. Use the input (for $j = 1$) and components $s(i).b_j, s(i).m_j$ to simiulate the actions of M on tape j. If in any step i machine M reads a symbol $\neq a_i$, i.e. differing from the guessed symbol, abort all consistency checks and reject.
  2. if no consistency check fails, accept.

you have seen this check in the exercises

3 Alternation bounded Turing Machines

complexity classes

$$
\begin{aligned}
ATIME_k^x(t(n)) &= \{L : L \text{ accepted by an } i-\text{alternation bounded } k-\text{tape TM}\} \\
ATIME^x(t(n)) &= \bigcup_k ATIME_k^x(t(n)) \\
ATIME^{fin}(t(n)) &= \bigcup ATIME^x(t(n))
\end{aligned}
$$

The latter class contains the languages which can be accepted with finitely many alternations

tape reduction

Lemma 6. *For time constructible $t(n)$*

$$\boxed{ATIME^x((t(n)) \subseteq ATIME_2^{2x+1}(t(n))}$$

Lemma 8. *For time constructible $t(n))$*

$$L \in ATIME^x(t(n)) \to \overline{L} \in ATIME^x(t(n))$$

Given an acceptor M_u for L obtain an acceptor $M_{\overline{u}}$ for $\overline{L}$ by exchanging in M_u

- accepting and rejecting states
- universal and existential states.

time hierarchy

Lemma 9. *Let $t(n)$ and $T(n)$ be time constructible and $t(n) = o(T(n))$. Then*

$$ATIME^x(t(n)) \subsetneq ATIME^{2x+2}(T(n))$$

3 Alternation bounded Turing Machines

complexity classes

$$\begin{aligned} ATIME_k^x(t(n)) &= \{L : L \text{ accepted by an } i-\text{alternation bounded } k-\text{tape TM}\} \\ ATIME^x(t(n)) &= \bigcup_k ATIME_k^x(t(n)) \\ ATIME^{fin}(t(n)) &= \bigcup ATIME^x(t(n)) \end{aligned}$$

The latter class contains the languages which can be accepted with finitely many alternations

tape reduction

Lemma 6. *For time constructible $t(n)$*

$$\boxed{ATIME^x((t(n)) \subseteq ATIME_2^{2x+1}(t(n))}$$

closure under complement

Lemma 8. *For time constructible $t(n))$*

$$L \in ATIME^x(t(n)) \to \overline{L} \in ATIME^x(t(n))$$

Given an acceptor M_u for L obtain an acceptor $M_{\overline{u}}$ for $\overline{L}$ by exchanging in M_u

- accepting and rejecting states
- universal and existential states.

time hierarchy

Lemma 9. *Let $t(n)$ and $T(n)$ be time constructible and $t(n) = o(T(n))$. Then*

$$ATIME^x(t(n)) \subsetneq ATIME^{2x+2}(T(n))$$

- show

$$ATIME^x(t(n)) \subseteq ATIME_2^{2x+1}(t(n)) \subsetneq ATIME_3^{2x+2}(T(n))$$

- for the second (proper) inclusion proceed as in the time hierarchy theorem:

1. use a universal 2 tape alternating machine M_u
2. with input $u\#v$ simulate 2-tape machine $M_{\overline{u}}$ for $t(n)/|\overline{u}|$ steps.
3. use an extra tape to count steps.

4 Assume $DTIME(n) = NTIME(n)$

Collapsing the alternation bounded hierarchy $ATIME^x(n)$

Lemma 10. *Assume $DTIME(n) = NTIME(n)$. Then for all x*

$$ATIME^x(n) = DTIME(n)$$

4 Assume $DTIME(n) = NTIME(n)$

Collapsing the alternation bounded hierarchy $ATIME^x(n)$

Lemma 10. *Assume* $DTIME(n) = NTIME(n)$*. Then for all* x

$$ATIME^x(n) = DTIME(n)$$

induction on x.

- $x = 0$: Let
$$L \in ATIME^0(n)$$
accepted my machine M.
  1. case: states of M existential:
$$L \in NTIME(n) = DTIME(n) \text{by assumption}$$

4 Assume $DTIME(n) = NTIME(n)$

Collapsing the alternation bounded hierarchy $ATIME^x(n)$

Lemma 10. *Assume $DTIME(n) = NTIME(n)$. Then for all x*

$$ATIME^x(n) = DTIME(n)$$

induction on x.

- $x = 0$: Let
$$L \in ATIME^0(n)$$
  accepted my machine M.
  1. case: states of M existential:
  $$L \in NTIME(n) = DTIME(n) \quad \text{by assumption}$$
  2. case: states of M universal. Then (proof of lemma 8)
  $$\overline{L} \in NTIME(n) = DTIME(n)$$
  $$\rightarrow L \in DTIME(n)$$

4 Assume $DTIME(n) = NTIME(n)$

Collapsing the alternation bounded hierarchy $ATIME^x(n)$

Lemma 10. *Assume $DTIME(n) = NTIME(n)$. Then for all x*

$$ATIME^x(n) = DTIME(n)$$

induction on x.

- $x = 0$: Let
$$L \in ATIME^0(n)$$
accepted my machine M.

  1. case: states of M existential:
  $$L \in NTIME(n) = DTIME(n) \quad \text{by assumption}$$
  2. case: states of M universal. Then (proof of lemma 8)
  $$\overline{L} \in NTIME(n) = DTIME(n)$$
  $$\rightarrow L \in DTIME(n)$$

- $x \rightarrow x+1$: Let M be $(x+1)$ alternation bounded and $O(n)$ time bounded acceptor for L. Modify M started with input w

  1. run until first alternation and interrupt there:
  2. partition tape 1 into $2k$ tracks and store there state of M, head postions and instriptions of all tapes
  3. erase tapes except tape 1; move head to start of inscription
  4. now run machine M', which reconstructs the configuration where M interrupted and then behaves like M started from the saved state.

4 Assume $DTIME(n) = NTIME(n)$

Collapsing the alternation bounded hierarchy $ATIME^x(n)$

Lemma 10. *Assume $DTIME(n) = NTIME(n)$. Then for all x*

$$ATIME^x(n) = DTIME(n)$$

induction on x.

- $x = 0$: Let
$$L \in ATIME^0(n)$$
accepted my machine M.
  1. case: states of M existential:
  $$L \in NTIME(n) = DTIME(n) \quad \text{by assumption}$$
  2. case: states of M universal. Then (proof of lemma 8)
  $$\overline{L} \in NTIME(n) = DTIME(n)$$
  $$\rightarrow L \in DTIME(n)$$

- $x \rightarrow x+1$: Let M be $(x+1)$ alternation bounded and $O(n)$ time bounded acceptor for L. Modify M started with input w
  1. run until first alternation and interrupt there:
  2. partition tape 1 into $2k$ tracks and store there state of M, head postions and instriptions of all tapes
  3. erase tapes except tape 1; move head to start of inscription
  4. now run machine M', which reconstructs the configuration where M interrupted and then behaves like M started from the saved state.

- machine M' is $O(n)$ time bounded and x-alternation bounded. By induction hypothesis we can replace it by deterministic $O(n)$-time bounded machine M''.
- replace in the modified machine M' by M''. The resulting machine is 0-alternation bounded. Apply the base case of the lemma.

4 Assume $DTIME(n) = NTIME(n)$

Collapsing the alternation bounded hierarchy $ATIME^x(n)$

Lemma 10. *Assume $DTIME(n) = NTIME(n)$. Then for all x*

$$ATIME^x(n) = DTIME(n)$$

induction on x.

- $x = 0$: Let
$$L \in ATIME^0(n)$$
accepted my machine M.
  1. case: states of M existential:
  $$L \in NTIME(n) = DTIME(n) \quad \text{by assumption}$$
  2. case: states of M universal. Then (proof of lemma 8)
  $$\overline{L} \in NTIME(n) = DTIME(n)$$
  $$\rightarrow L \in DTIME(n)$$

- $x \rightarrow x+1$: Let M be $(x+1)$ alternation bounded and $O(n)$ time bounded acceptor for L. Modify M started with input w
  1. run until first alternation and interrupt there:
  2. partition tape 1 into $2k$ tracks and store there state of M, head postions and instriptions of all tapes
  3. erase tapes except tape 1; move head to start of inscription
  4. now run machine M', which reconstructs the configuration where M interrupted and then behaves like M started from the saved state.

- machine M' is $O(n)$ time bounded and x-alternation bounded. By induction hypothesis we can replace it by deterministic $O(n)$-time bounded machine M''.
- replace in the modified machine M' by M''. The resulting machine is 0-alternation bounded. Apply the base case of the lemma.

padding

Lemma 11. *Assume $DTIME(n) = NTIME(n)$ and $t(n)$ is time constructible. Then*

$$ATIME^{fin}(t(n)) = DTIME(t(n))$$

Proof. Pad input w to $w\#^{t(|w|)-|w|}$ □

5 Segregators of graphs

5.1 Definitions

removing a set of nodes S from a dag and its adjacent edges

- Let $G = (V, E)$ be a DAG and $S \subset V$
- define $G - S = (V', E')$ by
  1. $V' = V \setminus S$
  2. $V' = E \cap (V' \times V')$

5 Segregators of graphs

5.1 Definitions

removing a set of nodes S from a dag and its adjacent edges

- Let $G = (V, E)$ be a DAG and $S \subset V$
- define $G - S = (V', E')$ by
  1. $V' = V \setminus S$
  2. $V' = E \cap (V' \times V')$

def: segregator A subset $S \subset V$ is an $s(n)$- segregator of dag $G = (V, E)$ if with $|V| = n$ holds

- $$|S| = O(s(n))$$
- every node in $G - S$ has at most $O(s(n))$ predecessors

5 Segregators of graphs

5.1 Definitions

removing a set of nodes S from a dag and its adjacent edges

- Let $G = (V,E)$ be a DAG and $S \subset V$
- define $G - S = (V',E')$ by
  1. $V' = V \setminus S$
  2. $V' = E \cap (V' \times V')$

def: segregator A subset $S \subset V$ is an $s(n)$- segregator of dag $G = (V,E)$ if with $|V| = n$ holds

- $$|S| = O(s(n))$$
- every node in $G - S$ has at most $O(s(n))$ predecessors

5.2 Block respecting Turing Machines

goal: simplify structure of TM computation graphs.

def: block respectig Let M be a deterministic $t(n)$-time bounded k-tape TM. On input with length n divide time into time intervals and tapes into blocks of lengthh λ. Machine M is *block respecting* if heads cross block boundaries only as the last step of time intervals.

5 Segregators of graphs

5.1 Definitions

removing a set of nodes S from a dag and its adjacent edges

- Let $G = (V, E)$ be a DAG and $S \subset V$
- define $G - S = (V', E')$ by
  1. $V' = V \setminus S$
  2. $V' = E \cap (V' \times V')$

def: segregator A subset $S \subset V$ is an $s(n)$- segregator of dag $G = (V, E)$ if with $|V| = n$ holds

- $$|S| = O(s(n))$$
- every node in $G - S$ has at most $O(s(n))$ predecessors

5.2 Block respecting Turing Machines

goal: simplify structure of TM computation graphs.

def: block respectig Let M be a deterministic $t(n)$-time bounded k-tape TM. On input with length n divide time into time intervals and tapes into blocks of lengthh λ. Machine M is *block respecting* if heads cross block boundaries only as the last step of time intervals.

Lemma 12. *Let $t(n)$ and $\lambda(n)$ be time constructible and let M be a $t(n)$-time bounded k-tape Turing machine. Then M can be simulated by a block respecting $O(t(n))$-time bounded $(k+1)$-tape Turing machine.*

5 Segregators of graphs

5.1 Definitions

removing a set of nodes S from a dag and its adjacent edges

- Let $G = (V, E)$ be a DAG and $S \subset V$
- define $G - S = (V', E')$ by
  1. $V' = V \setminus S$
  2. $V' = E \cap (V' \times V')$

def: segregator A subset $S \subset V$ is an $s(n)$- segregator of dag $G = (V, E)$ if with $|V| = n$ holds

-
$$|S| = O(s(n))$$
- every node in $G - S$ has at most $O(s(n))$ predecessors

5.2 Block respecting Turing Machines

goal: simplify structure of TM computation graphs.

def: block respectig Let M be a deterministic $t(n)$-time bounded k-tape TM. On input with length n divide time into time intervals and tapes into blocks of lengthh λ. Machine M is *block respecting* if heads cross block boundaries only as the last step of time intervals.

Lemma 12. *Let $t(n)$ and $\lambda(n)$ be time constructible and let M be a $t(n)$-time bounded k-tape Turing machine. Then M can be simulated by a block respecting $O(t(n))$-time bounded $(k+1)$-tape Turing machine.*

- extra tape (as usual) for counting up to λ
- for each block b_2 code on 3 tracks b_2 together with its neighbors b_1 and b_3 as shown in figure 1

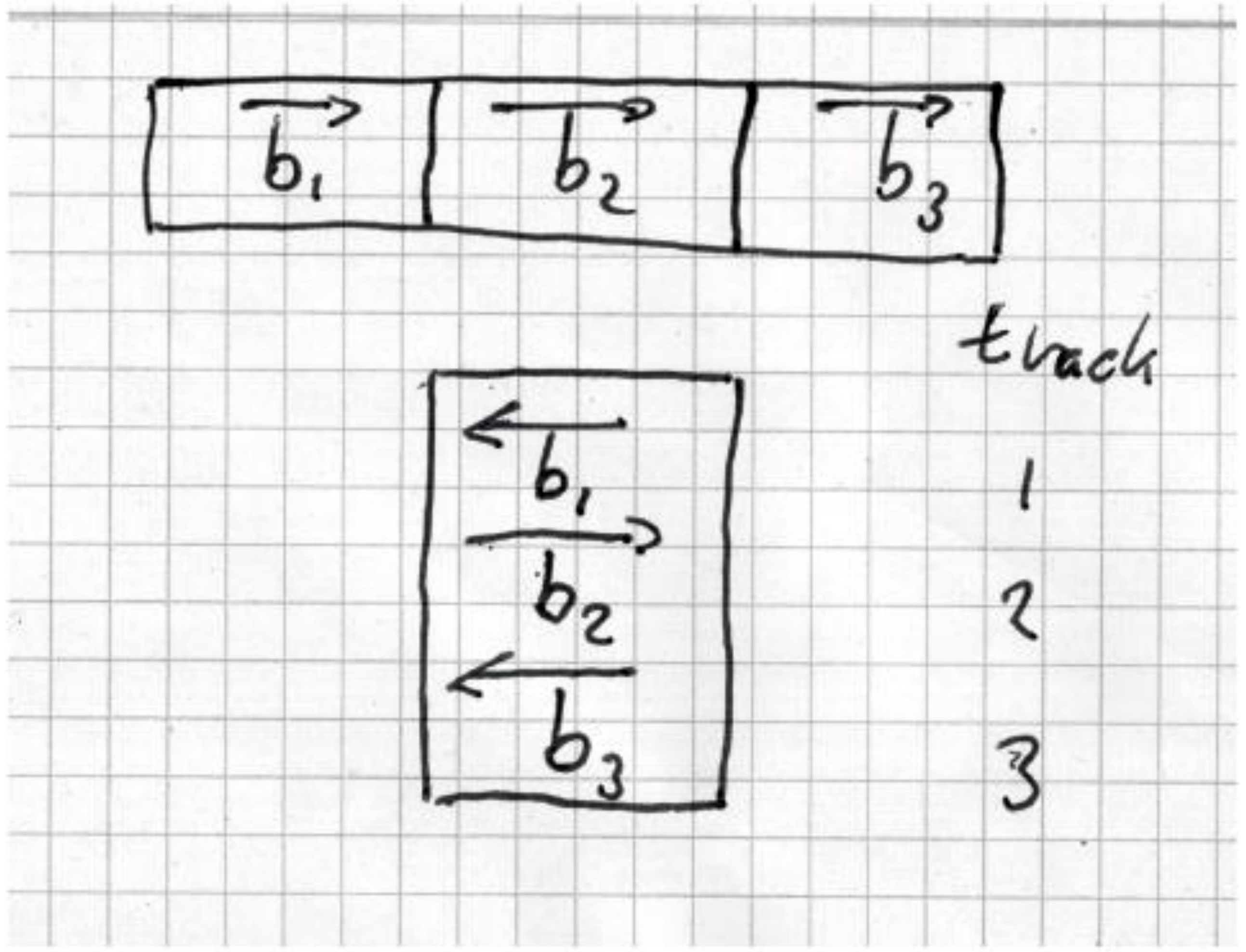


Figure 1: Folding 3 successive blocks on 3 tracks of 1 block. The outer blocks are written backwards on their tracks.

5 Segregators of graphs

5.1 Definitions

removing a set of nodes S from a dag and its adjacent edges

- Let $G = (V,E)$ be a DAG and $S \subset V$
- define $G - S = (V', E')$ by
  1. $V' = V \setminus S$
  2. $V' = E \cap (V' \times V')$

def: segregator A subset $S \subset V$ is an $s(n)$- segregator of dag $G = (V,E)$ if with $|V| = n$ holds

- $$|S| = O(s(n))$$
- every node in $G - S$ has at most $O(s(n))$ predecessors

5.2 Block respecting Turing Machines

goal: simplify structure of TM computation graphs.

def: block respectig Let M be a deterministic $t(n)$-time bounded k-tape TM. On input with length n divide time into time intervals and tapes into blocks of lengthh λ. Machine M is *block respecting* if heads cross block boundaries only as the last step of time intervals.

Lemma 12. *Let $t(n)$ and $\lambda(n)$ be time constructible and let M be a $t(n)$-time bounded k-tape Turing machine. Then M can be simulated by a block respecting $O(t(n))$-time bounded $(k+1)$-tape Turing machine.*

- extra tape (as usual) for counting up to λ
- for each block b_2 code on 3 tracks b_2 together with its neighbors b_1 and b_3 as shown in figure 1

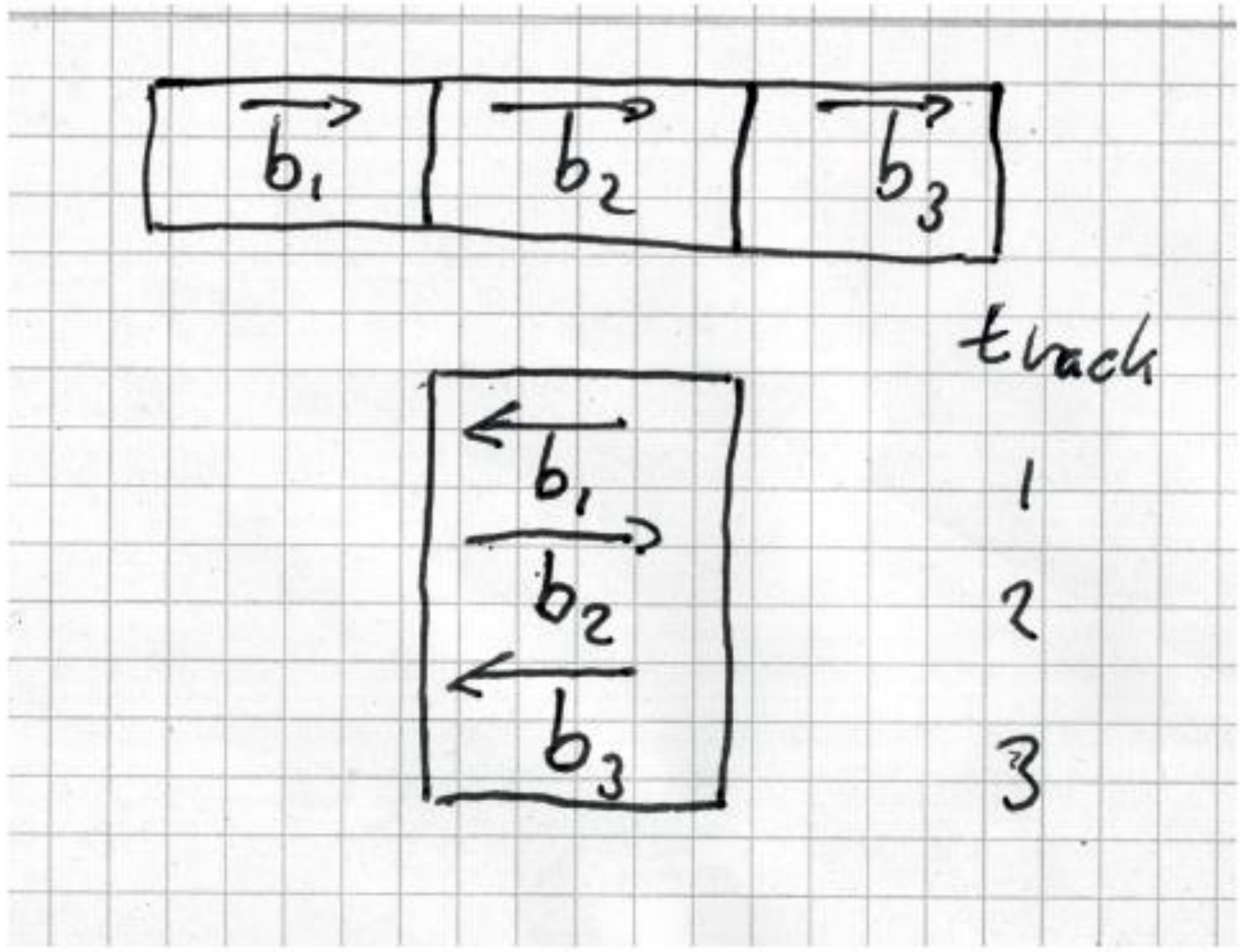


Figure 1: Folding 3 successive blocks on 3 tracks of 1 block. The outer blocks are written backwards on their tracks.

- for each time interval t
  1. first simulate without crossing block boundaries
  2. then fix the coding of blocks in a block respecting way.

5.2 Block respecting Turing Machines

goal: simplify structure of TM computation graphs.

def: block respectig Let M be a deterministic $t(n)$-time bounded k-tape TM. On input with length n divide time into time intervals and tapes into blocks of lengthh λ. Machine M is *block respecting* if heads cross block boundaries only as the last step of time intervals.

Lemma 12. *Let $t(n)$ and $\lambda(n)$ be time constructible and let M be a $t(n)$-time bounded k-tape Turing machine. Then M can be simulated by a block respecting $O(t(n))$-time bounded $(k+1)$-tape Turing machine.*

5.3 The structure of computation graphs for block respecting machines

- for a single tape and edges between successive time intervals: illustrated in figure 2.
  1. spine with edges $(t,t+1)$ between successive intervals
  2. spine + left half and spine + right half are both planaris planar
  3. in each half edegs outside the spine have a bracket structure.
- for k tapes: $2k$ such halves glued together at a common spine.

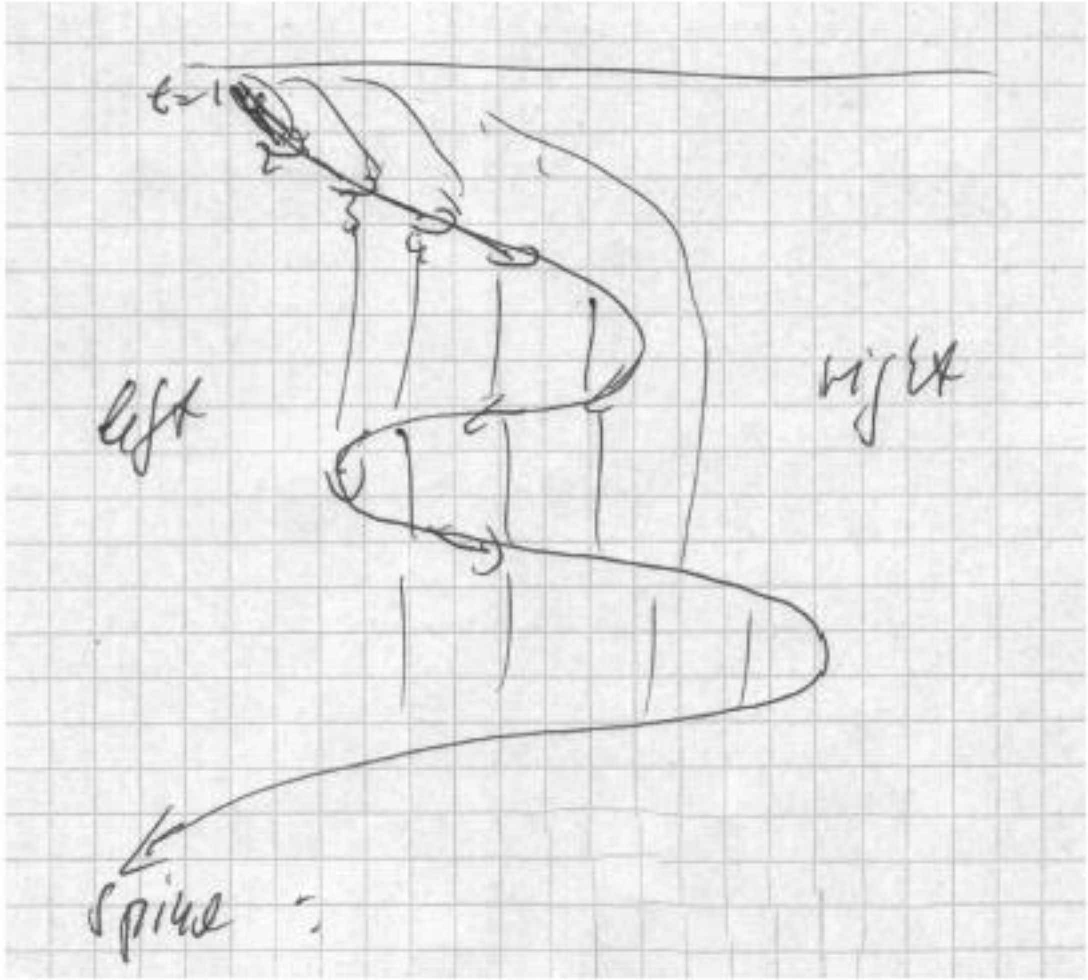


Figure 2: TM computation graph for 1 tape and the spine of a block respecting machine. Edges in each half do not cross

Szemeredi and Trotter !

5.4 Segregator lemma and consequences

def: log*

$$
\begin{array}{rcl}
T(1) & = & 2 \\
T(x) & = & 2^{T(x-1)} \\
T(x) & = & 2^{2^{2^{\cdots}}} \quad x \text{ times} \\
log^* n & = & \max\{x \,:\, T(x) \le n\}
\end{array}
$$

unbounded but *very* slowly growing function.

5.4 Segregator lemma and consequences

def: log*

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\
log^{*}n &= \max\{x \,:\, T(x) \leq n\}
\end{aligned}
$$

unbounded but *very* slowly growing function.

Segregator lemma

Lemma 13. *Let Let $G = (V, E)$ be a TM computation graph of a block respecting Turing machine and $|V| = n$. Then G has an $O(n/\log^{*} n)$-segregator*

Proof in section 6

5.4 Segregator lemma and consequences

def: $\log^*$

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \le n\} \end{aligned}$$

unbounded but *very* slowly growing function.

Segregator lemma

$\boxed{\text{\textbf{Lemma 13.} \textit{Let Let } G = (V,E) \textit{ be a TM computation graph of a block respecting Turing machine and } |V| = n\text{. \textit{Then} } G \textit{ has an } O(n/\log^* n)\text{\textit{-segregator}}}}$

Proof in section 6

fast simulation of deterministic machines with a bounded number of alternations

Lemma 14. *Let $t(n)$ be time constructible. Then*

$$DTIME(t(n)) \subseteq ATIME^2(t(n)/\log^*(t(n))$$

5.4 Segregator lemma and consequences

def: $\log^*$

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

unbounded but *very* slowly growing function.

Segregator lemma

Lemma 13. *Let Let $G = (V, E)$ be a TM computation graph of a block respecting Turing machine and $|V| = n$. Then G has an $O(n/\log^* n)$-segregator*

Proof in section 6

fast simulation of deterministic machines with a bounded number of alternations

Lemma 14. *Let $t(n)$ be time constructible. Then*

$$DTIME(t(n)) \subseteq ATIME^2(t(n)/\log^*(t(n))$$

Let M be a $C \cdot t(n)$-time bounded block respecting k-tape machine. With input of length n simulate as follows

- choose $t = C \cdot t(n)$ and time interval length and block size
$$\lambda = t^{2/3}$$
- existentially guess head positions at end of time intervals
- compute the computation graph using block size λ. This graph has $t^{1/3}$ nodes

5.4 Segregator lemma and consequences

def: $\log^*$

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

unbounded but *very* slowly growing function.

Segregator lemma

Lemma 13. *Let Let $G = (V, E)$ be a TM computation graph of a block respecting Turing machine and $|V| = n$. Then G has an $O(n/\log^* n)$-segregator*

Proof in section 6

fast simulation of deterministic machines with a bounded number of alternations

Lemma 14. *Let $t(n)$ be time constructible. Then*

$$DTIME(t(n)) \subseteq ATIME^2(t(n)/\log^*(t(n))$$

Let M be a $C \cdot t(n)$-time bounded block respecting k-tape machine. With input of length n simulate as follows

- choose $t = C \cdot t(n)$ and time interval length and block size
$$\lambda = t^{2/3}$$
- existentially guess head positions at end of time intervals
- compute the computation graph using block size λ. This graph has $t^{1/3}$ nodes
- existentially guess a segregator S of size
$$O(t^{1/3}/\log^*(t^{1/3})) = O(t^{1/3}/\log^* t)$$
- existentially guess results $res(i)$ for all $i \in S$. This takes time
$$O(t^{2/3} \cdot |S|) = O(t/\log^* t)$$

5.4 Segregator lemma and consequences

def: $\log^*$

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

unbounded but *very* slowly growing function.

Segregator lemma

Lemma 13. *Let Let $G = (V,E)$ be a TM computation graph of a block respecting Turing machine and $|V| = n$. Then G has an $O(n/\log^* n)$-segregator*

Proof in section 6

fast simulation of deterministic machines with a bounded number of alternations

Lemma 14. *Let $t(n)$ be time constructible. Then*

$$DTIME(t(n)) \subseteq ATIME^2(t(n)/\log^*(t(n))$$

Let M be a $C \cdot t(n)$-time bounded block respecting k-tape machine. With input of length n simulate as follows

- choose $t = C \cdot t(n)$ and time interval length and block size
$$\lambda = t^{2/3}$$
- existentially guess head positions at end of time intervals
- compute the computation graph using block size λ. This graph has $t^{1/3}$ nodes
- existentially guess a segregator S of size
$$O(t^{1/3}/\log^*(t^{1/3})) = O(t^{1/3}/\log^* t)$$
- existentially guess results $res(i)$ for all $i \in S$. This takes time
$$O(t^{2/3} \cdot |S|) = O(t/\log^* t)$$
- universally choose a node $i \in S$. Trace the set P of its predecessors in S or the input (if there are too many predecessors, reject because S is not a segregator) and compute $res(i)$ from the $res(j)$, $j \in P$ and the input. If the result equals the guessed $res(i)$ continue, otherwise reject. This takes time
$$O(t^{2/3} \cdot |P|) = O(t/\log^* t)$$
- accept iff the state at the end of the last time interval is accepting

5.4 Segregator lemma and consequences

def: $\log^*$

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

unbounded but *very* slowly growing function.

Segregator lemma

Lemma 13. *Let Let $G = (V,E)$ be a TM computation graph of a block respecting Turing machine and $|V| = n$. Then G has an $O(n/\log^* n)$-segregator*

Proof in section 6

fast simulation of deterministic machines with a bounded number of alternations

Lemma 14. *Let $t(n)$ be time constructible. Then*

$$DTIME(t(n)) \subseteq ATIME^2(t(n)/\log^*(t(n))$$

proving $DTIME(n) \neq NTIME(n)$: otherwise set $T(n) = n \log^* n$

$$\begin{aligned} ATIME^{fin}(T(n)) &\subseteq DTIME(T(n)) \quad \text{(lemma 11)} \\ &\subseteq ATIME^2(n) \\ &\subsetneq ATIME_3^6(T(n)) \quad \text{(time hierarchy)} \\ &\subseteq ATIME^{fin}(T(n)) \end{aligned}$$

6 Proof of the segregator lemma

def: log*

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

def: log*

6 Proof of the segregator lemma

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2\cdots}} \quad x \text{ times} \\
log^*n &= \max\{x : T(x) \leq n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\geq T(4) = 2^{16} \\
k &= \lceil \frac{\log^* n}{3} \rceil \\
&\geq 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

def: log*

6 Proof of the segregator lemma

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2\cdots}} \quad x \text{ times} \\
log^* n &= \max\{x \,:\, T(x) \le n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\ge T(4) = 2^{16} \\
k &= \lceil \frac{\log^* n}{3} \rceil \\
&\ge 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

Lemma 15.

$$e_\ell \le T(k+2\ell)$$

6 Proof of the segregator lemma

def: log*

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\
log^* n &= \max\{x : T(x) \leq n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\geq T(4) = 2^{16} \\
k &= \lceil \frac{\log^* n}{3} \rceil \\
&\geq 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

Lemma 15.

$$e_\ell \leq T(k+2\ell)$$

- induction on ℓ
- $\ell = 0$:

$$e_0 = 1 < 2 = T(1) \leq T(k+0)$$

- $\ell \to \ell + 1$:

$$
\begin{aligned}
e_{\ell+1} &\leq 2^{(\log k)\cdot(2+2T(k+2\ell)} \quad \text{IH} \\
&\leq 2^{T(k+2\ell+1)} \\
&= T(k+2(\ell+1))
\end{aligned}
$$

def: log*

6 Proof of the segregator lemma

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \le n\} \end{aligned}$$

6.1 Some very quickly growing sequences of numbers

Let

$$\begin{aligned} n &\ge T(4) = 2^{16} \\ k &= \lceil \frac{\log^* n}{3} \rceil \\ &\ge 2 \\ e_0 &= 1 \\ e_{\ell+1} &= k^{2+2e_\ell} \end{aligned}$$

Lemma 15.

$$e_\ell \le T(k + 2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \le n$

$$\begin{aligned} e_{k-1} &\le T(k + 2(k-1)) \text{ (lemma 15)} \\ &= T(3k-2) \\ &\le T(3(\frac{\log^* n}{3} + 2/3) - 2) \\ &= T(\log^* n) \\ &\le n \end{aligned}$$

def: $\log^*$

6 Proof of the segregator lemma

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\
log^* n &= \max\{x \,:\, T(x) \leq n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\geq T(4) = 2^{16} \\
k &= \lceil \frac{\log^* n}{3} \rceil \\
&\geq 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

Lemma 15.

$$e_\ell \leq T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \leq n$

Define for $\ell \in [1:k]$

$$
\begin{aligned}
d_0 &= 1 \\
d_\ell &= 2^{\lceil \log(n/e_{k-\ell})) \rceil} \quad \text{(next power of two after } n/e_{k-\ell})
\end{aligned}
$$

$$n/(e_{k-\ell}) \leq d_\ell \leq 2n/e_{k-\ell}$$

$$e_{k-\ell} \leq 2n/d_\ell$$

def: $\log^*$

6 Proof of the segregator lemma

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2\cdots}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

6.1 Some very quickly growing sequences of numbers

Let

$$\begin{aligned} n &\geq T(4) = 2^{16} \\ k &= \lceil \frac{\log^* n}{3} \rceil \\ &\geq 2 \\ e_0 &= 1 \\ e_{\ell+1} &= k^{2+2e_\ell} \end{aligned}$$

Lemma 15.

$$e_\ell \leq T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \leq n$

Define for $\ell \in [1:k]$

$$\begin{aligned} d_0 &= 1 \\ d_\ell &= 2^{\lceil \log(n/e_{k-\ell})) \rceil} \quad \text{(next power of two after } n/e_{k-\ell}) \end{aligned}$$

$$n/(e_{k-\ell}) \leq d_\ell \leq 2n/e_{k-\ell}$$

$$n/d_\ell \leq e_{k-\ell} \quad , \quad e_{k-\ell} \leq 2n/d_\ell$$

Lemma 17.

$$\begin{aligned} d_0 &= 1 && (1) \\ d_\ell &\mid d_{\ell+1} \quad for \quad 0 \leq \ell \leq k-1 && (2) \\ d_k &\leq 2n && (3) \\ \lceil n/d_\ell \rceil \cdot d_\ell &\leq 2n \quad for \quad 0 \leq \ell \leq k-1 && (4) \\ k^{\lceil n/d_{\ell+1} \rceil} &\leq \lceil n/d_\ell \rceil / k \quad for \quad 0 \leq \ell \leq k-1 && (5) \end{aligned}$$

def: $\log^*$

6 Proof of the segregator lemma

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2\cdots}} \quad x \text{ times} \\
log^* n &= \max\{x : T(x) \le n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\ge T(4) = 2^{16} \\
k &= \lceil \frac{\log^* n}{3} \rceil \\
&\ge 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

Lemma 15.

$$e_\ell \le T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \le n$

Define for $\ell \in [1:k]$

$$
\begin{aligned}
d_0 &= 1 \\
d_\ell &= 2^{\lceil \log(n/e_{k-\ell}) \rceil} \quad \text{(next power of two after } n/e_{k-\ell})
\end{aligned}
$$

$$n/(e_{k-\ell}) \le d_\ell \le 2n/e_{k-\ell}$$

$$n/d_\ell \le e_{k-\ell} \quad , \quad e_{k-\ell} \le 2n/d_\ell$$

Lemma 17.

$$
\begin{aligned}
d_0 &= 1 && (1) \\
d_\ell &\mid d_{\ell+1} \quad \textit{for} \quad 0 \le \ell \le k-1 && (2) \\
d_k &\le 2n && (3) \\
\lceil n/d_\ell \rceil \cdot d_\ell &\le 2n \quad \textit{for} \quad 0 \le \ell \le k-1 && (4) \\
k^{\lceil n/d_{\ell+1} \rceil} &\le \lceil n/d_\ell \rceil / k \quad \textit{for} \quad 0 \le \ell \le k-1 && (5)
\end{aligned}
$$

- (1) to (3): obvious

def: $\log^*$

6 Proof of the segregator lemma

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \le n\} \end{aligned}$$

6.1 Some very quickly growing sequences of numbers

Let

$$\begin{aligned} n &\ge T(4) = 2^{16} \\ k &= \lceil \frac{\log^* n}{3} \rceil \\ &\ge 2 \\ e_0 &= 1 \\ e_{\ell+1} &= k^{2+2e_\ell} \end{aligned}$$

Lemma 15.

$$e_\ell \le T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \le n$

Define for $\ell \in [1:k]$

$$\begin{aligned} d_0 &= 1 \\ d_\ell &= 2^{\lceil \log(n/e_{k-\ell})) \rceil} \quad \text{(next power of two after } n/e_{k-\ell}) \end{aligned}$$

$$n/(e_{k-\ell}) \le d_\ell \le 2n/e_{k-\ell}$$

$$n/d_\ell \le e_{k-\ell} \quad , \quad e_{k-\ell} \le 2n/d_\ell$$

Lemma 17.

$$\begin{aligned} d_0 &= 1 && (1) \\ d_\ell &\mid d_{\ell+1} \quad \textit{for} \quad 0 \le \ell \le k-1 && (2) \\ d_k &\le 2n && (3) \\ \lceil n/d_\ell \rceil \cdot d_\ell &\le 2n \quad \textit{for} \quad 0 \le \ell \le k-1 && (4) \\ k^{\lceil n/d_{\ell+1} \rceil} &\le \lceil n/d_\ell \rceil / k \quad \textit{for} \quad 0 \le \ell \le k-1 && (5) \end{aligned}$$

- (1) to (3): obvious
- (4):

$$\begin{aligned} \lceil n/d_\ell \rceil \cdot d_\ell &\le n + d_\ell \\ &< n + d_k/2 \quad (d_\ell | d_k) \\ &\le n + 2n/2 \end{aligned}$$

def: $\log^*$

6 Proof of the segregator lemma

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\
log^* n &= \max\{x : T(x) \le n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\ge T(4) = 2^{16} \\
k &= \left\lceil \frac{\log^* n}{3} \right\rceil \\
&\ge 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

Lemma 15.

$$e_\ell \le T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \le n$

Define for $\ell \in [1:k]$

$$
\begin{aligned}
d_0 &= 1 \\
d_\ell &= 2^{\lceil \log(n/e_{k-\ell}))\rceil} \quad \text{(next power of two after } n/e_{k-\ell}) \\
n/(e_{k-\ell}) &\le d_\ell \le 2n/e_{k-\ell} \\
n/d_\ell \le e_{k-\ell} &, \quad e_{k-\ell} \le 2n/d_\ell
\end{aligned}
$$

Lemma 17.

$$
\begin{aligned}
d_0 &= 1 && (1) \\
d_\ell &\mid d_{\ell+1} \quad for \quad 0 \le \ell \le k-1 && (2) \\
d_k &\le 2n && (3) \\
\lceil n/d_\ell \rceil \cdot d_\ell &\le 2n \quad for \quad 0 \le \ell \le k-1 && (4) \\
k^{\lceil n/d_{\ell+1} \rceil} &\le \lceil n/d_\ell \rceil / k \quad for \quad 0 \le \ell \le k-1 && (5)
\end{aligned}
$$

- (5):

$$
\begin{aligned}
\frac{n}{d_{\ell+1}} &= \frac{n}{2^{\lceil \log(n/e_{k-\ell-1}) \rceil}} \\
&\le \frac{n}{n/e_{k-\ell-1}} \quad \text{omit Gauss brackets} \\
&= e_{k-\ell-1} \\
k^{\lceil n/d_{\ell+1} \rceil} &\le k^{2n/d_{\ell+1}} \\
&\le k^{2e_{k-\ell-1}+2-2} \\
&= e_{k-\ell}/k^2 \\
&\le 2n/(d_\ell k^2) \quad \text{above} \\
&\le \lceil n/d_\ell \rceil / k \quad (k \ge 2)
\end{aligned}
$$

6 Proof of the segregator lemma

def: $\log^*$

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x \,:\, T(x) \le n\} \end{aligned}$$

6.1 Some very quickly growing sequences of numbers

Let

$$\begin{aligned} n &\ge T(4) = 2^{16} \\ k &= \left\lceil \frac{\log^* n}{3} \right\rceil \\ &\ge 2 \\ e_0 &= 1 \\ e_{\ell+1} &= k^{2+2e_\ell} \end{aligned}$$

Lemma 15.

$$e_\ell \le T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \le n$

Define for $\ell \in [1:k]$

$$\begin{aligned} d_0 &= 1 \\ d_\ell &= 2^{\lceil \log(n/e_{k-\ell})) \rceil} \quad \text{(next power of two after } n/e_{k-\ell}) \end{aligned}$$

$$n/(e_{k-\ell}) \le d_\ell \le 2n/e_{k-\ell}$$

$$n/d_\ell \le e_{k-\ell} \quad , \quad e_{k-\ell} \le 2n/d_\ell$$

Lemma 17.

$$\begin{aligned} d_0 &= 1 && (1) \\ d_\ell &\mid d_{\ell+1} \quad \mathit{for} \quad 0 \le \ell \le k-1 && (2) \\ d_k &\le 2n && (3) \\ \lceil n/d_\ell \rceil \cdot d_\ell &\le 2n \quad \mathit{for} \quad 0 \le \ell \le k-1 && (4) \\ k^{\lceil n/d_{\ell+1} \rceil} &\le \lceil n/d_\ell \rceil / k \quad \mathit{for} \quad 0 \le \ell \le k-1 && (5) \end{aligned}$$

- (5):

$$\begin{aligned} \frac{n}{d_{\ell+1}} &= \frac{n}{2^{\lceil \log(n/e_{k-\ell-1}) \rceil}} \\ &\le \frac{n}{n/e_{k-\ell-1}} \\ &= e_{k-\ell-1} \\ k^{\lceil n/d_{\ell+1} \rceil} &\le k^{2n/d_{\ell+1}} \qquad n/d_{\ell+1} \ge n/e_{k-1} \ge 1 \\ &\le k^{2e_{k-\ell-1}+2-2} \\ &= e_{k-\ell}/k^2 \\ &\le 2n/(d_\ell k^2) \quad \text{above} \\ &\le \lceil n/d_\ell \rceil / k \quad (k \ge 2) \end{aligned}$$

def: $\log^*$

6 Proof of the segregator lemma

$$\begin{aligned} T(1) &= 2 \\ T(x) &= 2^{T(x-1)} \\ T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\ log^* n &= \max\{x : T(x) \leq n\} \end{aligned}$$

6.1 Some very quickly growing sequences of numbers

Let

$$\begin{aligned} n &\geq T(4) = 2^{16} \\ k &= \lceil \frac{\log^* n}{3} \rceil \\ &\geq 2 \\ e_0 &= 1 \\ e_{\ell+1} &= k^{2+2e_\ell} \end{aligned}$$

Lemma 15.

$$e_\ell \leq T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \leq n$

Define for $\ell \in [1:k]$

$$\begin{aligned} d_0 &= 1 \\ d_\ell &= 2^{\lceil \log(n/e_{k-\ell}) \rceil} \quad \text{(next power of two after } n/e_{k-\ell}) \end{aligned}$$

$$n/(e_{k-\ell}) \leq d_\ell \leq 2n/e_{k-\ell}$$

$$n/d_\ell \leq e_{k-\ell} \quad , \quad e_{k-\ell} \leq 2n/d_\ell$$

Lemma 17.

$$\begin{aligned} d_0 &= 1 && (1) \\ d_\ell &\mid d_{\ell+1} \quad for \quad 0 \leq \ell \leq k-1 && (2) \\ d_k &\leq 2n && (3) \\ \lceil n/d_\ell \rceil \cdot d_\ell &\leq 2n \quad for \quad 0 \leq \ell \leq k-1 && (4) \\ k^{\lceil n/d_{\ell+1} \rceil} &\leq \lceil n/d_\ell \rceil / k \quad for \quad 0 \leq \ell \leq k-1 && (5) \end{aligned}$$

- (5):

$$\begin{aligned} \frac{n}{d_{\ell+1}} &= \frac{n}{2^{\lceil \log(n/e_{k-\ell-1}) \rceil}} \\ &\leq \frac{n}{n/e_{k-\ell-1}} \\ &= e_{k-\ell-1} \\ k^{\lceil n/d_{\ell+1} \rceil} &\leq k^{2n/d_{\ell+1}} \\ &\leq k^{2e_{k-\ell-1}+2-2} \\ &= e_{k-\ell}/k^2 \\ &\leq 2n/(d_\ell k^2) \quad \text{above} \\ &\leq \lceil n/d_\ell \rceil / k \quad (k \geq 2) \end{aligned}$$

6.2 Partitioning of nodes and edges

from now on consider:

- TM computation graph $G = (V, E)$ for k-tape TM
- indegree $r \leq 2k + 1$
- n nodes: $V = [1 : n]$.
- in order to show segregator lemma 13 it suffices to show

Lemma 18. *We can construct a segregator for G with*

1. *at most* $15rn/\log^* n$ *nodes*
2. *whose removal leaves at most* $6rn/\log^* n$ *predecessssors for each node*

6.2 Partitioning of nodes and edges

from now on consider:

- TM computation graph $G = (V,E)$ for k-tape TM
- indegree $r \leq 2k+1$
- n nodes: $V = [1:n]$.
- in order to show segregator lemma 13 it suffices to show

Lemma 18. *We can construct a segregator for G with*

1. *at most $15rn/\log^* n$ nodes*
2. *whose removal leaves at most $6rn/\log^* n$ predecesssssors for each node*

def: partitions P_ℓ of nodes for $0 \leq \ell \leq k-1$. See figure 3.

- for P_ℓ partition $V = [1:n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\leq d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$

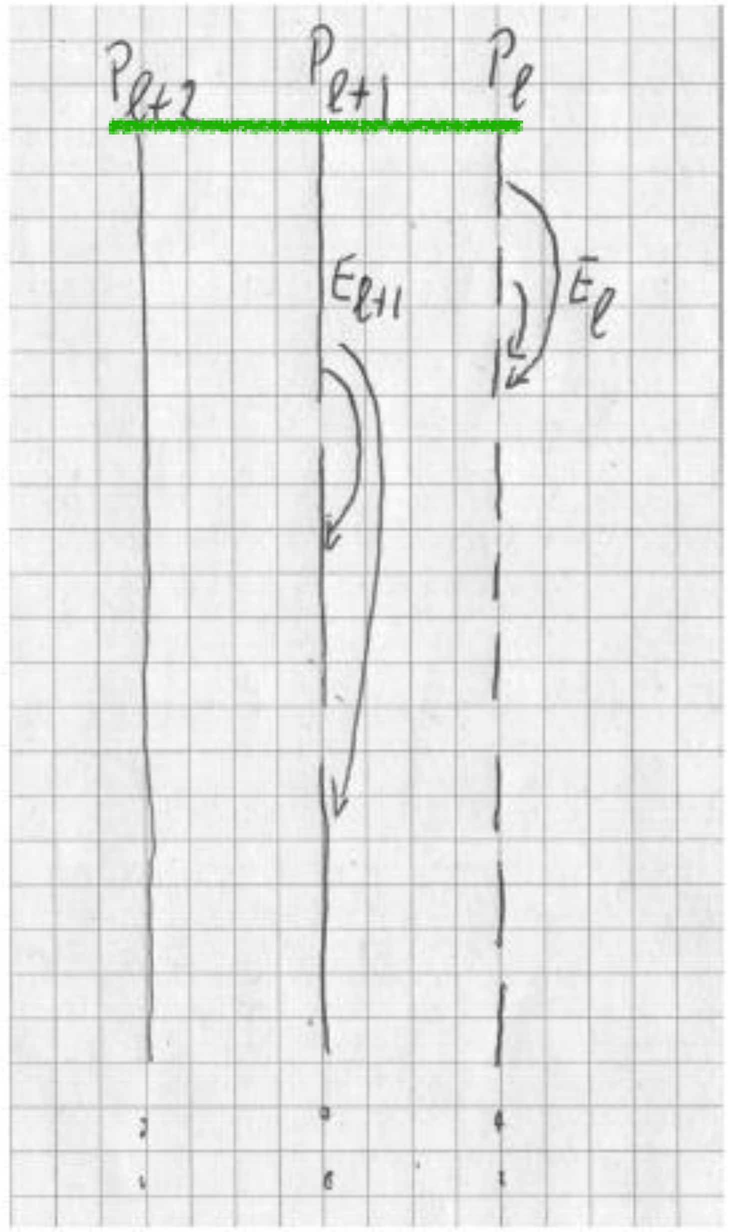


Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

6.2 Partitioning of nodes and edges

from now on consider:

- TM computation graph $G = (V, E)$ for k-tape TM
- indegree $r \leq 2k + 1$
- n nodes: $V = [1 : n]$.
- in order to show segregator lemma 13 it suffices to show

Lemma 18. *We can construct a segregator for G with*

1. *at most* $15rn/\log^* n$ *nodes*
2. *whose removal leaves at most* $6rn/\log^* n$ *predecessssors for each node*

def: partitions P_ℓ of nodes for $0 \leq \ell \leq k-1$. See figure 3.

- for P_ℓ partition $V = [1 : n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\leq d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$

partitioning of edges into sets E_ℓ

- place edge e into set E_ℓ if e crosses boundaries between blocks of P_ℓ but stays within a single block of the next coarser partition $P_{\ell+1}$

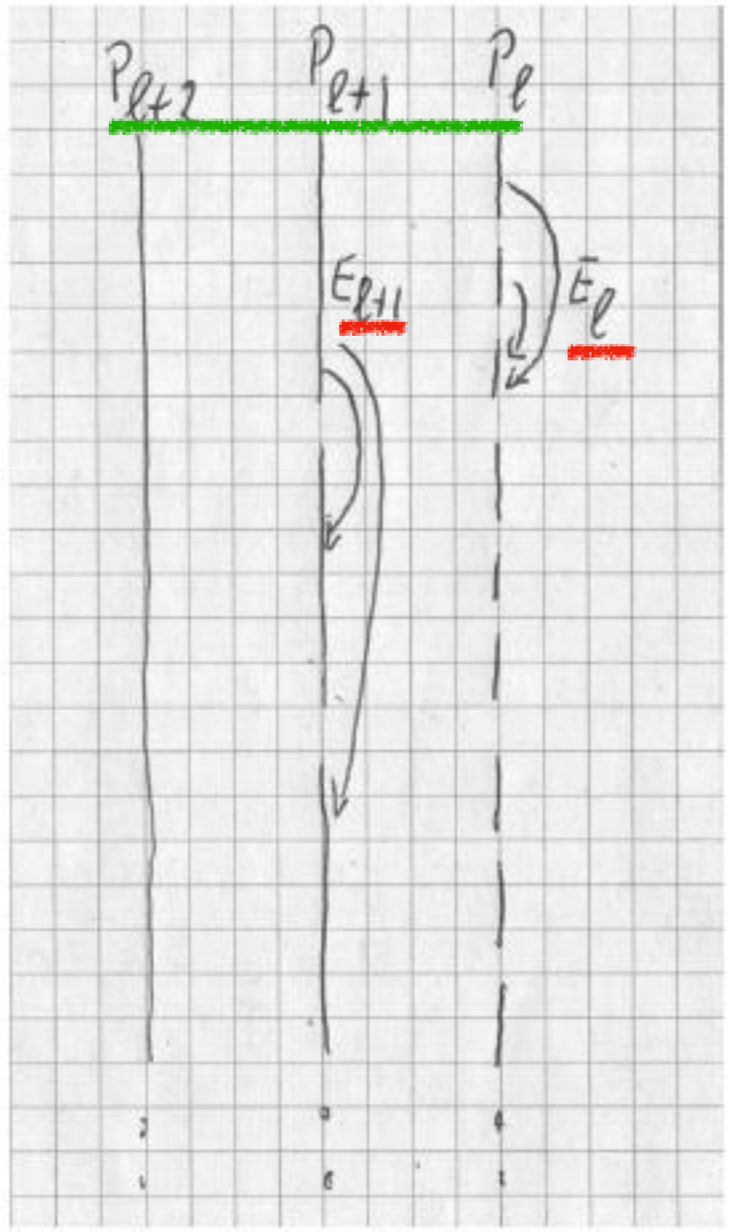


Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

6.2 Partitioning of nodes and edges

from now on consider:

- TM computation graph $G=(V,E)$ for k-tape TM
- indegree $r \leq 2k+1$
- n nodes: $V=[1:n]$.
- in order to show segregator lemma 13 it suffices to show

Lemma 18. *We can construct a segregator for G with*

1. *at most* $15rn/\log^* n$ *nodes*
2. *whose removal leaves at most* $6rn/\log^* n$ *predecessssors for each node*

def: partitions P_ℓ of nodes for $0 \leq \ell \leq k-1$. See figure 3.

- for P_ℓ partition $V=[1:n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\leq d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$

partitioning of edges into sets E_ℓ

- place edge e into set E_ℓ if e crosses boundaries between blocks of P_ℓ but stays within a single block of the next coarser partition $P_{\ell+1}$

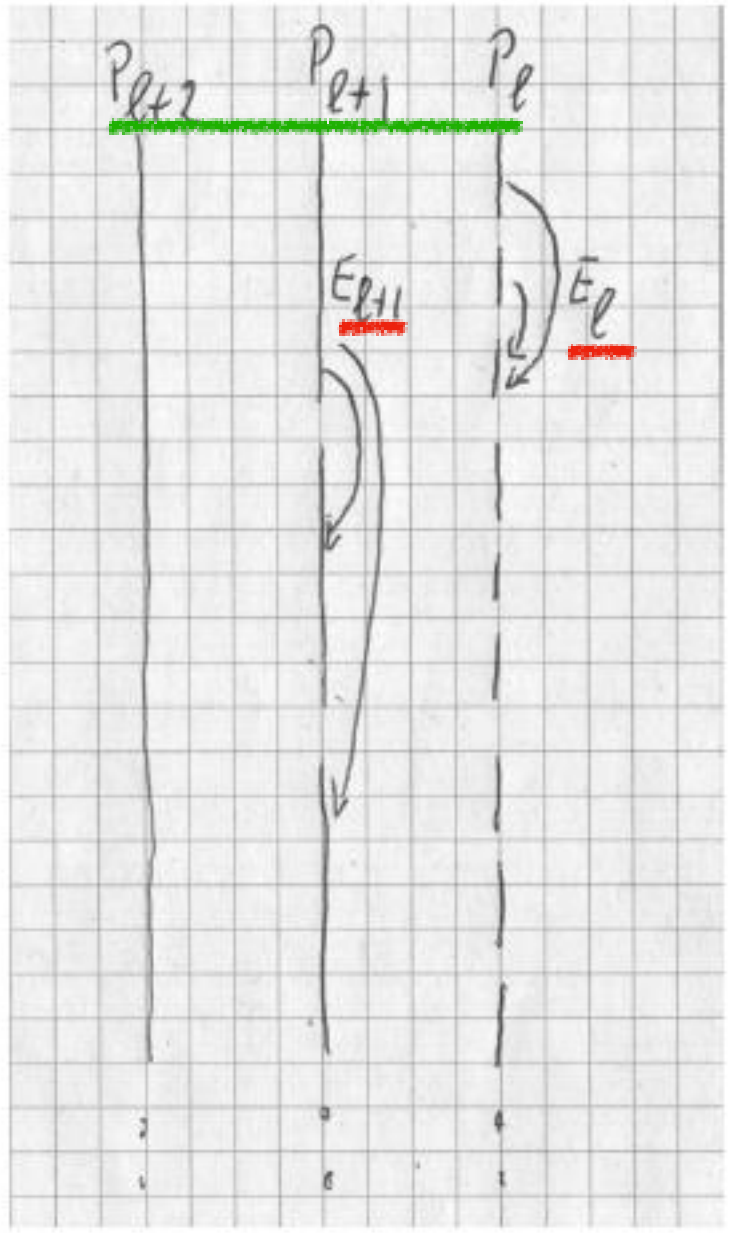


Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

- the number of edges is bounded by

$$|E| \leq r \cdot n$$

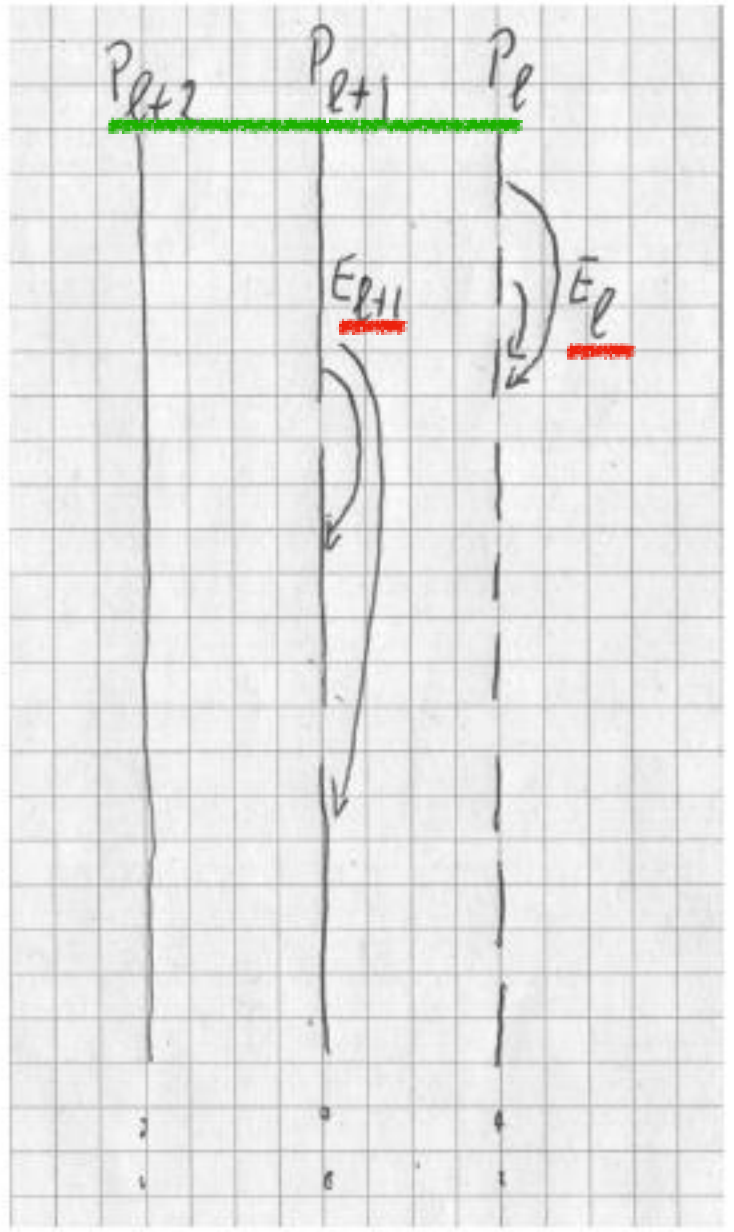
and there are k classes E_ℓ of edges

- in at least one of them the number of edges is at most

$$|E_\ell| \leq rn/k$$

def: partitions P_ℓ of nodes for $0 \le \ell \le k-1$. See figure 3.

- for P_ℓ partition $V = [1:n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\le d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$

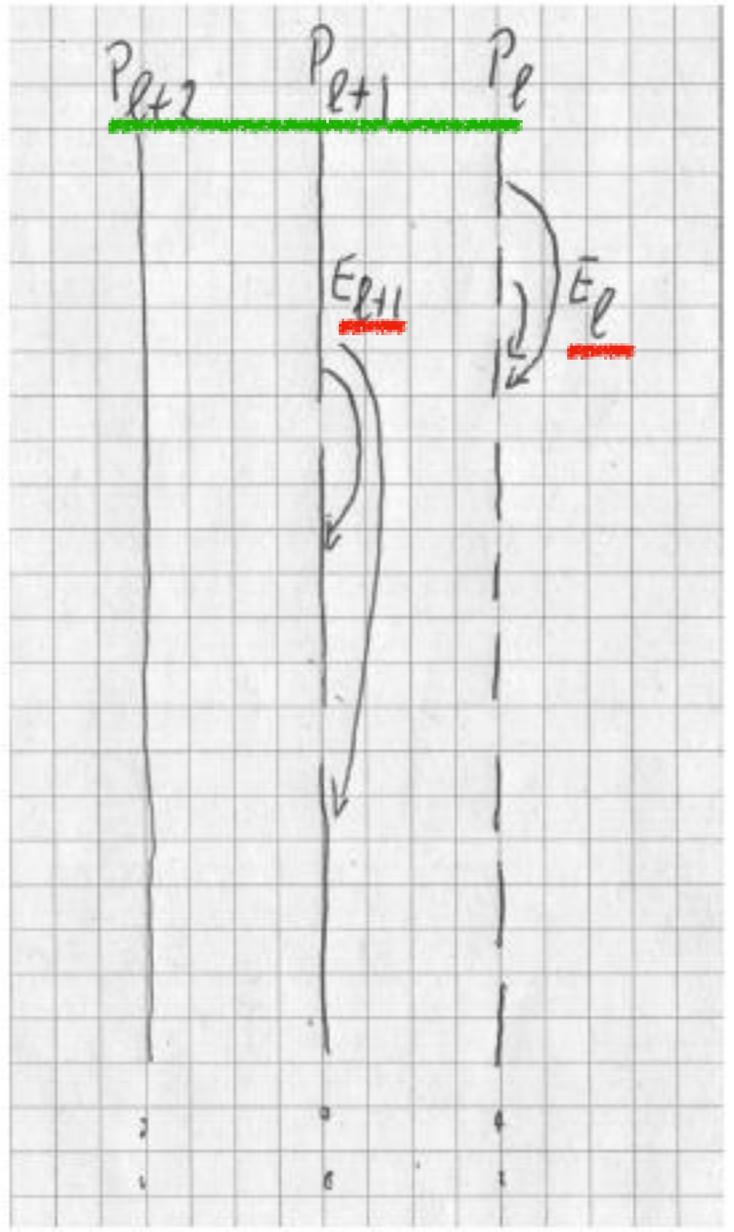


$|E_\ell| \le rn/k$

Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

def: partitions P_ℓ of nodes for $0 \le \ell \le k-1$. See figure 3.

- for P_ℓ partition $V = [1:n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\le d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$



$|E_\ell| \le rn/k$

Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

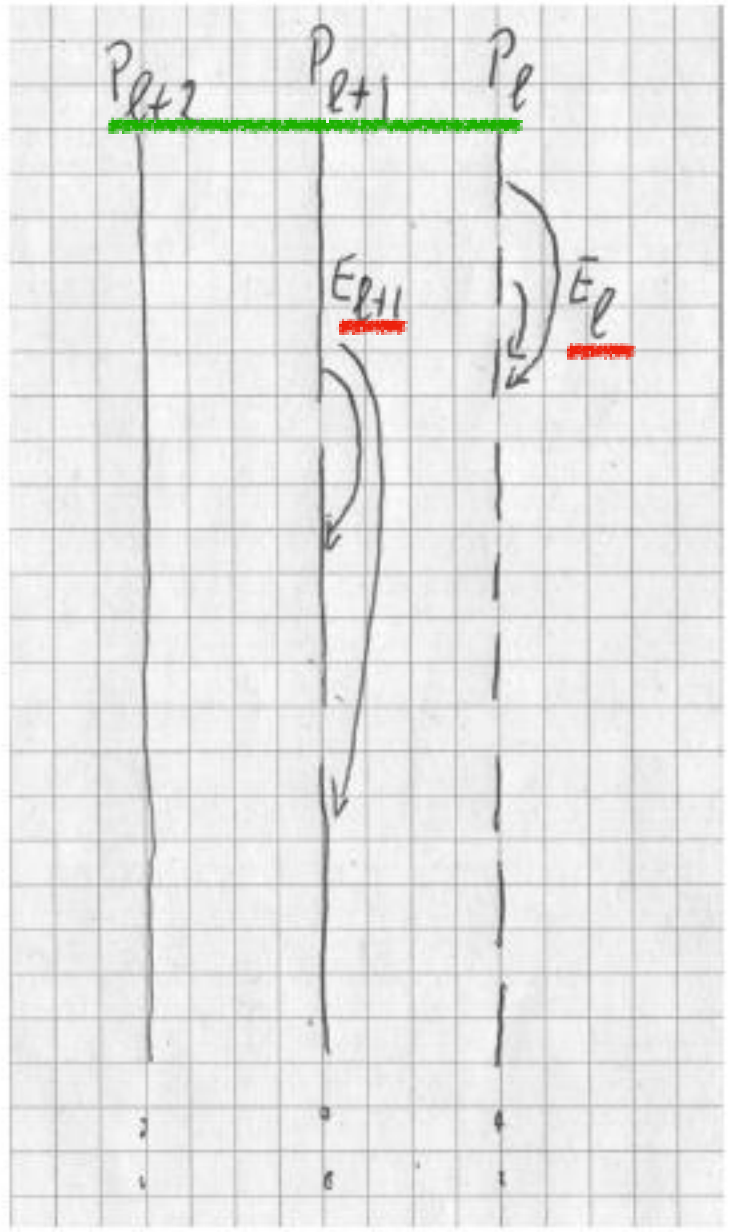
Let A be the set of start points of these edges.

$$\begin{aligned} A &= \{i : \exists j.(i,j) \in E\} \\ |A| &\le rn/k \end{aligned}$$

- removing A and adjacent edges gives graph $G - A$.

def: partitions P_ℓ of nodes for $0 \leq \ell \leq k-1$. See figure 3.

- for P_ℓ partition $V = [1:n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\leq d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$



$|E_\ell| \leq rn/k$

Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

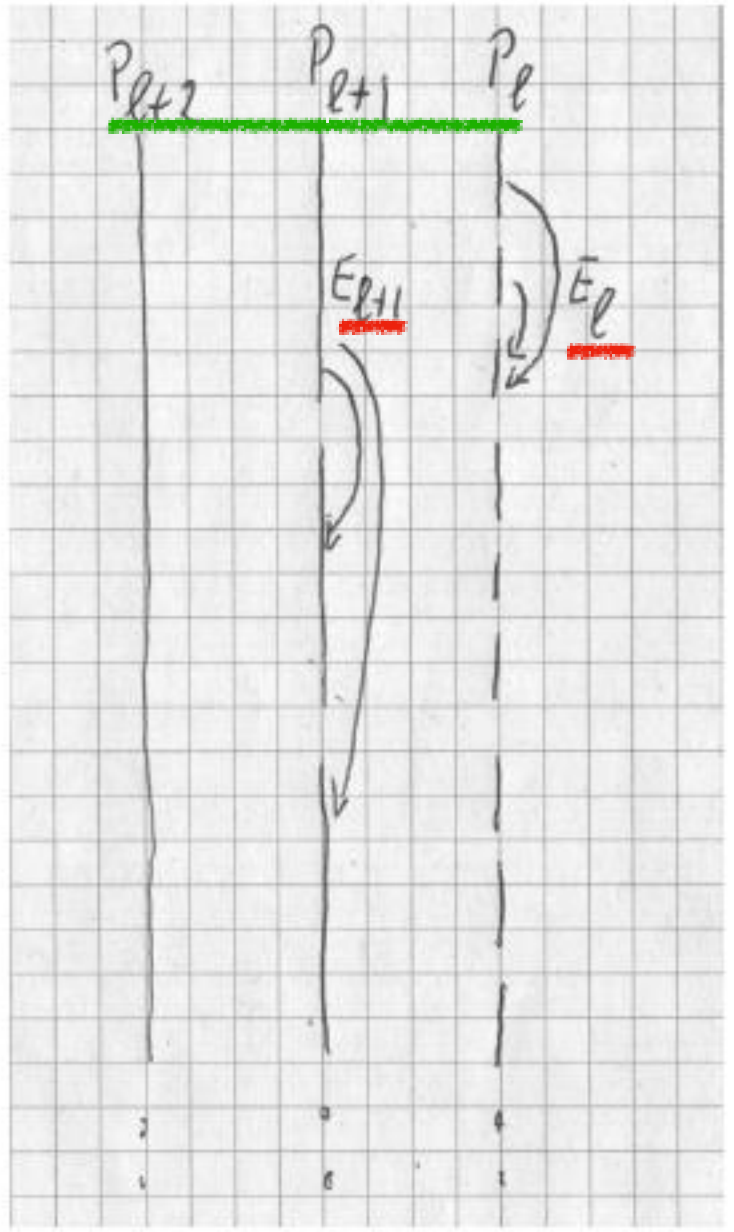
Let A be the set of start points of these edges.

$$\begin{aligned} A &= \{i : \exists j.(i,j) \in E\} \\ |A| &\leq rn/k \end{aligned}$$

- removing A and adjacent edges gives graph $G - A$.
- obtain graph $G^* = (V^*, E^*)$ by
  1. collapsing intervals in P_ℓ into nodes
     $$V^* = \{x^* : x^* \text{ is block of } P_\ell\}$$
     hence
     $$n^* = |V^*| = \lceil n/d_\ell \rceil$$
  2. for $x^* \neq y^*$ including an edge from x^* to y^* if there is an edge in $G - A$ from a node $x \in x^*$ to a node $y \in y^*$
     $$(x^*, y^*) \in E^* \leftrightarrow \exists x \in x^*, y \in y^*. \ (x,y) \in E \setminus E_\ell$$

def: partitions P_ℓ of nodes for $0 \le \ell \le k-1$. See figure 3.

- for P_ℓ partition $V = [1:n]$ into consecutive blocks of length d_ℓ nodes, except the last block, which has $\le d_\ell$ nodes.
- interval sizes very quickly growing
- P_ℓ is refinement of $P_{\ell+1}$ because $d_\ell | d_{\ell+1}$



$|E_\ell| \le rn/k$

Figure 3: Node partition P_ℓ refines $P_{\ell+1}$. Edges in E_ℓ cross block boundaries in P_ℓ but stay in the same block of $P_{\ell+1}$.

Let A be the set of start points of these edges.

$$\begin{aligned} A &= \{i : \exists j.(i,j) \in E\} \\ |A| &\le rn/k \end{aligned}$$

- removing A and adjacent edges gives graph $G - A$.
- obtain graph $G^* = (V^*, E^*)$ by
  1. collapsing intervals in P_ℓ into nodes
     $$V^* = \{x^* : x^* \text{ is block of } P_\ell\}$$
     hence
     $$n^* = |V^*| = \lceil n/d_\ell \rceil$$
  2. for $x^* \ne y^*$ including an edge from x^* to y^* if there is an edge in $G-A$ from a node $x \in x^*$ to a node $y \in y^*$
     $$(x^*, y^*) \in E^* \leftrightarrow \exists x \in x^*, y \in y^*. (x,y) \in E \setminus E_\ell$$
- edges (x,y) are either in the same block of P_ℓ or they go between different blocks of $P_{\ell+1}$ (beause edges of A are removed.
- therefore G^* is very shallow

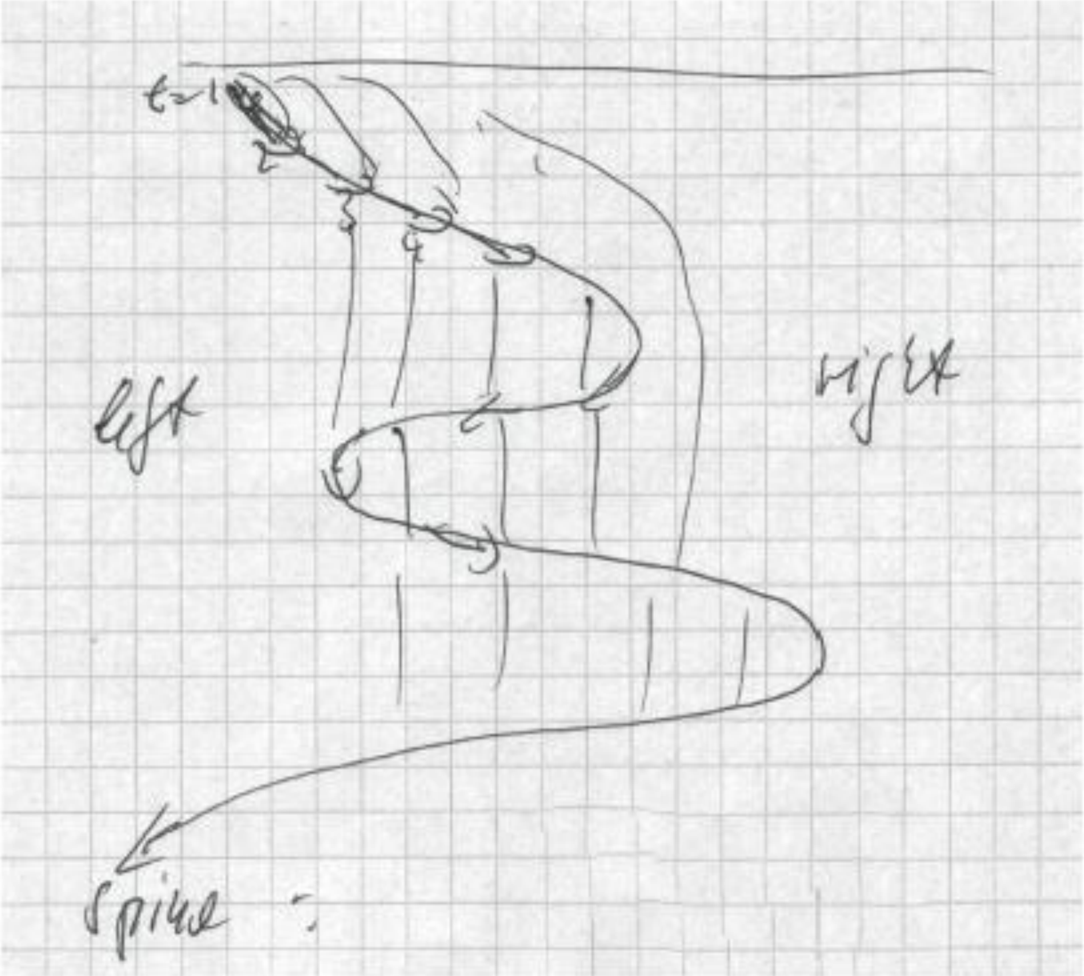
Lemma 19.

$$depth(G^*) \le \#\ blocks\ of\ P_{\ell+1} = \lceil n/d_{\ell+1} \rceil$$

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

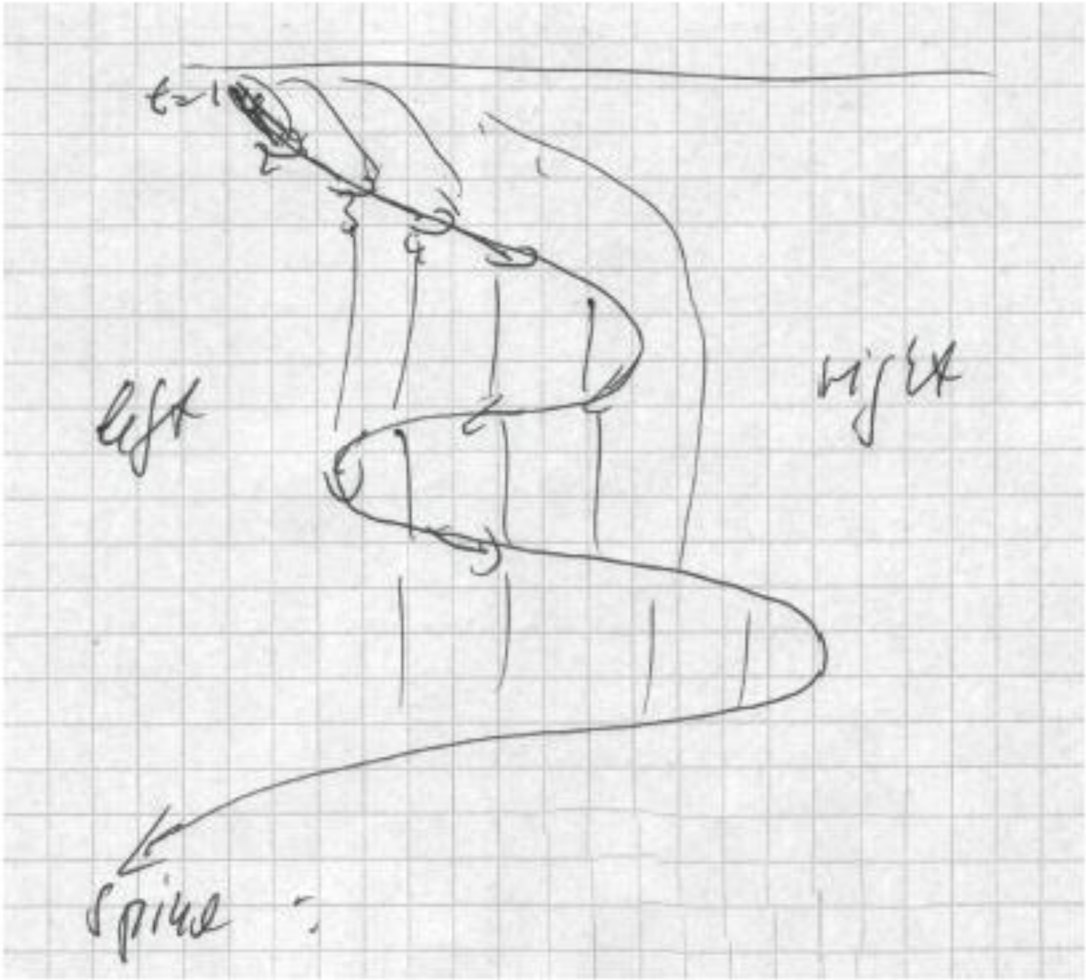
- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G^*_s

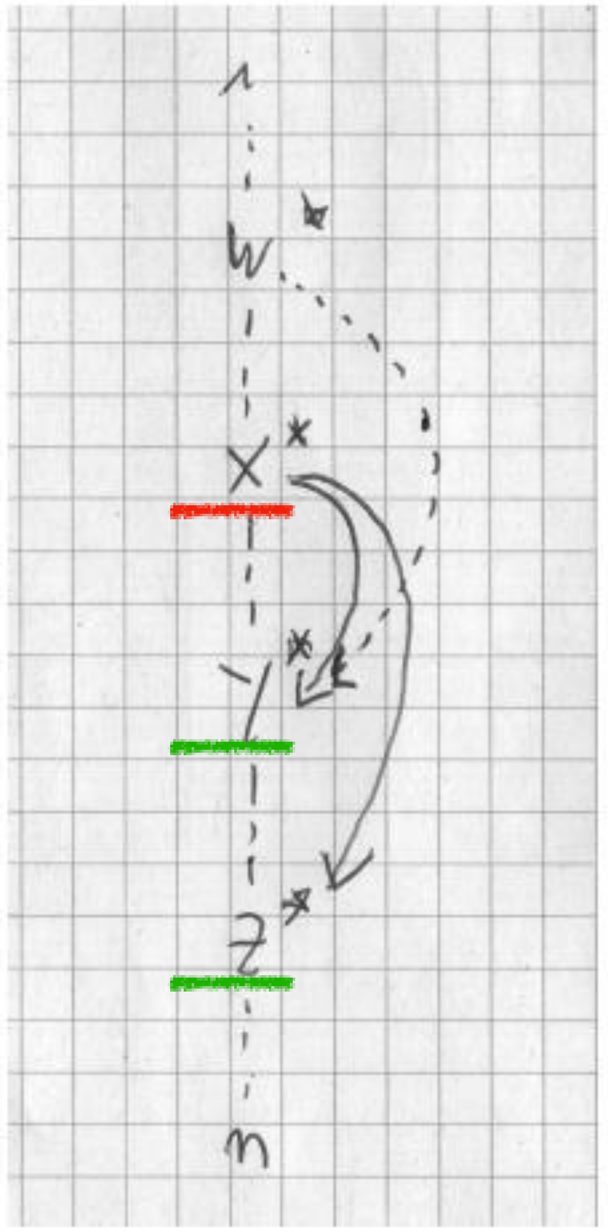


6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G_s^*



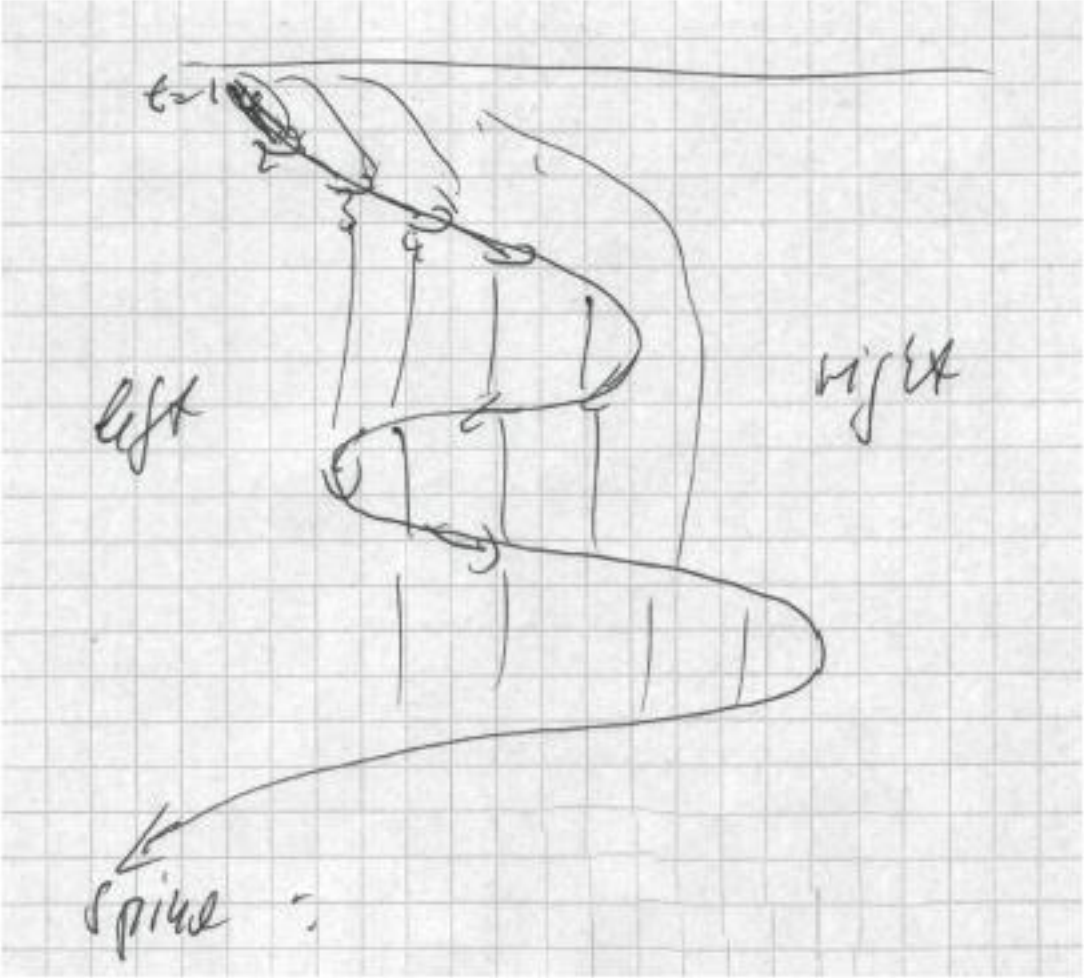


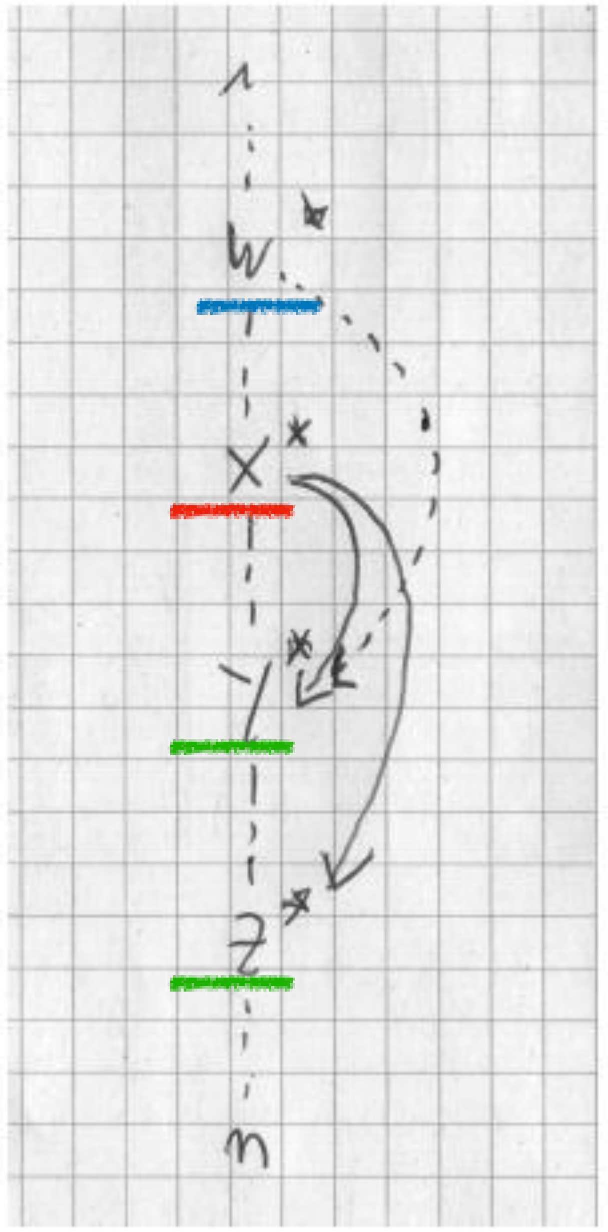
Lemma 20. *In any compresssed component graph G_i^* let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G_s^*





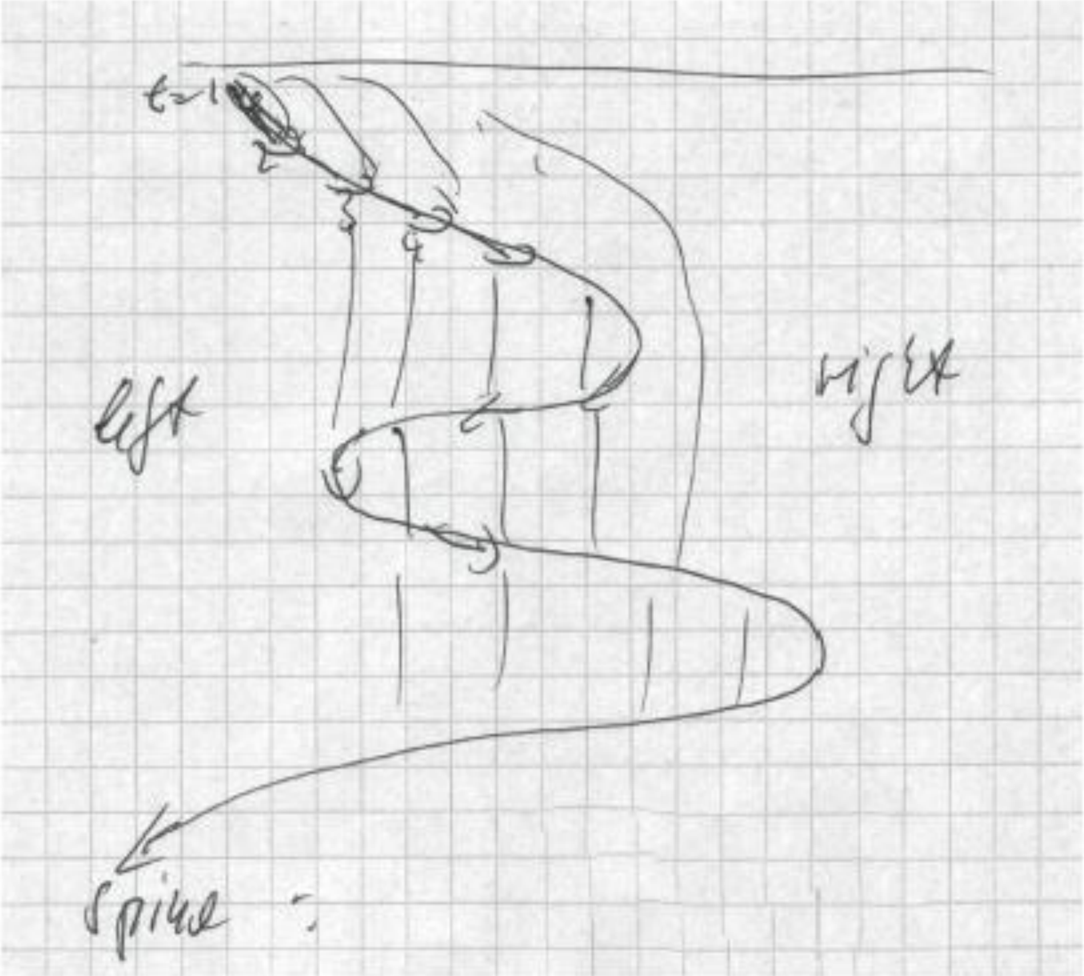
Lemma 20. *In any compresssed component graph G_i^* let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

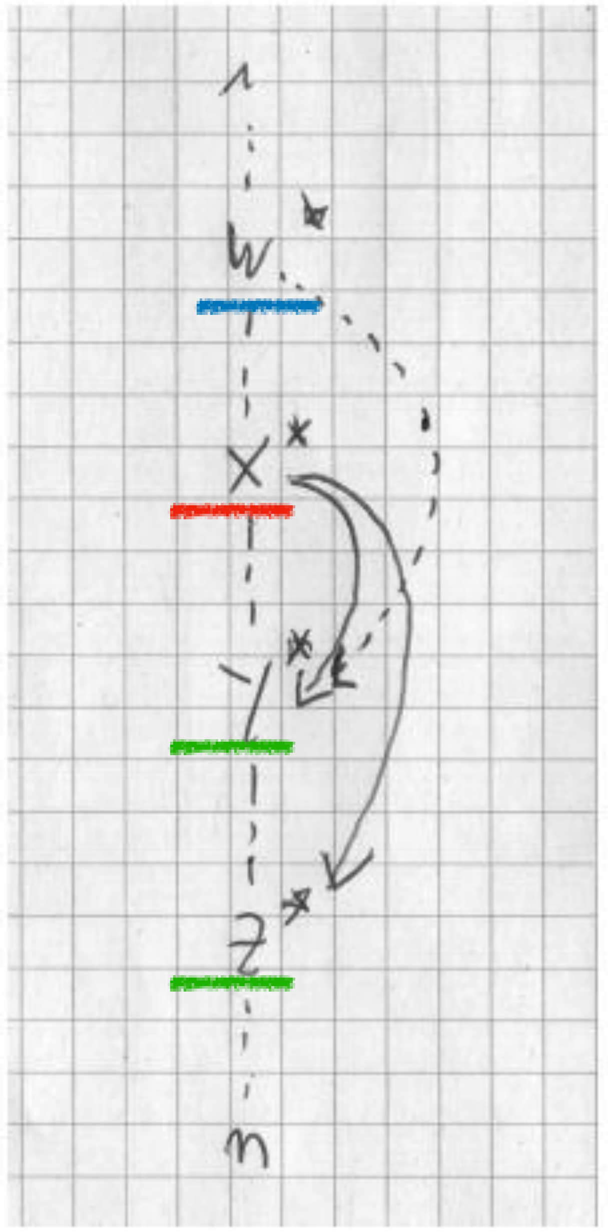
- assume there is an earlier direct predecessor w^* of y.
- then edges (x^*, z^*) and (w^*, y^*) would cross, and so would the edges from which they were constructed.
- contradicting the bracket structure of edges in G_s would be violated

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G^*_s





Lemma 20. *In any compresssed component graph G^*_i let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

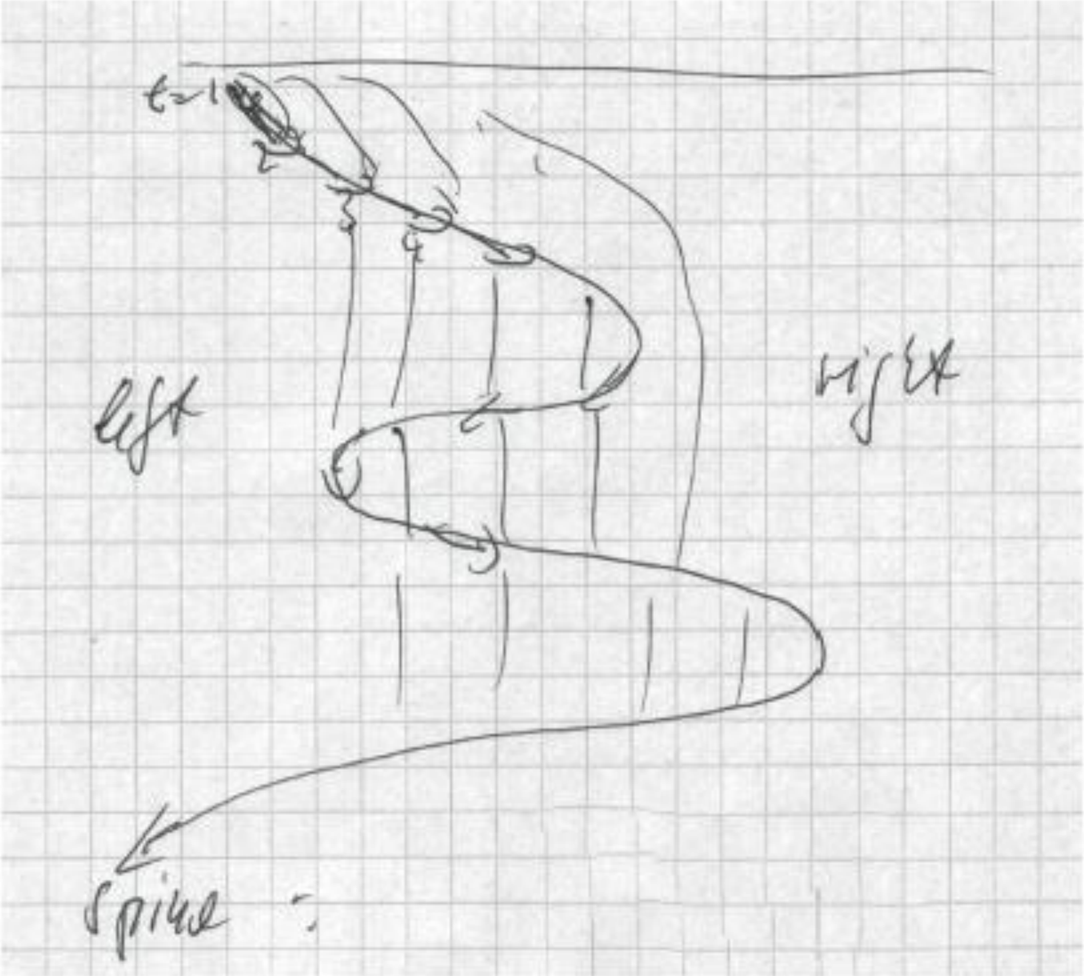
def: indegree of nodes in G^*_s

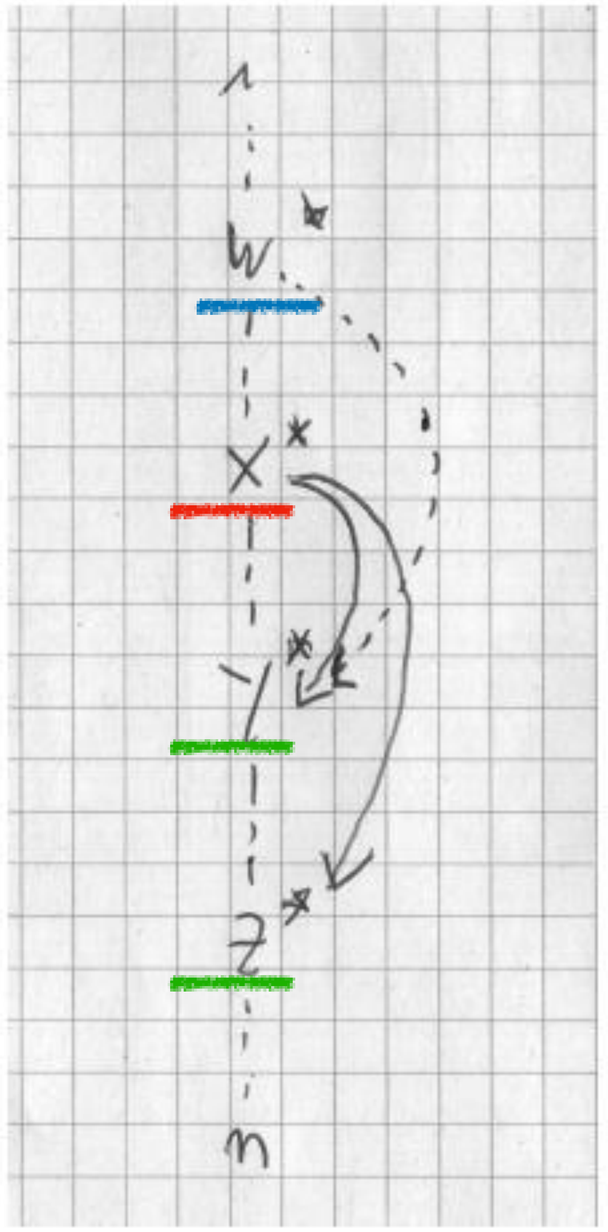
$$\begin{aligned} D_s(x^*) &= \#\text{ direct predecessors of } x^* \text{ in } G^*_s \\ D(x^*) &= \#\text{ direct predecessors of } x^* \text{ in } G^* \end{aligned}$$

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G^*_s





Lemma 20. *In any compresssed component graph G^*_i let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

def: indegree of nodes in G^*_s

$$\begin{aligned} D_s(x^*) &= \text{\# direct predecessors of } x^* \text{ in } G^*_s \\ D(x^*) &= \text{\# direct predecessors of } x^* \text{ in } G^* \end{aligned}$$

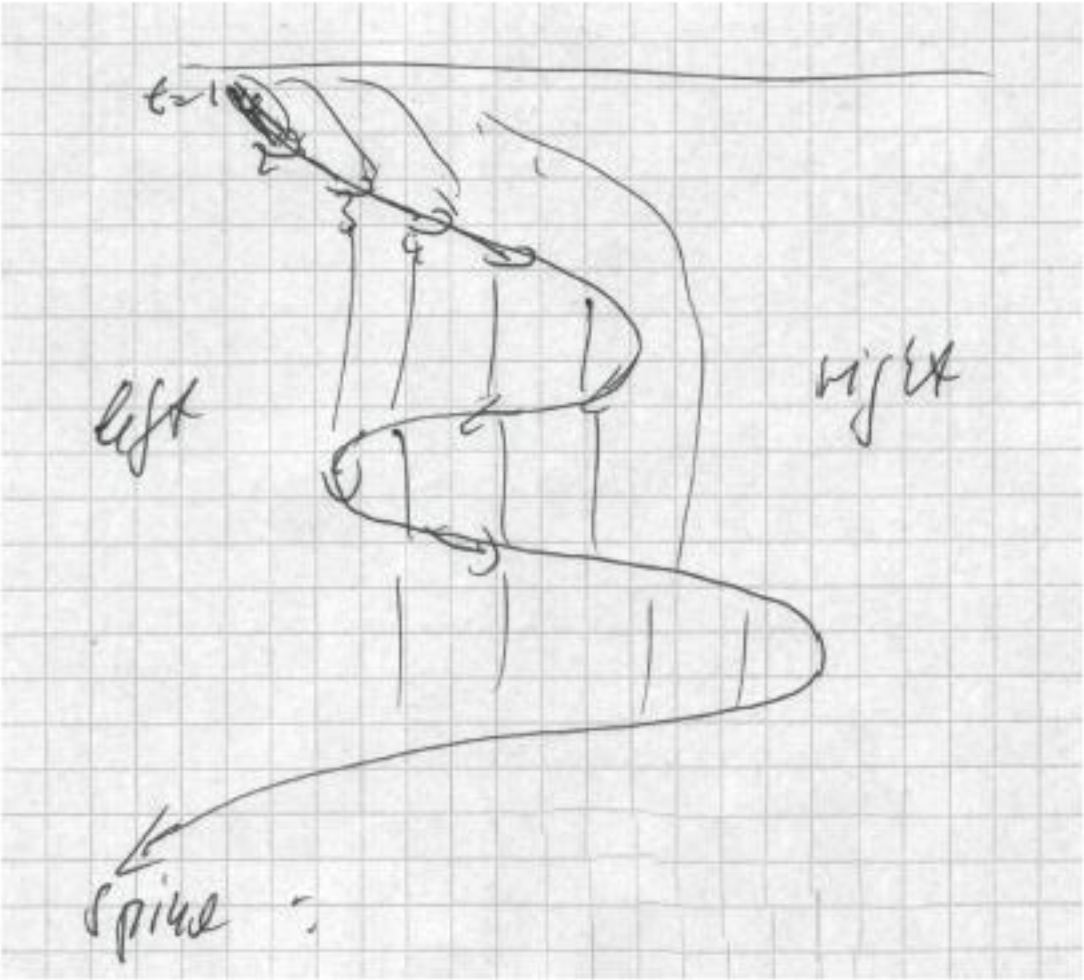
lemma 20 $\rightarrow$: different nodes have distinct direct predecessors except possibly for the first one:

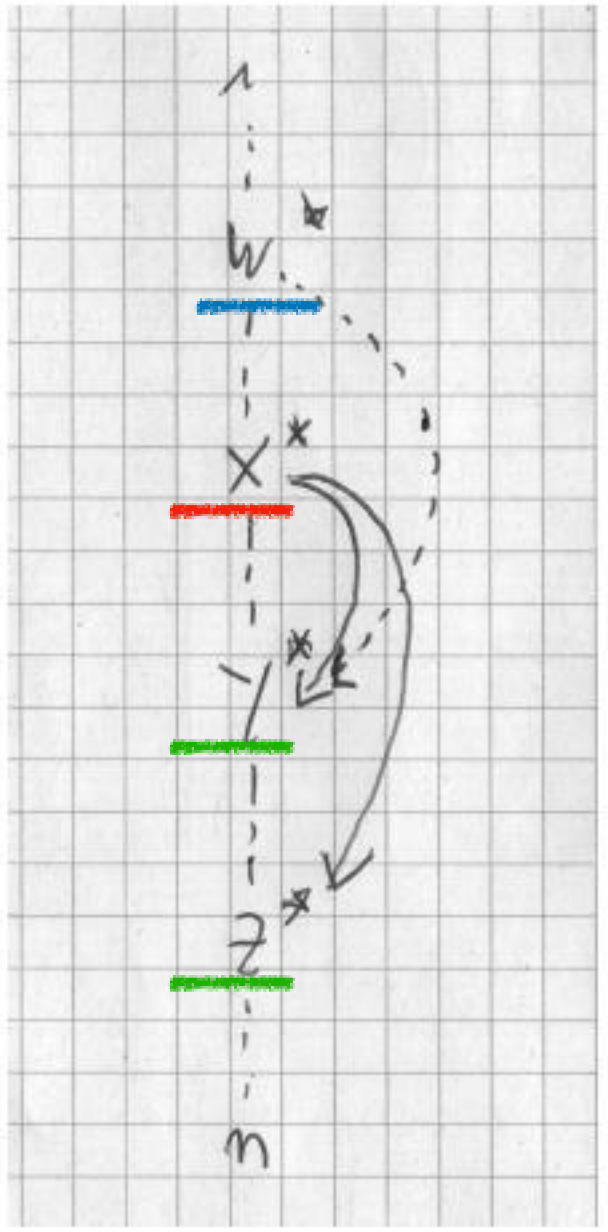
$$\begin{aligned} \sum_{x^*}(D_s(x^*)-1) &\leq n^* \\ \sum_{x^*} D_s(x^*) &\leq 2n^* \\ \sum_{x^*} D(x^*) &\leq 2rn^* \end{aligned}$$

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G_s^*





Lemma 20. *In any compresssed component graph G_i^* let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

def: indegree of nodes in G_s^*

$$\begin{aligned} D_s(x^*) &= \#\text{ direct predecessors of } x^* \text{ in } G_s^* \\ D(x^*) &= \#\text{ direct predecessors of } x^* \text{ in } G^* \end{aligned}$$

lemma 20 $\rightarrow$: different nodes have distinct direct predecessors except possibly for the first one:

$$\begin{aligned} \sum_{x^*}(D_s(x^*)-1) &\leq n^* \\ \sum_{x^*} D_s(x^*) &\leq 2n^* \\ \sum_{x^*} D(x^*) &\leq 2rn^* \end{aligned}$$

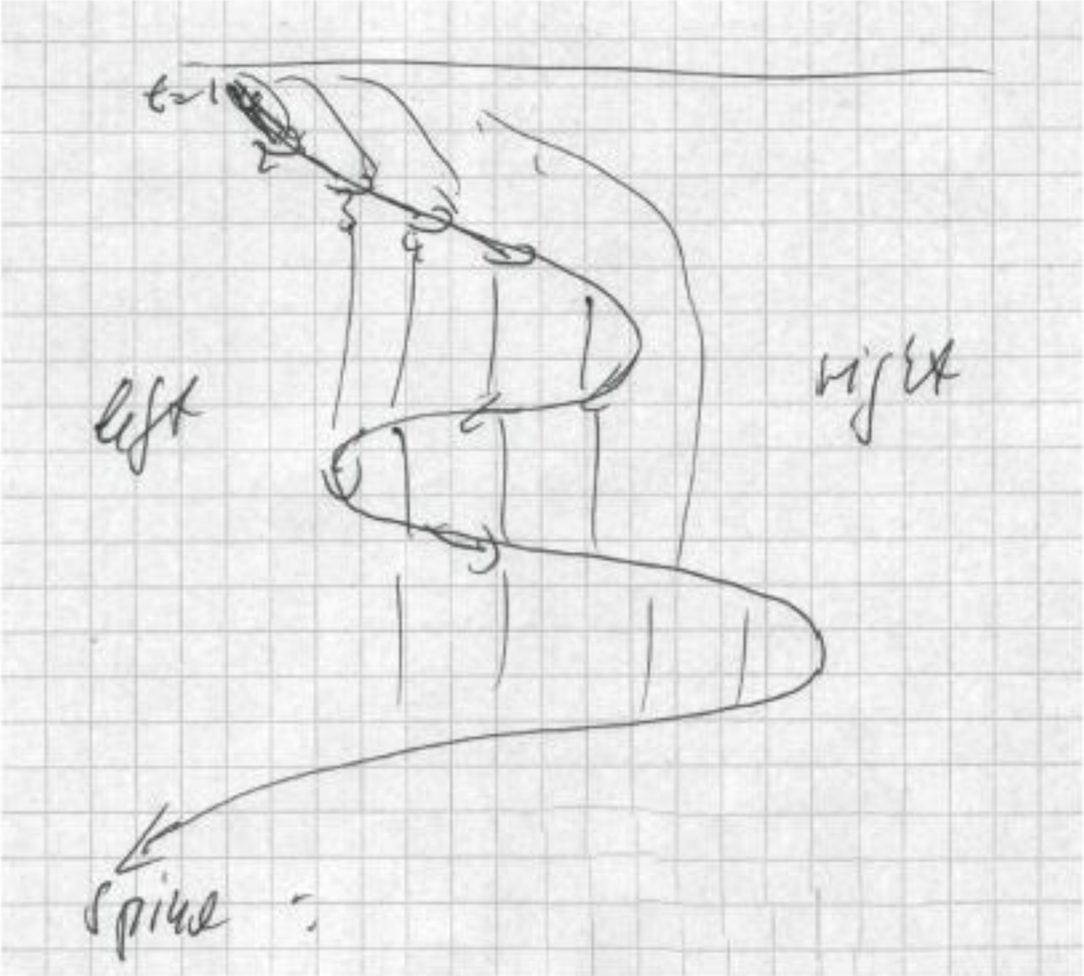
def: bad nodes

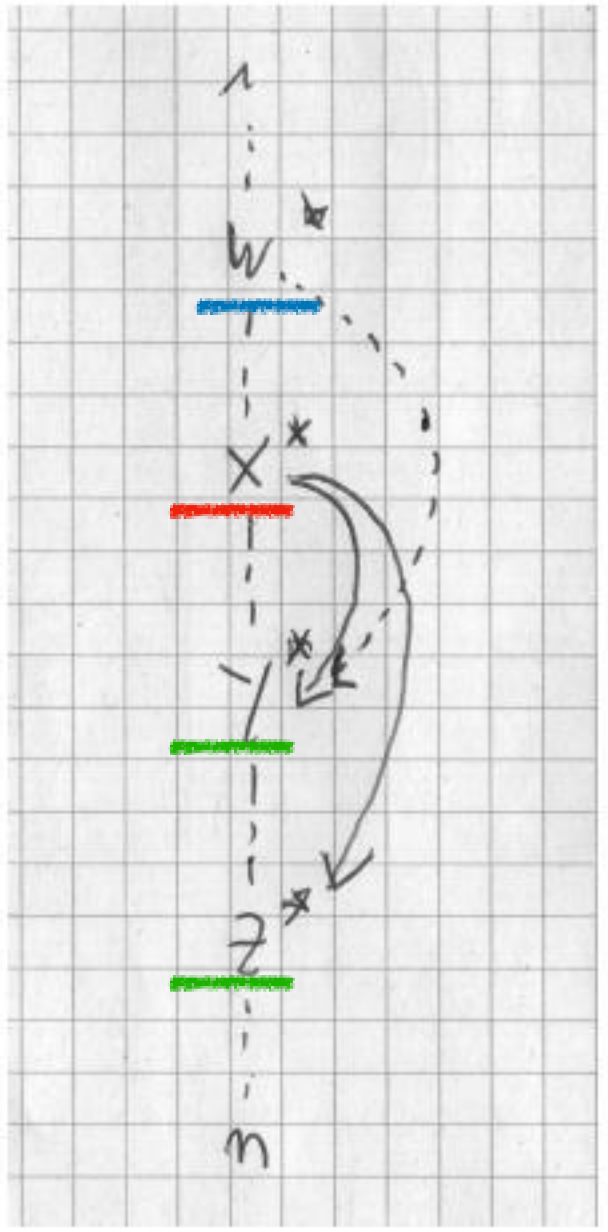
- $$x^* \text{ bad } \leftrightarrow D(x) > k$$

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G_s^*





Lemma 20. *In any compresssed component graph G_i^* let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

def: indegree of nodes in G_s^*

$$\begin{aligned}
D_s(x^*) &= \text{\# direct predecessors of } x^* \text{ in } G_s^* \\
D(x^*) &= \text{\# direct predecessors of } x^* \text{ in } G^*
\end{aligned}$$

lemma 20 $\rightarrow$: different nodes have distinct direct predecessors except possibly for the first one:

$$\begin{aligned}
\sum_{x^*}(D_s(x^*)-1) &\leq n^* \\
\sum_{x^*} D_s(x^*) &\leq 2n^* \\
\sum_{x^*} D(x^*) &\leq 2rn^*
\end{aligned}$$

def: bad nodes

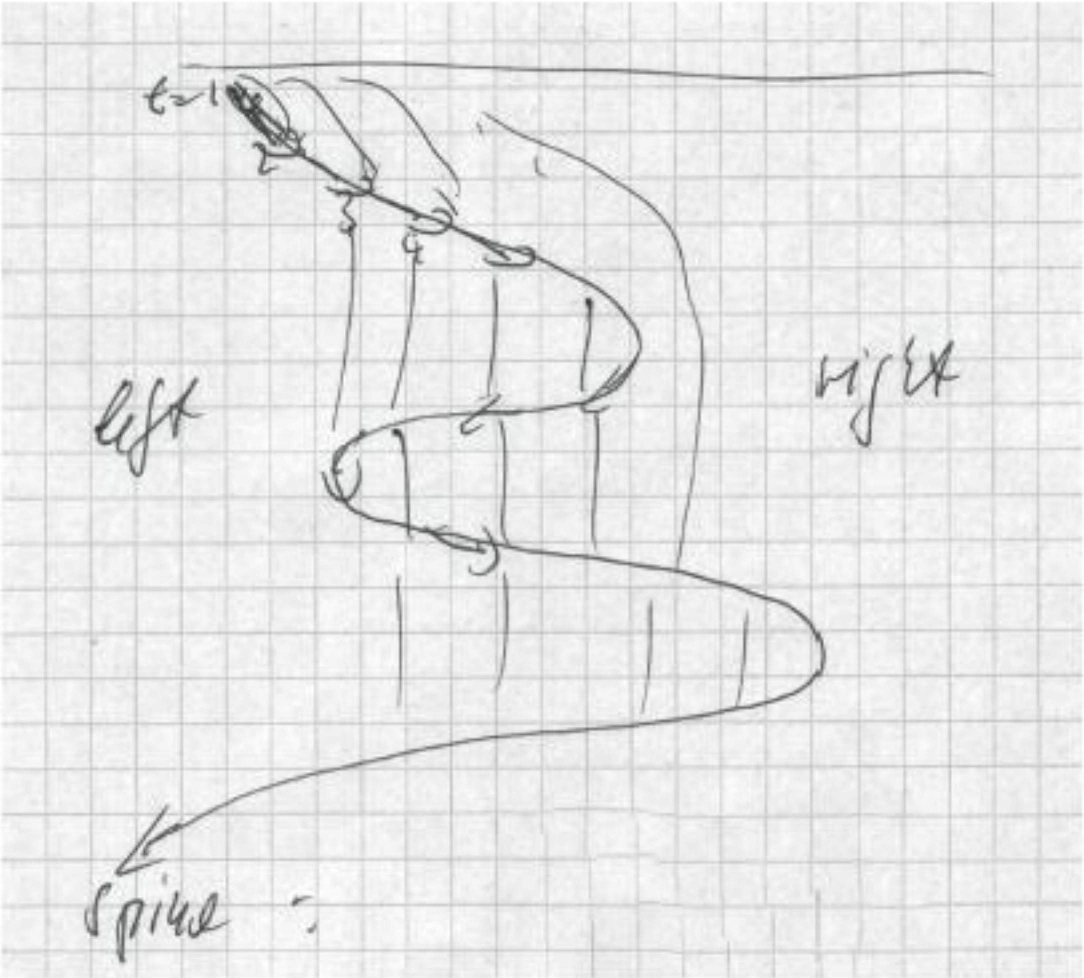
-
- $$x^* \text{ bad } \leftrightarrow D(x) > k$$

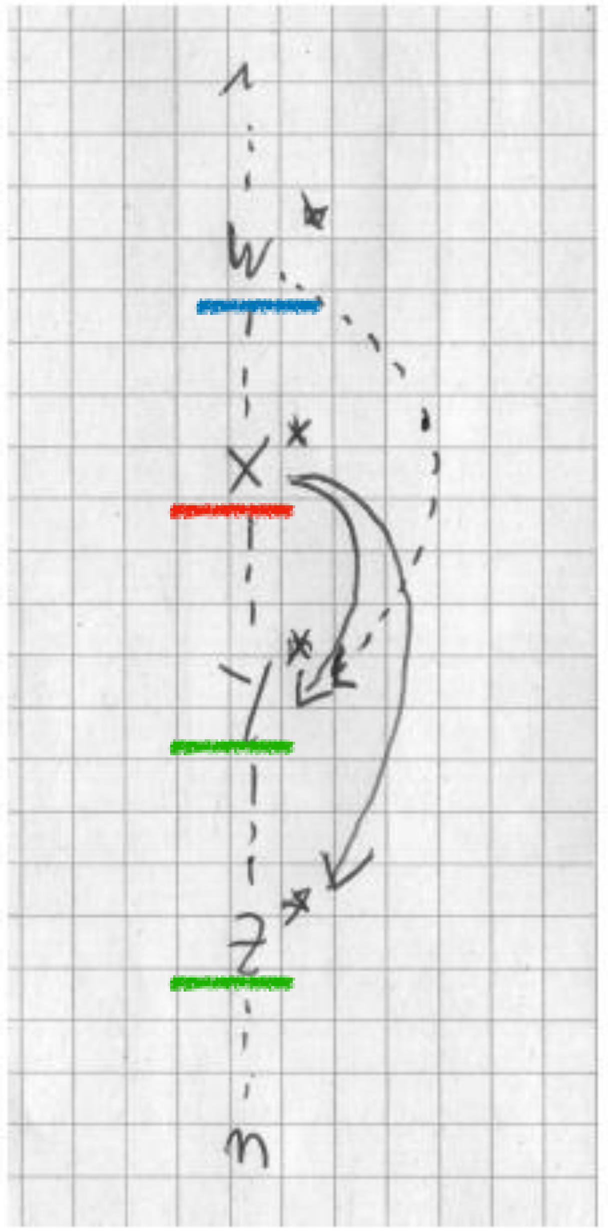
$$\begin{aligned}
B^* &= \{x^* \; : \; x^* \text{ bad}\} \\
|B^*| &\leq 2rn^*/k \\
B &= \{x \in x^* \; : \; x^* \text{ bad}\} \\
|B| &= |B^*| \cdot d_\ell \\
&\leq 2rn^* d_\ell/k \\
&\leq 2r\lceil n/d_\ell \rceil \cdot d_\ell/k \\
&\leq 4rn/k \quad \text{(lemma 17)}
\end{aligned}$$

6.3 fan in reduction

TM computation graph G is composed of planar graphs G_s which are

- the spine
- left half or right half for a tape with edges other than the spine.
- this gives composition of G^* from graphs G_s^*





Lemma 20. *In any compresssed component graph G_i^* let $y^* < z^*$ and let x^* be a direct predecessor of both x^* and y^*. Then x^* is first (earliest) direct predecessor of y^**

def: indegree of nodes in G_s^*

$$\begin{aligned} D_s(x^*) &= \#\,\text{direct predecessors of } x^* \text{ in } G_s^* \\ D(x^*) &= \#\,\text{direct predecessors of } x^* \text{ in } G^* \end{aligned}$$

lemma 20 $\rightarrow$: different nodes have distinct direct predecessors except possibly for the first one:

$$\begin{aligned} \sum_{x^*}(D_s(x^*)-1) &\leq n^* \\ \sum_{x^*} D_s(x^*) &\leq 2n^* \\ \sum_{x^*} D(x^*) &\leq 2rn^* \end{aligned}$$

def: bad nodes

-

$$x^* \text{ bad } \leftrightarrow D(x) > k$$

-

$$\begin{aligned} B^* &= \{x^* \,:\, x^* \text{ bad}\} \\ |B^*| &\leq 2rn^*/k \\ B &= \{x \in x^* \,:\, x^* \text{ bad}\} \\ |B| &= |B^*| \cdot d_\ell \\ &\leq 2rn^* d_\ell/k \\ &\leq 2r\lceil n/d_\ell\rceil \cdot d_\ell/k \\ &\leq 4rn/k \quad \text{(lemma 17)} \end{aligned}$$

6.4 Segregator

choose

$$S = A \cup B$$

size

$$\begin{aligned} |S| &\leq |A| + |B| \\ &\leq rn/k + 4rn/k \\ &= 5rn/k \\ &\leq 15rn/\log^* n \end{aligned}$$

6.4 Segregator

choose

$$S = A \cup B$$

size

$$\begin{aligned} |S| &\leq |A| + |B| \\ &\leq rn/k + 4rn/k \\ &= 5rn/k \\ &\leq 15rn/\log^* n \end{aligned}$$

number of predecessors

- let $U = G^* - B^*$
- $pred(x^*)$: number of predecessors of x^* in U

$$\begin{aligned} pred(x^*) &\leq D(x^*)^{depth(G^*)} \\ &\leq k^{depth(G^*)} \quad (x^* \text{ good}) \\ &\leq k^{\lceil n/d_{\ell+1} \rceil} \quad (\text{lemma 19}) \\ &\leq \lceil n/d_\ell \rceil / k \quad (\text{lemma 17 }) \end{aligned}$$

6.4 Segregator

choose

$$S = A \cup B$$

size

$$\begin{aligned} |S| &\le |A| + |B| \\ &\le rn/k + 4rn/k \\ &= 5rn/k \\ &\le 15rn/\log^* n \end{aligned}$$

number of predecessors

- let $U = G^* - B^*$
- $pred(x^*)$: number of predecessors of x^* in U

$$\begin{aligned} pred(x^*) &\le D(x^*)^{depth(G^*)} \\ &\le k^{depth(G^*)} \quad (x^* \text{ good}) \\ &\le k^{\lceil n/d_{\ell+1} \rceil} \quad (\text{lemma } 19) \\ &\le \lceil n/d_\ell \rceil / k \quad (\text{lemma } 17\ ) \end{aligned}$$

- $pred(x)$: number of predecessors of x in $G - (A \cup B)$

$$\begin{aligned} pred(x) &\le d_\ell \cdot pred(x^*) \\ &\le d_\ell \cdot \lceil n/d_\ell \rceil / k \\ &\le 2n/k \quad (\text{lemma } 17) \\ &\le 6rn/\log^* n \end{aligned}$$

def: $\log^*$

6 Proof of the segregator lemma

$$
\begin{aligned}
T(1) &= 2 \\
T(x) &= 2^{T(x-1)} \\
T(x) &= 2^{2^{2^{\cdots}}} \quad x \text{ times} \\
log^* n &= \max\{x : T(x) \le n\}
\end{aligned}
$$

6.1 Some very quickly growing sequences of numbers

Let

$$
\begin{aligned}
n &\ge T(4) = 2^{16} \\
k &= \lceil \frac{\log^* n}{3} \rceil \\
&\ge 2 \\
e_0 &= 1 \\
e_{\ell+1} &= k^{2+2e_\ell}
\end{aligned}
$$

Lemma 15.

$$e_\ell \le T(k+2\ell)$$

Lemma 16. *For sequence*

$$1 = e_0 < e_1 < \ldots < e_{k-1}$$

holds: $e_{k-1} \le n$

Define for $\ell \in [1:k]$

$$
\begin{aligned}
d_0 &= 1 \\
d_\ell &= 2^{\lceil \log(n/e_{k-\ell}) \rceil} \quad \text{(next power of two after } n/e_{k-\ell}\text{)}
\end{aligned}
$$

$$n/(e_{k-\ell}) \le d_\ell \le 2n/e_{k-\ell}$$

$$e_{k-\ell} \le 2n/d_\ell$$

Lemma 17.

$$
\begin{aligned}
d_0 &= 1 && (1) \\
d_\ell &\mid d_{\ell+1} \quad for \quad 0 \le \ell \le k-1 && (2) \\
d_k &\le 2n && (3) \\
\lceil n/d_\ell \rceil \cdot d_\ell &\le 2n \quad for \quad 0 \le \ell \le k-1 && (4) \\
k^{\lceil n/d_{\ell+1} \rceil} &\le \lceil n/d_\ell \rceil / k \quad for \quad 0 \le \ell \le k-1 && (5)
\end{aligned}
$$

- (5):

$$
\begin{aligned}
\frac{n}{d_{\ell+1}} &= \frac{n}{2^{\lceil \log(n/e_{k-\ell-1}) \rceil}} \\
&\le \frac{n}{n/e_{k-\ell-1}} \\
&= e_{k-\ell-1} \\
k^{\lceil n/d_{\ell+1} \rceil} &\le k^{2e_{k-\ell-1}+2-2} \quad \text{and (4)} \\
&= e_{k-\ell}/k^2 \\
&\le 2n/(d_\ell k^2) \quad \text{above} \\
&\le \lceil n/d_\ell \rceil / k \quad (k \ge 2)
\end{aligned}
$$