Word Equations in Nondeterministic Linear Space

Artur Jeż

Stuttgart, 5.04.2017

Word Equations

Definition

Given equation $U = V$, where $U, V \in (\Sigma \cup \mathcal{X})^*$;
Is there a substitution $S : \mathcal{X} \to \Sigma^*$ satisfying the equation?

Word Equations

Definition

Given equation $U = V$, where $U, V \in (\Sigma \cup \mathcal{X})^*$;
Is there a substitution $S : \mathcal{X} \to \Sigma^*$ satisfying the equation?

$aXbXYbbb = XabaabYbY \ (S(X) = aa, S(Y) = bb)$

Word Equations

Definition

Given equation $U = V$, where $U, V \in (\Sigma \cup \mathcal{X})^*$;
Is there a substitution $S : \mathcal{X} \to \Sigma^*$ satisfying the equation?

$aaabaabbbbb = aaabaabbbbbb \; (S(X) = aa, S(Y) = bb)$

Word Equations

Definition

Given equation $U = V$, where $U, V \in (\Sigma \cup \mathcal{X})^*$;
Is there a substitution $S : \mathcal{X} \to \Sigma^*$ satisfying the equation?

$aXbXYbbb = XabaabYbY$ $(S(X) = aa, S(Y) = bb)$

Extend S to a homomorphism $(\Sigma \cup \mathcal{X})^* \to \Sigma^*$, an identity on Σ.
Solution word: $S(U)$

Word Equations

Definition

Given equation $U = V$, where $U, V \in (\Sigma \cup \mathcal{X})^*$;
Is there a substitution $S : \mathcal{X} \to \Sigma^*$ satisfying the equation?

$aXbXYbbb = XabaabYbY \; (S(X) = aa, S(Y) = bb)$

Extend S to a homomorphism $(\Sigma \cup \mathcal{X})^* \to \Sigma^*$, an identity on Σ.
Solution word: $S(U)$

Known algorithms

Makanin 77 3NEXPTIME → EXPSPACE [Gutierrez 98]

Plandowski 99 PSPACE

J. 13 PSPACE

Main idea

- Recompression algorithm [J. 2013]
- Huffman coding of letters

Main idea

- Recompression algorithm [J. 2013]
- Huffman coding of letters

The proof is more complex

- how letters depend on fragments of original equation
- special coding — so worse than Huffman — but only for proof
- handle several possible problems:
  - many letters
  - many unique letters
  - and many other (perhaps artefacts of the proof)

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint)
  replace each $ab \in \Sigma_\ell \Sigma_r$ in w with a fresh c_{ab}

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression $(\Sigma_\ell, \Sigma_r$ are disjoint)
  replace each $ab \in \Sigma_\ell \Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell\Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell\Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

$$aaabbcccbbcccbbb$$

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell\Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

$$aaabbcccbbcccbbb$$
$$aaab_2\ c_3\ b_2\ c_3\ b_3$$

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell \Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

$$aaabbcccbbcccbbb$$
$$aaab_2\ c_3\ b_2\ c_3\ b_3$$

$\{a, c\}, \{b\}$ pair compression

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell \Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

$$aaabbcccbbcccbbb$$
$$aaab_2\ c_3\ b_2\ c_3\ b_3$$

$\{a, c\}, \{b\}$ pair compression

$$aaabbcccbbcccbbb$$

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell \Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

$aaabbcccbbcccbbb$

$aaa\,b_2\ c_3\ b_2\ c_3\ \ b_3$

$\{a, c\}, \{b\}$ pair compression

$aaabbcccbbcccbbb$

$aa\ d\ bcc\ e\ bcc\ e\ bb$

Compression operations

Compression operations

Given a word w:

- (Σ_ℓ, Σ_r) pair compression (Σ_ℓ, Σ_r are disjoint) replace each $ab \in \Sigma_\ell \Sigma_r$ in w with a fresh c_{ab}
- Σ block compression replace each maximal block $a^\ell \in \Sigma^*$ in w by a fresh a_ℓ. (maximal block: a^ℓ that cannot be extended).

$\{b, c\}$ block compression

$$aaabbcccbbcccbbb$$
$$aaa\,b_2\,c_3\,b_2\,c_3\,b_3$$

$\{a, c\}, \{b\}$ pair compression

$$aaabbcccbbcccbbb$$
$$aa\,d\,bcc\,e\,bcc\,e\,bb$$

- We want to perform it on $S(U)$ and $S(V)$.
- Occurrence can be partially in the equation and in the variable.

Preliminaries: explicit word

Checking equality of two explicit words

Require: two words u, v to be tested for equality

```
while |u| > 1 or |v| > 1 do
    Σ ← letters in u, v
    perform Σ-block compression
    while some pair in Σ^2 was not considered do
        guess partition of Σ to (Σ_ℓ, Σ_r)
        perform (Σ_ℓ, Σ_r) pair compression
test equality
```

Preliminaries: explicit word

Checking equality of two explicit words

Require: two words u, v to be tested for equality

```
while |u| > 1 or |v| > 1 do
    Σ ← letters in u, v
    perform Σ-block compression
    while some pair in Σ² was not considered do
        guess partition of Σ to (Σ_ℓ, Σ_r)
        perform (Σ_ℓ, Σ_r) pair compression
test equality
```

Phase: one iteration of the main loop.

Preliminaries: explicit word

Checking equality of two explicit words

Require: two words u, v to be tested for equality

```
while |u| > 1 or |v| > 1 do
    Σ ← letters in u, v
    perform Σ-block compression
    while some pair in Σ² was not considered do
        guess partition of Σ to (Σ_ℓ, Σ_r)
        perform (Σ_ℓ, Σ_r) pair compression
test equality
```

Phase: one iteration of the main loop.

Shortening

Consider consecutive ab in u, v at the beginning of the phase

$a = b$ compressed as a block

$a \neq b$ considered and compressed, or
one of them was compressed earlier

Pair Compression on word equation

In a solution word $S(U)$ or $S(V)$:

- pair is from the equation: OK, we replace it

Pair Compression on word equation

In a solution word $S(U)$ or $S(V)$:

- pair is from the equation: OK, we replace it
- it is from the substitution for a variable: OK, solution changes

Pair Compression on word equation

In a solution word $S(U)$ or $S(V)$:

- pair is from the equation: OK, we replace it
- it is from the substitution for a variable: OK, solution changes
- partially here and there: just pop the problematic letter out

Pair Compression on word equation

In a solution word $S(U)$ or $S(V)$:

- pair is from the equation: OK, we replace it
- it is from the substitution for a variable: OK, solution changes
- partially here and there: just pop the problematic letter out

PairCompression

for $X \in \mathcal{X}$ **do**
  let b: first letter of $S(X)$ ▷ Guess
  if $b \in \Sigma_r$ **then**
    replace each occurrence of X by bX ▷ Pop
  if $S(X) = \epsilon$ **then** ▷ Guess
    remove X from the equation
  let a: last ... ▷ symmetrically for the last letter and Σ_ℓ
perform pair compression on sides of the equation

Block Compression

BlockCompression

1: **for** $X \in \mathcal{X}$ **do**
2: let $S(X) = a^\ell w b^r$ ▷ Guess
3: replace X with $a^\ell X b^r$
4: **if** $S(X) = \epsilon$ **then** ▷ Guess
5: remove X from the equation
6: perform block compression on sides of the equation

The algorithm

Main algorithm

while sides of the equation are nontrivial **do**
  $\Sigma \leftarrow$ letters in the equation
  perform Σ-block compression
  while some pair in Σ^2 was not considered **do**
    guess partition of Σ to (Σ_ℓ, Σ_r) ▷ Important
    perform (Σ_ℓ, Σ_r) pair compression

The algorithm

Main algorithm

while sides of the equation are nontrivial **do**
    $\Sigma \leftarrow$ letters in the equation
    perform Σ-block compression
    while some pair in Σ^2 was not considered **do**
        guess partition of Σ to (Σ_ℓ, Σ_r) ▷ Important
        perform (Σ_ℓ, Σ_r) pair compression

A phase is one iteration of the main loop

Notes on analysis

A nondeterministic procedure is:

- **sound** transforms satisfiable to satisfiable, regardless of choices
- **complete** given a satisfiable instance it transforms it to a satisfiable one

Notes on analysis

A nondeterministic procedure is:

- **sound** transforms satisfiable to satisfiable, regardless of choices
- **complete** given a satisfiable instance it transforms it to a satisfiable one

In NLinSPACE we can analyse only "good choices":
if we exceed the space then we reject.

Solutions

Solution

- If there is a solution, there is one over $\Sigma =$ letters in the equation: Map all letters outside Σ to a fixed one in Σ.
- Done at the beginning of the phase.

Solutions

Solution

- If there is a solution, there is one over Σ = letters in the equation: Map all letters outside Σ to a fixed one in Σ.
- Done at the beginning of the phase.
- Then stick with corresponding solution.

Solutions

Solution

- If there is a solution, there is one over $\Sigma =$ letters in the equation: Map all letters outside Σ to a fixed one in Σ.
- Done at the beginning of the phase.
- Then stick with corresponding solution.

Corresponding nondeterministic choices

Given S the nondeterministic choices correspond to S, if they are done as if the algorithm knew S.

Solutions

Solution

- If there is a solution, there is one over $\Sigma =$ letters in the equation: Map all letters outside Σ to a fixed one in Σ.
- Done at the beginning of the phase.
- Then stick with corresponding solution.

Corresponding nondeterministic choices

Given S the nondeterministic choices correspond to S, if they are done as if the algorithm knew S.

- the first/last letter of $S(X)$
- length of a-prefix/suffix
- whether $S(X) = \epsilon$.
- Not: the choice of a partition.

Pair Compression: correctness

Lemma

PairCommpression is sound and complete.

Pair Compression: correctness

Lemma

PairCommpression is sound and complete. To be more precise: If $U = V$ *has a solution* S *then after PairCompression with corresponding nondeterministic choices the equation has a solution obtained by removing popped letters from* $S(X)$ *and performing pair compression on* $S(X)$.

Pair Compression: correctness

Lemma

PairCommpression is sound and complete. To be more precise: If $U = V$ has a solution S then after PairCompression with corresponding nondeterministic choices the equation has a solution obtained by removing popped letters from $S(X)$ and performing pair compression on $S(X)$.

Proof.

Soundness: let $U' = V'$ have a solution S'. Create S: take $S'(X)$, replace c_{ab} with ab and reattach the popped letters; this is $S(X)$. Then $S(U)$ is $S'(U')$ with c_{ab} replaced with ab.

Pair Compression: correctness

Lemma

PairCommpression is sound and complete. To be more precise: If $U = V$ has a solution S then after PairCompression with corresponding nondeterministic choices the equation has a solution obtained by removing popped letters from $S(X)$ and performing pair compression on $S(X)$.

Proof.

Soundness: let $U' = V'$ have a solution S'. Create S: take $S'(X)$, replace c_{ab} with ab and reattach the popped letters; this is $S(X)$. Then $S(U)$ is $S'(U')$ with c_{ab} replaced with ab.
Completeness: for those choices after popping each ab is either within variable or outside it. So the compression works. □

Block Compression: correctness

Lemma

BlockCompression is sound and complete. To be more precise: If $U = V$ has a solution S then after BlockCompression with corresponding nondeterministic choices the equation has a solution obtained by removing popped letters from $S(X)$ and performing block compression on $S(X)$.

Proof.

Proof as in the case of Pair Compression. □

Shortening property

Lemma

For S over Σ and the corresponding choices after one phase among each two consecutive letters in $S(U)$ at least one is compressed.

Shortening property

Lemma

For S over Σ and the corresponding choices after one phase among each two consecutive letters in $S(U)$ at least one is compressed.

Proof: as in the word case

Consider consecutive ab in the solution word:

- $a = b$ compressed as a block
- $a \neq b$ considered and compressed, or one of them was compressed earlier

Space consumption: initial notes

Block compression

Long blocks are a problem; a fix is already known:

- we do not guess explicit lengths, rather denote them as integer variables
- we calculate the blocks; lengths depends on those variables
- we identify the same lengths: equalities of linear expressions in terms of variables
- verify the system of such integer-equations
- compress

Space consumption: initial notes

Block compression

Long blocks are a problem; a fix is already known:

- we do not guess explicit lengths, rather denote them as integer variables
- we calculate the blocks; lengths depends on those variables
- we identify the same lengths: equalities of linear expressions in terms of variables
- verify the system of such integer-equations
- compress

Lemma

Block compression can be implemented in space linear in the size of the stored equation.

Space consumption: initial notes

Huffman coding

We need to recalculate Huffman coding.

- we build a labelled tree, labels to a leaf give the encoding
- calculate frequencies
- merge two least common symbols
- create a new node with two edges to those symbols, labelled with 0 and 1

This can be computed in space linear in the input.
Space bound is OK, just "delayed" by one step.

Dependency interval

For a letter in the equation we define a factor of the original equation, on which it depends.

Dependency interval

For a letter in the equation we define a factor of the original equation, on which it depends.

Definition (Dependency interval)

An interval of positions in the input equation is called a dependency interval (depint); basic depint has 1 position.
We associate a depint to each symbol in the equation; $D = \mathrm{dep}(i)$.

Dependency interval

For a letter in the equation we define a factor of the original equation, on which it depends.

Definition (Dependency interval)

An interval of positions in the input equation is called a dependency interval (depint); basic depint has 1 position.
We associate a depint to each symbol in the equation; $D = \mathrm{dep}(i)$.

- $D \sim D'$: the corresponding factors of initial equation are equal: $UV[D] = UV[D']$ (as sequence of letters and variables)
- we take their unions (only when result is an interval) $\mathrm{dep}(i) \cup \mathrm{dep}(j)$
- use $\supseteq, \subseteq$ have standard meaning ($\mathrm{dep}(i) \supseteq \mathrm{dep}(j)$)

Depints: idea

Depints

- assign to each letter in the equation a factor of the initial equation $UV[D]$
- letters with this fragment assigned are numbered $1, 2, \ldots, k$
- we assign to them codes $UV[D]\#1, UV[D]\#2, \ldots, UV[D]\#k$

Depints: idea

Depints

- assign to each letter in the equation a factor of the initial equation $UV[D]$
- letters with this fragment assigned are numbered $1, 2, \ldots, k$
- we assign to them codes $UV[D]\#1, UV[D]\#2, \ldots, UV[D]\#k$

Depints: idea

Depints

- assign to each letter in the equation a factor of the initial equation $UV[D]$
- letters with this fragment assigned are numbered $1, 2, \ldots, k$
- we assign to them codes $UV[D]\#1, UV[D]\#2, \ldots, UV[D]\#k$
- formally not encoding: assigns different codes to the same letter
- never assigns the same code to different letters
- worse than Huffman coding; enough to estimate its bit-size

How are dependency factors defined

dep(j) for j: position in the current equation

- initially: $\text{dep}(UV[i]) = \{i\}$

How are dependency factors defined

dep(j) for j: position in the current equation

- initially: $\text{dep}(UV[i]) = \{i\}$
- should be the same for compressed strings:
  when we compress a^ℓ inside $ba^\ell c$ with depints $D_b, D_1, D_2, \ldots, D_\ell, D_c$
  then each a gets a depint $D_b \cup D_1 \cup D_2 \cup \cdots \cup D_\ell \cup D_c$.

How are dependency factors defined

dep(j) for j: position in the current equation

- initially: $\text{dep}(UV[i]) = \{i\}$
- should be the same for compressed strings:
  when we compress a^ℓ inside $ba^\ell c$ with depints $D_b, D_1, D_2, \ldots, D_\ell, D_c$
  then each a gets a depint $D_b \cup D_1 \cup D_2 \cup \cdots \cup D_\ell \cup D_c$.
- (Σ_ℓ, Σ_r) compression: $a = UV[i] \in \Sigma_\ell$ with $\text{dep}(i) = D_1$ and
  $\text{dep}(i+1) = D_2$ gets a depint $D_1 \cup D_2$
  symmetrically for Σ_r.

Crucial properties

- For a depint D call $\mathrm{Pos}(D) = \{i \mid \mathrm{dep}(i) = D\}$ (in the current equation)
- $[i,j] \leq [i',j'] \iff i \leq i'$ and $j \leq j'$

Crucial properties

- For a depint D call $\mathrm{Pos}(D) = \{i \mid \mathrm{dep}(i) = D\}$ (in the current equation)
- $[i,j] \leq [i',j'] \iff i \leq i'$ and $j \leq j'$

(D1) $\mathrm{Pos}(D)$ is an interval (in the current equation).

(D2) For D, D' that have symbols in the equation, either: $D \leq D'$ or $D \geq D'$.

(D3) If $D \sim D'$ then $UV[\mathrm{Pos}(D)] = UV[\mathrm{Pos}(D')]$.

Dual view

- $\mathrm{Pos}_{\supseteq}(D) = \{j \mid \mathrm{dep}(j) \supseteq D\}$
- $\mathrm{Pos}_{\subseteq}(D) = \{j \mid \mathrm{dep}(j) \subseteq D\}$

We focus on $\mathrm{Pos}_{\supseteq}(D)$.

Dual view

- $\mathrm{Pos}_{\supseteq}(D) = \{j \mid \mathrm{dep}(j) \supseteq D\}$
- $\mathrm{Pos}_{\subseteq}(D) = \{j \mid \mathrm{dep}(j) \subseteq D\}$

We focus on $\mathrm{Pos}_{\supseteq}(D)$.

Lemma

$Pos_{\supseteq}(D)$ is an interval.

Dual view

- $\mathrm{Pos}_{\supseteq}(D) = \{j \mid \mathrm{dep}(j) \supseteq D\}$
- $\mathrm{Pos}_{\subseteq}(D) = \{j \mid \mathrm{dep}(j) \subseteq D\}$

We focus on $\mathrm{Pos}_{\supseteq}(D)$.

Lemma

$Pos_{\supseteq}(D)$ is an interval.

Proof.

We prove it together with D1–D3. Everything is easy induction except D3: $D \sim D' \Rightarrow UV[\mathrm{Pos}(D)] = UV[\mathrm{Pos}(D')]$.
The proof is simple with appropriate approach: through $\mathrm{Pos}_{\subseteq}(D)$ $\square$

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\mathrm{Pos}_{\subseteq}(D)$ and $\mathrm{Pos}_{\subseteq}(D')$

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\mathrm{Pos}_{\subseteq}(D)$ and $\mathrm{Pos}_{\subseteq}(D')$
- Claim: Those are intervals, corresponding letters are the same, corresponding depints are similar.

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\mathrm{Pos}_{\subseteq}(D)$ and $\mathrm{Pos}_{\subseteq}(D')$
- Claim: Those are intervals, corresponding letters are the same, corresponding depints are similar.
- Induction:

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\mathrm{Pos}_{\subseteq}(D)$ and $\mathrm{Pos}_{\subseteq}(D')$
- Claim: Those are intervals, corresponding letters are the same, corresponding depints are similar.
- Induction:
  - intervals: we only loose letters from both ends and perhaps gain from variables

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\mathrm{Pos}_{\subseteq}(D)$ and $\mathrm{Pos}_{\subseteq}(D')$
- Claim: Those are intervals, corresponding letters are the same, corresponding depints are similar.
- Induction:
  - intervals: we only loose letters from both ends and perhaps gain from variables
  - inside: everything is the same

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\mathrm{Pos}_{\subseteq}(D)$ and $\mathrm{Pos}_{\subseteq}(D')$
- Claim: Those are intervals, corresponding letters are the same, corresponding depints are similar.
- Induction:
  - intervals: we only loose letters from both ends and perhaps gain from variables
  - inside: everything is the same
  - interaction with outside: (proof by example)
    $\mathrm{dep}(a) \in D' \subseteq D_1$ is left-most with this property, and in Σ_r
    its depint is extended $\Rightarrow$ we no longer care about it
    it works the same for D_2.

Proof—ctd.

Proof.

- Fix depint $D \sim D'$; consider $\text{Pos}_{\subseteq}(D)$ and $\text{Pos}_{\subseteq}(D')$
- Claim: Those are intervals, corresponding letters are the same, corresponding depints are similar.
- Induction:
  - intervals: we only loose letters from both ends and perhaps gain from variables
  - inside: everything is the same
  - interaction with outside: (proof by example)
    $\text{dep}(a) \in D' \subseteq D_1$ is left-most with this property, and in Σ_r
    its depint is extended $\Rightarrow$ we no longer care about it
    it works the same for D_2.
- some inclusion-exclusion and intersections $\square$

Encoding

Definition (Encoding)

Fix depint D, encode letters with this depint as $U_0 V_0[D]\#1\#, U_0 V_0[D]\#2\#, \ldots$

- i in binary
- $U_0 V_0[D]$ as in the input equation
- for $D \sim D'$ the encoding is the same

Encoding

Definition (Encoding)

Fix depint D, encode letters with this depint as $U_0 V_0[D]\#1\#, U_0 V_0[D]\#2\#, \ldots$

- i in binary
- $U_0 V_0[D]$ as in the input equation
- for $D \sim D'$ the encoding is the same

- formally not encoding: assigns different codes to the same letter
- never assigns the same code to different letters
- is worse than Huffman coding
  enough to estimate its bit-size

Idea

Ensure that

- each variable pops $\mathcal{O}(1)$ letters per phase
- each $\mathrm{Pos}_{\supseteq}(D)$ expands by $\mathcal{O}(1)$ positions per phase

Idea

Ensure that

- each variable pops $\mathcal{O}(1)$ letters per phase
- each $\text{Pos}_{\supseteq}(D)$ expands by $\mathcal{O}(1)$ positions per phase

Then for a basic depint $|\text{Pos}_{\supseteq}(D)| = \mathcal{O}(1)$:

- old $\text{Pos}_{\supseteq}(D)$ looses $1/3$ of its positions (everything is compressed)

$$k' \leq \frac{2}{3}k + c \quad \Rightarrow \quad \text{bound } 3c$$

Idea

Ensure that

- each variable pops $\mathcal{O}(1)$ letters per phase
- each $\mathrm{Pos}_{\supseteq}(D)$ expands by $\mathcal{O}(1)$ positions per phase

Then for a basic depint $|\mathrm{Pos}_{\supseteq}(D)| = \mathcal{O}(1)$:

- old $\mathrm{Pos}_{\supseteq}(D)$ looses $1/3$ of its positions (everything is compressed)

$$k' \leq \frac{2}{3}k + c \quad \Rightarrow \quad \text{bound } 3c$$

The space is linear

- i occurs $3c$ times in all depints $\Rightarrow$ letter from the input occurs $3c$ times in the encodings
- each depint has length $\leq 3c$ $\Rightarrow$ numbers in the encoding are constant-length

Idea

Ensure that

- each variable pops $\mathcal{O}(1)$ letters per phase
- each $\mathrm{Pos}_{\supseteq}(D)$ expands by $\mathcal{O}(1)$ positions per phase

Then for a basic depint $|\mathrm{Pos}_{\supseteq}(D)| = \mathcal{O}(1)$:

- old $\mathrm{Pos}_{\supseteq}(D)$ looses $1/3$ of its positions (everything is compressed)

$$k' \leq \frac{2}{3}k + c \quad \Rightarrow \quad \text{bound } 3c$$

The space is linear

- i occurs $3c$ times in all depints $\Rightarrow$ letter from the input occurs $3c$ times in the encodings
- each depint has length $\leq$ 3c $\Rightarrow$ numbers in the encoding are constant-length

Make this in expectation.

Blocking

Recall

Σ: letters in the equation at the beginning of the phase.

Blocking

Recall

Σ: letters in the equation at the beginning of the phase.

Letters outside Σ (ones replacing compressed strings) are not popped.

Blocking

Recall

Σ: letters in the equation at the beginning of the phase.

Letters outside Σ (ones replacing compressed strings) are not popped.

A variable is left/right blocked when the left/right-most or second left/right most letter in $S(X)$ is outside Σ (or $|S(X)| = 1$).

Blocking

Recall

Σ: letters in the equation at the beginning of the phase.

Letters outside Σ (ones replacing compressed strings) are not popped.

A variable is left/right blocked when the left/right-most or second left/right most letter in $S(X)$ is outside Σ (or $|S(X)| = 1$).

Positions with letters outside Σ do not have their depint changed.

Blocking

Recall

Σ: letters in the equation at the beginning of the phase.

Letters outside Σ (ones replacing compressed strings) are not popped.

A variable is left/right blocked when the left/right-most or second left/right most letter in $S(X)$ is outside Σ (or $|S(X)| = 1$).

Positions with letters outside Σ do not have their depint changed.

A depint D is left/right blocked when the letter one or two to the left/right of $\mathrm{Pos}_{\supseteq}(D)$ is outside Σ (or there is one only one letter to the left/right).

Blocking

Recall

Σ: letters in the equation at the beginning of the phase.

Letters outside Σ (ones replacing compressed strings) are not popped.

A variable is left/right blocked when the left/right-most or second left/right most letter in $S(X)$ is outside Σ (or $|S(X)| = 1$).

Positions with letters outside Σ do not have their depint changed.

A depint D is left/right blocked when the letter one or two to the left/right of $\text{Pos}_{\supseteq}(D)$ is outside Σ (or there is one only one letter to the left/right).

Lemma

When a variable (depint) becomes left/right blocked then it stays so in this phase and pops (extends to) at most one letter.

Strategy

Our only nondeterministic choice is the partition.

Strategy

Our only nondeterministic choice is the partition.

Strategy

Choose the partition to alternatively halve the sums below:

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

$$\sum_{\substack{D\text{: basic depint} \\ \text{left-unblocked}}} 1 + \sum_{\substack{D\text{: basic depint} \\ \text{right-unblocked}}} 1$$

Strategy

Our only nondeterministic choice is the partition.

Strategy

Choose the partition to alternatively halve the sums below:

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

$$\sum_{\substack{D\text{: basic depint} \\ \text{left-unblocked}}} 1 + \sum_{\substack{D\text{: basic depint} \\ \text{right-unblocked}}} 1$$

- the first limits the number of letters popped from equations
- the second: extensions of $\text{Pos}_{\supseteq}(D)$

Strategy exists

Proof.

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

Strategy exists

Proof.

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

A random partition reduces this sum by 1/2: $S(X) = abc \dots$

Strategy exists

Proof.

$$\sum_{\substack{X\in\mathcal{X}\\\text{left-unblocked}}} n_X + \sum_{\substack{X\in\mathcal{X}\\\text{right-unblocked}}} n_X$$

A random partition reduces this sum by 1/2: $S(X) = abc\dots$

$a \notin \Sigma$ or $b \notin \Sigma$ or no b X is left-blocked

Strategy exists

Proof.

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

A random partition reduces this sum by 1/2: $S(X) = abc \ldots$

$a \notin \Sigma$ or $b \notin \Sigma$ or no b X is left-blocked

$c \notin \Sigma$ or no c 1/2 probability that a is left-popped

$a, b, c \in \Sigma$ 1/2 probability that b is compressed.

Strategy exists

Proof.

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

A random partition reduces this sum by 1/2: $S(X) = abc\dots$

$a \notin \Sigma$ or $b \notin \Sigma$ or no b X is left-blocked

$c \notin \Sigma$ or no c 1/2 probability that a is left-popped

$a, b, c \in \Sigma$ 1/2 probability that b is compressed.

Symmetrically for the right-hand side.

Strategy exists

Proof.

$$\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X$$

A random partition reduces this sum by 1/2: $S(X) = abc \dots$

$a \notin \Sigma$ or $b \notin \Sigma$ or no b X is left-blocked

$c \notin \Sigma$ or no c 1/2 probability that a is left-popped

$a, b, c \in \Sigma$ 1/2 probability that b is compressed.

Symmetrically for the right-hand side.
Go for expectation over all variables.

Strategy exists

Proof.

$$\sum_{\substack{X\in\mathcal{X}\\ \text{left-unblocked}}} n_X + \sum_{\substack{X\in\mathcal{X}\\ \text{right-unblocked}}} n_X$$

A random partition reduces this sum by 1/2: $S(X) = abc\ldots$

$a \notin \Sigma$ or $b \notin \Sigma$ or no b X is left-blocked

$c \notin \Sigma$ or no c 1/2 probability that a is left-popped

$a, b, c \in \Sigma$ 1/2 probability that b is compressed.

Symmetrically for the right-hand side.
Go for expectation over all variables.
Similarly for basic depints. $\square$

Upper-bounding the size of the equation

Lemma

The depint-encoding of an equation U, V is not larger than $H(U, V)$.

Upper-bounding the size of the equation

Lemma

The depint-encoding of an equation U, V is not larger than $H(U, V)$.

The proof works for arbitrary encoding of the input.
From now on we use only fix-size for each symbol (streamlining).

Upper-bounding the size of the equation

Lemma

The depint-encoding of an equation U, V is not larger than $H(U, V)$.

The proof works for arbitrary encoding of the input.
From now on we use only fix-size for each symbol (streamlining).

Let e is the bit-size of the initial encoding.

$$H_d(U, V) = \mathrm{e} \cdot \sum_{D\text{:basic depint}} |\mathrm{Pos}_{\supseteq}(D)|$$

$$H_n(U, V) = \sum_{D\text{:basic depint}} 2|\mathrm{Pos}_{\supseteq}(D)| \cdot \log(|\mathrm{Pos}_{\supseteq}(D)| + 1) \ ,$$

$$H(U, V) = H_d(U, V) + H_n(U, V)$$

Upper-bounding the size of the equation

Lemma

The depint-encoding of an equation U, V is not larger than $H(U, V)$.

The proof works for arbitrary encoding of the input.
From now on we use only fix-size for each symbol (streamlining).

Let e is the bit-size of the initial encoding.

$$H_d(U, V) = \mathrm{e} \cdot \sum_{D:\text{basic depint}} |\mathrm{Pos}_{\supseteq}(D)|$$

$$H_n(U, V) = \sum_{D:\text{basic depint}} 2|\mathrm{Pos}_{\supseteq}(D)| \cdot \log(|\mathrm{Pos}_{\supseteq}(D)| + 1) \ ,$$

$$H(U, V) = H_d(U, V) + H_n(U, V)$$

- the first upper bounds the bits used by $UV[D]$
- the second upper-bounds the bits used by numbers in $UV[D]\#i$

Proof

Size of all depints:
Let $D = i$ be a basic depint. $U_0 V_0[i]$ has size e and occurs in code for $|\mathrm{Pos}_{\supseteq}(D)|$ many symbols. This gives

$$\sum_{D\text{:basic depint}} e \cdot |\mathrm{Pos}_{\supseteq}(D)| = H_d(U, V)$$

Proof

Size of all depints:

Let $D = i$ be a basic depint. $U_0 V_0[i]$ has size e and occurs in code for $|\text{Pos}_{\supseteq}(D)|$ many symbols. This gives

$$\sum_{D\text{:basic depint}} e \cdot |\text{Pos}_{\supseteq}(D)| = H_d(U, V)$$

Size of numbers:

When depint D' has k positions, they use $\leq 2k \log k = h(k)$ bits.

$$\sum_{D'\text{: depint}} h(|\#\text{Pos}(D')|)$$

Proof

Size of all depints:

Let $D = i$ be a basic depint. $U_0 V_0[i]$ has size e and occurs in code for $|\mathrm{Pos}_{\supseteq}(D)|$ many symbols. This gives

$$\sum_{D\text{:basic depint}} e \cdot |\mathrm{Pos}_{\supseteq}(D)| = H_d(U, V)$$

Size of numbers:

When depint D' has k positions, they use $\leq 2k \log k = h(k)$ bits.

$$\sum_{D'\text{: depint}} h(|\#\mathrm{Pos}(D')|)$$

Easier to calculate:

$$D' = \underbrace{D_1 \cup D_2 \cup \cdots \cup D_\ell}_{\text{basic depints}} \quad \Rightarrow \quad \log |\#\mathrm{Pos}(D)| \leq \log |\mathrm{Pos}_{\supseteq}(D_i)|$$

Proof

Size of all depints:
Let $D = i$ be a basic depint. $U_0 V_0[i]$ has size e and occurs in code for $|\text{Pos}_{\supseteq}(D)|$ many symbols. This gives

$$\sum_{D\text{:basic depint}} e \cdot |\text{Pos}_{\supseteq}(D)| = H_d(U, V)$$

Size of numbers:
When depint D' has k positions, they use $\leq 2k \log k = h(k)$ bits.

$$\sum_{D'\text{: depint}} h(|\#\text{Pos}(D')|) \leq \sum_{D\text{:basic depint}} h(|\text{Pos}_{\supseteq}(D)|) = H_n(U, V)$$

Easier to calculate:

$$D' = \underbrace{D_1 \cup D_2 \cup \cdots \cup D_\ell}_{\text{basic depints}} \quad \Rightarrow \quad \log|\#\text{Pos}(D)| \leq \log|\text{Pos}_{\supseteq}(D_i)|$$

Outline of the proof

Let

- (U, V) equation at the beginning of the phase
- (U', V') equation at the end of the phase
- (U_0, V_0): input eqaution

Outline of the proof

Let

- (U, V) equation at the beginning of the phase
- (U', V') equation at the end of the phase
- (U_0, V_0): input eqaution

We show that the strategy guarantees that

$$H_d(U', V') = \frac{5}{6} H_d(U, V) + \alpha H_d(U_0, V_0)$$

Outline of the proof

Let

- (U, V) equation at the beginning of the phase
- (U', V') equation at the end of the phase
- (U_0, V_0): input eqaution

We show that the strategy guarantees that

$$H_d(U', V') = \frac{5}{6} H_d(U, V) + \alpha H_d(U_0, V_0)$$

By induction, this gives an $\mathcal{O}(H_d(U_0, V_0))$ bound on H_d.
And $H(U_0, V_0) = H_d(U_0, V_0) + H_n(U_0, V_0) = \mathcal{O}(e|U_0 V_0|)$.
Similar bound is shown for $H_n(U, V)$.
So the encoding is linear-size.

Limiting H_d: new letters

Bit-size of letters popped in one pair compression

At most 1 per not blocked side of occurrence of variable, bitsize e.

Limiting H_d: new letters

Bit-size of letters popped in one pair compression

At most 1 per not blocked side of occurrence of variable, bitsize e.

$$e \left(\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X \right)$$

Limiting H_d: new letters

Bit-size of letters popped in one pair compression

At most 1 per not blocked side of occurrence of variable, bitsize e.

$$e\left(\sum_{\substack{X\in\mathcal{X}\\ \text{left-unblocked}}} n_X + \sum_{\substack{X\in\mathcal{X}\\ \text{right-unblocked}}} n_X\right)$$

At the beginning of the phase: at most $2e|U_0 V_0| = 2n$.

Limiting H_d: new letters

Bit-size of letters popped in one pair compression

At most 1 per not blocked side of occurrence of variable, bitsize e.

$$e \left(\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked}}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked}}} n_X \right)$$

At the beginning of the phase: at most $2e|U_0 V_0| = 2n$.
By strategy: at least halved every other step, so at most

$$2n + 2n + n + n + \ldots \leq 8n$$

Extending $\mathrm{Pos}_{\supseteq}(D)$

Similar analysis for extension of basic depint.

Bit size of new $\mathrm{Pos}_{\supseteq}(D)$ letters?

At most 1 per not blocked side of basic depint.

Extending $\mathrm{Pos}_{\supseteq}(D)$

Similar analysis for extension of basic depint.

Bit size of new $\mathrm{Pos}_{\supseteq}(D)$ letters?

At most 1 per not blocked side of basic depint.

$$e \left(\sum_{\substack{D \text{ basic depint} \\ \text{left-unblocked}}} 1 + \sum_{\substack{D \text{ basic depint} \\ \text{right-unblocked}}} 1 \right)$$

Extending $\text{Pos}_{\supseteq}(D)$

Similar analysis for extension of basic depint.

Bit size of new $\text{Pos}_{\supseteq}(D)$ letters?

At most 1 per not blocked side of basic depint.

$$e \left(\sum_{\substack{D \text{ basic depint} \\ \text{left-unblocked}}} 1 + \sum_{\substack{D \text{ basic depint} \\ \text{right-unblocked}}} 1 \right)$$

At the beginning of the phase this is at most $2e|U_0 V_0| = 2n$.

Extending $\mathrm{Pos}_{\supseteq}(D)$

Similar analysis for extension of basic depint.

Bit size of new $\mathrm{Pos}_{\supseteq}(D)$ letters?

At most 1 per not blocked side of basic depint.

$$e \left(\sum_{\substack{D \text{ basic depint} \\ \text{left-unblocked}}} 1 + \sum_{\substack{D \text{ basic depint} \\ \text{right-unblocked}}} 1 \right)$$

At the beginning of the phase this is at most $2e|U_0 V_0| = 2n$.
The rest is similar as before.

Shortening of $\mathrm{Pos}_{\supseteq}(D)$

Recall

$$H_d(U,V) = \sum_{D\text{: basic depint}} e|\mathrm{Pos}_{\supseteq}(D)|$$

Shortening of $\mathrm{Pos}_{\supseteq}(D)$

Recall

$$H_d(U, V) = \sum_{D\text{: basic depint}} e|\mathrm{Pos}_{\supseteq}(D)|$$

- Consider $\mathrm{Pos}_{\supseteq}(D)$ at the beginning of the phase.
- By Lemma, among two positions in $\mathrm{Pos}_{\supseteq}(D)$ at least 1 is compressed.
- So $\mathrm{Pos}_{\supseteq}(D)$ looses 1/3 of its positions due to compression.

Shortening of $\mathrm{Pos}_{\supseteq}(D)$

Recall

$$H_d(U, V) = \sum_{D\text{: basic depint}} e|\mathrm{Pos}_{\supseteq}(D)|$$

- Consider $\mathrm{Pos}_{\supseteq}(D)$ at the beginning of the phase.
- By Lemma, among two positions in $\mathrm{Pos}_{\supseteq}(D)$ at least 1 is compressed.
- So $\mathrm{Pos}_{\supseteq}(D)$ looses $1/3$ of its positions due to compression.

$$H_d(U', V') \leq \frac{2}{3} H_d(U, V) + \mathcal{O}(H_d(U_0, V_0))$$

H_n

- p_D, e_D, k_D: popped $\text{Pos}_{\supseteq}(D)$, extended $\text{Pos}_{\supseteq}(D)$, $\text{Pos}_{\supseteq}(D)$ from the beginning of the phase

H_n

- p_D, e_D, k_D: popped $\text{Pos}_{\supseteq}(D)$, extended $\text{Pos}_{\supseteq}(D)$, $\text{Pos}_{\supseteq}(D)$ from the beginning of the phase
- previously

$$H_d(U,V) = e\sum_D k_D \to e\sum_D \left(\frac{2}{3}k_D + p_D + e_D\right) = H_d(U',V')$$

$$\text{where } \sum_D e_D + p_D = \mathcal{O}(|U_0 V_0|)$$

H_n

- p_D, e_D, k_D: popped $\text{Pos}_{\supseteq}(D)$, extended $\text{Pos}_{\supseteq}(D)$, $\text{Pos}_{\supseteq}(D)$ from the beginning of the phase
- previously

$$H_d(U,V) = e\sum_D k_D \to e\sum_D \left(\frac{2}{3}k_D + p_D + e_D\right) = H_d(U',V')$$

$$\text{where } \sum_D e_D + p_D = \mathcal{O}(|U_0V_0|)$$

- now

$$H_n(U,V) = \sum_D h(k_D) \to \sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right) = H_n(U',V'),$$

$$\text{where } \sum_D h(p_D + e_D) = \mathcal{O}(|U_0V_0|)$$

H_n

- p_D, e_D, k_D: popped $\mathrm{Pos}_{\supseteq}(D)$, extended $\mathrm{Pos}_{\supseteq}(D)$, $\mathrm{Pos}_{\supseteq}(D)$ from the beginning of the phase
- previously

$$H_d(U,V) = e\sum_D k_D \to e\sum_D \left(\frac{2}{3}k_D + p_D + e_D\right) = H_d(U',V')$$

$$\text{where } \sum_D e_D + p_D = \mathcal{O}(|U_0 V_0|)$$

- now

$$H_n(U,V) = \sum_D h(k_D) \to \sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right) = H_n(U',V'),$$

$$\text{where } \sum_D h(p_D + e_D) = \mathcal{O}(|U_0 V_0|)$$

- Something more is needed (true, but separate analysis):

.

H_n

- p_D, e_D, k_D: popped $\text{Pos}_{\supseteq}(D)$, extended $\text{Pos}_{\supseteq}(D)$, $\text{Pos}_{\supseteq}(D)$ from the beginning of the phase
- previously

$$H_d(U,V) = e\sum_D k_D \to e\sum_D \left(\frac{2}{3}k_D + p_D + e_D\right) = H_d(U',V')$$

$$\text{where } \sum_D e_D + p_D = \mathcal{O}(|U_0 V_0|)$$

- now

$$H_n(U,V) = \sum_D h(k_D) \to \sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right) = H_n(U',V'),$$

$$\text{where } \sum_D h(p_D + e_D) = \mathcal{O}(|U_0 V_0|)$$

- Something more is needed (true, but separate analysis):

$$\sum_{i=1}^{\infty} \frac{i}{2^i} = 2, \qquad h \text{ is almost linear.}$$

Idea

Estimate $\sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right)$

- first estimate $\sum_D h(p_D), \sum_D h(e_D)$ as $\mathcal{O}(|U_0, V_0|)$

Idea

Estimate $\displaystyle\sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right)$

- first estimate $\sum_D h\left(p_D\right), \sum_D h\left(e_D\right)$ as $\mathcal{O}(|U_0, V_0|)$
- $h(p_D + e_D) \leq 4h(p_D) + 4h(p_D)$

Idea

Estimate $\sum_{D} h\left(\frac{2}{3}k_d + p_D + e_D\right)$

- first estimate $\sum_D h\left(p_D\right), \sum_D h\left(e_D\right)$ as $\mathcal{O}(|U_0, V_0|)$
- $h(p_D + e_D) \leq 4h(p_D) + 4h(p_D)$
- $\sum_D h\left(p_D + e_D\right) = \mathcal{O}(|U_0, V_0|)$

Idea

Estimate $\sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right)$

- first estimate $\sum_D h(p_D), \sum_D h(e_D)$ as $\mathcal{O}(|U_0, V_0|)$
- $h(p_D + e_D) \leq 4h(p_D) + 4h(p_D)$
- $\sum_D h(p_D + e_D) = \mathcal{O}(|U_0, V_0|)$
- if $\frac{2}{3}k_D + p_D + e_D \leq \frac{5}{6}k_d$: OK

Idea

Estimate $\sum_D h\left(\frac{2}{3}k_d + p_D + e_D\right)$

- first estimate $\sum_D h(p_D), \sum_D h(e_D)$ as $\mathcal{O}(|U_0, V_0|)$
- $h(p_D + e_D) \leq 4h(p_D) + 4h(p_D)$
- $\sum_D h(p_D + e_D) = \mathcal{O}(|U_0, V_0|)$
- if $\frac{2}{3}k_D + p_D + e_D \leq \frac{5}{6}k_d$: OK
- otherwise $\frac{2}{3}k_d + p_D + e_D > \frac{5}{6}k_d \Rightarrow$:

$$\frac{2}{3}k_D + p_D + e_D \leq 5(p_D + e_D)$$

Idea

Estimate $\sum_{D} h\left(\frac{2}{3}k_d + p_D + e_D\right)$

- first estimate $\sum_D h\left(p_D\right), \sum_D h\left(e_D\right)$ as $\mathcal{O}(|U_0, V_0|)$
- $h(p_D + e_D) \leq 4h(p_D) + 4h(p_D)$
- $\sum_D h\left(p_D + e_D\right) = \mathcal{O}(|U_0, V_0|)$
- if $\frac{2}{3}k_D + p_D + e_D \leq \frac{5}{6}k_d$: OK
- otherwise $\frac{2}{3}k_d + p_D + e_D > \frac{5}{6}k_d \Rightarrow$:

$$\frac{2}{3}k_D + p_D + e_D \leq 5(p_D + e_D)$$

-

$$\begin{aligned}\sum_{D} h\left(\frac{2}{3}k_D + p_D + e_D\right) &\leq \frac{5}{6}\sum_{D} h\left(k_D\right) + \sum_{D} h\left(5p_D + 5e_D\right)\\ &\leq \frac{5}{6}H_n(U, V) + \mathcal{O}(|U_0, V_0|)\end{aligned}$$

$\sum_D h(p_D)$ ($\sum_D h(e_D)$ is similar)

This is only for D corresponding to a variable. Let D: position of X.

$\sum_D h(p_D)$ ($\sum_D h(e_D)$ is similar)

This is only for D corresponding to a variable. Let D: position of X.

$$h(p_D) = 2p_D \log p_D \leq 25 + \sum_{i \geq 1} i \cdot \left([X\text{left-unbl. in } I_i] + [X\text{right-unbl. in } I_i]\right).$$

$\sum_D h(p_D)$ ($\sum_D h(e_D)$ is similar)

This is only for D corresponding to a variable. Let D: position of X.

$$h(p_D) = 2p_D \log p_D \leq 25 + \sum_{i \geq 1} i \cdot ([X\text{left-unbl. in } I_i] + [X\text{right-unbl. in } I_i]) .$$

- right side is $\Omega(p_D^2)$: some side was unblocked for $p_D/2$ partitions.
- 25 just for small p_D

$\sum_D h(p_D)$ ($\sum_D h(e_D)$ is similar)

This is only for D corresponding to a variable. Let D: position of X.

$$h(p_D) = 2p_D \log p_D \leq 25 + \sum_{i \geq 1} i \cdot ([X\text{left-unbl. in } I_i] + [X\text{right-unbl. in } I_i]) .$$

- right side is $\Omega(p_D^2)$: some side was unblocked for $p_D/2$ partitions.
- 25 just for small p_D

$$\sum_{X \in \mathcal{X}} 25 n_X + \sum_{i \geq 1} i \cdot \left(\sum_{\substack{X \in \mathcal{X} \\ \text{left-unblocked in } I_i}} n_X + \sum_{\substack{X \in \mathcal{X} \\ \text{right-unblocked in } I_i}} n_X \right) .$$

- value in braces is $|U_0, V_0|$ initially and halves every other partition
- we sum-up a series $\sim \sum_i \frac{i}{2^i} = 2$.