Solid Hypercodes

UWORCS 2015

Taylor J. Smith

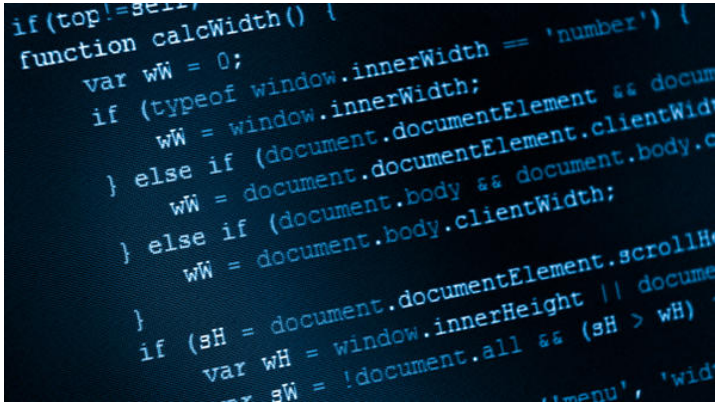
Department of Computer Science
University of Western Ontario
London, Ontario, Canada

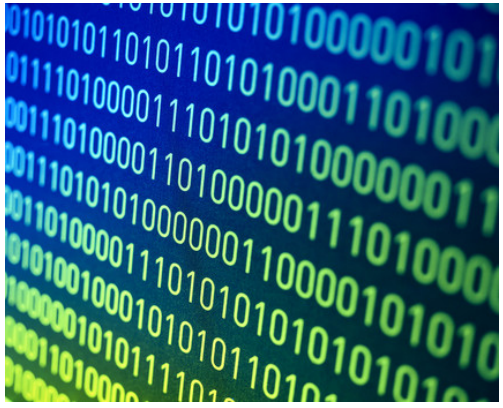
April 9, 2015

Table of Contents

Table of Contents

What is a code?





What is a code?

Terminology

0 symbol

What is a code?

Terminology

00110101 word (w)

What is a code?

Terminology

00110101
10100101
00000011
10010110
⋮

language (L)

What is a code?

Definition (Code)

A language L is a code if every word w has a unique L-factorization, that is, $u_1 u_2 \dots u_m = v_1 v_2 \dots v_n$ with $u_i, v_j \in L$ for all i and j implies $m = n$ and $u_i = v_i$ for all i.

What is a code?

Definition (Code)

A language L is a code if every word w has a unique L-factorization, that is, $u_1 u_2 \dots u_m = v_1 v_2 \dots v_n$ with $u_i, v_j \in L$ for all i and j implies $m = n$ and $u_i = v_i$ for all i.

Simpler Definition (Code)

A language L is a code if every string composed of words from L has a single decomposition.

What is a code?

Definition (Code)

A language L is a code if every word w has a unique L-factorization, that is, $u_1 u_2 \dots u_m = v_1 v_2 \dots v_n$ with $u_i, v_j \in L$ for all i and j implies $m = n$ and $u_i = v_i$ for all i.

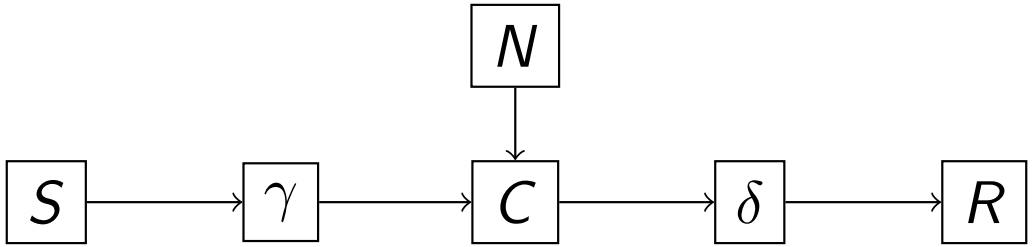
Simpler Definition (Code)

A language L is a code if every string composed of words from L has a single decomposition.

Example

$L = \{0, 1\}$
$w = 00110101$

The information processing and transmission model:



All communication is represented by these five components.

For any message w, it is expected that $\delta(\gamma(w)) = w$.

However, the presence of noise may introduce errors.

The development of communication methods that ensure less noise, and thus a more reliable transmission, lies at the foundation of channel coding.

Classes of codes

- Prefix codes
- Suffix codes
- Infix codes
- Bifix codes
- Overlap-free codes
- Solid codes
- Hypercodes
- ...

Solid codes

Definition (Solid code)

A language L is a solid code if it satisfies the following conditions:

1. no word in L is a subword of another word in L (infix-freeness)
2. no proper prefix of a word in L is a proper suffix of a word in L (overlap-freeness)

Definition (Solid code)

A language L is a solid code if it satisfies the following conditions:

1. no word in L is a subword of another word in L (infix-freeness)
2. no proper prefix of a word in L is a proper suffix of a word in L (overlap-freeness)

Example

$L = \{00111, 01\}$

Does not meet infix-free property.

Definition (Solid code)

A language L is a solid code if it satisfies the following conditions:

1. no word in L is a subword of another word in L (infix-freeness)
2. no proper prefix of a word in L is a proper suffix of a word in L (overlap-freeness)

Example

$L = \{00111, 01\}$

Does not meet infix-free property.

$L = \{0011, 1001\}$

Does not meet overlap-free property.

Definition (Solid code)

A language L is a solid code if it satisfies the following conditions:

1. no word in L is a subword of another word in L (infix-freeness)
2. no proper prefix of a word in L is a proper suffix of a word in L (overlap-freeness)

Example

$L = \{00111, 01\}$

Does not meet infix-free property.

$L = \{0011, 1001\}$

Does not meet overlap-free property.

$L = \{00111, 010111\}$

This is a solid code!

Definition (Hypercode)

A language L is a hypercode if no word in L is a proper subword of another word in L.

Definition (Hypercode)

A language L is a hypercode if no word in L is a proper subword of another word in L.

Example

$L = \{010, 0001000\}$

The first word is a proper subword of the second word.

Definition (Hypercode)

A language L is a hypercode if no word in L is a proper subword of another word in L.

Example

$L = \{010, 0001000\}$

The first word is a proper subword of the second word.

$L = \{1010000, 11100\}$

This is a hypercode!

Table of Contents

Definition (Solid hypercode)

A language L is a solid hypercode if it is both a solid code and a hypercode.

By using solid hypercodes, transmitted words become resistant to errors both between code words and within code words.

1010000 11100	1111100 11010
1100000 10100	10100000 11100
1100000 11010	10100000 101100
1110000 10100	10100000 111000
1110000 11010	10100000 111100
1111000 10100	10100100 111000
1111000 11010	10100100 111100
1111010 11000	10101000 111100
1111100 10100	10101100 111000

Properties

The **solid** aspect allows for synchronization capabilities.
The **hyper** aspect allows for thorough protection against errors.

Properties

The **solid** aspect allows for synchronization capabilities.
The **hyper** aspect allows for thorough protection against errors.

Checking for the solid hypercode property

1a. Check if the language is a block code.
- If so, then the language is hyper
- Otherwise, continue

Properties

The **solid** aspect allows for synchronization capabilities.
The **hyper** aspect allows for thorough protection against errors.

Checking for the solid hypercode property

1a. Check if the language is a block code.
- If so, then the language is hyper
- Otherwise, continue

1b. Check the embedding order of each pair of words.
- If one word can be embedded in the other, then stop
- Otherwise, check the next pair of words

Properties

The **solid** aspect allows for synchronization capabilities.
The **hyper** aspect allows for thorough protection against errors.

Checking for the solid hypercode property

1a. Check if the language is a block code.
- If so, then the language is hyper
- Otherwise, continue

1b. Check the embedding order of each pair of words.
- If one word can be embedded in the other, then stop
- Otherwise, check the next pair of words

2. Check the first symbol of each word.
- If symbols do not match, then stop
- Otherwise, check each pair of words for overlaps

Table of Contents

Overview

`Diplomarbeit.cpp`: a method of generating solid hypercodes [1]

- Functional, but very slow
- Short-term goal: to understand and improve this software tool
- Long-term goal: to generate large sets of solid hypercodes
- Problem: everything is written in German!

Analysis

A detailed analysis of the most important parts of the original work allows non-German speakers to understand the author's intent.

Two algorithms are discussed in the original work.

- Algorithm I, which enumerates all words and performs a check on each
  - Suffers from severe performance issues
- Algorithm II, which uses backtracking to enumerate possible words and "take a step back" if a check fails
  - Slightly better, but not yet perfect

Improvements

Performance

- modified `Next()` and `Previous()` methods to reduce redundant computation
- enhanced user input/output capabilities
- wrote new program to scan output file and keep "interesting" sets of code words

Cosmetic

- translated all variable names and comments to German
- added additional comments
- brought consistency to formatting and spacing

Improvements

Comparative run times of `Diplomarbeit.cpp` (in seconds)

Input	Original	Modified
500	28	26
550	39	38
600	39	38
650	39	37
700	39	38
750	39	38
800	39	38
850	52	50
900	166	158
950	1052	1013

Table of Contents

Conclusions

- One of the most important aspects of communication is resilience against errors
- Solid hypercodes are resilient against errors
- An efficient method of checking the solid hypercode property was developed
- A method of generating sets of solid hypercodes was analyzed
- This method was modified for greater performance
- There is still room for improvement!

Future work

- How can we implement this checking method in the real world?
- How can we check non-binary alphabets?
- Will the generation method run faster using this check?
- Are there any other redundancies in the generation method?

References

[1] C. Herrmann. Entwicklung von Methoden zur Aufzählung und Untersuchung von soliden Codes, Hyper-Codes und soliden Hyper-Codes. Diplomarbeit, Universität Potsdam, 2005. In German.

[2] T. J. Smith. A study of solid hypercodes. Bachelor's thesis, University of Western Ontario, 2015.