Применение системы остаточных классов в обработке текстовой информации

В.И. Шиян, С.М. Власов, А.А. Греф, В.А. Дударев, Н.М. Топорков

Кубанский государственный университет, Краснодар

Аннотация: В статье рассматривается применение системы остаточных классов в обработке текстовой информации. Система остаточных классов, основанная на принципах модулярной арифметики, представляет числа в виде наборов остатков по взаимно простым модулям. Такой подход обеспечивает возможность параллельного выполнения вычислений, потенциальное сжатие данных и повышенную устойчивость к помехам. В работе рассмотрены вопросы кодирования символов, параллельной обработки информации, обнаружения и коррекции ошибок, вычислительные преимущества при реализации полиномиальных хеш-функций, а также практические ограничения применения системы остаточных классов.

Ключевые слова: система остаточных классов, модулярная арифметика, обработка текста, параллельные вычисления, сжатие данных, помехоустойчивость, китайская теорема об остатках, полиномиальные хеши, коррекция ошибок, компьютерная лингвистика.

Введение

Современная обработка текста требует эффективных подходов, обеспечивающих скорость, устойчивость к ошибкам и сжатие [1]. Система остаточных классов (СОК), основанная на модулярной арифметике [2], является перспективным направлением. В работе исследуются возможности СОК для представления и обработки текстовых данных, анализируются её преимущества и ограничения, а также потенциальное применение в вычислительной лингвистике [3].

Математические основы СОК

СОК базируется на теории чисел и сравнениях по модулю [4]. Целое число X в $[0, M-1]$, где M – произведение взаимно простых модулей $m_1, m_2, \ldots, m_k$, однозначно представляется кортежем остатков:

$$X \leftrightarrow (x_1, x_2, \ldots, x_k), \quad (1)$$

где $x_i = X \mod m_i$.

Однозначность гарантируется китайской теоремой об остатках (КТО): для любого набора остатков $(r_1, r_2, \ldots, r_k)$, где $0 \le r_i < m_i$, существует единственное X в $[0, M-1]$, удовлетворяющее системе сравнений:

$$\begin{cases} X \equiv r_1 \mod m_1, \\ X \equiv r_2 \mod m_2, \\ \ldots, \\ X \equiv r_k \mod m_k, \end{cases} \tag{2}$$

Для восстановления X используются коэффициенты c_i, которые находятся по формуле:

$$c_i = \frac{M}{m_i} \cdot \left(\left(\frac{M}{m_i} \right)^{-1} \mod m_i \right), \tag{3}$$

где выражение $\left(\frac{M}{m_i} \right)^{-1} \mod m_i$ – модульный обратный элемент.

Представление текста в СОК

Текст кодируется путём отображения символов в целые числа [5]. Каждому символу s_j присваивается уникальный код c_j. Строка $S = s_1 s_2 \ldots s_L$ преобразуется в последовательность кодов $C = c_1, c_2, \ldots, c_L$. Для представления в СОК выбирается базис взаимно простых модулей $m_1, m_2, \ldots, m_k$, таких что их произведение M превышает максимальный код символа. Каждый код c_j независимо представляется кортежем остатков $(r_{j1}, r_{j2}, \ldots, r_{jk})$, где $r_{ji} = c_j \mod m_i$. Исходная строка преобразуется в матрицу остатков размером $L \times k$.

Например, строка "AB" в ASCII: $A=65$, $B=66$. Для базиса $m_1=3,\ m_2=5,\ m_3=7$:

$$A \to (65 \bmod 3=2,\ 65 \bmod 5=0,\ 65 \bmod 7=2) \to (2,\ 0,\ 2),$$

$$B \to (66 \bmod 3=0,\ 66 \bmod 5=1,\ 66 \bmod 7=3) \to (0,\ 1,\ 3).$$

Параллелизм в обработке

Ключевое преимущество СОК – присущий параллелизм. Арифметические операции над числами в СОК выполняются покомпонентно:

Сложение:

$$A+B \leftrightarrow \left(|a_1+b_1|_{m_1},\ |a_2+b_2|_{m_2},\ \ldots,\ |a_k+b_k|_{m_k}\right). \quad (4)$$

Умножение:

$$A \cdot B \leftrightarrow \left(|a_1 \cdot b_1|_{m_1},\ |a_2 \cdot b_2|_{m_2},\ \ldots,\ |a_k \cdot b_k|_{m_k}\right). \quad (5)$$

Вычитание, при $A \geq B$:

$$A-B \leftrightarrow \left(|a_1-b_1|_{m_1},\ |a_2-b_2|_{m_2},\ \ldots,\ |a_k-b_k|_{m_k}\right). \quad (6)$$

Корректность обеспечивается свойствами модулярной арифметики. Этот принцип распространяется на функции F, выразимые через арифметические операции над кодами символов. Вычисления можно выполнять независимо по каждому модулю m_i, используя только последовательности остатков $(r_{1i},\ r_{2i},\ \ldots,\ r_{Li})$. Результат восстанавливается из остатков по модулям с помощью КТО.

Пример: подсчёт символа 'A'. В канале $m_1=3$ считаем остатки 2, в $m_2=5$ – остатки 0, в $m_3=7$ – остатки 2. Совпадение по всем каналам гарантирует совпадение кода.

Сжатие данных

Объём памяти в битах для хранения L символов в СОК:

$$L \cdot \sum_{i=1}^{k} \lceil \log_2 m_i \rceil. \quad (7)$$

В 32-битном Unicode требуется $L \cdot 32$ бит. Если $\sum_{i=1}^{k} \lceil \log_2 m_i \rceil < 32$, достигается сжатие. Например, для модулей $m_1 = 251$, $m_2 = 253$, $m_3 = 255$: $M = 251 \cdot 253 \cdot 255 = 16193265 > 1114111$, $\lceil \log_2 251 \rceil = 8$, $\lceil \log_2 253 \rceil = 8$, $\lceil \log_2 255 \rceil = 8$. Итого 24 бита на символ против 32.

Однако сжатие ограничено:

1. Модули близкие к степеням двойки неэффективно используют битовые комбинации.

2. СОК обладает избыточностью для помехоустойчивости.

3. Стандартные алгоритмы сжатия устраняют избыточность СОК.

Перспективно сжатие числовых характеристик текста полиномиальные хеши, контрольные суммы, представленные меньшим набором модулей [6].

Помехоустойчивость

Избыточность СОК позволяет обнаруживать и корректировать ошибки. Искажение одного остатка в векторе делает его недопустимым – он не соответствует целому числу в $[0, M-1]$. Для коррекции ошибок в базис добавляют t избыточных взаимно простых контрольных модулей. Динамический диапазон расширяется до $M_{total} = M_{info} \cdot M_{red}$ [7, 8].

При приёме для каждого символа пытаются восстановить X' из $k+t$ остатков. Если $X' \geq M_{info}$ или не является кодом символа, фиксируется ошибка. Алгоритм коррекции: перебор гипотез об ошибке в одном остатке и

вычисление гипотетического X_{hyp}. Если X_{hyp} – допустимый код и удовлетворяет остальным остаткам, ошибка исправлена.

Пример: к базису $(3,\ 5,\ 7)$ добавлен $m_4 = 11$. Искажён вектор $(2,\ 3,\ 2,\ ?)$ вместо $(2,\ 0,\ 2,\ ?)$. Предполагая ошибку в m_2, находим $X_{hyp} = 65$ – допустимый символ 'A'.

Вычислительные преимущества

СОК эффективна для алгоритмов с арифметикой больших чисел, например, полиномиальных хешей [9]:

$$H(S) = \left(\sum_{i=1}^{L} c_{s_i} \cdot p^{L-i} \right) \bmod q. \tag{8}$$

Вычисления разбиваются на параллельные потоки по базису СОК $m_1,\ m_2,\ \ldots,\ m_k$, где $M > q$:

$$H_i(S) = \left(\sum_{j=1}^{L} r_{ji} \cdot \left| p^{L-j} \right|_{m_i} \right) \bmod m_i. \tag{9}$$

Итоговый $H(S)$ восстанавливается из $H_i(S)$ по КТО. Операции по малым модулям проще и быстрее, чем по большому q. Аналогичные преимущества – для скалярных произведений и свёрток.

Ограничения

1. Нелинейность лингвистических задач: морфологический анализ, синтаксический разбор, машинный перевод не имеют эффективных аналогов в СОК [10, 11].

2. Накладные расходы: кодирование/декодирование требует ресурсов. Восстановление числа по КТО имеет сложность $O(k^2)$ [12].

3. Сложность операций: сравнения, сортировки, неарифметические действия в СОК неэффективны.

4. Выбор базиса: требует взаимно простых модулей подходящего размера, близких к степеням двойки [13].

5. Избыточность: добавление контрольных модулей увеличивает объём данных [14].

6. Интеграция: сложность внедрения в существующие NLP-инфраструктуры.

7. Эффективность: операции по модулю на CPU/GPU могут уступать нативным.

Заключение

СОК предлагает теоретические преимущества для обработки текста: параллелизм арифметических операций, потенциальное сжатие и встроенная помехоустойчивость. Представление символов остатками позволяет независимо обрабатывать каналы модулей. СОК эффективна для вычислений с большими числами, таких как полиномиальные хеши.

Однако практическое применение ограничено неарифметической природой лингвистических задач, накладными расходами, сложностью нелинейных операций и проблемами интеграции. Современные методы нейронные сети, оптимизированные алгоритмы сжатия и коррекции часто эффективнее.

Перспективные направления: разработка специализированных сопроцессоров для СОК, гибридные подходы для вычисления хешей и контрольных сумм, эффективные алгоритмы сравнения и сортировки в СОК. Исследование СОК обогащает арсенал для специфических задач, требующих параллельной арифметики или повышенной надёжности.

Литература

1. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. М.: Сов. радио, 1968. 439 с.

2. Амербаев В.М. Теоретические основы машинной арифметики. Алма-Ата: Наука, 1976. 324 с.

3. Amos Omondi, Benjamin Premkumar. Residue Number Systems: Theory and Implementation. London: Imperial College Press, 2007. 296 p.

4. Ananda Mohan, P.V. Residue Number Systems. Cham: Springer International Publishing, 2016. 351 p.

5. Фомин С.В. Системы счисления. М.: Наука, 1987. 48 с.

6. Габидулин Э.М., Кшевецкий А.С., Колыбельников А.И. Защита информации. М.: МФТИ, 2011. 225 с.

7. Asmuth C., Bloom J. IEEE Transactions on Information Theory. 1983. Vol. 29, №2. pp. 208–210. DOI: 10.1109/TIT.1983.1056651

8. Червяков Н.И., Евдокимов А.А., Галушкин А.И., Лавриненко И.Н. и др. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. М.: ФИЗМАТЛИТ, 2012. 280 с.

9. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2003. 328 с.

10. Шиян В.И., Марков В.Н. Инженерный вестник Дона, 2025, №5. URL: ivdon.ru/ru/magazine/archive/n5y2025/10038

11. Егунова А.И., Комаров Р.С., Федосин С.А., Шибайкин С.Д., Жарков Р.А. Инженерный вестник Дона, 2025, №2. URL: ivdon.ru/ru/magazine/archive/n2y2025/9825

12. Финько О.А. Модулярная арифметика параллельных логических вычислений. М.: ИПУ РАН, 2003. 214 с.

13. Amir Sabbagh, Molahosseini, Leonel Seabra de Sousa, and Chip-Hong Chang (editors). Embedded Systems Design with Special Arithmetic and Number Systems. Cham: Springer International Publishing, 2017. 389 p.

14. Торгашев В.А. Система остаточных классов и надёжность ЦВМ. М.: Сов. радио, 1973. 118 с.

References

1. Akušskiy I.Ya., Yuditskiy D.I. Mashinnaya arifmetika v ostatochnykh klassakh [Machine arithmetic in residue classes]. Moskva: Sov. radio, 1968. 439 p.

2. Amerbaev V.M. Teoreticheskie osnovy mashinnoy arifmetiki [Theoretical foundations of machine arithmetic]. Alma-Ata: Nauka, 1976. 324 p.

3. Amos Omondi, Benjamin Premkumar. Residue Number Systems: Theory and Implementation. London: Imperial College Press, 2007. 296 p.

4. Ananda Mohan, P.V. Residue Number Systems. Cham: Springer International Publishing, 2016. 351 p.

5. Fomin S.V. Sistemy schisleniya [Numeral systems]. Moskva: Nauka, 1987. 48 p.

6. Gabidulin E.M., Kshvevetskiy A.S., Kolybel'nikov A.I. Zashchita informatsii [Information protection]. Moskva: MFTI, 2011. 225 p.

7. Asmuth C., Bloom J. IEEE Transactions on Information Theory. 1983. Vol. 29, №2. pp. 208–210. DOI: 10.1109/TIT.1983.1056651

8. Chervyakov N.I., Evdokimov A.A., Galushkin A.I., Lavrinenko I.N. i dr. Primenenie iskusstvennykh neironnykh setey i sistemy ostatochnykh klassov v kriptografii [Application of artificial neural networks and residue number systems in cryptography]. Moskva: FIZMATLIT, 2012. 280 p.

9. Vasilenko O.N. Teoretiko-chislovye algoritmy v kriptografii [Number-theoretic algorithms in cryptography]. Moskva: MTsNMO, 2003. 328 p.

10. Shiyan V.I., Markov V.N. Inzhenernyj vestnik Dona, 2025, №5. URL: ivdon.ru/ru/magazine/archive/n5y2025/10038

11. Egunova A.I., Komarov R.S., Fedosin S.A., Shibaykin S.D., Zharkov R.A. Inzhenernyj vestnik Dona, 2025, №2. URL: ivdon.ru/ru/magazine/archive/n2y2025/9825

12. Fin'ko O.A. Modulyarnaya arifmetika parallel'nykh logicheskikh vychisleniy [Modular arithmetic of parallel logical computations]. Moskva: IPU RAN, 2003. 214 p.

13. Amir Sabbagh, Molahosseini, Leonel Seabra de Sousa, and Chip-Hong Chang (editors). Embedded Systems Design with Special Arithmetic and Number Systems. Cham: Springer International Publishing, 2017. 389 p.

14. Torgashev V.A. Sistema ostatochnykh klassov i nadezhnost' TsVM [Residue class system and computer reliability]. Moskva: Sov. radio, 1973. 118 p.

Дата поступления: 9.07.2025

Дата публикации: 25.08.2025