УДК 004
EDN BJFSMN
https://www.doi.org/10.47813/mip.7.2026.20.3016

Оценка стратегий автоматической генерации TLA+ спецификаций с помощью LLM

С.Д. Боринских*, П.Д. Дробинцев

ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого», ул. Политехническая, д. 29, литера Б, г. Санкт-Петербург, 195251, Россия

*E-mail: ph.borinskikh@mail.ru

Аннотация. В данной работе исследуется автоматическая генерация формальных спецификаций TLA+ из требований, заданных на естественном языке, с использованием больших языковых моделей. Целью исследования является оценка влияния организации процесса генерации на корректность полученных формальных спецификаций. Проведено исследование на наборе из нескольких задач различной сложности с использованием пяти современных языковых моделей. Рассмотрены стратегии zero-shot, one-shot и few-shot, предварительная декомпозиция задачи, предоставление релевантного синтаксического контекста TLA+, а также их совместные комбинации. Сгенерированные спецификации проверялись средствами SANY и TLC, а обнаруженные ошибки в текстовом виде использовались для итеративного исправления не более чем за пять итераций. Результаты показывают, что синтаксический контекст наилучшим образом повышает качество первоначальной генерации: максимальные значения составили 82% SANY Pass@1 и 66% TLC Pass@1. При совместном использовании декомпозиции и синтаксического контекста качество в zero-shot достигло 78 и 62% соответственно. При этом все конфигурации достигли 100% TLC Pass@5. Показано, что увеличение объема контекста не всегда оправдывает рост использованных токенов и вычислительных затрат.

Ключевые слова: TLA+ спецификации, большие языковые модели, формальная верификация, стратегии промптинга, Chain-of-Thought, анализ ошибок SANY и TLC.

Для цитирования: Боринских, С. Д. Оценка стратегий автоматической генерации TLA+ спецификаций с помощью LLM / С. Д. Боринских, П. Д. Дробинцев // Модернизация, Инновации, Прогресс (MIP ENGINEERING-VII-2026): материалы VII Всероссийской (национальной) научной конференции (Красноярск, 11–12 сентября 2026 г.). – Красноярск: Красноярский краевой Дом науки и техники Российского Союза научных и инженерных общественных объединений, 2026. – Т. 20. – С. 137-150. – https://doi.org/10.47813/mip.7.2026.20.3016. – EDN: BJFSMN.

Evaluating strategies for automated TLA+ specification generation via LLM

S.D. Borinskikh*, P.D. Drobintsev

Peter the Great St. Petersburg Polytechnic University, Polytechnicheskaya Street, 29, St. Petersburg, 195251, Russia

*E-mail: ph.borinskikh@mail.ru

Abstract. This paper investigates the automatic generation of formal TLA+ specifications from requirements expressed in natural language using large language models. The study aims to evaluate the impact of the generation process organization on the correctness of generated specifications. An experimental study was conducted on a set of tasks of varying complexity using five state-of-the-art language models. The study considers zero-shot, one-shot, and few-shot prompting strategies, preliminary task decomposition, the provision of relevant TLA+ syntax context, and their combinations. The generated specifications were verified using SANY and TLC, while detected errors were used for iterative correction over no more than five iterations. The results show that providing syntax context most effectively improves the quality of initial generation, achieving maximum values of 82% SANY Pass@1 and 66% TLC Pass@1. When task decomposition and syntax context are combined, zero-shot generation achieves 78% SANY Pass@1 and 62% TLC Pass@1. At the same time, all configurations achieve 100% TLC Pass@5. The results demonstrate that increasing the amount of contextual information does not always justify the resulting increase in token usage and computational costs.

Keywords: TLA+ specifications, large language models, formal verification, prompting strategies, Chain-of-Thought, SANY and TLC error analysis.

For citation: Borinskikh, S. D. Evaluating strategies for automated TLA+ specification generation via LLM / S. D. Borinskikh, P. D. Drobintsev // Modernization, Innovations, Progress (MIP ENGINEERING-VII-2026): Proceedings of the VII All-Russian (National) Scientific Conference (Krasnoyarsk, September 11–12, 2026). – Krasnoyarsk: Krasnoyarsk Regional Science and Technology City

Hall of the Russian Union of Scientific and Engineering Public Associations, 2026. – Vol. 20. – P. 137-150. – https://doi.org/10.47813/mip.7.2026.20.3016. – EDN: BJFSMN.

1. Введение

Современное программное обеспечение становится все более сложным, что значительно повышает требования к методам его верификации, особенно для систем, в которых ошибки могут приводить к серьезным последствиям, и которые трудно обнаружить. Формальные методы позволяют дополнить тестирование ПО математически строгим описанием поведения системы.

Одним из языков формальной спецификации является TLA+ (Temporal Logic of Actions), разработанный Лесли Лампортом [1]. TLA+ позволяет описывать системы через состояния и допустимые переходы между ними, а инструменты SANY и TLC обеспечивают синтаксический анализ и модельную проверку спецификаций [2]. Однако разработка TLA+ требует специальных знаний в области формальных методов, что ограничивает практическое применение данного подхода.

Большие языковые модели (LLM) могут быть использованы для упрощения данного процесса за счет автоматического преобразования требований на естественном языке в формальные спецификации. Вместе с тем результаты последних исследований показывают, что получение корректной TLA+ спецификации остается сложной задачей. В работе Bisharat et al. [3] максимальная доля спецификаций, успешно прошедших SANY Pass@1, составила 26,6%, а TLC Pass@1 лишь 8,6%. Это указывает на существенный разрыв между синтаксической и поведенческой корректностью. В TLA+-Bench [4] также подчеркивается необходимость execution-grounded оценки с использованием TLC для обеспечения семантической корректности. Также существует работа TLA-Prover [5], в которой результат TLC используется как reward при обучении модели. Авторы достигают 30% TLC Pass@1 против 8,6% опубликованного baseline [3], что подтверждает ценность обратной связи от формального верификатора.

Таким образом, существующие исследования показывают, что основной проблемой остается низкая вероятность получения корректной спецификации с первой попытки. Это позволяет рассматривать не только выбор или обучение языковой модели, но и организацию процесса генерации и верификации, включая количество демонстрационных примеров, предварительную декомпозицию задачи отдельным шагом, предоставление релевантного синтаксического контекста TLA+ и итеративное исправление на основе обратной связи SANY

и TLC. Дополнительно возникает вопрос о соотношении качества получаемых спецификаций и затрат на генерацию и верификацию.

В отличие от работ, ориентированных преимущественно на сравнение LLM или обучение специализированных моделей, данная работа исследует влияние организации процесса генерации и формальных спецификаций TLA+ на качество и эффективность их получения.

2. Постановка задачи (Цель исследования)

В работе рассматривается задача автоматической генерации формальных спецификаций TLA+ по текстовому описанию системы на естественном языке с использованием больших языковых моделей (LLM).

Пусть R – текстовое описание требований к системе, M – используемая языковая модель, а C – дополнительный контекст, предоставляемый модели. В зависимости от экспериментальной стратегии контекст может включать примеры ранее решенных задач, описание синтаксиса TLA+, а также промежуточный план формализации системы.

Тогда задача генерации может быть представлена в общем виде (1):

$$S = M(R, C) \tag{1}$$

где S – сгенерированная TLA+ спецификация.

Полученная спецификация должна проходить два уровня проверки корректности. На первом уровне проверяется синтаксис спецификации инструментом SANY. На втором уровне проверяется соответствие спецификации заданным свойствам системы при помощи инструмента TLC. В случае если спецификация не проходит хотя бы одну из проверок, начинается итеративный процесс исправления по обратной связи от SANY и TLC (2):

$$S_{i+1} = M(R, C, S_i, E_i), \quad i \leq 5 \tag{2}$$

где S_i – спецификация на i-й итерации, а E_i – информация об ошибке из SANY или TLC.

Процесс повторяется до получения спецификации, успешно прошедшей все проверки, либо пока не будет достигнуто максимальное количество итераций. Количество итераций исправления ограничивается пятью.

Цель исследования – это оценка влияния стратегий построение промпта, предварительной декомпозиции задачи, предоставления релевантного контекста и итеративной формальной верификации на качество и эффективность генерации TLA+ спецификаций при помощи LLM.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Исследовать влияние стратегий zero-shot, one-shot и few-shot prompting на качество генерации TLA+ спецификаций.

2. Исследовать влияние декомпозиции задачи генерации посредством предварительного планирования формальной модели.

3. Исследовать влияние предоставления модели контекста синтаксиса TLA+ на корректность генерируемых спецификаций.

4. Исследовать совместное применение предварительного планирования и синтаксического контекста.

5. Исследовать возможность автономного исправления LLM спецификаций, не прошедших проверку SANY или TLC, на основе анализа сообщений об ошибках и Error Trace.

6. Исследовать соотношение между затратами и качеством сгенерированной спецификации.

На основании поставленных задач сформулированы следующие исследовательские вопросы:

1. **RQ1.** Как количество примеров в промпте (zero-shot, one-shot и few-shot) влияет на корректность генерируемых TLA+ спецификаций?

2. **RQ2.** Как предварительная декомпозиция задачи и предоставление релевантного синтаксического контекста TLA+ влияют на качество генерируемых спецификаций, и обеспечивает ли их совместное применение дополнительное улучшение?

3. **RQ3**. Насколько эффективно LLM способны автоматически исправлять ошибки SANY и TLC на основе текстовых сообщений об ошибках и Error Trace?

4. **RQ4.** Как соотносятся затраты на итеративную генерацию и формальную верификацию с качеством получаемых спецификаций?

В качестве основных критериев качества используются:

- **SANY pass@1** – доля спецификаций, успешно прошедших синтаксическую проверку с первой генерации;
- **TLC pass@1** – доля спецификаций, успешно прошедших модельную проверку с первой генерации;
- **SANY@5** – доля спецификаций, успешно прошедших SANY менее чем за пять итераций;
- **TLC@5** – доля спецификаций, успешно прошедших TLC менее чем за пять итераций;
- **Сред. SANY Pass # –** среднее значение итерации, на которой была впервые пройдена SANY проверка

- **Сред. TLC Pass # –** среднее значение итерации, на которой была впервые пройдена TLC проверка
- **количество итераций до успеха**;
- **количество использованных токенов**;
- **Pass Rate** – доля успешных проверок относительно их общего количества;
- **Токены / pass** – средний расход токенов на один успешный результат;
- **Токены / run** – средний расход токенов на одну попытку.

Таким образом, объектом исследования является процесс автоматической генерации, формальной верификации и итеративного исправления TLA+ спецификаций с использованием больших языковых моделей.

Предметом исследования является влияние стратегий формирования промпта, предварительной декомпозиции задачи, релевантного синтаксического контекста и обратной связи от инструментов формальной проверки на корректность, исправляемость и стоимость получаемых TLA+ спецификаций.

3. Методы и материалы исследования

3.1. Набор тестовых задач

Для оценки методов генерации формальных спецификаций был сформирован набор из 10 задач различной степени сложности. В качестве основы использовались задачи и примеры спецификаций TLA+, представленные в открытом репозитории tla_benchmark [6], которые содержит задачи из работы [4]. Выбор задач осуществлялся по Easy и Medium уровням сложности в соответствии с классификацией tla_benchmark.

Основной экспериментальный набор включает несколько задач каждого уровня сложности. Названия задач в соответствии с репозиторием [6] представлены в таблице 1.

Таблица 1. Названия задач в соответствии с репозиторием tla_benchmark.

Easy	Medium
1. AsynchInterface; 2. TCommit; 3. HourClock2; 4. DepthFirstTerminate; 5. Test46.	6. SimpleAllocator; 7. Barrier; 8. TokenRing; 9. DieHardTLA; 10. Clean.

Дополнительно были выделены три демонстрационные задачи, которые не включались в основной статистический набор:

- D1: Github1037 – Easy;
- D2: CoffeeCan100Beans – Medium;
- D3: EnvironmentController – Hard.

Демонстрационные задачи используются исключительно для формирования контекста при применении стратегий one-shot и few-shot prompting. Для каждой демонстрационной задачи модели предоставляется текстовое описание этой задачи и соответствующая TLA+ спецификация.

3.2. Исследуемые языковые модели

В исследовании рассматриваются пять современных больших языковых моделей, показывающих хорошие результаты в написании кода и агентском режиме работы:

1. Gemini 3.7 Flash;
2. GPT-5.6 Sol;
3. DeepSeek V4 Pro 0813;
4. Qwen 3.8 Max 0902;
5. GLM-5.3.

Для снижения влияния случайности генерации в экспериментах используется значение temperature = 0. Данный параметр снижает случайность выборки токенов, однако не гарантирует абсолютную детерминированность результатов. Дополнительно используется параметр reasoning_effort, который устанавливается в значение low.

3.3. Описание процесса и стратегий генерации спецификаций

В каждом исследовании изменяются три компонента контекста C, предоставляемого языковой модели: количество демонстрационных примеров, наличие предварительной декомпозиции задачи и наличие релевантного контекста с синтаксисом TLA+.

Первый изменяемый компонент контекста – это количество демонстрационных примеров. Рассматриваются три варианта формирования контекста:

1. zero-shot – в промпт не добавляется ни один пример;
2. one-shot – в промпт добавляется один пример решения задачи D1 из п. 3.2;
3. few-shot – в промпт добавляются все примеры решения задач D1-D3 из п. 3.2.

Второй компонент – это предварительная декомпозиция задачи, в рамках которой перед генерацией итоговой TLA+ спецификации модели предлагается сформировать структурированный план формализации. План содержит следующие элементы:

1. основные состояния системы;

2. переменные, необходимые для представления состояния;

3. начальное состояние системы;

4. переходы и действия, изменяющие состояние;

5. инварианты и другие свойства, которые должны выполняться.

Полученный план используется как промежуточное представление задачи, полученное отдельным шагом, и передается модели при последующем формировании TLA+ спецификации.

Третий компонент – это информация о синтаксисе TLA+ языка. Синтаксический контекст представляет собой набор блоков, описывающих основные конструкции языка. В состав справочника входят следующие категории:

1. структура TLA+ модуля;

2. объявления констант и переменных;

3. конструкции Init, действий и Next;

4. правила использования UNCHANGED;

5. инварианты;

6. основные логические, множественные и временные операторы;

7. конструкции для работы с функциями и условными выражениями;

8. правила подключения стандартных модулей через EXTENDS.

Важно, что синтаксический контекст не является полной документацией по языку TLA+, а представляет собой лишь компактный набор правил, используемых в блоьшинстве спецификаций.

На основании описанных компонентов формируются четыре основных эксперимента:

1. Генерация с различным числом примеров в промпте;

2. Генерация с различным числом примеров в промпте и предварительной декомпозицией;

3. Генерация с различным числом примеров в промпте и синтаксическим контекстом;

4. Генерация с различным числом примеров в промпте, предварительной декомпозицией и синтаксическим контекстом.

4. Полученные результаты

4.1. Результаты первого эксперимента

Первый эксперимент направлен на оценку качества генерации TLA+ спецификаций с различным количеством демонстрационных примеров в промпте. Сравнивались режимы zero-

shot, one-shot и few-shot без какого-либо дополнительного контекста. Результаты представлены в таблице 2.

Таблица 2. Результаты первого эксперимента. Агрегация по стратегиям промптинга.

Стратегия	SANY Pass@1	TLC Pass@1	SANY @5	TLC @5	Сред. SANY Pass #	Сред. TLC Pass #	Pass Rate	Токены / pass	Токены / run
Zero-shot	52%	42%	100%	100 %	3	3	53%	10 873	5 729
One-shot	**72%**	**56%**	100%	100 %	2	3	58%	10 454	6 026
Few-shot	70%	56%	100%	100 %	2	3	56%	16 173	9 027

Видно, что использование примеров повышает вероятность успешной генерации спецификации с первой попытки, как и для SANY, так и для TLC. Наилучшим оказался подход с one-shot, который по сравнению с zero-shot увеличивает SANY Pass@1 с 52 до 72%, а TLC Pass@1 с 42 до 56%. При этом переход от one-shot к few-shot практически не изменяет качество первой генерации, но увеличивает траты в среднем на почти 3000 токенов за попытку генерации.

В итоге, первый эксперимент показывает, что предоставление хотя бы одного примера спецификации в промпт без предварительного планирования и синтаксического контекста повышает вероятность успешной генерации, однако дальнейшее увеличение числа примеров не всегда дает прирост качества. Одновременно итеративная проверка и исправление позволяет существенно повысить итоговую успешность независимо от выбранной модели и количества демонстраций.

4.2. Результаты второго эксперимента

Второй эксперимент – это оценка качества генерации TLA+ спецификаций при использовании предварительной декомпозиции задачи. В данном эксперименте перед генерацией спецификации LLM дополнительно просили сформировать план формализации. Результаты представлены в таблице 3.

Таблица 3. Результаты второго эксперимента. Агрегация по стратегиям промптинга.

Стратегия	SANY Pass@1	TLC Pass@1	SANY @5	TLC @5	Сред. SANY Pass #	Сред. TLC Pass #	Pass Rate	Токены / pass	Токены / run
Zero-shot	70%	46%	100%	100%	3	3	48%	14,467	7,005
One-shot	72%	52%	100%	100%	3	3	53%	10,031	5,297
Few-shot	**74%**	**52%**	100%	100%	2	3	52%	14,963	7,822

Видно, что использование предварительного планирования выводит вероятность успешной генерации спецификации с первой попытки для всех количеств примеров в промпте примерно на один уровень для SANY, но слегка ухудшает результаты по TLC относительно первого эксперимента. Наилучшим в данном эксперименте оказался подход с few-shot.

Таким образом, наиболее значимый эффект декомпозиция дает при отсутствии демонстрационных примеров. Zero-shot с предварительным планированием достигает уровня SANY Pass@1, сопоставимого с one-shot и few-shot без декомпозиции, что указывает на способность структурированного плана частично компенсировать отсутствие примеров. При этом улучшение синтаксической корректности не сопровождается сопоставимым приростом TLC Pass@1.

4.3. Результаты третьего эксперимента

Третий эксперимент – это оценка качества генерации TLA+ спецификаций при использовании дополнительного контекста с синтаксисом TLA. В данном эксперименте в промпт добавляется контекст, содержащий релевантные для задач генерации правила синтаксиса TLA+. Результаты представлены в таблице 4.

Таблица 4. Результаты третьего эксперимента. Агрегация по стратегиям промптинга.

Стратегия	SANY Pass@1	TLC Pass@1	SANY @5	TLC @5	Сред. SANY Pass #	Сред. TLC Pass #	Pass Rate	Токены / pass	Токены / run
Zero-shot	70%	54%	100%	100%	3	3	52%	14,071	7 335
One-shot	76%	58%	100%	100%	2	3	53%	13,745	7 250
Few-shot	**82%**	**66%**	100%	100%	2	3	60%	15,650	9 352

Видно, что использование контекста с синтаксисом TLA+ значительно повышает вероятность успешной генерации спецификации с первой попытки, как и для SANY, так и для

TLC. Наилучшим оказался подход few-shot с рекордными SANY Pass@1 82% и TLC Pass@1 66%. Более того, он обеспечивает наивысший Pass Rate относительно трех экспериментов.

Таким образом, синтаксический контекст демонстрирует наиболее выраженный положительный эффект среди рассмотренных до этого стратегий, особенно при совместном использовании с few-shot.

4.4. Результаты четвертого эксперимента

Четвертый эксперимент исследует совместное использование предварительной декомпозиции задачи и синтаксического контекста TLA+. В отличие от предыдущих экспериментов, языковой модели одновременно предоставлялись структурированный план формализации и релевантные элементы синтаксиса TLA+. Результаты представлены в таблице 5.

Таблица 5. Результаты четвертого эксперимента. Агрегация по стратегиям промптинга.

Стратегия	SANY Pass@1	TLC Pass@1	SANY @5	TLC @5	Сред. SANY Pass #	Сред. TLC Pass #	Pass Rate	Токены / pass	Токены / run
Zero-shot	**78%**	**62%**	100%	100%	#3	#3	51%	11,352	5 799
One-shot	74%	58%	100%	100%	#3	#3	48%	11,823	5 660
Few-shot	72%	54%	100%	100%	#3	#3	53%	16,421	8 735

Видно, что совместное использование декомпозиции и синтаксического контекста дает наилучший результат в режиме zero-shot, где SANY Pass@1 достигает 78%, а TLC Pass@1 62%. Однако при добавлении демонстрационных примеров наблюдается обратная тенденция. В one-shot и few-shot показатели снижаются относительно их лучших показателей в других экспериментах.

Этот результат показывает, что влияние различных компонентов промптинга не является аддитивным, т.е предоставление модели одновременно большого количества структурированной информации и демонстрационных примеров не гарантирует повышения качества. Более того, данная стратегия иногда может привести и к снижению.

5. Выводы

В работе исследовано влияние различных стратегий организации генерации и формальных спецификаций TLA+ с использованием LLM. На основе проведенных

экспериментов получены ответы на поставленные исследовательские вопросы. Объединенные результаты сравнения стратегий промптинга представлены в таблице 6.

Таблица 6. Сравнение экспериментов по стратегии промптинга.

Эксперимент	Стратегия промптинга	Zero-shot SANY Pass@1 / TLC Pass @1	One-shot SANY Pass@1 / TLC Pass @1	Few-shot SANY Pass@1 / TLC Pass @1
1	Examples	52 / 42	72 / 56	70 / 56
2	Examples+ Decomposition	70 / 46	72 / 52	74 / 52
3	Examples+ Syntax context	70 / 54	**76 / 58**	**82 / 66**
4	Examples+ Decomposition + Syntax context	**78 / 62**	74 / 58	72 / 54

1. RQ1. Как количество примеров в промпте влияет на корректность генерируемых TLA+ спецификаций?

Использование хотя бы одного демонстрационного примера повышает качество генерации. В первом эксперименте переход от zero-shot к one-shot увеличивает SANY Pass@1 с 52 до 72%, а TLC Pass@1 с 42 до 56%. При дальнейшем увеличении количества примеров до трех сопоставимого прироста не наблюдается. Таким образом, на рассматриваемом наборе задач one-shot обеспечивает более эффективное использование демонстрационных примеров, тогда как увеличение их количества без дополнительного контекста не приводит к монотонному повышению качества.

2. RQ2. Как предварительная декомпозиция задачи и предоставление синтаксического контекста TLA+ влияют на качество спецификаций, и обеспечивает ли их совместное применение дополнительное улучшение?

Оба подхода повышают качество первоначальной генерации, однако их влияние зависит от количества примеров в промпте. Предварительная декомпозиция наиболее эффективна в zero-shot, где повышает SANY Pass@1 с 52 до 70%. В свою очередь, наличие контекста с синтаксисом TLA+ демонстрирует более устойчивый эффект: достигаются максимальные значения SANY Pass@1 82% и TLC Pass@1 66% в few-shot. Интересным наблюдением является то, что совместное применение декомпозиции и синтаксического контекста дает лучший результат для zero-shot: 78 и 62% соответственно, однако в остальных подходах данная стратегия не улучшает качество, либо незначительно его снижает.

Следовательно, эффект компонентов контекста не является аддитивным, а оптимальная конфигурация зависит от баланса использования исследуемых методов.

3. RQ3. Насколько эффективно LLM способны автоматически исправлять ошибки SANY и TLC на основе обратной связи?

Итеративное исправление спецификаций на основе ошибок из SANY и TLC существенно повышает итоговое качество генерации. Во всех исследованных конфигурациях SANY Pass@5 и TLC Pass@5 достигают 100%, несмотря на существенно более низкие показатели успешной генерации с первой попытки. Это показывает, что текстовые сообщения об ошибках SANY и TLC, а также Error Trace, являются эффективным механизмом обратной связи для последовательного исправления сгенерированных спецификаций.

4. RQ4. Как соотносятся затраты на итеративную генерацию и формальную верификацию с качеством получаемых спецификаций?

Таблица 7. Сравнение соотношения затрат и качества эксперимента.

Эксперимент	Стратегия	SANY @1	TLC @1	Tokens / Pass	Tokens / Run	Total Tokens	Total Time, c
1	Zero-shot	52%	42%	10 873	5 729	532 756	5 397,4
	One-shot	72%	56%	10 454	6 026	512 251	5 027,6
	Few-shot	70%	56%	16 173	9 027	776 308	5 636,7
2	Zero-shot	70%	46%	14 467	7 005	665 468	2 556,5
	One-shot	72%	52%	10 031	5 297	471 467	3 371,1
	Few-shot	74%	52%	14 963	7 822	688 300	3 558,2
3	Zero-shot	70%	54%	14 071	7 335	689 484	7 026,8
	One-shot	76%	58%	13 745	7 250	659 755	5 392,1
	Few-shot	**82%**	**66%**	**15 650**	**9 352**	**766 873**	**4 955,3**
4	**Zero-shot**	**78%**	**62%**	**11 352**	**5 799**	**533 533**	**2 804,6**
	One-shot	74%	58%	11 823	5 660	532 053	2 243,4
	Few-shot	72%	54%	16 421	8 735	821 051	3 892,1

Из таблицы 7 видно, что дополнительный контекст увеличивает затраты на потребляемые токены, однако рост затрат не всегда приводит к повышению качества.

Наилучшие показатели получил few-shot в эксперименте 3. Данный эксперимент обеспечивает максимальный SANY Pass@1 82% TLC Pass@1 66%, но требует в среднем 9352 токенов на попытку генерации и 766 873 токена суммарно на все задачи. В эксперименте 4 few-shot требует уже 821 051 токенов, при этом качество снижается до 72 и 54% соответственно.

Наилучшее сочетание качества, затраченных токенов и временных затрат наблюдается для zero-shot в эксперименте 4. В данном эксперименте были достигнуты SANY Pass@1 78%, TLC Pass@1 62%, было потрачено в среднем 5799 токенов на запуск и 533 533 токена суммарно, а генерация спецификаций для всех задач заняла 2804,6 с.

Таким образом, увеличение объема промпта само по себе не является оптимальной стратегией. Более эффективным оказывается подбор структуры контекста: синтаксический контекст и декомпозиция позволяют повысить качество первоначальной генерации, а итеративная обратная связь от SANY и TLC обеспечивает достижение 100% успешности в пределах пяти итераций.

Список литературы

1. Lamport, L. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers / L. Lamport. – Boston: Addison-Wesley, 2002. – 364 p.
2. Lamport, L. TLA+ Tools / L. Lamport. – 2022. – URL: https://lamport.azurewebsites.net/tla/tools.html (дата обращения: 13.09.2026).
3. Can LLMs Write Correct TLA+ Specifications? Evaluating Natural-Language-to-TLA+ Generation / A. Bisharat, B. Ortiz, E. Spencer [et al.] // Proceedings of the 21st International Conference on Software Technologies (ICSOFT 2026). – Setubal: SCITEPRESS, 2026. – P. 39–50. – https://doi.org/10.5220/0015070400004088.
4. TLA+-Bench: An Execution-Grounded Benchmark and Dataset for Natural-Language to TLA+ Specification Generation / A. Bisharat, E. Spencer, B. Ortiz [et al.]. – 2026. – arXiv:2607.23425. – https://doi.org/10.48550/arXiv.2607.23425. – URL: https://arxiv.org/abs/2607.23425 (дата обращения: 13.09.2026).
5. TLA-Prover: Verifiable TLA+ Specification Synthesis via Preference-Optimized Low-Rank Adaptation / E. Spencer, A. Bisharat, B. Ortiz [et al.] // Proceedings of the 21st International Conference on Software Technologies (ICSOFT 2026). – Setubal: SCITEPRESS, 2026. – P. 627-636. – https://doi.org/10.5220/0015234600004088.

6. tla_benchmark: repository / GitHub. – URL: https://github.com/LUC-AI4FM/tla_benchmark/tree/dataset-release (дата обращения: 13.09.2026).