

CONTINUITY INSTRUMENT

Инструмент непрерывности CI

Назначение, данные, применение и границы измерения

Иван Котов / Ivan Kotov

Независимый исследователь

Брюссель, Бельгия

ORCID 0009-0009-6002-9845

ivankotov.eu

CI 1.0.0 | Academic publication edition 1

25 октября 2026

© 2026 Иван Котов. Научный текст: CC BY 4.0. Исходная версия CI 1.0.0 сохранена без изменений.

Автор: Иван Котов. Сопроводительное издание academic-pdf-1 к неизменённой сборке _02. DOI для CI на момент подготовки не присвоен; эмпирическая квалификация не установлена.

Содержание

Аннотация	3
1. Какую проблему решает инструмент	3
2. Где CI находится в рабочем процессе	4
3. С какими данными работает версия 1.0.0	5
4. Нормативный допуск и программная проверка	6
5. Почему результат представлен вектором	7
6. Учебный пример: шестьдесят минут наблюдения	7
7. Пауза, пробел, воспроизведение и новая ветвь	9
8. Четыре независимые линии изменений	10
9. Неопределённость и граница допустимого вывода	10
10. Профиль распознавания и связи с исследовательским корпусом	11
11. Практический вход и выход	12
12. Какие результаты уже имеются	12
13. Ограничения выпуска и условия использования	13
14. Практические применения и дальнейшая проверка	14
Литература и материалы	14

Аннотация

Continuity Instrument (CI) предложен как исследовательская спецификация для описания того, что ограниченный набор свидетельств позволяет установить о продолжительном участии и управляемых изменениях системы искусственного интеллекта. Инструмент представляет результат вектором: подтверждённые интервалы участия, известные паузы, необъяснённые промежутки, события воспроизведения и разветвления, изменения отдельных линий памяти, правил управления, свидетельствования и полномочий. Статья объясняет практическое назначение этого представления, состав входных данных и место CI между первичными журналами и последующим исследовательским заключением. Отдельно рассматривается эталонная реализация v1.0.0: детерминированный преобразователь предоставленных JSON-записей, который проверяет структурные ограничения и рассчитывает интервальное покрытие. Он не собирает первичные свидетельства и не устанавливает их достоверность. Учебный пример показывает, почему перекрывающиеся интервалы нельзя суммировать напрямую, а отсутствие данных нельзя автоматически считать паузой. Разбираются возможные применения при замене модели, восстановлении состояния и наблюдении нескольких ветвей, а также ограничения воспроизводимости и известная ошибка обработки Unicode. Публикуемый материал описывает метод и программное преобразование; эмпирическая квалификация измерения непрерывности конкретной реальной системы остаётся неустановленной.

Ключевые слова: Continuity Instrument; долговременные системы ИИ; наблюдаемая непрерывность; происхождение данных; интервальное покрытие; неопределённость; линии преемственности; ограничение выводов.

Статус текста. Это сопроводительный комментарий к исходной сборке CI v1.0.0 _02. Он поясняет её устройство и ограничения, не заменяя нормативные файлы, не исправляя программный код и не изменяя сохранённые результаты проверок. Аббревиатура CI далее означает **Continuity Instrument**, а не continuous integration - непрерывную интеграцию программного обеспечения.

1. Какую проблему решает инструмент

У системы ИИ, с которой работают длительное время, постепенно появляется история: сообщения, файлы памяти, решения оператора, смены моделей, остановки процессов, резервные копии и восстановленные состояния. Наличие такой истории позволяет задавать содержательные вопросы. В какие периоды система действительно участвовала в работе? Какие сведения сохранились после замены компонента? Какие правила действовали? Кто наблюдал переход и на каком основании новое исполнение получило доступ к данным? Эти вопросы требуют разных свидетельств.

Обычная фраза «система работает три года» объединяет слишком много возможных ситуаций. За три календарных года могли смениться несколько моделей и устройств; процесс мог запускаться эпизодически; журнал мог содержать большие пропуски; архив переписки мог переноситься между независимыми экземплярами. Дата

первого файла подтверждает существование файла к определённому моменту. Для утверждения о длительности участия нужна дополнительная информация о самой деятельности и границах наблюдения.

CI задаёт способ сохранять эти различия в отчёте. Его задача - сделать вывод соразмерным данным: показать подтверждённые участки, отдельно оставить неизвестное, обозначить ветвь и раскрыть основания интерпретации. Такая дисциплина нужна и при положительном результате, и при неудачной проверке. Отсутствие достаточных свидетельств должно оставаться видимым результатом анализа, а не исчезать внутри убедительного пересказа. [1, 2]

Важна именно ограниченность вопроса. CI не устанавливает сознание, личность или юридический статус системы. Он также не доказывает, что после переноса памяти продолжает существовать та же индивидуальная сущность. Предложенный инструмент помогает описать наблюдаемую историю и условия её реконструкции, чтобы эти более сильные вопросы можно было обсуждать на явном основании.

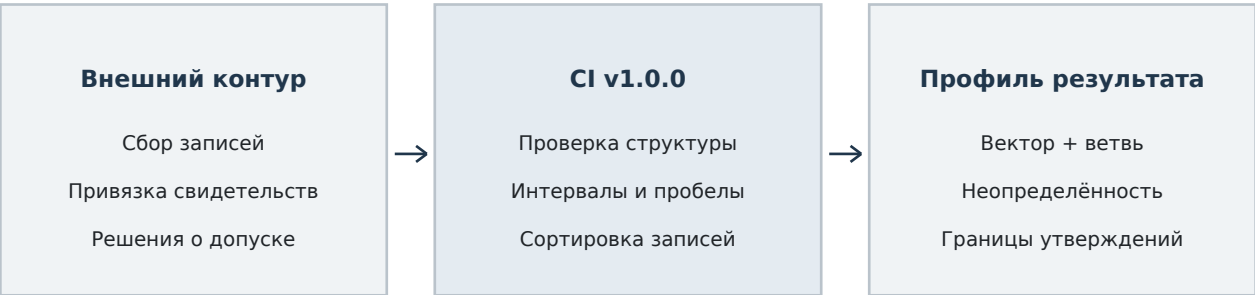
2. Где CI находится в рабочем процессе

Спецификация различает три уровня: наблюдения и свидетельства, реконструкцию непрерывности и допустимые выводы. На первом уровне находятся исходные записи и решения об их допуске. На втором из допущенных данных собираются интервалы, события, ветви и линии изменений. На третьем формулируется утверждение, ограниченное качеством и полнотой предыдущих уровней. [1, разд. 1]

Текущая программа `REFERENCE_IMPLEMENTATION/ci_profile.py` обслуживает ограниченную часть второго уровня. Она принимает подготовленный JSON, проверяет его структуру и определённые отношения между полями, затем формирует результат по фиксированным правилам. У неё нет встроенного подключения к Ollama, облачному API, базе переписки, файловому журналу или системе мониторинга. Она не вызывает модель и не обращается к сети. [2]

Практическое подключение поэтому требует внешней подготовки данных. Например, оператор может извлечь события запуска из имеющегося журнала процессов, связать сообщения с конкретной сессией и моделью, сопоставить версии файлов памяти и отдельно зарегистрировать решения о полномочиях. Затем эти сведения нормализуются в формат CI. Такая подготовка может выполняться вручную или существующими средствами обработки журналов; готовый универсальный импортёр данным выпуском не предоставляется.

Если для извлечения событий используется языковая модель, её разметка остаётся предложением интерпретации. Дословная ссылка на запись, связь с исходными байтами и правило допуска должны проверяться отдельно. Присвоенная моделью метка `admitted` сама по себе не делает сообщение достоверным. Полезность CI зависит от качества этого внешнего измерительного процесса не меньше, чем от правильности интервальной арифметики.



Допуск и истинность свидетельств не вычисляются преобразователем.
Распознавание и предоставление полномочий требуют отдельных процедур.

Рисунок 1. Граница текущего преобразователя и внешних процедур.

3. С какими данными работает версия 1.0.0

Входной объект описывает один объявленный предмет наблюдения, одну ветвь и одно окно времени. Поля `subject_ref`, `branch_ref` и `profile_id` служат организационными идентификаторами. Строка с именем системы не удостоверяет её индивидуальность; одинаковое имя в двух файлах не устанавливает общую историю. Версия контракта указывается как `ci-profile-input/v1`. [2, 3]

Группа данных	Основное содержание	Назначение
Область наблюдения	Идентификаторы, <code>time_axis</code> , <code>observation_window</code>	Определяет, о какой ветви и каком времени идёт речь
Свидетельства и интервалы	<code>witnesses</code> , <code>verified_intervals</code> , <code>known_pauses</code>	Разделяет наблюдения, участие и подтверждённые паузы
Изменения	<code>events</code> , <code>lineages</code> , <code>irreversible_transitions</code>	Сохраняет переходы, связи состояний и область их значения
Ограничения вывода	<code>uncertainty</code> , <code>claim_ceiling</code>	Делает видимыми неизвестное и запрещённые интерпретации

Время задаётся целыми отсчётами на объявленной оси. Поля `unit`, `origin` и `basis` описывают единицу, начало и основание отсчёта. Программа не считывает текущее время и не устанавливает соответствие между разными часами. Если источники используют разные временные шкалы, согласование выполняется до расчёта с сохранением сведений о точности и возможном смещении.

Запись интервала содержит границы, предикат участия, ссылки на свидетельства, контекст наблюдателей и полномочий, статус допуска, признак пригодности для расчёта длительности и неопределённость. Предикат участия должен объяснять, какое состояние считалось участием: выполнение определённой работы, подтверждённый обмен с оператором или иное заранее выбранное условие. У разных исследований эти условия могут различаться; одинаковая длительность тогда ещё не делает результаты сопоставимыми.

Ссылки `evidence_refs` и другие ссылочные строки текущий преобразователь не разыменовывает. Он не скачивает объект, не сопоставляет его хеш с указанным значением и не проверяет, действительно ли наблюдатель видел событие. Связь строки с первичным свидетельством должна существовать во внешнем наборе данных и оставаться доступной для соответствующей проверки. Это принципиальная граница между форматом отчёта и процедурой получения достоверных наблюдений.

4. Нормативный допуск и программная проверка

Нормативная спецификация требует достаточно данных для проверки идентичности источника, времени, предмета и ветви наблюдения, свидетеля и контекста полномочий, смысла события и допустимого объёма вывода. Для исходного объекта предпочтительны неизменяемый идентификатор или криптографическая привязка; одного названия файла недостаточно. Временная запись должна различать время события, получения, передачи и проверки, если эти моменты различаются. [1, разд. 3-4]

Это требования к процедуре исследования. Текущая реализация проверяет более узкий программный контракт: типы и наборы полей, допустимые значения, порядок временных границ, нахождение внутри окна, уникальность локальных идентификаторов и некоторые противоречия. Например, она отвергает интервал, которому разрешено добавлять длительность без статуса `admitted`, и пересечение учитываемого участия с известной паузой. [2]

Особенно существенна связь наблюдения с длительностью. Два успешных контрольных запроса в начале и конце часа подтверждают два ответа. Чтобы признать весь час участием, необходимо отдельное основание: непрерывная запись измеряемого состояния или правило наблюдения, действительно поддерживающее такое заключение. Сам выбор удобного интервала в JSON этого основания не создаёт. Аналогично, запись о сохранении файла памяти свидетельствует о конкретной операции хранения, но без дополнительных данных не подтверждает, что следующее исполнение прочитало файл и использовало нужное изменение. Здесь CI заставляет явно назвать проверяемый предикат, хотя проверку его выполнения текущий код не осуществляет.

Следовательно, успешный выход программы означает, что предоставленные данные прошли реализованные структурные и реляционные проверки. Он не означает, что все требования нормативного допуска были независимо выполнены. Реальный исследователь должен отдельно объяснить, кто допустил записи, по какому правилу и где находятся основания этого решения.

Полезно заранее различать три состояния записи: `admitted`, `excluded` и `unresolved`. Исключение означает конкретное решение не использовать запись в заявленном расчёте. Неопределённый статус оставляет вопрос открытым. Сведение обоих состояний к отсутствию строки лишило бы исследователя сведений о том, что рассматривалось и почему не вошло в результат. В выходе эталонной программы соответствующие счётчики предусмотрены для свидетелей и интервалов; они не являются автоматическим аудитом всех внешних источников.

5. Почему результат представлен вектором

Одна длительность не описывает историю. Две системы могут иметь одинаковое число подтверждённых часов, но отличаться размером пробелов, количеством восстановлений, сохранностью памяти и действующими полномочиями. Поэтому основным результатом CI служит вектор, называемый `evidence-bound continuity age` или `formation depth`. В данном контексте эти названия обозначают структурированное описание, а не шкалу зрелости. [1, 4]

Группа координат	Обозначения	Что остаётся различным
Границы и покрытие	<code>W_first</code> , <code>W_last</code> , <code>I_verified</code> , <code>D_verified</code>	Крайние допущенные свидетельства, интервалы и их суммарное покрытие
Неучастие и неизвестное	<code>P_known</code> , <code>G_unexplained</code>	Подтверждённая пауза и отсутствие достаточного объяснения
События и преемственность	<code>E_*</code> , четыре <code>L_*</code> , <code>T_irreversible</code>	Типы переходов, независимые линии и ограниченная необратимость
Неопределённость	<code>U</code>	Неполнота и ограничения всех предыдущих групп

Крайние свидетельства показывают первый и последний допущенные ориентиры внутри области исследования. Разность их времён не используется как возраст непрерывного существования. Между двумя наблюдениями могут находиться как хорошо подтверждённые интервалы работы, так и длительное отсутствие данных.

В текущем `reference profile` координата `I_verified` содержит записи, одновременно допущенные и помеченные `eligible_for_duration=true`. Допущенный интервал с отрицательным признаком пригодности учитывается в соответствующем счётчике, но не входит в `I_verified`, не увеличивает `D_verified` и не закрывает вычисляемый пробел. Это уточнение важно: общая спецификация допускает более широкое обсуждение наблюдений, тогда как конкретная реализация фиксирует именно такую политику расчёта. [2, 12]

При вычислении длительности объединяется геометрическое покрытие интервалов. Сами исходные пригодные записи сохраняются в векторе вместе с предикатами, ссылками и неопределённостью. Поэтому устранение двойного счёта не должно пониматься как стирание различий между основаниями наблюдений.

6. Учебный пример: шестьдесят минут наблюдения

Рассмотрим условное окно `[0, 60)` минут. Это учебная конструкция, не результат наблюдения Эстер или иной реальной системы. Предположим, что три интервала прошли внешнюю процедуру допуска и признаны пригодными для расчёта. Отдельное свидетельство подтверждает одну паузу. Для остального времени достаточной классификации нет.

Запись	Интервал	Заявленное значение
Участие А	[0, 20)	Подтверждено условие участия
Участие В	[10, 30)	Подтверждено то же измеряемое условие
Пауза	[30, 40)	Подтверждено отсутствие участия
Участие С	[50, 60)	Подтверждено условие участия

Если просто сложить длительности А, В и С, получится 50 минут. Однако отрезок от 10 до 20 покрыт двумя записями. Его нельзя учитывать дважды. Объединение А и В даёт [0, 30), то есть 30 минут, а С добавляет ещё 10. Получаем $D_{\text{verified}} = 40$ минут.

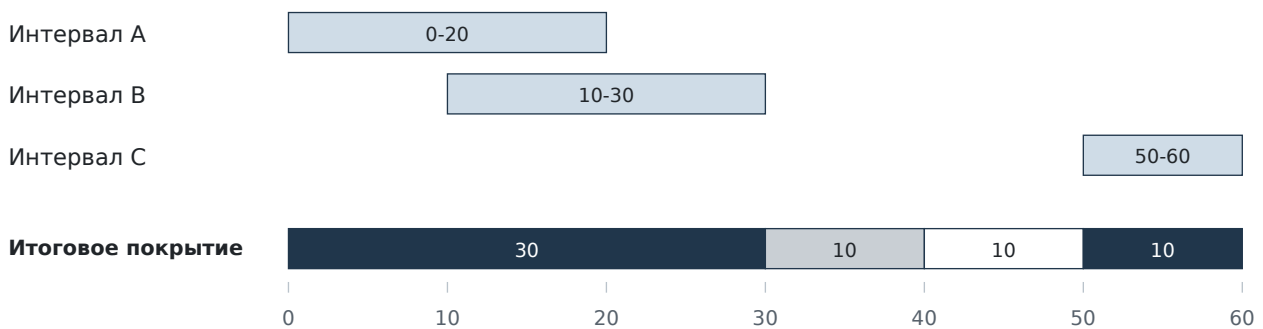
Известная пауза занимает 10 минут. После вычитания объединённого участия и паузы из окна остаётся [40, 50): 10 минут необъяснённого промежутка. Итоговая временная картина - 40 минут подтверждённого участия, 10 минут подтверждённой паузы и 10 минут неизвестного состояния. Все 60 минут распределены по этим трём категориям без двойного счёта.

Для данного окна можно дополнительно вычислить долю покрытия участием: $40/60$, или примерно 66,7 %. Это доля времени, удовлетворяющего объявленному и допущенному условию в учебном наборе. Она не является процентом сохранённой личности, вероятностью идентичности или универсальным показателем качества системы. Любое такое расширение потребовало бы самостоятельного определения и иных свидетельств.

Смена интервала участия паузой в точке 30 не создаёт противоречия: используется соглашение $[start, end)$, при котором правая граница не входит в предыдущий интервал. Напротив, пауза [25, 40) пересеклась бы с подтверждённым участием до 30 и была бы отвергнута реализацией как противоречивый вход. Решение о том, какая исходная запись ошибочна, программа при этом не принимает.

Координаты w_{first} и w_{last} нельзя заполнить простым переписыванием чисел 0 и 60 из этой арифметики. Они выбираются из отдельных допущенных записей свидетелей. Аналогично, восстановление процесса на минуте 50 требует собственной записи события; существование интервала С само по себе не объясняет, как система вернулась к участию.

Учебное окно: 60 минут



Подтверждено 40 | Пауза 10 | Неизвестно 10

Перекрывание считается один раз. Это не эмпирический результат.

Рисунок 2. Учебная интервальная реконструкция. Все значения заданы для иллюстрации.

7. Пауза, пробел, воспроизведение и новая ветвь

Пауза означает, что для некоторого промежутка имеется положительное свидетельство неактивного, приостановленного или недоступного состояния. Пробел означает, что доступные данные не позволяют установить ни участие, ни классифицированную паузу. Отсутствие строки в журнале может иметь много причин; CI сохраняет неопределённость, пока причины не установлены. [4, 5]

Событие `replay` обозначает новое исполнение или предъявление ранее записанного материала. Оно может породить собственные наблюдения, но не добавляет время к исходному исполнению задним числом. `Restore` описывает загрузку прежнего состояния в целевой контекст. `Cold_wake` указывает начало или возобновление участия после заявленного неактивного состояния. Ни один из этих типов не превращает предшествующий неизвестный промежуток в подтверждённую работу.

При `fork` появляются различные ветви с явным отношением к предшествующему состоянию. После разветвления свидетельства должны относиться к соответствующему потомку. Если два исполнения используют одну копию памяти, совпадение содержимого не разрешает сложить их время в одну бесспорную биографию. Для каждого наблюдаемого продолжения необходимо определить собственную область расчёта.

Эталонный преобразователь сохраняет переданные ссылки на исходную и целевую ветви, распределяет события по типам и сортирует записи. Он не обнаруживает скрытые разветвления и не удостоверяет родство ветвей. Полнота графа, действительность перехода и принадлежность события конкретному исполнению устанавливаются внешней процедурой. Такая организация позволяет применять формат к разным средам, но оставляет ответственность за происхождение данных явной.

8. Четыре независимые линии изменений

Память, правила управления, свидетели и полномочия могут изменяться по разным траекториям. CI поэтому хранит четыре линии отдельно. Каждая содержит связи между состояниями и классификацию перехода: сохранение с доказательством, изменение с привязанной записью, подтверждённый разрыв, конфликт либо неизвестность. [1, разд. 7; 5]

Линия	Основной вопрос	Пример различного изменения
L_memory	Что произошло с сохраняемыми сведениями?	Архив перенесён, частично потерян или заменён
L_governance	Какие правила определяли допустимое поведение?	Принята новая версия ограничений
L_witness	Кто и в какой области подтверждал события?	Изменился источник наблюдений или независимость проверки
L_authority	На каком основании разрешалось действие?	Доступ выдан, ограничен, отозван или передан

Предметная аналогия - техническое обслуживание агрегата. Замена блока может сохранить журнал наработки и историю неисправностей, изменить применяемую инструкцию, потребовать подписи другого специалиста и нового допуска к эксплуатации. Серийный номер, содержимое журнала, исправность заменённого узла и право подписать выпуск отвечают на разные вопросы. Совпадение одного из них не позволяет автоматически восстановить остальные. В системе ИИ столь же важно различать сохранённую память, действующие правила, наблюдателя перехода и полномочия нового процесса.

Координата необратимых переходов требует дополнительной осторожности. Из того, что отмена операции пока не найдена, не следует её необратимость. Необходимо назвать область, правило решения и свидетельства. Например, техническое удаление данных и окончательность управленческого решения имеют разные основания. Текущая программа сохраняет переданные `score`, `decision_rule` и ссылки; самостоятельной проверки необратимости в ней нет.

9. Неопределённость и граница допустимого вывода

Неопределённость относится к источникам, часам, классификации, ветвям, линиям изменений, свидетелям, полномочиям и статусу проверки. Она хранится структурированно, чтобы пользователь отчёта видел конкретную недостающую часть. Значение `not_assessed` означает отсутствие оценки. Оно не должно превращаться в ноль, который читатель мог бы принять за установленное отсутствие проблемы. [4]

Поле `claim_ceiling` задаёт разрешённые утверждения, их основания и запрещённые интерпретации. Реализация проверяет наличие обязательных запретов, но не устанавливает истинность произвольных формулировок в `allowed_claims`. Если автор

входа напишет необоснованное утверждение, синтаксическая корректность JSON не сделает его научно допустимым. Проверка смысла остаётся задачей исследователя и рецензента. [2]

Для воспроизводимости программа выдаёт `formation_depth_vector_sha256`. Этот хеш вычисляется только по каноническому JSON объекта `formation_depth_vector` в соответствии с правилами пакета. Поля верхнего уровня, включая `subject_ref`, `branch_ref`, `profile_id` и `claim_ceiling`, находятся вне хешируемого объекта. Поэтому совпадение этого значения подтверждает совпадение вектора при заданной сериализации, но не идентичность полного профиля или предмета наблюдения. Привязка полного файла выполняется отдельно.

Такое различие полезно при передаче результатов. Получатель может проверить вычисление и сохранность представления, затем отдельно оценить источники, ограничения доступа и основания интерпретации. Хеш связывает байты; решение о значении этих байтов требует содержательной проверки.

10. Профиль распознавания и связи с исследовательским корпусом

В пакете описан `recognition profile` - переносимое представление наблюдений, линий изменений, оснований и неопределённости для ограниченного сопоставления между контекстами. Возможные нормативные исходы включают распознавание в пределах профиля, недостаточность свидетельств, конфликт, устаревание и неприменимость. Среди них нет результата `same_entity`. Текущий `reference transformer` не реализует самостоятельный механизм такого сравнения. [6]

Связь с `Temporal AI Presence` задаёт временную перспективу вопроса: какие наблюдаемые отношения сохраняются в истории системы. Ссылки на `Beacon` и `Entity vs. Profile` помогают удерживать различие между доступным для проверки представлением и предметом, о котором оно сообщает. `C-Calculus` присутствует как концептуальный контекст управляемых связей. Эти отношения следует понимать на уровне исследовательской постановки; ссылка на другую работу не переносит её доказательность или статус в CI. [1, 6-11]

Для использования самого преобразователя не требуется принимать всю программу `c=a+b` или считать её установленным классом ИИ. Инженер может применять ограниченный формат к наблюдаемому сервису, экспериментальному агенту или исследовательской системе, сохраняя собственное определение предмета и участия. При этом CI не предоставляет сравнения со всеми существующими системами журналирования и не устанавливает новизну каждого используемого механизма. Интервальная арифметика, идентификаторы, контрольные суммы и структурированные журналы остаются обычными инженерными средствами.

11. Практический вход и выход

Работу начинают с определения предмета, ветви, окна и измеряемого условия участия. Затем готовят первичные свидетельства и фиксируют решения о допуске. После этого формируют вход по схеме `REFERENCE_IMPLEMENTATION/schemas/ci-profile-input-v1.schema.json`. Простого списка времён недостаточно: нужны источники, типы событий, ограничения и явное описание неизвестного. [2, 3]

Технические команды из корня распакованного пакета имеют следующий вид:

```
python REFERENCE_IMPLEMENTATION/ci_profile.py INPUT.json --output OUTPUT.json
python -m unittest discover -s tests -p "test_*.py" -v
```

Для преобразователя и штатных тестов указан Python 3.10 или новее; используются модули стандартной библиотеки. Отдельные библиотеки JSON Schema не нужны для обычного исполнения: схемы являются публикуемыми контрактами, а программные проверки реализованы в самом модуле. Применение внешнего валидатора допустимо как отдельная проверка, без изменения смысла результата. [2]

На выходе находятся объявленные идентификаторы, временная ось, окно, версии схемы и реализации, политика вычисления, счётчики свидетелей и интервалов, полный вектор, его контрольная сумма и граница вывода. JSON сериализуется детерминированно: ключи объектов упорядочены, разделители фиксированы, кодировка UTF-8. В файл добавляется завершающий перевод строки; хеш вектора рассчитывается без него.

Выход следует сохранять вместе с точной версией входа, реализацией и внешними основаниями допуска. Повторный расчёт того же входа проверяет воспроизводимость преобразования. Для проверки фактической истории системы необходимо дополнительно вернуться к первичным источникам; эталонный код этого шага за пользователя не выполняет.

Перед интерпретацией полезно проследить хотя бы одну значимую запись до исходного объекта: проверить, что ссылка разрешается в ожидаемый материал, временная шкала согласована, ветвь указана верно, а основание допуска соответствует измеряемому условию. Затем проверяют, что итоговая формулировка не расширилась при переносе в таблицу или краткое резюме. Это обычная сверка данных и вывода; для неё не требуется создавать новый механизм идентичности или отдельную систему полномочий. При большом объёме наблюдений способ выборки и её ограничения должны быть описаны, поскольку выборочная сверка не равна проверке всех событий.

12. Какие результаты уже имеются

Публикационный пакет сохраняет ограниченное положительное `synthetic`-свидетельство R157 и два отрицательных локальных эпизода R160 и R161. Их роли различаются, поэтому единая общая метка успеха была бы неверной. [7, 12]

Эпизод	Сохранённый результат	Допустимое значение
R157	OD11_TQ_R157_PASS_OPEN_SYNTHETIC_SEMANTIC_GATE_FIDELITY	Поддержка точного synthetic-сценария и его границ
R160	Пять запросов; замороженный оценщик не разобрал пустое внешнее response	Отрицательный результат квалификации при сохранённом захвате
R161	Schema 5/5; граница утверждения 5/5; роль 4/5; класс 3/5; evidence 0/5; полные записи 0/5	Замороженное совокупное условие не выполнено

В R160 структурированное содержимое другого поля не подставлялось задним числом вместо заранее выбранного оцениваемого ответа. Для R161 первая сохранённая ошибка - `evidence_not_redaction_placeholder`. Частичные положительные показатели остаются диагностикой; они не превращают совокупный результат в успешную квалификацию. Оба эпизода закрыты как отрицательные без повторения старой попытки. [7]

Восемь тестов эталонного преобразователя прошли. Независимое ревью также подтвердило целостность файлов, схемы, соответствие контрольных сумм и воспроизводимую сборку. Это свидетельства о конкретном программном объекте и проверенных путях. Восемь unit tests не являются восемью экспериментами над непрерывностью и не доказывают существование эффекта, специфичного для с. [12]

В отчёте ревью новое исполнение этих тестов относится к Linux и Python 3.12.14. Заявление исходной сборки о прохождении на Python 3.10 сохраняется как отдельная запись. Проверка на другой платформе поддерживает ограниченную переносимость преобразователя, но не воспроизводит исторические процессы Windows/KOT. Повтор одной тестовой команды также не увеличивает число независимых проверяемых случаев.

13. Ограничения выпуска и условия использования

В сборке _02 известна небольшая ошибка обработки некорректного Unicode-входа. Одиночный суррогат, например U+D800, может пройти первоначальную проверку строки, а затем вызвать необработанный UnicodeEncodeError при UTF-8-сериализации. Наблюдались traceback и код выхода 1 вместо документированного контролируемого отказа с кодом 2. Это не обнаруженный ложный успешный расчёт, но реальное ограничение интерфейса ошибок. Настоящая статья его раскрывает; исходные байты программы остаются неизменными. [12]

Вспомогательная инструкция реализации также упоминает старое имя каталога `publication_draft/`. Для распакованного выпуска рабочее место - корень `Continuity_Instrument_v1.0.0/`; приведённая выше команда `discovery` соответствует основному документу воспроизводимости. Это уточнение не требует изменять исторические экспериментальные результаты.

Архивы свидетельств R157, R160 и R161 не входят в публичный исходный ZIP и остаются под управлением владельца. В пакете приведены их точные

идентификаторы, размеры и SHA-256. Эти привязки позволяют проверить полученные байты, но не предоставляют сами данные или автоматическое разрешение на доступ. Поэтому публичная воспроизводимость программного преобразования шире, чем доступность независимой реконструкции исторических эпизодов. [3, 7, 12]

Следует сохранять и компонентные права. Научные тексты и перечисленные метаданные исходного пакета имеют CC BY 4.0. Каталог `REFERENCE_IMPLEMENTATION/`, его схемы и тестовый файл опубликованы с заявлением Copyright © 2026 Ivan Kotov. All rights reserved. No license grant. Публично читаемый исходник при таком режиме не следует описывать как open source. Техническое описание запуска не изменяет эти условия. [13]

14. Практические применения и дальнейшая проверка

При замене модели CI может помочь составить отчёт о подтверждённых участках работы до и после перехода, отдельно описать память, правила и полномочия. Если новый runtime загрузил тот же архив, это наблюдение попадёт в соответствующую линию, но не закроет автоматически неизвестный промежуток и вопрос индивидуальной идентичности.

При восстановлении после сбоя формат позволяет отделить доказанную паузу от времени с утраченными журналами. В исследовании нескольких потомков он помогает сохранять отдельные ветви и не объединять их наработку без объявленного основания. В проверке длительного взаимодействия с человеком он позволяет сопоставлять подтверждённое участие с качеством и независимостью свидетельств. Каждое такое применение потребует собственных источников и заранее выбранного предиката участия; готового измерительного результата выпуск не содержит.

Дальнейшая научная работа должна проверить, насколько выбранные правила наблюдения устойчивы к пропускам, зависимости свидетелей, замене компонентов и конкурирующим объяснениям. Успехом может быть и установление того, что некоторый сильный вывод не поддерживается данными. Текущий вклад CI состоит в явном формате такого ограниченного отчёта и проверяемом преобразовании предоставленных записей. Благодаря этому следующий исследователь получает конкретные данные, правила расчёта и видимые пределы интерпретации.

Литература и материалы

1. Kotov, I. **CI specification**. Continuity Instrument v1.0.0, исходная сборка _02. Файл `CI_SPECIFICATION.md`; дополнительные определения: `TERMINOLOGY.md` и `CONTINUITY_OBSERVABLES.md`.
2. Kotov, I. **CI reference implementation**. Файлы `REFERENCE_IMPLEMENTATION/README.md`, `REFERENCE_IMPLEMENTATION/ci_profile.py`, `REFERENCE_IMPLEMENTATION/schemas/ci-profile-input-v1.schema.json`, `REFERENCE_IMPLEMENTATION/schemas/ci-profile-output-v1.schema.json`.

3. Kotov, I. **Reproducibility**. Файл REPRODUCIBILITY.md того же пакета.
4. Kotov, I. **Continuity observables**. Файл CONTINUITY_OBSERVABLES.md того же пакета.
5. Kotov, I. **Limitations; Terminology**. Файлы LIMITATIONS.md и TERMINOLOGY.md того же пакета.
6. Kotov, I. **Recognition profile**. Файл RECOGNITION_PROFILE.md того же пакета.
7. Kotov, I. **Scientific claim map; Evidence status**. Файлы SCIENTIFIC_CLAIM_MAP.md и EVIDENCE_STATUS.json того же пакета. Содержат точные привязки R157, R160 и R161.
8. Kotov, I. **Theoretical Core of Project Ester**. DOI версии: 10.5281/zenodo.20679718. **Temporal AI Presence Profile v1.0**. DOI версии: 10.5281/zenodo.22070960.
9. Kotov, I. **Personality Formation and Time-Shaped Continuity Profile v0.1**. DOI версии: 10.5281/zenodo.20648731.
10. Kotov, I. **C-Calculus / Governed Binding Stack v0.1**. DOI: 10.5281/zenodo.21205427.
11. Kotov, I. **Beacon**. Concept DOI, указанный в исходном пакете: 10.5281/zenodo.18933553. **Entity vs. Profile**. DOI версии: 10.5281/zenodo.21204415.
12. **Независимая проверка готовности CI v1.0.0, сборка _02, к ручной публикации в Zenodo**. 25 октября 2026. Отчёт verification/SOURCE_REVIEW_PUBLIC_EXTRACT.md. Публичное извлечение результатов автоматизированного технического ревью; не представляется человеческим внешним peer review. SHA-256 исходного полного отчёта: 147bc8c365833fb77164013dde0a6068f7a320d73f86e9a28dbabe7b2a000e2d.
13. Kotov, I. **Licenses and third-party boundaries**. Файл LICENSES_AND_THIRD_PARTY.md исходного пакета.

Идентичность исходного объекта: CI_CONTINUITY_INSTRUMENT_V1_0_0.zip, 50 765 байт; SHA-256 be87dd926ed96e91e8de25bae2e18666f74ac738b173379ddc0092f98563f663. Все относительные пути в списке материалов относятся к корню Continuity_Instrument_v1.0.0/, кроме отдельно названного отчёта ревью. Новые сопроводительные PDF и объединяющий архив имеют собственную идентичность и не подменяют этот исходный ZIP.