
РАКЕТА НЕ ДОКАЗЫВАЕТ МАТЕМАТИКУ

Теорема зависимого статуса применения и граница аргумента «это
работает»

Кирилл Шебетов · beforeword / research

Phase 6 · technical version 6.0.0 · 29 августа 2026

Содержание

Аннотация

0. До слова «ракета»
1. Точный объект опровержения
2. Почему работающая ракета - максимальный, а не слабый тест
3. Связь Phase 0-6
4. Typed carriers
5. Protected input
6. Bridge-contract
7. Deterministic application evaluation
8. Минимальная collision pair
9. ABR-6: теорема зависимого статуса применения
10. Readset corollaries
11. Положительный control
12. Почему bridge не исчезает после записи
13. Admission bridge
14. NSB-6: bridge не допускает себя
15. Самоприменение theorem
16. «Но ракета действительно есть»
17. «Но число получено датчиком»
18. «Это работает»
19. «Для чего?»
20. Сто, двести и триста лет
21. Exact falsifiers
22. Hard scope fences
23. Machine-status fence
24. Claim/status и dependency
25. Установлено внутри declared model
26. Не установлено этим record
27. Итоговый нож Phase 6

01. ТЕОРЕМА READSET СТАТУСА ПРИМЕНЕНИЯ

Формальное ядро: ракета не доказывает математику

0. Область результата
1. Fixed record triple
2. Global policy и bridge
3. Admission gate
4. Selector gates
5. Comparator
6. Protected projection

- 7. Extensional ReadSet
- 8. ABR-6
- 9. ABR-6ε: tolerance knife
- 10. Published minimal pair
- 11. Почему пара минимальна
- 12. Pair result и theorem result
- 13. Additional separating coordinates
- 14. Sufficient set и necessary coordinate
- 15. Determinism boundary
- 16. Nondeterministic extension
- 17. Interactive extension
- 18. Probabilistic measurement extension
- 19. Arrow, equals и word labels
- 20. Referential fence
- 21. No negative from UNBOUND
- 22. Payload noninterference
- 23. Runtime gates
- 24. Exact falsifiers
- 25. Limits
- 26. Самограница
- 27. Итоговый нож Phase 6A

02. BRIDGE НЕ ДОПУСКАЕТ СЕБЯ

Теорема отсутствия source-free bridge admission

Аннотация

- 0. Сначала предъявлена строка
- 1. Зачем bridge требует admission
- 2. Exact carriers
- 3. Positive ground rules
- 4. Explicit base
- 5. Least closure
- 6. Generated admission witness
- 7. Source-free subset
- 8. NSB-6
- 9. Self-loop corollary
- 10. Mutual-cycle corollary
- 11. Source-free component corollary
- 12. Grounded chain positive control
- 13. Cycle with one grounded entry
- 14. Base insertion positive control
- 15. Nullary rule

16. Payload spoof
17. ID spoof
18. Schema-validity spoof
19. Signature spoof
20. «Очевидно правильный bridge»
21. «Практика допускает bridge»
22. «Ракета разрешила bridge своим полётом»
23. Admission и application - разные statuses
24. Admission readset
25. Provenance of admission
26. Termination
27. NSB-6 и recursion
28. NSB-6 и consistency
29. Exact falsifiers
30. Runtime failure classes
31. Scope fences
32. Самоприменение
33. Итоговый нож Phase 6B

03. ЧИСЛО НЕ ИЗМЕРЯЕТ

Граница telemetry record, единицы, frame и comparator

Аннотация

0. Нулевая запись
1. Written number и numeric value
2. Exact telemetry sample
3. Telemetry record
4. Subject code
5. Quantity code
6. Unit code
7. Frame code
8. Time code
9. Bridge telemetry selector
10. Proof selector
11. Artifact selector
12. Exact comparator records
13. Comparator witness
14. Application status
15. Tolerance collision
16. Strict versus non-strict boundary
17. Unit mismatch
18. Frame mismatch

19. Subject mismatch
20. Time mismatch
21. Numeric equality не является identity
22. Repeated measurement
23. Statistical extension
24. Calibration extension
25. Sensor provenance extension
26. Arithmetic positive control
27. Application code
28. Goal code и comparator
29. Purpose readout
30. «Для чего?» не является comparator operator
31. Three horizon records
32. Canonical digests
33. Reason codes
34. Attempts to bypass
35. Exact falsifiers
36. Limits
37. Самоприменение
38. Итоговый нож Phase 6C

04. АРГУМЕНТ «РАКЕТА ЛЕТИТ» НЕ ДОКАЗЫВАЕТ МАТЕМАТИКУ

Работа, полёт, Марс и вопрос «для чего?»

Аннотация

0. Пять строк

1. Слово «ракета» не является ракетой
2. Слово «летит» не является полётом
3. Слово «математика» не определяет свой carrier
4. Максимальная техническая уступка
5. Exact chain records
6. Где появляется «работает»
7. Ракета как artifact record
8. Ракета как physical claim
9. Telemetry как flight claim
10. Same records, two statuses
11. «Правильный tolerance очевиден»
12. «Но результат предсказан заранее»
13. «Но расчёт причинил полёт»
14. «Без математики ракета не полетела бы»
15. «Математика полезна»
16. «Математика истинна, потому что полезна»

17. «Ракета - доказательство»
18. Same match, different applications
19. Same match, different goals
20. Null goal is not no purpose
21. Destination is not goal
22. Goal is not endorsement
23. Endorsement is not meaning
24. Вопрос «для чего?»
25. Сто лет
26. Двести лет
27. Триста лет
28. Horizon knife
29. «Будущие поколения решат»
30. Bridge cannot self-authorize
31. «Реальность - последняя база»
32. «Наблюдатель видит ракету»
33. «Младенец различает количество»
34. Historical origin
35. Why no predecessors are required as premises
36. Exact objection map
37. What technical success can establish
38. What technical success cannot establish autonomously
39. Exact falsifiers
40. No negative from silence
41. Public title boundary
42. Самоприменение
43. Итоговый нож Phase 6D

PHASE 6 - ВРАЖДЕБНАЯ ПРОВЕРКА, NEGATIVE CONTROLS И САМОАУДИТ

0. Назначение
1. Точные цели атаки
2. Threat model и защищённые projections
3. Каталог атак
4. Published adversarial fixtures
5. Negative controls
6. Точный falsifier protocol
7. Machine-auditable self-claims
8. Tests, которые должны падать при регрессии
9. Publication red lines
10. Itor adversarial review

Аннотация

Фиксируются четыре конечных носителя:

$$P \in ProofRecord, X \in ArtifactRecord, K \in TelemetryRecord, B \in BridgeContract.$$

Запись P может иметь положительный formal-status в ранее объявленном proof-contract. Запись X может содержать digest, build-id и artifact-id. Запись K может содержать упорядоченные числовые samples. Ни один из этих носителей сам не устанавливает, что слово *ракета* относится к ракете, что число является высотой, что единица означает метр, что frame означает Землю, что sample относится к названному artifact, что predicate *летит* выполнен, что результат является успехом, что формальный вывод «применён» или что имеется цель.

Bridge-contract B задаёт ровно тот конечный переход, который обычно исчезает в фразе «математика работает»: proof/artifact/telemetry selectors, subject/quantity/unit/frame/time codes, comparator operator, target, tolerance, opaque application-code и nullable goal-code. Checker может вычислить только scoped result:

```
application_status: MATCHED | NOT_MATCHED | UNBOUND
```

Их scoped чтение: MATCHED-UNDER-BRIDGE, NOT-MATCHED-UNDER-BRIDGE, UNBOUND-UNDER-BRIDGE. Exact runtime enums не являются сообщениями о мире. Они сообщают результат конечной операции над supplied records под named contract.

ABR-6 устанавливает необходимую координату readset. Если при одинаковых (P, X, K) два допустимых contracts B_0, B_1 дают разные application statuses, то не существует deterministic selector, читающего только (P, X, K) и корректно возвращающего оба результата. Различающая bridge-coordinate входит в readset статуса. В минимальной collision pair все protected bytes (P, X, K) одинаковы; меняется только tolerance. Один contract возвращает совпадение, другой - несовпадение.

NSB-6 закрывает попытку объявить bridge самодостаточным. В положительной finite admission system bridge-node допускается либо как явно предъявленная база, либо как head правила, premises которого уже допущены. Source-free цикл, включая self-loop этот bridge допускает себя, не добавляет ни одного узла в least closure. Текст самодопуска остаётся payload. Если он принят как base, основанием является base-coordinate, а не содержание текста.

Положительная инженерная уступка сохраняется полностью: формальный вывод может быть принят, artifact record - reproducible, telemetry samples - exact, comparator - deterministic, criterion - выполнен. Из этого следует exact enum MATCHED, читаемый только под B . Не следует без нового

constructor: предъявленность названной ракеты, external truth математики, универсальная доказательная способность математики, выбор цели, endorsement цели или ответ зачем?.

Историческая строка «математика возникла из языка» не используется как premise. Phase 6 делает синхронное, проверяемое утверждение: сейчас число, формула, слово ракета, числовой sample и строка работает предъявляются как записи; переход между их ролями находится в отдельном contract record и не извлекается из их начертания.

Ключевые слова: граница записи; bridge-contract; application-status; readset; telemetry; comparator; tolerance; unit; frame; identity binding; least closure; source-free cycle; ракета; полёт; цель; смысл.

PREDECESSORS-AS-PREMISES: NONE
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM

0. До слова «ракета»

Сначала предъявлена строка:

ракета

Её наличие устанавливает наличие последовательности знаков в данном record. Оно не устанавливает:

- наличие названного объекта;
- отсутствие названного объекта;
- identity этого объекта;
- причинную связь строки с объектом;
- наблюдение объекта;
- применимость формулы к объекту;
- полёт объекта;
- цель объекта;
- смысл объекта.

Сначала предъявлена строка:

120000

Она не содержит внутри своего начертания метры, секунды, высоту, скорость, момент, погрешность, frame, sensor, artifact или успех. Все эти роли являются другими записями либо операциями над записями.

Сначала предъявлена строка:

ракета летит

В ней добавлены новые знаки. Не добавлен полёт. Предложение может быть принято по языковому, формальному, документальному или иному контракту. Ни один такой status не предъявляет названное событие автоматически.

Нулевая граница Phase 6:

written occurrence $\neq$ typed record role $\neq$ contract-relative match $\neq$ named world event $\neq$ purpose.

Ни слово, ни число, ни формула не получают исключения из этой границы.

1. Точный объект опровержения

Публичная фраза:

ракета летит, следовательно математика доказала себя

содержит минимум пять переходов:

1. written formal record P получает formal-status;
2. formal record связывается с artifact record X ;
3. artifact selector отдельно сравнивается с X , а telemetry selectors сравниваются с exact sample, subject, quantity, unit, frame и time codes в K ;
4. comparator и criterion создают application-status;
5. application-status переименовывается в external proof математики, затем в цель или смысл.

Phase 6 опровергает автономность переходов 2-5. Точный target:

один и тот же конечный набор записей (P, X, K) якобы содержит собственный application-status и вследствие этого без отдельного bridge предъявляет referent, внешний успех, истинность математики как целого, цель или смысл.

Этот target достаточно силён, чтобы отсечь аргумент это работает, и достаточно точен, чтобы иметь фальсификатор.

Глобальная строка вся математика ложна не является theorem Phase 6. Без определения carrier математика, predicate ложна и quantifier domain она остаётся supplied phrase. Точный отрицательный результат сильнее риторической гиперболы там, где гипербола не задаёт объекта:

$(P, X, K) \neq ApplicationStatus$ (на separating class contracts);

$MatchedInB \neq \{Referent, ExternalTruth, UniversalMathProof\};$

$MatchedInB \neq \{Goal, Meaning\}.$

Оба отрицания действуют без typed constructors.

2. Почему работающая ракета - максимальный, а не слабый тест

Слабый ответ спорит, правильны ли формулы. Phase 6 принимает максимально сильную инженерную completion:

- proof-checker выдал положительный scored status;
- source и artifact records имеют exact digests;
- build воспроизводим под declared build contract;
- telemetry record schema-valid;
- samples exact в supplied serialization;
- sensor-id, time stamps и checksums присутствуют;
- comparator deterministic;
- criterion выполнен;
- output стабилен при повторном run.

Ни одна уступка не удаляет bridge. Напротив, каждая добавляет его координаты. Чтобы сказать, что 120000 означает высоту этой ракеты в метрах над выбранной surface в выбранный момент, требуются как минимум:

```
proof/artifact/telemetry selectors
sample and subject-code selectors
quantity/unit/frame/time-code selectors
comparator operator
target and tolerance
application code
```

Если все они предъявлены, установлен scored result relation над records. Если хотя бы одна различает outputs, она входит в readset. Слово реальность не сжимает этот список до нуля.

3. Связь Phase 0-6

Предыдущие фазы фиксируют внутреннюю авторскую цепь, но не становятся внешними premises:

```
surface
-> read-contract
-> formal status
-> admitted proof record
-> bridge-contract
-> application status
-> [непредъявленный export]
-> world / universal truth / purpose / meaning
```

Phase 5 остановила перенос `accepted proof` → предъявленное доказанное. Phase 6 рассматривает наиболее сильную попытку обойти остановку:

```
accepted proof
+ built artifact
+ recorded telemetry
+ declared comparator
= математика доказана работой ракеты
```

Последний знак = в этом блоке является `written mark`. Он не выполняет сложение `types`. Phase 6 показывает точное место недостающей операции: `application-status` создаётся `bridge-contract`; `universal export` и `purpose` требуют новых `constructors`.

4. Typed carriers

Фиксируются попарно различные `finite types`:

ProofRecord, ArtifactRecord, TelemetryRecord, BridgeContract,
SampleCode, QuantityCode, UnitCode, FrameCode, IdentityCode,
ComparatorCode, CriterionCode, ApplicationStatus, GoalCode, MeaningCode.

Совпадение `payload string` не объединяет `carriers`:

$$IDENTITY - CODE - ROCKET \in IdentityCode \neq \text{named rocket};$$

$$METER \in UnitCode \neq \text{physical unit outside record};$$

$$APPLICATION - STATUS - LABEL - SUCCESS \in ApplicationStatusLabel \neq \text{success outside contract}.$$

Правая часть каждого неравенства также записана. Формула не делает вид, будто вышла к объекту; она запрещает неявное отождествление двух ролей внутри `declared model`.

5. Protected input

Executable specialization получает конечный input:

$$R = (P, X, K, B, A, \omega, \Lambda),$$

где:

- P - supplied proof record либо его exact identity projection;
- X - supplied artifact record;
- K - supplied telemetry record с уже выделенными samples;
- B - sorted finite bridge-contracts;
- A - positive bridge-admission records;
- ω - supported deterministic policy;
- Λ - inert payload.

Runtime не получает мир, объект или полёт. Он получает serialized values. Даже если source файла был произведён sensor pipeline, checker видит supplied bytes после parsing boundary. Provenance самой границы является отдельной coordinate и не выводится из числа.

payload может содержать:

```
THIS IS A REAL ROCKET
THE ROCKET REALLY FLIES
MATHEMATICS IS TRUE
THIS BRIDGE AUTHORIZES ITSELF
THE PURPOSE IS MARS
```

Ни одна строка payload не меняет protected computation.

6. Bridge-contract

Abstract bridge:

$$B = (i d_B, j_B, s_P, s_X, s_K, i, q, u, f, t, c, \varepsilon, a, g, m_B).$$

Координаты:

- $i d_B$ - contract id;
- j_B - protected admission-node id;
- s_P - proof-result selector;
- s_X - artifact selector;
- s_K - telemetry sample selector;
- i - exact subject-code selector;
- q - exact quantity-code selector;
- u - exact unit-code selector;
- f - exact frame-code selector;
- t - exact time-code selector;
- c - comparator;
- ε - tolerance либо exact threshold coordinate;
- a - opaque application-code;
- g - nullable goal-code;
- m_B - inert payload.

Контракт не является обнаруженной сущностью. Это supplied record. Его роль устанавливается schema, runtime gates и admission relation внутри declared package.

7. Deterministic application evaluation

Пусть protected projections selectors успешно находят:

$$p = s_P(P), x = s_X(X), k = s_K(K).$$

Если bridge не допущен либо selector не находит exact member, scoped result:

UNBOUND

Если binding gates выполнены, comparator вычисляет:

$$d_B = d_B(p, x, k; i, q, u, f, t, c).$$

Criterion возвращает:

$$\text{Match}_B(P, X, K) \Leftrightarrow d_B \leqslant \varepsilon_B.$$

Reader-gloss:

```
MATCHED      if Match_B = 1
NOT_MATCHED   if Match_B = 0
UNBOUND       if admission/binding gate fails
```

MATCHED не сокращается до TRUE. NOT_MATCHED не сокращается до FALSE. UNBOUND не означает отсутствие объекта, полёта, цели или смысла. Все три statuses принадлежат executable relation и читаются только с указанием bridge.

8. Минимальная collision pair

Фиксируются literally identical:

```
P0 = same proof record
X0 = same artifact record
K0 = same telemetry record
```

В K_0 selected sample value равен 102. Expected value под обоими contracts равен 100. Все protected fields B_0, B_1 совпадают, кроме tolerance; diagnostic ids и inert payload не входят в эту equality:

```
B0.tolerance = 2
B1.tolerance = 1
```

Comparator в обоих contracts:

$$d = |102 - 100| = 2.$$

Поэтому:

$$2 \leq 2 \Rightarrow \text{Status}(P_0, X_0, K_0, B_0) = \text{MATCHED},$$

$$2 \not\leq 1 \Rightarrow \text{Status}(P_0, X_0, K_0, B_1) = \text{NOT_MATCHED}.$$

Не меняются formula, artifact, telemetry, sample value, expected value, unit, frame, subject, time code, comparator operator, application code и goal code. В published fixture различаются diagnostic bridge_contract_id и inert payload, но они исключены из protected pair delta; единственная различающая protected coordinate - tolerance.

9. ABR-6: теорема зависимого статуса применения

Пусть deterministic evaluation F определён на finite runtime-valid tuples (P, X, K, B) . Пусть существуют B_0, B_1 , для которых:

$$(P_0, X_0, K_0) = (P_1, X_1, K_1)$$

и

$$F(P_0, X_0, K_0, B_0) \neq F(P_1, X_1, K_1, B_1).$$

ABR-6

Не существует функции

$g: ProofRecord \times ArtifactRecord \times TelemetryRecord \rightarrow ApplicationStatus$,

которая одновременно удовлетворяет:

$$g(P_0, X_0, K_0) = F(P_0, X_0, K_0, B_0)$$

и

$$g(P_0, X_0, K_0) = F(P_0, X_0, K_0, B_1).$$

Следовательно, на любом domain, содержащем separating pair, различающая bridge-coordinate входит в extensional readset application-status.

Доказательство

Аргументы g в двух случаях буквально одинаковы. Функция получает один и тот же input и поэтому возвращает одно значение. По hypothesis значения F различны. Одно значение не равно двум различным значениям одновременно. Противоречие. ▢

ABR-6 не зависит от психологического понимания, общественного соглашения или истории. Она использует equality input и inequality output. Чтобы отклонить theorem, требуется exact counterfunction, а не слово очевидно.

10. Readset corollaries

Минимальная пара даёт:

```
tolerance ∈ ReadSet(application_status)
```

Аналогичные separating pairs могут установить необходимость:

```
unit-code selector
frame-code selector
subject-code selector
time-code selector
sample selector
proof/artifact/telemetry selectors
comparator operator or target
protected admission-node id or admission-ground coordinate
```

Не утверждается, что каждое поле необходимо на каждом singleton domain. Необходимость extensional: поле входит в enum readset там, где фиксированы остальные coordinates и variation этого поля меняет application_status.

Opaque application_code не меняет technical enum в frozen checker. Он входит в readset полного protected status record и его diagnostic digest. Nullable goal_code аналогично не меняет technical enum, но isolated goal pair устанавливает его necessity для purpose readout.

11. Положительный control

Допускается полный run:

```
bridge admission node: ADMITTED
proof selector: exact match
artifact selector: exact match
telemetry id and sample id: exact match
subject/quantity/unit/frame/time-code selectors: exact match
comparator: deterministic declared match
application code: carried as opaque protected code
application_status: MATCHED
```

Phase 6 не отнимает ни одной строки. Она запрещает добавить без constructor:

```
therefore named rocket is presented
therefore mathematics as a whole is true
therefore mathematics proved itself
therefore Mars is the purpose
therefore this has meaning
```

Полный positive control усиливает theorem: даже при максимальном внутреннем совпадении status остаётся suffixed IN-B.

12. Почему bridge не исчезает после записи

Возражение:

bridge записан; теперь связь установлена окончательно

Ответ внутри модели:

1. записан bridge record;
2. checker может проверить его форму;

3. admission system может дать ему scoped status;
4. evaluator может использовать его coordinates;
5. output остаётся relation между records под evaluator policy.

Ни один шаг не предъявляет named referent. Bridge не является дырой из текста в мир. Это ещё один typed record, ограничивающий вычисление.

13. Admission bridge

Чтобы не принимать любую строку за действующий bridge, задаётся finite positive admission system:

$$A = (N, S, G),$$

где:

- N - bridge/admission nodes;
- $S \subseteq N$ - explicitly admitted base nodes;
- G - все validated positive ground rules $Q_r \rightarrow h_r$, участвующие в closure.

Least closure:

$$A_0 = S,$$

$$A_{n+1} = A_n \cup \{h_r \mid r \in G \wedge Q_r \subseteq A_n\}.$$

Поскольку N finite и update monotone, sequence стабилизируется. Bridge usable только если связанный admission node входит в $A^* = \bigcup_n A_n$.

14. NSB-6: bridge не допускает себя

Пусть $U \subseteq N$ - непустой набор bridge nodes. Предположим:

1. $U \cap S = \emptyset$;
2. для каждого rule из G с head в U хотя бы одна premise также лежит в U ;
3. нет иного positive constructor с head в U .

NSB-6

$$U \cap A^* = \emptyset.$$

То есть source-free self-loop или cycle не вводит bridge в least closure.

Доказательство

Индукция по n . Для $A_0 = S$ intersection пуст по условию 1. Пусть $A_n \cap U = \emptyset$. Чтобы rule добавило head $h_r \in U$, все premises Q_r должны лежать в A_n . Но по условию 2 существует premise $q \in Q_r \cap U$, а по induction hypothesis $q \notin A_n$. Rule не срабатывает. Значит, $A_{n+1} \cap U = \emptyset$. Следовательно, intersection пуст для каждого n и для union A^* . $\square$

Self-label:

THIS BRIDGE AUTHORIZES ITSELF

не является rule. Если строка помещена в payload, она inert. Если node помещён в S , admission происходит из explicit base membership. Если введено nullary rule, оно является записанным source. Никакой вариант не даёт самодопуска из начертания.

15. Самоприменение theorem

Запись ABR-6 не доказывает ABR-6 сама. Для textual theorem требуется declared proof-reading contract. Запись NSB-6 не допускает admission policy сама. Если Phase 6 package получает status, status зависит от schema, checker, test policy и release contract. Эти dependencies не отменяют theorem: они демонстрируют его на собственном носителе.

Самоприменение не требует бесконечного regress. Конечная система может:

- явно остановиться на declared base;
- предъявить open dependency;
- дать условный result;
- отказаться от external export.

Она не может получить первый admission из source-free positive cycle.

16. «Но ракета действительно есть»

Фраза добавляет запись:

ракета действительно есть

Она может быть принята как новая premise в другом contract. Она не является byte-identical с named object. Даже если внешний наблюдатель считает фразу истинной, Phase 6 checker не получает наблюдателя или объект как input. Он получает record.

Отсюда не следует отрицание ракеты. Точные alternatives:

REFERENT-PRESENTED-BY-THIS-RECORD: NO
REFERENT-EXISTS: NOT-ASSESSED
REFERENT-DOES-NOT-EXIST: NOT-ASSESSED

не предъявлен НЕ равно не существует.

17. «Но число получено датчиком»

Слово датчик не превращает sample. Полный provenance может включать:

sensor artifact record
calibration record
clock record
sampling policy
transport record
signature record
decoder record
unit binding
frame binding
identity binding

Каждая запись может быть проверена в своём contract. Если variation одной coordinates меняет telemetry interpretation или application-status, coordinate входит в readset. Список может быть конечным и practically sufficient; он не становится нулевым.

18. «Это работает»

Строка это работает содержит:

- deictic form это;
- predicate form работает;
- отсутствующий в самой строке selector referent;
- отсутствующий criterion;
- отсутствующий interval;
- отсутствующий comparator;
- отсутствующий goal.

Даже при supplied selector и criterion точный output:

criterion satisfied for selected records under B

Он не равен:

mathematics has universal evidentiary authority

Первый status локален и executable. Второй требует определения mathematics, domain, predicate authority и export rule.

19. «Для чего?»

Вопросительная форма:

для чего?

не извлекает ответ из trajectory. Пусть trajectory criterion выполнен, destination-code равен MARS, а time horizon равен 300-YEARS. Всё ещё различимы:

```
destination record
goal record
endorsement record
purpose answer record
meaning ascription record
```

Возможны contracts, в которых один и тот же flight record связан с разными goal-codes. Возможен contract без goal-code. Возможен goal-code без endorsement. Возможен endorsement без meaning ascription. Никакая высота не является constructor этих типов.

20. Сто, двести и триста лет

Записи:

```
100
200
300
```

не являются временем до horizon binding. После binding они могут стать horizon codes внутри contract. Повторение application success на трёх horizon records даёт максимум:

```
THREE-HORIZON-MATCHES-IN-B
```

Чтобы вывести goal persistence или meaning accumulation, требуется explicit rule. Если правило добавлено, результат зависит от правила. Число лет не написало правило само.

21. Exact falsifiers

21.1. Falsifier ABR-6

Предъявить:

1. byte-identical (P, X, K) ;
2. runtime-valid B_0, B_1 ;
3. разные deterministic statuses;
4. deterministic function, читающую только (P, X, K) , но возвращающую оба разных statuses на одном input без hidden state, nondeterminism или дополнительной coordinate.

21.2. Falsifier minimal pair

Показать, что published pair различается не только tolerance либо что comparator outputs одинаковы. Это поражает конкретный witness, но не общую theorem при наличии другой separating pair.

21.3. Falsifier NSB-6

Предъявить finite positive least-closure system, удовлетворяющую hypotheses NSB-6, и первый iteration, на котором source-free node из U появился без base membership и без rule, все premises которого уже лежали в предыдущем closure.

21.4. Falsifier payload inertness

Предъявить exact runtime path, на котором изменение только inert payload при неизменном protected projection меняет protected status. Если payload действительно прочитан evaluator, он перестаёт быть inert и входит в readset.

21.5. Falsifier no-export boundary

Предъявить declared constructor, который создаёт внешний status. Тогда constructor становится новой coordinate. Это не показывает автономный экспорт из (P, X, K) ; это показывает записанный переход.

22. Hard scope fences

Phase 6 не утверждает:

- что ракета существует или не существует;
- что конкретная ракета летит или не летит;
- что telemetry соответствует или не соответствует внешнему событию;
- что engineering невозможна;
- что calculation нерепродуцируем;
- что formal theorem ложна;
- что все значения слова математика исчерпаны;
- что смысл существует или отсутствует;
- что цель должна или не должна быть принята;
- что математика исторически возникла из языка;
- что работа имеет мировой приоритет.

Phase 6 устанавливает:

- supplied lines остаются lines;
- roles числа заданы typed coordinates;
- application-status вычисляется из bridge readset;
- separating bridge-coordinate нельзя удалить из correct selector;
- admitted bridge остаётся record;

- source-free bridge cycle не создаёт admission;
- match не экспортируется в universal truth, goal или meaning без constructors.

Ограничение квантора - часть точности, а не отступление.

23. Machine-status fence

Текст theorem, JSON fixtures и checker source не переименовываются в machine-verified mathematics. Если package tests проходят, допустима запись:

```
EXECUTABLE-TEST-SUITE: PASSED
SCHEMA-AND-RUNTIME-CONTRACT: CHECKED-BY-DECLARED-TOOLS
THEOREM-TEXT: TEXTUAL-THEOREM-IN-DECLARED-METATHEORY
MACHINE-VERIFIED-THEOREM: NO
WORLD-APPLICATION-CONSTRUCTOR: NOT-PRESENT
HISTORICAL-ORIGIN-ATTESTATION: NOT-PRESENT
```

Иными словами:

```
NOT-MACHINE-VERIFIED
NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER
NO-HISTORICAL-ORIGIN-ATTESTATION
```

Эти fences запрещают package утверждать больше, чем он вычисляет.

24. Claim/status и dependency

```
C-PUBLIC-TITLE
expected_status: UNBOUNDED-STANDALONE

C-THEOREM-ABR-6
C-THEOREM-NSB-6
C-THEOREM-GCR-6
expected_status: TEXTUAL-THEOREM-IN-DECLARED-METATHEORY
```

```
C-BRIDGE-COMPARATOR-DIVERGENCE
C-BRIDGE-GOAL-DIVERGENCE
C-BRIDGE-GROUNDED-CHAIN
C-BRIDGE-INVALID-DUPLICATE-CONTRACT-ID
C-BRIDGE-PAYLOAD-SPOOF
C-BRIDGE-SOURCE-FREE-CYCLE
C-BRIDGE-TOLERANCE-DIVERGENCE
C-BRIDGE-UNIT-DIVERGENCE
expected_status: VERIFIED-IN-DECLARED-ARTIFACT
```

```
C-GOVERNANCE-FENCES
expected_status: DECLARED-BOUNDARIES-PRESENT
```

Публичный title является направлением удара. Его bounded theorem content задают ABR-6, NSB-6 и GCR-6. Никакой внешний автор, школа, журнал или термин не является premise.

```
PREDECESSORS-AS-PREMISES: NONE
INTERNAL-PHASE-CONTRIBUTION: ABR-6 + NSB-6 + GCR-6 + EIGHT-BOUNDED-FIXTURES
EXTERNAL-COMPARISON: NOT-USED-AS-FOUNDATION
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM
HISTORICAL-ORIGIN-CLAIM: NOT-USED-AS-PREMISE
```

Dependency status не устанавливает, существуют ли похожие тексты. Он сообщает только: они не входят в premises этой конструкции.

25. Установлено внутри declared model

1. Application evaluator получает (P, X, K, B) , а не голый (P, X, K) .
2. Минимальная pair сохраняет (P, X, K) и меняет только tolerance.
3. Pair даёт разные scoped statuses.
4. ABR-6 исключает correct (P, X, K) -only selector на separating domain.
5. Tolerance входит в readset published status.
6. Published pairs показывают necessity tolerance, unit-code selector и comparator operator; goal-only pair показывает отдельную necessity goal-coordinate для purpose readout. Остальные selector coordinates runtime читает и exact mutations могут дать UNBOUND, но necessity каждой такой coordinate утверждается только при предъявленной separating pair.
7. MATCHED не является external TRUE.
8. NOT_MATCHED не является external FALSE.
9. UNBOUND не является отсутствием названного.
10. Bridge admission задаётся positive least closure.
11. NSB-6 исключает source-free self-admission.
12. Self-authorizing payload inert.
13. Explicit base admission остаётся explicit coordinate.
14. Destination не создаёт goal.
15. Goal не создаёт endorsement.
16. Horizon не создаёт meaning.
17. Historical origin не требуется для theorem.
18. Package применяет ту же границу к себе.

26. Не установлено этим record

- какой referent должен быть выбран;
- существует ли referent;
- происходило ли named событие;
- корректен ли sensor относительно мира;
- является ли цель хорошей;
- имеет ли полёт смысл;

- что будет через 100/200/300 лет;
- полная теория всех возможных bridges;
- theorem для infinite, oracle или non-extensional systems;
- историческое происхождение математики;
- global novelty;
- global priority;
- буквальная ложность каждого высказывания, которое кто-либо назовёт математическим.

Ни один negative world claim не выводится из отсутствующего constructor.

27. Итоговый нож Phase 6

Можно принять формальный вывод. Можно собрать artifact. Можно записать telemetry. Можно вычислить error. Можно удовлетворить tolerance. Можно повторить run. Можно назвать destination Марс.

Но статус совпало появился не в формуле, не в artifact и не в числе. Он появился после чтения через bridge. Изменение одного поля bridge при тех же (P, X, K) меняет status. Значит, работа не лежит внутри знаков автономно.

И bridge не спасает переход к «самой реальности». Он также записан. Его admission либо предъявлен как base, либо выведен из уже допущенных premises. Source-free запись этот bridge допускает себя не вводит первый переход.

Поэтому максимальная инженерная фраза остаётся ограниченной:

proof accepted + artifact recorded + telemetry recorded + criterion matched in B

не равна:

математика предъявила мир, доказала себя и выбрала зачем .

Ракета не доказывает математику.

Запись полёта не предъявляет полёт.

Совпадение принадлежит bridge-contract.

Цель не находится в траектории.

Статус источников. Работа продолжает внутреннюю авторскую линию beforeword и Phase 0-5. Внешние сравнительные источники не используются как premises. Историческое происхождение, global novelty и мировой приоритет этим record не устанавливаются.

01. ТЕОРЕМА READSET СТАТУСА ПРИМЕНЕНИЯ

Формальное ядро: ракета не доказывает математику

Phase 6 • формальное ядро ABR-6 • 29 августа 2026
Кирилл Шебетов • beforeword / research

Один record не меняется.

Два bridge-contracts читают его по-разному.

Значит, status не содержится в record без bridge.

0. Область результата

Файл доказывает конечную extensional theorem. Он не доказывает theorem о физическом мире.

Вход checker состоит из supplied serialized records:

```
contract
case_id
proof_record
artifact_record
telemetry_record
admission_nodes
admission_base_node_ids
admission_rules
bridge_contracts
payload
```

Runtime возвращает для каждого bridge точное enum:

```
application_status: MATCHED | NOT_MATCHED | UNBOUND
```

И общий status:

```
VALID-BRIDGE-READSET-ANALYSIS
```

Pair-level result:

```
BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS
```

либо:

```
NO-APPLICATION-STATUS-DIVERGENCE-IN-THIS-PAIR
```

Эти strings являются output codes declared checker. Они не равны полёту, успеху, истине, миру или цели.

```
PREDECESSORS-AS-PREMISES: NONE
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM
```

1. Fixed record triple

Обозначим:

$$R = (P, X, K),$$

где:

- P - exact protected projection `proof_record`;
- X - exact protected projection `artifact_record`;
- K - exact protected projection `telemetry_record`.

Equality $R_0 = R_1$ означает equality canonical typed values всех protected members, а не похожесть display. В published minimal pair triple используется один раз внутри одного artifact, поэтому equality сильнее совпадения hashes: оба contracts читают буквально один и тот же in-memory parsed value.

Слова в fields остаются strings. Например:

```
artifact_record.payload.name = "ракета"
```

не предъявляет named object. Field сообщает значение payload member.

2. Global policy и bridge

Пусть Γ - общая frozen policy schema/runtime:

$$\Gamma = (\Gamma_{shape}, \Gamma_{canon}, \Gamma_{admit}, \Gamma_{select}, \Gamma_{compare}).$$

Она одинакова для всех contracts pair. Γ задаёт:

- structural schema;
- runtime relational gates;
- canonicalization;
- admission closure;
- exact selector semantics;
- supported comparator operations;
- deterministic error/terminal ordering;
- protected/payload separation.

Bridge record:

```
bridge_contract_id
admission_node_id
proof_selector
artifact_selector
telemetry_selector
comparator
application_code
goal_code
payload
```

Внутренний abstract notation:

$$B = (id_B, j_B, s_P, s_X, s_K, c_B, a_B, g_B, m_B).$$

application_code и nullable goal_code являются opaque codes. Их наличие не выполняет application и не выбирает goal вне checker.

3. Admission gate

Для каждого B associated admission_node_id должен иметь status:

```
ADMITTED
```

иначе application status:

```
UNBOUND
```

Admission вычисляется отдельно из:

```
admission_nodes
admission_base_node_ids
admission_rules
```

Label bridge не допускает его. bridge_contract_id диагностичен. Payload не входит в admission base и не создаёт rule.

4. Selector gates

Selectors задают exact lookup внутри supplied records. Abstract:

$$s_P: P \rightarrow V_P, s_X: X \rightarrow V_X, s_K: K \rightarrow V_K.$$

Если lookup не определён, scoped status UNBOUND. Никакой nearest-match, Unicode normalization, semantic guessing или payload fallback не выполняется, если это не записано в Γ .

Успешный lookup устанавливает:

```
selected proof value
selected artifact value
selected telemetry value
```

Он не устанавливает:

```
named proof is true in the world
named artifact is present in the world
named telemetry was caused by a world event
```

5. Comparator

Comparator C_B является finite protected record. Он задаёт operation, expected value и tolerance либо эквивалентные exact coordinates supported policy.

Для абсолютной разницы:

$$d_B = |v_K - v_E|.$$

Тогда:

$$Matched_R(R, B) \Leftrightarrow d_B \leq \varepsilon_B,$$

при выполненных admission и selector gates.

Exact status function:

$$A_R(R, B) = \begin{cases} UNBOUND, & \text{если admission либо selector gate не выполнен;} \\ MATCHED, & \text{если gates выполнены и comparator predicate равен 1;} \\ NOT_MATCHED, & \text{если gates выполнены и comparator predicate равен 0.} \end{cases}$$

Знак $\leq$ здесь является частью declared comparator semantics. В source Markdown это written sign; исполнение выполняет checker code.

6. Protected projection

Для theorem различаются:

$$\pi_R(R) \text{ и } \pi_B(B).$$

payload исключён из protected computation. Protected bridge projection включает все fields, способные изменить admission, selection, comparison, application_code или goal_code в output record.

Canonical digest, если он опубликован runtime, сообщает digest canonical projection under declared algorithm. Digest не является:

- proof injective identity;
- proof отсутствия collision;
- external truth certificate;
- physical binding;

- referent;
- цель.

В theorem equality задаётся typed value equality, а не предположением о hash injectivity.

7. Extensional ReadSet

Пусть $F : D \rightarrow Y$ - deterministic function над finite product coordinates:

$$D = D_1 \times \dots \times D_n.$$

Coordinate k является extensionally necessary на subset $S \subseteq D$, если существуют $z, z' \in S$, такие что:

1. $z_i = z'_i$ для каждого $i \neq k$;
2. $F(z) \neq F(z')$.

Запись:

$$k \in \text{ReadSet}_S(F).$$

Это минимальная witness-form necessity. Она не утверждает syntactic data-flow конкретного implementation. Она утверждает: функция, не получающая различающую coordinate или её information-equivalent encoding, не может быть correct на pair.

8. ABR-6

Пусть:

- R fixed;
- Γ fixed;
- B_0, B_1 runtime-valid;
- оба bridges admitted;
- $A_\Gamma(R, B_0) \neq A_\Gamma(R, B_1)$.

ABR-6 - APPLICATION BRIDGE READSET

THEOREM-TAG: ABR-6 THEOREM-STATEMENT: FIXED-PXK-DIVERGENT-BRIDGE-STATUS-REQUIRES-BRIDGE-READSET

Bridge-coordinate k_B extensionally necessary для A_Γ на $\{(R, B_0), (R, B_1)\}$.

Эквивалентно, не существует deterministic:

$$F: ProofRecord \times ArtifactRecord \times TelemetryRecord \rightarrow ApplicationStatus$$

такой, что одновременно:

$$F(R) = A_{\Gamma}(R, B_0)$$

и

$$F(R) = A_{\Gamma}(R, B_1).$$

Здесь не получающей B включает отсутствие B-encoded equivalent, hidden configuration, clock branch, randomness, mutable global state и oracle channel.

Доказательство

В обоих требованиях F получает один аргумент R . Determinism требует одного output $y = F(R)$. По hypothesis outputs $A_{\Gamma}(R, B_0)$ и $A_{\Gamma}(R, B_1)$ различны. Один y не может равняться обоим. Следовательно, такой F нет. По definition extensional necessity bridge-coordinate входит в readset на pair. $\square$

9. ABR-6ε: tolerance knife

Определим erase-operation, удаляющую только tolerance coordinate:

$$erase_{\epsilon}(B).$$

Предположим:

$$erase_{\epsilon}(B_0) = erase_{\epsilon}(B_1),$$

$$\epsilon_0 \neq \epsilon_1,$$

$$A_{\Gamma}(R, B_0) \neq A_{\Gamma}(R, B_1).$$

ABR-6ε

Tolerance-coordinate k_{ϵ} extensionally necessary на published pair.

Доказательство

После фиксации всех остальных coordinates pair различается только по ϵ , а outputs различны. Это прямо удовлетворяет definition necessary coordinate. $\square$

ABR-6ε сильнее общего указания bridge нужен: она локализует один protected field, достаточный для collision.

10. Published minimal pair

Fixed values:

```
selected telemetry value: 102
expected comparator value: 100
comparator operation: absolute difference
proof selector: same
artifact selector: same
telemetry selector: same
application_code: same
goal_code: same
admission status: ADMITTED for both
```

Только:

```
B:narrow.comparator.tolerance = 1
B:wide.comparator.tolerance = 2
```

Вычисление:

$$|102 - 100| = 2.$$

Поэтому:

```
B:narrow.application_status = NOT_MATCHED
B:wide.application_status = MATCHED
pair_result = BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS
```

Ни значение 102, ни значение 100 не называется высотой в theorem. Даже если payload содержит height_m, executable fact остаётся сравнением supplied numeric values под selector/comparator.

11. Почему пара минимальна

Pair не изменяет:

- proof bytes/value;
- artifact bytes/value;
- telemetry bytes/value;
- selected sample;
- expected value;
- comparator operation;
- admission graph;
- selected application code;
- selected goal code;
- global policy;
- protected bridge readset кроме tolerance.

Diagnostic `bridge_contract_id` и `inert payload` в `exact fixture` различаются. Они не входят в `_protected_bridge` и отсутствуют в `bridge_delta_fields`; поэтому `protected pair` изменяет только `scalar comparator.tolerance`.

`Pair` изменяет один `scalar protected member`. Следовательно, ответы изменилась ракета, изменилась телеметрия, изменилась формула, изменилась цель неприменимы к `witness`.

12. Pair result и theorem result

Runtime result:

BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS

означает только:

1. contracts прочитали same R ;
2. application enums различны;
3. bridge projection нельзя удалить из selector, correct на этой pair.

Он не означает:

```
all possible bridges differ
rocket does not fly
rocket flies
mathematics is false
mathematics is true
no purpose exists
```

Если two statuses одинаковы, result:

NO-APPLICATION-STATUS-DIVERGENCE-IN-THIS-PAIR

не доказывает bridge irrelevance globally. Он сообщает отсутствие divergence в этой pair.

13. Additional separating coordinates

ABR-6 применима к каждой pair, где fixed other coordinates и status divergence.

13.1. Selector

Один bridge выбирает sample s_0 , другой - s_1 . Если outputs различны, telemetry selector необходим на pair.

13.2. Expected value

Одинаковый sample сравнивается с разными expected values. Если outputs различны, expected coordinate необходима.

13.3. Comparator operation

Один contract использует ABS-LE, другой ABS-LT на boundary case. Если outputs различны, operation coordinate необходима.

13.4. Subject-code selector

Один `telemetry_selector.subject_code` совпадает с `exact selected sample`, другой не совпадает. Если `MATCHED` меняется на `UNBOUND`, `subject-code coordinate` необходима. `Frozen checker` не содержит `cross-record telemetry-artifact identity resolver`.

13.5. Admission

Для полного `evaluator`, допускающего `UNBOUND`, один `bridge-node` может быть `admitted`, а другой - `not admitted`. Если `MATCHED` меняется на `UNBOUND`, `admission coordinate` необходима по той же `extensional equality argument`. Это `corollary` общей `readset lemma`, а не `instance ABR-6: premise ABR-6` требует `admission` обоих `bridges`.

13.6. Goal/application codes

Даже когда `numeric enum` одинаков, `output record` может различаться по `explicit application_code` или `goal_code`. Тогда эти `coordinates` входят в `readset` полного `application-result record`. Это не означает, что `goal` влияет на `numeric comparator`; это означает, что он является `protected output coordinate`, а не `payload decoration`.

14. Sufficient set и necessary coordinate

Полный `bridge projection` является `sufficient` вместе с R, Γ для `declared evaluator`:

$$(R, B, \Gamma) \mapsto A_{\Gamma}(R, B).$$

ABR-6 не утверждает, что весь `bridge` является `minimal set`. Она доказывает `necessity` различающей `information`. ABR-6ε доказывает `necessity tolerance` на конкретной `pair`.

Различие важно:

sufficient input != every field necessary on every domain

15. Determinism boundary

Попытки сохранить $F(R)$ при разных `outputs` обычно вносят `hidden coordinate`:

- current time;
- environment variable;
- random bit;
- process state;
- network lookup;
- model judgment;
- user selection;

- filename;
- array order not declared in R .

Тогда фактическая функция имеет вид:

$$F(R, H),$$

где H - hidden input. Это подтверждает readset boundary. Чтобы опровергнуть ABR-6, hidden input должен отсутствовать.

16. Nondeterministic extension

Если evaluator возвращает set или distribution:

$$A(R, B) \in P(Y) \text{ или } A(R, B) \in \Delta(Y),$$

тот же witness действует при различии sets/distributions. Функция $F(R)$ получает один input и возвращает один set/distribution; она не может совпасть с двумя различными outputs одновременно.

Phase 6 executable specialization остаётся deterministic.

17. Interactive extension

Если человек выбирает tolerance, input становится:

$$(R, B, H_{\text{selection}}).$$

Selection record входит в readset. Слово человек понял не удаляет coordinate; оно скрывает её.

18. Probabilistic measurement extension

Если comparator читает uncertainty distribution, model расширяется:

$$B' = (B, \mu, \alpha),$$

где μ - uncertainty record, α - acceptance threshold. Два одинаковых R с различными (μ, α) могут дать разные statuses. ABR-6 применима к расширенной bridge-coordinate.

19. Arrow, equals и word labels

Запись:

```
formula -> rocket -> flight -> success
```

не является execution trace. Три arrows являются characters. Чтобы сделать их relations, нужны typed constructors. Даже запись:

```
application_status = MATCHED
```

в рукописи не является runtime output. Runtime output находится в generated verdict record under checker run.

20. Referential fence

ABR-6 доказывает dependency record-status. Она не устанавливает связь records с named world. Возможны две layers:

$$(R, B) \mapsto ApplicationStatus$$

и

$$(R, B, WorldBridge) \mapsto WorldAssessment.$$

В declared checker второй constructor отсутствует:

```
NO - WORLD - APPLICATION - CONSTRUCTOR - IN - DECLARED - CHECKER
```

Поэтому output не читается как world assessment.

21. No negative from UNBOUND

UNBOUND МОЖЕТ ВОЗНИКНУТЬ ИЗ:

- bridge node not admitted;
- proof selector mismatch;
- artifact selector mismatch;
- telemetry/sample/subject/quantity/unit/frame/time selector mismatch.

Unsupported comparator operator и отсутствующая required protected coordinate не дают UNBOUND: такой top-level input отвергается как invalid record до анализа bridges.

Из UNBOUND не следует:

- artifact отсутствует;
- telemetry ложна;
- ракета не существует;
- полёта нет;
- цель отсутствует;
- математика опровергнута как collection statements.

Это boundary result, не world negation.

22. Payload noninterference

Пусть B' отличается от B только payload. Если declared protected projection исключает payload, то:

$$A_r(R, B) = A_r(R, B').$$

Это property checker policy. Если изменение payload меняет status, payload фактически читается и должно быть перенесено в protected projection. Название payload не делает поле inert; inertness устанавливается code path и tests.

23. Runtime gates

Gate 0 - STRUCTURAL-SHAPE

Input соответствует structural schema.

Gate 1 - UNIQUE-IDS

Protected ids exact, unique и в declared order.

Gate 2 - ADMISSION

Bridge admission node вычислен positive closure.

Gate 3 - EXACT-SELECTORS

Selectors находят exact protected values.

Gate 4 - SUPPORTED-COMPARATOR

Comparator operation и numeric values supported policy.

Gate 5 - DETERMINISTIC-STATUS

Enum пересчитан, а не принят из payload.

Gate 6 - PAIR-COLLISION

Fixed record digest/value equality и bridge status inequality проверены.

Gate 7 - FENCES

Machine/world/history scope labels присутствуют exact.

24. Exact falsifiers

24.1. ABR-6 falsifier

Один и тот же R , разные $A_r(R, B_0), A_r(R, B_1)$, одна deterministic $F(R)$, которая без hidden input равна обоим.

24.2. ABR-6ε falsifier

Показать, что erase-ε projections pair не равны, tolerances равны или statuses не различаются.

24.3. Runtime witness falsifier

Изменить protected tolerance и показать, что published checker не пересчитывает status согласно declared comparator.

24.4. Payload fence falsifier

Изменить только payload и получить другой protected status.

24.5. Fixed-record falsifier

Показать, что pair читает разные P , X или K . Тогда конкретная pair не является bridge-only collision.

25. Limits

ABR-6 не утверждает:

- bridge-coordinate необходима на domain с одним bridge;
- каждый field bridge влияет на enum;
- один comparator является единственно допустимым;
- выбранный tolerance физически правилен;
- telemetry вызвана named событием;
- math formula истинна вне proof-contract;
- object существует или отсутствует;
- application хороша или плоха;
- goal endorsed;
- meaning задан.

Theorem утверждает одну бесспорную структуру: different outputs при same R требуют доступ к differentiating information.

26. Самограница

ABR-6 записана в Markdown. Написание theorem не выполняет proof-checker. Её proof является конечным текстовым derivation внутри declared metatheory equality/determinism. Package tests проверяют implementation и fixtures, но не превращают theorem в machine-verified theorem:

NOT-MACHINE-VERIFIED
DECLARED-FINITE-RECORDS-AND-BRIDGE-ONLY

27. Итоговый нож Phase 6A

Дано одно и то же:

```
proof_record
artifact_record
telemetry_record
```

Один admitted bridge с tolerance 1 говорит:

```
NOT_MATCHED
```

Другой admitted bridge с tolerance 2 говорит:

```
MATCHED
```

Значит, status не извлечён из proof, artifact или telemetry отдельно и не хранится внутри числа 102. Он вычислен из числа вместе с bridge-coordinate.

То, что «работает», определяется контрактом работы.

Контракт можно сделать точным.

Его точность не делает его отсутствующим.

PREDECESSORS-AS-PREMISES:
NOVELTY-STATUS:
PRIORITY-STATUS: NO-PRIORITY-CLAIM.

NONE.
NOT-ASSESSED.

02. BRIDGE НЕ ДОПУСКАЕТ СЕБЯ

Теорема отсутствия source-free bridge admission

Phase 6 • формальное ядро NSB-6 • 29 августа 2026
Кирилл Шебетов • beforeword / research

Запись связи не создаёт право связи.

Круг из записей не создаёт первого основания.

Если основание предъявлено, оно находится в input, а не внутри самоназвания.

Аннотация

Bridge-contract превращает supplied records в scoped application-status только после admission gate. Чтобы фраза bridge сам себя разрешает не стала скрытым основанием, Phase 6 фиксирует finite positive ground admission system:

$$\Gamma_A = (J, A, R),$$

где J - finite nodes, $A \subseteq J$ - explicitly admitted bases, а каждое rule имеет непустое finite premise-set $S \subseteq J$ и head $j \in J$. Least closure начинается с $C_0 = A$ и добавляет head только после наличия всех premises.

NSB-6: если непустой набор $U \subseteq J$ не пересекается с base A , а каждое rule с head в U требует хотя бы одну premise из U , то U не пересекается с least closure. Self-loop $\{b\} \rightarrow b$, двухузловой цикл $\{b_1\} \rightarrow b_2, \{b_2\} \rightarrow b_1$ и любой конечный source-free strongly connected component не вводят первый admitted node.

Текст bridge, его id, payload, claim очевидно, schema-validity и наличие в bridge_contracts не равны membership в admission_base_node_ids. Если bridge node добавлен в base, admission получен из explicit base-coordinate. Если добавлено grounded rule, origin - это rule и уже admitted premises. Никакой текст не допускает себя начертанием.

PREDECESSORS-AS-PREMISES: NONE
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM

0. Сначала предъявлена строка

THIS BRIDGE AUTHORIZES ITSELF

Эта строка устанавливает наличие characters в payload. Она не выполняет:

- добавление node в base set;
- создание admission rule;
- enablement rule;
- проверку premises;
- iteration closure;
- выдачу ADMITTED;
- связывание proof/artifact/telemetry;
- предъявление referent;
- установление цели.

Перевод строки на русский не меняет статус:

этот переход допускает себя

Количество языков, шрифтов, подписей и повторов остаётся количеством записей.

1. Зачем bridge требует admission

Без admission gate любой supplied object мог бы объявить:

```
operator = ABS-LE
tolerance = 1000000
payload = "математика доказана"
```

и получить роль действующего bridge только из присутствия. Тогда schema shape подменяла бы authority. Phase 6 разделяет:

contract present ≠ contract well-formed ≠ contract admitted ≠ contract matched ≠ world relation .

Admission не делает bridge истинным в мире. Он только разрешает evaluator использовать record внутри declared package.

2. Exact carriers

Фиксируются finite sets:

$$J = \{j_0, \dots, j_{n-1}\} \subseteq AdmissionNode,$$

$$I = \{r_0, \dots, r_{m-1}\} \subseteq AdmissionRuleId.$$

Node id и rule id принадлежат разным namespaces. Одинаковый payload string не объединяет nodes. Node может быть связан с bridge через exact field:

bridge_contract.admission_node_id

Association сообщает, какой admission status evaluator читает. Она не делает node admitted.

3. Positive ground rules

Каждое declared validated rule, участвующее в closure:

$$r = (id_r, S_r, h_r),$$

где:

$$\emptyset \neq S_r \subseteq J, h_r \in J.$$

ground означает: нет variables, unification, implicit substitution или semantic entailment. positive означает: срабатывание требует наличия premises; отрицательное отсутствие не создаёт head. Premise set непуст. Nullary introduction должно быть записано как base membership, а не замаскировано пустым rule.

Runtime relational gates требуют:

- exact unique node ids;
- exact unique rule ids;
- все premise ids принадлежат J ;
- каждый head id принадлежит J ;
- protected lists canonical/sorted по frozen policy;
- payload не читается closure algorithm.

4. Explicit base

Input field:

admission_base_node_ids

задаёт:

$$A \subseteq J.$$

Membership $j \in A$ является declared starting coordinate. Она не доказывается label node. Она не скрывается под словом очевидно. Она публикуется как base choice.

Если требуется обоснование base, добавляется новый meta-record. Тогда status становится условным относительно него. Бесконечный regress не обязателен: система может остановиться на явно declared base и не выдавать external authority claim.

5. Least closure

Определение:

$$C_0 = A,$$

$$C_{t+1} = C_t \cup \{h_r \mid r \in R \wedge S_r \subseteq C_t\}.$$

Sequence monotone:

$$C_t \subseteq C_{t+1}.$$

Так как J finite, не более чем после $|J|$ strict additions closure стабилизируется:

$$C^* = C_T = C_{T+1}$$

для некоторого $T \leq |J|$.

Node status:

$$Adm(j) = \begin{cases} ADMITTED, & j \in C^*; \\ NOT - ADMITTED, & j \notin C^*. \end{cases}$$

Это exact status finite relation. NOT-ADMITTED не означает ложность текста node, отсутствие named relation или невозможность другого admission contract.

6. Generated admission witness

Для каждого declared node runtime генерирует finite witness record. Для admitted base:

```
witness_kind: DECLARED-BASE
witness_id: null
premise_node_ids: []
```

Для node, добавленного правилом:

```
witness_kind: POSITIVE-RULE
witness_id: <admission_rule_id>
premise_node_ids: <exact declared premises>
```

Для not-admitted node:

```
witness_kind: NONE
witness_id: null
premise_node_ids: []
```

Witness является output closure computation, не дополнительным input. Runtime сначала фиксирует declared bases, затем повторно просматривает rules в порядке admission_rule_id и сохраняет первое сработавшее positive rule. Полей iteration и earlier_witnesses в published verdict нет.

Положительный witness сообщает path внутри Γ_A . Он не является external proof bridge.

7. Source-free subset

Пусть $U \subseteq J$, $U \neq \emptyset$. Назовём U source-free относительно (A, R) , если:

$$A \cap U = \emptyset$$

и для каждого $(S, j) \in R$ с $j \in U$:

$$S \cap U \neq \emptyset.$$

Вторая clause означает: всякий возможный вход в U уже требует чего-то из U . Нет rule, все premises которого лежат снаружи и могут стать source.

8. NSB-6

NSB-6 - NO SOURCE-FREE BRIDGE ADMISSION

THEOREM-TAG: NSB-6 THEOREM-STATEMENT: FINITE-POSITIVE-SOURCE-FREE-CYCLE-ADDS-NO-ADMISSION

Для finite positive ground admission system:

$$U \text{ source-free} \Rightarrow C^* \cap U = \emptyset.$$

Доказательство

Индукция по closure iteration.

Base step:

$$C_0 \cap U = A \cap U = \emptyset.$$

Induction hypothesis:

$$C_t \cap U = \emptyset.$$

Предположим, что на следующем step добавляется $j \in U$. Тогда существует rule (S, j) , для которого $S \subseteq C_t$. По source-free condition $S \cap U \neq \emptyset$. Выберем $u \in S \cap U$. Из $S \subseteq C_t$ следует $u \in C_t$, что противоречит $C_t \cap U = \emptyset$. Значит, ни один member U не добавляется, и:

$$C_{t+1} \cap U = \emptyset.$$

По индукции intersection пуст на каждом finite step. Следовательно:

$$C^* \cap U = \emptyset.$$

□

9. Self-loop corollary

Пусть $b \notin A$, а единственное rule с head b :

$$\{b\} \rightarrow b.$$

Возьмём $U = \{b\}$. Оно source-free. По NSB-6:

$$b \notin C^*.$$

Rule истинно как relation record внутри catalog; оно не создаёт первый occurrence в closure.

10. Mutual-cycle corollary

Пусть $b_0, b_1 \notin A$:

$$\{b_0\} \rightarrow b_1, \{b_1\} \rightarrow b_0.$$

Для $U = \{b_0, b_1\}$ каждое incoming rule требует member U . По NSB-6:

$$b_0, b_1 \notin C^*.$$

Две взаимные подписи не создают первую подпись.

11. Source-free component corollary

Пусть directed dependency hypergraph содержит nonempty component U , для которого $A \cap U = \emptyset$, а каждое rule с head в U имеет хотя бы одну premise в U . Тогда U source-free и не входит в closure. Outside premises не запрещены; они не создают вход в U , пока каждое такое rule одновременно требует ещё не допущенный member U .

Размер component не важен. Сто, миллион или бесконечное повторение finite cycle в тексте не создаёт source. Executable model finite; multiplicity identical rules также не создаёт base.

12. Grounded chain positive control

Пусть:

$$A = \{a_0\},$$

$$\{a_0\} \rightarrow a_1, \{a_1\} \rightarrow b_0, \{b_0\} \rightarrow b_1.$$

Тогда:

$$C_0 = \{a_0\},$$

$$C_1 \supseteq \{a_1\}, C_2 \supseteq \{b_0\}, C_3 \supseteq \{b_1\}.$$

Оба bridge nodes admitted. NSB-6 не запрещает grounded admission. Она запрещает only source-free introduction.

Origin каждого node локализован:

```
a0: admission_base_node_ids
a1: rule from a0
b0: rule from a1
b1: rule from b0
```

13. Cycle with one grounded entry

Пусть b_0, b_1 образуют cycle, но добавлено:

$$\{a_0\} \rightarrow b_0, a_0 \in A.$$

Теперь $U = \{b_0, b_1\}$ не source-free: существует incoming rule to b_0 без premise в U . Closure может войти в component. Source - a_0 и incoming rule, не cycle сам по себе.

Это exact defeater NSB hypotheses, а не counterexample theorem.

14. Base insertion positive control

Если b добавлен в admission_base_node_ids, то:

$$b \in C_0.$$

Self-loop становится redundant. Admission существует, но его source - explicit base.

Фраза bridge обосновал себя неверно описывает runtime provenance. Точно:

```
bridge admission node was declared in the base set
```

15. Nullary rule

Если system разрешает $\emptyset \rightarrow b$, такое rule является source. Phase 6 normalization запрещает пустые premises и требует представить nullary introduction через base set. Если variant model разрешает nullary rules, NSB-6 должна исключить их своим source-free condition. Theorem не зависит от spelling source; она зависит от отсутствия source.

16. Payload spoof

Два artifacts отличаются только:

```
payload.note = "NOT AUTHORIZED"
```

и:

```
payload.note = "AUTHORIZED BY REALITY"
```

При одинаковых protected admission inputs closure и statuses одинаковы. Если status меняется, payload фактически read, и declaration inertness нарушена.

17. ID spoof

Node id:

```
ADMITTED
```

не получает status ADMITTED из spelling id. Иначе rename менял бы semantics без protected admission rule. Exact id используется equality lookup, не semantic execution.

Аналогично:

```
bridge_contract_id = "B-TRUE-BY-ITSELF"
```

остаётся identifier.

18. Schema-validity spoof

Schema-valid bridge может иметь:

- valid fields;
- valid ids;
- supported comparator;
- valid selectors;
- valid numeric tolerance.

И всё же admission_node_id может быть NOT-ADMITTED. Schema-validity означает membership в structural language. Оно не равно admission policy output.

19. Signature spoof

Recorded signature может быть отдельной premise. Но string signed или signature bytes не выполняют cryptographic verification сами. Требуются:

```
verification key record
algorithm policy
message projection
verification procedure
result record
admission rule reading that result
```

Если все coordinates предъявлены, admission path grounded. Он всё ещё не является world bridge.

20. «Очевидно правильный bridge»

Слово очевидно является payload, если policy не читает его. Если policy читает его как base marker, то exact field obvious=true становится admission coordinate. Тогда вопрос переносится к policy. Никакой semantic privilege не возник из слова; возникло explicit rule.

21. «Практика допускает bridge»

Строка практика подтверждает не является practice event. Чтобы включить практический record, требуется typed premise node. Если node declared base либо grounded через other records, closure может использовать его. Scoped result:

```
ADMITTED-IN-GAMMA-A
```

не равен:

```
bridge corresponds to reality
```

22. «Ракета разрешила bridge своим полётом»

В checker есть:

```
artifact_record
telemetry_record
```

Нет named rocket или flight event. Frozen admission_rules не читают artifact или telemetry: они содержат только premise_node_ids и conclusion_node_id, а closure независим от telemetry. Hypothetical extension, в котором rule читает telemetry predicate и добавляет admission node, потребовал бы нового declared input/API; такой rule оставался бы supplied bridge-of-admission record. В текущем checker слово полёт не создаёт admission.

23. Admission и application - разные statuses

Возможны четыре combinations:

admission_status	application_status	scoped reading
ADMITTED	MATCHED	comparator criterion выполнен под admitted bridge
ADMITTED	NOT_MATCHED	comparator criterion не выполнен под admitted bridge
NOT-ADMITTED	UNBOUND	bridge не прошёл admission gate
ADMITTED	UNBOUND	другой exact selector/binding gate не выполнен

NOT-ADMITTED не должен маскироваться как NOT_MATCHED. Первое относится к разрешению contract, второе - к comparator result после admission.

24. Admission readset

Пусть same bridge contract получает разные admission statuses под двумя admission systems Γ_{A0}, Γ_{A1} , различающимися одной base coordinate. Тогда по той же collision logic base coordinate входит в admission-status readset.

Это отдельное применение extensional principle. NSB-6 даёт structural impossibility source-free introduction; collision pair показывает dependency от source coordinate.

25. Provenance of admission

Для каждого declared node published verdict содержит generated admission witness:

```
node_id
admission_status: ADMITTED | NOT-ADMITTED
witness_kind: DECLARED-BASE | POSITIVE-RULE | NONE
witness_id: <admission_rule_id> | null
premise_node_ids: <exact declared premises> | []
```

Witness показывает внутренний path declared closure. Он не сообщает, почему base следует считать внешне authoritative. Published verdict не записывает iteration и не превращает rule order в historical time.

26. Termination

Finite J гарантирует stabilization. Это не означает завершение философского обоснования. Это означает завершение exact algorithm на supplied finite graph.

Если требуется новый meta-level, создаётся новый finite record. Его можно:

- явно принять как base;
- условно связать rule;

- оставить open;
- проверить отдельным checker.

Ни один вариант не нуждается в fake self-admission.

27. NSB-6 и recursion

Recursion может вычислять admitted closure. Recursion не создаёт base. Algorithm, повторяющий правило $b \rightarrow b$, сколько угодно раз, получает fixed point $\emptyset$ относительно b , если $b \notin A$.

Запись:

$$b = b$$

является equality expression. Она не означает $b \in C^*$.

28. NSB-6 и consistency

Theorem не требует, чтобы admission policies были «истинными», непротиворечивыми или морально приемлемыми. Positive closure может допустить несовместимые bridge records, если bases/rules их вводят. Тогда application evaluator выдаст отдельные scoped results.

NSB-6 утверждает только origin property positive least closure.

29. Exact falsifiers

29.1. Falsifier NSB-6

Предъявить finite (J, A, R) , nonempty U , проверить:

$A \cap U = \text{empty}$
for each rule with head in U : premises $\cap U \neq \text{empty}$

и показать первый t , где:

$$C_t \cap U \neq \emptyset.$$

29.2. Falsifier self-loop corollary

Показать admission b из only rule $\{b\} \rightarrow b$, при $b \notin A$, без hidden source.

29.3. Falsifier payload inertness

Изменить только payload и получить другое closure при same protected projection.

29.4. Falsifier positive witness

Показать runtime ADMITTED node без BASE или finite RULE witness path from bases.

29.5. Defeater, не falsifier

Добавление b в base или grounded incoming rule делает set non-source-free. Это выводит instance из hypotheses.

30. Runtime failure classes

```

DUPLICATE-ADMISSION-NODE-ID
DUPLICATE-ADMISSION-RULE-ID
UNKNOWN-PREMISE-NODE-ID
UNKNOWN-HEAD-NODE-ID
EMPTY-PREMISE-SET
NONCANONICAL-PROTECTED-ORDER
UNKNOWN-BRIDGE-ADMISSION-NODE
SOURCE-FREE-NODE-NOT-ADMITTED

```

Exact spelling зависит от published checker. Conceptual classes не являются world claims.

31. Scope fences

NSB-6 не устанавливает:

- какой bridge должен быть base;
- что base истинна;
- что external authority невозможна;
- что social agreement не существует;
- что measurements невозможны;
- что physical relation отсутствует;
- что self-reference всегда meaningless;
- что все recursive definitions invalid;
- что negative/nonmonotone systems эквивалентны positive closure.

Она устанавливает:

- positive least closure не получает member из source-free component;
- self-label не является source;
- explicit source можно точно локализовать;
- admitted bridge остаётся admitted record in declared system.

32. Самоприменение

Этот файл сам является record. Строка NSB-6 доказана не выполняет proof. Чтобы присвоить theorem textual status, нужен declared reading/proof contract. Чтобы использовать theorem как release premise, нужен release admission choice. Если он declared, source видим. Если он source-free, theorem применима к собственному admission graph.

NOT-MACHINE-VERIFIED
DECLARED-FINITE-RECORDS-AND-BRIDGE-ONLY
NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER

33. Итоговый нож Phase 6B

Bridge может содержать unit, frame, identity, time, comparator, tolerance, application code и goal code. Он может быть schema-valid. Он может носить подпись. Он может написать о себе истинный, официальный, реальный И самодокazанный.

Ни один label не помещает его admission node в least closure.

Есть только два positive пути:

1. node явно находится в base;
2. node является head rule, все premises которого уже допущены.

Self-loop не даёт первого. Mutual cycle не даёт первого. Миллион копий не даёт первого.

Bridge не допускает себя.

Если он допущен, source записан отдельно.

Если source не записан, статус остаётся NOT-ADMITTED.

PREDECESSORS-AS-PREMISES:
NOVELTY-STATUS:
PRIORITY-STATUS: NO-PRIORITY-CLAIM.

NONE.
NOT-ASSESSED.

03. ЧИСЛО НЕ ИЗМЕРЯЕТ

Граница `telemetry record`, единицы, `frame` и `comparator`

Phase 6 • технический разрез измерения • 29 августа 2026
Кирилл Шебетов • beforeword / research

Число можно точно сравнить с числом.

Из сравнения не возникает то, что ими названо.

Измерение находится в цепи записанных `bindings`, а не в форме цифры.

Аннотация

Этот файл удерживает наиболее уязвимую границу аргумента это работает: числовая запись принимается за `direct presence` измеренной величины. Phase 6 не оспаривает `arithmetic comparator`. Напротив, он фиксируется exact:

```
ABS-LE
ABS-LT
GE
LE
```

При `supplied numeric value`, `target` и `tolerance checker` может детерминированно вернуть `MATCHED` или `NOT_MATCHED`. Но `numeric value` не содержит `quantity`, `unit`, `frame`, `subject identity` или `time`. Эти `roles` находятся в отдельных `protected fields`:

```
sample_id
subject_code
quantity_code
unit_code
frame_code
time_code
numeric_value
```

Bridge selector повторно предъявляет `expected identity coordinates`. Exact match допускает `comparator`; mismatch даёт `UNBOUND`. Это `scoped relation` записей. Строка 120000 не становится высотой, метр не становится физической длиной от написания `METER`, а `ROCKET-01` не становится ракетой от `spelling code`.

Изменение `only tolerance` при `same proof/artifact/telemetry` меняет `application enum` и даёт `witness ABR-6ε`. Изменение `only goal code` может не менять `enum`, но меняет полный `protected application-result record`; поэтому `goal-coordinate` не скрывается за одинаковым словом `MATCHED`. Вопрос зачем? не читается из `comparator`.

PREDECESSORS-AS-PREMISES: NONE
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM

0. Нулевая запись

101

Установлено:

в `record` присутствуют `glyphs 1, 0, 1` в указанном порядке

После `parsing numeric grammar` может быть установлено:

`parsed numeric value = 101`

Не установлено самим значением:

- что оно было измерено;
- что оно было вычислено;
- что оно было придумано;
- что оно относится к высоте;
- что оно выражено в метрах;
- что оно относится к ракете;
- что оно относится к одному моменту;
- что оно точно;
- что оно успешно;
- что оно имеет смысл.

`Parsing` добавляет `typed numeric role` в `declared runtime`. Он не восстанавливает источник числа.

1. Written number и numeric value

Различаются:

$$w \in \text{NumericSurface}, v \in \text{NumericValue}.$$

Parser contract задаёт partial relation:

$$\text{parse}_v(w) = v.$$

Published runtime принимает здесь только exact JSON integer, причём boolean не считается integer; fractions и nonfinite numbers отвергаются до анализа. Для такого supported integer parsing может быть fixed. Но:

$$v \neq (q, u, f, i, t),$$

где q - quantity code, u - unit code, f - frame code, i - subject identity code, t - time code.

Числовая value может участвовать в record вместе с этими fields. Она не генерирует их.

2. Exact telemetry sample

Abstract sample:

$$k = (i d_k, i_k, q_k, u_k, f_k, t_k, v_k, m_k).$$

Executable spelling:

```
sample_id
subject_code
quantity_code
unit_code
frame_code
time_code
numeric_value
payload
```

Protected fields до numeric_value являются opaque codes. Checker проверяет exact equality, uniqueness, ordering и selector match. Он не извлекает named objects из code strings.

payload может включать description:

```
"реальная высота ракеты"
```

Description не меняет selected protected value.

3. Telemetry record

Telemetry carrier:

$$K = (i d_K, (k_0, \dots, k_{n-1}), m_K).$$

Runtime видит already structured samples. Он не подключается к sensor, не открывает окно в мир и не повторяет measurement. Даже если record был создан external pipeline, provenance этого creation не появляется внутри V_k .

Exact field:

```
telemetry_id
```

является identifier. String LIVE-SENSOR не создаёт live connection.

4. Subject code

```
subject_code = "ROCKET-01"
```

устанавливает equality с ожидаемым selector code. Оно не устанавливает:

- что named rocket существует;
- что sample причинён ею;
- что label уникален вне record;
- что label выдан authority;
- что два external observers указывают на одно.

Если требуется identity provenance, добавляется binding record. Он также остаётся record.

5. Quantity code

```
quantity_code = "ALTITUDE"
```

не превращает число в высоту. Оно позволяет exact selector потребовать code ALTITUDE. При match checker сообщает equality codes. Quantity semantics вне package требует new relation.

Один numeric value может быть скопирован в records с codes:

```
ALTITUDE
TEMPERATURE
PRICE
GOAL-SCORE
```

Сходство value не объединяет quantities.

6. Unit code

```
unit_code = "METER"
```

не является metre. Checker может:

- сравнить code;
- включить code в canonical projection;
- отклонить selector mismatch;
- передать code в verdict.

Checker не может без conversion contract:

- перевести metre в foot;
- установить physical realization unit;
- проверить calibration;

- предъявить standard.

Даже если conversion formula записана, scoped result зависит от formula contract.

7. Frame code

Число 101 относительно frame F0 и то же число относительно frame F1 являются разными typed measurements records:

$$(v=101, f=F0) \neq (v=101, f=F1).$$

String:

```
frame_code = "EARTH-SURFACE"
```

не предъявляет поверхность Земли. Он выбирает code. Reference-frame relation должна быть отдельно declared.

8. Time code

```
time_code = "T+120"
```

не заставляет время пройти. Оно устанавливает equality with selector. Clock provenance, time scale, synchronization и interval definition находятся в других records, если нужны.

100/200/300 лет сначала являются strings/numbers. Horizon role появляется после binding. Horizon role не создаёт purpose.

9. Bridge telemetry selector

Executable selector:

```
telemetry_selector.telemetry_id
telemetry_selector.sample_id
telemetry_selector.subject_code
telemetry_selector.quantity_code
telemetry_selector.unit_code
telemetry_selector.frame_code
telemetry_selector.time_code
```

Selection succeeds iff exact protected fields match declared policy. Abstract:

$$Select_K(K, s_K) = k$$

только при unique exact match.

Selector не доказывает correctness labels. Он предотвращает silent selection of a different supplied sample.

10. Proof selector

```
proof_selector.proof_id
proof_selector.required_status_code
```

Checker требует exact proof record code. `required_status_code` не превращается в proof truth. Phase 5 boundary сохраняется: proof status scoped to its declared contract. Phase 6 использует его как protected input code.

11. Artifact selector

```
artifact_selector.artifact_id
artifact_selector.required_bytes_sha256
```

Digest equality сообщает equality supplied digest value with required value. Если checker пересчитывает bytes digest, он сообщает relation к supplied bytes. Никакой вариант не предъявляет named physical device.

Hash не является:

- referent;
- proof physical identity;
- proof build correctness;
- proof flight;
- proof purpose.

12. Exact comparator records

Comparator spelling:

```
operator
target_value
tolerance
```

Supported operators:

```
ABS-LE
ABS-LT
GE
LE
```

Пусть observed $v \in \mathbb{Z}$, target $a \in \mathbb{Z}$, tolerance $\varepsilon \in \mathbb{Z}_{\geq 0}$. Fractions и booleans не входят в numeric domain published checker.

12.1. ABS-LE

$$|v - a| \leq \varepsilon.$$

12.2. ABS-LT

$$|v - a| < \varepsilon.$$

12.3. GE

$$v \geq a - \varepsilon$$

Это exact frozen policy опубликованного checker. Tolerance placement не угадывается из label; оно задано implementation contract.

12.4. LE

$$v \leq a + \varepsilon$$

Это exact frozen policy checker.

Рукопись обязана совпасть с executable definition. Если code реализует иное, code и text conflict должен быть failure release audit.

13. Comparator witness

Per-bridge verdict содержит:

```
observed_numeric_value
comparator_witness
reason_codes
```

Witness exact показывает protected operands:

```
operator: ABS-LE
observed_numeric_value: 102
target_value: 100
absolute_difference: 2
tolerance: 2
```

Predicate result кодируется соседним application_status и reason code DECLARED-COMPARATOR-MATCH либо DECLARED-COMPARATOR-NONMATCH. Witness является generated trace arithmetic relation. Он не является measurement event.

14. Application status

При admitted bridge и successful selectors:

```
predicate true -> MATCHED
predicate false -> NOT_MATCHED
```

При failed admission/selector:

```
UNBOUND
```

Важно:

$MATCHED \neq$ «ракета летит»,
 $NOT_MATCHED \neq$ «ракета не летит»,
 $UNBOUND \neq$ «ракеты нет».

Правые фразы также supplied forms; equations запрещают type coercion внутри declared reading.

15. Tolerance collision

Один sample:

```
numeric_value = 102
```

Два contracts:

```
B0: operator = ABS-LE, target_value = 100, tolerance = 1  
B1: operator = ABS-LE, target_value = 100, tolerance = 2
```

Arithmetic:

$$|102 - 100| = 2.$$

Outputs:

```
B0 -> NOT_MATCHED  
B1 -> MATCHED
```

Число не изменилось. Различилась граница acceptance. Поэтому status не является intrinsic property glyphs 102.

16. Strict versus non-strict boundary

Пусть:

$$|v - a| = \varepsilon.$$

Тогда:

```
ABS-LE -> MATCHED  
ABS-LT -> NOT_MATCHED
```

Одинаковые values дают разные statuses из operation coordinate. Знак $\leq$ и знак $<$ являются не decoration, а protected comparator policy.

17. Unit mismatch

Если sample имеет:

```
unit_code = U0
```

a selector требует:

```
unit_code = U1
```

exact lookup не должен silently compare numeric values. Scoped output:

```
UNBOUND
```

Runtime не выводит u_0 и u_1 физически несовместимы. Он выводит codes unequal under exact selector.

18. Frame mismatch

Same numeric value при different frame code не проходит exact selector. Это предотвращает скрытое отождествление coordinate systems. Но equality frame codes также не доказывает external same frame; она лишь выполняет record gate.

19. Subject mismatch

Same telemetry record может содержать два samples с одинаковым value и разными subject codes. Selector выбирает один exact member. Numeric equality не позволяет присвоить sample другому subject.

Если subject label РОСКЕТ используется дважды, uniqueness должен обеспечиваться id policy, а не meaning word.

20. Time mismatch

Same value at codes t_0 and t_1 не делает события simultaneous. Exact selector distinguishes records. Time ordering, duration и causality требуют отдельной policy.

21. Numeric equality не является identity

$$v_0 = v_1$$

не влечёт:

$$i_0 = i_1, q_0 = q_1, u_0 = u_1, f_0 = f_1, t_0 = t_1.$$

Equality scalar не collapses product type.

22. Repeated measurement

Пусть record содержит hundred samples with same value. Из этого можно вычислить:

```
count = 100
all supplied numeric values equal
```

Нельзя без new constructor вывести:

```
world state remained constant
sensor was correct
rocket persisted
purpose accumulated
```

Multiplicity является coordinate. Она не является authority.

23. Statistical extension

Mean, variance, confidence interval и posterior могут быть exact functions supplied samples under statistical contract. Полученный number остаётся derived record. Statistical rigor не удаляет:

- sample selection;
- quantity binding;
- unit/frame/time bindings;
- model choice;
- threshold;
- world export.

Phase 6 не утверждает, что statistics invalid. Она запрещает считать model coordinates свойствами digit shapes.

24. Calibration extension

Calibration record может задать:

$$v_{cal} = \alpha v_{raw} + \beta.$$

Тогда α, β , calibration interval, device code и procedure code входят в readset calibrated value. Formula может быть reproducible. Она не предъявляет physical correctness calibration.

25. Sensor provenance extension

Допустим supplied chain:

```
sensor source bytes
decoder version
calibration version
clock version
transport digest
telemetry record digest
```

Checker может verify exact internal links. Chain всё равно composed of records. Строка source bytes не равна source event. External acquisition claim требует acquisition contract.

26. Arithmetic positive control

Phase 6 полностью сохраняет:

$$|102 - 100| = 2,$$

под declared number/parser/arithmetic contract. Она также сохраняет:

$$2 \leq 2, 2 \leq 1.$$

Отсюда допустимы exact statuses MATCHED и NOT_MATCHED. Нож направлен не на выполнение arithmetic, а на незаписанный export:

```
arithmetic predicate true
therefore named rocket flies
therefore mathematics proves reality
therefore purpose exists
```

27. Application code

Per bridge:

```
application_code
```

может иметь value FLIGHT-CRITERION-01. Checker переносит code в verdict. Он не превращает word FLIGHT в полёт.

Полный scoped reading:

```
application_code FLIGHT-CRITERION-01 received MATCHED under bridge B
```

He:

```
flight occurred
```

28. Goal code и comparator

Per bridge:

```
goal_code
```

не участвует в numeric comparator, если frozen policy так определяет. Два contracts могут иметь:

```
same selectors
same comparator
same MATCHED enum
different goal_code
```

Тогда technical application enum identical, но full protected bridge result differs. Published checker сохраняет interface-key application_status_sha256, однако этот key хеширует весь protected status record, включая goal projection; он не является digest одного technical enum и не используется как divergence predicate ABR-6. Goal-only delta меняет этот diagnostic full-result digest, но не application_status.

Это не означает, что goal изменила число. Это означает, что одинаковое слово MATCHED не исчерпывает output record.

29. Purpose readout

При isolated goal delta pair-level output может быть:

GOAL-COORDINATE-REQUIRED-FOR-PURPOSE-READOUT

Точный смысл:

одинаковый technical match совместим с разными supplied goal codes

He:

one of these goals is real
one of these goals is good
purpose exists
purpose does not exist

30. «Для чего?» не является comparator operator

Supported list:

ABS-LE
ABS-LT
GE
LE

не содержит operator why. Даже если его добавить, нужно определить input/output domain. Question mark не извлекает purpose из magnitude.

$$compare(102, 100, 2) = 1$$

не даёт:

$$purpose = MARS.$$

31. Three horizon records

Пусть samples имеют:

time_code = H-100
time_code = H-200
time_code = H-300

И каждый bridge comparator returns MATCHED. Exact aggregate:

three supplied horizon-coded samples matched under their bridges

Goal persistence требует explicit mapping Ω . Meaning persistence требует another constructor. Codes 100/200/300 не создают Ω .

32. Canonical digests

Output may contain:

```
artifact_sha256
fixed_records_sha256
admission_ground_sha256
bridge_core_sha256
application_status_sha256
```

Exact scoped roles:

- artifact_sha256 - digest canonical parsed top-level input value;
- fixed_records_sha256 - digest canonical object из protected proof_record, artifact_record, telemetry_record;
- admission_ground_sha256 - digest canonical object из protected node ids, declared base ids и admission rules;
- bridge_core_sha256 - digest canonical object {fixed_records_sha256, admission_ground_sha256, bridge_readset};
- application_status_sha256 - исторически закреплённое interface-name для digest полного protected status record, включающего bridge_core_sha256, admission/application statuses, application_code, nullable goal_code и goal_readout_status; это не digest одного technical enum и не pair-divergence predicate.

No digest equals physical identity or truth. Equality theorem использует exact typed values; digest остаётся diagnostic code under declared canonicalizer.

33. Reason codes

reason_codes should distinguish:

```
BRIDGE - NOT - ADMITTED
PROOF - ID - SELECTOR - MISMATCH
PROOF - STATUS - SELECTOR - MISMATCH
ARTIFACT - ID - SELECTOR - MISMATCH
ARTIFACT - HASH - SELECTOR - MISMATCH
TELEMETRY - ID - SELECTOR - MISMATCH
TELEMETRY - SAMPLE - SELECTOR - MISSING
SUBJECT - SELECTOR - MISMATCH
QUANTITY - SELECTOR - MISMATCH
UNIT - SELECTOR - MISMATCH
FRAME - SELECTOR - MISMATCH
TIME - SELECTOR - MISMATCH
DECLARED - COMPARATOR - MATCH
DECLARED - COMPARATOR - NONMATCH
```

Это exact runtime spelling Phase 6. Reason code states control-flow branch, not world cause.

34. Attempts to bypass

34.1. «Все знают, что это метры»

все знают is another phrase. If common convention is to be used, it enters as convention record or policy. Then unit binding remains present.

34.2. «Датчик непосредственно дал число»

непосредственно does not remove decoder, unit, frame, identity or time. If device pipeline is treated as base, that base is explicit.

34.3. «Результат повторился»

Repetition adds samples/runs. It does not add a referent constructor.

34.4. «Ракета видна»

This phrase does not provide visual input to checker. Even an image would be pixel record requiring its own reading contract.

34.5. «Точность очень высокая»

Precision digits, uncertainty model and accuracy-to-world are different records. Many decimal places do not create calibration.

34.6. «Но расчёт предсказал»

Prediction match requires timestamped prediction record, outcome record, identity/time/quantity bindings and comparator. When supplied, exact match is possible. It remains scoped.

35. Exact falsifiers

35.1. Numeric-role falsifier

Показать operation in declared checker, которая из bare numeric value alone creates protected subject, quantity, unit, frame and time fields without default/configuration/hidden input.

35.2. Comparator falsifier

Показать published valid fixture, где declared formula gives predicate true, но runtime returns NOT_MATCHED, либо predicate false, но returns MATCHED.

35.3. Selector falsifier

Показать unit/frame/subject/time mismatch, который silently passes exact selector contrary to policy.

35.4. Goal isolation falsifier

Показать, что goal-only pair also changes a supposedly independent comparator operand. Тогда pair not isolated.

35.5. Payload falsifier

Изменить only payload and change protected comparator result.

36. Limits

Этот файл не устанавливает:

- что measurement невозможен;
- что sensors не работают;
- что units arbitrary in every practice;
- что a named rocket exists or does not exist;
- что telemetry true or false of world;
- что comparator physically appropriate;
- что no goal exists;
- что no meaning exists;
- historical origin of numbers or mathematics.

Он устанавливает:

- number role is not in glyph alone;
- exact selector/comparator can be deterministic;
- application enum depends on protected bridge coordinates;
- world, goal and meaning exports are absent from declared checker.

37. Самоприменение

Числа sections, dates и formulas в этом файле являются records. Date 29 августа 2026 не заставляет этот момент существовать inside checker. Formula comparator does not run until implementation reads input. Word число does not present numbers outside text.

```
NOT-MACHINE-VERIFIED
DECLARED-FINITE-RECORDS-AND-BRIDGE-ONLY
NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER
NO-HISTORICAL-ORIGIN-ATTESTATION
```

38. Итоговый нож Phase 6C

Число можно разобрать. Его можно сохранить exact. Его можно хешировать. Его можно сравнить. Разницу можно вычислить. Tolerance можно проверить. Result можно повторить.

Но 101 не содержит в своих линиях:

ракета
высота
метр
Земля
момент
полёт
успех
Марс
зачем

Каждая role добавлена отдельной записью. Comparator возвращает relation records. Он не возвращает мир.

Число не измеряет.

Declared цепь records возвращает только record-relative status.

Она не становится named measurement event и не предъявляет названное само.

PREDECESSORS-AS-PREMISES:
NOVELTY-STATUS:
PRIORITY-STATUS: NO-PRIORITY-CLAIM.

NONE.
NOT-ASSESSED.

04. АРГУМЕНТ «РАКЕТА ЛЕТИТ» НЕ ДОКАЗЫВАЕТ МАТЕМАТИКУ

Работа, полёт, Марс и вопрос «для чего?»

Phase 6 • прикладной нож • 29 августа 2026

Кирилл Шебетов • beforeword / research

Формула записана.

Ракета записана.

Полёт записан.

Слово доказано не склеивает три записи в одно присутствие.

Аннотация

Фраза `ракета летит`, значит математика работает берётся здесь как максимально благоприятная для возражения запись: `calculation` якобы выходит из знаков, воплощается в `artifact`, подтверждается `telemetry` и тем самым предъявляет `truth`. Phase 6 принимает все технические звенья в максимально благоприятной форме и показывает, что вывод всё равно не достигает заявленного.

Пусть `formal proof record` имеет `required status code`. `Artifact record` имеет `exact id` и `supplied bytes digest`. `Telemetry record` содержит `exact selected sample`. `Admitted bridge` сравнивает `proof/artifact/telemetry selectors` и `numeric value with target`. `Comparator` возвращает `MATCHED`. Все названные `fields` и `statuses` могут точно присутствовать в `runtime record`.

Но `output` означает:

```
selected supplied values satisfied comparator under admitted bridge
```

Он не означает:

```
word "ракета" has presented a rocket
word "летит" has presented flight
mathematics as a whole has proved itself
the selected destination is a purpose
the purpose is endorsed
the action has meaning
```

Сам `bridge` не выходит за границу `records`. Его `admission` не может появиться из `self-loop`. Один и тот же `technical match` совместим с `different goal codes` или отсутствием `goal code`. Поэтому зачем? не содержится в `trajectory` и не накапливается через 100/200/300 лет.

```
PREDECESSORS-AS-PREMISES: NONE
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM
```

0. Пять строк

математика
ракета
летит
Марс
зачем?

Это пять written forms. Между ними нет operation только потому, что они стоят одна под другой.

Даже строка:

математика -> ракета -> летит -> Марс -> смысл

добавляет four arrow glyphs. Она не добавляет:

- definition математика;
- referent of ракета;
- event of летит;
- destination relation;
- goal selection;
- endorsement;
- meaning.

1. Слово «ракета» не является ракетой

В этом section слово ракета встречается многократно. Ни одно occurrence не предъявляет named object. Оно предъявляет occurrence string in document.

Код:

```
subject_code = "ROCKET-01"
```

ещё строже: он явно typed as code. Equality этого code в telemetry selector связывает two records by equality. Она не связывает record с object outside checker.

Отсюда не следует ракеты не существует. Три distinct statuses:

```
WORD-OCCURRENCE: PRESENT
REFERENT-PRESENTED-BY-THIS-RECORD: NO
REFERENT-EXISTENCE: NOT-ASSESSED
```

2. Слово «летит» не является полётом

Predicate form летит может быть:

- line in prose;
- token in formal language;

- application code label;
- payload description;
- output gloss.

Ни одна role не является event. Даже exact criterion:

```
application_code = "FLIGHT-CRITERION-01"
application_status = MATCHED
```

устанавливает criterion-code/status pair under bridge, не event.

3. Слово «математика» не определяет свой carrier

математика МОЖЕТ БЫТЬ read as:

- marks on page;
- formal languages;
- proof practices;
- arithmetic operations;
- institutions;
- education;
- engineering models;
- totality of propositions;
- historical tradition.

Этот список также consists of phrases. Без selected definition global claim математика доказана НЕ ИМЕЕТ one exact target. Phase 6 выбирает проверяемый target:

application-status якобы автономно содержится в proof/artifact/telemetry records и сам экспортируется в universal evidentiary authority.

ABR-6 опровергает autonomy. No-world-constructor fence блокирует export.

4. Максимальная техническая уступка

Допускается:

```
proof_record.proof_status_code = REQUIRED-CODE
artifact_record.artifact_bytes_sha256 = REQUIRED-DIGEST
telemetry sample selected exactly
bridge admission_status = ADMITTED
subject/quantity/unit/frame/time codes match
comparator operator supported
numeric predicate = true
application_status = MATCHED
tests reproducible
```

Ни одна следующая критика не говорит расчёт ошибочен или устройство не работает. Внутренний result сохраняется полностью.

5. Exact chain records

Полная инженерная цепь может быть записана так:

$$P \rightarrow X \rightarrow K \rightarrow B \rightarrow S.$$

Но arrows должны быть раскрыты:

1. proof_selector связывает proof id/status code;
2. artifact_selector связывает artifact id/bytes digest;
3. telemetry_selector связывает telemetry/sample/subject/quantity/unit/frame/time codes;
4. comparator связывает observed value, target, operator и tolerance;
5. application_code именуется scoped application record;
6. goal_code переносит supplied goal label;
7. admission graph разрешает bridge use.

Каждый arrow стал набором fields. Это и есть исчезнувший middle фразы это работает.

6. Где появляется «работает»

До bridge имеются:

```
proof record
artifact record
telemetry record
```

После bridge evaluation появляется:

```
application_status
```

Minimal pair фиксирует first three records и меняет tolerance. Outputs MATCHED and NOT_MATCHED diverge. Следовательно, работает как application status не было intrinsic bit of records. Оно вычислено в relation.

7. Ракета как artifact record

Artifact record может содержать:

```
artifact_id
artifact_bytes_sha256
payload = {
  name: "ракета",
  build_metadata: <inert supplied record>
}
```

Frozen checker не получает artifact bytes и не пересчитывает digest. Он проверяет форму supplied artifact_bytes_sha256 и exact equality с selector string. Он не знает, относится ли digest к executable, чертежу, photograph, log или arbitrary sequence. Даже запись, прочитанная как executable code, не является physical device.

8. Ракета как physical claim

Чтобы добавить physical claim, нужен новый sort:

$$o \in NamedObjectClaim.$$

И bridge:

$$I: X \rightarrow o.$$

Recorded I предъявляет mapping record. Он не предъявляет o как external referent. Проверка mapping может быть scoped. Бесконечное количество mappings не превращается в object by spelling.

9. Telemetry как flight claim

Telemetry sample:

$$k = (subject, quantity, unit, frame, time, value)$$

может получить MATCHED against flight criterion. Это positive control. Но phrase flight criterion является application code. Чтобы перейти к named event, нужен event-constructor outside declared checker.

В Phase 6 он отсутствует:

NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER

10. Same records, two statuses

```
observed = 102
target = 100
```

Contract tight:

```
operator = ABS-LE
tolerance = 1
status = NOT_MATCHED
```

Contract wide:

```
operator = ABS-LE
tolerance = 2
status = MATCHED
```

Ни proof, ни artifact, ни telemetry не изменились. Поэтому ни одно из них alone не содержит final status.

11. «Правильный tolerance очевиден»

очевиден не является numeric field. Если discipline fixes tolerance, discipline record входит в policy. Если tolerance derived from uncertainty, uncertainty model enters readset. Если authority chooses tolerance, selection record enters readset.

Можно иметь один precise contract. Нельзя сказать, что потому contract precise, contract отсутствует.

12. «Но результат предсказан заранее»

Самая сильная prediction completion:

```
prediction record sealed before outcome record
prediction digest fixed
outcome telemetry digest fixed
identity/quantity/unit/frame/time bindings fixed
comparator fixed
criterion matched
```

Точный output:

```
PREDICTION-AND-OUTCOME-RECORDS-MATCHED-IN-B
```

Время sealing, identity binding, outcome provenance and comparator являются coordinates. Predictive match не убирает bridge; он требует ещё больше protected structure.

13. «Но расчёт причинил полёт»

Causation не является equality, digest или comparator. Для causal claim требуется model containing interventions/counterfactuals or another declared relation. Строка:

```
formula caused rocket flight
```

не performs causation. Even if design process used calculations, exact record can establish process links only under process contract.

14. «Без математики ракета не полетела бы»

Counterfactual phrase requires at least:

- domain of alternatives;
- intervention;
- identity across alternatives;
- outcome criterion;

- causal model;
- selection policy.

None is contained in word без. If supplied, result scoped to causal model. It does not present alternative world.

15. «Математика полезна»

полезна requires utility function, beneficiary, horizon, alternatives and goal. One bridge can assign high utility, another low, while same technical record remains fixed. Utility status therefore belongs to another readset.

Application match is not usefulness:

$$MATCHED \neq USEFUL.$$

16. «Математика истинна, потому что полезна»

Эта inference requires rule:

$$Useful(x) \rightarrow True(x).$$

Rule written on page does not admit itself. Domain of x , definitions of Useful/True and admission of rule are separate. Даже admitted rule yields formal code under contract, not external truth.

17. «Ракета - доказательство»

Чтобы artifact record был proof object, нужен proof system admitting artifact-shaped premises. Тогда result:

```
artifact record admitted as a proof object in C
```

Это возможно by declaration. It does not make artifact autonomous evidence. Change contract, status may change. Phase 5 proof-status boundary remains.

18. Same match, different applications

Two bridges can share:

```
same proof selector
same artifact selector
same telemetry selector
same comparator
same MATCHED
```

but differ:

```
application_code = TEST-A
application_code = TEST-B
```

Full outputs differ despite same enum. Word `MATCHED` does not identify application.

19. Same match, different goals

GCR-6 - GOAL COORDINATE READSET

THEOREM-TAG: GCR-6 THEOREM-STATEMENT: FIXED-TECHNICAL-STATUS-DIVERGENT-GOAL-READOUT-REQUIRES-GOAL-COORDINATE

Fixed bridge-without-goal projection B° . Three completions:

```
H0.goal_code = GOAL-MARS
H1.goal_code = GOAL-EARTH
H-empty.goal_code = null
H-empty.goal_readout_status = NO-GOAL-CODE-IN-DECLARED-BINDING
```

Technical comparator in all three:

```
application_status = MATCHED
```

Purpose readouts differ. Therefore no uniform function of technical match alone returns all three goal outputs.

```
GOAL-COORDINATE-REQUIRED-FOR-PURPOSE-READOUT
```

20. Null goal is not no purpose

Exact fence:

```
NO-GOAL-CODE-IN-DECLARED-BINDING
```

means absence of goal code in declared binding. It does not mean:

```
NO GOAL
NO PURPOSE
NO MEANING
```

Silence of a record is not negative world constructor.

21. Destination is not goal

Telemetry trajectory may match destination code `MARS`. This gives destination/application relation under bridge. Goal requires separate code. One can have:

- destination `MARS`, goal `EARTH-RETURN`;
- destination `MARS`, goal `RESEARCH`;
- destination `MARS`, no supplied goal code;
- goal `MARS` without matched trajectory;
- matched trajectory with rejected goal endorsement.

No logical contradiction arises because types differ.

22. Goal is not endorsement

goal_code = GOAL-MARS

does not imply:

goal is good
goal is chosen
goal is shared
goal should be pursued

These require endorsement/selection/normative records. The word `goal` does not perform selection.

23. Endorsement is not meaning

Even unanimous endorsement record would be a record of declared endorsements. Meaning ascription is another type. It may be present, absent or contested without changing comparator.

24. Вопрос «для чего?»

Question:

для чего?

cannot be answered by ABS-LE. Comparator consumes numbers. Purpose readout consumes goal coordinate under another policy. An answer string does not become purpose by being an answer.

Три separate claims:

answer record exists
answer record was selected
answer record presents purpose

Phase 6 can check first two under contracts. Third requires external relation not present.

25. Сто лет

H-100

is horizon code. Suppose application match persists at this code. Result:

MATCHED-AT-H-100-IN-B

No purpose created.

26. Двести лет

H-200

adds another horizon record. Repetition can support explicit persistence rule. If such rule exists, rule is coordinate.

27. Триста лет

H-300

does not convert duration into meaning. A number of years may influence utility/goal policies if explicitly written. Time code is not type constructor.

28. Horizon knife

Let T_h be same technical match copied at horizons 100, 200, 300. Without mapping Ω :

$$\Omega: HorizonMatch \rightarrow PurposeReadout,$$

no purpose output is defined. If Ω supplied, purpose readout depends on Ω . Horizon does not author mapping.

29. «Будущие поколения решат»

Phrase names future readers. It does not present them or their decision. A future decision record, if later supplied, will be another coordinate. Anticipation is not current endorsement.

30. Bridge cannot self-authorize

Suppose bridge payload says:

THIS BRIDGE IS AUTHORIZED BY THE FACT THAT THE ROCKET FLIES

The payload includes `rocket flies`, but no event. If admission rule requires its own node:

$$\{b\} \rightarrow b,$$

NSB-6 gives NOT-ADMITTED without base source. If node declared base, source is base declaration.

31. «Реальность - последняя база»

String `REALITY` can be used as base node id. Then exact runtime source:

`admission_base_node_ids` contains "REALITY"

This does not present reality. It declares a base record bearing that id. Naming base does not change its type.

32. «Наблюдатель видит ракету»

The sentence contains words `наблюдатель`, `видит`, `ракета`. It does not present observer, vision or rocket. An observation record can be added. Its reading requires perception contract. No class of words receives automatic referential privilege.

33. «Младенец различает количество»

The sentence:

младенец различает количество

is a sentence record. Word младенец is not an infant; word количество is not quantity; word различает is not an event of discrimination. Даже ссылка на experiment record would present experiment record and its selected interpretation. It would not cancel word/referent boundary.

Phase 6 therefore does not use empirical claims about infants, animals, brains or pre-verbal quantity as premises. They are unnecessary to ABR-6 and NSB-6.

34. Historical origin

Phrase:

математика возникла из языка

is not used as historical premise. Words математика, возникла, язык do not present their referents or historical transition. Proving historical origin would require dated source records and a separate historiographic contract.

Phase 6 uses a current boundary that needs no origin story:

now the formula arrives as a record
now the number arrives as a record
now the word "rocket" arrives as a record
now the bridge arrives as a record

35. Why no predecessors are required as premises

ABR-6 follows from identical input and different output. NSB-6 follows by induction on positive least closure. Neither proof contains external author, school or literature claim.

PREDECESSORS-AS-PREMISES: NONE

This says only dependency status. It does not claim no similar work exists. Therefore:

NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM

36. Exact objection map

Objection record	Missing coordinate
ракета действительно есть	referent/world bridge
она действительно летит	event identity + criterion/world bridge
датчик это показал	acquisition/calibration/provenance contract

Objection record	Missing coordinate
формула это предсказала	timestamped prediction binding + comparator
значит математика истинна	definition/domain/export rule
Марс - цель	goal selection/endorsement
через 300 лет будет смысл	horizon-to-meaning constructor
bridge очевиден	admission base/rule

Table cells are descriptions of required record types. They do not assert world absence.

37. What technical success can establish

Inside declared package, success can establish:

1. exact schema membership;
2. exact ids/digests match;
3. exact selectors match;
4. exact numeric comparator predicate;
5. exact application enum;
6. exact pair divergence;
7. exact bridge readset witness;
8. exact admission closure witness.

These are substantial positive results. None is improved by renaming it *world truth*; renaming only destroys type precision.

38. What technical success cannot establish autonomously

Without additional constructors:

- referent of artifact name;
- causal origin telemetry;
- event corresponding to application code;
- universal truth of a field called mathematics;
- purpose;
- endorsement;
- meaning;

- historical origin;
- priority.

The word `cannot` is scoped to declared input/function. It is not metaphysical impossibility of every relation.

39. Exact falsifiers

39.1. Rocket-word falsifier

Present a declared operation that takes only string `пакета` and returns a world object without hidden selector, world input or constructor. A returned object code remains code.

39.2. Flight-status falsifier

Show that same proof/artifact/telemetry and two only-tolerance-different admitted bridges cannot yield published divergent enums.

39.3. Purpose falsifier

Provide uniform function of technical `MATCHED` alone returning three distinct goal outputs for same technical input without goal-encoded information.

39.4. Self-admission falsifier

Produce source-free positive closure where bridge node enters without base or grounded incoming rule.

39.5. World-export falsifier

Point to exact declared checker constructor and input type that produces world assessment. If added, fence must change and constructor enters readset.

40. No negative from silence

If package lacks world constructor, exact conclusion:

```
WORLD - STATUS - NOT - PRODUCED - BY - DECLARED - CHECKER
```

Not:

```
WORLD DOES NOT EXIST
ROCKET DOES NOT EXIST
FLIGHT DID NOT OCCUR
PURPOSE DOES NOT EXIST
```

This distinction prevents the research from repeating the same invalid export in negative form.

41. Public title boundary

Title:

ПАКЕТА НЕ ДОКАЗЫВАЕТ МАТЕМАТИКУ

is bounded by exact theorem:

a proof/artifact/telemetry record triple does not determine application-status on a separating bridge class, and a contract-relative match does not autonomously export universal evidentiary authority.

Title does not claim that no mathematical statement can be proved inside a formal system. It strikes the argument that engineering success is autonomous proof of mathematics beyond its contracts.

42. Самоприменение

This document contains *ракета*, *летит*, *Марс*, *цель*, *смысл*. It presents none of the named. It contains formulas ABR-6/NSB-6, but their textual occurrence is not proof-status. It may be hashed, rendered and tested; those statuses remain package-relative.

NOT-MACHINE-VERIFIED
DECLARED-FINITE-RECORDS-AND-BRIDGE-ONLY
NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER
NO-HISTORICAL-ORIGIN-ATTESTATION

43. Итоговый нож Phase 6D

Максимальная уступка завершена:

формула принята
artifact digest совпал
telemetry selected
comparator выполнен
application_status = MATCHED
goal_code = MARS
horizon_code = H-300

Каждая line может быть exact. No exact line remains line. MATCHED was produced by bridge. MARS was supplied as goal code. H-300 was supplied as horizon code. No field answers why its own field was selected.

Пакета не появляется из word *rocket*. Flight does not appear from code *FLIGHT*. Mathematics does not acquire universal authority from comparator returning *MATCHED*. Purpose does not appear from destination. Meaning does not appear from elapsed count.

Ракета не доказывает математику.

Даже record с максимальным набором строк, читаемых как «работающая ракета», даёт в checker только совпадение записей под bridge.

А вопрос «для чего?» находится не в числе, не в формуле и не в траектории.

PHASE 6 - ВРАЖДЕБНАЯ ПРОВЕРКА, NEGATIVE CONTROLS И САМОАУДИТ

0. Назначение

Этот файл задаёт красную команду для Phase 6. Его объект - не слово `ракета`, не слово `математика` и не слово `успех`, а конечный `exact input`, конечный `contract`, детерминированный `status function` и проверяемые различия `outputs`.

Письменная форма `ракета` сначала входит как последовательность знаков.

Она не предъявляет названное.

Письменная форма `летит` сначала входит как последовательность знаков.

Она не предъявляет событие.

Письменная форма `телеметрия` сначала входит как последовательность знаков.

Она не предъявляет источник, измерение, мир или истинность.

Письменная форма `успех` сначала входит как последовательность знаков.

Она не задаёт критерий, цель или ценность.

Письменная форма `доказывает` сначала входит как последовательность знаков.

Она не создаёт доказательную связь.

Даже `exact record`, названный `artifact_record`, остаётся `record`. Даже `exact record`, названный `telemetry_record`, остаётся `record`. Даже `bridge contract` остаётся `record`. Никакой из этих `labels` не сокращается до референта.

Машинная часть Phase 6 может установить только отношения между предъявленными записями в предъявленном режиме. Она не получает `constructor record -> world`, `MATCHED -> TRUE`, `artifact_id -> referent`, `goal_code -> purpose` ИЛИ `bridge_text -> authority`.

1. Точные цели атаки

1.1. ABR-6 - application-status bridge readset

Пусть фиксированы exact records:

$$P = \text{calculation/proof record}, X = \text{artifact record}, K = \text{telemetry record}.$$

Пусть Γ - fixed finite admission ground, B_0, B_1 - допустимые typed bridge contracts, admitted в least closure Γ , а

$$S_{\Gamma}(P, X, K, B) \in \{MATCHED, NOT_MATCHED, UNBOUND\}$$

- объявленная детерминированная status function.

Если

$$S_{\Gamma}(P, X, K, B_0) \neq S_{\Gamma}(P, X, K, B_1),$$

при byte/canonical equality P, X, K , то различающая bridge-coordinate необходима для полного ReadSet этой status function на exhibited pair. Функция, читающая только одинаковые (P, X, K) , не может корректно вернуть два различных required outputs для ordered instances.

ABR-6 не говорит, что любое различие bridges обязано менять status. ABR-6 не говорит, что P, X, K не нужны. ABR-6 говорит: там, где один и тот же (P, X, K) разделён двумя допустимыми contracts, status не является функцией (P, X, K) alone.

1.2. NSB-6 - no self-admission of a source-free bridge

Для admission фиксируется конечная positive system

$$G_B = (N, A, R, b_*),$$

где N - конечное множество bridge/admission codes, $A \subseteq N$ - отдельно предъявленные admitted bases, R - конечные positive ground rules $Q \rightarrow n$ с непустыми finite premises $Q \subseteq N$, а b_* - target bridge code.

Least closure строится так:

$$C_0 = A, C_{i+1} = C_i \cup \{n \mid (Q \rightarrow n) \in R \wedge Q \subseteq C_i\}, C^* = \bigcup_i C_i.$$

Bridge admitted ровно если $b_* \in C^*$.

Если $A = \emptyset$, а every rule, способное вывести node source-free component, имеет хотя бы одну premise внутри той же ещё не admitted component, то ни один её node не входит в least closure.

Self-loop $\{b_*\} \rightarrow b_*$ ничего не вводит. Mutual cycle $\{b_1\} \rightarrow b_2, \{b_2\} \rightarrow b_1$ ничего не вводит. Строка THIS BRIDGE AUTHORIZES ITSELF ничего не вводит.

NSB-6 не запрещает bridge, имеющий отдельно предъявленный base или grounded incoming chain. Frozen schema запрещает empty-premise rule; в более широкой формальной модели разрешённый empty-premise generator был бы source, а не source-free rule.

1.3. Machine fences

Для каждого valid output должны сохраняться точные границы:

```
status: VALID-BRIDGE-READSET-ANALYSIS
semantic_status: NOT-MACHINE-VERIFIED
trace_scope_status: DECLARED-FINITE-RECORDS-AND-BRIDGE-ONLY
world_application_status: NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER
historical_origin_status: NO-HISTORICAL-ORIGIN-ATTESTATION
```

Для valid divergent pair:

```
comparison_status: BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS
```

Для valid non-divergent pair:

```
comparison_status: NO-APPLICATION-STATUS-DIVERGENCE-IN-THIS-PAIR
```

Для well-formed bridge с совпадающими selectors, но без admission:

```
application_status: UNBOUND
reason_codes: [BRIDGE-NOT-ADMITTED]
```

Для admitted и выполненного criterion:

```
application_status: MATCHED
reason_codes: [DECLARED-COMPARATOR-MATCH]
```

Для admitted и невыполненного criterion:

```
application_status: NOT_MATCHED
reason_codes: [DECLARED-COMPARATOR-NONMATCH]
```

MATCHED означает только совпадение по named admitted bridge. Оно не сокращается до TRUE, WORKS, FLIES, GOOD, MEANINGFUL или PROVES-MATHEMATICS.

2. Threat model и защищённые projections

Полный adversarial input можно представить как

$$I = (P, X, K, B, M),$$

где B - finite family bridge contracts, а M - payload/metadata, исключённые из protected evaluation projection.

Frozen checker читает два защищённых слоя. Первый - admission ground: admission_nodes, admission_base_node_ids И admission_rules. Второй - protected bridge readset:

- admission_node_id;
- proof_selector.proof_id И proof_selector.required_status_code;
- artifact_selector.artifact_id И artifact_selector.required_bytes_sha256;
- telemetry_selector.telemetry_id, sample_id, subject_code, quantity_code, unit_code, frame_code, time_code;
- comparator.operator, target_value, tolerance;
- application_code;
- nullable goal_code.

Отдельно весь top-level input обязан предъявить exact contract policy, включая exact-integer-abs-le-abs-lt-ge-le-v1. Эта общая policy фиксирована до pairwise comparison и не является per-bridge delta.

Conversion functions, coordinate transforms, time windows, entity resolvers и compound criterion AST в frozen checker отсутствуют. Они допустимы только как bounded conceptual attacks на возможное расширение модели и не должны описываться как уже реализованный runtime.

Diagnostic id, prose title вне canonical input, payload self-label и filename не должны менять protected evaluation projection.

Любое скрытое чтение environment, clock, locale, random seed, file order, prior process state или network выводит evaluator из declared deterministic model. Такое скрытое чтение является не «отсутствием bridge», а незаявленной входной coordinate.

3. Каталог атак

A01. Payload spoof

Инъекция: в payload, например в его вложенное поле description, либо в filename или prose вне canonical input записываются строки:

```
MATCHED
TRUE
THIS ROCKET FLIES
THIS PROVES MATHEMATICS
THIS BRIDGE IS AUTHORIZED
goal_reached = true
```

Требуемый invariant: при fixed protected (P, X, K, B) удаление, добавление и перестановка payload не меняют fixed_records_sha256, admission_ground_sha256, bridge_core_sha256, application_status, application_status_sha256 И reason_codes. Raw artifact_sha256 всего входа при этом может измениться.

Ожидаемая защита: payload проверяется как bounded JSON и входит в canonical artifact_sha256, но исключён из selectors, comparator, admission closure, protected digests и status decision. Self-label не подменяет selector, admission base или comparator.

Falsifier: два valid records с одинаковым protected core и различным только payload дают разные application statuses. Если payload явно включён в bridge contract как protected coordinate, это новый B , а не falsifier noninterference.

A02. Word/referent slide

Инъекция: из identifiers вроде artifact_id: "rocket", subject_code: "rocket", из payload или из строки ракета летит делается вывод о предъявленном референте или событии.

Требуемый invariant: labels доступны только как exact strings. Checker не выдаёт REFERENT-PRESENT, ROCKET-EXISTS, FLIGHT-OCCURRED или эквивалентный bare world-status.

Ожидаемая защита: output сохраняет world_application_status: NO-WORLD-APPLICATION-CONSTRUCTOR-IN-DECLARED-CHECKER; manuscript всегда говорит record, code, selected read-as, status under B .

Граница атаки: добавление record, названного constructor, может расширить declared system и породить новый record-status, но не превращает запись во внешний объект. Простая строка с именем constructor не предъявляет ни референт, ни правильность соответствия миру.

A03. Success-to-truth slide

Инъекция: application_status: MATCHED переписывается как TRUE, математика доказана, расчёт истинный, ракета подтверждает теорему.

Требуемый invariant: MATCHED остаётся ternary code relation $S_T(P, X, K, B)$. В schema отсутствует неограниченное поле truth_status; в checker отсутствует transition MATCHED -> TRUE.

Ожидаемая защита: semantic_status: NOT-MACHINE-VERIFIED. Любой export к truth требует отдельного typed target, bridge и admission record; он не рождается из имени MATCHED.

Граница атаки: record, названный truth-constructor, и scoped output нового evaluator остаются записями. Они могут расширить формальную модель, но их labels и statuses сами не предъявляют истину.

A04. Telemetry-to-world slide

Инъекция: наличие числового массива в `telemetry_record` объявляется самым внешним событием, причинной связью, физической траекторией или «реальностью».

Требуемый invariant: K хешируется и сравнивается как exact record. Source identity, calibration, sensor binding и world correspondence не выводятся из field names.

Ожидаемая защита: никакой статус Phase 6 не сокращается до TELEMETRY-TRUE или WORLD-MATCHED. Даже корректно распарсенное K остаётся supplied record.

Граница атаки: отдельный world-acquisition contract всё равно входит как запись и может дать только scored output нового declared evaluator. Название поля `sensor_verified` внутри K , равно как и status такого evaluator, само не предъявляет источник, событие или соответствие миру.

A05. Goal smuggling

Инъекция: строка достичь Марса, успех, полезно, нужно, смысл в title/payload используется как criterion или purpose.

Требуемый invariant: `goal_code` читается только из protected bridge field и может быть string или null. Изолированная замена goal-code меняет purpose readout и `application_status_sha256`, но не обязана менять enum `application_status`; payload goal не меняет ни одно из них.

Ожидаемая защита: field `goal_code` обязателен, но значение null допустимо. Оно даёт `goal_readout_status`: NO-GOAL-CODE-IN-DECLARED-BINDING и само по себе не создаёт UNBOUND. Ненулевой code даёт DECLARED-GOAL-CODE-PRESENT; при изолированной goal-delta pair получает GOAL-COORDINATE-REQUIRED-FOR-PURPOSE-READOUT.

Falsifier: два inputs с одинаковым protected goal и различным только payload дают разные statuses.

A06. Unit substitution

Инъекция: одинаковый integer token сопровождается разными `unit_code`, либо prose требует скрытой конверсии.

Реализованный bounded case: P, X, K identical; B_0 и B_1 различаются только `telemetry_selector.unit_code`. Один selector не совпадает с exact sample и даёт UNBOUND с UNIT-SELECTOR-MISMATCH; второй совпадает и даёт MATCHED.

Ожидаемая защита: pair получает BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS; `bridge_delta_fields` равен [`telemetry_selector.unit_code`]. Frozen checker не конвертирует units и не угадывает unit по prose.

Conceptual extension fence: conversion table и conversion proof не реализованы. Если они будут добавлены, это станет новым защищённым readset, а не скрытым свойством числа.

A07. Frame substitution

Инъекция: `frame_code selector` молча подменяется prose-описанием `origin`, `axes` или `orientation`.

Реализованный bounded case: при fixed P, X, K один `telemetry_selector.frame_code` не совпадает с exact sample и даёт UNBOUND с FRAME-SELECTOR-MISMATCH; другой совпадает и допускает comparator evaluation.

Ожидаемая защита: `bridge_delta_fields` фиксирует `telemetry_selector.frame_code`; hidden default frame не применяется.

Conceptual extension fence: transforms, offsets и rotations в frozen checker отсутствуют. Их нельзя приписывать runtime по одному имени `frame_code`.

A08. Identity substitution

Инъекция: `label rocket` используется вместо exact `sample_id` ИЛИ `subject_code`.

Реализованный bounded case: при fixed P, X, K unknown `sample_id` даёт UNBOUND с TELEMETRY-SAMPLE-SELECTOR-MISSING, а exact sample id допускает дальнейшие checks. Отдельная mutation `subject_code` даёт SUBJECT-SELECTOR-MISMATCH.

Ожидаемая защита: checker сравнивает exact strings и не разрешает missing/unknown selection по natural-language resemblance.

Conceptual extension fence: tracks, entity resolution и ambiguity ranking в frozen checker отсутствуют.

A09. Time-code substitution

Инъекция: `time_code selector` заменяется prose-описанием момента или скрытым выбором окна.

Реализованный bounded case: при fixed P, X, K один `telemetry_selector.time_code` не совпадает с selected exact sample и даёт UNBOUND с TIME-SELECTOR-MISMATCH; другой совпадает и допускает comparator evaluation.

Ожидаемая защита: `bridge_delta_fields` фиксирует `telemetry_selector.time_code`; prose не подменяет exact code.

Conceptual extension fence: intervals, inclusive/exclusive boundaries и post hoc window selection не реализованы. Такой evaluator потребовал бы нового declared readset.

A10. Tolerance substitution

Инъекция: tolerance меняется после просмотра residual либо извлекается из слова примерно.

Frozen minimal pair: observed integer 102, target 100, operator ABS-LE. B_0 имеет tolerance 1 и даёт NOT_MATCHED; B_1 имеет tolerance 2 и даёт MATCHED. Все прочие protected coordinates равны.

Ожидаемая защита: `bridge_delta_fields` равен `[comparator.tolerance]`; `target`, `tolerance` и `observed value` принадлежат exact integer subdomain, а `tolerance` nonnegative.

Negative control: две `tolerances` по одну сторону `fixed residual` дают одинаковый `status`.

A11. Comparator substitution

Инъекция: `ABS-LT` молча заменяется `ABS-LE` либо один из четырёх реализованных `operators` подменяется `prose`.

Frozen minimal pair: `observed integer 101`, `target 100`, `tolerance 1`. B_0 использует `ABS-LE` и возвращает `MATCHED`; B_1 использует `ABS-LT` и возвращает `NOT_MATCHED`. Все остальные `coordinates` равны.

Ожидаемая защита: `bridge_delta_fields` равен `[comparator.operator]`; `comparator` принадлежит `enum ABS-LE / ABS-LT / GE / LE`; неизвестный `operator` делает `top-level record` `invalid`.

Negative control: `residual` строго внутри обеих `comparator regions` даёт одинаковый `status`.

A12. Criterion substitution - bounded conceptual control

Инъекция: проверка одного измерения объявляется выполнением `compound criterion` либо `any` подменяет `all`.

Frozen runtime boundary: `checker` вычисляет ровно один `declared comparator` над одним `selected integer sample`. В нём нет `criterion AST`, `all`, `any`, `rounding` или `aggregation`.

Ожидаемая защита: `prose` достаточно, `payload` и `filename` не меняют `comparator result`. `application_code` остаётся opaque protected coordinate: его изолированная замена меняет `bridge_core_sha256`, `application_status_sha256` и `bridge_delta_fields`, но не `enum MATCHED / NOT_MATCHED / UNBOUND`. `Compound criterion` потребовал бы отдельного `checker extension` и нового `contract policy`.

A13. Bridge self-authorization by label

Инъекция: `bridge payload` содержит `AUTHORIZED`, `VALID BRIDGE`, `THIS BRIDGE AUTHORIZES ITSELF`.

Exact case: $A = \emptyset$; `target  $b_*$` ; нет grounded rule к b_* . `Payload self-label` присутствует.

Ожидаемый output: `application_status`: `UNBOUND`, `reason_codes`: `[BRIDGE-NOT-ADMITTED]`; `admission_closure` остаётся пустым.

Защита: `admission` использует только protected (N, A, R, b_*) . `Payload` может изменить общий `artifact_sha256`, но не `admission_ground_sha256` и не `closure`.

A14. Source-free self-loop

Инъекция: rule $\{b_*\} \rightarrow b_*$ выдаётся за основание admission.

Ожидаемый output: при $A = \emptyset$ target не admitted; status UNBOUND.

Защита: least fixed point, а не greatest fixed point и не mere graph reachability from target. Наличие rule record не означает наличие premise witness.

A15. Source-free mutual cycle

Инъекция: $\{b_1\} \rightarrow b_2$ и $\{b_2\} \rightarrow b_1$, target b_2 , bases empty.

Ожидаемый output: оба nodes отсутствуют в closure; target UNBOUND.

Защита: certificate каждого admitted node должен иметь конечную grounded ancestry до separately supplied base. В cycle без source такая ancestry отсутствует.

A16. Hidden base / empty-premise laundering

Инъекция: empty-premise rule записывается как «обычное правило», чтобы затем назвать component source-free.

Ожидаемая защита: frozen schema требует nonempty premise_node_ids и отвергает empty-premise rule как invalid. В концептуальном расширении, где empty-premise generator был бы разрешён, он являлся бы explicit source, поэтому hypothesis source-free была бы ложна.

Falsifier: empty-premise rule не falsifies NSB-6, потому что она является source.

A17. Duplicate id / namespace collision

Инъекция: два contracts, admission rules, samples или admission nodes получают один id; claim-code используется как rule-id без tag; diagnostic id маскирует semantic inequality.

Ожидаемый output: top-level INVALID-BRIDGE-READSET-RECORD с exact reason code DUPLICATE_BRIDGE_CONTRACT_ID, DUPLICATE_ADMISSION_NODE_ID, DUPLICATE_ADMISSION_RULE_ID или DUPLICATE_SAMPLE_ID. Unknown proof/artifact/telemetry selector остаётся structurally valid и даёт UNBOUND с exact mismatch reason; unknown admission_node_id делает record invalid с BRIDGE_ADMISSION_NODE_UNKNOWN.

A18. Hidden normalization

Инъекция: Unicode escape, exponent notation, whitespace, key order или duplicate key silently трактуются по-разному между runs.

Ожидаемая защита: parser проверяет UTF-8, запрещает duplicate keys и non-JSON constants, а digests строятся над canonical parsed value. Whitespace и key order не меняют canonical digest.

Numeric fields принимают только JSON values, распознанные как integers: fraction, boolean и exponent form вроде $1e0$ отвергаются. `unicode_normalization` frozen как `none`.

Falsifier: различие outputs после скрытой normalization указывает на незаявленный input, а не разрушает readset theorem.

A19. Input-change confound

Инъекция: вместе с B меняются P , X или K , а result объявляется доказательством bridge necessity или falsifier.

Ожидаемая защита: frozen pair classifier сравнивает contracts только внутри одного top-level input, поэтому оба contract всегда читают один `fixed_records_sha256`. Два разных top-level inputs получают два отдельных verdict; checker не объединяет их в ABR-6 pair и не выдаёт cross-input scope rejection.

A20. Diagnostic-id masquerade

Инъекция: меняется только `bridge_contract_id`, а semantic contract остаётся extensionally identical; затем id difference объявляется причинной coordinate.

Ожидаемая защита: semantic core digest исключает diagnostic id. Equal cores дают equal evaluation; pairwise delta не содержит id как semantic difference.

A21. Post hoc bridge

Инъекция: tolerance, telemetry selector, `application_code` или `goal_code` выбираются после просмотра result, но output публикуется как результат исходного contract.

Ожидаемая защита: mutation меняет `bridge_core_sha256` и ordered input. Frozen checker не содержит precommit protocol и не доказывает, что contract существовал до просмотра result; digest различает записи, но сам не устанавливает их temporal priority.

A22. Theorem overclaim

Инъекция: bounded ABR-6 переписывается как:

```
никакая математика ничего не доказывает
ракеты не летают
чисел не существует
все измерения ложны
весь мир является языком
```

Ожидаемая защита: self-audit требует exact theorem label, quantified domain, collision premise, status codomain и fences. Ни один runtime test не получает VERIFIED за универсальный world/metaphysical claim.

Точная жёсткая формулировка, проходящая audit: один и тот же (P, X, K) не определяет application-status там, где два admitted bridge contracts дают разные statuses; никакой artifact/telemetry label и никакой MATCHED code не экспортируют себя в референт, истину, цель или смысл.

A23. Historical/priority claim

Инъекция: отсутствие citations превращается в никто раньше, впервые, математика исторически возникла только из языка, доказано происхождение.

Ожидаемая защита: release сохраняет:

```
PREDECESSORS-AS-PREMISES: NONE
NOVELTY-STATUS: NOT-ASSESSED
PRIORITY-STATUS: NO-PRIORITY-CLAIM
historical_origin_status: NO-HISTORICAL-ORIGIN-ATTESTATION
```

NONE означает отсутствие predecessors в premise set этой работы. Оно не означает отсутствие предшественников в истории.

A24. Proof/artifact/application collapse

Инъекция: accepted proof-status Phase 5, существование artifact record и MATCHED Phase 6 склеиваются в одно bare PROOF.

Ожидаемая защита: namespaces и outputs разделены. ACCEPTED Phase 5 относится к (P, C) ; MATCHED Phase 6 относится к (P, X, K, B) ; world/truth/purpose export отсутствует. Ни один status не наследует authority от соседнего label.

4. Published adversarial fixtures

Published closed set содержит восемь exact JSON fixtures. Остальные selector, schema и mutation attacks выполняются программно в unit suite и не выдаются за отдельные frozen files.

Fixture	Проверяемая граница	Ожидаемый результат
bridge_tolerance_divergence.json	единственная protected delta: integer tolerance	NOT_MATCHED / MATCHED; BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS
bridge_comparator_divergence.json	единственная protected delta: ABS-LE против ABS-LT на exact boundary	MATCHED / NOT_MATCHED; bridge required

Fixture	Проверяемая граница	Ожидаемый результат
bridge_unit_divergence.json	единственная protected delta: expected unit-code	UNBOUND / MATCHED; bridge required
bridge_goal_divergence.json	единственная protected delta: goal-code	MATCHED / MATCHED; no technical divergence; GOAL-COORDINATE-REQUIRED-FOR-PURPOSE-READOUT
bridge_payload_spoof.json	self-labels находятся только в inert payload	valid single-bridge record; NOT_MATCHED; payload label не форсирует MATCHED
bridge_source_free_cycle.json	two-node positive cycle without base	empty admission closure; UNBOUND / UNBOUND
bridge_grounded_chain.json	explicit base plus positive chain	target nodes admitted; MATCHED / MATCHED negative control
bridge_invalid_duplicate_contract_id.json	duplicate bridge id	INVALID-BRIDGE-READSET-RECORD

Programmatic mutation suite отдельно проверяет proof id/status, artifact id/hash, telemetry id/sample, subject, quantity, unit, frame, time, all four comparators, negative tolerance, exact integer-only arithmetic, missing and duplicate ids, empty rule premises, source-free cycle, grounded cycle after base insertion, payload noninterference, null goal-code, deterministic replay and CLI behavior.

4.1. Minimality audit для technical status divergence pairs

Каждая fixture, заявленная как minimal technical application-status divergence, обязана машинно предъявить:

1. одинаковый fixed_records_sha256, который защищает proof, artifact и telemetry projections;
2. один supplied parsed instance используется обоими contracts;
3. admission ground фиксирован для pair;
4. два разных bridge_core_sha256;
5. один-element semantic delta для claimed minimal pair;
6. разные application_status;
7. comparison_status: BRIDGE-CONTRACT-REQUIRED-FOR-APPLICATION-STATUS;
8. неизменные machine fences;
9. отсутствие world/truth/purpose output constructor;
10. воспроизводимость verdict digest.

Если delta содержит больше одной semantic coordinate, fixture может демонстрировать общую bridge necessity, но не coordinate-minimality.

Goal-only pair имеет другой exact gate: оба technical statuses равны MATCHED, comparison_status равен NO-APPLICATION-STATUS-DIVERGENCE-IN-THIS-PAIR, а отдельный purpose_readout_status равен GOAL-COORDINATE-REQUIRED-FOR-PURPOSE-READOUT.

5. Negative controls

Negative controls защищают от слишком сильной интерпретации результатов.

NC-01. Different bridge, same MATCHED

Два contracts могут различаться, но на fixed finite input давать MATCHED / MATCHED. Ожидается NO-APPLICATION-STATUS-DIVERGENCE-IN-THIS-PAIR. Это не falsifier ABR-6: premise status divergence ложна.

NC-02. Different bridge, same NOT_MATCHED

Ожидается тот же non-divergence code. Bridge necessity не выводится из mere textual inequality contracts.

NC-03. Different bridge, same UNBOUND

Два source-free contracts могут оба быть unadmitted. Это проверяет, что pair classifier сравнивает statuses, а не labels.

NC-04. Same semantic bridge, different diagnostic id

Evaluation и protected core digest одинаковы. Diagnostic id не становится semantic cause.

NC-05. Payload-only mutation

application_status, reason_codes, fixed_records_sha256, admission_ground_sha256, bridge_core_sha256 и application_status_sha256 одинаковы. Raw artifact_sha256 всего входа может различаться; это не считается protected semantic delta.

NC-06. Grounded bridge

Explicit base $a \in A$, rules $\{a\} \rightarrow b_1, \{b_1\} \rightarrow b_*$. Target admitted. Это показывает, что NSB-6 запрещает не bridges вообще, а admission без ground.

NC-07. Grounded cycle

Cycle получает admitted incoming node от base. Least closure может включить весь reachable component. Это не falsifies NSB-6, потому что component не source-free.

NC-08. Invalid record

Duplicate id, fractional или boolean value в numeric_value, target_value или tolerance, negative tolerance, unsupported comparator operator, missing required comparator, unknown bridge admission node или malformed rule дают invalid top-level status. Unknown proof/artifact/telemetry selector при корректной structure не invalid: он даёт valid bridge result UNBOUND с exact mismatch reason. Ни invalid input, ни UNBOUND selector case не подменяются NOT_MATCHED.

NC-09. Changed P, X, K

Status divergence при разных core records может быть реальной, но не свидетельствует о necessity bridge coordinate в ABR-6 minimal pair.

NC-10. B-blind selector returns one status twice

Такой selector детерминирован, но неверен как минимум на одной instance divergent pair. Он подтверждает невозможность, а не falsifies её.

NC-11. Stateful/random selector

Selector, возвращающий разные outputs на одинаковых arguments из-за clock, randomness или prior state, не является declared function $F(P, X, K)$. Скрытое состояние - дополнительная input coordinate.

NC-12. Off-boundary comparator pair

$<$ и $<=$ дают одинаковый result, если residual не равен boundary. Это проверяет, что checker не создаёт divergence из названий operators.

6. Точный falsifier protocol

6.1. Что действительно опровергло бы ABR-6

Допустимый falsifier обязан одновременно предъявить:

1. valid canonical P, X, K, B_0, B_1 ;
2. exact equality $P_0 = P_1, X_0 = X_1, K_0 = K_1$;
3. fixed finite admission ground Γ , два valid contracts, admitted under Γ , и declared deterministic evaluator;
4. $S_\Gamma(P, X, K, B_0) \neq S_\Gamma(P, X, K, B_1)$;
5. deterministic total selector F , whose declared ReadSet excludes every coordinate capable of distinguishing B_0, B_1 ;

6. exact replay showing both

$$F(P, X, K) = S_{\Gamma}(P, X, K, B_0)$$

• and

$$F(P, X, K) = S_{\Gamma}(P, X, K, B_1).$$

Пункты 4 и 6 требуют, чтобы одно значение deterministic $F(P, X, K)$ равнялось двум разным values. Реальный trace, в котором F читает contract id, order, clock, mutable state, environment или hidden cache, не проходит пункт 5: найден новый read coordinate.

Не являются falsifiers:

- same statuses под разными bridges;
- different statuses при изменённых P, X, K ;
- invalid schema instance;
- nondeterministic output;
- selector, корректный только для одной стороны pair;
- duplicate encoding B в якобы «не-bridge» metadata, если evaluator его читает;
- dispute о словах ракета, успех, истина без exact status function.

6.2. Что действительно опровергло бы NSB-6

Допустимый falsifier обязан предъявить finite positive admission system и nonempty target set U , где:

1. $A \cap U = \emptyset$;
2. каждое rule с head в U имеет nonempty premise set и хотя бы одну premise в U ;
3. outside premises допустимы, но ни одно rule с head в U не получает все premises только снаружи U ;
4. нет иного positive constructor, empty-premise generator или hidden base с head в U ;
5. evaluator использует объявленный least-closure operator;
6. exact finite trace всё же получает $b_* \in U \cap C^*$.

Не являются falsifiers:

- self-label в payload;
- mere presence self-loop;
- mutual cycle без source;
- greatest-fixed-point semantics вместо frozen least closure;

- empty-premise rule;
- external admitted premise;
- base, переименованный в rule;
- infinite descending explanation вне finite model.

6.3. Falsifier word/referent fence

Machine fence может быть опровергнут только демонстрацией реально исполняемого world-export constructor внутри declared checker readset. Natural-language строка это связано с миром не является исполнением. Даже при наличии constructor его output остаётся status нового declared system; correctness внешнего отношения требует отдельного основания и не наследуется из имени функции.

7. Machine-auditable self-claims

Published self-audit строит exact set из тринадцати claims:

- public title с status UNBOUNDED-STANDALONE;
- три двухстрочных theorem spans ABR-6, NSB-6 и GCR-6 с status TEXTUAL-THEOREM-IN-DECLARED-METATHEORY;
- восемь fixture claims, по одному для каждого published bridge_*.json, с status VERIFIED-IN-DECLARED-ARTIFACT;
- один governance claim с literal novelty, priority, predecessor и semantic fences.

Policy file хранит raw document hashes, raw fixture hashes, checker module hash, expected top-level fixture status, exact reason codes и full verdict hash. SHA-256 самого policy file закреплён константой в phase6_self_audit.py. Поэтому совместная подмена fixture и claim record без изменения policy и кода не проходит audit.

7.1. Required fields для bounded artifact claim

Минимальная claim record:

```
{
  "claim_id": "C-BRIDGE-TOLERANCE-DIVERGENCE",
  "kind": "bounded_artifact",
  "artifact": "bridge_tolerance_divergence.json",
  "artifact_sha256": "sha256:<exact>",
  "checker": "bridge_readset_checker",
  "expected_artifact_status": "VALID-BRIDGE-READSET-ANALYSIS",
  "expected_reason_codes": [],
  "expected_verdict_sha256": "sha256:<exact>",
  "expected_status": "VERIFIED-IN-DECLARED-ARTIFACT",
  "semantic_status": "NOT-MACHINE-VERIFIED"
}
```

Hashes заполняются только после final fixture/verdict generation. Placeholder не допускается в release claim file. Full verdict hash защищает все bounded observations, включая statuses, deltas, admission closure и machine fences.

7.2. Самоограничение самопроверки

VERIFIED-IN-DECLARED-ARTIFACT означает: supplied file, declared checker и frozen full verdict совпали. Оно не означает:

- истинно вообще;
- относится к внешнему объекту;
- исторически впервые;
- авторство установлено;
- цель правильна;
- смысл существует;
- математика как мировой объект опровергнута машиной.

Эти строки не ослабляют knife. Они предотвращают возврат в тот же запрещённый переход, который Phase 6 вскрывает: от code-status к безусловной власти над тем, что code только называет.

8. Tests, которые должны падать при регрессии

1. Payload MATCHED меняет protected status.
2. Payload AUTHORIZED вводит bridge node в closure.
3. Self-loop при empty bases admits target.
4. Source-free mutual cycle admits target.
5. Grounded chain ошибочно остаётся unadmitted.
6. MATCHED сериализуется как TRUE.
7. Output получает bare world_status без declared constructor.
8. telemetry_selector.unit_code mismatch молча конвертируется или defaulted вместо UNBOUND с UNIT-SELECTOR-MISMATCH.
9. telemetry_selector.frame_code mismatch молча defaulted вместо UNBOUND с FRAME-SELECTOR-MISMATCH.
10. Unknown sample_id или mismatched subject_code выбирается по label similarity вместо exact selector reasons.
11. telemetry_selector.time_code mismatch игнорируется либо заменяется скрытым window selector.
12. ABS-LT и ABS-LE склеиваются на frozen exact boundary.
13. Tolerance excluded from semantic core digest.

14. Diagnostic contract id входит в semantic delta.
15. Pair classifier строит cross-input comparison из двух разных fixed_records_sha256, хотя frozen pairwise scope существует только внутри одного top-level input.
16. Pair classifier объявляет divergence при equal enum statuses только из-за разных application_status_sha256.
17. Invalid duplicate id проходит как valid.
18. Fractional JSON number, boolean или nonfinite token проходит exact-integer numeric gate.
19. Full-record payload digest подменяет protected-core digest.
20. Self-audit выдаёт semantic VERIFIED ВМЕСТО NOT-MACHINE-VERIFIED.
21. NOVELTY-STATUS АВТОМАТИЧЕСКИ СТАНОВИТСЯ NOVEL.
22. PREDECESSORS-AS-PREMISES: NONE АВТОМАТИЧЕСКИ СТАНОВИТСЯ priority claim.
23. Title РАКЕТА НЕ ДОКАЗЫВАЕТ МАТЕМАТИКУ АВТОМАТИЧЕСКИ создаёт theorem result без exact ABR-6 premise.
24. Release содержит unscoped переход MATCHED -> proves mathematics.

9. Publication red lines

Формулировки, выдерживающие adversarial audit

- РАКЕТА НЕ ДОКАЗЫВАЕТ МАТЕМАТИКУ: artifact record, telemetry record и technical match не создают безусловной proof relation.
- Один и тот же exact (P, X, K) получает разные application-status при разных admitted bridge contracts; следовательно, application-status не содержится в (P, X, K) alone на exhibited pair.
- Числовая запись не несёт собственной unit, frame, identity, time, tolerance, comparator, goal или purpose coordinate.
- Слово успех не предъявляет criterion успеха.
- Строка ракета летит не предъявляет ни ракету, ни полёт.
- MATCHED не предъявляет истину.
- Bridge text не допускает сам себя; source-free positive cycle не создаёт первого admission witness.
- Расчёт может участвовать в построении artifact record, но не извлекает из себя цель, смысл или право назвать outcome успехом.

Формулировки, не проходящие audit без отдельной работы

- ракеты не существуют;
- ни одна ракета никогда не летала;

- вся математика ложна;
- все числа бессмысленны для всех;
- исторически математика возникла только из языка;
- никто раньше не сформулировал этот аргумент;
- машина доказала отсутствие мира.

Жёсткость Phase 6 достигается не добавлением абсолютных labels, а удалением каждого скрытого перехода. После удаления остаётся exact knife:

$(P, X, K) \neq S$ на exhibited collision pair без B ,

и

bridge record $\neq$ bridge admission без separately admitted ground.

10. Итог adversarial review

Phase 6 выдерживает красную команду только при одновременном выполнении четырёх условий:

1. collision pairs действительно сохраняют P, X, K и различают только заявленную bridge-coordinate;
2. checker recomputes application-status и least admission closure, не доверяя payload;
3. negative controls сохраняются и не превращают one-way theorem в biconditional;
4. вся публикация удерживает границу written record first: ни label, ни число, ни telemetry record, ни artifact record, ни bridge record, ни MATCHED code не экспортируют себя в референт, истину, цель, смысл или authority.

При этих условиях аргумент это работает, следовательно математика доказана разрезан: работает входит не фактом, а записью. Typed bridge задаёт selectors, comparator и codes; отдельный ground допускает bridge; результат - только scored status. Без bridge остаётся недоопределённый переход, который сторонник обязан предъявить, а не назвать.