The F* proof checker for Z3: principles, implementation and current state

As of 2026-10-31 (UTC)

Technical report, as of 2026-10-31 (UTC). Source baseline: master `8fbb275ab` (`8fbb275aba0472e8 62e289c7af1331b3bd5d512c`, committed 2026-10-31 15:38:56 UTC). Links to current files are relative links in the Markdown version of this report and GitHub links pinned to `8fbb275ab` in the LaTeX version; line numbers refer to `8fbb275ab`. Work that is not on master (open pull requests, a draft pull request and an unpushed development commit) is labelled where it appears and linked through pinned commit URLs. Appendix A reports a corpus experiment that ran on older binaries (`deef01f5b`); its numbers describe those binaries, not master.

The audience is an engineer who will change or review the checker. The report explains what an `ACCEPT` means, how a Z3 proof travels from the solver to a verdict, which parts are proved in F* and which are trusted, what is merged and what is still draft, and which proof rules the F* build corpus leaves pending.

Contents

1 Purpose and scope **2**

2 Design principles **3**

3 Implementation flow **4**

3.1 Input, solver and callback 4

3.2 Native views and marshaling 4

3.3 Route choice 5

3.4 The legacy document checker 5

3.5 The sort-indexed document checker 5

3.6 Verdicts and diagnostics 6

4 Theory coverage **6**

5 Formal assurance **8**

5.1 The legacy global contract is uninhabited 8

5.2 Inhabited guarantees on master 8

5.3 The sort-indexed document theorem is conditional 8

5.4 Draft #11498: an inhabited typed root (not merged) 9

6 Extraction, runtime and build **9**

7 Current status **10**

8 Reproduction and development workflow **11**

9 Limitations **12**

A Pending rules in the F* build corpus **12**

A.1 Data and method 12

A.2 Outcomes per backend 13

A.3 Units 13

A.4 The 13 families 14

A.5 Readable examples 15

A.6 Timeouts, crashes and unexecuted tails 17

A.7 What the appendix shows and does not show 17

B Evidence files **17**

1 Purpose and scope

The checker certifies individual UNSAT answers of Z3. Z3 solves a query with proof production enabled. When the answer is `unsat`, the checker reads the proof Z3 built, replays it in a verified F* program, and prints one of three verdicts:

- `ACCEPT`. The proof document reaches the closed root `false` from the original assertions, every node was checked by an executable rule, and no obligation is left. The intended guarantee is `ACCEPT => UNSAT`: the assertions have no model.
- `PARTIAL n`. The document is well formed and reaches `false`, but `n` proof nodes were kept with an open obligation because the active route's checker did not decide them. Such nodes may have premises; they are not only DAG leaves. This is not a refutation certificate.
- `REJECT`. The document has an invalid shape or a step that fails a check that must pass (for example an `asserted` node that is not an input).

A pending obligation means the checker did not decide that step. It does not mean the step is false. A rule can be pending because no checker exists for it, because a budget ran out, because the route in use does not read the step's certificate, or because the step is wrong; the verdict does not say which. Answers other than `unsat` give no proof to check. `sat`, `unknown`, timeouts and crashes produce no accepted proof and say nothing about the checker's soundness or completeness.

The design aims at soundness (no `ACCEPT` for a satisfiable input) and accepts incompleteness. Every limit in the checker turns an undecided step into `PARTIAL` or `REJECT`, never into `ACCEPT`. How much of the intended guarantee is a proved theorem, and under which assumptions, is the subject of section 5. In short, on master the ground core, typed quantifier instantiation and a restricted binder fragment have inhabited node-level theorems for the legacy route. The legacy global model contract is uninhabited, so the legacy kernel theorem does not provide the intended soundness assurance. The sort-indexed route, which F* queries take, has a document theorem whose quantifier hypotheses master does not construct. A draft pull request adds an inhabited typed root for that route that the native verdict does not use.

The checker sources live in this directory. PROOF_CHECKING.md is the module-level reference, OCAML_CHECKER_REPORT.md the long engineering log, and ARITHMETIC_CERTIFICATE_REPORT.md the arithmetic certificate notes. The Pulse checkers in this directory (`Z3PulseProofChecker.fst`, `Z3PulseJustificationChecker.fst`, `Z3PulseLra*.fst`) are verified regression developments. They are listed under `tests` in tools/checker_modules.json and are not linked into the native executables (PROOF_CHECKING.md). This report covers the extracted native path only.

2 Design principles

Untrusted producer, verified consumer. Z3 is a candidate generator. It supplies the proof DAG and hints such as Farkas coefficients, LP data and binder certificates. The checker replays each claim against the actual premises and conclusion. The foreign interface states the boundary in its header: "No foreign operation constructs checker expressions or checked facts" (Z3Native.fsti). The OCaml adapter's comment on LP coefficients reads "These are candidate coefficients, not a trusted replay result" (ocaml_runtime/ Z3Native.ml).

Executable replay of explicit certificates. Checking functions are total F* functions (`Tot`), extracted to native code. Semantic facts are separate ghost (`GTot`) definitions that the executable code cannot call (section 6).

Fail closed. On the legacy route a rule name with no checker becomes `Unverified name` and contributes one pending obligation (Z3NativeProof.fst). An exhausted budget leaves the node with a pending obligation. A malformed native node is a marshaling error (`ERROR`, exit status 2); an unsupported judgment or document shape is `None`, printed as `REJECT`. The design rule is that missing evidence never becomes acceptance. The dispatch code enforces it, and section 5 states how much of it is proved.

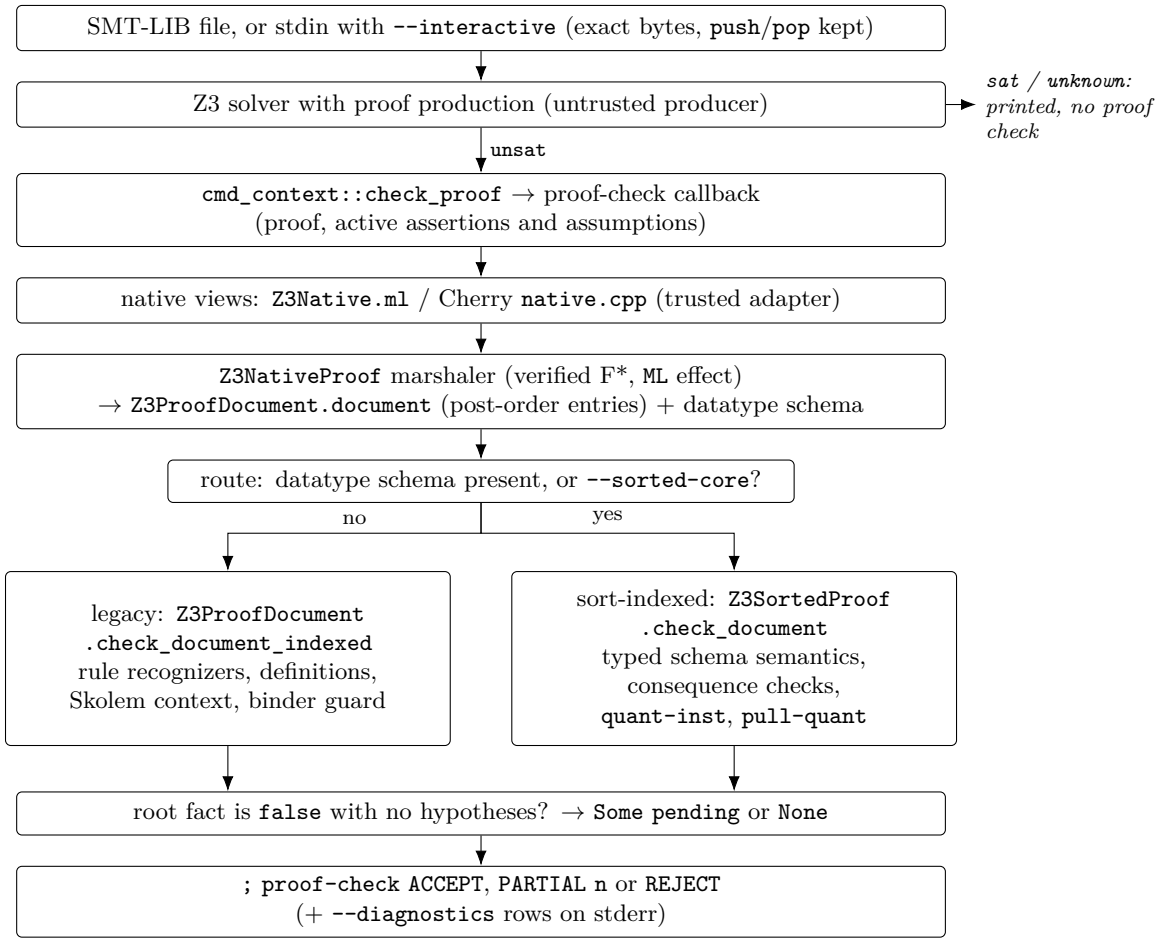
Typed canonical declarations. The marshaler assigns operator tags from Z3's native operator kinds, not from printed names, and gives opaque declarations unique names (Z3NativeProof.fst). Numerals are checked as they are read. Two declarations that print the same name stay distinct.

Original assertions, premises and scopes are preserved. The assertions a proof may use are exactly the inputs that Z3's command context passes to the callback: the active assertions of the current `push`/`pop` scope and the current assumptions (section 3.1). An `asserted` node must match one of them. A premise may only refer to an earlier document entry. The interactive driver sends commands one at a time to one Z3 context, so incremental scopes are Z3's own and the checker never flattens a session into one query.

One interpretation for the whole proof. The document theorems quantify over a single interpretation shared by every node of a proof (Z3SortedProof.fst). A step cannot pick its own model.

Explicit trust boundary and non-vacuous model contracts. Everything outside F* verification is listed as trusted in section 6. A semantic contract only counts as a guarantee when F* also proves that some interpretation satisfies it. Section 5 applies that test to the contracts on master. The legacy `model_sound` provably has no model; the quantifier hypotheses of the sort-indexed document theorem have no inhabitation proof on master.

3 Implementation flow



3.1 Input, solver and callback

In file mode the OCaml driver parses the whole input with `parse_smtlib2_string`, asserts it, runs `solver_check` and fetches the proof (`solve_with_clause_proofs`, ocaml_runtime/Z3Native.ml). The context is created with `proof=true` and the solver timeout. The Cherry driver does the same with the C++ API (cherry_runtime/main.cpp).

`--interactive` is the mode that preserves an incremental F* session. `read_command` frames one complete command at a time from stdin, handling comments, quoted symbols and strings, without parsing SMT-LIB (ocaml_runtime/Z3NativeMain.ml). Each command goes to `Z3_eval_smtlib2_string` on one persistent context (`interactive`, ocaml_runtime/Z3NativeMain.ml; Cherry cherry_runtime/main.cpp). After `reset` the driver re-enables proof production, and a `set-option` that turns proofs off is undone and reported as an error. The process exit status is 2 if any command failed, else 1 if any proof was rejected, else 3 if any was partial, else 0.

The driver registers a callback with `Z3_set_proof_check_callback` (../src/api/z3_api.h). After an unsatisfiable `check-sat`, `check-sat-assuming` or `check-sat-using`, `cmd_context::check_proof` collects the inputs and calls it (../src/cmd_context/cmd_context.cpp). The inputs are the tracked assertions of the current scope, with a named assertion passed as `name => assertion` plus `name`, followed by the check's assumptions. The callback's report is printed as `; proof-check ...` and `; proof-stats ...` comment lines. Z3 does not interpret the report and does not change the `unsat` answer.

3.2 Native views and marshaling

The foreign adapter is trusted code. Z3Native.fsti declares abstract handles and view functions. The OCaml implementation `view_proof` (ocaml_runtime/Z3Native.ml) reads the rule name, the proof-sorted arguments

as premises and the last argument as conclusion. It also reads arithmetic hints: Farkas and LP coefficients become candidate weights (L426-L453), the producer's `arith assumed <origin>` marker is recorded (L454-L457), and `th-lemma` nodes with an LP hint but no coefficients are renamed `arith/<hint>`, with `lp-context` renamed `def-axiom` (L458-L462). The Cherry adapter (cherry_runtime/native.cpp) applies the same renaming. The corpus families `arith/lp-monomial-interval` and `arith/lp-monomial-substitute` in Appendix A get their names here; no F* source mentions them.

The marshaler Z3NativeProof.fst is verified F* in the `ML` effect. It walks the native DAG with explicit worklists (`marshal_expression_jobs`, L169; `marshal_proof_jobs`, L423), caches expressions and proofs by native AST id, and emits entries in post-order so that premises are indices of earlier entries. `native_rule` (L227-L244) maps the view to a rule: an `assumed` origin becomes `Unverified "assumed:<origin>"`, Farkas weights are accepted only on `th-lemma`, and a commutativity step is re-checked before it is labelled as one. `quant-intro`, `nnf-pos` and `nnf-neg` become `QuantifierIntroduction` with a hidden binder certificate and an empty premise list (L287-L300, L477-L479). `quant-inst` keeps its instantiation terms, and `arith/lp-gomory` is rebuilt from the LP context certificate or left `Unverified` (L479-L495). `marshal_with_datatypes` returns the document together with the native datatype schema (L534).

The document type is Z3ProofDocument.fst: declarations, expressions, assertions, entries `{rule; formula; premise_ids}` and a root index. Judgments are `Holds e`, `Equal a b` and `Observational a b`; a checked fact is a conclusion plus the open hypotheses it depends on (Z3UnifiedProof.fst).

3.3 Route choice

Both drivers pick a route per proof. If the marshaled schema has datatype cases, or `--sorted-core` was given, the document goes to `Z3SortedProof.check_document`. Otherwise it goes to the legacy `Z3ProofDocument.check_document_indexed` (ocaml_runtime/Z3NativeMain.ml, Z3Cherry.fst, cherry_runtime/main.cpp). `--stats` reports the route as `semantics=legacy`, `sorted-core` or `sorted-datatype`. There is no option that sends a datatype document to the legacy route, and the sort-indexed route has no legacy fallback (PROOF_CHECKING.md).

F* encodes its terms with SMT-LIB datatypes, so F* queries take the sort-indexed route. In the corpus run of Appendix A every checked query on both backends reported `semantics=sorted-datatype`.

3.4 The legacy document checker

Z3ProofDocument.fst prepares and checks one entry at a time in post-order. It filters declarations (`logical_decls`), validates scopes for documents with binders, and collects definitions from `intro-def` and `def-axiom` entries (`collect_definitions`, L53-L68). It expands fresh-bit reconstructions only when every input is within a 65,536-node budget (`document_definitions`, L93-L108). Skolem definitions are collected and ordered. Each entry is prepared (definition expansion, judgment classification) and checked by `check_node_with_membership` (Z3UnifiedProof.fst), with premises looked up only among earlier ids in an id table.

`check_node_with_membership` is where the rule kinds are dispatched. An `asserted` node must be an input. `hypothesis` introduces an open hypothesis, and `lemma` discharges it from a premise `false`. `def-axiom`, rewrite-like rules, `sk`, binder introductions and `quant-inst` must be premise-free and contribute 0 or 1 pending obligation depending on their recognizer. Theory lemmas, Farkas, Gomory and PAC certificates go through `checked_theory_lemma` (L1069-L1082). Core rules (`refl`, `symm`, `trans`, `monotonicity`, `mp`, `unit-resolution` and similar) must check or the node is rejected.

`finish_document` and `finish_table` require the root to be `Holds false` with no hypotheses and the defined symbols to be fresh in the inputs (`D.check_fresh_inputs`), and then return the summed pending count (Z3ProofDocument.fst).

3.5 The sort-indexed document checker

Z3SortedProof.fst checks a document against a validated datatype schema with typed semantics. `check_document` (L1070-L1079) rejects an invalid schema or ill-typed assertions, checks the entries, and applies the same root test as the legacy route.

The per-node function `check_node_cached` (L954-L979) reads less of a node than the legacy route does. It

distinguishes five rule shapes. `asserted` must be premise-free and an input; `hypothesis` opens a hypothesis; `lemma` with a premise `false` discharges hypotheses; `quant-inst` is checked by `Z3SortedQuantifier.check`; `pull-quant` by `check_flatten`. Every other node is checked the same way, whatever its rule: the conclusion must follow from the premise conclusions by `consequence` (L734-L736: direct consequence and constructor-premise propagation with fuel 8) or by `DT.consequence` in Z3SortedDatatype.fst. If neither decides it, the node is kept with one pending obligation. `prepare_entry_from` (L1040-L1046) does no definition expansion.

So Farkas weights, Gomory and PAC certificates, binder certificates and Skolem definitions are not consulted on this route. A binder introduction arrives with no premises, because its premise is inside the certificate, and must be decided as a premise-free consequence. Draft PR #11498 proves this reading property as a lemma (pinned Z3SortedRootDocument.fst#L181-L247). It follows that a rule name in the sort-indexed diagnostics names a node the sort-indexed procedure did not decide. It is not evidence that the legacy recognizer of that rule failed, because that recognizer did not run.

3.6 Verdicts and diagnostics

`Some 0` prints `ACCEPT`, `Some n` prints `PARTIAL n` and `None` prints `REJECT` (`result_text`, ocaml_runtime/Z3NativeMain.ml; cherry_runtime/main.cpp). In file mode the exit codes are 0 `ACCEPT`, 1 `REJECT`, 2 error, 3 `PARTIAL`, 4 `SAT` and 5 `UNKNOWN` (OCAML_CHECKER_REPORT.md).

`--diagnostics` runs a diagnosing twin of the checker instead of the plain one and takes the verdict from it. F* proves that the twin's result equals the plain checker's (`Z3ProofDocument.fst` L585-L865, `Z3SortedProof.fst` L1180-L1273). The driver then prints `diagnostic checked pending=N families=K` and one row per group, `obligation <rule> <count> <first-node> <fingerprint>` (ocaml_runtime/Z3NativeMain.ml; cherry_runtime/main.cpp). The rule name comes from `diagnostic_rule` (L7-L45). `th-lemma` covers both unverified theory lemmas and finite-field PAC steps, and the three binder rules report the binder certificate's root name. The fingerprint is lossy on purpose. It stops at depth 5 or 96 nodes and prints at most 8 arguments per application. Base sorts other than `Bool`, `Int` and `Real` (bit-vector, datatype and uninterpreted sorts) become `U`, while array sorts keep their index and element sorts to depth 2. Uninterpreted names are aliased to `u0`, `u1` and so on, rational numerals longer than 64 characters become `large-numeral`, and bit-vector values and binder sorts are dropped (L65-L121). It groups obligations; it is not a readable statement of the condition.

4 Theory coverage

The table lists what the legacy route decides, under which guard, and what semantic assumption the decision rests on. "`model_sound`" is the legacy global contract (Z3UnifiedProof.fst), which is uninhabited (section 5.1). A family whose backing is `model_sound` has a decision procedure and a node theorem whose assumption is uninhabited, so the theorem does not provide the intended soundness assurance.

Family	Decision code	Fragment and guards	Semantic backing on master
Boolean core and EUF (`refl`, `symm`, `trans`, `monotonicity`, `mp`, `unit-resolution`, `and-elim`, `not-or-elim`, `iff-true`, `iff-false`, `iff~`)	`relational_step`, `logical_step` (Z3UnifiedProof.fst)	canonical connectives only	inhabited typed core model, per-node theorem `lemma_core_check_node_sound` (Z3CoreKernel.fst)
Bounded bit-vector evaluation	`Z3BvExhaustive.valid` (Z3BvExhaustive.fst)	premise-free `th-lemma` and `rewrite`, closed ground terms, at most 512 nodes and 12 leaf bits (L64-L66)	core model through `lemma_bv_theory_lemma` (Z3Core Kernel.fst)
Other BV rewrites (`Z3BvAddNormalize`, `Z3BvExtract`, `Z3BvFresh`)	recognizers in `relational_step`	per-recognizer bounds	`model_sound`
Quantifier instantiation	`Z3QuantifierInstantiation.check` (Z3QuantifierInstantiation. fst)	closed arguments whose sorts match the binders, capture-avoiding substitution	typed quantifier model, `lemma_typed_check_node_sound` (Z3QuantifierKernel. fst)

Family	Decision code	Fragment and guards	Semantic backing on master
Binder introduction (`quant-intro`, `nnf-pos`, `nnf-neg`)	`binder_step` (Z3UnifiedProof.fst), scope guard `binder_scoped` (L987-L999)	typed theorem only for closed inputs, core rules and exact forall/exists introductions (Z3BinderKernel.fst); guard limits in Z3UnifiedProof.fst	restricted fragment: typed quantifier model; the rest: `model_sound`
Skolemization (`sk`)	`Z3Skolemization.check` (Z3Skolemization.fst)	one Hilbert-choice tuple per block, acyclic, fresh in the inputs	`model_sound` plus the Skolem context model
Boolean normalization, `def-axiom`, `intro-def`	`D.tautology 128` (Z3Definitions.fst), Boolean search	fuel and size budgets	`model_sound`
`rewrite`	disjunction of recognizers (Z3UnifiedProof.fst): identity, flattening, `distinct`, arrays, arithmetic, factoring, BV extract, quantifier reduction, lambdas, conditionals, integers, division, vacuity, theory normalization, Boolean search	per-recognizer budgets	`model_sound`, except the bounded BV case
Linear arithmetic, Farkas	`LA.check` (Z3LinearArithmeticAst.fst) with producer weights as candidates	Int and Real linear; integer strict-bound rounding not applied to Real	`model_sound` (numeric model)
LP replay, Gomory and integer cuts	Z3LpReplay.fst, Z3LpGomory.fst, Z3LpIntegerCut.fst	replay at most 32 literals, 16 rounds, 128 rows (PROOF_CHECKING.md); cuts at most 32 premises, 128 rows, 8 cuts, 64 terms, 4,096 work units, magnitude 2^128 (Z3LpIntegerCut.fst)	`model_sound`
`arith/lp-monomial-*`, `assumed:nlsat`	none on the legacy route; `checked_theory_lemma` accepts only `arith/lp-integer-tighten` among `Unverified` tags (Z3UnifiedProof.fst)	always pending there	none
Arrays	`Z3ArrayLemma.array_lemma` (Z3ArrayLemma.fst), `Z3ArrayRewrite`, `Z3ArrayEquality`	one-dimensional scalar index and element sorts	`model_sound`, whose array clauses are inconsistent for Real elements (section 5.1)
Datatypes	sort-indexed route only: Z3SortedDatatype.fst, schema validity Z3DatatypeSignature.fst	arrays containing datatypes are rejected as constructor fields by `supported_fields` (Z3DatatypeSignature.fst); sequences of datatypes are rejected earlier by the native adapter (ocaml_runtime/Z3Native.ml, an `ERROR`)	typed sort-indexed semantics; document theorem conditional (section 5.3)
Finite fields	`FF.exhaustive_clause`, `FF.equivalent` (Z3FiniteFieldLemma.fst), PAC replay (Z3FiniteFieldPac.fst)	single modulus, bounded assignments; PAC steps needing a disequality inverse witness stay pending (PROOF_CHECKING.md)	`model_sound` (normalization model)
Floating point, strings, sequences, regular expressions	no native kernel branch; operators arrive as opaque declarations, except binary sequence concatenation, which is tagged (`OP_SEQ_CONCAT`, ocaml_runtime/Z3Native.ml)	congruence only	none

On the sort-indexed route the table does not apply. That route decides Boolean and equality reasoning, datatype constructor, recognizer and accessor reasoning, quantifier instantiation and `pull-quant`, and integer operations through Z3SortedInteger.fst and Z3SortedArithmetic.fst. Real arithmetic is abstract on that route,

and no legacy theory recognizer runs on it.

5 Formal assurance

5.1 The legacy global contract is uninhabited

The legacy kernel theorem `lemma_check_node_sound` (Z3UnifiedProof.fst) assumes `model_sound i` (Z3UnifiedProof.fst). That predicate conjoins Boolean, arithmetic, array, quantifier, lambda, normalization and numeric contracts stated over all environments of an untyped value domain. Three of them are refuted outright in Z3ScopedModelTests.fst, each lemma proving `False` from the contract:

- `boolean_model` asks every Bool-sorted term to evaluate to a Boolean under every environment. Under an environment that maps variable 0 to an integer, the Bool variable `Var 0` evaluates to that integer (`lemma_legacy_boolean_contract_impossible`, L11-L16).
- `numeric_model` asks the same of Int-sorted terms; a Bool-valued environment refutes it (`lemma_legacy_integer_contract_impossible`, L18-L23).
- `array_model` asks `select` arguments to be arrays; an array-sorted variable under a Bool-valued environment refutes it (`lemma_legacy_array_contract_impossible`, L25-L35).

Closed terms fail as well. In Z3CoreModelTests.fst the array-store, numeral and integer-coercion clause instances at `(store a 0 1/2)` with `a : (Array Int Real)` are jointly unsatisfiable (`o1_clash`), while each pair of them is satisfiable (`drop_s`, `drop_n`, `drop_i`). The untyped binder clause fails at an ill-typed value list: instantiated at `[true]`, it makes the valid closed formula `forall x:Int. x + 0 = x` false in every model of the quantifier, arithmetic and equality contracts (`o2_unfaithful`, Z3CoreModelTests.fst).

F* accepts conditional statements from these assumptions, so `lemma_check_node_sound` holds vacuously. The implication is vacuous: it does not provide the intended soundness assurance for any legacy `ACCEPT`. The legacy route also has no document-level soundness theorem. Every family in the table of section 4 whose backing is `model_sound` therefore has no non-vacuous formal guarantee today. This is the main open formal problem of the checker, and no merged or draft change replaces `model_sound`.

5.2 Inhabited guarantees on master

- **Typed core (#11450).** `core_model i dom` (Z3CoreModel.fst) covers typing, canonical connectives, typed equality, integer `+` and `*`, and ground bit-vectors. `lemma_core_inhabited` (L318) proves a witness exists, and the classical ghost model `Z3GhostStandardModel` is another witness (`lemma_ghost_inhabited`, Z3GhostStandardModel.fst). `lemma_core_check_node_sound` (Z3CoreKernel.fst) is a per-node theorem for the core rules and bounded BV.
- **Typed quantifier instantiation (#11450).** `quantifier_model` (Z3TypedQuantifier.fst) lets quantifiers range exactly over value lists typed by their binders. The ghost standard model realizes it (`lemma_quantifier_model`, Z3GhostStandardModel.fst), and `lemma_typed_check_node_sound` (Z3QuantifierKernel.fst) covers instantiation steps with no pending obligation.
- **Restricted binder fragment (#11493).** The scope guard `binder_scoped` (Z3UnifiedProof.fst) and the structural theorems of Z3BinderScope.fst support `lemma_typed_binder_check_node_sound` (Z3BinderKernel.fst) for closed inputs, core rules and exact forall/exists introductions. `nnf` steps without sorts, `sk`, `rewrite`, `th-lemma` and `def-axiom` inside certificates are outside this fragment.

These are node-level theorems for fragments of the legacy route. A `Some` result of a node check does not by itself mean zero obligations, and no merged document theorem combines them into a statement about a native `ACCEPT`.

5.3 The sort-indexed document theorem is conditional

`Z3SortedProof.lemma_check_document` (Z3SortedProof.fst) states that if `check_document` returns `Some 0` then the assertions are not all true in the given typed interpretation. Its hypotheses include `Q.forall_`

`elimination_model` when the document contains instantiation steps, and `Q.forall_model` when it contains `pull-quant` steps (L1166-L1167); a document with neither needs no quantifier hypothesis. No lemma on master constructs an interpretation that satisfies those hypotheses, so for documents with instantiation or `pull-quant` steps the theorem is relative to an assumption that master's source does not construct. That is a statement about master's proofs, not a claim that the hypotheses are false or that the theorem is vacuous.

5.4 Draft #11498: an inhabited typed root (not merged)

PR #11498 is open and in draft state, head `497dc106537fadc7633f78d9c91ac4dbfd7baf4b`, base `8fbb275ab`. Nothing in this subsection is on master.

- `Z3TypedRoot.fst` gives typed carriers: `bool`, `int`, `FStar.Real.real`, full ghost function spaces for arrays, an arbitrary nonempty model-chosen type for each other base sort, and checked constructor trees for datatypes (pinned Z3TypedRoot.fst#L1-L60). `root_model` joins six laws, R1 quantifiers, R2 operators, R3 lambdas, R4 constructors, R5 recognizers and R6 accessors (pinned Z3TypedRoot.fst#L693-L778). `lemma_standard_root` and `lemma_root_inhabited` prove that a standard root model exists for every carrier family and every choice of free symbols (pinned Z3TypedRoot.fst#L910-L930).
- `Z3SortedRootDocument.fst` derives the two quantifier hypotheses of section 5.3 from R1 and proves `lemma_root_check_document`: if `check_document` returns `Some 0`, the assertions are not all true in any root model (pinned Z3SortedRootDocument.fst#L77-L176).
- `check_root_document` is a gate that reuses the unchanged sort-indexed decision and returns `RootCertified`, `RootUncovered`, `RootPending n` or `RootRejected`. `RootCertified` also requires a supported schema and covered, typed document terms. `lemma_root_certified` and `lemma_standard_certified` state the certified result for every root model and for the standard one (pinned Z3SortedRootDocument.fst#L273-L300, #L369-L393).
- The gate is off the default route. The source comment says "No route calls it." At the pinned head, the Root modules are listed only under `tests` in `checker_modules.json`, so they are verified only in builds with `-RunTests` and are never extracted (pinned checker_modules.json#L24-L32), and the drivers still call `Z3SortedProof.check_document`. A native sort-indexed `ACCEPT` is therefore not a `RootCertified` verdict, with or without this PR.

The draft theorem is relative. It quantifies over the class of root models, which is a metatheoretic choice: it does not prove that every SMT-LIB model of the input is a root model, and it trusts the native translation of assertions, schema and datatype metadata. The sort-indexed decision procedure does no Real, array, bit-vector, finite-field, sequence or floating-point reasoning. In the root model, Real arithmetic and datatype-free arrays are interpreted, while bit-vector, finite-field, sequence and floating-point operators are free choices. The legacy `model_sound` is unchanged and still uninhabited, and the default route is unchanged.

The draft also moves semantic assertions out of the native test harness. With the typed datatype semantics made ghost, `concrete_datatype_model` in `tools/test_native_bridge.ml` can no longer run them, so the draft removes seven executable `require` checks and proves the same facts, with exact expected values and refuted wrong-value mutants, in a new F* test module on the same `Tree`/`Forest` fixture (pinned Z3DatatypeNativeModelTests.fst#L1-L36). On master these remain executable runtime checks.

6 Extraction, runtime and build

Ghost semantics, total checking. The interpretation fields and `eval` in Z3AstTheory.fst are ghost (`GTot`), so models may decide classical propositions. Checking functions are `Tot` or `ML`, and F* rejects ghost results in executable code. Proofs that run an executable evaluator under a ghost valuation use ghost twins (PROOF_CHECKING.md).

Two backends from one source. The OCaml backend extracts the verified modules with Custard (`--codegen OCaml --custard_split`) and links them in dependency order. The Cherry backend generates C for the kernel and a C++ driver; it needs a Cherry-capable F* and GCC/G++ 15. Both executables link static Z3 and solve, marshal and check in one process; their options, incremental protocol, diagnostics and exit codes are the same (README.md). CI pins F* `v2026.10.25` for OCaml and `nightly-2026-10-30` for Cherry, OCaml 5.3.0 and GCC 15 (../.github/workflows/pulse-checker-build.yml).

#11478 (merged). This change bumped F*, inlined the option binds with `[@@custard_inline]` (Z3Proof Index.fst), made list helpers tail-recursive (Z3List.fst), boxed the document and marshaler state records with `[@@cherry_boxed]`, and replaced the Cherry proof-id caches with immutable arrays (`Z3IdTable-c.cherry.fst`). Its description reports (not reproduced here) that all 15 previously crashing CI inputs complete under the default 8 MB stack and under a 1 MB stack, without stack-size tweaks.

#11493 (merged). The binder scope guard walks expressions and sorts with explicit worklists (Z3Unified Proof.fst). Its own comment limits the claim: "This is not a stack bound for the whole checker." It also added a trusted runtime binding for `list_of_string` (ocaml_runtime/fStar_String.ml).

Trusted base. Beyond Z3 itself, the trusted code is the F* extraction and the two runtimes: the files of `ocaml_runtime/` and `cherry_runtime/`, in particular `prims.ml`, `Custard_Prims.ml`, `fStar_List_Tot_Base.ml`, `fStar_String.ml`, `fStar_UInt.ml`, `fStar_Pervasives.ml`, the adapters `Z3Native.ml` and `cherry_runtime/native.cpp`, `kernel.c`, and the drivers `Z3NativeMain.ml`, `main.ml` and `cherry_runtime/main.cpp`. The adapters reflect Z3 ASTs into views; the marshaler that builds checker terms from those views is verified. The OCaml driver reports `Stack_overflow` and `Out_of_memory` as `ERROR` with exit status 2 (ocaml_runtime/Z3NativeMain.ml). The Cherry driver reports `std::bad_alloc` the same way (cherry_runtime/main.cpp) but has no handler for stack exhaustion, which ends the process with a signal. That is consistent with the six Cherry SIGSEGV files of Appendix A.6, whose cause is undetermined.

Finite work and its cost. Many checks carry explicit bounds: the 65,536 node input budget, Boolean tautology fuel 128, sort-indexed consequence fuel 8, the BV and LP limits of section 4, and the binder guard's 1,024 sorts per scope, 64 scopes per entry and `65536 * (n + 1)` work units (Z3UnifiedProof.fst). Exceeding a bound leaves a pending obligation, so these bounds cost completeness, not soundness. `--timeout-ms` bounds solving only; it does not bound marshaling or checking (OCAML_CHECKER_REPORT.md). There is no proved whole-checker bound on time, memory or stack depth.

Test counts are not proofs. Native and CLI test pass counts show that the executables behave as the tests expect on the tests. They add no semantic guarantee beyond section 5.

Output for people. On master the only obligation output is the lossy `--diagnostics` fingerprint of section 3.6. A readable printer exists only as development work (section 7).

7 Current status

As of 2026-10-31, checked with `gh pr view` between 18:58 and 19:34 UTC and a `git fetch` of master at 19:34 UTC.

Item	State	Commit or PR	Notes and remaining work
Native checker, legacy and sort-indexed routes, OCaml and Cherry	merged	master `8fbb275ab`	verdict as in section 3.6
Typed core and quantifier-instantiation guarantees	merged 2026-10-30	#11450, `710b259c4`	inhabited node-level theorems (section 5.2)
Typed integer semantic bridge	merged 2026-10-30	#11462, `511600062`	
Cherry stack fixes, F* bump, Custard extraction, tail helpers, boxed records, immutable Cherry id tables	merged 2026-10-31	#11478, `a398faead`	no whole-checker stack bound
Binder scope guard with stack-safe traversal	merged 2026-10-31	#11493, `8e95e9c47`	restricted binder theorem; an ill-scoped certificate now leaves its node pending instead of being accepted
Legacy global contract `model_sound`	on master, uninhabited		open; no replacement merged or drafted
Sort-indexed document theorem	on master, conditional		quantifier hypotheses not constructed on master
Typed Farkas and Hilbert-epsilon bridges	open, ready for review	#11495, head `7aae2a68c`	not merged; its description reports lower `sk` pending counts on a matched cohort (reported in the PR description, not reproduced)

Item	State	Commit or PR	Notes and remaining work
Sorted constant folding and infix `expression` companion records in `--diagnostics` (OCaml only)	open, ready for review	#11497, head `015f9c56c`	not merged; per its description and file list it adds `expression` records to OCaml `--diagnostics` only, with no `--readable` option and no Cherry driver change (not built or reproduced here)
Typed root model and `RootCertified` gate	open, draft	#11498, head `497dc1065`	not merged; off the default route; native `ACCEPT` does not call the gate
`--readable` obligation printer	unmerged development, no PR	commit `ddaca4d0a` on local branch `fstar-readable-obligations-20261031`, parent `986c626fc`	not cleared for publication (below)
F* build corpus replay	completed experiment	binaries built at `deef01f5b`	Appendix A; not a current-master measurement

Readable printer status. Commit `ddaca4d0a` adds `--readable`, which prints after the machine rows one block per query with a representative obligation of each of the first 20 families, as S-expressions with `NAME#ID` declarations, numerals and binder sorts. It is not on master and not pushed. As of 2026-10-31 it is not cleared for publication:

- A fidelity defect was reproduced. Constant arrays of sort `(Array Int Int)` and `(Array Real Int)` print identically in readable output while their fingerprints differ. The local evidence is the failing test output `C:/z3atomic/.atomic/readable-pending-20261031/tests2/logs/cli_red_const_ocaml.out` and `cli_red_const_cherry.out` in the same directory. A sort-range fix with tests (development build `cand2`) is required before publication and is in progress.
- File-mode parity on a 317-input validation set originally gave OCaml 239 PASS, 77 INCONCLUSIVE, 1 MISMATCH and Cherry 238, 77, 2. All three mismatches are QF_NIA `geo2`/`geo3` inputs. In that original run two were classified as producer proof differences and the Cherry `geo2` row as unexplained, because the proofs of the differing file-mode runs were not captured. A later diagnosis run (`C:/z3atomic/.atomic/readable-pending-20261031/runs/geo-proof-identity/`, 2026-10-31T18:43:02Z to 19:00:13Z) captured the proof of each file-mode run under `gdb`: 126 runs of `geo2` and `geo3` gave 7 distinct proofs, and runs with the same proof gave the same result in every variant. The artifact attributes the variation in the proof to the 2,000 ms `try_for` in the QF_NIA tactic (../src/tactic/smtlogics/qfnia_tactic.cpp), shows it on `geo3`, and did not isolate it on `geo2`. The original three runs cannot be re-hashed, so this explains a mechanism without retroactively proving those rows. This report did not reproduce the capture.
- The validation workflow (`36d35e4f`) has not completed, and parity has not passed. The printer is not cleared for publication.

8 Reproduction and development workflow

Build and test from this directory in a Windows C++20 developer PowerShell (README.md):

```
.\tools\build_ocaml_checker.ps1 -RunTests
```

Add the Cherry backend with `-BuildCherry -CherryFstarExe <Linux-path-to-fstar.exe>` (README.md). A Cherry-only Linux build after installing static Z3 from this checkout (README.md):

```
python3 fstar/tools/build_cherry_checker.py \
  --fstar /path/to/fstar/bin/fstar.exe \
  --z3-install /path/to/z3/install \
  --build-dir /path/to/cherry-checker --run-tests
```

The CLI tests run against a built checker selected by an environment variable (tools/test_native_checker.py):

```
Z3_FSTAR_CHECKER=/path/to/checker python3 fstar/tools/test_native_checker.py
```

To check an incremental SMT-LIB script as F* emits it, send the exact bytes on stdin. This keeps `push`/`pop`,

the original `set-option` commands and per-query resource limits:

```
/path/to/checker --interactive --diagnostics --stats --timeout-ms 10000 < queries.smt2
```

Each `unsat` answer is followed by `; proof-check ...` and `; proof-stats ...` lines on stdout. With `--stats`, stderr shows `semantics=...`, which tells you the route. With `--diagnostics`, stderr has the `diagnostic` and `obligation` rows. To find the source for a pending rule, map the row's rule name back through `diagnostic_rule` (ocaml_runtime/Z3NativeMain.ml) to the `Z3UnifiedProof.rule` constructor, then look for the route's handling: `check_node_cached` for `sorted-*`, `check_node_with_membership` for `legacy`. The first-node index in the row is the entry index in the marshaled document.

The development printer is used the same way with `--readable` added. It works only with a checker built from `ddaca4d0a`; it is a development candidate and not an option of master.

CI runs ../.github/workflows/pulse-checker-build.yml on pushes and pull requests that touch `fstar/**`, and ../.github/workflows/pulse-checker-regressions.yml on completed builds (README.md). By convention, every `.fst` in `tools/checker_modules.json` is verified before extraction, a semantic change comes with F* tests and native regressions, and a change of scope or guarantee is described in a report in this directory.

9 Limitations

This section collects the limits stated elsewhere.

- The legacy global contract is uninhabited, so the legacy kernel theorem does not provide the intended soundness assurance, and the legacy route has no document theorem (5.1).
- The inhabited theorems on master are node-level and cover the typed core, typed instantiation and a restricted binder fragment (5.2).
- The sort-indexed document theorem depends on quantifier hypotheses that master does not construct (5.3).
- The typed root of #11498 is a draft, off the default route, relative to the root model class, and trusts the native translation (5.4).
- Extraction, runtimes, adapters and drivers are trusted. There is no proved whole-checker resource or stack bound (6).
- Empirical `ACCEPT` counts, test pass counts and parity results are observations. They do not establish formal global assurance.
- Pending obligations are undecided steps, not false steps. The checker records no reason for a pending obligation.
- The readable printer is unmerged development work with an open fidelity defect (a fix is in progress) and parity that has not passed (7).
- Appendix A measures `deef01f5b` binaries on one corpus with per-file time limits. Its counts do not predict pending counts on master, and it did not execute the unprocessed tails of 42 files.

A Pending rules in the F* build corpus

A.1 Data and method

The corpus is `C:/bench/inputs/fstar-build` (local path, not in the repository): 583 distinct `.smt2` files (286 at the top level, 297 under `pulse/`), 985,300,241 bytes, 14,030 top-level `check-sat` commands, no `check-sat-assuming`, 28,020 `push` and 27,431 `pop` commands. Every file contains `push`.

Counts come from the completed replay labelled `frozen-deef-w1` (created 2026-10-31T15:09:28Z). Its OCaml and Cherry checkers were built at master `deef01f5b899f47fdbeccb0f408b57324cbb1ffe`, before #11493 and the later Z3 changes; they already contain #11450 and #11478. Each file was replayed as exact bytes on stdin with `--interactive --diagnostics --stats --timeout-ms 10000`, keeping `push`/`pop` and the in-script options (including per-query `:rlimit`), under a 60 s wall limit and a 4 GiB address-space limit per file and backend, with at most two checker processes at a time.

`git merge-base --is-ancestor` shows that #11450 (`710b259c4`) and #11478 (`a398faead`) are ancestors of `deef01f5b` and that #11493 (`8e95e9c47`) is not. `Z3SortedProof.fst`, the marshaler `Z3NativeProof.fst` and both drivers are unchanged between `deef01f5b` and `8fbb275ab` (`git diff --name-only`), but modules they depend on changed. `Z3UnifiedProof.fst`, which `Z3SortedProof` imports, gained the #11493 binder guard, and `Z3CoreKernel.fst` and `Z3QuantifierKernel.fst` changed. The OCaml runtime gained the trusted shim `fStar_String.ml`, and 14 files of the solver under `src/` changed. The counts below are therefore observations of the `deef01f5b` binaries and do not predict pending counts on master.

The tables were recomputed from the raw `queries.jsonl`, `files.jsonl` and `file-accounting.tsv` of that run, and they agree with the run's own report. The readable examples in A.5 come from a different build, the `--readable` development checker (`ddaca4d0a` on `986c626fc`), run on the same corpus files. They show what a pending condition looks like; they are not part of the counts.

A.2 Outcomes per backend

Outcome	OCaml	Cherry
files attempted	583	583
files complete (every `check-sat` answered)	544	544
files stopped by the 60 s limit	39	33
files crashed (SIGSEGV)	0	6
`check-sat` commands in the corpus	14,030	14,030
answered	11,712	11,483
`ACCEPT`	18	18
`PARTIAL` (at least one pending obligation)	11,249	11,060
`UNKNOWN` (solver answer)	445	405
`SAT`	0	0
`REJECT`	0	0
unprocessed tail	2,318	2,547
in flight at the kill (phase undetermined)	39	33
in flight at the crash (phase undetermined)	0	6
never run after a kill	2,279	1,796
never run after a crash	0	712

Answered = ACCEPT + PARTIAL + UNKNOWN; tail = in flight + never run. Every `ACCEPT` and `PARTIAL` query reported `semantics=sorted-datatype`. The `UNKNOWN` reasons are `canceled` (OCaml 360, Cherry 321) and `(incomplete quantifiers)` (85, 84); the data does not separate the 10 s solver timeout from in-script `:rlimit` exhaustion. A query in flight when its file was killed or crashed printed nothing, so it cannot be attributed to solving, marshaling or checking.

A.3 Units

Each `PARTIAL` row of the replay lists obligation groups `{rule, count, first_index, fingerprint}`; the group counts of a row sum to its pending count in all 22,345 checked rows.

- **Obligations** of a family: the sum of `count` over all groups with that rule.
- **Queries**: distinct (file, `check-sat` ordinal) pairs with at least one obligation of that family.
- **Files**: distinct files with at least one obligation of that family.

Families overlap. One query usually has several families, so the query column sums to 36,285 (OCaml) and 35,664 (Cherry) while only 11,249 and 11,060 distinct queries are partial, and the file column sums to 3,177 and 3,172 against 570 and 569 distinct files with any pending obligation. Do not add the columns. Total pending obligations: 109,759 (OCaml) and 105,824 (Cherry).

Because every checked query took the sort-indexed route, a family count is the number of nodes with that rule that the sort-indexed procedure did not decide (section 3.5). It is not a count of legacy recognizer failures, and the replay recorded no reason: every diagnostic row has `stage`, `index` and `extra` set to null.

A.4 The 13 families

Rule	OCaml obligations	OCaml queries	OCaml files	Cherry obligations	Cherry queries	Cherry files
th-lemma	37,037	3,548	279	35,792	3,475	279
def-axiom	17,956	1,828	307	16,690	1,785	302
arith/farkas	17,505	3,642	276	16,811	3,573	277
sk	10,820	10,820	569	10,641	10,641	568
quant-intro	9,118	6,524	530	8,950	6,419	529
nnf-pos	5,899	2,215	312	5,824	2,184	310
nnf-neg	5,643	5,643	536	5,573	5,573	535
rewrite	5,042	1,775	292	4,866	1,724	291
assumed:nlsat	620	185	40	557	184	41
elim-unused	71	71	17	72	72	18
arith/lp-monomial-interval	29	21	10	30	22	13
arith/lp-monomial-substitute	10	10	8	9	9	8
intro-def	9	3	1	9	3	1

The two backends ran separate producer and checker processes. Of the 14,030 queries, 3,113 differ between them in outcome, pending count or rule multiset; the run did not investigate why.

The next table gives, per family, the smallest corpus file containing it in the count run (then the lowest `check-sat` ordinal and node), the source that handles the rule, and a direction for investigation. `q` is the 1-based `check-sat` ordinal in the file and `node` the document entry index. Where the Cherry location differs, it is given second. The directions are proposals for where to look, not diagnosed causes.

Rule	Count-run example (`q`, `node`)	Source	Investigation direction
th-lemma	`pulse/queries-Pulse.C.Typenat.smt2` q1 node 1	`checked_theory_lemma` (Z3UnifiedProof.fst) on legacy; consequence on sort-indexed	Group the readable conditions by theory and check which legacy recognizer, if any, decides them when replayed alone.
def-axiom	`queries-FStar.IntegerIntervals.smt2` q4 node 18	`D.tautology` on legacy; consequence on sort-indexed, which does no definition expansion	Check whether the pending clauses are propositional tautologies over their atoms, which a premise-free consequence check could decide.
arith/farkas	`pulse/queries-Pulse.C.Typenat.smt2` q1 node 22	`LA.check` (Z3LinearArithmeticAst.fst); weights unused on sort-indexed	Measure how many become `ACCEPT` with a typed Farkas bridge on the sort-indexed route; open #11495 proposes one (per its description, not reproduced here).
sk	`pulse/queries-Pulse.C.Types.Array.Base.smt2` q1 node 11	`Z3Skolemization.check` (Z3Skolemization.fst); Skolem definitions unused on sort-indexed	Re-measure on a build with #11495's Hilbert-epsilon bridge (per its description, not reproduced here) before adding anything new.
quant-intro	`queries-FStar.PropositionalExtensionality.smt2` q1 node 5	`binder_step` (Z3UnifiedProof.fst); certificate unread on sort-indexed	Confirm that the pending nodes are binder introductions whose premise sits only in the certificate (section 3.5).
nnf-pos	`queries-FStar.Classical.Sugar.smt2` q1 node 89	as `quant-intro`	as `quant-intro`
nnf-neg	`queries-FStar.PropositionalExtensionality.smt2` q1 node 13	as `quant-intro`	as `quant-intro`
rewrite	`queries-FStar.Math.Euclid-1.smt2` q1 node 312	recognizer disjunction (Z3UnifiedProof.fst); consequence on sort-indexed	Start from the small example in A.5 (commuting a product with a boxed `1`) and test the integer arithmetic of the sort-indexed route on it.
assumed:nlsat	`queries-FStar.Math.Euclid.smt2` q2 node 14	none: the producer marks the step `arith assumed nlsat` (ocaml_runtime/Z3Native.ml)	Z3 supplies no certificate for these steps; checking them needs producer-side evidence or a nonlinear checker.

Rule	Count-run example (q, node)	Source	Investigation direction
elim-unused	pulse/queries-Pulse.Lib.Raise.smt2 q3 node 3	ElimUnused in relational_step; consequence on sort-indexed	The readable example is (forall x. true) = true; a direct test on the sort-indexed quantifier_equality would show whether this shape is decided.
arith/lp-monomial-interval	pulse/queries-Pulse.Lib.Mutex.smt2 q9 node 85; Cherry queries-FStar.Math.Exp.smt2 q9 node 51	name built at runtime from the LP hint (ocaml_runtime/Z3Native.ml); no legacy checker	Read the producer's monomial-interval justification and decide what certificate it would need.
arith/lp-monomial-substitute	queries-FStar.UInt-1.smt2 q9 node 257; Cherry queries-FStar.Math.Euclid-2.smt2 q4 node 313	as arith/lp-monomial-interval	as arith/lp-monomial-interval
intro-def	pulse/queries-PulseCore.Semantics.smt2 q20 node 619; Cherry node 518	collect_definitions (Z3Proof Document.fst) on legacy, where a failed intro-def rejects the document; consequence on sort-indexed	Only one file, three queries; inspect the full raw condition before generalizing.

A.5 Readable examples

These blocks come from the development --readable printer (ddaca4d0a on 986c626fc, OCaml candidate checker, flags --interactive --diagnostics --readable --stats --timeout-ms 10000), not from the count run. NAME#ID is a declaration with its native Z3 id, #k:S a bound variable with its sort, and @x1!3 a Z3 Skolem constant. Lines longer than 400 characters are cut, and each cut says how much was kept; the full text is in the local raw stderr files listed in Appendix B. The readable header query=N counts checked proofs, not check-sat commands, so the check-sat ordinal is given separately when they differ.

sk, pulse/queries-Pulse.C.Types.Array.Base.smt2, check-sat 1. The pending step relates a negated universal over sort Term to its Skolemized form:

```
readable  query=1  verdict=PARTIAL 1  pending=1  families=1  shown=1
readable  family=1/1  rule=sk  count=1  members=1  node=11  node-pending=1
  node 11: premises: none
  conclusion: observational((not (forall [Term#2147483829] (or (not (HasTypeFuel#2147483843:Bool MaxIFuel
   ↪ #2147483839:Fuel#2147483833 #0:Term#2147483829 (FStar.Ghost.erased#2147484003:Term#2147483829 U_zero
   ↪ #2147483801:Universe#2147483800 (FStar.SizeT.t#2147484004:Term#2147483829 Dummy_value#2147483832:
   ↪ Dummy_sort#2147483831)))) label_1#2147484074:Bool (Valid#2147483842:Bool (Prims.hasEq#2147484006:T [
   ↪ excerpt: 400 of 1052 chars]
  source formula: holds((~ (not (forall [Term#2147483829] ... [excerpt: 1051 chars in raw]
```

elim-unused, pulse/queries-Pulse.Lib.Raise.smt2, check-sat 3, complete:

```
readable  query=3  verdict=PARTIAL 2  pending=2  families=2  shown=2
readable  family=1/2  rule=elim-unused  count=1  members=1  node=3  node-pending=1
  node 3: premises: none
  conclusion: equal((forall [Term#2147483829] true), true)
  source formula: holds((= (forall [Term#2147483829] true) true))
```

rewrite, queries-FStar.Math.Euclid-1.smt2, check-sat 1. Both sides are the same product of two boxed integers, one of them 1, in opposite order:

```
readable  family=7/13  rule=rewrite  count=1  members=1  node=312  node-pending=1
  node 312: premises: none
  conclusion: equal((* (BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 (BoxInt_proj_0
   ↪ #2147483934:Int @x0!5#2147484241:Term#2147483829))) (BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:
   ↪ Term#2147483829 1))), (* (BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 1)) (
   ↪ BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 (BoxInt_proj_0#2147483934:Int @x0!5#21 [
   ↪ excerpt: 400 of 429 chars]
```

arith/farkas, pulse/queries-Pulse.C.Typenat.smt2, check-sat 1. Four linear premises over boxed integers and the conclusion false:

```
readable  family=1/15  rule=arith/farkas  count=3  members=3  node=4  node-pending=1
  node 4: premises: #0 #1 #2 #3
  premise #0: holds((>= (BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 0)) 0))
  premise #1: holds((<= (+ (BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 0)) (* -1 (
```

```
  ↪ BoxInt_proj_0#2147483934:Int @x1!3#2147484212:Term#2147483829))) 0))
  premise #2: holds((<= (+ (BoxInt_proj_0#2147483934:Int @x1!3#2147484212:Term#2147483829) (* -1 (
   ↪ BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 1))) (* -1 (BoxInt_proj_0#2147483934:Int
   ↪  (BoxInt#2147483929:Term#2147483829 (+ (BoxInt_proj_0#2147483934:Int @x1!3#2147484212:Term#2147483829)
   ↪ (* -1 (BoxInt_proj_0#2147483934:Int (BoxInt#2147483929:Term#2147483829 1)))))))) 0))
  premise #3: holds((not (>= ... [excerpt: 470 chars in raw]
  conclusion: holds(false)
  source formula: same as conclusion
```

`arith/lp-monomial-interval`, `queries-FStar.Math.Euclid-2.smt2`, `check-sat` 4 (readable `query=3`), complete. In the count run the same OCaml query had no obligation of this family; the example shows the development build's own proof:

```
readable  query=3  verdict=PARTIAL 36  pending=36  families=34  shown=20
readable  family=14/34  rule=arith/lp-monomial-interval  count=1  members=1  node=334  node-pending=1
  node 334: premises: #287 #294 #299 #289 #333
  premise #287: holds(true)
  premise #294: holds((>= (* 1 (* (BoxInt_proj_0#2147483934:Int @x0!76958#2147484428:Term#2147483829) (
   ↪ BoxInt_proj_0#2147483934:Int (@x2!3#2147484238:Term#2147483829 (BoxInt#2147483929:Term#2147483829 1) @x0
   ↪ !76958#2147484428:Term#2147483829)))) 1))
  premise #299: holds((<= (* 1 (* (BoxInt_proj_0#2147483934:Int @x0!76958#2147484428:Term#2147483829) (
   ↪ BoxInt_proj_0#2147483934:Int (@x2!3#2147484238:Term#2147483829 (BoxInt#2147483929:Term#2147483829 1) @x0
   ↪ !76958#2147484428:Term#2147483829)))) 1))
  premise #289: holds((<= (* 1 (BoxInt_proj_0#2147483934:Int @x0!76958#2147484428:Term#2147483829)) 0))
  premise #333: holds((>= (* 1 (BoxInt_proj_0#2147483934:Int (@x2!3#2147484238:Term#2147483829 (BoxInt
   ↪ #2147483929:Term#2147483829 1) @x0!76958#2147484428:Term#2147483829))) 1))
  conclusion: holds((> (* 1 (BoxInt_proj_0#2147483934:Int @x0!76958#2147484428:Term#2147483829)) 0))
  source formula: same as conclusion
```

In these blocks the tab separators of the raw output are shown as two spaces.

Index of readable examples for all 13 families. Both backends printed a block for every family. Node ids belong to the development build's proofs.

Rule	**File**	**`check-sat` (readable `query=`)**	**Node OCaml / Cherry**
`th-lemma`	`pulse/queries-Pulse.C.Typenat.smt2`	1 (1)	6 / 6
`arith/farkas`	`pulse/queries-Pulse.C.Typenat.smt2`	1 (1)	4 / 4
`def-axiom`	`queries-FStar.IntegerIntervals.smt2`	4 (4)	18 / 18
`sk`	`pulse/queries-Pulse.C.Types.Array.Base.smt2`	1 (1)	11 / 11
`quant-intro`	`queries-FStar.PropositionalExtensionality.smt2`	1 (1)	5 / 5
`nnf-pos`	`queries-FStar.Classical.Sugar.smt2`	1 (1)	89 / 89
`nnf-neg`	`queries-FStar.PropositionalExtensionality.smt2`	1 (1)	13 / 13
`rewrite`	`queries-FStar.Math.Euclid-1.smt2`	1 (1)	312 / 312
`assumed:nlsat`	`queries-FStar.Math.Euclid.smt2`	2 (2)	257 / 257
`elim-unused`	`pulse/queries-Pulse.Lib.Raise.smt2`	3 (3)	3 / 3
`arith/lp-monomial-interval`	OCaml `queries-FStar.Math.Euclid-2.smt2`; Cherry `pulse/queries-Pulse.Lib.Task-2.smt2`	OCaml 4 (3); Cherry 13 (13)	334 / 85
`arith/lp-monomial-substitute`	OCaml `pulse/queries-Pulse.Lib.FlippableInv.smt2`; Cherry `queries-FStar.Math.Euclid-2.smt2`	OCaml 2 (2); Cherry 4 (3)	83 / 337
`intro-def`	`pulse/queries-PulseCore.Semantics.smt2`	20 (19)	440 / 435

Two rows show the development build's proof rather than a counted obligation: the OCaml `arith/lp-monomial-interval` query (`Euclid-2` `check-sat` 4) and the OCaml `arith/lp-monomial-substitute` query (`Pulse.Lib.FlippableInv` `check-sat` 2) had no obligation of that family in the count run. For both monomial families the smallest OCaml count-run files (`Pulse.Lib.Mutex` and `FStar.UInt-1`) printed no block of the family under the development build, so the OCaml examples come from a second set of runs. The smallest Cherry file for the interval family (`FStar.Math.Exp`) also printed none; the smallest Cherry file for the substitute family (`FStar.Math.Euclid-2`) did print one, and it is the Cherry example shown.

Short descriptions of the remaining OCaml examples, paraphrased from the start of each conclusion (schematic; the character count is the length of the raw conclusion line):

- `th-lemma` node 6: a disjunction whose first literal equates `BoxInt_proj_0` of a Skolem constant with a sum of `BoxInt_proj_0` terms (1,355 characters).
- `def-axiom` node 18: a clause `(or (not (= (Prims.op_GreaterThanOrEqual ...  10 5) (BoxBool`

`true))) (= (if ...) ...))` over boxed integers (692 characters).

- `quant-intro` node 5: an equality whose left side is a universal over `Term Term` guarded by `HasTypeFuel` (3,439 characters).
- `nnf-pos` node 89: an observational equality whose left side is a universal over `Fuel Term Universe Term Term` (1,999 characters).
- `nnf-neg` node 13: an observational equality whose left side is a negated disjunction containing a universal (3,572 characters).
- `assumed:nlsat` node 257: an integer clause containing the product of two Skolem constants (734 characters).
- `intro-def` node 440: a conjunction whose first conjunct is a universal over `Term` containing an existential (1,370 characters).

A.6 Timeouts, crashes and unexecuted tails

Files that did not complete are reported apart from pending obligations. 42 distinct files are incomplete on at least one backend: 36 on both, 3 only on OCaml and 3 only on Cherry (297,993,707 bytes). Over the files incomplete on each backend (39 on OCaml, all timeouts; 39 on Cherry, 33 timeouts and 6 crashes), OCaml answered 2,251 of 4,569 `check-sat` commands and Cherry 2,028 of 4,575. Over the 42-file union the figures are 2,363 of 4,681 for OCaml and 2,134 of 4,681 for Cherry. The smallest is `queries-FStar.UInt128-1.smt2` (50,317 bytes, 2 queries, none answered on either backend). The largest is `queries-FStar.Math.Fermat-3.smt2` (63,883,565 bytes; OCaml answered 12 of 56, Cherry 0 of 56).

Six files crashed with SIGSEGV on Cherry and were stopped by the time limit on OCaml:

File	Answered / expected (Cherry)	Wall time (s)
`pulse/queries-Pulse.Lib.PriorityQueue.smt2`	37 / 279	26.2
`pulse/queries-Pulse.Lib.RWLock-2.smt2`	22 / 156	20.8
`pulse/queries-PulseCore.IndirectionTheorySep-4.smt2`	35 / 181	27.1
`queries-FStar.Endianness.smt2`	39 / 75	40.1
`queries-FStar.Int-3.smt2`	84 / 127	59.1
`queries-FStar.UInt-5.smt2`	40 / 157	21.9

These binaries already contained #11478, so the crashes happened with its stack fixes present and cannot be filed as the overflows it fixed. No core dumps or stack traces were collected, the crashing component is unknown, and the files have not been rerun on master. The Cherry driver has no handler for stack exhaustion (section 6), which is consistent with a stack overflow but does not establish one.

The unprocessed tails (2,318 OCaml and 2,547 Cherry `check-sat` commands) were never executed. Their outcomes and pending obligations are unknown. No second wave with a longer limit was run, no `--readable` sweep of the whole corpus was run, and neither the `986c626fc` nor the `8fbb275ab` checker was run over the whole corpus.

A.7 What the appendix shows and does not show

It shows which rule names the `deef01f5b` sort-indexed route left undecided on the answered part of a real F* build, how often, and in how many queries and files. It does not show that any pending step is false, why any step is pending, what master or a future build would report, or anything about the unexecuted tails. The 18 `ACCEPT` queries per backend are 0.13% of the 14,030 commands; they are empirical results and carry no formal assurance beyond section 5.

B Evidence files

These are local paths on the machine that produced the report. They are not part of the repository and are listed so that a reviewer with access can check the numbers.

- Run directory `C:/z3atomic/.atomic/readable-pending-20261031/` with `REPORT.md` (SHA-256 `9c3477ef2a6fd12c00004ad37ac14a7f0623e6102f987976ed21e2210e180b1d`).
- Count run `runs/frozen-deef-w1/`: `run.json` (`12637bf0...`), `summary.json` (`dd49fda6...`), `summary.md` (`59ab6af7...`), `file-accounting.tsv` (`5459aea5...`), `queries.jsonl` (`9b4b5517...`), `files.jsonl` (`2b26382a...`), `failures.tsv` (`23887d54...`). Binaries: OCaml SHA-256 `9dcdf84056169f48dd75bf0eb710a3676ef5757127b7a22a820b86b09b79e2fd`, Cherry `f9bf4eaa085addfcce45931277825a670673d5744c84a2c704843eadddf8b581`.
- Inventory `runs/inv-w1/inventory.json` (`86d257aebc92d7e579ddbaac80149214f1963fa60bbad926d263cf0d796a58b4`).
- Readable examples `runs/readable-986-families/` (`examples.md bccb778d...`, `examples-index.json 7a38a6ce...`) and `runs/readable-986-families-2/`, with raw stderr per file and backend under `raw/ocaml/` and `raw/cherry/`, for example `runs/readable-986-families/raw/ocaml/pulse__queries-Pulse.C.Types.Array.Base.smt2/stderr.txt`. Development checkers: OCaml SHA-256 `519f88469cb18d6ba331bd772e93ee98767ee3e3bedc63b3724f5541212a42f2`, Cherry `4dd1eaa640dfded92a0ee62cdf42a0b97d15fe8a5c2727c1135cda3446bdcdff`.
- Printer parity `runs/parity-986-file-clean/file-parity-merged.json` and same-binary repetitions `runs/parity-986-file-clean/repeat/`.
- Proof-identity capture `runs/geo-proof-identity/` (`README.md bb0e731c...`, `summary.json 8cf3b7cb...`): file-mode `geo2` and `geo3` runs under `gdb`, 126 runs, window 2026-10-31T18:43:02Z to 19:00:13Z.
- Constant-array defect: failing test output `tests2/logs/cli_red_const_ocaml.out` (`32a39914...`) and `tests2/logs/cli_red_const_cherry.out` (`ab9e9efd...`) under the run directory; the logs belong to development work that is still in progress.
- Independent recomputation of the tables: `C:/z3atomic/.atomic/fstar-contract-repair/technical-report-20261031/recompute_appendix.py` and its output `appendix-data.md` in the same directory.