RESEARCH MANUSCRIPT

Strata

Lossless Compression of Bootable ISO Collections and Nested File Streams

Anonymous manuscript prepared for *Higher Categorical Semantics*

7 November 2026

Abstract

A utility collection consists of ordered file occurrences, many of which already contain compressed or recursively packaged data. Bootable ISO images are the primary case. Their exact bytes determine sector addresses, optical boot metadata and, for hybrid images, disk-layout structures. We specify STC1, an indexed archive that groups inputs by file category, shares exactly equal blocks, and selects a verified encoder by serialized cost from a configurable portfolio. Entries are restored in original order. An indexed range reader recovers the sector intervals of one image by decoding only intersecting blocks. A Lean 4 implementation includes a kernel-checked generic framing theorem and a Linux command-line compressor and decompressor.

We model nested streams as finite typed trees with ordered ports. One-layer contexts form arrows, their chains form simplices, and evaluations along different bracketing trees are related by a contractible space of refinements. Groupoids of storage presentations distinguish coherent relabeling from changes in logical data. A sheaf of byte assignments on a dependency graph describes gluing and supports a local read theorem. We prove a minimum-size result for a fixed block layout and codec portfolio, a reduction showing the combinatorics of shared candidate selection, and an observational criterion for boot through a virtual media adapter. The experiments separate corpus size, storage size, range access and restoration equality, and include already-compressed streams and recursively archived inputs.

Keywords. Bootable ISO, exact recompression, nested streams, Segal spaces, Lean.

CONTENTS / I

Storage and semantics

01 The archive contract 5

02 Bootable images as byte streams 10

03 Information bounds and cost models 15

04 The recursive type of streams 20

05 Coherence of nested compression 25

06 Descent over an archive index 30

07 Extents, dependencies and range locality 35

08 Exact recovery from internal streams 40

09 The STC1 wire format 45

Storage problem. Fit the largest useful collection of rescue, installation, diagnostic and recovery media into a fixed capacity. Recover any chosen image byte for byte and locate the blocks that contain any requested sector interval.

Semantic problem. Specify the distinction between an occurrence of a file, its byte value, the layer that contains it, and a storage representative for those bytes. State composition before optimization so a candidate that changes a required occurrence is excluded independently of its size.

Scope. Inputs may belong to different categories. Within a category the order of occurrences is stable. Explicit nested members preserve their on-disk bytes, even when an independent reader can recursively expose their catalogs. ISO classification in the executable uses a filename suffix, while ISO standards govern what a media reader can interpret.

CONTENTS / II

Implementation, evidence and deployment

10 Choosing a compression plan 50

11 Lean framing and the executable 55

12 Range equality and boot equivalence 60

13 A hierarchical index and search layer 65

14 Failure semantics and resource limits 70

15 Evaluation on bootable ISO collections 75

16 Nested archives and document experiments 80

17 Capacity, migration and deployment 85

18 Relationship to prior work and synthesis 90

A Reproduction and command reference 95

B A complete small wire trace 96

C Reader conformance checklist 97

R References 98

Reading by goal. Chapters 1–3 give the preservation contract and the compression lower bounds. Chapters 4–8 develop the category of contexts, indexed reconstruction and reversible visibility into compressed members. Chapters 9–11 are the implementer's specification. Chapters 12–17 connect the format to boot adapters, virtual filesystems, measurements and storage policy. Chapter 18 locates the contribution in compression, storage and higher category theory.

PRELIMINARIES

Notation and evidence

Bytes and positions. Let $\mathbb{B} = \{0, \ldots, 255\}$ and $\mathbb{B}^*$ be the finite byte strings, with juxtaposition for concatenation. Byte offsets and occurrence ordinals are zero-based. The interval $x[a, b)$ contains offsets $a \leq i < b$. All lengths in the format are bytes unless explicitly labeled sectors, bits, KiB or MiB. One MiB is 2^{20} bytes. $\mathrm{Fin}(n)$ is the ordered set of n slots.

Kinds of equality. Logical equality preserves occurrence order, declared labels and contents. Byte equality compares each original representation, including internal compressed members. Equivalence of presentations preserves a logical value and the dependency information specified in its context. Presentation isomorphisms are typed and cannot identify two original occurrences merely because their bytes agree.

Evidence. Mathematical statements are proved under their displayed hypotheses. Lean checks the statements named in Chapter 11. The executable and its shell-command boundary are exercised separately. Compression numbers describe named corpora, block sizes and candidate policies. The semantics of a media adapter is stated as a conditional result. Boot through a firmware implementation requires a separate integration test.

Reproducibility. The companion contains Lean source, commands, corpus provenance and measured output. External utilities are selected on the local system. The manuscript references public specifications and articles at their canonical or publisher URLs. It distinguishes the implemented STC1 v1 reader from proposed optional graph encodings and firmware adapters.

Authorship and automation. The public-facing byline is anonymous for editorial handling. An AI drafting system assisted with exposition, implementation and experiment orchestration. The source, checks and explicit claim boundaries are included for author and editorial review.

1 The archive contract

An emergency tool collection can include several editions of the same installer, separate BIOS and UEFI utility images, rescue environments and reference PDFs. Distributing every image as an opaque independent object preserves interoperability and leaves cross-image repetition unexploited. Placing the images in a single solid compressed stream can exploit nearby repetition at the cost of reading the solid stream when one sector is requested. Strata assigns each input to a category, encodes a sequence of independently recoverable blocks, and retains the original order in an explicit catalog.

Let C be a set of category identifiers and N a set of names. The value of a collection is

$$F = \Big((c_i, n_i, x_i)\Big)_{i=0}^{m-1} \ \in \ \mathrm{List}(C \times N \times \mathbb{B}^*).$$

The ordinal i identifies an occurrence. Equal names, categories or byte strings can occur at several ordinals. An archive implementation supplies an encoding E and partial decoder D satisfying $D(E(F)) = F$ on supported inputs. The supported CLI input is a sequence of finite regular files under its index, label and block limits. The mathematical framing handles every finite sequence of finite byte strings.

Three objectives have different units. The packed representation occupies bytes. A range read costs physical bytes, decoded bytes and invocations. The usable utility collection counts original occurrences or assigns their operational values. A Pareto comparison must state which dimensions are fixed. We optimize serialized size over a declared finite portfolio under a fixed range-read design. Disk utility selection is a separate capacity-constrained problem. These choices permit meaningful comparisons with both separate compressed images and a solid archive.

CHAPTER 1

Occurrences, names and order

Suppose the arguments are (`tools.iso`, `manual.pdf`, `tools.iso`), with the first and third arguments from different directories. The catalog has three rows. During packing the PDF can be read before or after both ISO entries, according to the physical category order. Its ordinal stays 1. If the ISO bytes agree, the two ISO rows may share every data block. A consumer still sees two entries, and can recover either through its ordinal. Thus deduplication acts on resources and indexing acts on occurrences.

Let s be the stable ordering of ordinals by category and then by ordinal. A packer traverses $s(0), \dots, s(m-1)$. A separately recorded map $I(i)$ gives each occurrence its span list. For a decoder B of those spans, the outer result is

$$\big((c_i, n_i, B(I(i)))\big)_{i \in \mathrm{Fin}(m)}.$$

The original sequence is recovered even though the physical layout follows s. There is no dependence on labels being unique. For example, two files called `disk.iso` can be materialized under separate numbered directories with a manifest linking both paths to their occurrence records.

The command-line profile records the basename available through the host's path string interface and the original contents. It has no fields for directory parents, symlink targets, permissions, extended attributes or sparse-file allocation. Files that need those properties can be supplied as existing byte-exact containers. Another profile can extend N to a metadata record and revise the restoration map. Such an extension needs an explicit path and overwrite policy because names then influence filesystem operations.

CHAPTER 1

Four observable interfaces

The storage contract can be factored into four calls with different privileges. *Catalog* returns names, categories and lengths with no data-block decoding. *Range* takes an occurrence, offset and count and returns a byte interval. *Extract* is the special case in which the interval covers the full occurrence. *Interpret* passes a reconstructed stream to an application-specific reader, such as ISO 9660 or PDF. Each call has a smaller or larger trust and resource boundary.

Let $R(A, i, a, n)$ be the range interface for archive A. Its specification, for $a + n \leq |x_i|$, is

$$R(E(F), i, a, n) = x_i[a, a + n).$$

Whole restoration follows by setting $a = 0$ and $n = |x_i|$. Conversely a whole decoder can define R by slicing, but this reduction says nothing about random access. The storage graph determines whether range implementation actually avoids unrelated data. Indexed payload blocks are the units of physical access in STC1. The catalog itself remains a finite structure loaded at open time.

Interpret introduces a second category of observations. ISO path lookups, PDF attachment names and a nested Strata catalog interpret the bytes of a member. Those observations may be recorded in a derived, versioned index. The raw-byte range contract remains the basis for validating that index and for reproducing its inputs. As a result, a parser can refuse a malformed file, or a security policy can decline to inspect a member, while the containing archive can still restore it exactly.

CHAPTER 1

What makes a nested collection

A logical collection can contain an entry x whose payload is itself a valid archive. That payload is an ordinary byte value at its parent slot. If a reader opens it, it discovers child slots, and each child slot again has its own byte value and category. A path such as $(7, 2, 4)$ selects an occurrence at each successive layer. A PDF can similarly expose an embedded file, and an ISO can expose a directory entry containing a compressed filesystem image. The path must record which kind of opening operation occurs at each edge.

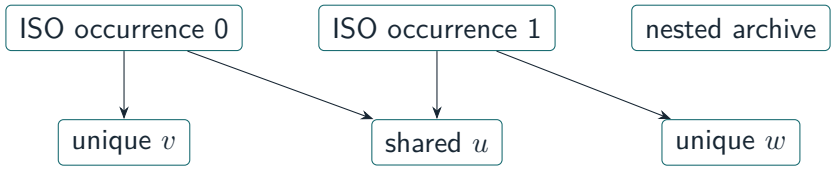
An *opaque view* stops at a byte value. A *structural view* adds an opening witness that names a parser and records its ordered outputs. Two catalog policies can offer different structural views over the same archive. A recursive repacker requires a stronger witness that explains how to reconstruct the original outer bytes from its exposed child representations. Chapters 4 and 8 formalize that distinction.

Consider an ISO containing `filesystem.squashfs`. Decoding the SquashFS files and constructing another SquashFS volume can preserve file contents while changing superblocks, compressor output, padding and the position of files in the outer image. For byte recovery the parent occurrence retains either the original stored data or a reconstructive witness accounting for every change. A filesystem-facing view can still display the child files independently of that choice. Nested visibility is an indexing operation. Recursive recompression is a separate storage operation with an exactness proof obligation.

CHAPTER 1

Running example and validation obligations

Use a rescue collection with three input images a, b, c, a manual p, and a prior archive q. Suppose a and b have a common aligned image prefix u, with $a = uv$, $b = uw$, and $c = z$. With extents aligned to the end of u, a physical dictionary stores u once and the catalog describes the two uses. The bytes of q can be decoded as a file and passed to the same reader. The optional recursive profile can expose its children to the catalog. These are independently testable mechanisms.



A falsifier for byte recovery changes one byte, omits a slot, or swaps two distinct ordinal outputs. A falsifier for range locality requires reading an unrelated block in a single-range operation on a stable archive, after catalog loading. A falsifier for the portfolio claim exhibits an admitted candidate with smaller serialized cost for the same physical block. A failure to boot requires a separate comparison with the source on the same firmware and adapter. Every claim names the interface and the evidence that can test it.

2 Bootable images as byte streams

An ISO image is a file that serves as a block-addressed representation of optical media. ECMA-119 specifies the optical volume and file structure, including volume descriptors and logical block addressing [31]. The disk image as delivered can carry multiple related structures. Some hybrid images include disk partition data and boot code in addition to optical descriptors. An archiver can preserve all of them with the same byte contract, even when it has no interpretation for one of the structures.

For a medium exposing sectors of size s, the logical block read at address ℓ is the interval

$$x[s\ell, s(\ell + 1)).$$

The common optical boot path uses 2,048-byte sectors. An adapter must use the geometry of the interface it advertises. A hybrid image exposed as a hard disk can be interpreted through a different sector geometry and boot path. The image bytes are the same, but the function converting a device request to a byte range can change. Sector size belongs in the adapter configuration, not in an inference from the extension of the archived filename.

A filesystem-level copy of the visible directory tree is a different preservation task. The same directory tree can have distinct volume descriptors, boot catalogs, order of sectors and system-area contents. An ISO importer may index those directory entries and still archive the full original image. The catalog of images, a catalog of ISO directory entries and an inventory of utilities are three views whose identifiers must remain distinct.

CHAPTER 2

El Torito and firmware boot paths

El Torito defines an optical boot record and a catalog pointing to boot images [32]. The catalog records include the validation entry and boot entries, with media type and image location information. A reader that makes the expected sectors available at their original logical addresses preserves the structural data the firmware will consult. Copying just files visible to a mounted filesystem can omit the boot image or alter where it is referenced.

UEFI optical boot uses the El Torito no-emulation format for an EFI system partition, as specified in the media access chapter of UEFI 2.11 [33]. Firmware selects a path according to platform and configuration. BIOS compatibility and UEFI compatibility are therefore two tests. A hybrid ISO used as USB mass storage adds a further distinction. For example, an image can be suitable as a virtual optical device while an unrelated hard-disk launch fails because the firmware interprets its system area differently.

For a source image x, archive extraction is a transformation of a stored representation into the same x. It preserves the byte material needed by each valid path. An implementation claiming that an archive is directly bootable needs to demonstrate the equivalent device behavior of the adapter, including the device class, sizes and driver handoff. A test that compares image hashes exercises byte restoration. Firmware choice, kernel mounting and Secure Boot policy require observations at the device interface. Chapter 12 gives the corresponding logical implication.

CHAPTER 2

Hybrids, sectors and compressed filesystems

A hybrid distribution may combine optical metadata and disk partition metadata so it can be used on more than one kind of medium. Tools such as *xorriso* describe system-area and El Torito information as separate inspection targets [34]. A successful inspector can record several views of one image without rewriting it. Where boot sectors, partition tables, backup headers or padding occupy unusual offsets, indexed byte extents preserve them in the same way as ordinary file data.

A rescue ISO can carry a kernel, an initramfs and an already-compressed root filesystem. SquashFS and EROFS use internal layouts that make their own choices about compression, blocks and read amplification [40, 41]. Outer compression can remove spare padding and exact repeated blocks across versions. For content inside an internal compressed filesystem, the outer codec typically sees a different statistical regime from that of uncompressed directories. A change in the inner compressor, dictionary or metadata can cause broad changes in the stored bytes despite a small logical update.

A useful experimental comparison holds the exact image bytes fixed and separates three effects. Independent packing measures the residual compressibility of each image. Shared aligned block packing measures identical material at reusable boundaries. Structural peeling, when a reconstruction witness exists, measures the possible gain from decoding known internal streams and accounting for their original wire bytes. The third method incurs parsing, residual and access costs and can be more expensive than the first two.

CHAPTER 2

Boot media as a two-stage system

A practical installation contains a small bootstrap that firmware already understands and a larger collection optimized for storage. A firmware-visible EFI system partition or a supported multiboot loader can provide the first stage. The first-stage program selects a utility and either materializes the ISO, passes it to an existing compatible boot mechanism, or exposes it through a virtual device. The placement of the bootstrap matters because it must be available before the archive reader is running.

GRUB provides a loopback command that exposes the contents of an image as a GRUB device. Its documentation also explains the need for operating-system support in a loopback boot arrangement [35, 36]. Ventoy demonstrates a user-facing workflow for selecting image files from one drive [37]. These systems motivate an archive-to-image boundary. Their format support is a separate interoperability contract from STC1.

The storage optimization includes the size of the bootstrap, recovery catalog and temporary extraction space. A packer can minimize archive bytes while leaving insufficient workspace to stage the selected ISO on the same device. A virtual read-only block adapter reduces that staging demand by retaining only a bounded decoded cache. It adds code to the boot chain and requires an explicit strategy for handing the backing image to the next stage. This tradeoff is visible in a capacity model.

CHAPTER 2

An interoperability matrix

Operation	Bytes supplied	External component and evidence
Catalog ISO entries	Indexed metadata	Catalog reader can list and filter labels
Extract an ISO	Complete original image	Byte comparison against source
Read a sector	Original requested interval	Indexed range test and geometry mapping
Boot restored ISO	Complete ISO as media	Firmware and operating system in a named mode
Boot from archive	Virtual media protocol	Preboot adapter, protocol parity and handoff tests
Inspect inner files	ISO directory and file extents	Parser, filesystem profile and bounded descent

A test matrix records source checksum, image size, archive checksum, codec identifiers, extracted checksum, firmware version, machine class and outcome. When the test runs under QEMU, a BIOS and UEFI test require explicitly selected firmware and can have different expected success markers. A diagnostic image that displays a menu needs an observation of that menu or a controlled fixture with a serial marker. A hang, reset or display capture without an identified marker is a different result.

The approach also covers ordinary images without boot metadata. Their byte and range equality use the same proof. A classification label such as `iso` guides alignment and display. Boot inventory metadata can be entered from a verified external test and associated with the image digest. Repacking an unchanged image leaves that association meaningful even when it changes the storage codec.

3 Information bounds and cost models

An injective encoder sends distinct inputs to distinct outputs. For a fixed length n, the set of byte strings has cardinality 256^n. The number of byte strings of length less than n is

$$\sum_{j=0}^{n-1} 256^j = \frac{256^n - 1}{255} < 256^n.$$

Some n-byte string therefore has output length at least n under any lossless encoder. The count includes the empty output. The bound uses only injectivity. It also applies to sequences, by considering the subclass of singleton sequences of n-byte inputs.

Proposition 3.1 (Finite compression fraction)**.** If X is uniform on the n-byte strings and E is injective, then for $1 \le k \le n$,

$$\Pr[|E(X)| \le n - k] \le \frac{256^{n-k+1} - 1}{255\,256^n}.$$

Proof. The numerator of the count is the number of strings of length at most $n-k$. Injectivity allows at most one input per such output. Divide by 256^n. □

The packer always has an admissible raw payload for each block. It compares compressed payloads with raw under the complete variable part of the block's serialized cost. Archive-wide framing, catalog, checksums and occurrence multiplicities still occupy bytes. In particular a tiny or empty collection has nonzero representation length. The raw archive is the relevant size baseline for a fixed container and access contract.

CHAPTER 3

Entropy, dependence and shared structure

For a discrete random variable X with distribution p, define the entropy $H(X)$ by

$$H(X) = -\sum_x p(x) \log_2 p(x).$$

For a prefix code with lengths $\ell(x)$, Kraft's inequality and the nonnegativity of relative entropy yield $\mathbb{E}\ell(X) \geq H(X)$ [11, 12]. If $\sum_x 2^{-\ell(x)} = K$, take $q(x) = 2^{-\ell(x)}/K$. Then

$$\mathbb{E}\ell = H(p) + D(p\|q) - \log_2 K \geq H(p).$$

A file codec can rely on outer framing for lengths. Comparisons must count that framing and any external dictionaries required for decode.

The chain rule

$$H(X_1, \ldots, X_m) = \sum_{i=1}^{m} H(X_i \mid X_{<i})$$

identifies what joint storage can exploit. For two images, the difference between independent and joint ideal coding is $I(X_1; X_2)$. A shared kernel, repeated padding or retained compressed filesystem blocks create dependence in the byte strings even when each image contains substantial high-entropy content. For the deterministic example $X_2 = X_1$, the joint entropy is $H(X_1)$, because the second occurrence adds no entropy after the first. Its catalog row still has a representation cost.

Entropy describes a distribution. A concrete archive establishes repeated content by byte comparison, observes exact sizes for candidates, and constrains how much conditional dependence the decoder can exploit without reading unrelated data. An engineering model links a distributional claim to a corpus and a concrete access workload.

CHAPTER 3

An accounting identity for deduplication

Write the ordered logical blocks in a candidate layout as $b_1, \dots, b_M$ and let π map them to representatives in a physical dictionary D. If $\pi(i) = \pi(j)$, exact comparison establishes $b_i = b_j$. The raw byte subtotal of the unique dictionary is $\sum_{d \in D} |d|$. The logical raw subtotal is $\sum_i |b_i|$. The raw saving is

$$\sum_{d \in D} (m_d - 1)|d|, \qquad m_d = |\pi^{-1}(d)|.$$

The formula holds for any exact equivalence classes and any deterministic enumeration of the representatives.

If the chosen physical payload costs are s_d, the archive length is

$$L = L_{\text{constant}} + L_{\text{entries}} + \sum_{d \in D} \Big(s_d + L_{\text{block}}(d) \Big).$$

Reference sharing saves space exactly when its payload and index terms sum to less than the alternative. For two identical 2 MiB blocks, a second reference is usually small compared with another payload. For thousands of distinct five-byte strings, block descriptors can dominate. Bundling small files within a category amortizes these descriptors and gives a compressor opportunities to reuse nearby contexts.

If π maps identical byte strings that arise from different original files, the value-level multiplicity remains m_d. Replacing a sequence by a set can appear to improve compression because it discards a required part of the value. The accounting identity separates the cost of values from the cost of their occurrences.

CHAPTER 3

Access imposes a separate lower bound

A reader that emits n requested bytes performs at least n bytes of logical output work. With independent compressed blocks of uncompressed length at most B, a positive interval of length n intersects at most $\lceil n/B \rceil + 1$ blocks in a fixed grid. The decoder may therefore process far more bytes than it emits for small requests. For a one-sector read at an aligned point in a 2 MiB block, the ratio of full-block decoded bytes to requested bytes is 1024.

Consider two compressed layouts for an image of length N. A solid layout has payload cost S and forces a cold decoder to process a long prefix before an arbitrary late range. A block layout has payload cost $S + \delta$ and decodes each intersecting block independently. When reads are uniformly distributed across sectors, the expected prefix work of the first layout scales with N for a prefix decoder without checkpoints. Work in the second scales with the chosen block size. This contrast explains the optimization constraint even when $\delta > 0$.

For an arbitrary extent list, shared or short blocks can invalidate the regular-grid count. Let K be the set of distinct physical blocks touched by a request. Then decoded bytes are bounded by $B|K|$ for a reader that decodes each once. A one-block cache can perform repeated decodes for nonconsecutive uses. Chapter 7 gives a bound in terms of visited extents, and Chapter 10 separates the fixed-layout optimum from choosing the layout itself.

CHAPTER 3

Finite portfolios and universal compressors

A portfolio is a finite collection of encoders paired with decoders. Some candidates finish, some time out and some fail verification. For a fixed block b, let $\mathcal{A}_b$ be the candidate representations that are legal under the resource policy and satisfy $d(c(b)) = b$ on the checked input. Raw belongs to $\mathcal{A}_b$. Comparing actual serialized costs supplies a concrete minimum over this set. If an encoded candidate's cost exceeds the best one, the comparison can skip its decode without affecting the minimum.

Lempel and Ziv's sequential coding theorem studies a specific form of universality over sources and asymptotics [13]. It is compatible with the finite counting obstruction. Real toolkits use bounded windows, block resets, dictionaries, entropy coders and speed levels. An encoder can trade CPU time for a different payload, and a higher named level can yield a larger representation. A portfolio records the best observed admissible representation without a monotonicity requirement on levels.

An external algorithm that destroys information cannot have a uniform left inverse on all byte strings. A wrapper can supply a residual r so $d(c(b), r) = b$, or let the raw choice win. A black-box test witnesses one input's round trip with a particular decoder run. Future decoding depends on compatibility with the stored representation and the codec identifier. Where an application needs long-term preservation, the identifier must specify enough information for another implementation of the decoder.

4 The recursive type of streams

Let C be a set of logical types and let Σ be a finitary typed signature. An operation o has output $t(o) \in C$, arity $k(o)$, and an ordered sequence of input colors $s_1(o), \ldots, s_{k(o)}(o)$. Nullary operations carry raw data or an opaque-file witness. Other operations can form homogeneous lists, typed records, or wrappers equipped with a recipe. For $X \in \mathbf{Set}^C$ define

$$(P_\Sigma X)_c = \coprod_{\{o : t(o) = c\}} \prod_{j=1}^{k(o)} X_{s_j(o)}.$$

Operations include their literal parameters. For example, an opaque node stores a particular byte string, and a wrapper operation records a particular validated recipe, plus the types of its exposed child streams. A broad archive color can include a sum of admitted categories.

A P_Σ-algebra assigns a set of values to each color and an evaluation map for each operation. Ordered trees are the initial such algebra. This is the typed polynomial tree construction [1, 2]. To see initiality here, for each operation node evaluate its immediate subtrees, then apply the operation in the target algebra. Termination follows from finite height. The resulting map is an algebra morphism, and induction on height forces its values on every tree.

A practical format constrains values further. A recipe describing a delta can fail when a base has the wrong length. Use an algebra of partial evaluations, or enrich input colors to carry the relevant length and interpretation. The semantic results refer to certified recipes on their specified inputs. On untrusted data a parser returns an error rather than creating a witness it has not checked.

CHAPTER 4

Occurrence addresses and equality

A node address is a finite word of positive slot numbers. The empty word addresses the root. For an operation with arity k, its child addresses have first component in $\{1, \dots, k\}$. Address composition is concatenation. To read an address, begin at the root and follow each child index; any index beyond arity is undefined. Logical equality of two ordered trees requires equality of operation labels and of the corresponding subtree at every port. Position j and position $j + 1$ remain distinct when the values at both are identical.

An archive can offer several opening interpretations of the same underlying bytes. An ISO occurrence can have an optical directory view and a hybrid disk view. These are distinct operations o_{optical} and o_{disk}. A path through the combined namespace can use pairs consisting of a view identifier and a child slot. The view adds structure derived from the bytes, with a specified profile and parser version. The underlying byte value remains available at the occurrence before either view is followed.

Lemma 4.1 (Unique evaluation of an address)**.** For a fixed ordered typed tree and a valid address a, there is exactly one addressed subtree. If ab is valid, selecting ab equals selecting a and then b.

Proof. There is one root. Each next index selects exactly one ordered child of the current node. Induction on the length of a proves both assertions. □

The word of indices survives physical deduplication. A reference graph can realize several addressed nodes with the same byte resource. The logical tree describes where the occurrences are; the graph describes what is needed to read them.

CHAPTER 4

Homogeneous layers and typed substitution

For a given color c, a homogeneous collection operation $L_{c,n}$ has n inputs all of color c. Its output can have a distinct collection color $\ell(c)$. A wrapper $W\colon \ell(c) \to d$ makes that collection a file value of color d. If $d = c$, the same family can occur at several depths. If $d \neq c$, further nesting follows whatever substitutions the signature admits. An archive of bootable ISO images uses $L_{\mathrm{ISO,n}}$; a nested general archive can be another entry with its own category.

The formulation distinguishes preserving input order from commutativity. The fact that two categories can be packed in any physical order concerns a storage representative. Within $L_{c,n}$, exchanging two distinct child values changes the term. If a separate application truly wants an unordered multiset, it can use an action of a symmetric group on ports. The displayed specification uses planar ports, because order determines sequence semantics and often contributes to precise reconstruction.

An n-hole context is a finite operation tree with typed placeholders at distinct leaves. Substitution uses the indicated hole order. Grafting contexts induces the operations of a free colored nonsymmetric operad. The identity at c is a single hole of color c. Associativity holds because the node paths of a substituted subtree are formed by concatenation. Units hold because the path of a hole at the root is empty. Multihole substitution models assembling a layer from several separately decoded streams and later replacing their evaluation schedule.

CHAPTER 4

Forgetting structure without losing bytes

Let V_c be the set of byte values for color c, possibly augmented by interpretation metadata. A representation evaluator wire assigns a byte value to every certified term. For a raw node raw(b), set wire $= b$. For a structured wrapper with immediate children t_j, set

$$\mathrm{wire}(W_\rho(t_1, \ldots, t_k)) = \rho\Big(\mathrm{wire}(t_1), \ldots, \mathrm{wire}(t_k)\Big).$$

The recipe ρ is specified on the child's byte values and the stored residual. The evaluator does not identify distinct terms. A raw node and a structured node can both realize the same bytes, and several structural witnesses may expose different information.

Proposition 4.2 (Sound opacity)**.** Replace a subtree t of a certified term by an opaque leaf of the same color with value wire(t). The byte value at the root is unchanged, provided all ancestor recipes operate on child values.

Proof. At the replaced node the values agree by definition. At its parent the ordered child tuple remains unchanged. Repeated application of this argument reaches the root. □

The reverse direction is a discovery problem. Given an opaque leaf, a parser can propose a structured witness. Promotion is sound only after its reconstruction is verified on that leaf. A viewer can use a weaker opening witness because it makes a different claim. It reports what the parser sees inside the original bytes and preserves the opaque leaf as its provenance.

CHAPTER 4

Recursion, sharing and finite evaluation

Each well-founded logical term is a finite tree. Storing all of its nodes separately can duplicate a large byte value at several occurrences. A physical directed acyclic graph can share those values by addressable block identifiers. The logical tree is obtained by unfolding the physical recipes at each referenced slot. A stable order on ports fixes the unfolding even if the graph's vertices are allocated in another order.

For a graph with edges from a resource to the dependencies it reads, assign rank 0 to a resource with no dependencies and rank $1 + \max r$ otherwise. Every finite acyclic graph has such a rank. Recursive evaluation decreases it strictly. Memoization replaces repeated evaluation of the same vertex with reuse of a validated value. The memory contract can discard a cached value and later recompute it. That policy changes work and latency while preserving the semantic map from graph to byte values.

The categorical construction starts with actual compression streams. Each node has an ordered interface and an interpretation rule, so a stack of nested streams determines a stack of substitutions. Alternative plans can change chunk identities, decompression order, cached resources and grouping of a chain of evaluations. The next chapter gives categorical data for tracking those changes while retaining the outer wire bytes and the inner occurrence structure.

5 Coherence of nested compression

Fix a signature and certified typed terms. Define a context category $\mathcal{C}$. Its objects are terms, and an arrow $u \to v$ is a one-hole typed context K with $K[u] = v$. The identity is the bare hole. Given $K\colon u \to v$ and $J\colon v \to w$, the composite is $J[K[-]]$, a context from u to w. Equality here is syntactic equality of the substituted term with its specified labels. Each elementary step can be a layer whose designated child is the hole and whose other children are fixed.

Composition represents packaging several nested streams in order. For example, let u be a filesystem byte image, K an ISO recipe exposing u, J an archive-of-images recipe, and H another container. The chain $u \to K[u] \to J[K[u]] \to H[J[K[u]]]$ has three successive layers. Contracting an edge in the plan for evaluating the chain composes two contexts. It does not remove that layer's data from the file that must be recovered.

The ordinary nerve of $\mathcal{C}$ sends $[n]$ to functors $[n] \to \mathcal{C}$. In degree two a composable pair includes its composite. The middle face gives the composite, and the outer faces retain the component arrows. In degree three the face identities express compatibility of the two orders of composition. This simplicial object records the incidence of chains of actual layers. The next construction allows variation of storage presentations along the chains.

CHAPTER 5

A Segal model with presentation equivalences

Let $\mathcal{P}$ have objects (t, G), where G is a validated finite storage presentation of term t with named occurrence roots. An arrow consists of a context $K : t \to u$ and an injective morphism of resource presentations carrying the distinguished roots to their images under the hole address of K. It preserves ordered resource recipes, byte values and port maps. Inclusion of a subarchive as a tagged region is an example. Identities and composition use context substitution and composition of resource maps.

Use the simplicial groupoid

$$W_n = \operatorname{core}\Big(\operatorname{Fun}([n], \mathcal{P})\Big).$$

Its objects are chains of nested representations. Its morphisms are commuting levelwise isomorphisms. Taking classifying spaces gives Rezk's classifying diagram, a complete Segal model after the usual fibrant replacement [3, 4]. In a concrete Segal comparison, the right side assembles arrows with isomorphisms between adjacent endpoints. Transport each downstream arrow along the supplied endpoint isomorphism. This produces a strict chain and an isomorphism back to the given assembly. Chain isomorphisms are exactly the compatible levelwise isomorphisms, which gives the equivalence

$$W_n \simeq W_1 \times^h_{W_0} \cdots \times^h_{W_0} W_1.$$

A representation may have automorphisms, for example a pair of indistinguishable unreferenced support resources with labels absent from the public interface. Their transposition gives a nonidentity loop in the core. Original occurrence ordinals are fixed by such a loop. A minimal ordered presentation can have trivial automorphisms and still has the composition simplices of its chain category.

CHAPTER 5

The coherence space of evaluations

Fix a chain of n layers $K_1, \ldots, K_n$. A partial bracketing is a laminar set of proper intervals $[i, j] \subsetneq [1, n]$ of length at least two. Laminar means that two intervals are disjoint or one contains the other. An interval says that the corresponding composite is to be evaluated as one unit. Binary bracketings are maximal elements under inclusion. The empty interval set denotes a single unbracketed composite of all layers.

Define Q_n by ordering these sets by reverse inclusion. Removing brackets is a morphism. The nerve has simplices of successive removals and a terminal vertex, the empty set. The unique natural transformation from the identity to the constant functor at that vertex induces a contraction of $|NQ_n|$. This is the elementary contractibility argument underlying the associahedral model of coherent associativity [5]. For $n = 3$ the two binary bracketings each map to the unbracketed expression. For four layers, the partial bracketings include the five full binary bracketings and intermediate contractions.

The interpretation from this refinement category to composed byte functions is constant after evaluation: every plan computes $K_n[\cdots K_1[-]\cdots]$ as a context and applies the same certified recipes. The constancy follows from associativity of substitution and the value-level evaluation law of Chapter 4. A different codec implementation can appear in the plan only through a comparison that verifies the same bytes at the interface being replaced. A path of valid regroupings therefore transports the same reconstruction claim.

CHAPTER 5

Independent subtrees and higher cells

A layer with two designated child streams permits a second kind of coherence. Let independent reconstruction steps r and s act on different ports of a binary constructor. On a pair (x, y),

$$(r \times \mathrm{id})(\mathrm{id} \times s)(x, y) = (r(x), s(y)) = (\mathrm{id} \times s)(r \times \mathrm{id})(x, y)$$

on the common domain. The resulting commuting square expresses that evaluation order does not change the tuple delivered to the parent. The same argument on m independent ports gives an m-cube. Faces fix some steps, and edges perform one additional step. In a physical implementation two ports can share a decoded resource; memoization becomes a choice of computation schedule over the same tuple.

There is a resource qualification. Steps that write the same temporary file, reserve the same scarce cache slot or consume a single-use stream are not independent actions on a product of byte values. Their scheduling category must include the shared resource state. Independence can be recovered by disjoint staging or by replacing a writer with an immutable binding. The coherent byte semantics tells which reorderings preserve output. An operational semantics tells which reorderings are permitted by the actual machine.

The planar operad of Chapter 4 records substitutions with several holes. One can organize its operations by planar trees rather than by chains. Tree-shaped nerves and the corresponding homotopy theory give a standard ambient language for operadic composition [6]. For STC1 the chain and product constructions suffice to state the correctness of the supported nesting and access operations.

CHAPTER 5

Forgetting a layer and factoring its read

A decomposition of a long evaluation into layers determines a factorization of the corresponding context arrow. The set of admissible binary cuts is ordered by which contiguous groups are precomposed. The final result can be obtained from any compatible chain of refinements. When the computational policy assigns a cost to a plan, the refinement diagram need not have constant costs. Contractibility of the evaluation space is a coherence statement about its output, while minimum cost is an optimization problem on its objects.

For a small example let L expose the extent of a member in an archive, and M expose a subrange of that member. The composite read is determined by the inner-to-outer range map. For independent blocks it is possible to decide which blocks are needed before decoding any. For a wrapped member compressed as one stream the parent may need to decode a larger region. Both plans represent the same logical read if the same bytes reach the child interface. Their physical access costs differ.

This difference supplies a natural link to incidence methods. The nerve of an ordinary category is Segal and therefore an example of a decomposition space [7]. Its incidence construction counts or parametrizes compatible factorizations of arrows under appropriate finiteness conditions. Here a cost computation can index over a specified finite family of such factorizations. An unrestricted class of archive contexts is generally infinite. A finite portfolio and a finite depth bound give a concrete summation or optimization domain.

6 Descent over an archive index

A catalog and a resource graph determine which decoded values must be available for a read. Let G be a finite directed acyclic graph whose edges point toward dependencies. Write $D(v)$ for the dependency set of v. An open subset U of its vertices is downward closed under these edges, so $v \in U$ implies $D(v) \subseteq U$. Arbitrary unions and intersections of these subsets are downward closed. This is the finite Alexandrov topology of the reachability order, with the empty open included.

At each vertex v, let X_v be a set of values, usually bounded byte strings with a specified length. A deterministic local recipe f_v takes the ordered dependency tuple to X_v. A vertex can instead admit a relation R_v between its proposed value and its dependency tuple. A source constant is a recipe with no dependency arguments. Define

$$\mathcal{A}(U) = \{(x_v)_{v \in U} \mid (x_v, (x_d)_{d \in D(v)}) \in R_v \text{ for all } v \in U\}.$$

An inclusion $V \subseteq U$ gives a restriction map by projecting onto V. Downward closure ensures the result continues to satisfy all local relations.

The definition attaches partial computation values to dependency-closed regions. A catalog view can select vertices whose addressed byte intervals support a query. Closing under dependencies gives a minimal region on which its computation can be interpreted. A format with direct data blocks yields particularly small such regions, because data blocks have no stored base dependencies.

CHAPTER 6

Gluing compatible byte assignments

Theorem 6.1 (Descent for validated values)**.** The presheaf $\mathcal{A}$ is a sheaf on the dependency topology for covers by union.

Proof. Let $U = \bigcup_i U_i$ and take sections $x^{(i)} \in \mathcal{A}(U_i)$ agreeing on every pairwise intersection. For each $v \in U$, pick any i with $v \in U_i$ and put $x_v = x^{(i)}_v$. The agreement condition makes this independent of i. All dependencies of v belong to that same U_i. Hence the local relation holds for v in U. The resulting section restricts to the given family, and any other such section agrees on every vertex. □

The proof also works for the empty cover of the empty open, whose section consists of the empty assignment. If the recipes are total, deterministic and have fixed source inputs, each region has a unique section by induction on rank. With alternative sources or permitted interpretations, a region can have several sections. The sheaf axiom describes which choices agree and can be combined.

The topos $\mathrm{Sh}(\mathrm{Open}(G))$ is therefore an ambient category for local reconstruction evidence [8]. A slice over a sheaf of allowed requests records evidence indexed by query type or authorization context. These constructions concern valid sections and restrictions. Storage size is a global functional on a presentation and does not automatically inherit a sheaf structure. In particular, two local archives can each store a copy of a shared block even though their sections agree on that block.

CHAPTER 6

A descent example with a shared base

Suppose u is an immutable shared block, v a verified delta against u, and w another delta against u. The dependency topology contains opens

$$U_v = \{u, v\}, \qquad U_w = \{u, w\}, \qquad U_v \cap U_w = \{u\}.$$

A local assignment on U_v consists of the decoded u and the resulting bytes of v; similarly for U_w. If the local assignments agree on u, there is a unique assignment on their union. This is the situation encountered when two image occurrences have edits against the same stored basis.

The same local assignments can have many physical realizations. One team can package the base independently with each delta, another can distribute a single shared base, and a third can use full materialization for v. They may agree on the relevant sections even if graph sizes, encoders and failover properties differ. Their comparison needs a common interface that includes the requested values and the permitted access behavior.

A read-only device export specializes this example. Each request selects an open that includes the physical dependencies of its sectors. Compatible successful reads that overlap agree on the shared sector bytes. An I/O error can leave a request with no section. The sheaf theorem applies to compatible sections that exist and does not force a failed physical read to succeed. A cache is sound when a value returned from a prior section is still validated against the same archive version and vertex identity.

CHAPTER 6

The observable range as a local section

Let entry e have a list of logical extents, and let $U_e(I)$ be the dependency closure of the blocks supporting interval I. At fixed archive version the range operation is a function

$$\rho_{e,I} \colon \mathcal{A}(U_e(I)) \longrightarrow \mathbb{B}^{|I|}.$$

It takes the appropriate byte slices and joins them in logical order. When $I \subseteq J$, one has $U_e(I) \subseteq U_e(J)$. Slicing the result for J to I equals restricting the assignment and then applying $\rho_{e,I}$. The equality follows by composing the corresponding affine offsets in each extent.

This relation supplies a naturality test for range APIs. A caller may divide a large range into smaller reads, cache the results, and join them. A correct reader produces the same bytes in either case. Sector geometry or block boundaries affect the list of physical dependencies, while interval restriction affects the logical output. The equality is also useful for independent verification: compare a full extraction with several overlapping range calls that cross physical block boundaries.

For graph encodings whose blocks have delta bases, $U_e(I)$ can be strictly larger than the intersecting extent list. Exposing the declared dependencies lets an optimizer calculate the closure before reading. A stream codec with no restart support instead places all relevant prior states in that closure. Locality depends on the selected presentation, and $\rho_{e,I}$ is stable under changes of presentation that preserve the required byte assignments.

CHAPTER 6

Local catalogs, permissions and equivalences

A hierarchy can index several kinds of openings of the same bytes. One model assigns to an address the set of available typed views and parser versions. Restriction along a path forgets unrelated siblings. A cover of a directory can be refined by subdirectories, but the namespace may contain aliases, and a policy can prohibit opening certain branches. Treat an alias as an additional occurrence label, and a policy filter as part of the address context. Then two visible branches can share storage without making their permission sets equal.

Use the groupoid-valued analogue of $\mathcal{A}$ when it matters to remember storage presentation isomorphisms. Compatible objects with isomorphisms on overlaps require the usual cocycle condition on triple intersections. Gluing up to coherent equivalence is a stronger assertion than the set-valued theorem above. For a concrete construction, one can keep assignments of vertex bytes strict and let a separate groupoid track diagrams of permitted renamings. The set-valued result applies to the byte projection.

This distinction guides a reader API. A byte operation returns a single string or an error. A provenance operation can return witnesses for which member, parser and storage graph supplied the string. Different successful witnesses can be equivalent under stable renaming or caching. The byte operation forgets that extra information, while the provenance operation retains the higher cells that compare witnesses. Chapter 13 applies these levels to indexing utilities inside a nested collection.

7 Extents, dependencies and range locality

An extent is a triple (b, o, ℓ) where b names a decoded block, o is an offset in it, and $\ell > 0$ is the number of bytes taken. Validity requires $o + \ell \le |B_b|$. If entry i has extents $e_0, \dots, e_{k-1}$, define cumulative logical offsets

$$a_0 = 0, \qquad a_{j+1} = a_j + \ell_j, \qquad x_i = \mathop{\Vert}_{j<k} B_{b_j}[o_j, o_j + \ell_j).$$

The extent order is the concatenation order. Physical block IDs and their offsets can appear repeatedly and need not increase. For an empty file the extent list is empty and the concatenation is the empty string.

A stored block descriptor supplies the compressed length, decoded length, a codec ID and an integrity check on the decoded result. A format with dependencies also supplies a base list or a recipe label. A verifier builds or checks a topological order and validates all bounds before following a descriptor. STC1 v1 uses independent physical block descriptors and implements sharing by letting several extents name the same descriptor.

For an interval $I = [p, q)$, extent j contributes exactly if $I \cap [a_j, a_{j+1}) \neq \varnothing$. This is a logical test performed before decoding that block. The read maps the intersection into the block by adding $o_j - a_j$, with arithmetic ordered to avoid unsigned underflow. The next page proves the result for complete entries, subranges and shared blocks uniformly.

CHAPTER 7

Range reconstruction

Theorem 7.1 (Intersection rule)**.** If all referenced decoded blocks have their declared values and the entry extents concatenate to x_i, then the ordered concatenation of the selected intersections for $I = [p, q) \subseteq [0, |x_i|)$ is $x_i[p, q)$. The empty range returns the empty string.

Proof. The half-open logical extent intervals partition $[0, |x_i|)$. Intersecting them with I partitions I. On the intersection with extent j, logical position r corresponds to block position $o_j + (r - a_j)$ by the definition of extent concatenation. Reading those subintervals in increasing j emits every requested byte once and in order. □

The inequality for a nonempty intersection can be tested as $a_j < q$ and $p < a_{j+1}$. A streaming reader advances a_j cumulatively, skips earlier extents and stops once $a_j \geq q$. The reference reader follows that method. A large-image catalog can store an in-memory prefix array and binary-search for the first intersecting extent, or exploit fixed-grid alignment and compute its ordinal directly. These improve metadata search without changing payload selection or the proof.

The theorem handles boundary cases explicitly. At $p = a_{j+1}$, extent j contributes nothing. When an image's final extent is shorter than the others, $q \leq |x_i|$ keeps the read within the validated final slice. A reader that assembles an entire entry as one in-memory buffer proves the same equality but has a different memory and access bound. The CLI streams each contributing slice into a staging destination.

CHAPTER 7

How far must decoding reach?

In a general resource graph, let S be the set of block IDs intersecting a read. Its closure $\mathrm{cl}(S)$ adds all transitive dependencies. If each resource v has an operation count $t(v)$ and output bound $b(v)$, a schedule evaluating every member of the closure once does work at most

$$\sum_{v \in \mathrm{cl}(S)} t(v)$$

for those recipes, plus indexing, validation and output. The sum of decoded lengths is at most $\sum_{v \in \mathrm{cl}(S)} b(v)$. This is a work bound, not the peak live memory. A dependency can be discarded when no later request in that schedule needs it.

For STC1 v1, $\mathrm{cl}(S) = S$. The Lean implementation retains one last decoded block per range operation and can decode a repeated, nonconsecutive physical block again. If m logical extents intersect the interval, at most m block decodes are attempted, with decoded-byte bound Bm for maximum block size B. The Python virtual-media reader keeps an LRU of up to four blocks by default and can reduce repeated decode work. Both designs require the index at open time.

An ISO member begins on an independent block boundary in the writer. For fixed block size B and a positive requested length n, its extent grid intersects at most $1 + \lceil (n-1)/B \rceil$ logical blocks. A 2,048-byte aligned sector is contained in one such block when B is a multiple of 2,048 and the sector exists in the image. Deduplication changes the physical block ID without changing those logical cut positions.

CHAPTER 7

Block identity and exact aliasing

A hash table quickly proposes potential equal blocks. The preimage for SHA-256 is the whole decoded block, and the digest is a lookup key. On a hit the writer retrieves the previously stored representative and compares the full byte arrays. Only equality creates an alias. Thus the representation remains correct even in the hypothetical presence of a hash collision. The index still contains a digest for corruption checking when data are read.

For each logical block j, maintain a map $\pi(j)$ to a physical ID. During construction it can name a new physical block or a checked existing one. The input file records provisional extents in logical coordinates. Once all blocks are sealed, replace each logical ID by $\pi(j)$ in its extents. Because the compared decoded arrays agree, replacing the ID preserves every permitted slice of the block.

In a cross-archive store a reference can outlive the particular packing run. It then needs stronger lifetime, garbage-collection and authentication semantics than a local ID. A public digest can reveal that two archives share an object, and serving a missing-object error can reveal whether it exists. The basic format is self-contained. It records all physical blocks inside the same archive and avoids needing a remote dictionary or cross-user cache for successful restoration.

CHAPTER 7

A graph extension for deltas

A prospective extension can store a node delta(b, δ) with decoded bytes $f_\delta(B_b)$. VCDIFF specifies an interoperable vocabulary for encoding differences using literals and copies, among other operations [23]. A writer can try a candidate base and verify the output before admission. The wire profile must identify which delta semantics are used and whether a copy reads from a base or already emitted target bytes.

Depth and closure limits become format policy. With dependency rank d and maximum out-degree k, the naïve upper bound on the number of reachable vertices is $1 + k + \cdots + k^d$, truncated by the graph size. For a chain the bound is $d + 1$. Repeated bases can decrease the size of the closure but increase the fan-out of users relying on one resource. The one-layer cost comparison of STC1 no longer applies unchanged because selecting a delta may require retaining a new base.

A reference can point to an earlier physical identifier to guarantee acyclicity by construction. If the index admits forward references, the parser has to verify acyclicity and a depth limit. The natural place to add such a feature is a format version with explicit dependency fields. A v1 reader treats codec IDs as independent-block filters, so a new codec cannot use undeclared archive dependencies while preserving v1 access guarantees.

8 Exact recovery from internal streams

Suppose w is an encoded member and $d(w) = y$ is its decoded byte string. Even if d is the decoder of a lossless compressor, it can have several distinct accepted wire inputs with the same decoded value. Gzip permits header metadata and multiple encoded choices [18, 17]. A fresh encoder e guarantees $d(e(y)) = y$ on its domain. Recovering w requires $e(y) = w$ or some supplementary information.

A reversible opening of w is a pair (y, r) with a reconstruction function g satisfying

$$d(w) = y, \qquad g(y, r) = w.$$

The residual r can be a format parameter, a difference against a deterministic re-encoding, or the complete original bytes. The last choice is always available and provides a baseline. When the view itself is useful, recording y in a derived index need not make y part of the primary storage cost. When the purpose is compression, the packer accounts for every new dependency and retained residual.

If the wire stream represents a sequence $x_1, \ldots, x_k$, the same definition uses a tuple. g must know the exact child interface, including order and representation. Two PDF objects with identical rendered text can have different bytes. Two boot ISO versions can expose identical directory entries and still have different catalog or padding bytes. Residual verification is against the original bytes, with application-level equality as a separately documented property.

CHAPTER 8

A constructive residual and its cost

Choose a deterministic reference encoder e_0. Let $w_0 = e_0(d(w))$. A delta encoder can construct $\Delta = \mathrm{diff}(w_0, w)$ whose decoder P satisfies $P(w_0, \Delta) = w$. Then

$$g(y, \Delta) = P(e_0(y), \Delta).$$

This construction works for every supported w when the delta format has an all-literal mode. It makes no claim that Δ is small. Exact recompression projects such as precomp and preflate explicitly address recovering particular compressed representations [25, 26]. The reference encoder and residual profile are part of the versioned dependency of the stored witness.

A byte-cost comparison for an optional candidate is

$$\underbrace{L(\Delta) + L(\text{recipe}) + L(\text{new support})}_{\text{opening and reconstruction}} \quad \text{versus} \quad \underbrace{L(w \text{ as stored})}_{\text{opaque representation}}.$$

Shared y can make the new support term small. A much larger decoded y can make it expensive. If the cost model also restricts random access, the packer must bound the cost of re-encoding a requested part of w, or retain a checkpoint or local representation.

To validate the candidate, reconstruct and compare to the full original member before publication. The check is local to its stated codec versions and reproducible inputs. An exact encoder that depends on environmental timestamps, machine-dependent dictionaries or unspecified heuristics should be pinned or replaced by a reconstruction procedure defined at the wire level.

CHAPTER 8

Nested witness composition

Proposition 8.1 (Reversible opening through a tree)**.** If every opened node has a verified reconstruction function from its immediate children and residual, and each opaque leaf is restored exactly, then evaluating the stored reconstruction tree yields the original wire bytes of every node.

Proof. Proceed by height. At height zero the retained opaque representation is exact. Assume the child wire values at a node are reconstructed. The verified reconstruction function then receives the same ordered tuple as in its check, so the node wire bytes agree. This establishes the claim for all nodes. □

For two nested compression steps, one can reconstitute the outer representation after reconstituting the inner payload, or precompose the recipes and execute one composite. The partial-domain condition is that the intermediate representation is valid input to its parent recipe. This is the algebraic condition carried by typed contexts and tracked in the chain category. A proof that only tracks normalized child content would not establish the proposition's byte-level conclusion.

A failure during candidate discovery leaves a valid opaque representation. A failure during decode of a committed structural recipe is an archive error because its proof obligations were not met by the observed data or execution environment. A second raw representation may provide deliberate redundancy, at a real storage cost. A plan that relies on redundancy for availability should make its recovery priority and verification rules explicit.

CHAPTER 8

PDF member streams

ISO 32000-2 specifies object syntax, stream filters, revisions and embedded files [42, 43]. A PDF can contain Flate-encoded streams, JPEG or JPEG 2000 image payloads, cross-reference streams and embedded file streams. An embedded file can itself be a PDF, an archive, or any admitted file type. A parser can display an associated file with its declared relationship and still preserve the exact containing PDF.

A byte-preserving structural candidate identifies disjoint original spans. For each recognized stream it records the original span and a reconstruction witness. Unrecognized regions, gaps and cross-reference data remain exact literal slices. A detector must respect declared lengths, indirect references, revision state and stream syntax. Searching for the token `endstream` in arbitrary compressed bytes can mistake data for a boundary. A parser's discovery result is a candidate, and the reconstructed complete PDF is the witness that bounds any mistaken parse.

Incremental updates can retain obsolete objects and reference a previous cross-reference section. Signed PDFs can bind specific byte ranges. Reserializing an object graph or combining pages can change those bytes even when the visible document remains the same. Full image equality is the relevant criterion in the main application, and full PDF byte equality provides the analogous criterion for documents. The experiment keeps original PDF members intact and exercises a limited, explicit exact-wrapper example separately.

CHAPTER 8

ISO internal streams and boundary-aware candidates

An ISO can contain an archive, compressed root filesystem, boot image and documentation simultaneously. A future structural packer can use filesystem extent discovery to propose block boundaries at the starts and ends of compressed members. It can preserve the system area, El Torito metadata and unassigned regions as literal pieces. Such discovery changes the candidate partition. It need not change the decoded bytes, and it can improve exact sharing when two revisions place the same internal member at different sector offsets.

A stronger candidate interprets the member's own compression. For an initramfs represented by a concatenation of compressed or uncompressed segments, the witness must account for ordering, padding and every member header. For an embedded filesystem, global references or metadata can prevent a short local witness. A format-specific comparison should measure whether reconstructing original sectors requires a large dependency closure. Boot access amplifies any hidden global dependency.

An implementable intermediate profile extracts only proven byte intervals, deduplicates them by equality, and records an ordered overlay with literal gaps. It can share original wire bytes even when the inner parser is unable to reproduce the exact compressor. The proof reduces to concatenation of an exact partition and the range intersection theorem. STC1 v1 uses sector-aligned fixed or content-defined blocks for ISO inputs and provides the baseline against which these more complex candidate partitions can be evaluated.

9 The STC1 wire format

A v1 archive is a finite byte string with four parts. The first four bytes are ASCII `STC1`. The next region contains stored block payloads in descriptor order, with no per-block framing inside that region. A footer-index follows. The last 24 bytes contain a 20-character, zero-padded decimal index length and the ASCII word `STIX`.

$$\underbrace{\texttt{STC1}}_{4} \parallel D_0 \parallel \cdots \parallel D_{k-1} \parallel I \parallel \underbrace{\mathrm{dec}_{20}(|I|) \parallel \texttt{STIX}}_{24}.$$

The right endpoint of the data region is the beginning of I. Given total wire length L and a validated footer length h, that endpoint is $L - 24 - h$. The archive can be written forward to a staging file, while the reader can discover the index with a tail read.

The writer commits to this version through the leading magic and matching footer. A newer format with dependencies or another binary grammar needs a different version marker. Payload boundaries are provided by the index, so payloads may include arbitrary magic-like byte strings. Searching the archive data for a marker cannot replace footer validation.

The format uses byte length and hashing of decoded blocks rather than file semantics. An image can be partially read regardless of its suffix as long as the extents are valid. The block-size, fixed or content-defined cut strategy, and classification policy belong to the writer. A reader infers exact reconstruction from the stored descriptors and has no need to rerun the writer's classification.

CHAPTER 9

Field framing and canonical primitives

A field is terminated by byte 0. Inside a field, data byte 0 is written as $(1, 0)$ and data byte 1 as $(1, 1)$. Every other byte is written unchanged. Byte 1 initiates a quote and its following byte must be 0 or 1. An empty field is the single terminator, and an empty field list is the empty string. A valid index contains at least its two count fields.

Index fields are UTF-8. Count, size, offset and ordinal fields are ordinary ASCII nonnegative decimal with no leading zeros except the number zero, and are at most 20 digits. The footer is a separate fixed-width decimal field whose leading zeros are mandatory when needed. A block codec ID is one to 32 characters from lowercase ASCII letters, decimal digits, hyphen and underscore. The decoded-block digest is 64 lowercase hexadecimal characters representing SHA-256 [27]. Category strings have one to 128 UTF-8 bytes. The source label has at most 4,096 UTF-8 bytes.

A strict parser rejects an escape followed by any other byte and rejects an incomplete or unterminated final field. It then requires the grammar of counts and rows to consume the whole field list. These rules eliminate multiple field encodings for the same index. Codec payloads can still have several valid forms that decode to the same block. Format conformance demands correct decoding, whereas reproducible packing additionally pins the writer and candidate configuration.

CHAPTER 9

Blocks, extents and occurrence rows

Row	Fields in order
Index prefix	Block count k, entry count m
Block b	Codec ID, decoded length, stored length, SHA-256
Entry i	Category, name, original byte length, extent count r_i
Extent j in i	Physical block ordinal, offset in decoded block, extent byte length

The two count fields are followed by exactly k block rows and m entry rows. Each entry row is immediately followed by its r_i extent triples. The first block's stored offset is 4. Each subsequent offset is the previous offset plus the previous stored length. The final endpoint must equal the index start. An entry's extent lengths must sum to its declared size. Each extent length is positive and the selected decoded slice must lie within the referenced block.

A decoded block is nonempty and at most 16 MiB. Its stored representation is at most 16 MiB plus 65,536 bytes. A custom independent filter can produce an empty representation for a particular decoded block, if verified. A raw block must have equal stored and decoded lengths. An empty file uses no extents. Block references can occur more than once across entries and within an entry. The header has no recursive references between block descriptors, so validating extents does not require graph traversal.

CHAPTER 9

Writing the physical representation

The writer classifies each input from limited leading bytes and the filename suffix. The built-in rule recognizes `STC1` and `%PDF-` first, then an admitted lowercase suffix, then the PNG signature for files lacking an admitted suffix, then `other`. An input with the category `iso` starts on its own independent block boundary. Other adjacent inputs of the same category can contribute to the same block. The reading order is stable within categories, and the index is emitted in original ordinal order.

As the current decoded buffer reaches its block limit, the writer hashes it and looks up a possible earlier representative. A verified byte match adds a logical-to-physical alias. Otherwise the writer selects a codec and appends the chosen payload. It maintains a block descriptor and physical offset for each newly stored block. Entries can reference portions of the current block before its final physical ID is decided. The writer remaps these provisional IDs once all buffers are finalized.

At the end the writer emits the index and footer. It refuses a destination that already exists and uses a sibling staging directory for the final rename. A reader validates the index before following extents, decodes each required block with its named filter, and checks both length and digest before using the block in a range. Stable archive contents and compatible filter executables are prerequisites for repeated reads.

CHAPTER 9

Format versions and local codec identifiers

The built-in identifiers are `raw`, `gzip`, `xz` and `zstd`. Raw stores exactly the decoded bytes. The remaining IDs refer to independently decodable streams from the corresponding command-line formats [18, 19, 20]. The reference writer invokes a fresh process per candidate block, so a block has no state from another block. User-defined IDs are local agreements pairing an encoder and decoder with the command-line tool. The index carries the ID as data. It cannot cause a command to be downloaded or chosen from an archive-supplied executable path.

An unknown ID is harmless to catalog inspection, because opening the index does not decode payload blocks. Extracting a range that requires that ID fails with a diagnostic unless a suitable locally configured decoder is present. This property supports an inventory service that lists images it cannot currently decode. To distribute an archive, include a complete codec compatibility record or use the built-in profiles.

An open format specification also needs a rule for reserved fields. Version one has a closed grammar and rejects trailing index fields. A new metadata or authentication layer can bind v1 bytes externally, or define a new container version. Ignoring unexpected fields would create ambiguity between readers with different interpretations, particularly if new fields changed the security meaning or dependencies of a block.

10 Choosing a compression plan

Fix input snapshots, a layout and exact byte-equivalence classes of its logical blocks. The block table has one row for each class. For decoded bytes x, candidate codec c and stored payload w, the part of the archive length varying with that choice is

$$\kappa(c, w) = |w| + |\mathrm{UTF8}(c)| + |\mathrm{decimal}(|w|)| + 2.$$

The constant two is the pair of field terminators for the codec ID and stored length. The fields use only ASCII bytes other than 0 and 1, so field escaping does not change these counts. The decoded length, digest and record counts are fixed. The footer size is fixed even if the index length changes.

Theorem 10.1 (Portfolio minimum at fixed layout)**.** Suppose the choices of stored payload for distinct physical blocks are independent. Among all admissible choices at the fixed layout, picking for each block a candidate of minimum κ minimizes total wire length.

Proof. Write the length as $K + \sum_b \kappa(c_b, w_b)$. The term K is the same for every candidate vector. Each sum term is minimized independently, so the sum of these minima bounds every vector and is attained by the indicated one. □

The candidate set comprises available encoders that complete and whose matching inverse returns the exact block. Raw is always admissible. If the representation cannot fit the reader's index limit, it is outside the publishable profile. If the index cap constrains the candidate vector, independence may fail. The implementation publishes the unconstrained minimum only when its index fits.

CHAPTER 10

Round-trip screening and deterministic tie handling

The writer begins with raw. It then runs each encoder on the same bytes. When a candidate is shorter under κ, it invokes the associated decoder with a bound and compares the byte arrays. A candidate that fails or differs cannot replace the incumbent. For a candidate with cost at least the incumbent, decoding cannot improve the minimum. The code uses strict comparison, so a tie retains the earlier candidate. This makes the tie rule observable and preserves raw if it has a tied cost.

With a stable input and deterministic tools, the fixed-layout algorithm is deterministic given the candidate ordering. Executable availability, configured limits and codec implementations can change results. An environment-dependent compressor may still be lossless and useful. Its representation must be decodable later under the codec identifier, even if a repeat packing run chooses another representation.

The optional layout portfolio compares fixed 512 KiB, 2 MiB and 8 MiB targets and a sector-aligned content-defined layout with a 2 MiB cap. It retains the shortest completed serialization when all its trials succeed. A smaller block size can share more equal material across ISO revisions. A larger block size can reduce index overhead or give a codec more history. The ordering between these sizes is a property of the collection and writer, so a numerical comparison uses actual wire length. The auto mode needs temporary space for the best archive and the current trial.

CHAPTER 10

Ordered segmentation as a shortest path

If cross-segment support resources are disabled, a category's contiguous partition can be optimized by dynamic programming. Let $i < j$ denote logical cut positions and let $w(i, j)$ be the exact cost of the best admissible independent representation for the segment $[i, j)$, including its additive index terms. Disallow segments exceeding the read budget by assigning infinite cost. Then

$$D(0) = 0, \qquad D(j) = \min_{0 \le i < j} \{D(i) + w(i, j)\}.$$

The final cut predecessor of any partition gives a candidate in this minimum. Conversely a predecessor of finite cost can be extended by its segment. Induction on j proves the recurrence. Storing a predecessor gives reconstruction of the optimal partition.

A shortest-path model admits cut sites at sectors, estimated file boundaries, content-defined cut points, or periodic checkpoints. Candidate generation can use a rolling fingerprint [15, 16]. Once endpoints are fixed, stored lengths must be measured or bounded by an explicit predictor; only measurements support the exact finite minimum. If global record-count digits or other fields vary nonadditively, add state for those fields or compare complete candidate serializations.

The STC1 writer uses a fixed block grid per layout trial and does not enumerate all feasible partitions. A more extensive search can use the same read semantics. Its implementation must budget the number of codec invocations and the cost of reading candidate spans, especially for multi-gigabyte images.

CHAPTER 10

Dependencies couple local choices

Let R be optional stored resources with nonnegative costs, and let every logical target t admit several candidate plans. A plan has a local cost and a requirement set $D(t, p) \subseteq R$. Choosing one plan per target incurs the sum of its local costs and the costs of the union of its requirements. Shared dictionary selection and delta bases have this form.

Proposition 10.2. Minimum-cost candidate selection with shared support resources is NP-hard, even when every candidate has exactly one requirement and zero local cost.

Proof. From a set cover instance with universe U and sets $S_1, \ldots, S_n$, create a unit-cost resource r_j for each S_j. Each $u \in U$ is a target. It has a candidate requiring r_j exactly when $u \in S_j$. Any feasible assignment selects a union of resources whose sets cover U. Any cover supplies a candidate for each u. Thus the two optima agree. Set cover is NP-hard [9]. □

This reduction addresses the abstract selection problem. It does not imply hardness for the independent block selector of this manuscript. A resource graph extension can solve small candidate instances exactly, restrict bases to cheap local neighbors, or use a heuristic and compare its complete serialization. Its proof obligation is losslessness and feasibility. Any claim about optimality must name the search class and objective.

CHAPTER 10

Capacity and useful variants

A boot drive with storage budget B must accommodate a bootstrap and recovery reserve B_0, then the selected image collection. Assign utility value v_i to occurrence i, and let $L(S)$ be the complete archive length for subset S under a fixed layout policy. The selection problem is

$$\max_{S} \sum_{i \in S} v_i \quad \text{subject to} \quad B_0 + L(S) \leq B.$$

Equal v_i gives maximum count. A utility that supports a unique machine class can be weighted more heavily than a duplicate release. When the objective counts covered capabilities, a set coverage function replaces the sum.

With independent sizes this reduces to the familiar additive capacity model. With exact sharing the marginal cost of adding a related ISO can be smaller than the cost of adding it to an empty collection. For a fixed resource set per image and additive block costs, the union cost is monotone submodular. Real repacking can change layouts and selected code lengths, so that structural property must be established for the chosen planning model.

A sensible evaluation records both the number of unique images and any repeated occurrences. A repeated occurrence can demonstrate exact deduplication without adding a new utility. The published-corpus comparison treats three distinct source images as distinct inputs and reports the effect of reusing their actual blocks.

11 Lean framing and the executable

The companion uses Lean 4.24.0 [10]. Its pure core works with finite lists of unsigned bytes and represents malformed input by `Option`. Let F be field framing and P the parser. The main lemma is $P(F(x)) = \mathrm{some}(x)$ for every finite list of finite byte strings. No assumption about content or size is required in that theorem.

The writer emits in reverse using a tail-recursive accumulator. For one byte b the token function returns $[1, b]$ when $b \in \{0, 1\}$, and $[b]$ otherwise. The proof relates reversing the final accumulator to the concatenation of those token lists. The parser state contains a quote flag, a reversed current field and a reversed list of completed fields. A one-token lemma changes the current field by exactly one byte. A one-field lemma then restores the accumulator to the boundary state and adds the completed field. Induction over the input list proves the framing theorem. A separate adapter lemma proves the same result on Lean's byte arrays.

The decoder detects malformed field framing before the catalog grammar is interpreted. Length checking, SHA-256 validation, block I/O and atomic staging are executable checks, not theorems about the Lean runtime or filesystem. The implementation uses the proven framer for the index. Large media data stream through independently stored byte arrays and ordered extent slices. The range correctness argument is the mathematical theorem of Chapter 7.

CHAPTER 11

A codec-agnostic round trip

The pure interface includes a decoder defined on the image of its encoder. The following excerpt omits the bodies of the two composition functions.

```
abbrev Octets := List UInt8
structure Codec where
  encode : Octets -> Octets
  decode : Octets -> Option Octets
  roundTrip : forall x, decode (encode x) = some x
theorem decompress_compress (codec : Codec)
    (files : List Octets) :
    decompress codec (compress codec files) = some files := by
  simp [compress, decompress, codec.roundTrip, Frame.decode_encode]
```

Here `compress` passes a framed sequence to `codec.encode` and `decompress` binds the result of `codec.decode` to the frame parser. For every `Codec`, the first simplification rewrites codec decode of encode to `some`. The second rewrites the remaining field round trip. Nested archives are admissible byte arrays and do not require a special case in this proof.

The companion also proves `checkedCandidate_restores`. If a black-box decoder returns the precise framed bytes of one sequence on a given candidate, parsing those bytes returns that sequence. This theorem separates a local witness from a globally verified codec. The CLI's block candidate check uses the corresponding byte equality at a smaller interface. A user can attach any suitable binary filter, and a non-inverting candidate is ineligible for storage.

CHAPTER 11

From mathematical fields to large images

Encoding an ISO as a single list of individual bytes is convenient for a small proof and expensive as an executable plan. The runtime holds one current decoded block, candidate buffers and small index structures rather than a full file-sized list. It reads regular files in block-sized chunks, seals each block and records extents. File boundaries within a block are represented in the index. The implementation can handle an individual file larger than a block and a sequence larger than the available in-memory byte buffer.

The mathematical composition theorem applies to ordered file fields. The streaming construction instantiates a different representation of boundaries. If each input is cut into ordered slices and each slice names an exact decoded interval, concatenating those intervals reproduces the file by the extent theorem. In the writer the array of occurrence rows is indexed by the initial argument ordinal. Thus reordering inputs for compression and resolving aliases preserve the sequence. Kernel checking is confined to the named pure theorems, while the streaming connection is justified by the stated invariants and tested through CLI operations.

A key performance distinction is data memory versus metadata memory. Candidate data are bounded by configured block sizes and temporary I/O. The reader loads the full bounded footer index and materializes its fields. A future reader can stream-parse it or page catalog sections. The v1 limit makes the current behavior finite and predictable for ordinary utility collections.

CHAPTER 11

The external codec boundary

The executable configures `gzip`, `xz`, and `zstd` as byte-oriented filters. It accepts additional filters by an identifier and two local executable paths. Each configured filter reads bytes from standard input and writes bytes to standard output. A wrapper can supply options or adapt an algorithm that normally accepts file paths. The stored archive never names the executable path. Decoder choice follows the local configuration and the checked codec ID. Nonzero exit, a time or output limit, and an incorrect round trip disqualify a candidate.

Subprocesses are invoked with positional arguments through a fixed redirection script. Data are passed through temporary files, preserving embedded zeros and non-text bytes. The command runner enforces wall-clock and output-file limits and an address-space limit for filter processes on its Linux profile. User-configured executables run as local programs with the caller's ordinary permissions. The resource controls are bounds for the ordinary pipeline, not a sandbox for arbitrary third-party code.

For a configured format with a stable decoder, validating $d(w) = x$ at encode time ensures the stored payload represents x. At read time the decoder checks declared byte length and the stored SHA-256 of x. Replacing an installed decoder under a custom ID can cause extraction to fail even when the archive itself is intact. A distributor should include codec versions and a decoding recipe in the companion metadata.

CHAPTER 11

Commands, checks and proof scope

```
lake build
strata pack utilities.stc --auto -- rescue.iso disk.iso guide.pdf
strata list utilities.stc iso
strata find utilities.stc rescue iso
strata range utilities.stc 0 32768 2048 pvd.bin
strata extract utilities.stc 0 recovered.iso
strata unpack utilities.stc restored
```

The catalog command emits JSON lines in ordinal order. The range command accepts an offset and length in bytes. Extraction into a directory places each occurrence under its ordinal, so duplicate labels can be recovered without overwrite. A manifest records the path used. An explicitly chosen output pathname is available for a single occurrence.

The build checks the frame theorem, its byte-array adapter, the generic composition theorem and the checked-candidate theorem. The executable tests include mixed category input, zero-length input, duplicate content and labels, nested archives and PDFs, an admitted local filter and a rejected lossy candidate. Public ISO comparisons use full byte equality in addition to range checks. A separate Python reader exercises interoperability with the same v1 writer and is used as the backing store for the experimental virtual optical device.

The release contains source and a Linux x86-64 executable. Linux-specific I/O and codec tools are runtime prerequisites, and custom codec definitions must be supplied again when decoding. A native Windows or firmware build is a distinct port.

12 Range equality and boot equivalence

Define a read-only device interface by its reported byte capacity N, logical sector size s, media type and its response to a legal read (ℓ, k). For a device backed by image x, the successful read is

$$(\ell, k) \longmapsto x[s\ell, s(\ell + k)).$$

A device can also report status, reject invalid requests, and take part in protocol negotiation. For a firmware consumer, these responses and the order of their delivery can be part of the observable behavior.

Let $A = E(F)$ and let occurrence i hold x. The range theorem gives the byte response of a Strata-backed device from $R(A, i, s\ell, sk)$. Length and geometry need to match the reference medium. An optical export advertises the same logical capacity as the original optical image and uses the sector mapping appropriate to that protocol. A different transport, such as a network block device, can deliver the bytes to a virtual machine whose virtual CD-ROM controller presents optical geometry to the guest.

The preservation result applies to any boot behavior determined by the media transcript in a fixed execution environment. Firmware and loaders can also observe clocks, hardware, errors, trust databases, driver initialization and the selection of a boot entry. Byte equivalence is a necessary interface property for the proposed method. Full boot parity is an integration property of a particular environment and device adapter.

CHAPTER 12

An observational theorem for a read-only consumer

Theorem 12.1 (Media transcript equality)**.** Let two read-only devices have equal geometry and agree on responses to every request admitted by a consumer. Fix the consumer’s initial state and external inputs. If each transition is a deterministic function of its state and the next response, then the two executions have equal state and request transcripts up to any finite step.

Proof. The initial states agree. Equal state determines the same next request and the hypothesis gives the same response. The transition therefore produces equal next states. Induction proves the assertion at every finite step, including a step that emits a boot marker. □

The theorem also applies to nondeterministic consumers under the same sequence of external choices and scheduling events. It does not manufacture a successful boot for an image that never reaches one on the reference medium. Its consequence is conditional: when the source execution reaches an observable success state and the adapter meets the response contract under the same conditions, the archive-backed execution reaches that state.

For a particular ISO, validation can compare reads of the volume descriptors, boot catalog and boot-image sectors before running the full test. These range checks are local evidence. They do not substitute for actual loader execution. The small test image in the companion emits a deterministic marker from its boot code, so the experiment can distinguish seeing a catalog from reaching the program at the catalog’s target.

CHAPTER 12

Materialization and virtual media

The simplest deployable path is to extract an image into an ordinary file and give it to a tool that already accepts ISOs. This path needs temporary storage equal to the selected image, and the extractor can calculate a source-matching digest before handing it over. The archive remains mounted separately from the selected image. A user can run an existing VM or supported multiboot workflow with the restored file.

A virtual-media path exposes the same bytes on demand. The companion includes a single-client, read-only NBD adapter and a range reader. QEMU can place a read-only NBD-backed node behind a virtual optical drive [39, 38]. The adapter parses the catalog once, retains a small decoded-block cache and answers bounded range requests with bytes checked against the block digest. A connected local socket is sufficient. The guest sees a virtual CD-ROM, while the host keeps the image in the compressed archive.

These paths differ in failure propagation. Materialization can verify the whole image before launch. On-demand reading can encounter a missing codec or damaged block after boot begins. A host can preflight the archive or selected image to detect such errors. An integration test should state which path it used and which environment delivered the virtual device. A QEMU host-side adapter and a bare-metal UEFI driver solve different transport problems.

CHAPTER 12

Bootstrapping and handoff on a real machine

A bare-metal design needs an independently readable bootstrap. It would open the archive from a firmware-accessible filesystem, select an image, verify its manifest, and expose the bytes in a form the next stage understands. An EFI application could implement a read-only protocol for its child images. For a Linux-based recovery launcher, the bootstrap could instead bring up a userspace or kernel block adapter and pass a virtual device to an existing user-space tool.

The operating system boot sequence can leave the firmware services behind. If it later needs files from the original ISO, it must rediscover the same backing bytes through a mechanism available after that transition. An early block protocol installed by firmware is insufficient when the next stage cannot retain or reproduce it. A supported loader/kernel pair can carry a handle, a device identity or an image-location parameter into its initramfs. Another route extracts the whole image into storage that remains visible.

Secure Boot evaluation is separate. Keeping image bytes exact preserves the inputs to any signature verifier, but the firmware still has its own allowed keys, revocations and execution policies. A new preboot adapter can be part of that trust chain. Packaging several signed images together does not confer authorization on the archive's bootstrap.

CHAPTER 12

Boot-focused layout and read budgeting

A utility image often has a small boot-critical working set followed by larger reads. A profile can keep volume descriptors, catalogs and loader regions in small independently decoded blocks, then use larger blocks for filesystem content. For a capacity-sensitive design, consider an optimization with weighted expected reads,

$$\min_{P} L(P) + \lambda \sum_{r \in \mathcal{R}} p_r \, W_P(r),$$

where P is a layout, $L(P)$ its serialized bytes, p_r a request weight, and $W_P(r)$ the physical or decoded work. The sector-alignment invariant constrains feasible cuts. The fixed v1 writer permits block-size and content-defined policies and can support extent layouts produced by an alternative writer.

A load trace from a verified guest can estimate $\mathcal{R}$. Such a trace belongs to one firmware, machine and guest profile. Changing the image order in a boot menu does not change its sector addresses, while changing block cuts changes physical requests. A warm LRU cache can coalesce repeated reads. Capacity estimates should separately include cache memory, the archive itself and any fallback staging space.

The host prototype represents a concrete migration path. It can test range-backed optical boot without writing to a disk or changing firmware. Subsequent ports can reuse the wire grammar and byte-level specification while substituting a native I/O and codec implementation.

13 A hierarchical index and search layer

A compressed utility filesystem can be defined as a read-only namespace over immutable archive versions. At the first level it presents categories and named file occurrences. A category is a view computed from catalog rows. It need not coincide with a stored directory or one contiguous compressed region. Each occurrence can be shown both in original order and in its category. A path component includes the ordinal so repeated names remain unambiguous.

Let v identify an archive version. An address can consist of v and a typed path

$$(i_0, \theta_1, i_1, \ldots, \theta_d, i_d),$$

where each i_j is an occurrence within a layer and θ_j selects the opening profile. The path can stop at raw bytes or request a listing of child names. A version makes cached offsets meaningful. When an archive is repacked, a stable occurrence identifier can preserve an application link if its ordinal mapping and source digest are carried across the migration.

The Lean command `list` reads the footer index and emits rows. Its `find` command performs case-insensitive substring matching over stored names, optionally restricted by category. These operations require no decoded data blocks. They are sufficient for discovering which utility image to open from a collection of known labels. A hierarchical index adds interpretation inside the selected occurrence and must record the parser and data bounds it used.

CHAPTER 13

Indexed reads through nested archives

The Python companion treats an archive as a source with size and a byte-range method. A member of that archive is another such source. If the member starts with a valid STC1 header and footer, it can be passed to the same archive reader. This permits a dotted address such as `2.0` to select member 0 of the archive stored at occurrence 2. The inner footer is reached by asking for a range of the parent member.

Reading an inner block in general involves two layers. The inner index gives the inner stored offset and count. The parent member maps this interval to one or more outer extents. Those outer blocks are decoded and checked first. The resulting byte string is then submitted to the inner block decoder. Induction over the nested address shows that each inner reader receives the same byte intervals it would have received from an extracted copy. This is a concrete composition of range-backed byte interfaces.

The Python tool limits dotted path depth and cache sizes. Opening a nested member can consume decoded blocks merely to find its footer and index. In a deep archive a single range may amplify work at several layers. A production filesystem can cache immutable inner indexes and use an explicit recursive read budget. An alternate format can expose additional address maps to avoid decoding entire outer blocks for small inner metadata requests.

CHAPTER 13

A primary ISO directory view

The companion searches primary ISO 9660 names by reading the volume descriptor and directory records through the member range interface. It recognizes a primary volume descriptor, checks that its logical block size is the supported 2,048 bytes, and follows the recorded root directory extent. The directory reader checks the paired little- and big-endian integers it uses, bounds directory bytes and depth, and omits the special current and parent entries.

This profile lists primary names. Rock Ridge, Joliet, UDF and other interpretations require corresponding parsers and produce separate views [31, 34]. The prototype can report a multi-extent record as such and does not claim to join all alternate file layouts. It can locate names like `ISOLINUX.BIN;1` without reading the entire outer ISO. Each emitted result is associated with its member address and a range in the image.

For an ISO containing a compressed filesystem, a second parser can use a slice source constructed from its extents, provided the parser supports the inner format. This is the same nesting law as for a direct nested Strata archive. A production implementation needs to account for an inner filesystem's decompression and indexing cost. A precomputed catalog of selected paths is a valid performance optimization when it is bound to the same bytes and parser version.

CHAPTER 13

Derived search structures and provenance

An inverted index for utility names or paths maps normalized tokens to occurrence addresses. For content search, the parser also records offsets or ranges of the origin so results can be verified. A row can carry

$$(v, a, p, n, h, I).$$

Here the fields are the archive version, occurrence address, parser ID, normalization ID, source digest and logical interval. Names, paths and terms have different normalization rules. A casefolded label query can retrieve several distinct byte labels. An ISO version number in a primary name and a display-friendly alias may both be useful.

A derived index can be generated incrementally from cacheable subtrees. When an input's exact content and parser version are unchanged, reuse the old derived records and update the occurrence address mapping if required. If a parser fails or a budget is reached, record the status alongside the indexed region. A negative search over a partial catalog has different meaning from a negative search over a completely inspected scope.

Keep the text index separate from lossless payload verification. Reading a matched byte interval can validate a query's origin. Proving that no match exists requires a completeness condition on parsing and normalization. Such an index is compatible with the sheaf view of local byte assignments, but the search engine also needs a versioned statement about which local regions it covered.

CHAPTER 13

From immutable archive to a virtual filesystem

A filesystem frontend can map `readdir` to catalog or derived rows, `getattr` to declared sizes and metadata, and `read` to the range method. In a read-only prototype, writes, renames and links are rejected at that frontend. An update can build a new archive version, verify it, and atomically switch a reference. Active readers pin their original version until their handles close. That policy provides coherent reads across long-running extraction and nested searches.

The implementation does not need to materialize all nested files to list the namespace. It opens a typed layer on demand and caches authenticated catalog data. Peak memory is controlled by decoder limits and cache policy. Concurrent lookups can share immutable decoded blocks, but should use distinct temporary filter paths or a worker pool. The command-line reference implementation uses one temporary directory and is sequential within an invocation.

This design can expose a media namespace and a human-oriented document namespace over the same archive. For example, a boot menu lists only entries tagged as utility images while a search frontend discovers PDF documentation and embedded rescue metadata. Shared physical blocks remain implementation details. Stable occurrence identity, verified bytes, and explicitly typed opening operations are the contract visible to callers.

14 Failure semantics and resource limits

An archive is untrusted input to the reader. Validation begins with size, leading magic, footer width and bounds. The index grammar is decoded only after its byte length is checked. The implementation then validates block offsets, row counts, extent bounds and length totals before reading block payloads. Each filter output is checked against its declared length and SHA-256. The selection of a local filter occurs after the index's codec identifier has been validated.

The integrity statement is conditional on trusted expected values. If an adversary can change both the stored block and its digest in an unsigned index, a digest comparison alone cannot authenticate the replacement. A distribution can authenticate the entire archive digest or a signed manifest that binds the header, entry order, names, extents, codec IDs and block digests. If partial download is important, an authenticated root with proofs for accessed descriptors can avoid downloading every payload for verification. Merkle-tree protocols provide relevant patterns [29].

Corruption, policy failure and absent data have distinct meanings. A truncated footer is a framing failure. An unsupported codec is a capability failure. A mismatched decoded digest is a data-integrity failure. A nested parser rejecting an embedded child is an interpretation failure and does not invalidate restoration of its outer bytes. Diagnostic messages and tests can maintain these distinctions.

CHAPTER 14

Expansion, nesting and cumulative budgets

An archive can describe a large logical result from small physical data. Repeating a block reference m times yields a logical length $m|b|$ with one stored representation. Valid byte equality places no upper bound on that amplification. A length check per block protects against one class of decompression attack, but an application needs request-wide limits for all emitted bytes and all selected entries.

Nested streams can compound amplification. For a sequence of allowed expansion factors $\alpha_1, \ldots, \alpha_d$, an opaque upper estimate is their product. This estimate can be too large or too small when dependencies overlap or codecs have hidden memory costs. A robust reader accounts for actual decoded input/output at each layer and carries a remaining budget through recursive calls. Useful dimensions include decoded bytes, output bytes, allocated index records, dependency depth, operations and wall-clock time.

The CLI bounds each block and its codec invocation, the size of the footer index and the candidate output. An unpack operation intentionally emits all declared members and has no archive-wide output quota. It should be run with an appropriate destination capacity when input is untrusted. A filesystem and virtual-device service can instead bound each request, cache, and concurrent worker pool. A failed request returns an error rather than interpreting a partial byte string as a valid full record.

CHAPTER 14

Equality checks, confidentiality and support reuse

The writer verifies a block alias by byte comparison. SHA-256 provides an index for possible matches and a practical integrity check [27]. This arrangement separates the mathematical equality claim at packing from the computational integrity assumption during reading. On-demand access to an archive received from elsewhere may need both integrity and origin authentication. Transport-level confidentiality alone does not bind the semantics of every selected image.

Deduplication can expose cross-input relationships through stored size, reference counts or presence checks. The side-channel problem is well documented in cloud storage [28]. A local boot utility archive usually has one owner and one distribution boundary. A hosted multi-user index can partition dictionaries by trust domain and avoid exposing comparisons with another user's data. Encryption or signing should be specified at a versioned layer. Authenticated encryption combines confidentiality and integrity under its key and nonce requirements [30].

Compression before encryption can leak information when an attacker influences plaintext and observes sizes. A static signed archive of public utilities differs from a service that mixes private and adversarially supplied files in a shared compression context. Format choice, deployment context and threat model determine which claims apply. The storage proof concerns byte reconstruction and depends on no confidentiality assumption.

CHAPTER 14

Names, publication and process isolation

A reader should not interpret archive names as arbitrary paths. The CLI writes whole extraction under new ordinal directories, sanitizes path delimiters and control characters for the materialized basename, and records the original label in a manifest. The single-entry operation takes its destination from the caller. Staging and no-clobber publication guard against partial visible outputs on ordinary process failure. Crash durability across power loss additionally requires filesystem-specific synchronization and recovery policy.

Custom codecs are local programs chosen by the operator. The writer's check can detect a program that fails to invert a candidate. It cannot certify the program's side effects or later behavior. The filter resource controls bound conventional output, runtime and virtual memory. An operator handling unfamiliar codec binaries needs an isolation policy appropriate to those programs. A version of the archive can restrict itself to a known codec allowlist for automatic processing.

Archive inputs should remain stable during packing or reading. A pack operation opens files again after classification, and its exactness applies to the byte stream it reads. For a source that may change concurrently, use snapshots or immutable copies. A reader can pin an opened file in a long-running service and authenticate its version so replacement paths and stale cached data cannot mix interpretations.

CHAPTER 14

Validation as a sequence of claims

The checker first validates container structure. It then verifies each requested decoded block's exact length and expected digest. If an archive was built by the verified candidate path, its block and extent invariants link this output to the original byte strings. An interpreter then checks the format of a member before claiming its nested structure. If boot is requested, firmware or the guest adds its own acceptance conditions. Each stage can fail without implying that earlier, narrower claims were false.

This staged validation is useful for recovery. A partial corruption may affect one physical block and all occurrence extents that reference it. The catalog can identify those occurrences without reading unrelated data. A duplicated support resource would trade space for resilience. A separate release digest identifies whether a restored file should be treated as the intended source image. No-checksum recovery of a damaged image is a different operation from the lossless decoder.

Experiments exercise corrupted payload, truncated trailer, range bounds and an existing destination. Parser fuzzing, systematic denial-of-service evaluation and a signed distribution are further engineering tasks. A production boot frontend should bound its dependency closure and record whether each image passed a verified test in the firmware mode it offers.

15 Evaluation on bootable ISO collections

The main storage experiment uses three published Tiny Core x86 optical images. Core 16.1 and TinyCore 16.1 came from the ibiblio archive. Core 16.2 came from the Princeton mirror's 16.x release directory. The companion records the download URLs and measured SHA-256 values. The downloaded files matched the accompanying release MD5 files. MD5 here is a provenance cross-check, while the SHA-256 values pin the exact experiment inputs [44, 45].

Image	Original bytes
`Core-16.1.iso`	19,410,944
`Core-16.2.iso`	19,412,992
`TinyCore-16.1.iso`	25,165,824
Total	63,989,760

The test machine is Linux x86-64. Lean 4.24.0 compiled the writer and CLI reader. The compression utilities were gzip 1.12, XZ Utils 5.4.5 and Zstandard CLI 1.5.5. The default portfolio used gzip level 9 with a deterministic header, xz preset 6 with one worker and zstd level 15 with one worker. Timing includes candidate execution, verification, hashing and serialization. It is a single observed run per displayed policy, useful for reproducing approximate work and unsuitable as a throughput distribution.

Each profile produced an independent archive. Unpacking recovered all three members. The program compared each recovered byte array to its corresponding original and independently parsed the stored block table to calculate index overhead and sharing.

CHAPTER 15

Measured size and layout selection

Policy	Archive bytes	Physical blocks	Extents
Raw, fixed 2 MiB	63,993,007	32	32
Raw or zstd, fixed 2 MiB	62,352,470	32	32
Full portfolio, fixed 512 KiB	62,284,309	124	124
Full portfolio, fixed 2 MiB	62,271,978	32	32
Full portfolio, fixed 8 MiB	62,276,363	9	9
Full portfolio, sector CDC	**38,848,049**	74	119

The raw fixed profile contains 3,247 bytes of index and markers. The full fixed 2 MiB profile saves 1,717,782 bytes relative to the original inputs, approximately 2.68%. No fixed profile in the table found identical aligned blocks across the three images. The best full fixed archive stores 15 raw blocks, seven gzip blocks, nine xz blocks and one zstd block.

Sector-aligned content-defined chunking (CDC) saves 25,141,711 bytes, about 39.29% of the original total. The 119 logical extents use 74 physical blocks. The unique decoded payload subtotal is 40,130,560 bytes, so exact reuse accounts for 23,859,200 bytes of the original subtotal. Compression then reduces the stored block payloads to 38,840,321 bytes, with 7,728 bytes of index and markers.

The result identifies the key improvement on this corpus. Selecting a stronger codec over the same fixed blocks changed the result modestly. Moving the boundaries to content-determined sector positions exposed shared byte intervals that fixed alignment had obscured.

CHAPTER 15

Independent and solid comparisons

A separate baseline compresses each ISO independently with gzip, xz and zstd. The smallest of those representations for each image used xz and totaled 62,275,044 bytes. That subtotal does not include an independent catalog or integrity manifest. The solid baseline is a deterministic USTAR stream of the three named files, compressed as a unit. Its best observed candidate also used xz, at 62,274,180 bytes. Its file boundaries and names are contained in the tar stream. In that layout, reading a late image range with a sequential decoder can require processing earlier data.

The best fixed Strata archive is 2,202 bytes shorter than the measured solid baseline, a difference of approximately 0.004% of the source. The comparison does not support a broad compression-rate conclusion between those two layouts. The major result is the 38,848,049-byte content-defined archive, approximately 37.62% below the measured solid xz baseline. It retains independent blocks and exact sector reads.

The benchmark describes three related releases from one family and uses a finite portfolio. The effect depends on byte-level overlap and the alignment at which it can be found. Images from unrelated projects, or independent encodings of the same filesystem, can provide much less exact sharing. The full trial results include each baseline codec's size, per-image checksums, archive lengths, physical block and extent counts, and all tested read ranges.

CHAPTER 15

Sector ranges and a primary directory query

The cross-reader range experiment used the full fixed 2 MiB archive and six queries per ISO. The set included a 512-byte system-area slice, the 2,048-byte sector at logical block address 16, the next descriptor sector, a 4,096-byte interval straddling the first 2 MiB boundary, the last complete sector, and an empty range at end of file. Each query was evaluated by the Lean CLI and by an independent Python parser using the same wire grammar. The results were compared with the literal source intervals.

The cold Python reader decoded one block for each of the four nonempty single-block queries, two for the boundary-spanning request, and zero for the empty interval. All 18 byte comparisons passed. The physical block sizes and accessed offsets are part of the recorded profile. In the content-defined layout, complete optical sectors remain within one block because cuts are made at sector boundaries. A request crossing a content-defined boundary can require two or more blocks, according to the extent list.

The primary ISO view found `ISOLINUX` paths in all three images. This check exercises format interpretation over the archive range interface and does not certify each published image's firmware boot behavior. The controlled boot experiment isolates that second claim using a purpose-built fixture.

CHAPTER 15

BIOS and UEFI boot observations

The controlled El Torito image was built with xorriso 1.5.6. It contained a BIOS no-emulation boot image and an EFI system partition carrying a minimal x86-64 EFI application. Each boot path writes a fixed marker to QEMU's debug console and exits through the debug-exit device. The fixture was 4,575,232 bytes. Its Strata representation was 2,465 bytes with 64 KiB target blocks. Most of the fixture is empty optical and FAT allocation space, so this compression ratio measures fixture construction rather than a typical utility image.

Firmware	Original ISO	Restored ISO	Archive-backed optical device
SeaBIOS	Marker, 85	Marker, 85	Marker, 85
OVMF UEFI	Marker, 85	Marker, 85	Marker, 85

The QEMU version was 8.2.2. In the archive-backed trials QEMU used a read-only NBD node behind an emulated IDE CD-ROM and a connected local socket. The BIOS path made five successful reads and decoded two top-level blocks. The UEFI path made 54 successful reads and decoded two top-level blocks. Exit status 85 is the expected encoding of the fixture's chosen debug-exit value. All six trials reached the firmware-specific marker. The BIOS and UEFI results concern this fixture and these emulated optical interfaces. Hardware firmware, guest handoff for full operating systems, and Secure Boot configurations remain separate integration cases.

16 Nested archives and document experiments

A second experiment checks the recursive model without assuming that outer compression can always exploit inner decoded similarity. Start with four occurrences: a 131,072-byte deterministic pseudorandom stream, a repetitive text file, an empty stream and a second occurrence of the first byte stream. Pack them into an inner archive with a 32 KiB block target. Its exact size was 131,871 bytes, including 705 bytes of index and markers. Six stored blocks supported ten extents. Four physical blocks were raw and two used zstd.

Make two valid one-page PDFs. Each has an embedded file named `bundle.stc` whose decoded value is that same inner archive. One Flate stream uses zlib level 1 and the other level 9. The two compressed stream byte strings differ. They are 131,759 and 131,547 bytes respectively, and their decoded bytes are equal. The PDF files include the corresponding lengths, object tables and byte offsets. They are 132,649 and 132,437 bytes.

Pack (A, inner, B, A) as an outer collection. The ordinal order crosses the PDF and Strata categories. All four restored files were compared byte for byte against the originals. An independent PDF attachment reader recovered each original embedded archive. Extracting the final occurrence inside either attachment yielded the same 131,072-byte source. This chain exercises

$$\text{outer archive} \longrightarrow \text{PDF} \longrightarrow \text{embedded archive} \longrightarrow \text{payload}.$$

It is a concrete case where each decoder exposes another compressed sequence.

CHAPTER 16

Equal decodings, unequal wire bytes

The two Flate streams are an explicit counterexample to the proposed identity $e(d(w)) = w$ for every accepted w. If $d(w_1) = d(w_2)$ and $w_1 \neq w_2$, a function e maps the common decoded value to one byte string. It can reproduce at most one of the two. The distinct stream lengths in the experiment make the inequality observable even before byte comparison.

The PDF generator supplies enough information to write both variants intentionally. One could store the compressor version, input, level and exact surrounding spans as a candidate witness. That candidate still depends on reproducible compressed output for each declared encoder. An external PDF would generally require a reconstruction residual or a wire-level recipe. The experiment uses the stored opaque PDF bytes for its main round-trip claim and uses the two Flate streams to illustrate the reconstruction requirement.

Both embedding and outer packing preserve the nested archive's exact bytes. A graphical viewer might render two documents with the same page content even when their attachment encodings differ. The archive decoder has a stronger obligation. Any replacement intended to preserve only appearance belongs to a distinct optimization objective and should be offered as a separately labelled derivative.

CHAPTER 16

Nested random access and small-file layout effects

The independent reader resolved address `1.1`. First it opened occurrence 1 as an inner Strata archive, then selected its occurrence 1, the text stream. No intermediate archive file was required. Resolving the child index caused outer block reads at offsets determined by the inner header and footer. The same recursive mechanism can export an ISO stored several archive levels down, subject to depth and codec constraints.

The outer test archive was 530,221 bytes. Its raw input occurrences totaled 529,606 bytes. It used 18 physical blocks for 20 extent references and 1,782 bytes of index and markers. This case grows by 615 bytes. The repeated PDF bytes do not all fall on matching blocks, because same-category small-file packing carries the current block across file boundaries. Independent codecs removed only part of the overhead.

The test is informative for policy. A category that contains nested PDFs of modest size can benefit from local joint compression. A repeated file whose beginning moves relative to the fixed grid may instead benefit from an independent file boundary or content-defined cuts. The trial records its actual negative result and preserves an exact raw fallback at the block level. It shows why archive overhead belongs in the optimization objective and why raw-block fallback alone does not guarantee that a complete archive is smaller than its input subtotal.

CHAPTER 16

Custom filters and invalid candidates

A local test codec encodes an all-zero block as its 32-bit length and decodes that length into zeros. It does not define a total lossless compressor on all byte strings. For the test input its four-byte wire representation passed the exact round-trip check and was stored under a custom identifier. Extracting the block without that local decoder failed. Supplying the configured decoder restored exactly 2,048 zero bytes.

A second local executable returned success and no output. The matching executable also returned no output, so the round-trip check failed for the nonempty block. The packer retained the raw representation. This test distinguishes a command's success status from the semantic property required for lossless storage. An encoder can be useful on a narrow domain and rejected everywhere else.

Additional negative checks changed a raw payload byte, truncated the footer, requested a range past end of file and attempted to overwrite an existing archive. The CLI rejected each operation. The damaged data failed the decoded digest test and did not publish a result. The tests target specific failure modes of the observed implementation. They complement the Lean proof of pure framing, whose statement quantifies over all finite input sequences.

CHAPTER 16

What the examples validate

The real ISO experiment verifies that selected layout policies affect cross-image compression on a published corpus. The boot fixture verifies a read-backed optical transport in two named virtual firmware paths. The document experiment verifies a different kind of nesting, in which a parser exposes an embedded payload that itself has an archive index. Each supplies evidence at a different interface of the higher categorical model.

A new recursive format extension can use the same experimental design. First compare original and restored bytes. Next exercise selected inner reads through each candidate plan. Then record dependency closures and check whether coarser and finer evaluation plans agree. Finally compare wire size under a declared candidate set. A candidate that reveals much more logical structure can be valuable for indexing even when it costs extra bytes to store exact reconstruction witnesses.

For other homogeneous families, the baseline needs comparable bytes and features. A PNG series can test identical stored image data at shifted offsets. A sequence of versioned PDFs can test embedded fonts and common images. A collection of firmware or package files can test repeated compressed payloads and metadata changes. In each case the crucial question is whether reusable equality exists at the chosen boundaries, and whether the cost of exposing or representing it fits the access budget.

17 Capacity, migration and deployment

For a utility collection, begin with a capacity ledger. Let T be the usable filesystem space. Let M include a bootstrap, manifest, reserved recovery components and any fixed overhead. Let Q be required scratch space for the chosen operating mode. The available archive capacity is $T - M - Q$. For a virtual read-only device, Q can be small disk space and a separate memory cache budget. For materialization, Q must accommodate the selected uncompressed image on a suitable filesystem.

If a collection of three inputs occupies 63,989,760 bytes and its indexed archive occupies 38,848,049 bytes, 40 MiB of available archive capacity holds all three in the measured sector-CDC representation. The same capacity holds the two smaller originals but cannot hold all three as original files. This numerical example assumes the capacity already excludes the reserved overhead. It illustrates the user's objective in units of available images, separately from a compression percentage.

A utility menu can count capabilities instead of image names. A capability row can describe architecture, firmware support, the recovery functions a tool provides and an independently verified version. Several versions of one image can be valuable when compatibility differs. Large overlap between them changes their marginal storage cost. Record which image served a test and the provenance of its observations so a newer version's boot claim does not silently inherit an older one's result.

CHAPTER 17

Operating the compressor on a new collection

Start with immutable source images and record their sizes and trusted checksums. Category `iso` is inferred from a lowercase-normalized suffix in the reference writer. A mislabeled file still preserves its bytes but can receive a less suitable layout. The 2 MiB fixed default is a baseline. Use `-iso-cdc` to propose sector-aligned content boundaries, or `-auto` to compare the fixed and content-defined layouts as complete archives. The fast policy limits the built-in codec portfolio to raw and zstd.

Content-defined cuts sample 16 bytes at 128-byte intervals in each 2,048-byte sector, starting with the first byte of the sector. A 64-bit rolling calculation within that sector selects a fixed residue modulo 256. The writer permits a cut after a 64 KiB minimum, unless the configured cap is smaller, and forces one at the cap. A matching region shifted by whole sectors has identical sampled anchor predicates. If both paths take the same anchor and their later sector bytes agree, their subsequent cut choices agree until the content differs or one entry ends.

The sampling hash proposes cuts. It has no authority to establish equality or integrity. Final decoded blocks must be compared in full before storage is shared. Repetitive or adversarial sectors can skew the spacing of anchors, so maximum block size and exact byte checks remain separate invariants.

CHAPTER 17

Capacity planning with marginal costs

Suppose each image i requires a fixed set R_i of block resources and a private index cost h_i. If resources have fixed costs $c_r \geq 0$, define

$$L(S) = K + \sum_{i \in S} h_i + \sum_{r \in \bigcup_{i \in S} R_i} c_r.$$

For $S \subseteq U$ and $j \notin U$, the resources newly needed by j under U form a subset of those newly needed under S. Therefore

$$L(S \cup \{j\}) - L(S) \;\geq\; L(U \cup \{j\}) - L(U).$$

This is diminishing marginal storage cost. A bundle of closely related ISO releases can be cheaper than the sum of its independent representations.

The model is exact for a frozen resource plan with additive metadata. The current writer compares complete serializations, whose count fields and block IDs may change. Computing marginal costs by repacking candidate subsets can therefore give a different answer from the frozen-resource calculation. A practical planner can use the frozen estimate to propose a small set of candidates and then measure their exact packed lengths.

When utility benefits are all one and storage is additive, sorting by size and taking the smallest that fit is optimal. An exchange replaces any chosen larger item by an unchosen smaller item without reducing count or feasibility. Shared storage breaks this exchange argument. A group of larger but related images can collectively cost less than unrelated small images.

CHAPTER 17

Migration, compatibility and preservation

A conservative migration takes an old collection, reads its declared image checksums, creates a new STC1 archive and verifies restored bytes for every occurrence. It then tests required boot cases against restored or archive-backed media. Only after those steps does it switch the advertised archive version. Keeping a small independently readable recovery image helps recover when a decoder executable or format version changes.

Publishing an archive can include a versioned manifest with source release links, SHA-256, category, media geometry expectations and test observations. The BagIt specification and Oxford Common File Layout illustrate independent fixity and preservation-oriented packaging practices [46, 47]. Those schemes have different storage and access goals from a seekable utility archive, but their inventory disciplines transfer directly.

A distribution that uses custom codecs also needs an auditable specification and portable decoder for each custom ID. The reference adapter implements built-in wire families. The compiled CLI targets Linux x86-64 and can be used from a compatible Linux environment. Porting to another host substitutes equivalents for range I/O, staging and subprocess control. The format itself is independent of the operating system used to produce it.

CHAPTER 17

A progression from archive to filesystem

The initial deployment exposes archive-level list, find, extract and range operations. The nested view adds archive addresses and primary ISO path search. A virtual optical export permits an emulator to use an ISO without first creating an expanded file. These pieces exercise the proposed architecture using the same original byte contract. An operating-system mount can use the range and catalog APIs and add its own caching, permissions and update policies.

A later writer can add variable block candidates around known inner ISO extents and reusable compressed components. Its verification routine must continue to test reconstruction of every original sector it changes. A later reader can use paged or authenticated index nodes to avoid loading a large catalog wholesale. These changes are separable. The public index can improve while the underlying wire format and block semantics stay fixed.

The right production criterion is capacity under a stated compatibility profile. Report bytes needed for the archive and bootstrap, the number and capability coverage of included images, peak scratch usage, cold and warm read work, and the tested firmware paths. These measurements show whether an additional available utility is a consequence of compression, a change in the mix of tools, or a smaller reserve.

18 Relationship to prior work and synthesis

General-purpose compression already provides the individual stream codecs used here. Deflate combines dictionary-style matching and entropy coding [17]. XZ and Zstandard provide independent stream or frame formats with different speed and history tradeoffs [19, 20]. Solid archive formats can jointly compress related files, and the 7z format offers configurable compression methods [22]. The primary comparison in this manuscript is to a compressed tar of original images, because it supplies a simple full-sequence storage baseline with boundaries and labels.

A seekable compressed stream adds restart points and a map from logical offsets to frames. Zstandard's seekable format documents one such arrangement [21]. STC1 uses independent payloads and extends the indexing concern to a collection of ordered file occurrences. A file occurrence can concatenate slices of several physical blocks, and distinct occurrences can reuse those blocks. A reader sees a stable logical interface even when a file's physical references point backward.

SquashFS and EROFS bring compression and random access to read-only filesystem images [40, 41]. A utility ISO can already contain one of them. Treating an ISO as an immutable byte address space makes the outer archive compatible with such inner choices. Recreating an inner filesystem is a distinct operation with its own exactness and boot obligations.

CHAPTER 18

Deduplication, differencing and reconstruction

Rsync uses rolling checksums and strong matching to find pieces that need not be transferred again [14]. LBFS uses content-based fingerprints to exploit file similarity despite shifts [15]. FastCDC develops efficient content-defined chunking for deduplication [16]. The sector-aligned policy here specializes this lineage to optical-image range access. It keeps each sector intact and tests equality of resulting blocks before sharing.

Git packfiles provide an established example of multiple object representations and explicit delta bases [24]. VCDIFF standardizes a portable representation of differences [23]. These systems motivate an optional resource graph. The independent-block STC1 profile instead makes the dependency closure of a data block trivial, which simplifies cold sector reads. A graph extension should compare its added compression with the extra decode work and failure fan-out introduced by bases.

Exact recompression of internal encoded streams has practical precedents in projects such as precomp and preflate [25, 26]. Their goals explain why the manuscript requires reconstruction witnesses. Decoded equality alone can be used for an index or as a hint for candidate selection. A reproducible encoding or residual is what makes that equality useful for byte-exact storage.

CHAPTER 18

Higher categories as interface semantics

The recursion of actual compressed streams supplies the operations. A node describes an ordered family of child ports and a reconstruction rule. Substitution plugs one verified layer into another, and a chain of these substitutions gives a simplex of the context nerve. Presentation isomorphisms supply changes of naming or structure over the same observable value. The Rezk construction records these alongside the chains [3].

The role of higher coherence is concrete. An inner archive can be read by extracting it and interpreting it separately, or by composing the outer member-range map with the inner offsets. A recipe can be scheduled with different parenthesizations and independent sibling evaluations. When they satisfy the same byte interface and resource conditions, these plans agree on the requested result. The refinement space gives the coherence of the association choices, and the dependency sheaf gives the gluing of successful local byte assignments.

The categorical tools are used with the level of structure present. Ordered archive occurrences do not acquire a symmetry from deduplication. Cost optimality does not follow from an equivalence of presentations. A presentation may be changed by an isomorphism while preserving data and increasing its physical representation length. Treating these distinctions explicitly lets the formal model explain both correctness and why a search over layouts remains necessary.

CHAPTER 18

Results that transfer to other collections

Three mathematical results transfer without format-specific parsing. First, lossless codec composition with an injective frame preserves arbitrary ordered sequences. Second, an extent partition and exact block values imply the range reconstruction theorem. Third, a fixed independent layout with additive serialized costs admits a per-block optimum over its candidate portfolio. Any format family with the same interface can use these results.

Sector-aligned content boundaries use additional structure supplied by ISO media. Other formats can choose different atomic units. A PDF importer can propose verified stream spans, a sequence of PNGs can use chunk boundaries, and a package store can use member descriptors. The data must still be compared byte for byte before any reuse. If the format stores a structural witness, the writer must verify the composed reconstruction against the original wire representation.

The corpus result cannot be exported to an arbitrary family. Its reusable byte overlap was demonstrated by the reduction from 119 logical extents to 74 physical blocks. A different corpus needs the same inventory and equality checks. The engineering value of the common interface is that one can repeat the experiment with other layouts or codecs without changing how extraction and sector reads are judged.

CHAPTER 18

Conclusion and testable extensions

The reported archive stores three related published ISO images in 38,848,049 bytes and restores all three exactly. The implementation selects codecs by serialized cost, can compare a finite family of block layouts, and admits user-supplied filters only through checked reconstruction. The Lean proof covers the generic sequence frame and codec composition. The archive format uses the verified framer for its index and a separate streaming proof argument for data extents.

The virtual-media fixture reaches a BIOS and a UEFI boot marker under QEMU, including when its media reads are supplied from the compressed archive. The nested PDF experiment exercises a chain whose wire representations cannot in general be regenerated by decode and fresh encode alone. The tested name views show how a hierarchical search system can expose actual child streams without materializing every file.

A further study can compare sector fingerprints, filesystem-informed partitioning and bounded delta graphs across a heterogeneous set of independently bootable tools. It should pin the same restoration, range and provenance checks and record storage and cold-read tradeoffs. The formal semantics gives the invariant for such changes. A new presentation or schedule is admissible when it preserves every demanded occurrence, every requested byte and the explicitly specified access contract.

APPENDIX A

Reproduction and command reference

Unpack the companion, enter its `lean` directory, and use the pinned Lean 4.24.0 toolchain with `lake build`. The included Linux x86-64 executable can also be used directly. Linux installations need coreutils, util-linux, and the chosen codec binaries. The Python tools use the standard library and zstd for zstd blocks. The document check also uses Poppler's `pdfdetach`.

```
bin/strata pack collection.stc --iso-cdc -- \
  Core-16.1.iso Core-16.2.iso TinyCore-16.1.iso
bin/strata list collection.stc iso
bin/strata find collection.stc tiny iso
bin/strata range collection.stc 1 32768 2048 pvd.bin
bin/strata extract collection.stc 1 recovered.iso
cmp Core-16.2.iso recovered.iso
python3 tools/strata_view.py collection.stc 1 --query isolinux
python3 experiments/evaluate.py bin/strata iso-inputs measured.json
```

The evaluation script's `CORPUS` constant lists the source URLs and SHA-256 digests. The experiment downloads no data implicitly and runs no commands against physical media. The `-nested-only` option runs the generated archive and PDF checks without the public ISOs. The boot fixture script takes explicit QEMU and xorriso paths and can also take OVMF, clang, LLD, dosfstools and mtools paths for the UEFI checks.

```
python3 experiments/boot_check.py bin/strata measured-boot.json
```

An external encoder pair can be registered with `-codec ID ENCODER DECODER`. The programs take bytes on standard input and return bytes on standard output. Supply the same codec identifier with a compatible decoder when extracting. The catalog can be inspected before installing a custom decoder.

APPENDIX B

A complete small wire trace

Consider two occurrences called `a.iso`, both with the same three-byte data $x = (0, 1, 255)$. In raw mode each ISO starts a new logical block. The first is stored and the second aliases it after exact equality. Physical block 0 starts at byte offset 4, has decoded and stored lengths 3, and stores the three bytes unchanged. Its digest field contains the lowercase SHA-256 of those three decoded bytes. Let h be that 64-character ASCII field.

The ASCII index fields, before zero terminators, are

counts	1, 2
block 0	`raw`, 3, 3, h
entry 0	`iso`, `a.iso`, 3, 1
extent 0	0, 0, 3
entry 1	`iso`, `a.iso`, 3, 1
extent 0	0, 0, 3.

The fields take 117 bytes including terminators. Their start is byte 7 and the footer begins at byte 124. With the 24-byte footer, the archive is 148 bytes. Each occurrence is visible at its own ordinal, although all data extents refer to the same physical block.

A range request for occurrence 1, offset 1, length 2 maps to decoded block interval $[1, 3)$ and returns $(1, 255)$. When the same bytes are instead treated as a field of the generic framer, their encoding is $(1, 0, 1, 1, 255, 0)$. The payload region of this raw archive does not apply field escaping. Only the index has the field framing.

APPENDIX C

Reader conformance checklist

1. Read size and verify the four leading and four trailing magic bytes. Parse the fixed-width index length with checked arithmetic. Confirm the index range is within the archive.
2. Decode fields strictly. Require UTF-8 and canonical decimal representations where prescribed. Verify counts and consume every field exactly once.
3. Verify physical offsets and the payload endpoint. Check codec identifier syntax and decoded lengths before allocating output. Match each codec to an explicitly supported implementation.
4. Validate every extent's physical block, offset and length. Confirm every file's total size matches the ordered extents. Keep duplicate labels and repeated blocks as separate notions.
5. For each read, select only intersecting extents. Verify the length and digest of each decoded block before using it. Return the exact ordered intersection bytes or an error.
6. Treat names as catalog data. If materializing files, enforce explicit path, overwrite and staging rules, and preserve original labels in metadata.
7. When exposing a nested view, pin the parser profile, keep the original occurrence accessible, and propagate a resource policy.
8. When exposing a device, bind the selected ordinal, capacity, sector geometry and mode. Test the resulting boot path on the firmware and transport claimed.

The conformance rules state a reader's byte behavior. A writer can experiment with ordering, category layout and the set of independent compressors while meeting those same rules. An extension that requires additional decoder inputs, such as a shared delta base, needs a dependency-aware specification rather than an undocumented change to an existing codec ID.

References

[1] J. Kock. Polynomial functors and trees. *International Mathematics Research Notices*, 2011(3), 609–673. arXiv:0807.2874.

[2] N. Gambino and J. Kock. Polynomial functors and polynomial monads. *Mathematical Proceedings of the Cambridge Philosophical Society*, 154(1), 153–192, 2013. arXiv:0906.4931.

[3] C. Rezk. A model for the homotopy theory of homotopy theory. *Transactions of the American Mathematical Society*, 353(3), 973–1007, 2001. arXiv:math/9811037.

[4] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies 170, Princeton University Press, 2009. Author's manuscript.

[5] J. D. Stasheff. Homotopy associativity of H-spaces. I. *Transactions of the American Mathematical Society*, 108(2), 275–292, 1963. Bibliographic record.

[6] I. Moerdijk and I. Weiss. Dendroidal sets. *Algebraic & Geometric Topology*, 7, 1441–1470, 2007. arXiv:math/0701293.

[7] I. Gálvez-Carrillo, J. Kock and A. Tonks. Decomposition spaces, incidence algebras and Möbius inversion I: Basic theory. *Advances in Mathematics*, 331, 952–1015, 2018. arXiv:1512.07573.

[8] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic*. Springer. Publisher record.

[9] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, 85–103, 1972. Publisher record.

[10] L. de Moura and S. Ullrich. The Lean 4 theorem prover and programming language. *CADE 28*, 625–635, 2021. Publisher record.

[11] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 379–423 and 623–656, 1948. Full reprint.

[12] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9), 1098–1101, 1952. Original article, archived by IIT Delhi.

Compression, deduplication and integrity

[13] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3), 337–343, 1977. IEEE Information Theory Society.

[14] A. Tridgell and P. Mackerras. The rsync algorithm. Australian National University, TR-CS-96-05, 1996. Institutional repository.

[15] A. Muthitacharoen, B. Chen and D. Mazières. A low-bandwidth network file system. *SOSP*, 174–187, 2001. Authors' paper.

[16] W. Xia et al. FastCDC: A fast and efficient content-defined chunking approach for data deduplication. *USENIX ATC*, 101–114, 2016. USENIX paper.

[17] P. Deutsch. DEFLATE compressed data format specification version 1.3. RFC 1951, 1996. RFC Editor.

[18] P. Deutsch. GZIP file format specification version 4.3. RFC 1952, 1996. RFC Editor.

[19] L. Collin and I. Pavlov. The .xz file format. Format specification.

[20] Y. Collet and M. Kucherawy. Zstandard compression and the `application/zstd` media type. RFC 8878, 2021. RFC Editor.

[21] Zstandard contributors. Zstandard seekable compression format. Format specification.

[22] I. Pavlov. 7z format. 7-Zip format overview.

[23] D. G. Korn, J. P. MacDonald, J. C. Mogul and K.-P. Vo. The VCDIFF generic differencing and compression data format. RFC 3284, 2002. RFC Editor.

[24] Git contributors. Git pack format. Format documentation.

[25] C. Schneider et al. precomp-cpp. Project and implementation.

[26] preflate contributors. preflate: lossless recompression of DEFLATE streams. Project and implementation.

[27] National Institute of Standards and Technology. *Secure Hash Standard*. FIPS 180-4, 2015. NIST publication.

[28] D. Harnik, B. Pinkas and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security & Privacy*, 8(6), 40–47, 2010. IBM Research record.

[29] B. Laurie et al. Certificate Transparency version 2.0. RFC 9162, 2021. RFC Editor.

[30] D. McGrew. An interface and algorithms for authenticated encryption. RFC 5116, 2008. RFC Editor.

Media, filesystem and preservation specifications

[31] Ecma International. *ECMA-119: Volume and File Structure of CDROM for Information Interchange*. Standard page.

[32] Phoenix Technologies and IBM. *El Torito Bootable CD-ROM Format Specification*, version 1.0, 1995. Specification archived by MIT.

[33] UEFI Forum. *UEFI Specification 2.11*, Chapter 13, Media Access. Specification.

[34] T. Schmitt and libburnia contributors. *xorriso* manual. Manual archived by Debian.

[35] GNU GRUB contributors. The `loopback` command. GNU GRUB manual.

[36] GNU GRUB contributors. Loopback booting. GNU GRUB manual.

[37] Ventoy project. Ventoy image boot manager. Project documentation.

[38] QEMU contributors. System emulator invocation. QEMU documentation.

[39] Network Block Device contributors. Network Block Device protocol. Protocol specification.

[40] Linux kernel contributors. Squashfs 4.0 filesystem. Kernel documentation.

[41] Linux kernel contributors. EROFS: Enhanced Read-Only File System. Kernel documentation.

[42] International Organization for Standardization. *ISO 32000-2:2020, Document management: Portable document format, Part 2: PDF 2.0*. PDF Association standard page.

[43] PDF Association. PDF 2.0 Application Note 002: Associated files. Application note.

[44] Tiny Core Linux. x86 release archive for 16.1. Published images and checksums.

[45] Tiny Core Linux. x86 16.x releases. Princeton mirror.

[46] J. Kunze et al. The BagIt file packaging format (V1.0). RFC 8493, 2018. RFC Editor.

[47] Oxford Common File Layout editors. *OCFL 1.1 Specification*. Version 1.1.