Modal Verification Patterns for Software Systems

Chapter 1:
A Tale of Two Concepts: Resource and its Context

Ismail Kuru and Colin S. Gordon
Department of Computer Science
Drexel University

October 13, 2025

Part I

Systems View

The Essentials in Systems Programming

$$\underbrace{\text{pointer va}}_{\text{a virtual reference}} \quad := \quad \overbrace{\text{malloc (size)}}^{\text{a supposedly allocated physical resource}}$$

Memory Location Virtualization

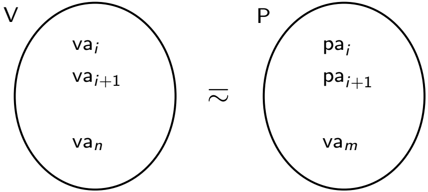


Figure: Virtualization: The Deception of Memory Abundance

Memory Location Virtualization: Abstraction

An Address Space with Logical Name γ

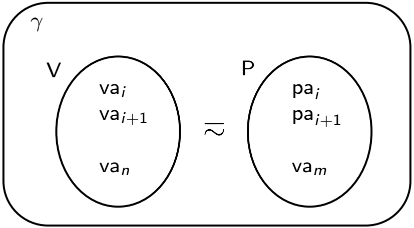


Figure: Address-Spaces: Named Containers for Virtual Memory Mappings

A Program Named γ_n

```
pointer va :=
malloc(size)
```

A Program Named γ_m

```
pointer va :=
malloc(size)
```

- A program is abstracted as a *named address-space*
- A container of *virtual-to-physical* memory resource mappings

Address-Space Container of Virtual-to-Physical Mappings

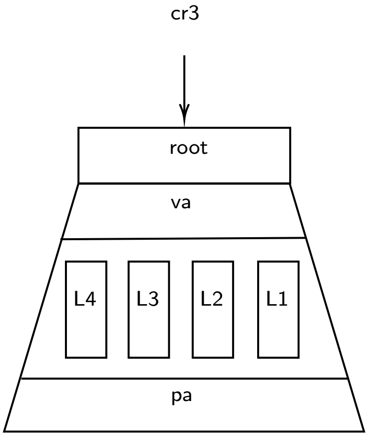


Figure: Depicting an Address-Space with its Essential Aspects

The Current View of Memory

The register cr_3 points to the current view of the memory, i.e., the loaded address space in the memory

.

The Essentials in Systems Programming

$$\underbrace{\text{FILE}* \ \text{fptr}}_{\text{a file handle}} := \overbrace{\text{fopen(filename, mode);}}^{\left\{\begin{array}{c}\text{a supposedly accesible data at somewhere in the computer} \\ \text{which makes its potential mode unknown : in\_memory or on disk or ...}\end{array}\right\}}$$

File Page Virtualization

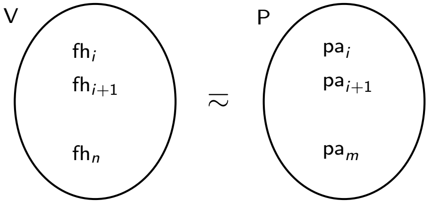
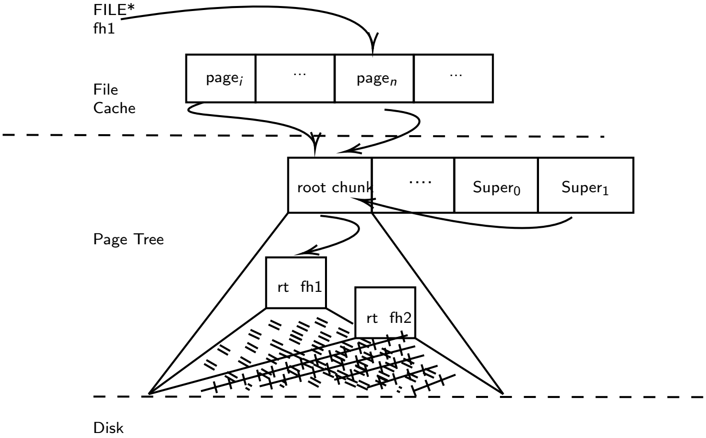


Figure: Virtualization: The Deception of Disk-Page Abundance

A Global Disk-Page Tree



File Data Virtualization: Wait! Maybe a Bit More!

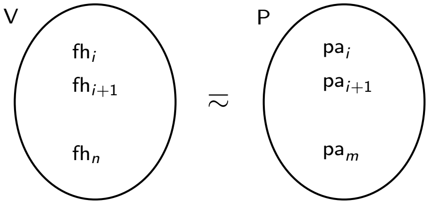


Figure: Virtualization: The Deception of Disk-Page Abundance Parameterized *under* **Some Consistency Model**

File Data Virtualization: Abstraction

A Disk-Page Tree with Logical Name γ_n?

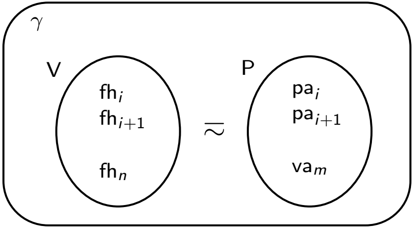


Figure: A Global Disk-Page Tree: Named Containers for File-Page to Disk-Page Mappings?

An Updated File in the Named Disk-Tree γ_n

```
FILE* fh1 :=
 write(data)
```

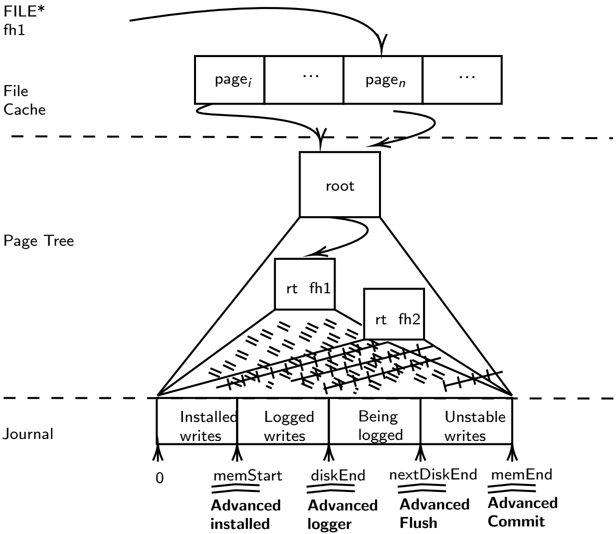
An Read-Only Access to a File in the Named Disk-Tree γ_n

```
FILE fh2 :=
 read(data,sz)
```

- Could a global disk tree as a container work for virtual-to-physical disk resources?
- Maybe? But not always!

A Global File Page Tree with Multiple Views

Consistency models can impose multi-mode-views on the disk page tree



An Example for a Consistency Model: Journalling

- Indices uniquely naming the consistent pieces of disk and the updates to be inserted into the disk
- Certain pages of the global tree are valid under different *views* to it
- Recovery, Atomicity

...

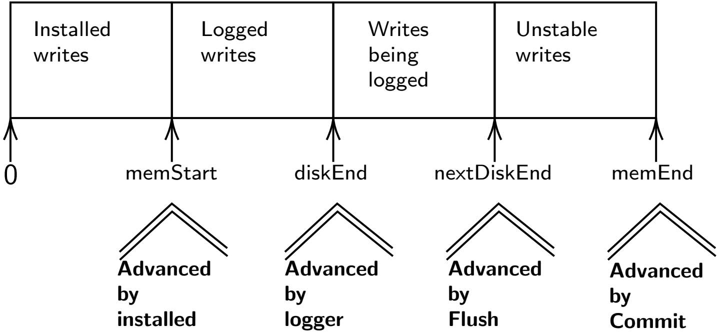


Figure: Depicting Journalling Model

Another Example for a Consistency Model: Copy-On-Writes Filesystems

- Updates are done on newly allocated resources
- Snapshots are collections of updates
- A uniquely identifying snapshotting identifier naming the consistent pieces of disk.
- Snapshot updates appear on the disk atomically: always have a consistent *view* of the disk-tree
- Recovery, Atomicity ....

Part II

Logical View

Resource: Unital Element as a Fact Matters Most

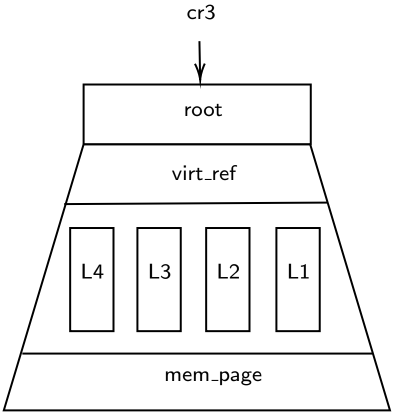
$\{P\}\ C\ \{Q\}$

- What matters most inside P for for the program action C *contingently*?
- Well-known points-to assertion, e.g., $\mathrm{virt\_ref} \mapsto \mathrm{mem\_page}$

A Virtual Memory Pointsto

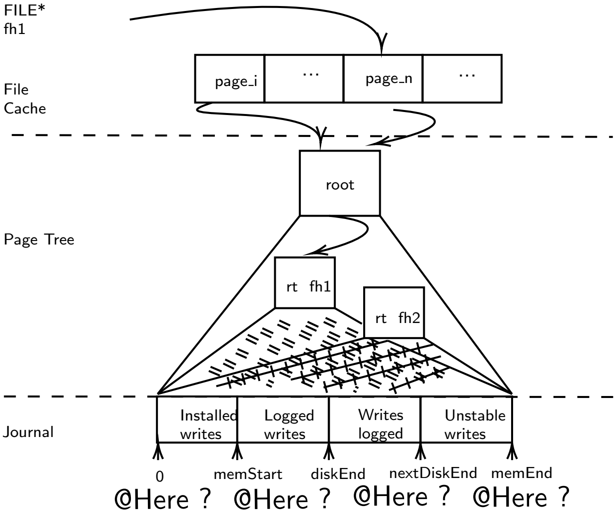
- virt_ref $\mapsto$ mem_page

@Here ?



A Disk Page Pointsto

- An expected points-to assertion, e.g., page_ref $\mapsto_q$ page

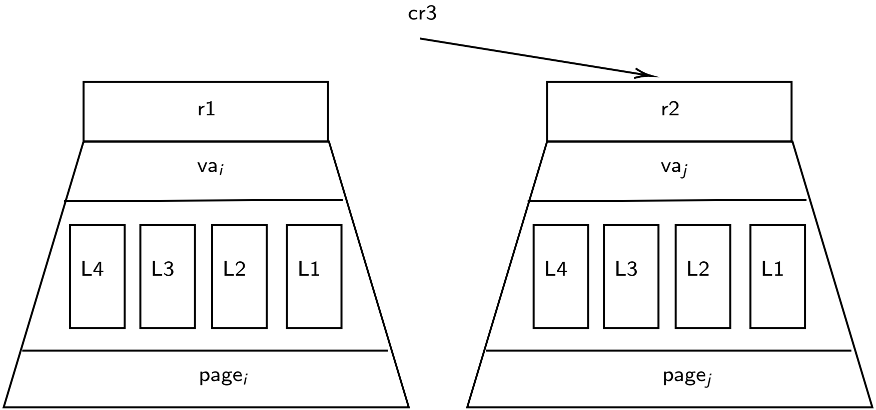


@Here ?: Resource Context

- The *habitat* of a resource determining its scope of validity

Habitat of Virtual Memory Mappings

$$\{[r1](va_i \mapsto page_i) * va_j \mapsto page_j\}cr3 := r1\{va_i \mapsto page_i * [r2](va_j \mapsto page_j)\}$$



Habitats for Disk Resources in Journaling

- A specification of *recovery* would require both
  - Explicitly naming on the resources that can be inferred from their uniquely identifying resource context name
  - Losing duality of resource contexts in specifications

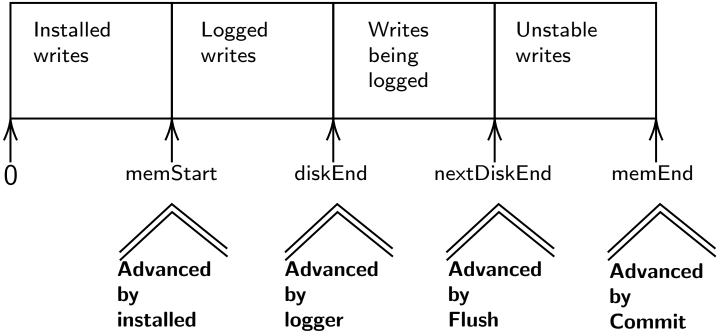


Figure: Depicting Journalling Model

Modal Decomposition of Program-Logics

Modality	Context	Elements	Nominalization	Context Steps
Post-Crash+	$\Diamond\ P$	$\ell \mapsto_n^{\overline{\gamma}} v$	Strong	Crash Recovery
NextGen!	$\overset{t}{\hookrightarrow}\ P$	Own (t(a))	Strong	Determined Based on the Model*
StackRegion*	$\overset{ICut^n}{\hookrightarrow}\ P$	$\boxed{\text{n}}\ \ell \mapsto v$	Strong	Alloc and Return to/from stack
Actor#	$@_\iota$ P	Variable values	Weak	Send Message
Memory-Fence^x	$\triangle_\pi$ and $\triangledown_\pi$	$\ell \mapsto v$	Weak	Fence Acquire and Release
Address Space?	[r]P	$\ell \mapsto v$	Weak	Address Space Switch
Ref-Count&	$@_\ell$ P	$\ell_1 \mapsto v$	Weak	Allocating, Dropping and Sharing a Reference

$*$*The StackRegion Modality is an instance of NextGen (called the Independence Modality in* [Vindum et al.(2025)]).
+[Chajed(2022), Chajed et al.(2019), Tej Chajed and contributors(2023)]
! [Vindum et al.(2025)]
[Gordon(2019)]
? [Kuru and Gordon(2024), Kuru and Gordon(2025)]
& [Wagner et al.(2024)]
x [Doko and Vafeiadis(2016), Doko and Vafeiadis(2017), Dang et al.(2019)]

Remarks

- This paper:
  - First steps in identifying key pieces in building a program logic for real systems
  - Nominalization as "*naming resource contexts and its resources*" is in the paper
- The verification pattern concepts are not specific to separation logic!
  - Actor modelling in Dafny [Gordon(2019)]
- This is an introductory chapter
  - The next chapter is on the interaction between resource contexts.

End

- I am on the postdoc job market: https://ismailkuru.github.io/
- Happy to take questions now!

Tej Chajed. 2022.
Verifying a concurrent, crash-safe file system with sequential reasoning.
Ph.D. Dissertation. Machetutes Institute of Technology, Cambridge, MA.
Available at https://dspace.mit.edu/handle/1721.1/144578.

Tej Chajed, Joseph Tassarotti, M. Frans Kaashoek, and Nickolai Zeldovich. 2019.
Verifying concurrent, crash-safe systems with Perennial. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (Huntsville, Ontario, Canada) *(SOSP '19)*. Association for Computing Machinery, New York, NY, USA, 243–258.
doi:10.1145/3341301.3359632

Hoang-Hai Dang, Jacques-Henri Jourdan, Jan-Oliver Kaiser, and Derek Dreyer. 2019.
RustBelt meets relaxed memory.
Proc. ACM Program. Lang. 4, POPL, Article 34 (Dec. 2019), 29 pages.
doi:10.1145/3371102

Marko Doko and Viktor Vafeiadis. 2016.
A Program Logic for C11 Memory Fences. In *Proceedings of the 17th International Conference on Verification, Model Checking, and Abstract Interpretation - Volume 9583* (St. Petersburg, FL, USA) *(VMCAI 2016)*. Springer-Verlag, Berlin, Heidelberg, 413–430.
doi:10.1007/978-3-662-49122-5_20

Marko Doko and Viktor Vafeiadis. 2017.
Tackling Real-Life Relaxed Concurrency with FSL++. In *Programming Languages and Systems: 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings* (Uppsala, Sweden). Springer-Verlag, Berlin, Heidelberg, 448–475.

doi:10.1007/978-3-662-54434-1_17

Colin S Gordon. 2019.
Modal assertions for actor correctness. In *Proceedings of the 9th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control*. 11–20.

Ismail Kuru and Colin S. Gordon. 2024.
Modal Abstractions for Virtualizing Memory Addresses.
arXiv:2307.14471 [cs.PL] https://arxiv.org/abs/2307.14471

Ismail Kuru and Colin S. Gordon. 2025.
Modal Verification Patterns for Systems Software. In *Proceedings of the 13th Workshop on Programming Languages and Operating Systems* (Seoul, Republic of Korea) *(PLOS '25)*. Association for Computing Machinery, New York, NY, USA, 25–33.
doi:10.1145/3764860.3768337

Josep Tassarotti Tej Chajed and contributors. 2023.
Post-crash modality in Perennial's Coq Mechanization.
https://github.com/mit-pdos/perennial/blob/master/src/goose_lang/crash_modality.v

Simon Friis Vindum, Aïna Linn Georges, and Lars Birkedal. 2025.
The Nextgen Modality: A Modality for Non-Frame-Preserving Updates in Separation Logic. In *Proceedings of the 14th ACM SIGPLAN International Conference on Certified Programs and Proofs* (Denver, CO, USA) *(CPP '25)*. Association for Computing Machinery, New York, NY, USA, 83–97.
doi:10.1145/3703595.3705876

Andrew Wagner, Zachary Eisbach, and Amal Ahmed. 2024.
Realistic Realizability: Specifying ABIs You Can Count On.

Proc. ACM Program. Lang. 8, OOPSLA2, Article 315 (Oct. 2024), 30 pages.
doi:10.1145/3689755