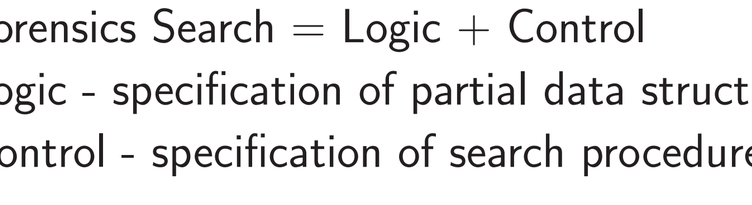
Compositional Declarative Forensics

Corin Pitcher, Radha Jagadeesan and James Riely
FPL, DePaul University, Chicago, IL

DePaul University

Forensics

- Forensics investigation model (Sremack, 2004)



- Address challenges in analysis phase programmatically

Challenges for Forensics Programmatic Analysis

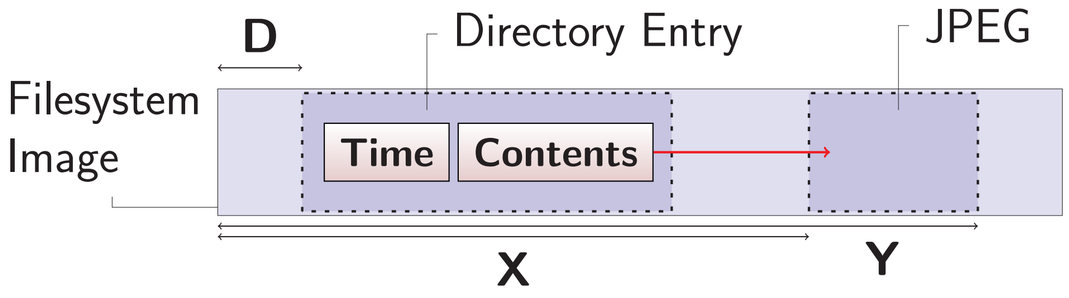
- Many new systems require new tools
- Difficulty in repurposing existing tools
- Automated composition of tools
- Codifying forensic search procedures as programs

Approach: Compositional Declarative Forensics

- Forensics Search = Logic + Control
- Logic - specification of partial data structures
- Control - specification of search procedures

Example: JPEGs Created in Time Range?

- Search filesystem image for JPEG files with creation timestamps between **START** and **END**



- Formalization as logical query

$$\begin{aligned} & \mathbf{DirectoryEntry(image, D)} \\ \wedge\ & \mathbf{TimeStamp(image, D, Time)} \\ \wedge\ & \mathbf{START} \leq \mathbf{Time} \leq \mathbf{END} \\ \wedge\ & \mathbf{Contents(image, D, image[X..Y])} \\ \wedge\ & \mathbf{ValidJPEG(image, image[X..Y])} \end{aligned}$$

Example: Specification of Filesystem Structures

- **LE(N, A, O)** calculates an integer from **N** bytes in little-endian format from array **A** at offset **O**

$$\mathbf{LE(N, A, O)} \triangleq \mathbf{\Sigma_{0 \leq i < N} A[O + i] * 256^i}$$

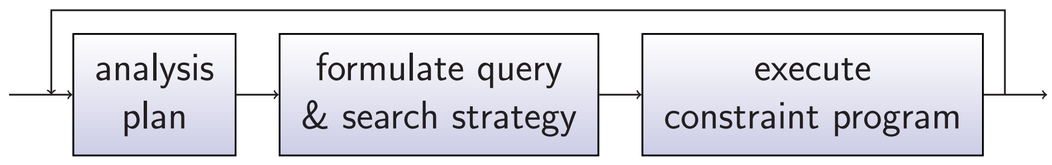
- **TimeStamp** for an NTFS MFT entry at offset **D**

$$\begin{aligned} & \mathbf{TimeStamp(image, D, Time)} \triangleq \\ & \quad \exists \mathbf{i, j}.\ \mathbf{0 \leq i < 1024} \wedge \\ & \qquad \mathbf{0 \leq j < 1024 - i - 7} \wedge \\ & \qquad \mathbf{LE(4, image, D + i) = 16} \wedge \\ & \qquad \mathbf{LE(2, image, D + i + 20) = j} \wedge \\ & \qquad \mathbf{LE(8, image, D + i + j) = Time} \end{aligned}$$

- At offset **i** in the MFT entry, 16 represents MFT entry attribute with type `$STANDARD_INFORMATION`
- Value **Time** is found by chasing pointers

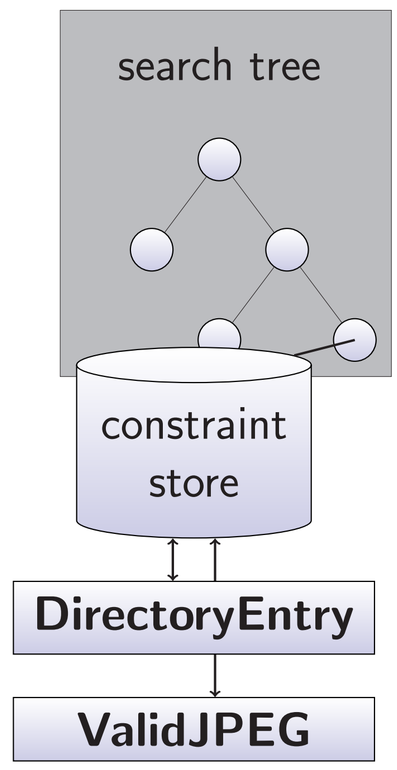
Implementation: Constraint Programming

- Execute forensics queries as constraint programs



Constraint Progamming

- Each search tree node is a constraint store
- Successful leaf nodes are solutions to the query
- Logical predicates and existing forensics tools encoded as constraints
- Symmetric interaction between constraints via constraint store
- Partial information in store avoids unnecessary search



Taming Memory Requirements for Forensics

- Must control memory usage and size of search trees
  - variable domains are large
  - $\mathbf{X \in [0, 2^{42})}$ for an offset in a 4TB volume image
  - large images cannot be held in memory
- Custom constraints for accessing and scanning images
- Careful use of branching
- Efficient Gecode-based implementation

Experimental Results

- Task - Find MFT entries pointing to JPEG files
- Data source - 3 related 4.5GB NTFS disk images
  - `original.img` - NIST CFReDS hacking case
  - `deleted.img` - large directory deleted from `original.img`
  - `formatted.img` - quick-format applied to `original.img`
- Restrictions
  - `all` - search for JPEG files anywhere
  - `allocated` - search for JPEG files in allocated clusters
  - `unallocated` - search for JPEG files in unallocated clusters
- Execution time, # of solutions, search tree size

	all	**allocated**	**unallocated**
`original.img`	27s 272 solns 48,271 nodes	27s 272 solns 48,271 nodes	25s 0 solns 42,971 nodes
`deleted.img`	25s 121 solns 47,833 nodes	25s s 121 solns 47,833 nodes	25s 0 solns 45,515 nodes
`formatted.img`	28s 272 solns 46,993 nodes	1s 0 solns 0 nodes	27s 272 solns 46,993 nodes

Gecode 3.7.3; Linux kernel v3.2.0-29; Xeon E3-1230 3.30 GHz; 16GB RAM

Mail: {cpitcher,rjagadeesan,jriely}@cs.depaul.edu WWW: http://fpl.cs.depaul.edu/