Open Universiteit

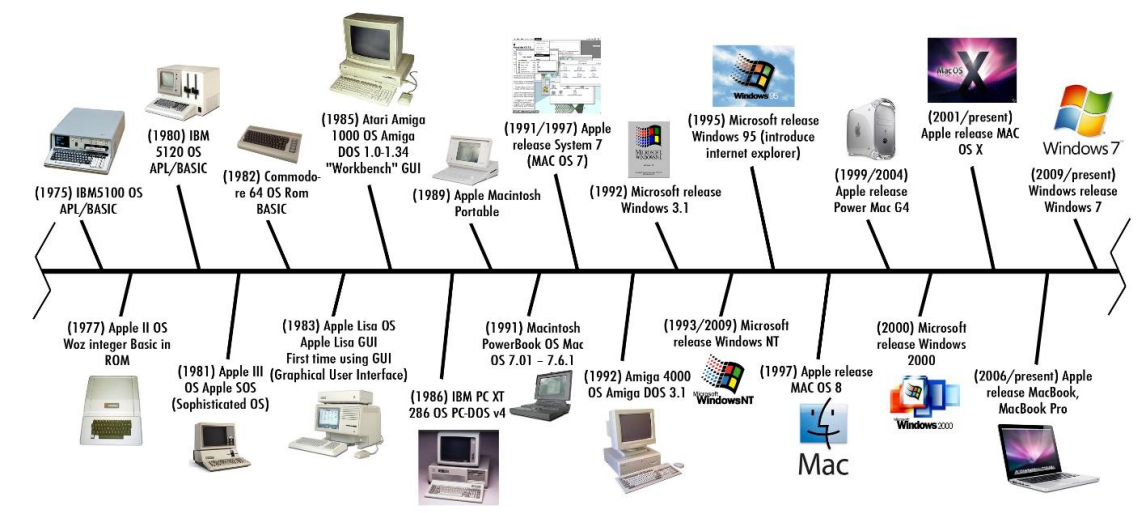
Deducing File History from NTFS Timestamps

Vincent van der Meer

Open University Research Seminar Informatica

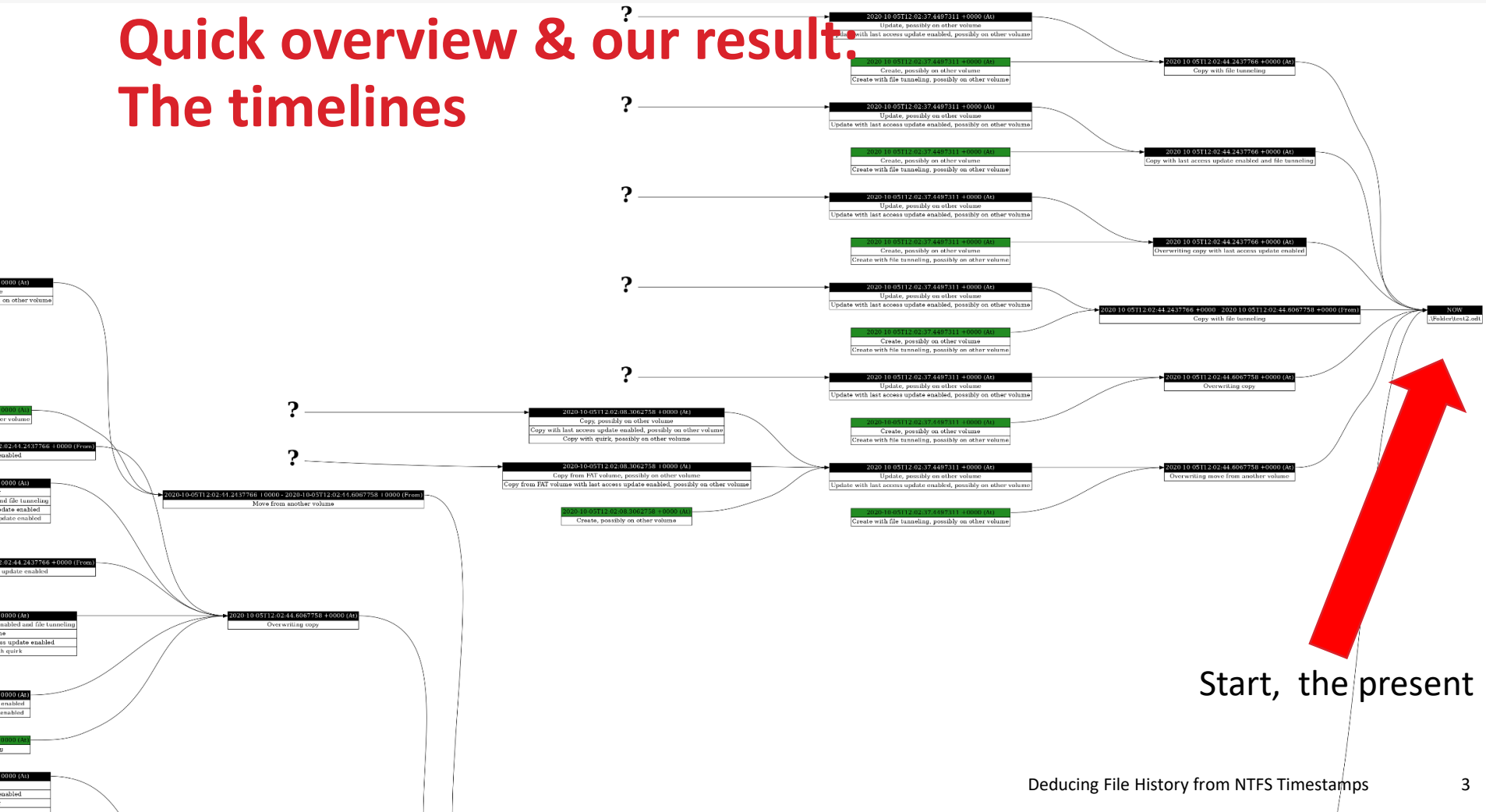
March 23, 2021

Quick overview:
We want a timeline



Like that, but instead of a computer file

Quick overview & our result.
The timelines



Start, the present

Authors

Authors

Jelle Bouma	-	Student at the OU
		This study is based upon his BSc thesis.
Hugo Jonker	-	Daily supervisor
Vincent van der Meer	-	PhD candidate

Unraveling a computer file's history

Where do you come from?
How did you get here?
How long have you been here?

Unraveling a computer file's history

Where do you come from?
How did you get here?
How long have you been here?

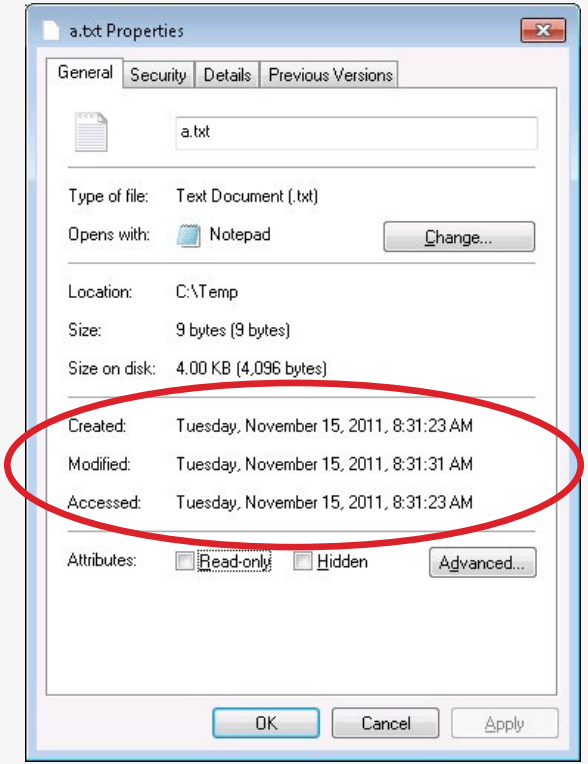
I don't have any answers…
But I do have some clues!

1) System related logfiles
2) The file's timestamps

Background: Timestamps

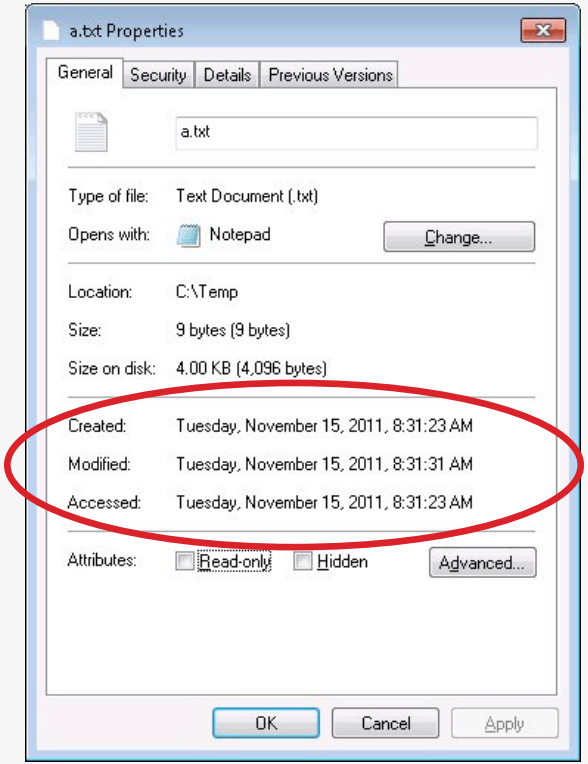
> "A **timestamp** is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second."
>
> - wikipedia.org/wiki/Timestamp

Background: 3, 4 or more timestamps?



Created:	File creation
Modified:	Last time file data was changed
Accessed:	Last time file data was accessed

Background: 3, 4 or more timestamps?



Created:	File creation
Modified:	Last time file data was changed
Accessed:	Last time file data was accessed
Changed:	Last time the file's MFT-entry was changed

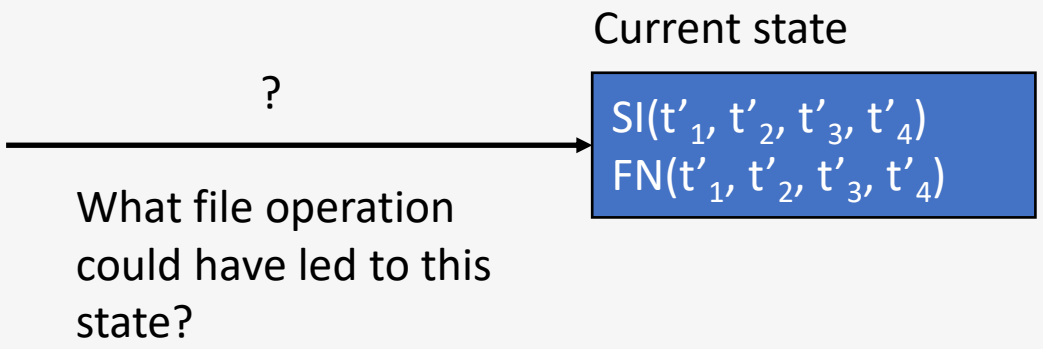
Background: Timestamp

- NTFS stores 8 timestamps per file

- **Notation**: two tuples of four timestamps.
  - SI(t_1, t_2, t_3, t_4) and FN(t_1, t_2, t_3, t_4)
  - SI = Standard Information, FN = File Name

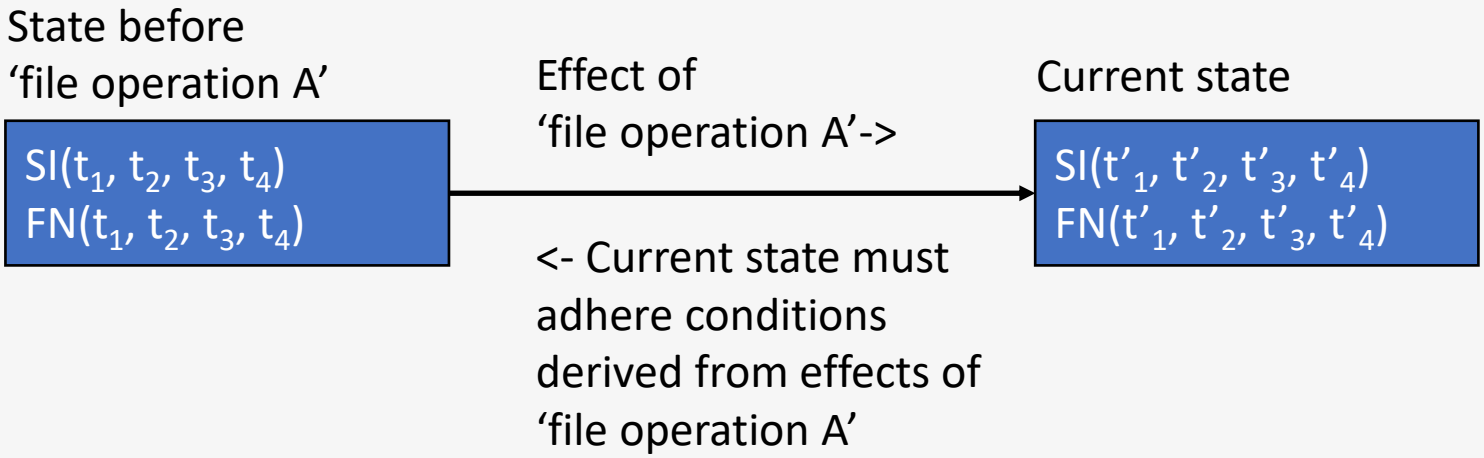
- Each NTFS timestamp has a 100 ns. accuracy

Backward reasoning with file operations

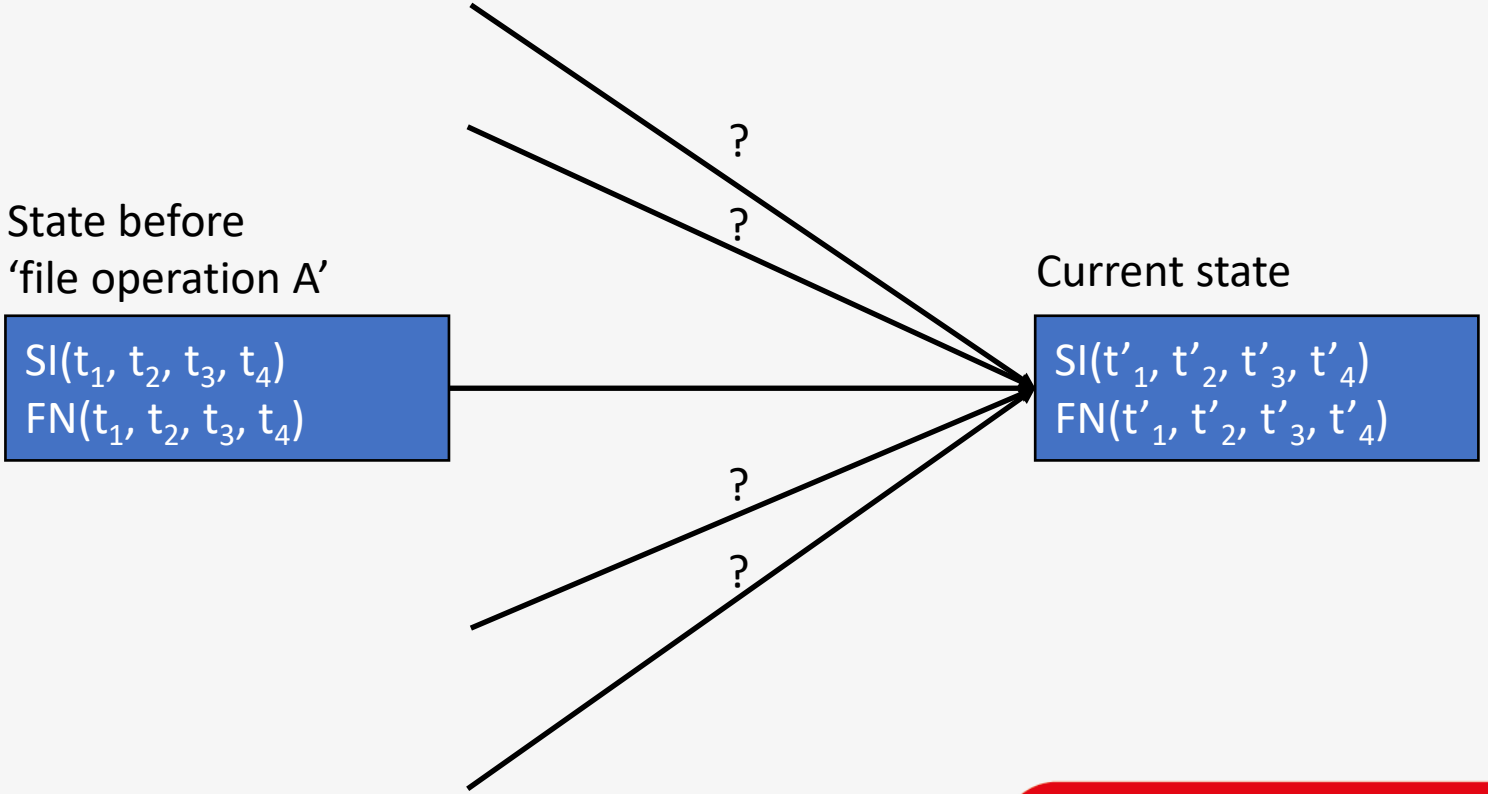


Backward reasoning with file operations

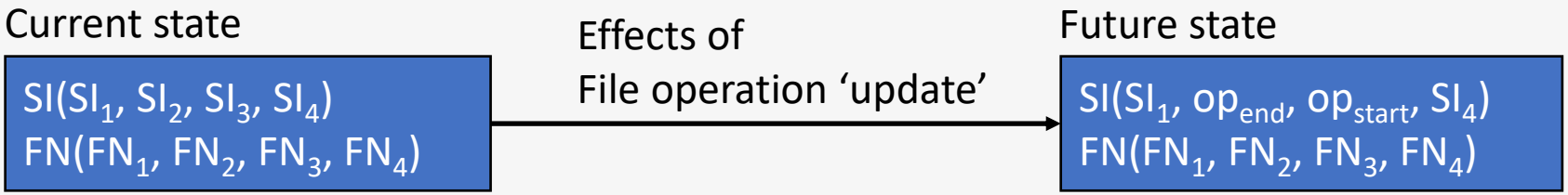
Backward reasoning with file operations



Backward reasoning with file operations



Example: file operation ‘update’



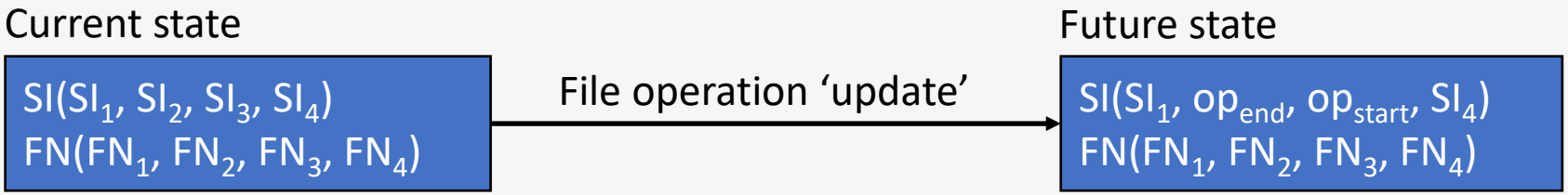
Description of the file operations

Operation	$SI' =$				$FN' =$			
Create	$(op_{start},$	$op_{start},$	$op_{start},$	$op_{start})$	$(op_{start},$	$op_{start},$	$op_{start},$	$op_{start})$
Copy	$(op_{start},$	$src.SI_2,$	$op_{end},$	$op_{start})$	$(op_{start},$	$op_{start},$	$op_{start},$	$op_{start})$
Update	$(SI_1,$	$op_{end},$	$op_{start},$	$SI_4)$	$(FN_1,$	$FN_2,$	$FN_3,$	$FN_4)$
Move within volume	$(SI_1,$	$SI_2,$	$op_{end},$	$SI_4)$	$(SI_1,$	$SI_2,$	$SI_3,$	$SI_4)$
Move from another volume	$(src.SI_1,$	$src.SI_2,$	$op_{end},$	$op_{start})$	$(op_{start},$	$op_{start},$	$op_{start},$	$op_{start})$
Overwriting copy	$(SI_1,$	$src.SI_2,$	$op_{start},$	$SI_4)$	$(FN_1,$	$FN_2,$	$FN_3,$	$FN_4)$
Overwriting move from another NTFS volume	$(src.SI_1,$	$src.SI_2,$	$op_{start},$	$SI_4)$	$(FN_1,$	$FN_2,$	$FN_3,$	$FN_4)$
Rename	$(SI_1,$	$SI_2,$	$op_{start},$	$SI_4)$	$(SI_1,$	$SI_2,$	$SI_3,$	$SI_4)$
Attribute change	$(SI_1,$	$SI_2,$	$op_{start},$	$SI_4)$	$(FN_1,$	$FN_2,$	$FN_3,$	$FN_4)$
Delete	$(SI_1,$	$SI_2,$	$SI_3,$	$SI_4)$	$(FN_1,$	$FN_2,$	$FN_3,$	$FN_4)$
Access	$(SI_1,$	$SI_2,$	$SI_3,$	$SI_4)$	$(FN_1,$	$FN_2,$	$FN_3,$	$FN_4)$
Extract zip file	$(\texttt{valA},$	$\texttt{valA},$	$op_{end},$	$\texttt{valA})$	$(op_{start},$	$op_{start},$	$op_{start},$	$op_{start})$

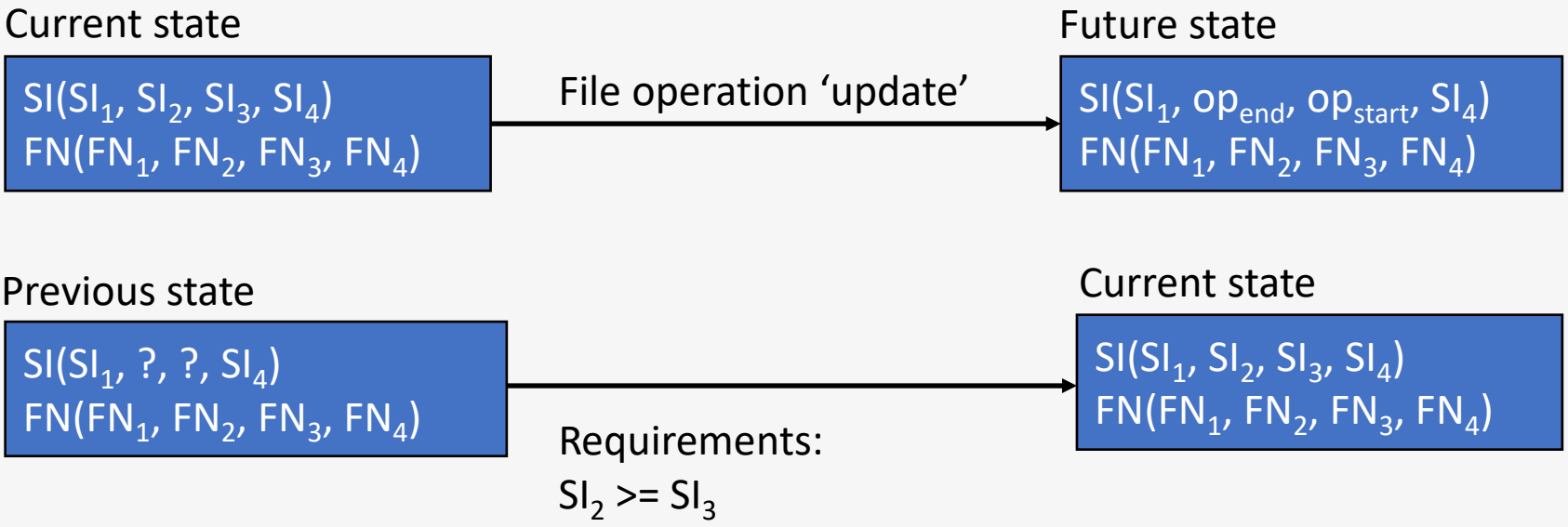
`valA`: $src.SI_2$ rounded up to even seconds plus time zone difference (tzd), i.e.,

$$\texttt{valA} = 2 \cdot 10^7 \cdot \left\lceil \frac{src.SI_2}{2 \cdot 10^7} \right\rceil + tzd.$$

Example: file operation ‘update’



Example: file operation ‘update’



Loss of information & requirements

With backwards reasoning, loss of information occurs.

- However, sometimes bits information can be restored due to the effects of file operations.

Requirements

- Each file operation that has an effect on timestamps, requirements can be described for matching the operation to a given tuple of timestamps.

Requirements can be:

- A minimum set of timestamps that must be known (i.e. that must have a value to be evaluated)
- The accuracy of specific timestamps (i.e.: 100 nanoseconds, seconds, or even seconds)
- Timestamps that must be equal, unequal, larger, of smaller than other timestamps within the tuple.

Description of the previous state of file operations

Operation	$SI^{t-1} =$				$FN^{t-1} =$			
Create	(?,	?,	?,	?)	(?,	?,	?,	?)
Copy (source)	(?,	SI_2,	?,	?)	(?,	?,	?,	?)
Update	(SI_1,	?,	?,	SI_4)	(FN_1,	FN_2,	FN_3,	FN_4)
Move within volume	(FN_1,	FN_2,	FN_3,	FN_4)	(?,	?,	?,	?)
Move from another volume (source)	(SI_1,	SI_2,	?,	?)	(?,	?,	?,	?)
Overwriting copy								
– target	(SI_1,	?,	?,	SI_4)	(FN_1,	FN_2,	FN_3,	FN_4)
– source	(?,	SI_2,	?,	?)	(?,	?,	?,	?)
Overwriting move from another NTFS volume								
– target	(?,	?,	?,	SI_4)	(FN_1,	FN_2,	FN_3,	FN_4)
– source	(SI_1,	SI_2,	?,	?)	(?,	?,	?,	?)
Rename	(FN_1,	FN_2,	FN_3,	FN_4)	(?,	?,	?,	?)
Attribute change	(SI_1,	SI_2,	?,	SI_4)	(FN_1,	FN_2,	FN_3,	FN_4)
Delete	(SI_1,	SI_2,	SI_3,	SI_4)	(FN_1,	FN_2,	FN_3,	FN_4)
Access	(SI_1,	SI_2,	SI_3,	SI_4)	(FN_1,	FN_2,	FN_3,	FN_4)
Extract zip file (source)	(?,	SI_2,	?,	?)	(?,	?,	?,	?)

Description of requirements & information loss

Operation	Required condition	Information known	
		SI-tuple	FN-tuple
Create	$SI_i = SI_{5-i}$ and $SI_i = FN_i$, for $1 \leq i \leq 4$	(F,F,F,F)	(F,F,F,F)
Copy	$SI_3 \geq SI_1$ and $SI_1 = SI_4$ and $FN_{1...4} = SI_1$	(F,-,F,F)	(F,F,F,F)
Update	$SI_2 \geq SI_3$.	(-,F,F,-)	(-,-,-,-)
Move within volume	$FN'_{1,2,4} = SI'_{1,2,4}$.	(T,T,T,T)	(F,F,F,F)
Move from another volume	$FN_{1...4} = SI_4$ and $SI_3 \geq SI_4$.	(-,-,F,F)	(F,F,F,F)
Overwriting copy	True.		
– target		(-,F,F,-)	(-,-,-,-)
– source		(F,-,F,F)	(F,F,F,F)
Overwriting move from other NTFS volume	True.		
– target		(F,F,F,-)	(-,-,-,-)
– source		(-,-,F,F)	(F,F,F,F)
Rename	$FN_{1,2,4} = SI_{1,2,4}$.	(T,T,T,T)	(F,F,F,F)
Attribute change	True.	(-,-,F,-)	(-,-,-,-)
Delete	True.	(-,-,-,-)	(-,-,-,-)
Access	True.	(-,-,-,-)	(-,-,-,-)
Extract zip file	$FN_i = FN_{5-i}$, $1 \leq i \leq 4$ and $SI_{1,2,4} = SI_{4,2,1}$ and $SI_1 = 2k \cdot 10^7 + tzd$, for some $k \in \mathbb{N}$, where $tzd = 0 \vee \lvert tzd \rvert > 10^{10}$.	(F,T,F,F)	(F,F,F,F)

Algorithm 1: Match Operation

Input: A tuple of timestamps, information status, file operation F to be matched

Output: True if the given set of timestamps and information status could have been the result of F, otherwise False.

Description: Match Operation tries to find an incompatible situation for the given timestamps, information status, and a file-operation. If no incompatible situation is found, then the operation is applicable.

Algorithm 1: Match Operation

To try to reject an operation, the following actions are performed:

- Is the operation applicable to the file? Reject otherwise.
- Is the precision of the resulting timestamps compatible with the precision constraints imposed by operation? Reject otherwise.
- Are pairs of timestamps compatible with specific constraints?
  - If REQ(TS1 = TS2) /\ TS1 ≠ TS2 → reject
  - If REQ(TS1 ≠ TS2) /\ TS1 = TS2 → reject
  - If REQ(TS1 > TS2) /\ TS1 ≤ TS2 → reject
  - If REQ(TS1 < TS2) /\ TS1 ≥ TS2 → reject

- If the operation could not be rejected, accept.

Algorithm 2: Create Sequence

Input: Entry

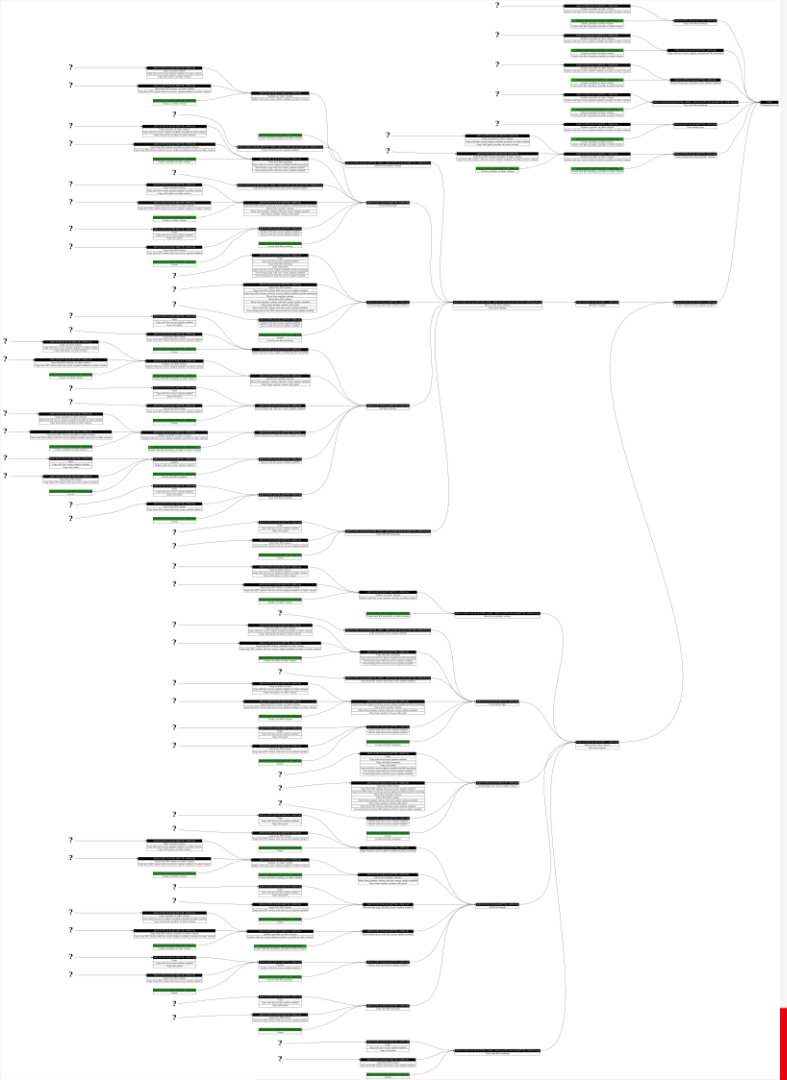
(Entry consists of a tuple of timestamps, information status, and position within a Sequence)

Output: Sequences (consecutive and compatible entries)

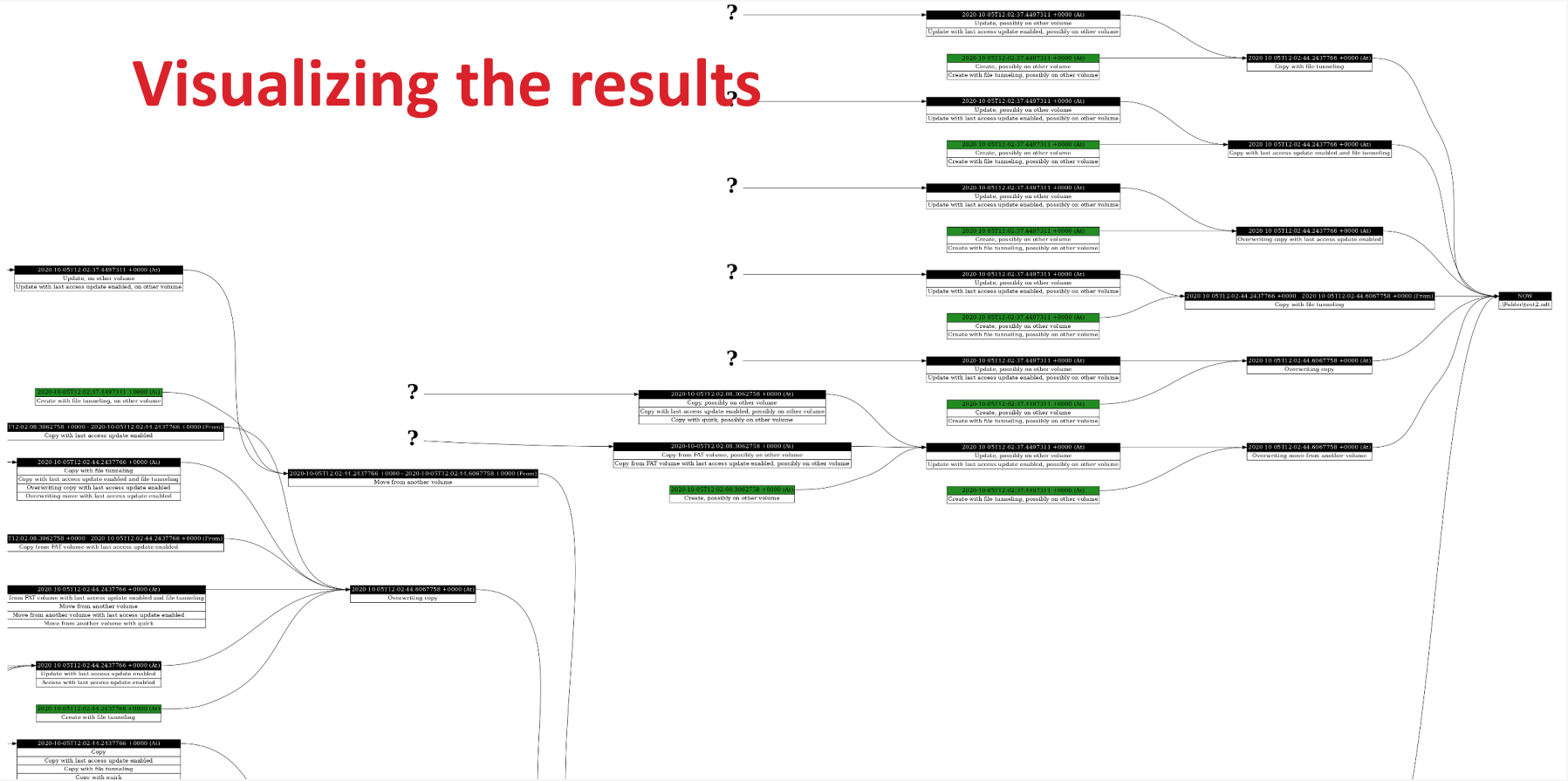
Description: Finds all possible sequences of a file, where each sequence consists of one or more subsequent matching file operations given the state of the file at that point in time.

A sequence ends (i.e. cannot go further back in time) because either the timestamps to evaluate are exhausted, or because of an operation that created the file in the observed file system.

Visualizing the results



Visualizing the results




?
2020-10-05T12:02:37.4497311 +0000 (At)
Update, possibly on other volume
Update with last access update enabled, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Create, possibly on other volume
Create with file tunneling, possibly on other volume
2020-10-05T12:02:44.2437766 +0000 (At)
Copy with file tunneling
?
2020-10-05T12:02:37.4497311 +0000 (At)
Update, possibly on other volume
Update with last access update enabled, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Create, possibly on other volume
Create with file tunneling, possibly on other volume
2020-10-05T12:02:44.2437766 +0000 (At)
Copy with last access update enabled and file tunneling
?
2020-10-05T12:02:37.4497311 +0000 (At)
Update, possibly on other volume
Update with last access update enabled, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Create, possibly on other volume
Create with file tunneling, possibly on other volume
2020-10-05T12:02:44.2437766 +0000 (At)
Overwriting copy with last access update enabled
?
2020-10-05T12:02:37.4497311 +0000 (At)
Update, possibly on other volume
Update with last access update enabled, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Create, possibly on other volume
Create with file tunneling, possibly on other volume
2020-10-05T12:02:44.2437766 +0000 - 2020-10-05T12:02:44.6067758 +0000 (From)
Copy with file tunneling
NOW
.\Folder\test2.odt
?
2020-10-05T12:02:37.4497311 +0000 (At)
Update, possibly on other volume
Update with last access update enabled, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Create, possibly on other volume
Create with file tunneling, possibly on other volume
2020-10-05T12:02:44.6067758 +0000 (At)
Overwriting copy
2020-10-05T12:02:08.3062758 +0000 (At)
Copy, possibly on other volume
th last access update enabled, possibly on other volume
Copy with quirk, possibly on other volume
2020-10-05T12:02:08.3062758 +0000 (At)
Copy from FAT volume, possibly on other volume
olume with last access update enabled, possibly on other volume
2020-10-05T12:02-08.3062758 +0000 (At)
Create, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Update, possibly on other volume
Update with last access update enabled, possibly on other volume
2020-10-05T12:02:37.4497311 +0000 (At)
Create, possibly on other volume
Create with file tunneling, possibly on other volume
2020-10-05T12:02:44.6067758 +0000 (At)
Overwriting move from another volume

Extension of the described results

Files originating from other file systems

Native Windows supported files systems:

- FAT32
- exFAT
- ReFS

Impacts file operations:

- Copy
- Move from another volume
- Overwriting copy
- Overwriging move from another volume

Other system-related factors of influence

File tunneling

A NTFS feature that automatically copies specific timestamps from a recently deleted file

Last Access updating

A system setting that controls under what conditions the SI_3 timestamp is updated or not.

Discussion

Limitations

- Sequences are made on the assumption that there is a minimum amount of known timestamps that must be known for a file operation to be matched.

- The deduced sequences of file operations are 100% complete given our descriptions of file operations effects and requirements.

- However, due to the loss of information, these sequences are not all possible sequences in reality:
  - Sequences could be longer
  - Sequences could have more branches

Thank you for your attention

Time for questions!

Contact information

Name: Vincent van der Meer

E-mail: vincent.vandermeer@zuyd.nl

Linkedin: https://linkedin.com/in/vvandermeer