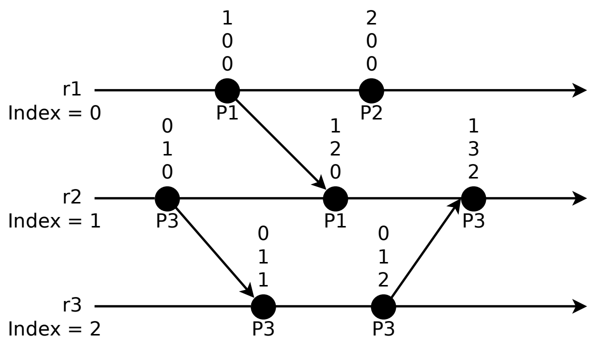

Figure 10: Each memory region has an assigned vector clock and an index for the local counter. Three processes, p_1, p_2, and p_3, interact with the memory regions. Whenever a process accesses a memory region the vector clock the process saw last and the local vector clock are combined and the counter at the index of the region is increased.

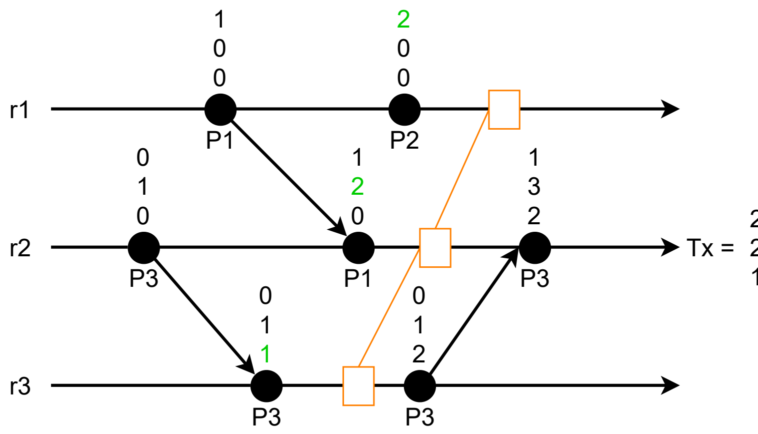

Figure 11: In a consistent snapshot the values of the global time at each region's index and that region's vector at the index are equal.

also been proposed (Palutke, Ruderich, Wild and Freiling, 2020).

5. Measuring Consistency

While theoretical quality criteria are an important step towards understanding which factors influence the usefulness of a memory snapshot, the question remains how these criteria can be measured. Because of the limitations of previous measurement approaches, we describe an alternative method to evaluate a snapshot with regard to causal consistency.

Since it is difficult to trace all causal relationships in a system, we suggest to only keep track of causal relationships in a part of the system. Tracking causal relationships within one process is a manageable task. It also allows to perform the evaluation for closed source tools and on different operating systems. The idea is to use a simple test program in which memory regions are allocated, and causally dependent changes on the regions, observed using vector clocks. If quasi-instantaneous consistency should be measured instead, realtime timestamps can be used in place of vector clocks.

5.1. Using vector clocks

Vector clocks are a concept from distributed computing that allows to track logical time by ordering events (Mattern, 1989). Usually, vector clocks are assigned to different processes in a distributed system. When events are executed by a process or messages between different processes are exchanged, the clocks need to be updated. As we want to observe changes on memory regions, all subsequent examples assign vector clocks to memory regions not processes, the original definitions by Mattern (1989) are adapted accordingly.

The idea is to assign a counter to each memory region that increases every time an event (i.e., an access by a process) is executed on the region. Such counters allow to track causal relationships between events in the following way: Additionally, to the local counter, each memory region's vector clock has fields for all other memory regions' counters. If we assume a system with n memory regions, each region has a vector clock (a vector of counters) C of size n. Each region is assigned a unique index to this vector at which its local counter is found. Whenever a process accesses a memory region it saves the region's vector clock. When it accesses the next region the two vector clocks are compared and for each index the higher value is chosen. Then the local counter is incremented.

Causal relationships can be detected with vector clocks by ordering them using the happened-before relation (Mattern, 1989): For two vector clocks, C_i and C_j, $C_i < C_j$ holds iff

$$\forall x \in \{1, \dots, n\} : C_i[x] \leq C_j[x]) \\ \wedge (\exists x : C_i[x] < C_j[x])$$

Whenever this does not hold for two vector clocks, the causing events are concurrent to each other. Fig. 10 shows an example for three memory regions and their assigned vector clocks. Each time a process accesses a memory region the vector clocks are updated. Using the definition we can, for example, see that the event caused by process p_2 on region r_1 is only causally dependent on the event caused by p_1 on the same memory region and concurrent to all other events.

With the help of the vector clocks, inconsistencies in a snapshot (or cut) can be found. First, the *global time* vector t_s of the snapshot s needs to be calculated. This vector consists of the highest value for each index in all vector clocks (Mattern, 1989) as

$$t_s = sup(C_1, ..., C_n).$$

Next, each region's vector clock is compared to the global time. More precisely, the value of the region's vector clock at its index is compared to t_s at the same index. Snapshot s is consistent iff $t_s = (C_1[1], ..., C_n[n])$ (Mattern, 1989). Fig. 11 shows an example for a consistent snapshot. Comparing the

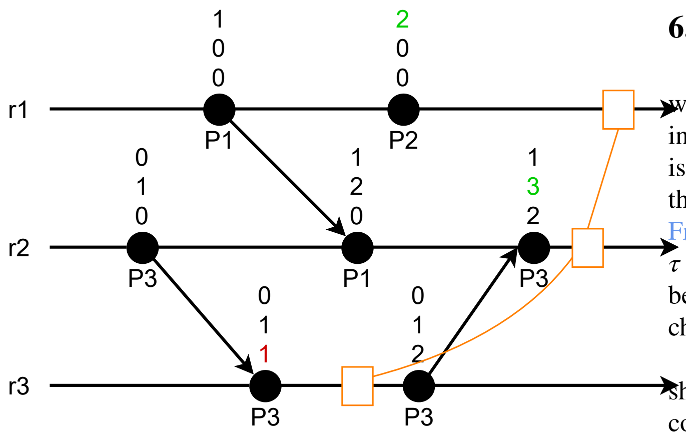

Figure 12: This snapshot is inconsistent because the event on region r_3 on which the last event included in the snapshot on region r_2 is causally dependent is not included as well. This becomes evident when comparing the vector clock of r_3 with the global time.

global time to the regions' vector clocks shows that for all memory regions the value at the respective index is equal to the global time vector at the same index. This means that for all regions the causing event of the latest access on them is included in the snapshot. Fig. 12 shows an example for an inconsistent snapshot. In this case the last event on r_3 is missing from the snapshot. This is a problem as the last event on r_2 which is included in the snapshot is causally dependent on the event. Therefore, the vector clock has not been updated yet and the inconsistency can be identified by comparing the vector clock to the global time vector.

5.2. Using realtime clocks

While they allow to capture any programmable cause-effect relationship, vector clocks are rather expensive in terms of memory. A cheap replacement of vector clocks is to simply take the measurement of a realtime clock as timestamp (if such a clock exists). With this idea, the same approach as described above can be used to track the sequence in which events occurred: Each memory region is assigned a single realtime timestamp which corresponds to the time that the most recent event happened in that memory region. The vector of all such timestamps is called the *current time*.

A snapshot algorithm now has to keep track of these most recent timestamps during the acquisition of memory regions. In analogy to the definitions for vector clocks, the global time t_s of a snapshot s is the vector of these timestamps, one for each memory region. A snapshot s is consistent if the current time is equal to the global time t_s. This is a sufficient criterion for quasi-instantaneous consistency not a necessary one. Finding a *necessary* criterion is an open question.

Obviously, this method is much more space efficient than using vector clocks, but it can only be used to check for *quasi-instantaneous* consistency and not for causal consistency.

6. Defining Integrity

We briefly revisit the concept of integrity. Integrity wishes to capture the degree to which a snapshot was influenced by the measurement method itself. To do this, it is necessary to distinguish changes on storage that are due to the snapshot mechanism and those that are not. Vömel and Freiling (2012) do this by defining a specific point in time τ which indicates the "start" of the measurement. Changes before τ are not due to the measurement mechanism and changes after τ affect integrity.

In the definition of Vömel and Freiling (2012), a snapshot satisfies *integrity with respect to* τ if the memory contents did not change between this point in time and the time of the acquisition of the region, formally:

$$\forall r \in R : \tau \leq s(r).t \implies \forall t' \in T : \tau \leq t' \leq s(r).t : s(r).v = m(r, t')$$

With this definition, whenever a memory region's content changes after τ, integrity is not satisfied anymore. We therefore call it *restrictive integrity*. However, if the original value is restored before the memory region is added to the snapshot, then the result is the same as if the change never happened. We therefore propose a slightly weaker definition of integrity, called *permissive integrity*, that allows changes in memory after τ as long as the value that is written to the snapshot is the same as the value that existed in memory at time τ.

Definition 3 (permissive integrity)**.** *A snapshot s satisfies* integrity with respect to time τ *iff*

$$\forall r \in R : \tau \leq s(r).t \implies s(r).v = m(r, \tau)$$

This definition is more permissive and enables new acquisition techniques that selectively overwrite memory regions if the snapshot contains the original data. Obviously, a snapshot the satisfies restrictive integrity with respect to τ also satisfies permissive integrity with respect to τ.

7. Relations between the Quality Criteria

In the original definitions of Vömel and Freiling (2012), the three notions of correctness, atomicity and integrity were not fully independent. In fact, integrity appeared to be unnecessarily strong and complex: A snapshot that satisfied integrity also satisfied atomicity and correctness. From a conceptual point of view, it is better to have definitions that do not imply each other to separate concerns.

Fig. 13 shows an overview of the quality criteria and their relations with respect to implication. The implications between the different consistency definitions (see section 3) and between the two integrity definitions (see section 6) are already integrated. As might be expected, instantaneous consistency and restrictive integrity are the strongest notions and do not imply each other in any way. Weaker consistency and integrity definitions are, however, not so easily separable.

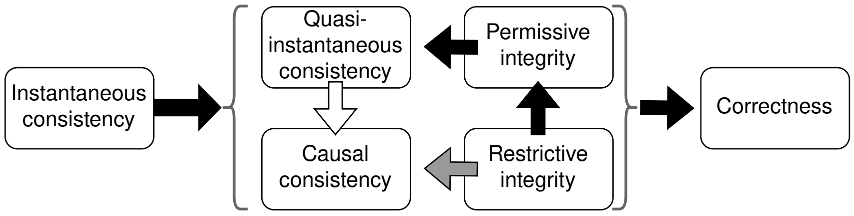

Figure 13: The relationships between the different quality criteria. Black arrows indicate implications without further assumptions. The gray arrow stands for an implication that only exists if we assume that we only observe causal relationships between modifying events. The implication shown by the white arrow assume that all events are uniquely modifying.

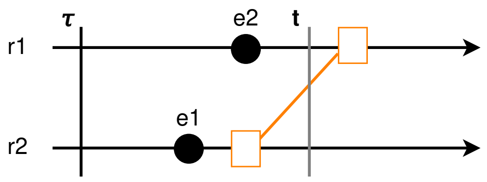

Figure 14: While this snapshot does not satisfy integrity with respect to τ it still satisfies quasi-instantaneous consistency because an instantaneous snapshot at time t would have contained the same values.

As already observed above, their relations also depend on further assumptions about the observability of events.

We first note that restrictive integrity implies causal consistency under the assumption that only modifying events are observed. This is because restrictive integrity, for every memory region, disallows *any* state changes between τ and the time the snapshot is taken. Therefore, if all events are modifying, no event can happen between τ and the snapshot, and therefore the snapshot must be causally consistent.

The relation between permissive integrity and quasi-instantaneous consistency is particularly delicate. Fig. 14 shows a quasi-instantaneous snapshot that does not satisfy permissive integrity: The snapshot does not satisfy (permissive) integrity with regard to τ but a point in time t can be found at which an instantaneous snapshot would have contained the same values. So quasi-instantaneous consistency does not imply permissive integrity. The inverse, however, is true.

Proposition 2. *Every snapshot that satisfies permissive integrity with respect to τ also satisfies quasi-instantaneous consistency.*

Proof. Let s be a snapshot that satisfies permissive integrity with respect to τ. From the definition this implies that

$$\forall r \in R : \tau \leq s(r).t \Rightarrow s(r).v = m(r, \tau).$$

Now consider the instantaneous snapshot s' taken at time τ. Since s' was taken at time τ, for all memory regions r holds that $s'(r).t = \tau$ and $s'(r).v = m(r, \tau)$. This means that s and s' were taken at different times but contain the same values, namely the values stored in memory at time τ. So overall there exists an instantaneous snapshot that has the same values as s. Therefore, s is quasi-instantaneous. □

Note that Proposition 2 holds without any further assumptions about the observability of events. Therefore, restrictive integrity also implies quasi-instantaneous consistency without any further assumptions. If we assume that we have only uniquely modifying events, both quasi-instantaneous consistency and permissive integrity imply causal consistency. Integrity and consistency therefore seem hard to disentangle completely from each other. If events do not necessarily change the values of memory regions, then permissive integrity and causal consistency are independent of each other.

Both integrity definitions imply correctness because they compare the contents of the snapshot with the contents of memory. If the acquisition method were functioning incorrectly this comparison would be likely to fail. This fact shows that correctness is not really necessary as an independent concept. Integrity and consistency suffice to determine the quality of snapshots.

8. Legal Implications

Knowing about the quality of the memory snapshots produced by different tools under certain circumstances can help investigators to choose the tool best suited for a concrete investigation. But does it also oblige them to use the best available tool?

When we try to answer this question, we have to think about the evidential value of the memory snapshot. Because the quality of the memory snapshot influences the reliability and completeness of the subsequent analysis results, their evidential value is also influenced by the memory snapshot's evidential value. The value a piece of evidence has is equal to the probability that deductions based on it will be true (Heinson, 2016). As the evidence a court grounds its decision in has to be of the highest possible quality to justify a conviction (Hannich, 2019)[§261 recital 5 ff.] investigators should strive for gathering evidence with an as high as possible evidential value. The evidential value of data is determined by the forensic process with which it was gathered. Among others its authenticity, and integrity as well as the reliability of used methods should be ensured (Fröwis, Gottschalk, Haslhofer, Rückert and Pesch, 2020). Tools that are known to produce incorrect memory snapshots must be excluded from usage in an investigation because this would also cast doubt on any analysis results and conclusions